

# Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.



## Ateliers Arduino

par X. HINAULT

[www.mon-club-elec.fr](http://www.mon-club-elec.fr)



Tous droits réservés – 2012.

**Ce document légèrement payant est soumis au droit d'auteur et est réservé à l'usage personnel.**

Afin d'encourager la production de supports didactiques de qualité, ce document est légèrement payant.

La licence d'utilisation est attribuée pour un usage personnel uniquement, dans le cercle familial. Mise en ligne et diffusion non autorisées.

Si vous n'êtes pas le détenteur de la licence attribuée pour l'usage de ce document, soyez sympa, merci d'acheter votre exemplaire personnel ici : <https://monclubelec.dpdcart.com/>

Pour tout problème lié à l'utilisation de ce document, veuillez envoyer une copie ici : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : [support@mon-club-elec.fr](mailto:support@mon-club-elec.fr)

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

Lancer également le logiciel Arduino et connecter votre carte Arduino afin de pouvoir tester au fur et à mesure les codes d'exemples !

## 1. *Intro*

L'objectif ici est d'apprendre à utiliser des moteurs (ou motoréducteurs) à courant continu avec une carte Arduino et à en contrôler le sens et la vitesse à l'aide d'une interface adaptée.

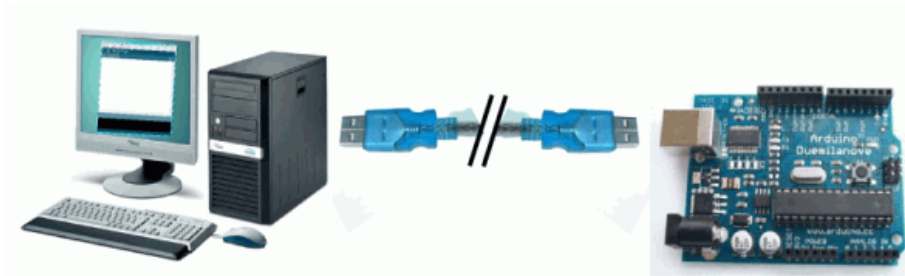


**Prêt ? C'est parti !**

## 2. Matériel nécessaire pour les ateliers Arduino

Pour cet atelier, vous aurez besoin de tout ou partie des éléments suivants pour pouvoir réaliser les exemples proposés :

### De l'espace de développement Arduino

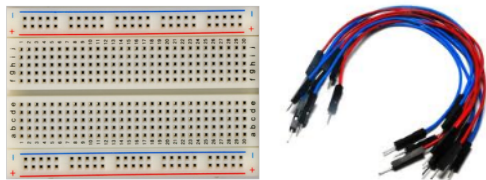


L'espace de développement Arduino associe :

- un ordinateur sous Windows, Mac Os X ou Gnu/Linux (Ubuntu)
- avec le logiciel Arduino installé (voir : <http://www.arduino.cc/>)
- un câble USB
- une carte Arduino UNO ou équivalente.

disponible chez : <http://shop.snootlab.com/> ou <http://www.gotronic.fr/>

### Du nécessaire pour réaliser des montages sans soudure

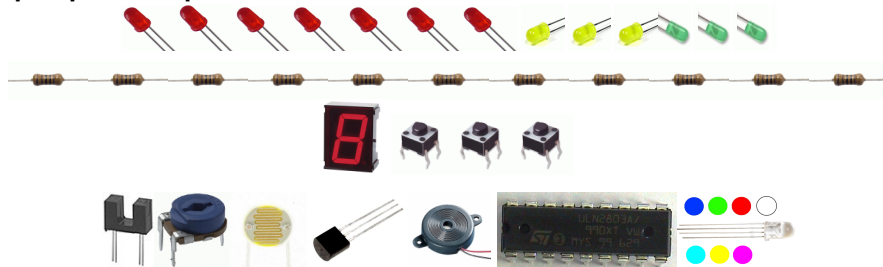


Pour réaliser des montages sans soudure, vous aurez besoin :

- d'une plaque d'essai ou breadboard moyenne (450 points)
- de quelques câbles souples (ou jumpers) mâle/mâle

disponible chez : <http://www.gotronic.fr/>

### De quelques composants de base



**Pour vous simplifier la vie, nous avons négocié ce kit pour vous !**

Vous pouvez commander ce kit complet directement en 1 clic chez notre partenaire  
<http://www.gotronic.fr/> avec le code express **701710**

**GO TRONIC**  
ROBOTIQUE ET COMPOSANTS ÉLECTRONIQUES

Pour plus de détails, voir : [http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)

Pour les ateliers Arduino niveau débutant, vous devrez idéalement disposer des composants suivants :

- des LEDs 5mm Rouges(x20), Vertes (x5) et 3 Jaunes (x5)
- digit à cathode commune rouge 13mm (x1)
- Résistances (1/4w - 5%) de 270 Ohms (x20), 4,7K Ohms (x1), 1K Ohms (x1)
- mini bouton-poussoir (x3)
- Opto-fourche (x 1)
- Résistance variable linéaire 10K (x 1)
- Photo-résistance 7mm (x 1)
- Capteur de température LM35DZ (-55/+150°C - 10mV/°C) (x 1)
- Capsule son piézoélectrique (x 1)
- ULN 2803A (CI amplificateur 8 voies, 500mA/ voie) (x 1)
- LED 5mm multicolore RVB cathode commune (x 1)

### 3. Matériel spécifique nécessaire pour cet atelier

Pour cet atelier vous aurez besoin également :

**Une carte d'interface Arduino Motor shield v3 (basée sur un L298 – « double pont en H » 2A) ou équivalente**



Cette version la plus récente du Motor Shield **livrée montée** est basée sur un CI 298 (double pont en H) :

- permet de contrôler **2 moteurs CC ou 1 moteur pas à pas bipolaire**
  - chaque moteur est contrôlé par une broche de sens et une broche de vitesse PWM
  - dispose de LEDs de fonctionnement et de diodes de protection + bouton reset
  - intensité maximale de **2A/ phase ou moteur (4A en tout)**
  - dispose d'un **capteur analogique d'intensité intégré** (+++) (1,65V/A) pour chaque phase/moteur
  - dispose de connecteurs droits pour 2 entrées analogiques, 2 sorties PWM et TWI
- (Pour mémoire, les connecteurs droits TinkerKit analog IN et OUT ont le brochage suivant : +5V / broche / 0V)
- alimentation entre 7 et 12V / 4000mA
  - borniers à vis pour Vin et les 2 sorties moteurs
  - report des broches de la carte Arduino sur connecteur droit femelle

**IDEAL pour un robot mobile utilisant 2 moteurs CC puissants !**

Dispo ici : <http://shop.snootlab.com/arduino/186-arduino-motor.html>

**Idéalement 2 moto-réducteurs ou moteurs CC 6-12V – 1 à 2A maxi**



ou

**Une alimentation externe 6-12V – 2 à 4A**



ou

Purchased by Franck Ourion, [franck.ourion@univ-lorraine.fr](mailto:franck.ourion@univ-lorraine.fr) #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.

#### 4. Remarque



Dans les pages qui suivent, je prends le temps de bien détailler l'utilisation de l'alimentation interne de la carte Arduino pour plusieurs raisons :

- pour vous apprendre à avoir le bon raisonnement technique et ne pas faire n'importe quoi...
- pour vous éviter de « griller » bêtement votre carte Arduino,
- pour n'utiliser une interface que lorsque cela est nécessaire, s'en passer lorsque cela est possible et donc éviter des dépenses inutiles,
- pour savoir et comprendre ce que vous faites lorsque vous utiliserez un circuit d'interface moteur.

Au final, avec un minimum de réflexion, vous sauverez rapidement quelques dizaines d'euros et vous vous ferez plaisir à moindre frais !

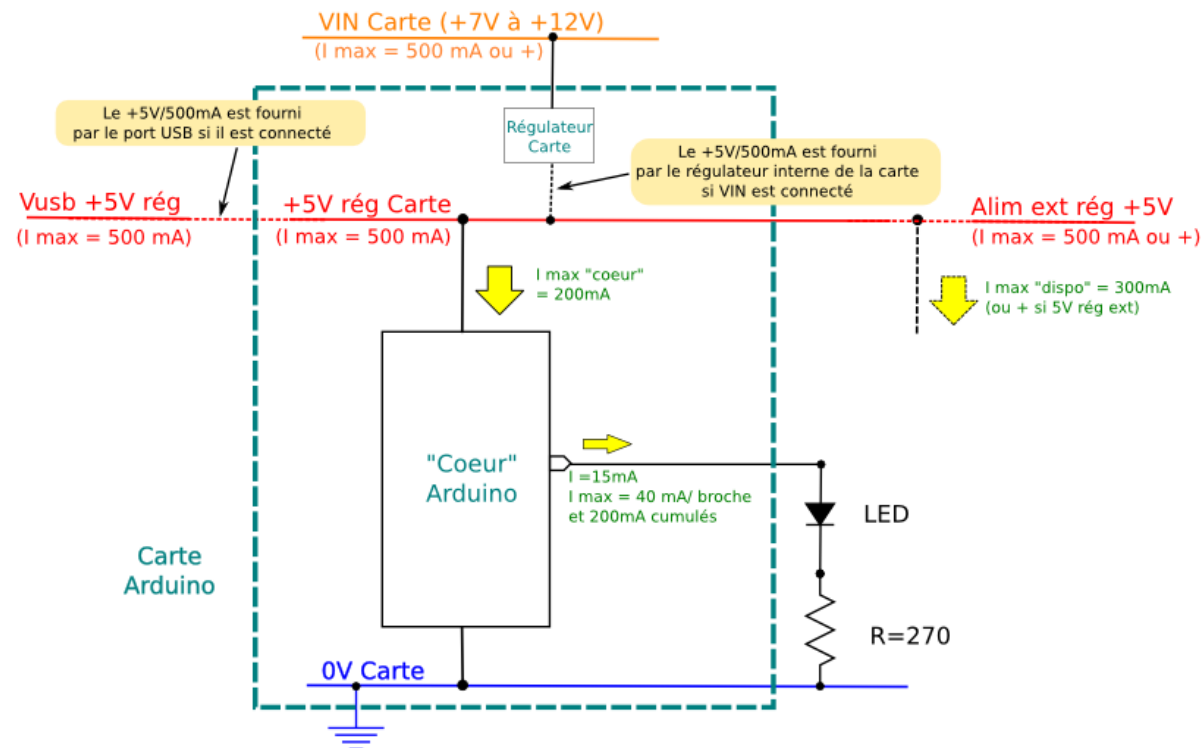
## 5. Technique : l'alimentation de la carte Arduino

Lorsque l'on utilise un moteur, il est essentiel d'avoir toujours à l'esprit les notions d'intensité et de tension de la carte Arduino. Comme on l'a déjà vu, la carte Arduino intègre une alimentation interne régulée de 5V / 500mA :

- soit en provenance du port USB
- soit à partir de l'alimentation Vin 7-12V / 500mA (ou +)

Il est essentiel de distinguer :

- **l'intensité maximale (200mA) que peut fournir le « coeur »** de la carte Arduino et limité à **40mA par broche** en sortie mais **200mA maximum** pour l'ensemble des broches réunies. Une LED en série avec une résistance utilisera par exemple 15mA fournis par le « coeur » Arduino.
- **l'intensité maximale (500mA) que peut fournir l'alimentation +5V régulé/500mA**. Cette alimentation de +5V/500mA de la carte Arduino peut également être mise en parallèle avec une alimentation externe +5V régulée au besoin.
- **On dispose donc de 300mA supplémentaires maxi pour alimenter des dispositifs 5V directement à partir de l'alimentation de la carte Arduino (mais pas à partir des broches !!)**



## 6. Les moteurs et moto-réducteurs à courant continu ou CC : concrètement

**Les moteurs CC seuls** : pas chers (dès 3-4€), tailles variées, mais peu utilisables « tel que » sauf pour les rotation rapides simples (plusieurs milliers tr/min - hélices)



**Les moto-réducteurs CC (= moteurs + engrenages)** sont beaucoup plus utiles : vitesse de rotation réduite (centaines tr/min) et force de rotation importante.



**Une difficulté souvent rencontrée lors la mise en oeuvre d'un moteur : réaliser une fixation simple...**

La gamme de moteur MFA, distribuée notamment par GoTronic, propose plusieurs modèles de moteurs et moto-réducteurs CC **avec support intégré.**



## 7. Les moteurs et moto-réducteurs CC : fiche technique

### Description

- Un moteur ou un moto-réducteur à courant continu tourne dans un sens lorsqu'il est mis sous tension. Le sens de rotation est inversé, si on inverse la polarité de l'alimentation.

### Type de mouvement

- Rotation continue maximale très rapide et à couple moyen à faible pour les moteurs CC simples (plusieurs milliers de tours par minute)
- Rotation continue maximale moins rapide (dizaines ou centaines de tours par minute) avec un couple élevé pour les moto-réducteurs.

### Brochage

- Un moteur ou un moto-réducteur à rotation continue dispose de 2 broches d'alimentation le 0V et le +V

### Principe de contrôle du servomoteur

- Le sens de rotation d'un moteur ou moto-réducteur CC est contrôlé par la polarité d'alimentation : le moteur tourne dans un sens à la mise sous tension et tourne dans l'autre sens si on inverse la polarité.
- La vitesse de rotation d'un moteur CC (supporte une plage d'alimentation +/- large selon les modèles) peut être contrôlée par :
  - la tension d'alimentation utilisée : plus elle est élevée et plus la vitesse est rapide,
  - la largeur de l'impulsion PWM utilisée pour l'alimenter.

### Caractéristiques mécaniques

- Un moteur ou un moto-réducteur CC se caractérise par :
  - la plage d'alimentation supportée : par exemple de 6 à 12V
  - l'intensité à vide et bloquée (plusieurs centaines de mA voire plusieurs A)
  - son couple en g.cm ou Kg.cm, bloqué et à vide
  - sa vitesse de rotation en tours/minute, bloqué et à vide

### Caractéristiques électriques

- La plage de tension d'alimentation est variable selon les moteurs, entre 3V à 30V selon les modèles. Avec Arduino, prendre un modèle 6-12V.
- L'intensité de fonctionnement est de l'ordre de quelques centaines de mA à quelques A selon les modèles : une interface est indispensable !

### Maintien de position « hors tension »

- Un moteur CC ne tient pas bien une position hors tension
- **Un moto-réducteur tient une position fixe hors tension**

### Codage

- Facile à mettre en oeuvre, sous réserve de l'utilisation d'une interface, par la gestion simple de 2 broches numériques en sortie.

### Interface de « puissance »

- **INTERFACE « de puissance » OBLIGATOIRE** pour une utilisation avec une carte Arduino en raison de l'intensité de fonctionnement.
- Grosso-modo, il existe **2 types** d'interfaces (2 broches /moteur ) :
  - soit chaque broche contrôle un sens et la vitesse par PWM
  - soit une broche contrôle le sens et l'autre la vitesse par PWM

### Principe d'alimentation

- **Obligation également d'utiliser également une alimentation externe adaptée** (sauf éventuellement pour les petits moteurs...), capable de fournir une intensité suffisante.

### Prix unitaire

- Des modèles à moins de 10€ existent. Mais oompter 20-25€ pour un seul bon moto-réducteur.

### Coût global unitaire de mise en oeuvre

- moteur (10-20€) + 1/2 interface (20€/2) = **20 à 30€/moteur actif !**

### Utilisation type

- Les moteurs CC sont très utiles pour réaliser des **robot mobiles roulant** en raison de leur force de rotation.

### Avantages

- **Moto-réducteur : Force importante et position arrêt fixe**
- Facile à contrôler (via interface) par 2 broches E/S par moteur.

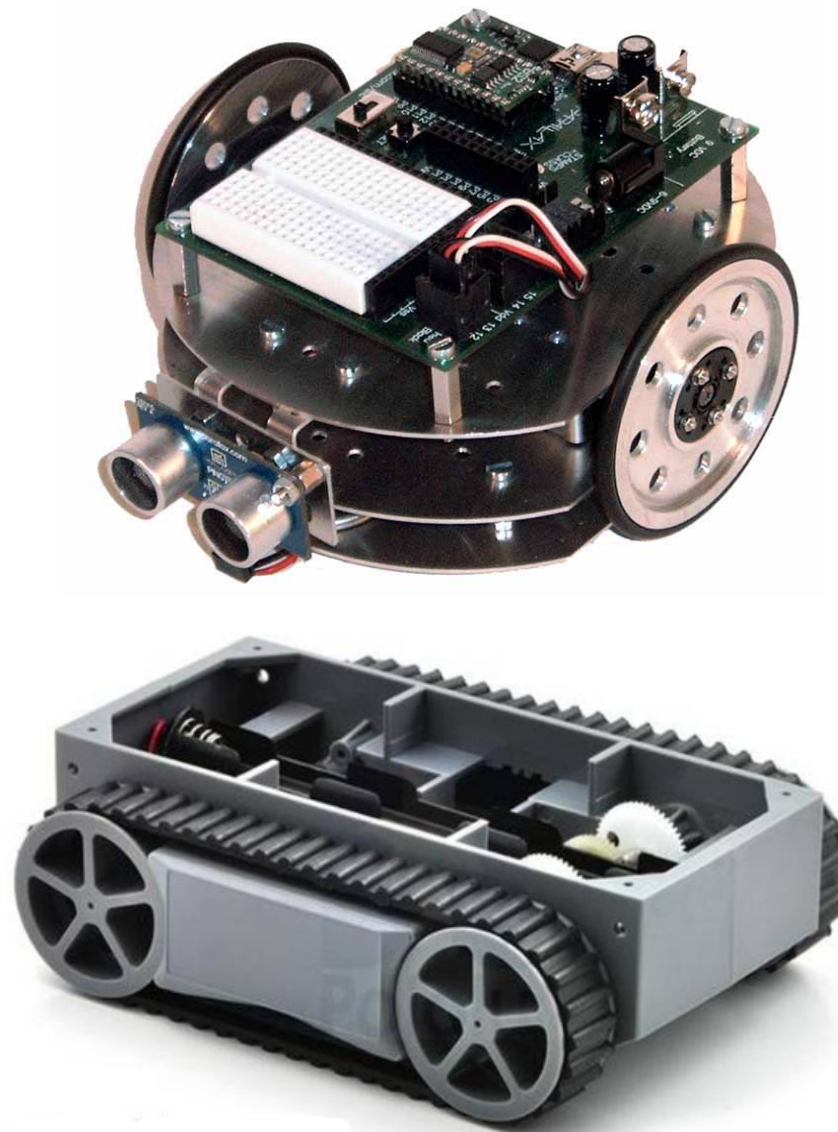
### Inconvénients / limites

- le bruit : les moteurs CC sont bruyants et les moto-réducteurs moins si ils sont de qualité.
- le prix et le poids.
- alimentation fort ampérage parfois nécessaire (et donc poids alim...)



## 8. Les moteurs et moto-réducteurs CC : exemples d'utilisation

L'utilisation « type » des moteurs CC est la motorisation d'un robot mobile roulant



Purchased by Franck Ourion, [franck.ourion@univ-lorraine.fr](mailto:franck.ourion@univ-lorraine.fr) #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.

## 9. Rappel : Caractéristiques électriques d'une broche Numérique Arduino en sortie

Une broche numérique Arduino individuelle en sortie peut-être considérée **comme une « mini »-alimentation** :

- fournissant une tension de **5V régulé** au niveau HAUT et 0V au niveau BAS.
- pouvant fournir **au maximum 40mA** pour une seule broche en sortie.

Pour l'ensemble des broches numériques :

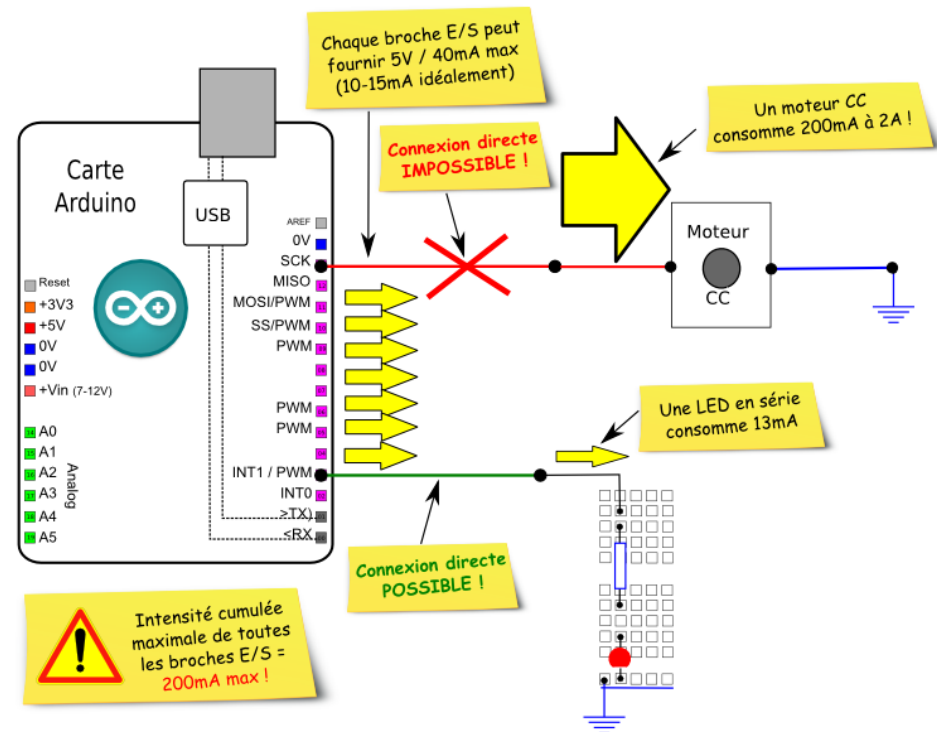
- **l'intensité maximale cumulée ne doit pas dépasser les 200mA.**
- Ainsi, pour éviter les soucis, dans l'hypothèse où l'on utilisera au maximum une 15aine de broches en sortie simultanément, considérer que l'on peut utiliser sans danger  $200\text{mA}/15 = 13\text{mA}$  / broche soit en pratique 10 à 15mA par broche.

**En pratique : considérer qu'une broche Arduino fournit 5V / 10-15mA**

Voici quelques exemples, pour se faire une idée de ce que l'on peut connecter directement sur une broche numérique d'une carte Arduino en sortie :

- une broche d'un autre CI numérique consommera dans les 1 mA, => **connexion directe POSSIBLE !**
- une LED en série avec sa résistance consommera dans les 10-20mA selon la résistance utilisée, => **connexion directe POSSIBLE !**
- la broche de commande d'un servomoteur consommera dans les 5mA, => **connexion directe POSSIBLE !**
- un moteur CC consommera dans les 250mA => **connexion directe IMPOSSIBLE !** (Dans ce cas, on devra utiliser une interface de puissance comme nous le verrons)

**En pratique : TOUJOURS se demander « quelle intensité va être utilisée ? »**



## 10. Notion d'amplification de puissance et d'interface de puissance

A présent, vous allez comprendre facilement le concept d'amplification de puissance. C'est amusant d'allumer des LEDs, mais on va rapidement avoir envie de commander avec une carte Arduino plein d'autres choses et notamment des dispositifs de la vie réelle...

Si on utilise la carte Arduino telle quelle, les possibilités sont assez limitées : on ne pourra allumer/contrôler que des dispositifs :

- fonctionnant en 5V (1V = 1 Volt = 1000 millivolts)
- ne nécessitant pas plus de 40 mA (1 mA = 1 milliampère = 0,001 Ampère)

Or de nombreux dispositifs ont caractéristiques bien différentes :

- un moteur à courant continu nécessitera 6 à 12V et 200mA à 2A
- un relais fonctionnera en 12V et nécessitera 100mA par exemple
- un ruban à LED fonctionnera en 12V et jusqu'à plusieurs A
- etc...

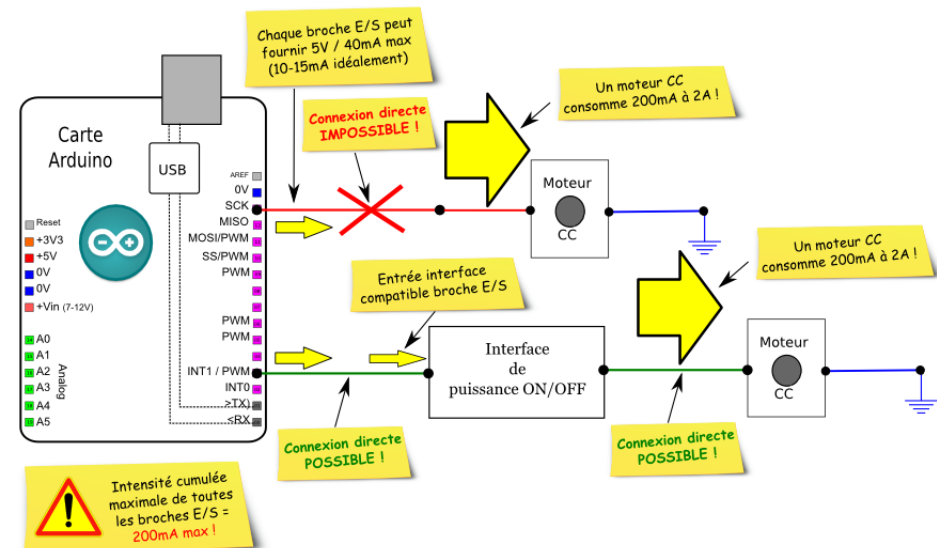
La solution passe par une « [interface de puissance](#) » : c'est un circuit électronique ou une carte électronique selon les cas, qui va assurer un double rôle :

- adapter la tension
- adapter l'intensité

De cette façon, à partir de la sortie numérique 5V/10mA, on va pouvoir contrôler des appareils :

- entre 5V et 12V voire 30V (adaptation en tension)
- consommant 100mA, 300mA voire plusieurs A (adaptation en intensité)

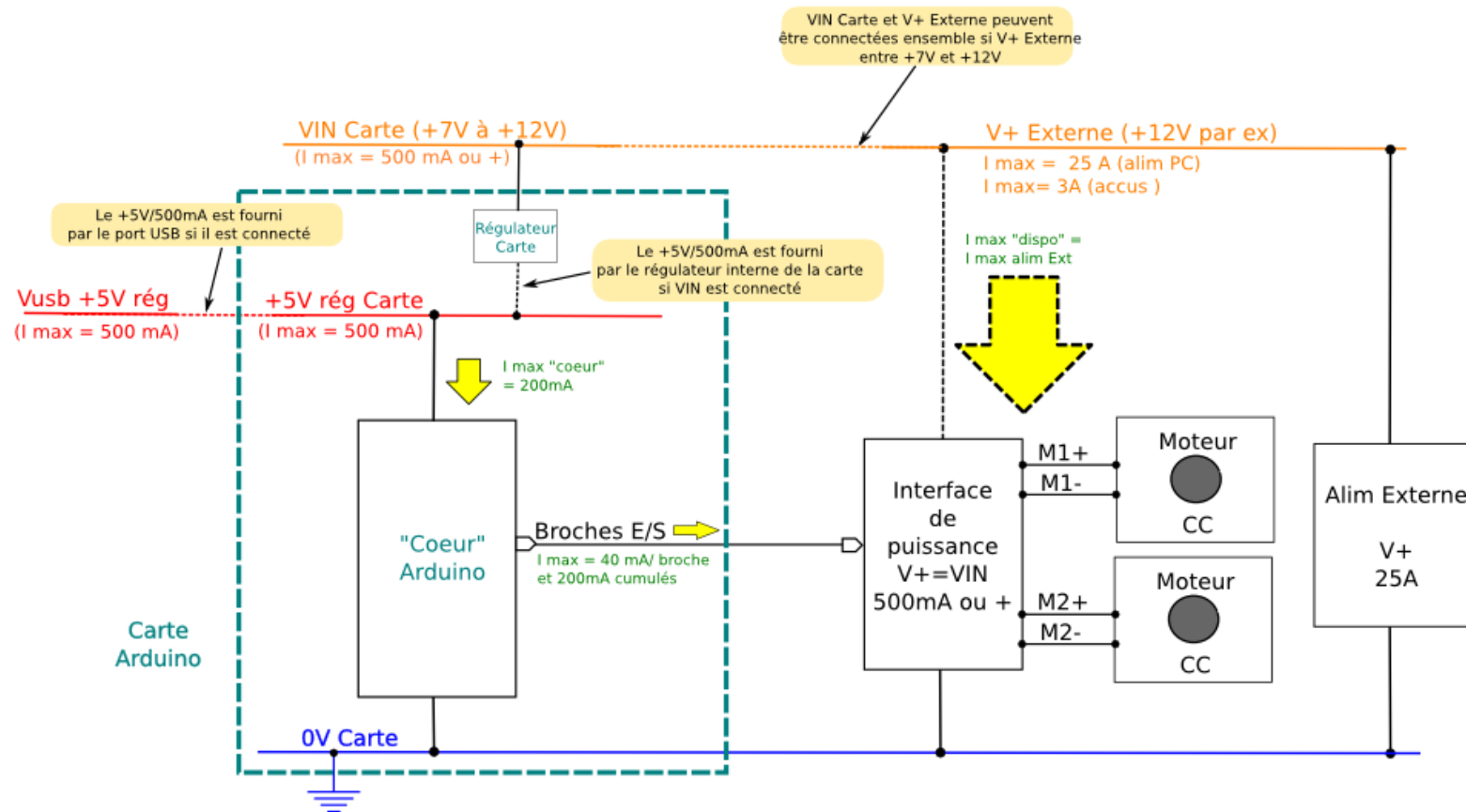
On pourra de cette façon potentiellement contrôler toutes sortes de dispositifs à partir d'une simple carte Arduino !



## 11. Les moteurs et moto-réducteurs CC : schéma électrique type d'utilisation d'une interface avec Arduino

Pour utiliser un moteur ou un moto-réducteur CC avec une carte Arduino, l'utilisation d'une carte d'interface est OBLIGATOIRE !

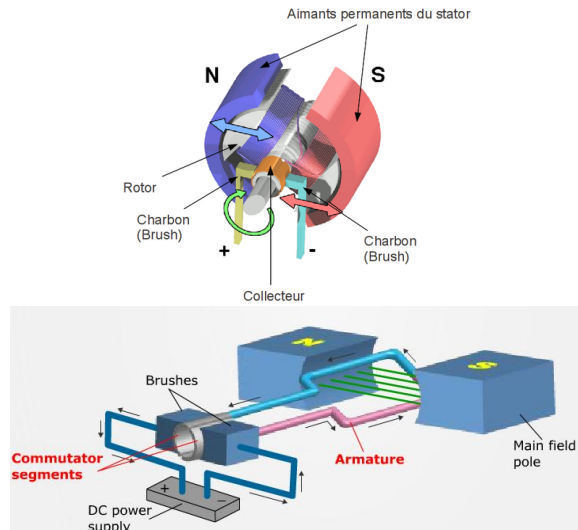
Bien noter le **triple rôle** de l'interface : adaptation en tension et en intensité ainsi que le contrôle du sens.



## 12. Les moteurs et moto-réducteurs CC : le principe de contrôle

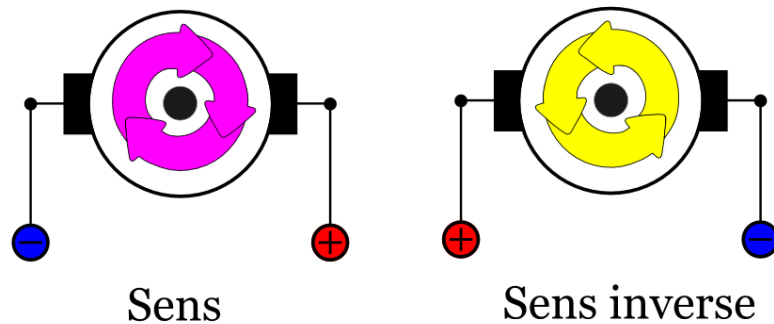
### Principe général fonctionnement d'un moteur CC

La mise sous tension du moteur entraîne la rotation de la bobine qui se trouve entre 2 aimants : le sens de rotation dépend la polarité de la bobine.



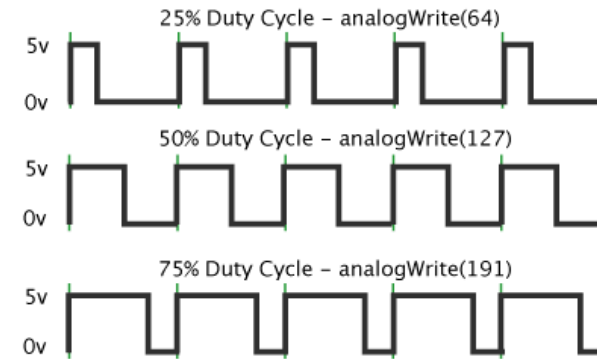
### Principe de contrôle du sens de rotation

- Le principe de contrôle du sens de rotation est facile à mettre en évidence manuellement (facile à tester par vous même) :
  - si on connecte une broche du moteur au + et l'autre au - d'une alimentation, le moteur se met à tourner dans un sens.
  - si on inverse le + et le -, le moteur va se mettre à tourner dans l'autre sens.



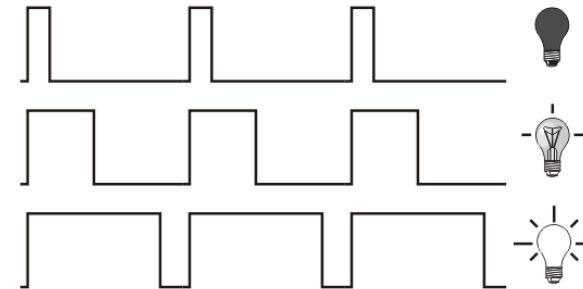
### Rappel du concept de Modulation Largeur Impulsion

- Pour simuler un comportement « analogique » avec une broche numérique, on joue sur la largeur de l'impulsion mise sur la broche :



### Principe du contrôle de la rotation

- Le principe du contrôle de la vitesse de rotation est le même que le principe du contrôle de la luminosité d'une LED par impulsion PWM :



- La différence fondamentale réside dans le fait :
  - qu'au lieu de la luminosité, c'est la vitesse du moteur qui va varier
  - qu'il est impossible de connecter directement le moteur sur une broche Arduino et qu'il faudra utiliser une interface.

#### A retenir :

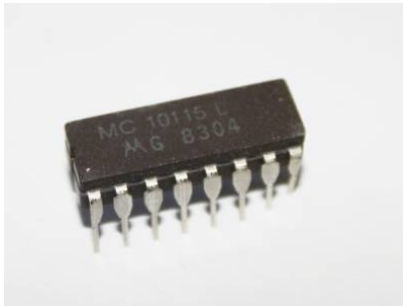
le **sens de rotation** d'un moteur CC est contrôlé par la **polarité** de l'alimentation

la **vitesse de rotation** d'un moteur CC est contrôlée par la **largeur de l'impulsion** appliquée au moteur.

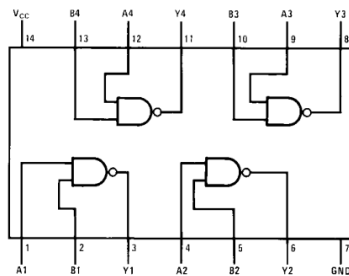
### 13. Info technique : les circuits intégrés en boîtier DIL

Petit préalable : présentation des circuits intégrés DIL... En fait, il s'agit d'un format de boîtier de circuit intégré courant :

- petit boîtier noir en plastique rigide
- présentant toute une série de broches métalliques latérales
- avec un numéro dessus correspondant à la référence du circuit



Un circuit intégré est un circuit électronique miniaturisé et ayant une fonction particulière fixe : un compteur, une bascule JK, portes logiques, etc...



Exemple de schéma fonctionnel – 74LS00 – 4 portes NAND

Un circuit intégré dispose de plusieurs broches :

- Chaque broche du circuit intégré DIL va avoir une fonction précise qui est décrite dans une fiche technique appelée « datasheet » en anglais.
- Dans le cas d'un circuit intégré numérique, les broches métalliques seront des entrées ou sorties numériques.
- Un circuit intégré type dispose également, comme tout dispositif électrique de 2 broches d'alimentation : le 0V et le +5V classiquement.

Les circuits intégrés numériques existent en plusieurs familles :

- circuits analogiques, numériques, mixtes,
- chaque famille est dénommée avec des lettres et des nombres
- l'une d'entre elle est courante : les 74LSxx où xx est un nombre.
- il en existent pleins d'autres, les lettres utilisées étant souvent associées à un fabricant et le nombre à une fonction

74LS00	4 portes NAND à 2 entrées	DIL 14
74LS02	4 portes NOR à 2 entrées	DIL 14
74LS03	4 NAND à 2 entrées col. ouv.	DIL 14
74LS04	Sextuple inverseur	DIL 14
SN7406	Sextuple inverseur buffer	DIL 14
74LS06	Sextuple inverseur buffer	DIL 14
74LS07	Sextuple buffer	DIL 14
74LS08	4 portes AND à 2 entrées	DIL 14
74LS09	4 AND à 2 entrées col. ouv.	DIL 14
74LS09D-CMS	4 AND à 2 entrées col. ouv.	SO14
74LS12	3 NAND à 3 entrées col. ouv.	DIL 14
74LS14	Sextuple inverseur trigger	DIL 14
74LS15	3 AND à 3 entrées col. ouv.	DIL 14

Exemple de nom de circuit et leur fonction



Position type des broches 0V et +5V d'un CI logique.

Noter le sens du circuit indiqué par une marque sur le boîtier.

Avec Arduino, on aura très peu besoin de circuits intégrés externes, la plupart des fonctions pouvant être réalisées par programmation. Un programme élaboré peut être l'équivalent de plusieurs dizaines de circuits intégrés associés !



## 14. Les moteurs et moto-réducteurs CC : les circuits d'interface ON/OFF (contrôle de la **vitesse** seule dans 1 sens) : Exemple du CI ULN 2803

### Fonction

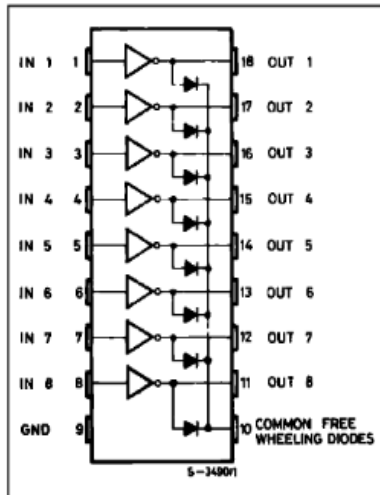
- Ce circuit très répandu et bon marché (<2 Euros) est au format DIL 18 est constitué de 8 étages d'amplification inverseur
- chaque étage :
  - peut-être commandé par une entrée numérique
  - et peut fournir 500mA en sortie, sous une tension pouvant aller jusqu'à 50V.



### Brochage

Le brochage de ce CI est très simple et très pratique :

- 2 broches d'alimentation : 0V (broche 9) et V+ (broche 10).
- 8 broches d'entrées de commande numériques (1 à 8) face chacune à une broche de sortie de puissance (10 à 18)



### Principe de fonctionnement

Le fonctionnement est très simple. Pour chaque étage, on a :

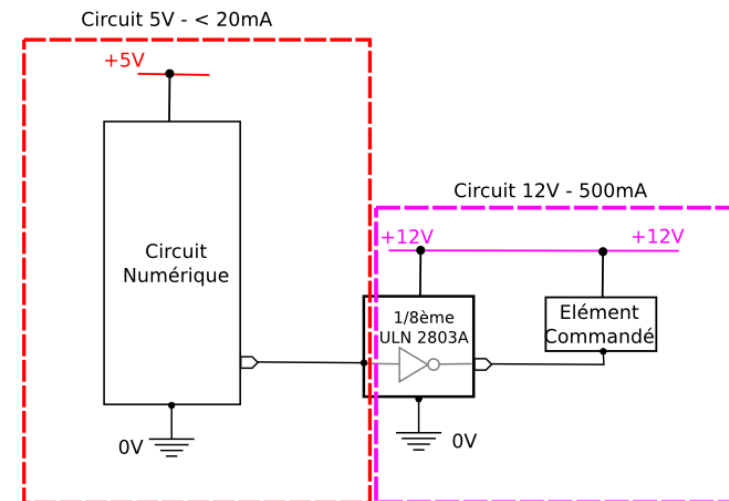
entrée numérique	Tension Sortie	Effet
<b>HAUT</b>	0V	<b>MARCHE</b>
<b>BAS</b>	V(+5V à +50V !)	<b>ARRET</b>

Ainsi, la sortie de puissance est le reflet inversé de l'entrée de commande et l'élément électrique commandé est en MARCHE sur le niveau HAUT.

### Circuit de fonctionnement type

Les éléments électriques à commander doivent être connectés entre le V+ et une broche de sortie :

- c'est le niveau HAUT, et uniquement le niveau HAUT, qui peut commander la mise sous tension de l'élément commandé.
- Ainsi, il n'est pas possible de commander l'élément électrique par un niveau BAS, en connectant l'élément entre la sortie et le 0V. **Ceci limite le fonctionnement de l'ULN2803 à un fonctionnement ON/OFF simple.**

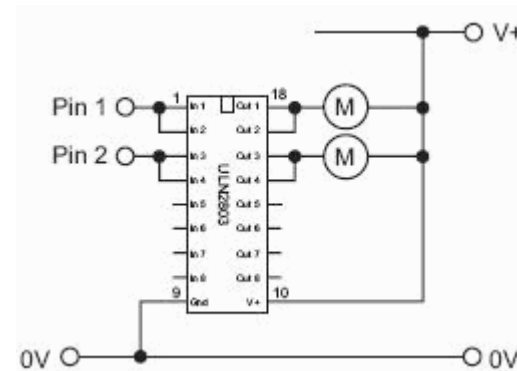


### Plusieurs avantages :

- amplification ON/OFF sur 8 voies !
- entrée numérique en face de la sortie amplifiée !
- petite taille (simple boîtier DIL)
- pas cher et répandu
- adaptation en tension de 6V à 30V
- adaptation en sortie jusqu'à 500mA/voie
- possibilité de coupler les voies 2 à 2
- rapide = laisse passer impulsions

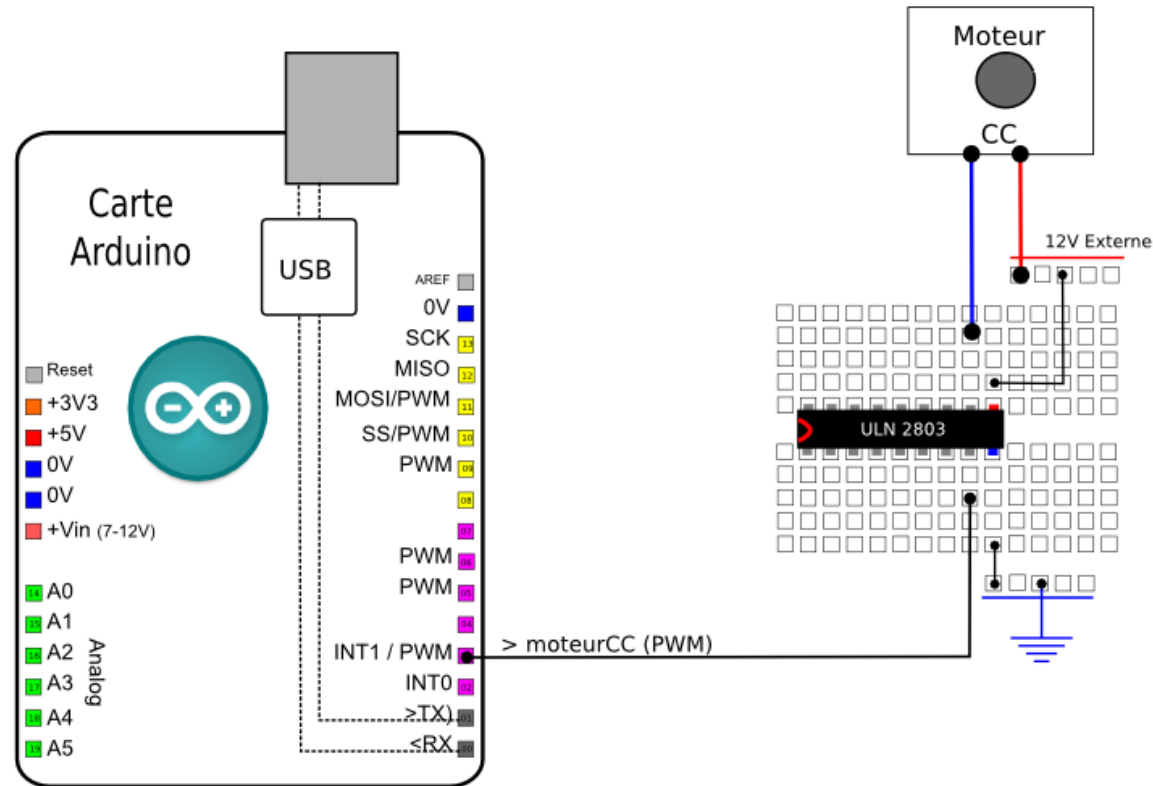
### Truc d'utilisation

Si l'on souhaite disposer d'une intensité de plus de 500mA, on peut coupler plusieurs entrées et les sorties correspondantes :



**Bien comprendre que ce type de circuit « ON/OFF » ne permet pas d'inverser la polarité aux bornes du moteur :**  
il ne pourra donc pas contrôler le sens de rotation mais seulement la vitesse. Le moteur tournera toujours dans le même sens.

**15. Les moteurs et moto-réducteurs CC : les circuits d'interface ON/OFF (contrôle de la vitesse seule dans 1 sens) : le montage type d'un ULN 2803 avec une carte Arduino**



Utiliser ici un moteur CC 6-12V / 500mA maxi.

Noter que ce montage ne permet pas de contrôler le sens de rotation du moteur, mais uniquement sa vitesse par impulsion PWM.

Pour contrôler également le sens de rotation, il faudra utiliser une interface « moteur » capable à la fois de contrôler le sens ET la vitesse du moteur.

**Truc :** pour réaliser ce montage facilement, je vous conseille tout simplement d'utiliser un bloc secteur entre 6-12V (à adapter à votre moteur – 500mA maxi !) pour alimenter la carte Arduino (Vin) et vous câblerez l'ULN 2803 en alimentant votre plaque d'essai avec Vin au lieu de +5V.

## 16. Contrôle simple de la vitesse (seule) d'un moteur (ou moto-réducteur CC) : le programme

### Ce qu'on va faire ici...

- Le programme que je vous propose est très simple : on va simplement faire varier la vitesse du moteur de 0 (=arrêt) à 100% de la vitesse.
- Ce programme ressemble point pour point à celui que nous avons utilisé pour faire varier la luminosité d'une LED. La différence fondamentale est ici « technique » (utilisation d'un ULN 2803) mais au niveau programmation, c'est exactement la même chose ou presque.

### Entête déclarative

- On déclare :
  - une constante de broche de contrôle du moteur. Cette broche doit être de type PWM (la 3,5,6,9,10 ou 11 – signalée par ~ devant)
  - une variable **int** fixant la vitesse
  - une variable **int** appelée largeur et fixant la largeur de l'impulsion

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

const int MOTEUR=3; // constante désignant la broche du moteur
int vitesse=50; // variable fixant la durée de la pause en ms

int largeur=0; // variable pour la largeur d'impulsion
```

### Fonction **setup()**

- On commence par configurer en sortie la broche de contrôle du moteur avec l'instruction **pinMode**(broche, sens),
- puis on réalise un test initial du moteur :
  - on met la broche du moteur à 1 pendant 1 seconde, le moteur tourne alors à fond.
  - puis on arrête le moteur pendant 1 seconde.

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    pinMode(MOTEUR, OUTPUT); // met la broche en sortie

    //-- test du moteur --
    digitalWrite(MOTEUR,HIGH); // moteur à fond
    delay(1000); // 1 seconde
    digitalWrite(MOTEUR,LOW); // moteur à l'arrêt
    delay(1000);
} // fin de la fonction setup()
```

## Fonction `loop()`

- A l'aide d'une première boucle `for()`, on augmente progressivement la vitesse de 0 à 100% en élargissant progressivement l'impulsion PWM générée avec l'instruction `analogWrite(broche,largeur)`
- De la même façon, on utilise une seconde boucle `for()` pour réduire la vitesse de 100% à 0%.
- Noter l'utilisation de la fonction `map()` qui permet de transposer la valeur en % (échelle 0-100) en la valeur correspondante sur l'échelle PWM 0-255.
- Des instructions `delay()` fixent la pause entre chaque changement de vitesse.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    for (int i=0; i<=100; i++) { // boucle croissante

        largeur= map(i,0,100, 0,255); // ré-échelonne i (0-100%) vers (0-255)
        analogWrite(MOTEUR,largeur); // génère impulsion PWM voulue sur la broche
        delay(vitesse); // pause

    } // fin for i

    for (int i=100; i>=0; i--) { // boucle décroissante

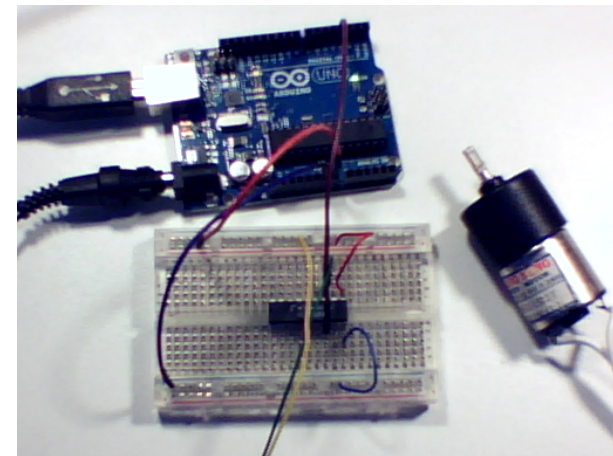
        largeur= map(i,0,100, 0,255); // ré-échelonne i (0-100%) vers (0-255)
        analogWrite(MOTEUR,largeur); // génère impulsion PWM voulue sur la broche
        delay(vitesse); // pause

    } // fin for i

} // fin de la fonction loop()
```

## Fonctionnement du programme

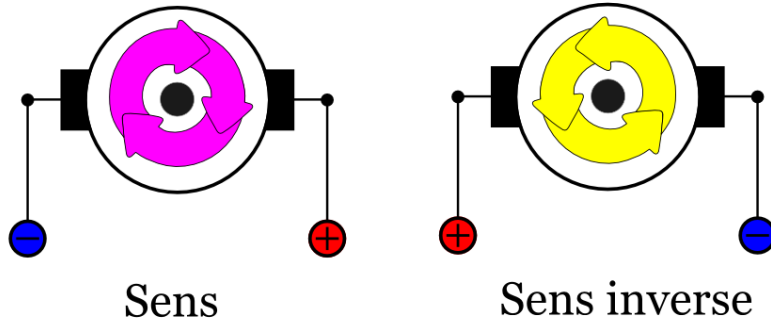
- Une fois la carte Arduino programmée :
  - le moteur tourne à fond 1 seconde puis s'arrête,
  - puis le moteur se remet à tourner progressivement de plus en plus vite de 0% à 100%
  - puis la vitesse diminue à nouveau de 100% jusqu'à l'arrêt (0%)
  - et ainsi de suite...



## 17. Les moteurs et moto-réducteurs CC : principe de contrôle du sens par une interface

### Principe de contrôle du sens de rotation

- Le principe de contrôle du sens de rotation est facile à mettre en évidence manuellement (facile à tester par vous même) :
  - si on connecte une broche du moteur au + et l'autre au - d'une alimentation, le moteur se met à tourner dans un sens.
  - si on inverse le + et le -, le moteur va se mettre à tourner dans l'autre sens.



### Le problème...

- A la main, vous voyez bien le principe... vous connectez les 2 fils + et - de votre alimentation au moteur qui tourne dans un sens... puis vous faites l'inverse et ça tourne dans l'autre sens... Mais vous n'allez pas passer votre temps à tenir les 2 fils pour inverser la polarité... et votre carte Arduino n'a pas de mains !!



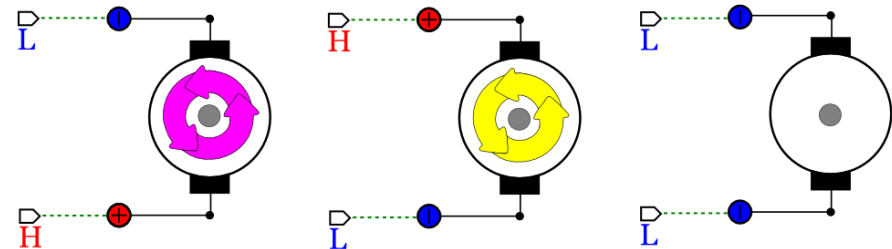
- Blague à part, on est bien embêté : comment réussir uniquement à partir des broches numériques de la carte Arduino (et sans les mains bien sûr...) à inverser la polarité aux bornes de notre moteur CC ???

### Imaginons un peu...

- Ce qui serait bien, c'est un circuit qui serait capable de mettre en correspondance le niveau HAUT/BAS présent sur une broche numérique de l'Arduino avec la polarité + / - de l'alimentation...
- Ainsi :
  - un niveau HAUT mettrait le V+ sur la borne correspondante du moteur...
  - un niveau BAS mettrait le V- sur la borne correspondante du moteur...



- De cette façon, on pourrait du coup contrôler le moteur facilement avec 2 broches numériques :
  - si la première broche est HAUT et l'autre BAS, le moteur tournerait dans un sens
  - à l'inverse la première broche est BAS et l'autre HAUT, le moteur tournerait dans l'autre sens par inversion de la polarité
  - si les 2 broches étaient soit HAUT soit BAS, le moteur serait arrêté.



- En un mot, un circuit qui ferait avec 2 broches numériques ce que l'on fait avec nos 2 mains... lorsque l'on inverse le sens de rotation d'un moteur CC manuellement.

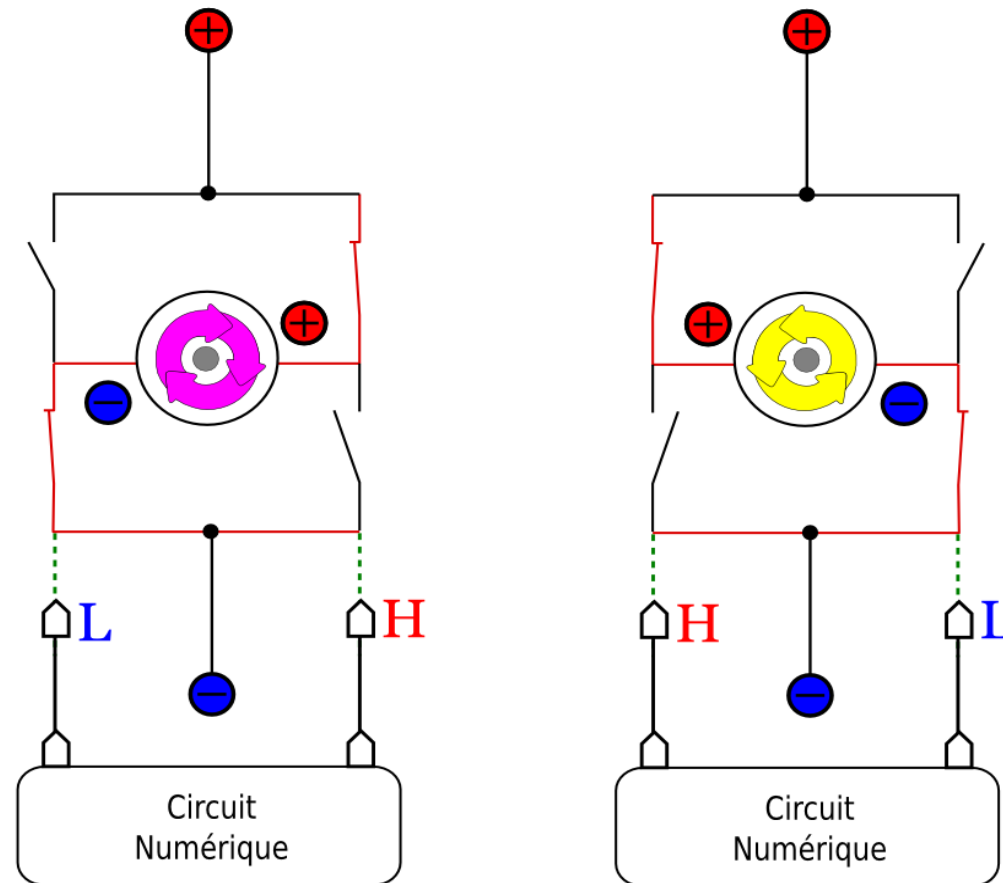
### La solution ?

- Une fois de plus la solution existe : c'est le « pont en H » !



## 18. Les moteurs et moto-réducteurs CC : contrôle du **sens** : le concept de « pont en H » (ou H-bridge)

- Le principe interne d'un circuit « H-Bridge » ou « pont en H » est le suivant : **à chaque broche numérique de commande sont associés électroniquement l'équivalent de 2 « interrupteurs » contrôlant la connexion d'une broche du moteur au V+ et au V-.** Un pont en H comporte donc 4 « interrupteurs » internes en tout (en réalité ce sont des transistors de puissance un peu particuliers).
- Ainsi :
  - lorsqu'une broche de commande est au niveau **HAUT**, la broche correspondante du moteur est au niveau **V+**
  - lorsqu'une broche de commande est au niveau **BAS**, la broche correspondante du moteur est au niveau **V-**
  - les 2 broches numériques permettent ainsi de contrôler les 2 sens de rotation : tout se passe au final comme si chaque broche au niveau HAUT contrôlait un sens.**



## 19. Les moteurs et moto-réducteurs CC : les circuits d'interface « pont en H » en pratique

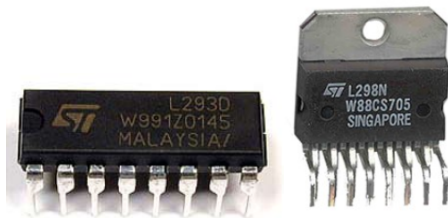
### Le « pont en H » en pratique...

- Encore une fois, vous devez vous dire qu'un circuit comme ça, ça doit coûter un max... Et bien pas tant que ça... Il existe des circuits intégrés double « Pont en H » pour quelques euros : notamment le L293D ou encore le L298.
- Ces circuits seuls ne sont pas très faciles à utiliser seuls (ils nécessitent des composants externes) et ils sont donc intégrés dans des cartes d'extension (ou shields) utilisables avec Arduino pour un coût de l'ordre de 15-20€.

( CI simple, shields, etc...)

### Deux exemples de CI double « pont en H »

- La plupart des interfaces de contrôle de moteurs utilisées avec Arduino sont basées autour de 2 circuits double « Pont en H » très utilisés.
- Le L293 est un CI double « pont en H » de **puissance modérée** :
  - en boîtier DIL, de l'ordre de 3€
  - permettant de contrôler 2 moteurs CC ou 1 moteur pas à pas bipolaire (on verra ça plus tard)
  - capable de fournir **600mA** par moteur (1,2A en pic)
  - sous une tension de 5 à 36V
- Le L298 est un CI double « pont en H » de **puissance élevée** :
  - en boîtier MultiWatt ou SOC, de l'ordre de 6€
  - permettant de contrôler 2 moteurs CC ou 1 moteur pas à pas bipolaire
  - capable de fournir **2000mA** par moteur (3A en pic)
  - sous une tension de 7 à 46V



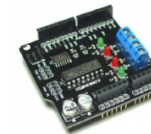
- Il en existe pleins d'autres qui fonctionnent sur le même principe, mais ce sont les 2 plus importants à connaître en pratique.

### Les cartes et shields d'interface pour moteurs CC

- En pratique, il est plus simple d'utiliser des petites cartes électroniques toutes prêtes qui vont intégrer tous les composants externes nécessaires ainsi que les borniers à vis.
- On peut utiliser avec Arduino :
  - des shields, enfichables broche à broche sur la carte Arduino
  - des cartes électroniques autonomes

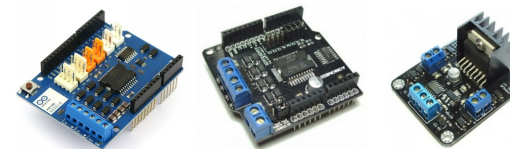
### Exemples de cartes d'interfaces à base de L293 x1

- Ces cartes permettent de contrôler **2** moteurs CC ou 1 moteur pas à pas bipolaire.



### Exemples de cartes d'interface à base de L298 x1

- Ces cartes permettent de contrôler **2** moteurs CC ou 1 moteur pas à pas bipolaire.



### Exemples de cartes d'interfaces à base de L293 x2

- Ces cartes permettent de contrôler **4** moteurs CC ou 1 moteur pas à pas bipolaire.



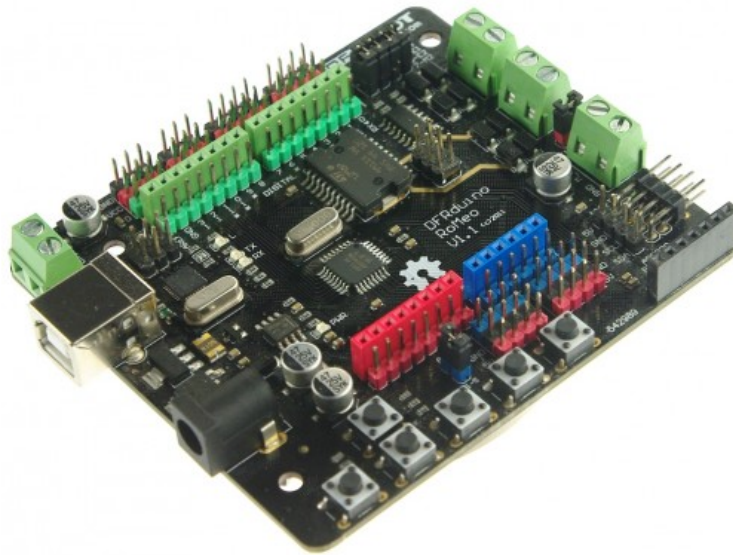
### Principe d'utilisation avec Arduino

- Selon les modèles, ces interfaces disposeront : soit d'une **broche de direction + une broche de vitesse** (PWM) par moteur, soit de **deux broches contrôlant chacune un sens et la vitesse (PWM) dans ce sens** par moteur

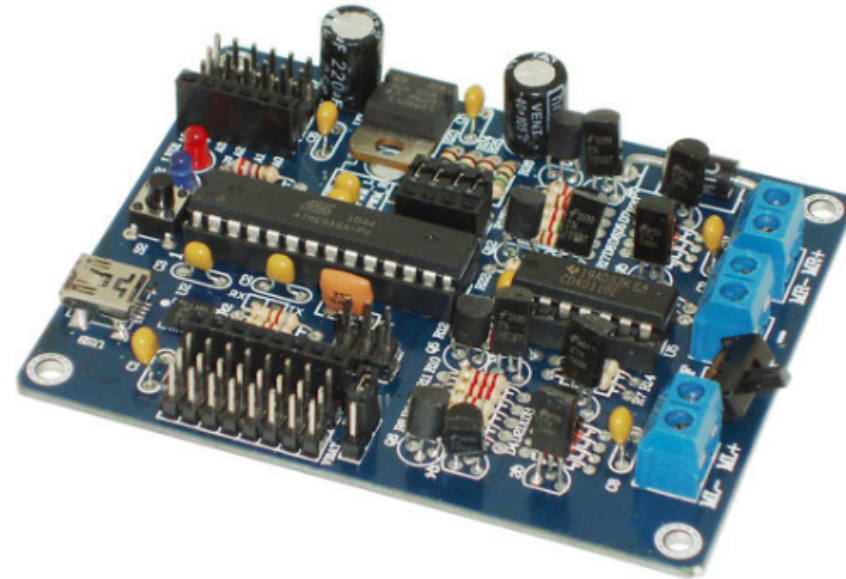
## 20. Pour info : les cartes d'interfaces moteurs CC « intégrées »

### Présentation

A noter qu'il existe également des interfaces moteurs couplées au micro-contrôleur sur une même carte. Deux exemples :



La carte Romeo de DFRobot (prix constaté ~32€)



La carte RS015 de Dagu (prix constaté ~26€)

### Qu'en penser ?

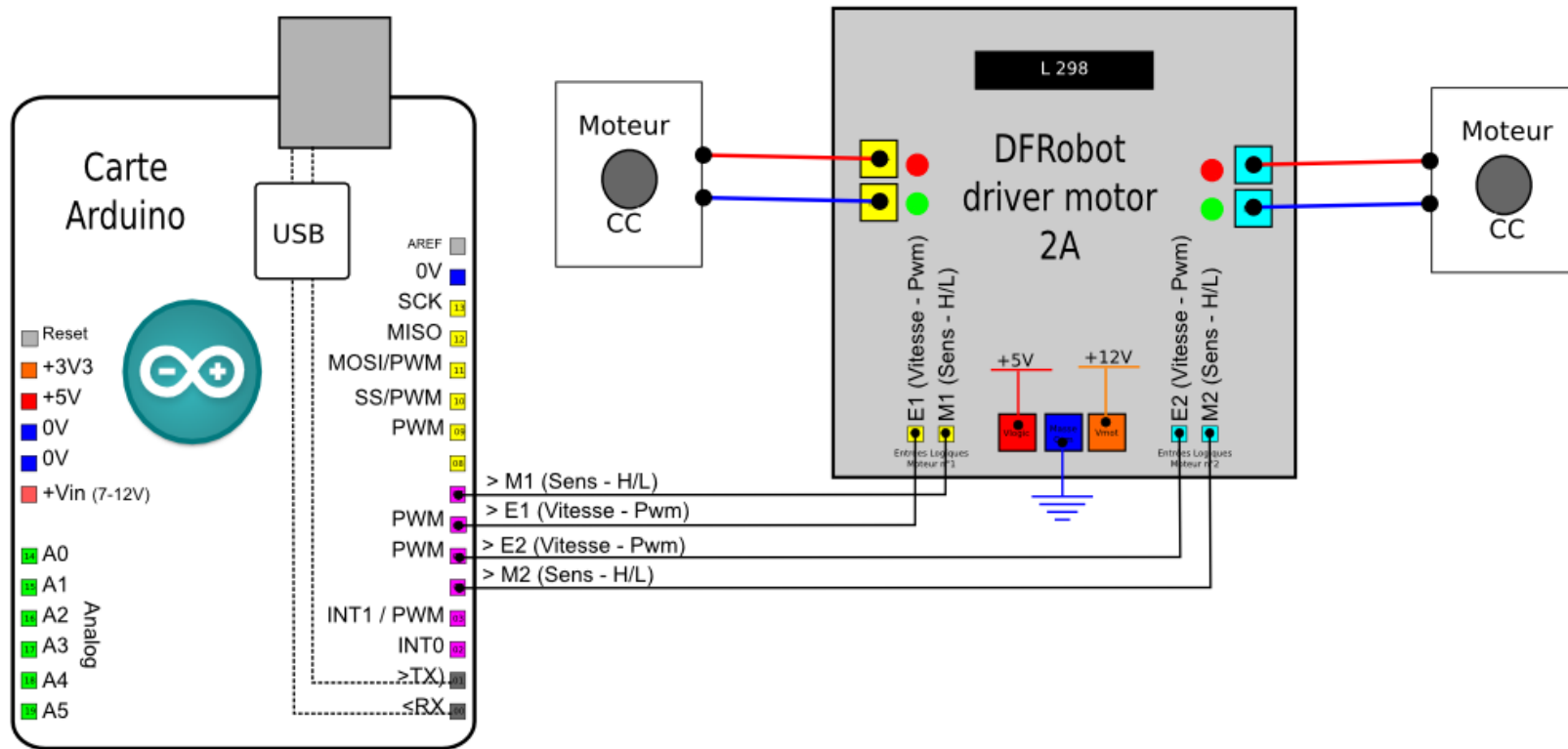
- le coût est souvent moindre que l'achat séparé Carte Arduino + interface moteur (-30% environ)
- sur un projet précis utilisant des moteurs, cela peut être utile, d'autant que la connectique est souvent bien pratique (la RS015 de Dagu)
- mais pour de l'expérimentation ou une utilisation ponctuelle des moteurs, c'est probablement mieux d'utiliser des shields dédiés sauf évidemment si le coût n'est pas très supérieur... d'autant que l'on perd souvent des broches par rapport à une carte Arduino et le shields ne seront pas forcément utilisables.
- d'autre part, la compatibilité réelle avec Arduino est-elle au rendez-vous ?

### Mon avis

En pratique, enfin, comme dans la vie d'ailleurs, **il est bon de ne pas trop associer les choses entre-elles sur un robot**, car si quelque chose grille, on se retrouve avec tous les problèmes à la fois et les réparations sont plus compliquées (toute la carte à changer, tous les câblages à refaire, etc...). Avec l'utilisation de cartes séparées moteur/Arduino, il est facile de tester l'un et l'autre séparément et d'identifier le problème, de changer le composant ou la carte voulue. C'est un avis personnel, mais tiré d'une expérience de plusieurs années de fabrication de robots. Quelques euros de plus font parfois économiser de nombreuses heures de temps perdu !

**En conclusion : certaines cartes intégrées peu chères peuvent être intéressantes sur un projet de robot dédié.**

## 21. Les moteurs et moto-réducteurs CC : les circuits d'interface sens/vitesse (contrôle de la vitesse et du sens) : le montage type de 2 moteurs avec une carte Arduino



Purchased by Franck Ourion, [franck.ourion@univ-lorraine.fr](mailto:franck.ourion@univ-lorraine.fr) #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.

## 22. Les moteurs et moto-réducteurs CC : Interfaces de contrôle : Synthèse

### Les interfaces « ON/OFF » simples (ex : ULN 2803) :

- réalisent l'**adaptation en tension** (broche E/S en 5V ==> moteur 6-30V )
- réalisent l'**adaptation en intensité** (broche E/S en 15mA ==> moteur 200mA à plusieurs Ampères ! )
- sont « **transparentes** » aux impulsions PWM et donc permettent le **contrôle de la vitesse**
- ne permettent pas le contrôle de la polarité et donc ne permettent **pas le contrôle du sens de rotation**



Le RotoShield de chez Snootlab, basé sur 2 x L293.

### Les interfaces à double « pont en H » (ex : L293D, L298, ou équiv.) :

- réalisent l'**adaptation en tension** (broche E/S en 5V ==> moteur 6-30V )
- réalisent l'**adaptation en intensité** (broche E/S en 15mA ==> moteur 200mA à plusieurs Ampères ! )
- sont « **transparentes** » aux impulsions PWM et donc permettent le **contrôle de la vitesse**
- **permettent le contrôle de la polarité et donc permettent le contrôle du sens de rotation**

Selon les modèles, ces interfaces disposeront :

- soit d'**une broche de direction + une broche de vitesse (PWM)** par moteur
- soit de **deux broches contrôlant chacune un sens et la vitesse (PWM)** dans ce sens par moteur

Un CI double « pont en H » permet de contrôler soit 2 moteurs CC, soit un moteur pas à pas bipolaire.

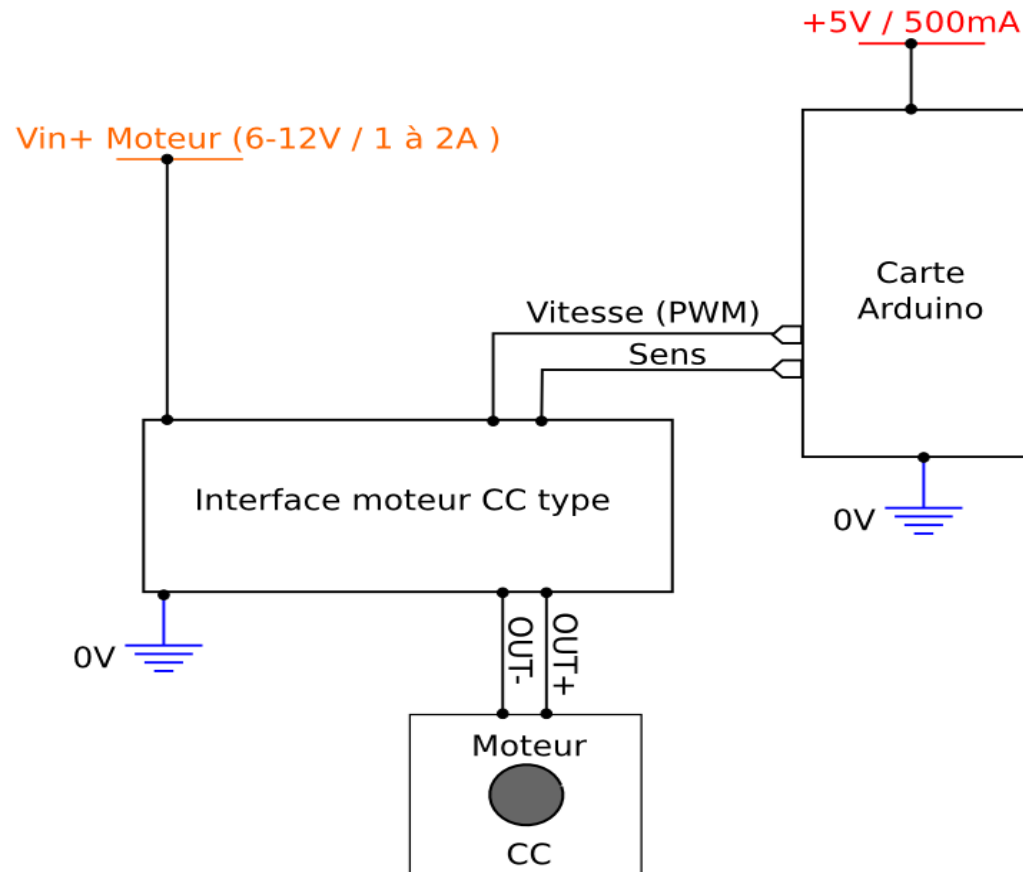
## 23. Contrôle du sens et de la vitesse d'un moteur (ou moto-réducteur CC) : le schéma théorique

Pour contrôler le sens et la vitesse d'un moteur CC, nous allons avoir besoin d'un étage « pont en H ». Plusieurs interfaces sont disponibles et 2 modes de fonctionnement existent :

- soit d'une **broche de direction + une broche de vitesse** (PWM) par moteur, (le plus courant et aussi le plus pratique)
- soit de **deux broches contrôlant chacune un sens et la vitesse (PWM) dans ce sens** par moteur

Nous prenons ici l'exemple d'une interface type pour moteur CC fonctionnant avec 1 broche de sens et une broche de vitesse (PWM) :

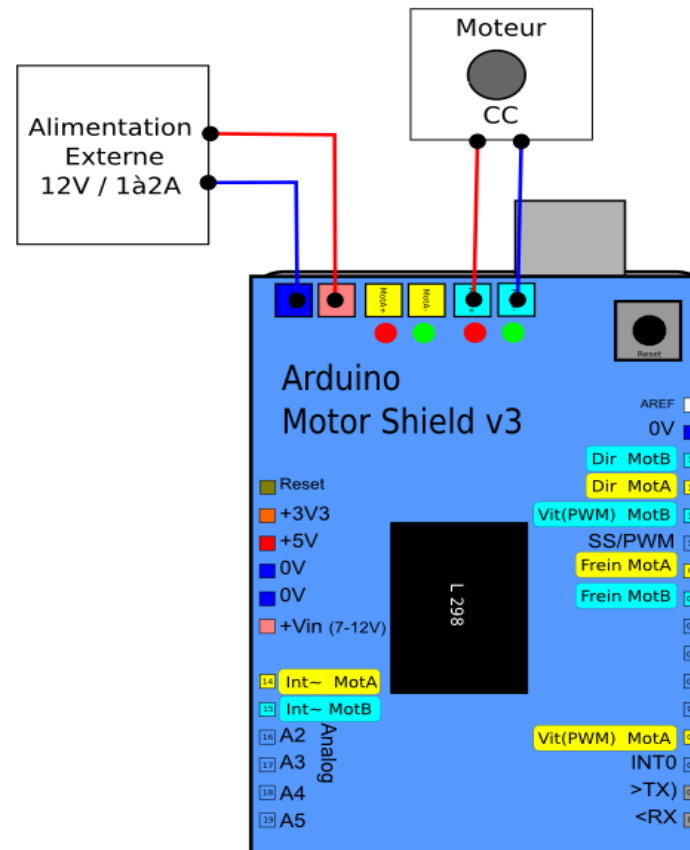
- une broche de la carte Arduino contrôlera le sens
- l'autre broche de la carte Arduino contrôlera la vitesse (PWM)





## 24. Contrôle du sens et de la vitesse d'un moteur (ou moto-réducteur CC) : exemple de montage

Nous allons utiliser ici l'interface Arduino motor shield (V3) (basée sur un L298) et qui permet le contrôle de **2 moteurs CC jusqu'à 2A chacun**. Le montage consiste à enficher le motor shield sur la carte Arduino, tout simplement, à connecter l'alimentation externe sur le bornier Vin et un moteur sur l'un des deux borniers moteur disponible :



Remarquer que chaque « étage moteur » dispose sur ce shield :

- d'une broche de **sens** ou direction (broche Dir)
- d'une broche de **vitesse** (broche Vit) de type PWM
- d'une broche de frein (pas utilisée ici) qui bloque le moteur si à HIGH – pas utilisé dans ce programme
- d'une entrée analogique permettant de connaître l'intensité utilisée par le moteur (sortie =  $0V + 1,65V/A$  (soit 3,3V pour 2A) – pas utilisé dans ce programme

Plus de détails sur ce shield ici : [http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.MATERIELArduinoShieldArduinoMotoShieldV3L298](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.MATERIELArduinoShieldArduinoMotoShieldV3L298)

Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.

## 25. Contrôle du sens et de la vitesse d'un moteur (ou moto-réducteur CC) : le programme

Ce qu'on va faire ici...

- Le programme que je vous propose est assez simple : on va reprendre la même structure que le programme précédent, mais cette fois, on va gérer également le sens de rotation.
- On va faire tourner le moteur (ou le moto-réducteur selon ce que vous utilisez...) dans un sens en augmentant progressivement puis réduisant la vitesse,
- puis on va faire tourner le moteur (ou le moto-réducteur selon ce que vous utilisez...) dans l'autre sens en augmentant progressivement puis réduisant la vitesse
- Rien de bien compliqué donc, mais les bases du contrôle d'un moteur sens et vitesse seront ainsi posées. Le point essentiel ici est l'utilisation d'une broche de commande du sens du moteur et d'une broche PWM pour la vitesse.

### Entête déclarative

- On déclare :
  - une constante de broche de contrôle de la vitesse du moteur. Cette broche est de type PWM (ici, la 11 – moteur B du shield)
  - une constante de broche de contrôle du sens de rotation du moteur, de type numérique (ici, la 13 – moteur B du shield)
  - une variable **int** fixant la pause entre 2 changement de la vitesse
  - une variable **int** appelée largeur et fixant la largeur de l'impulsion
- On déclare également 2 constantes appelée AVANT et ARRIERE correspondant respectivement à l'état de la broche de sens correspondant. Ceci permet d'écrire un programme plus lisible.

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

const int vitesseMOTEUR=11; // constante désignant la broche vitesse
(PWM) du moteur
const int sensMOTEUR=13; // constante désignant la broche sens moteur

int pause=50; // variable fixant la durée de la pause entre 2 changement
de vitesse en ms

int largeur=0; // variable pour la largeur d'impulsion PWM

const int AVANT=HIGH; // constante sens avant
const int ARRIERE=LOW; // constante sens arrière
```

### Fonction **setup()**

#### Configuration des broches utilisées

- On commence par configurer en sortie la broche de contrôle de la vitesse du moteur avec l'instruction **pinMode**(broche, sens),
- et de la même façon, on configure en sortie la broche de contrôle du sens de rotation du moteur.

#### Test initial du moteur

- puis on réalise un test initial du moteur :
  - on met la broche du moteur à 1 pendant 1 seconde, le moteur tourne alors à fond.
  - puis on arrête le moteur pendant 1 seconde.

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    pinMode(vitesseMOTEUR, OUTPUT); // met la broche en sortie
    pinMode(sensMOTEUR, OUTPUT); // met la broche en sortie

    //-- test du moteur --

    digitalWrite(sensMOTEUR, AVANT); // fixe la rotation en sens AVANT

    digitalWrite(vitesseMOTEUR,HIGH); // moteur à fond
    delay(1000); // 1 seconde

    digitalWrite(vitesseMOTEUR,LOW); // moteur à l'arrêt
    delay(1000); // 1 seconde

} // fin de la fonction setup()
```

## Fonction `loop()`

- On commence par ouvrir une boucle globale qui réalisera 2 passage au sein de la fonction `loop()` :
  - au premier passage, on fixe le sens en AVANT
  - au second passage, on fixe le sens en ARRIERE
- Ensuite, dans les 2 cas, on réalise une variation de la vitesse dans chaque sens :
  - A l'aide d'une première boucle `for()`, on augmente progressivement la vitesse de 0 à 100% en élargissant progressivement l'impulsion PWM générée avec l'instruction `analogWrite(broche,largeur)`
  - De la même façon, on utilise une seconde boucle `for()` pour réduire la vitesse de 100% à 0%.
- Noter l'utilisation de la fonction `map()` qui permet de transposer la valeur en % (échelle 0-100) en la valeur correspondante sur l'échelle PWM 0-255.
- Des instructions `delay()` fixent la pause entre chaque changement de vitesse.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    for (int j=1; j<=2; j++) { // 2 passages

        if (j==1) digitalWrite(sensMOTEUR, AVANT); // fixe la rotation
        en sens AVANT
        if (j==2) digitalWrite(sensMOTEUR, ARRIERE); // fixe la rotation
        en sens ARRIERE

        for (int i=0; i<=100; i++) { // boucle croissante

            largeur= map(i,0,100, 0,255); // ré-échelonne i (0-100%) vers
            (0-255)
            analogWrite(vitesseMOTEUR,largeur); // génère impulsion PWM
            voulue sur la broche
            delay(pause); // pause

        } // fin for i

        for (int i=100; i>=0; i--) { // boucle décroissante

            largeur= map(i,0,100, 0,255); // ré-échelonne i (0-100%) vers
            (0-255)
            analogWrite(vitesseMOTEUR,largeur); // génère impulsion PWM
            voulue sur la broche
            delay(pause); // pause

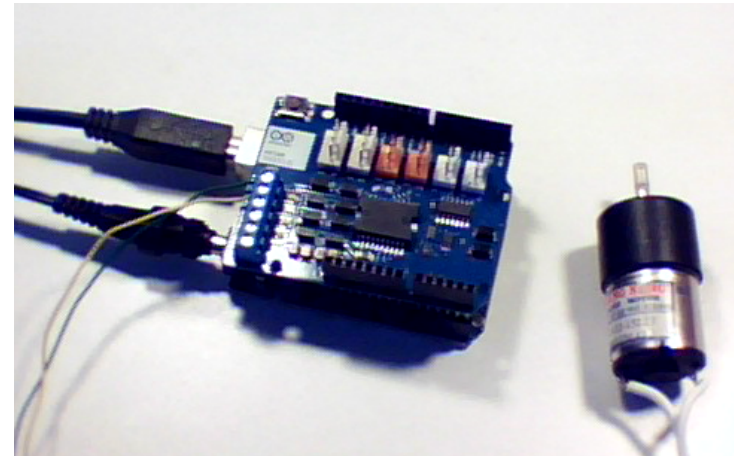
        } // fin for i

    } // fin for j - 2 passages

} // fin de la fonction loop()
```

## Fonctionnement du programme

- Une fois la carte Arduino programmée :
  - le moteur tourne à fond en sens AVANT 1 seconde puis s'arrête,
  - puis, en sens AVANT, le moteur se remet à tourner progressivement de plus en plus vite de 0% à 100% puis la vitesse diminue à nouveau de 100% jusqu'à l'arrêt (0%)
  - puis, en sens ARRIERE, le moteur se remet à tourner progressivement de plus en plus vite de 0% à 100% puis la vitesse diminue à nouveau de 100% jusqu'à l'arrêt (0%)
  - et ainsi de suite...



### IMPORTANT

IL EST INDISPENSABLE D'ALIMENTER LA CARTE ARDUINO OU LE SHIELD AVEC UNE **ALIMENTATION EXTERNE 6 à 12V / 1 à 2 A (en fonction du moteur que vous utilisez)** à brancher dans le connecteur d'alimentation ou sur le bornier Vin du shield.

## 26. Truc technique... : une alimentation régulée de « labo » à pas cher !

### Un point essentiel avec les moteurs (ou moto-réducteurs) à courant continu : l'intensité de l'alimentation !

La principale difficulté lors de la mise en oeuvre des moteurs ou moto-réducteurs à courant continu est d'ordre électrique, la programmation n'est pas plus difficile que ce que vous avez déjà vu. Un moteur va consommer une intensité qui peut aller de 500mA jusqu'à 2A selon les modèles. De plus, lors de la mise sous tension, il existe souvent un pic en intensité plus élevé. Dans l'hypothèse de 2 moteurs ou moto-réducteurs à courant continu en 12V, et consommant chacun 1,5A par exemple, il va falloir fournir 3A minimum.

**Le point crucial ici va donc être de pouvoir fournir une telle intensité à la tension voulue avec une alimentation adaptée.**

Une solution simple et peu coûteuse est l'utilisation d'une alimentation de PC de récupération.

### BON à savoir :

une alimentation de PC de récupération, dite alimentation ATX, est une alimentation régulée peu coûteuse (<10€) et qui fournit du 5V régulé sous plusieurs ampères, du 12V régulé sous plusieurs ampères, et aussi du 3.3V régulé, du -12V régulé, etc.. Pratique !

A comparer à un bloc secteur 220V AC / 6-12V DC qui ne fournira que 0.5 à 1A dans le meilleur des cas, pour le même prix...

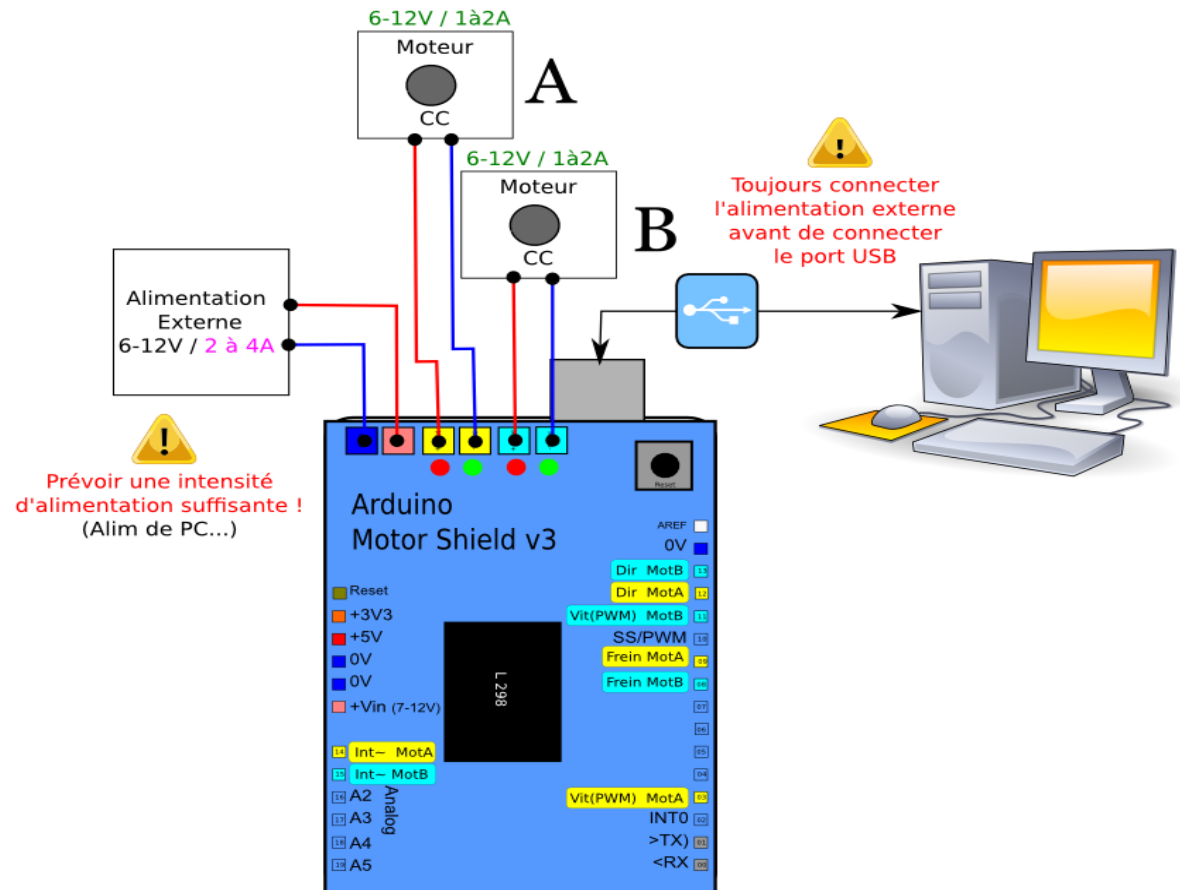
Avec ça, vous êtes sur de pouvoir alimenter tous vos projets, y compris (et surtout) si vous utilisez des moteurs ou de nombreux servomoteurs !



Pour activer l'alimentation, au niveau du connecteur ATX, il suffit de mettre un strap entre la broche verte (16) et la masse (0V - noir)

## 27. Contrôler 2 moteurs (ou motoréducteurs) à courant continu depuis le Terminal Série : le montage

Nous allons utiliser ici l'interface Arduino motor shield (V3) (basée sur un L298) et qui permet le contrôle de **2 moteurs CC jusqu'à 2A chacun**. Le montage consiste à enficher le motor shield sur la carte Arduino, tout simplement, à connecter l'alimentation externe sur le bornier Vin et un moteur sur chacun des deux borniers moteur disponibles :



Remarquer que chaque « étage moteur » dispose sur ce shield :

- d'une broche de **sens** ou direction (broche Dir)
- d'une broche de **vitesse** (broche Vit) de type PWM
- d'une broche de frein (pas utilisée ici) qui bloque le moteur si à HIGH – pas utilisé dans ce programme
- d'une entrée analogique permettant de connaître l'intensité utilisée par le moteur (sortie =  $0V + 1,65V/A$  (soit 3,3V pour 2A) – pas utilisé dans ce programme

Plus de détails sur ce shield ici : [http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.MATERIELArduinoShieldArduinoMtroShieldV3L298](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.MATERIELArduinoShieldArduinoMtroShieldV3L298)

Purchased by Franck Ourion, franck.ourion@univ-lorraine.fr #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.



## 28. Contrôler 2 moteurs (ou motoréducteurs) à courant continu depuis le Terminal Série : le programme

Un des objectifs premiers à l'utilisation des moteurs ou motoréducteurs avec contrôle de la vitesse et du sens est la réalisation d'un robot mobile. Avec 2 moto-réducteurs utilisés avec une interface pour 2 moteurs CC, il est en effet assez facile de contrôler le sens de rotation et la vitesse pour chaque moteur et donc de programmer des actions synchronisées entre les moteurs, notamment la marche avant, la marche arrière, la bifurcation vers la droite ou vers la gauche et bien sûr l'arrêt ! Le tout, avec 2 broches de la carte Arduino par moteur (soit 4 en tout). Dans ce programme, je vous propose de poser les bases du contrôle d'un tel robot à partir du Terminal Série. Ce programme permettra de tester facilement le fonctionnement d'un robot motorisé de cette façon.

### Entête déclarative

#### Déclarations des constantes et variables pour les moteurs

- On déclare pour chaque moteur :
  - une constante de broche de contrôle de la vitesse du moteur. Cette broche est de type PWM (ici, la 3 et la 11 – moteur A et B)
  - une constante de broche de contrôle du sens de rotation du moteur, de type numérique (ici, la 12 et la 13 – moteur A et B)
- On déclare également
  - une variable `int` appelée vitesse et fixant la largeur de l'impulsion
- On déclare également 2 constantes appelée AVANT et ARRIERE correspondant respectivement à l'état de la broche de sens correspondant. Ceci permet d'écrire un programme plus lisible.

#### Variables pour la gestion du port série en réception

- En ce qui concerne la réception sur le port série, on déclare :
  - une variable `int` pour stocker l'octet en réception (code ASCII du caractère),
  - une variable `char` et un `String` pour la réception de la chaîne de caractère sur le port série.

```
//--- entete déclarative
// = déclarer ici variables et constantes globales

//--- variables et constantes pour les moteurs
const int vitesseMoteurD=11; // constante désignant la broche vitesse
(PWM) du moteur
const int sensMoteurD=13; // constante désignant la broche sens moteur

const int vitesseMoteurG=3; // constante désignant la broche vitesse
(PWM) du moteur
const int sensMoteurG=12; // constante désignant la broche sens moteur

int vitesse=50; // variable pour la largeur d'impulsion PWM 0 à 100%

const int AVANT=HIGH; // constante sens avant
const int ARRIERE=LOW; // constante sens arrière

//--- variables pour réception chaîne sur port Série
int octetReception=0; // variable de réception octet
char caractereReception=0; // variable de réception caractère
String chaineReception=""; // déclare un objet String vide
```

## Fonction **setup()**

### **Configuration des broches utilisées pour les moteurs**

- On commence par configurer en sortie la broche de contrôle de la vitesse du moteur avec l'instruction **pinMode**(broche, sens),
- et de la même façon, on configure en sortie la broche de contrôle du sens de rotation du moteur.

### **Initialisation de la communication série**

- on initialise la communication série avec l'instruction **Serial.begin**(vitesse). On utilisera 115200 bauds.

### **Messages d'initialisation**

- on affiche la liste des chaînes reconnues par le programme ce qui facilitera l'utilisation

```
//--- la fonction setup() : exécutée au début et 1 seule fois
void setup() {

    //---- initialisation moteurs ----
    pinMode(vitesseMoteurD, OUTPUT); // met la broche en sortie
    pinMode(sensMoteurD, OUTPUT); // met la broche en sortie

    pinMode(vitesseMoteurG, OUTPUT); // met la broche en sortie
    pinMode(sensMoteurG, OUTPUT); // met la broche en sortie

    //---- initialisation port série ----
    Serial.begin(115200); // initialise la vitesse de la connexion série
    //-- utilise la meme vitesse dans le Terminal Série

    Serial.println("--- Instructions reconnues: ---");
    Serial.println("stop");
    Serial.println("avantD");
    Serial.println("avantG");
    Serial.println("arriereD");
    Serial.println("arriereG");
    Serial.println("enAvant");
    Serial.println("enArriere");
    Serial.println("tourneD");
    Serial.println("tourneG");
    Serial.println("-----");

} // fin de la fonction setup()
```

## Fonction `loop()`

### Gestion du port Série

- A ce niveau, on va « écouter » le port Série en testant l'arrivée d'un caractère à l'aide d'une boucle `while` pour tester la présence d'un octet dans la file d'attente du port série avec la fonction `Serial.available()`
- Tant qu'un octet différent du saut de ligne est présent on ajoute le caractère à la chaîne de réception.
- Et si c'est un saut de ligne que l'on reçoit :
  - on affiche la chaîne reçue
  - on teste à l'aide de conditions `if()` le contenu de la chaîne. On appelle la fonction voulue du programme en conséquence.
  - puis on sort de la boucle `while`.

```
//--- la fonction loop() : exécutée ensuite en boucle sans fin
void loop() {

    while (Serial.available()>0) { // si un caractère en réception

        octetReception=Serial.read(); // lit le 1er octet de la file
        d'attente

        if (octetReception==10) { // si Octet reçu est le saut de ligne
            Serial.print ("Saut de ligne reçu : ");
            Serial.println ("Chaîne reçue = "+chaîneReception); // affiche
            la chaîne reçue

            //--- effectue action si la chaîne attendue est reçue
            if (chaîneReception=="avantD") moteurDroit(AVANT,vitesse);
            if (chaîneReception=="avantG") moteurGauche(AVANT,vitesse);

            if (chaîneReception=="arriereD") moteurDroit(ARRIERE,vitesse);
            if (chaîneReception=="arriereG")
            moteurGauche(ARRIERE,vitesse);

            if (chaîneReception=="enAvant") enAvant(vitesse);
            if (chaîneReception=="enArriere") enArriere(vitesse);

            if (chaîneReception=="tourneD") tourneD(vitesse);
            if (chaîneReception=="tourneG") tourneG(vitesse);

            if (chaîneReception=="stop") stopMoteurs();

            chaîneReception=""; //RAZ le String de réception
            delay(100); // pause
            break; // sort de la boucle while
        } // fin if

        else { // si le caractère reçu n'est pas un saut de ligne
            caractereReception=char(octetReception); // récupère le caractère
            à partir du code Ascii
            chaîneReception=chaîneReception+caractereReception; // ajoute la
            caractère au String
            delay(1); // laisse le temps au caractères d'arriver
        } // fin else

    } // fin while - fin de réception de la chaîne
} // fin loop
```

## Fonctions gérant les mouvements individuels des servomoteurs

### ***Fonction de gestion du sens de rotation et de la vitesse du moteur Droit***

- Cette fonction reçoit le sens de rotation à utiliser ainsi que la vitesse à utiliser entre 0 et 100%
- Cette fonction ne renvoie rien (type void)
- Le code de la fonction consiste à :
  - à fixer le sens voulu du moteur
  - à fixer la largeur d'impulsion voulue
- noter l'utilisation de la fonction `map()` à l'intérieur de la fonction `analogWrite()` qui permet ici de ré-échelonner la vitesse depuis l'échelle 0-100% vers l'échelle PWM 0-255.

### ***Fonction de gestion du sens de rotation et de la vitesse du moteur Gauche***

- On procède de la même façon pour le moteur Gauche.

```
//----- fonctions controle marche AVANT/ARRIERE moteurs

void moteurDroit(int sensIn, int vitesseIn) {

    digitalWrite(sensMoteurD, sensIn); // en marche avant

    analogWrite(vitesseMoteurD, map(vitesseIn, 0, 100, 0, 255)); // impulsion
    PWM vitesse

} // fin moteurDroit

void moteurGauche(int sensIn, int vitesseIn) {

    digitalWrite(sensMoteurG, sensIn); // en marche avant

    analogWrite(vitesseMoteurG, map(vitesseIn, 0, 100, 0, 255)); // impulsion
    PWM vitesse

} // fin moteurGauche
```

## Fonctions gérant les mouvements synchronisés des servomoteurs

### **Fonction gérant la marche avant**

- Cette ne renvoie rien (type void)
- Cette fonction reçoit en paramètre la vitesse (entre 0 et 100%)
- Le code de cette fonction consiste à mettre en marche avant les 2 moteurs en se basant sur la fonction de gestion individuelle de la marche avant.

### **Fonction gérant la marche arrière**

- Cette ne renvoie rien (type void)
- Cette fonction reçoit en paramètre la vitesse (entre 0 et 100%)
- Le code de cette fonction consiste à mettre en marche arrière les 2 moteurs en se basant sur la fonction de gestion individuelle de la marche arrière.

### **Fonction gérant la bifurcation à droite et à gauche**

- Cette ne renvoie rien (type void)
- Cette fonction reçoit en paramètre la vitesse (entre 0 et 100%)
- Le code de cette fonction consiste à mettre en marche arrière l'un des 2 moteurs et en marche arrière le second en se basant sur la fonction de gestion individuelle de la marche arrière/avant.

### **La fonction d'arrêt des moteurs**

- Cette fonction stoppe les 2 moteurs à l'aide en application une impulsion PWM nulle sur la broche de contrôle de la vitesse de chacun des moteurs.

```
//--- fonctions controle actions synchronisées

void enAvant(int vitesseIn) {
    moteurDroit(AVANT, vitesseIn);
    moteurGauche(AVANT, vitesseIn);
} // fin enAvant

void enArriere(int vitesseIn) {
    moteurDroit(ARRIERE, vitesseIn);
    moteurGauche(ARRIERE, vitesseIn);
} // fin enArriere

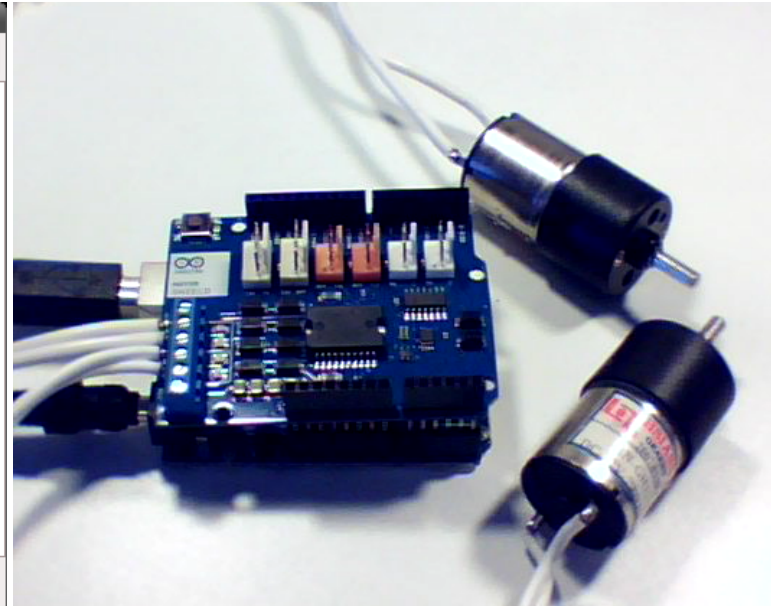
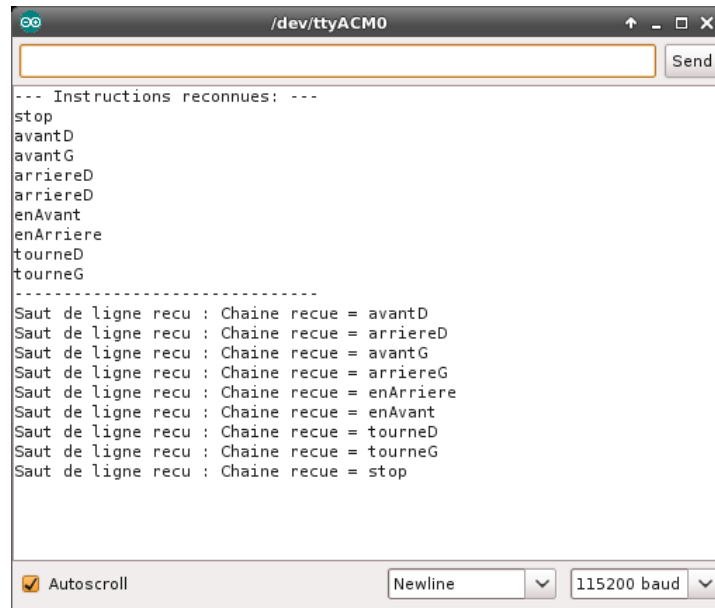
void tourneD(int vitesseIn) {
    moteurDroit(ARRIERE, vitesseIn);
    moteurGauche(AVANT, vitesseIn);
} // fin tourneD

void tourneG(int vitesseIn) {
    moteurDroit(AVANT, vitesseIn);
    moteurGauche(ARRIERE, vitesseIn);
} // fin tourneG

//--- fonction stop ---
void stopMoteurs() {
    analogWrite(vitesseMoteurD,0); // arret moteur
    analogWrite(vitesseMoteurG,0); // arret moteur
} // fin stop
```

## Fonctionnement du programme

- Ouvrir le Terminal Série (Tools > Serial Monitor) et fixer le débit à la même valeur que celle utilisée pour l'instruction `Serial.begin(vitesse)`. Ici, **115200** bauds.
- Régler également les paramètres de transmission de la chaîne de caractère à l'aide de la 2ème liste défilante. **Mettre sur « New Line »** pour ajout du « saut de ligne » après la chaîne saisie.
- saisir l'une des chaînes reconnues : les moteurs réalisent l'action demandée ! Très très pratique pour tester son petit robot par le port Série.



Les chaînes de caractères contrôlent les rotations individuelles ou synchrones des 2 moto-réducteurs !

**Bravo !**

A ce stade, vous avez tous les éléments en main pour construire un robot mobile d'une charge de 1 à 2 kilos !

## 29. Complément : Théorie utile : la notion de « couple moteur »

### Introduction

Dans les pages précédentes, on a parlé du couple d'un moteur. Voici quelques rappels de notions utiles. Il paraît que c'est super compliqué, le couple moteur... il paraît...

### Une force s'exprime en Newton



Vous connaissez tous je pense l'histoire de Newton qui assis sous un pommier, aurait reçu une pomme sur la tête... ce qui lui aurait donné l'idée des lois de la gravitation universelle et la fameuse formule :  $F = m \cdot g$  (avec  $F$  la force en Newton,  $m$  la masse en kg et  $g$  l'accélération en  $m.s^{-2}$ )

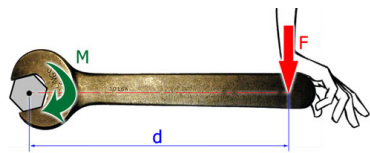
Tout ça pour dire qu'une force se mesure en Newton. Concrètement, **1 newton est la force capable de communiquer à une masse de 1 kilogramme une accélération de 1  $m.s^{-2}$**

Pour faire plus simple, une masse de 1kg exerce une force de 1 kg . 9,81  $m.s^{-2}$  soit 10 newtons environ. Encore plus simple : 1 boîte de chocolat de **100g** exerce une force de **1**

**Newton** (sur la Terre bien sûr...).

Remarque : on confond souvent le poids (qui est une force) et la masse (en kg)...Essayez ça pour voir au magasin : « Je voudrais 10 Newtons d'orange s'il vous plaît... » (1 kg quoi...)

### Le moment d'une force

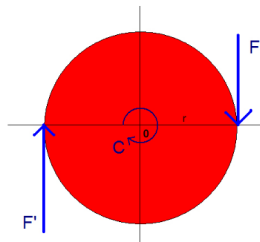


On appelle « moment » d'une force son « effet » sur l'axe de rotation. Le moment vaut :  $M = F \cdot d$  où  $d$  est le rayon ou distance à l'axe.

Avec une clé longue, on dévisse plus facilement un écrou !

### Le couple d'un axe de rotation, c'est quoi ?

**Un couple, c'est la résultante de l'action de 2 moments symétriques** sur un axe = la somme des forces est nulle. Un schéma vaut mieux qu'un long discours :



### La formule

Si on exerce une force  $F$  à une distance  $R$  de l'axe, le couple  $C$  (en  $N.m$ ) sera :

$$C = F \times 2 \times \text{Rayon} = F \times \text{Diamètre}$$

(remarquer que c'est en fait la somme des couples  $F \times R$ )

A l'inverse, pour un couple donné, la force à une distance  $R$  de l'axe vaut :

$$F = C / (2 \times \text{Rayon}) = C / D$$

### Le couple.. à la main.

Prenons l'exemple du volant que vous tournez à 2 mains : chaque main est à la même distance de l'axe et vous exercez une force équivalente avec chaque main : le couple moteur est la force que vous exercez  $\times$  le diamètre de l'axe.

Imaginons que c'est un grand volant (1 m de diamètre) et que vous exercez une force de 10 Newtons sur chaque main : on dira que le couple vaut 10N.m ! Compris ?

### Le couple... au moteur.

A l'inverse, un axe de moteur va tourner par lui même : dans ce cas, c'est la mécanique interne du moteur qui exerce la force de rotation sur l'axe... qui tourne. Dans ce cas, le couple va exprimer la force qui est exercée pour chaque côté du diamètre (soit à  $R$  de l'axe) selon  $F = C / D$ .

### Exemple concret

Une fiche technique indique un couple moteur de 3kg.cm.... ça veut donc dire quoi ?? Que le moteur exerce une force équivalente à 3kg pour 1cm de diamètre soit 0,5cm de l'axe mais aussi :

- 1 kg pour 3cm de diamètre soit à 1,5 cm de l'axe
- 30g pour 1m de diamètre soit à 50cm de l'axe
- $C/D$  kg pour  $D$  cm de diamètre soit à  $D/2$  de l'axe

Autre exemple : un servomoteur a un couple de 4kg.cm en 6Vcc... ce qui veut dire une force de 4kg pour 1cm de diamètre, soit 4kg à 0,5cm de l'axe...

### Quelques conséquences :

- des engrenages de réduction vont réduire la vitesse... et augmenter le couple.
- plus une roue est grande et plus la force motrice sera faible
- ...



### 30. Les éléments du langage Arduino étudiés dans cet atelier

#### Structure

#### Variables et constantes

#### Fonctions

##### Sortie analogique (impulsion)

- [analogWrite](#)(broche, valeur)

##### Temps

- [delay](#)(ms)

##### Math

- [map](#)(valeur, fromLow, fromHigh, toLow, toHigh)

La documentation complète du langage Arduino en français est disponible ici :  
[http://www.mon-club-elec.fr/pmwiki\\_reference\\_arduino/pmwiki.php?n=Main.ReferenceMaxi](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.ReferenceMaxi)

### **31. A présent, vous devriez être capable :**

- Contrôler des moteurs ou des moto-réducteurs à courant continu (CC) à l'aide d'une interface basée sur un « pont en H » et d'en contrôler la vitesse de rotation et le sens.

## Table des matières

Moteurs : Apprendre à utiliser des moteurs (ou moto-réducteurs) à courant continu (ou CC) avec une carte Arduino.

Intro |

Matériel nécessaire pour les ateliers Arduino |

Matériel spécifique nécessaire pour cet atelier |

Remarque |

Technique : l'alimentation de la carte Arduino |

Les moteurs et moto-réducteurs à courant continu ou CC : concrètement |

Les moteurs et moto-réducteurs CC : fiche technique |

Les moteurs et moto-réducteurs CC : exemples d'utilisation |

Rappel : Caractéristiques électriques d'une broche Numérique Arduino en sortie |

Notion d'amplification de puissance et d'interface de puissance |

Les moteurs et moto-réducteurs CC : schéma électrique type d'utilisation d'une interface avec Arduino |

Les moteurs et moto-réducteurs CC : le principe de contrôle |

Info technique : les circuits intégrés en boîtier DIL |

Les moteurs et moto-réducteurs CC : les circuits d'interface ON/OFF (contrôle de la vitesse seule dans 1 sens) : Exemple du CI ULN 2803 |

Les moteurs et moto-réducteurs CC : les circuits d'interface ON/OFF (contrôle de la vitesse seule dans 1 sens) : le montage type d'un ULN 2803 avec une carte Arduino |

Contrôle simple de la vitesse (seule) d'un moteur (ou moto-réducteur CC) : le programme |

Les moteurs et moto-réducteurs CC : principe de contrôle du sens par une interface |

Les moteurs et moto-réducteurs CC : contrôle du sens : le concept de « pont en H » (ou H-bridge) |

Les moteurs et moto-réducteurs CC : les circuits d'interface « pont en H » en pratique |

Pour info : les cartes d'interfaces moteurs CC « intégrées » |

Les moteurs et moto-réducteurs CC : les circuits d'interface sens/vitesse (contrôle de la vitesse et du sens) : le montage type de 2 moteurs avec une carte Arduino |

Les moteurs et moto-réducteurs CC : Interfaces de contrôle : Synthèse |

Contrôle du sens et de la vitesse d'un moteur (ou moto-réducteur CC) : le schéma théorique |

Contrôle du sens et de la vitesse d'un moteur (ou moto-réducteur CC) : exemple de montage |

Contrôle du sens et de la vitesse d'un moteur (ou moto-réducteur CC) : le programme |

Truc technique... : une alimentation régulée de « labo » à pas cher ! |

Contrôler 2 moteurs (ou motoréducteurs) à courant continu depuis le Terminal Série : le montage |

Contrôler 2 moteurs (ou motoréducteurs) à courant continu depuis le Terminal Série : le programme |

Complément : Théorie utile : la notion de « couple moteur » |

Les éléments du langage Arduino étudiés dans cet atelier |

A présent, vous devriez être capable : |

**Bravo !**  
vous avez terminé cet atelier Arduino !



Prêt pour la suite ? Retrouvez de nombreux autres thèmes d'ateliers Arduino ici :

[http://www.mon-club-elec.fr/pmwiki\\_mon\\_club\\_elec/pmwiki.php?n=MAIN.ATELIERS](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)