

Moteurs : Apprendre à utiliser des servomoteurs à rotation continue avec une carte Arduino.



Ateliers Arduino

par X. HINAULT

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

Pour obtenir tout autres types de licence d'utilisation (enseignement, commercial, etc...), veuillez contacter l'auteur ici : 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

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 servomoteurs à rotation continue avec une carte Arduino afin d'être en mesure d'en contrôler la vitesse, le sens de rotation et de réaliser facilement un petit robot didactique.

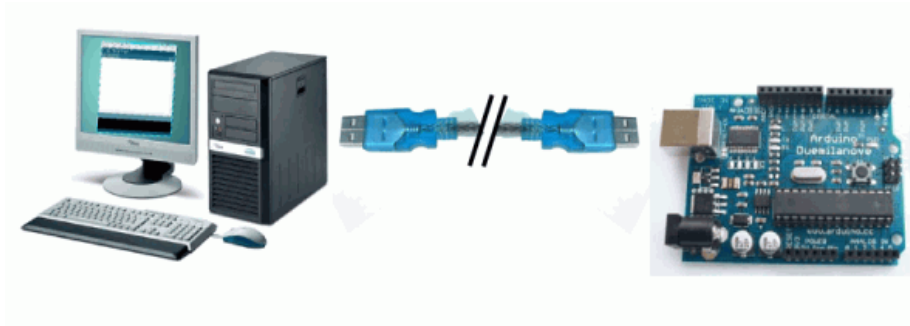


Prêt ? C'est parti !

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

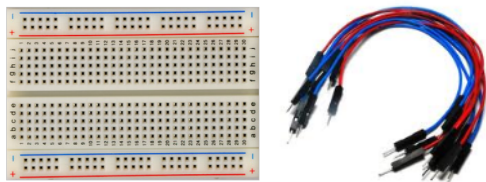


L'espace de développement Arduino associe :

- 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

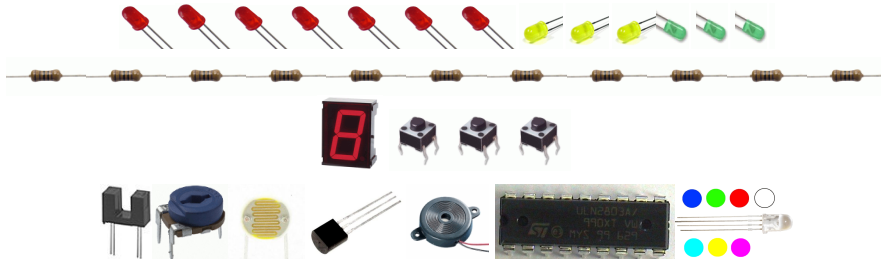


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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

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 :

1 à 2 Servomoteurs à rotation continue



Avec un servomoteur à rotation continue , **une seule broche de la carte Arduino suffit pour contrôler la vitesse et le sens du servomoteur, sans aucune autre interface !**

Dispose d'un connecteur type servomoteur dit « mâle » correspondant à l'association de 3 connecteurs femelle : broche numérique PWM servo / +5V / 0V

Couple : 3.2kg.cm en 4.8V

Consommation : 100mA env.

Alimentable directement par le +5V de la carte Arduino jusqu'à 3 servomoteurs. Au-delà, avec la carte Arduino, prévoir une alimentation externe régulée 5V (alim PC) ou non (Vin=6V).

Modèles possibles suggérés :

- un Futaba S3003 modifié dispo ici : <http://store.easyrobotics.fr/servomoteur/12-servomoteur-futaba-s3003.html> (choisir le modèle « rotation continue »)
- une paire de servomoteurs avec mini-roues dispo ici : <http://www.gotronic.fr/art-paire-de-servos-roues-s04nfwh-11504.htm>

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 !

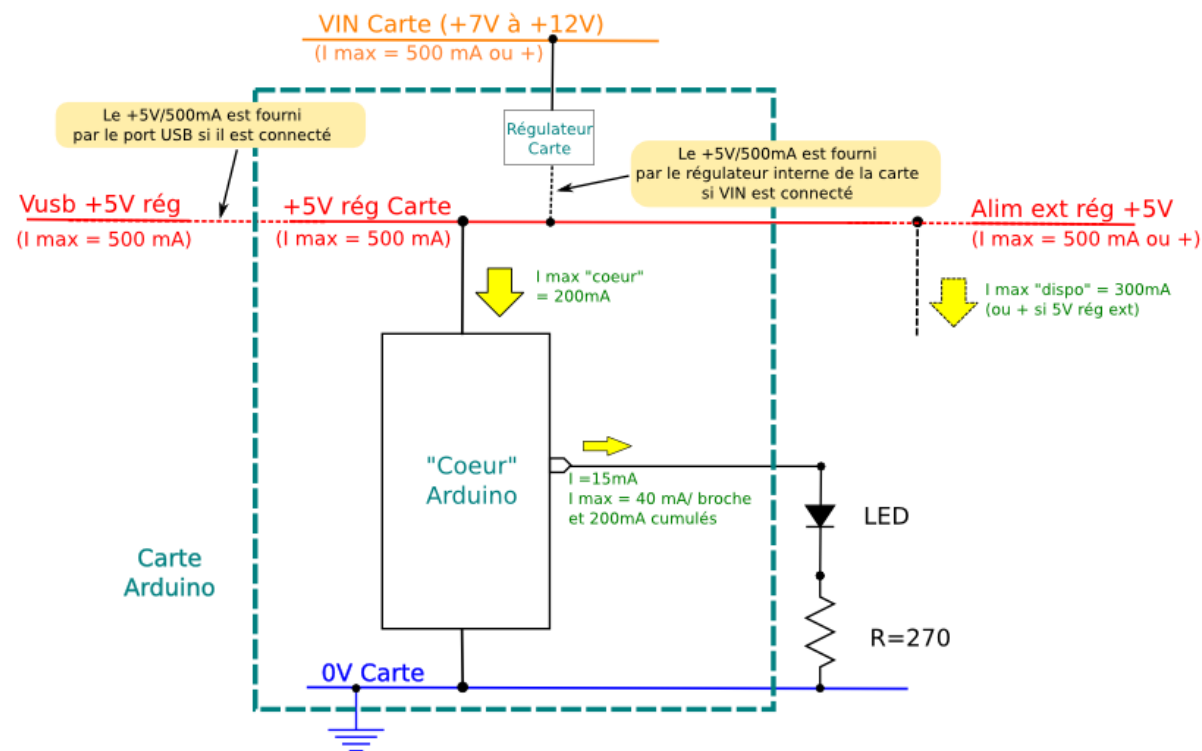
5. Rappel : Technique : l'alimentation de la carte Arduino

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 +)

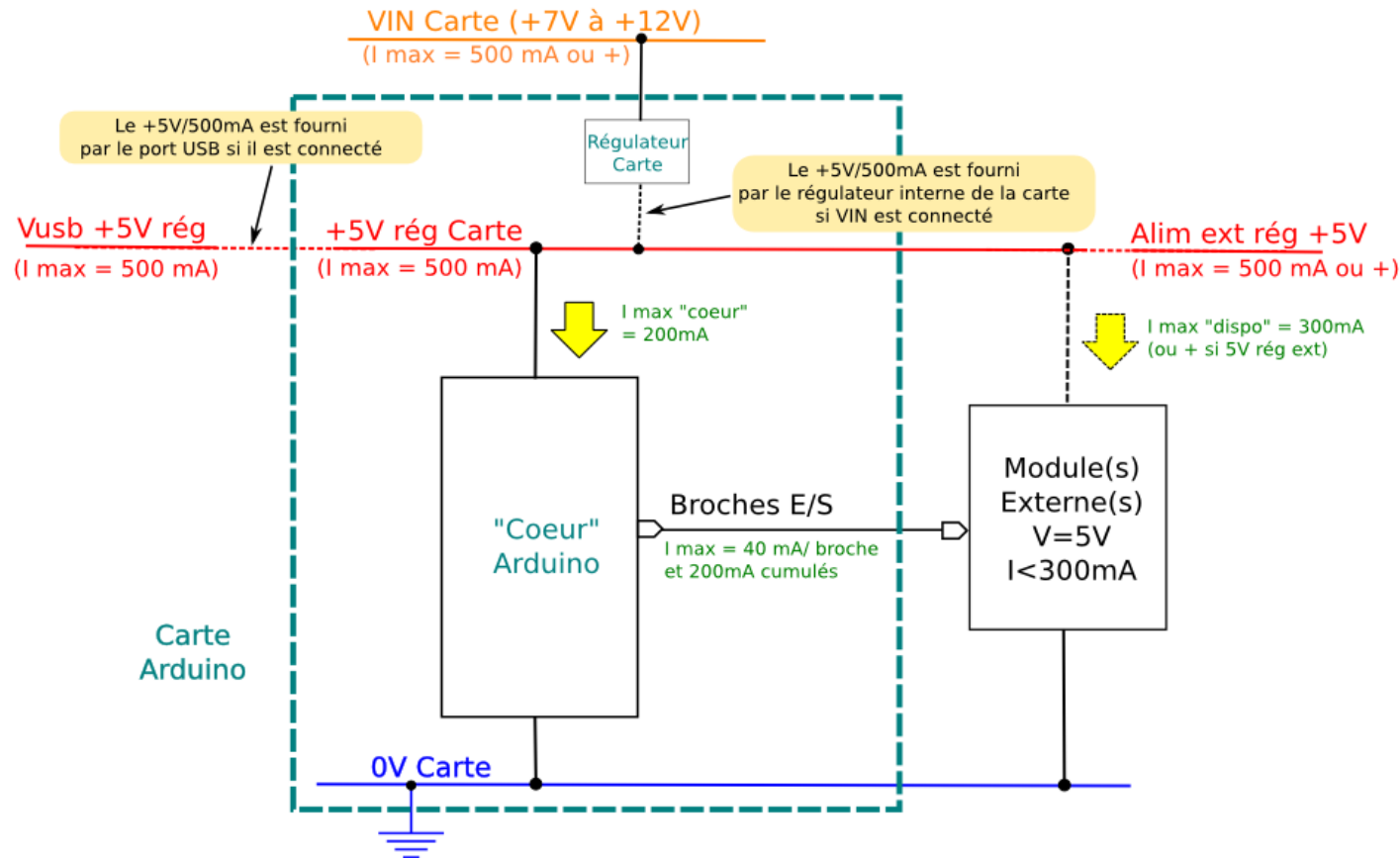
Il est essentiel de distinguer :

- **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. Rappel : Technique : utiliser un dispositif 5V / < 300mA avec la carte Arduino

Prenons à présent le cas d'un dispositif contrôlé par une ou plusieurs broches numériques et consommant moins de 300mA. On aura le schéma des alimentations suivant :

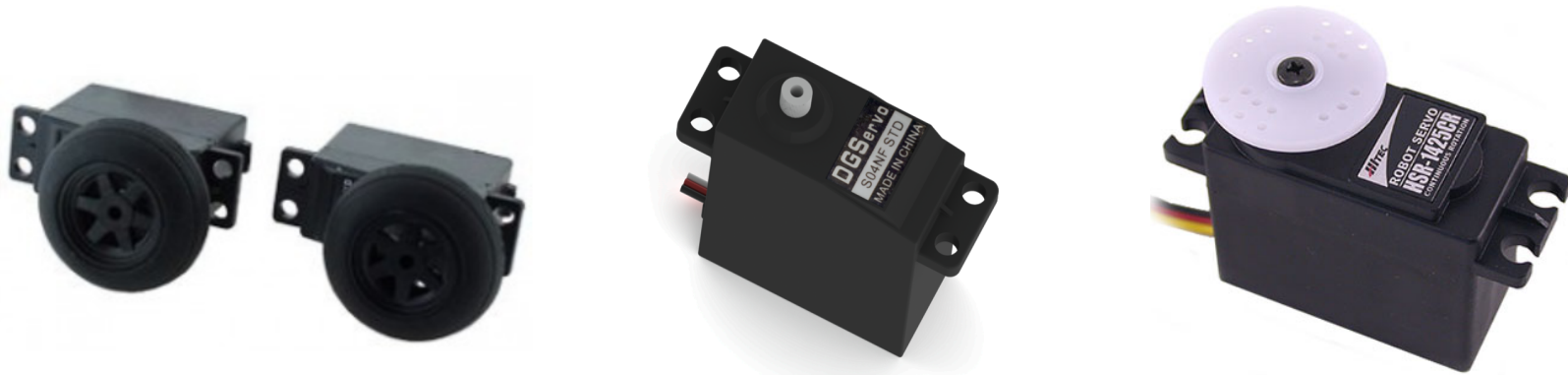


A retenir : Un dispositif utilisant une ou plusieurs broches numériques de contrôle (mais broches numériques uniquement !) et consommant 5V / <300mA pourra être alimenté directement par l'alimentation de la carte Arduino 5V/500mA.

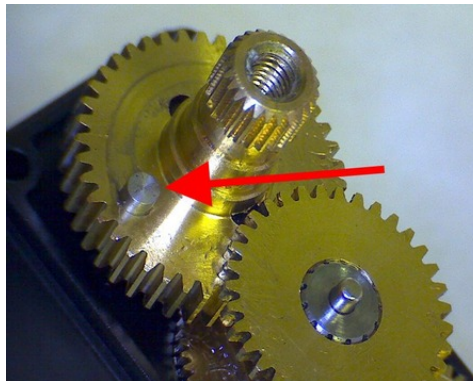
La bonne nouvelle : un servomoteur est contrôlé par une broche de type numérique (quelques mA) et consomme 100mA environ sur son alimentation principale. On pourra donc l'utiliser directement avec une carte Arduino ! (jusqu'à 3 voire 4 à la fois en pratique sans interface ni alimentation externe...)

7. Servomoteurs : les servomoteurs à rotation continue : concrètement

Les servomoteurs à rotation continue « prêts à l'emploi » sont plutôt rares bien que très pratiques !
Une référence utile, le So4NF de DGServo ou encore le HSR1425CR de Hitec parmi d'autres... Dispo chez www.gotronic.fr
Ou encore le Futaba S3003 modifié disponible chez <http://www.easyrobotics.fr/>



Certains servomoteurs standards peuvent être modifiés en servomoteurs à rotation continue...
Vous trouverez des tutoriels expliquant cela « à la pelle » sur internet. Attention tous les servomoteurs ne sont pas modifiables.



Pour découvrir tout le matériel existant autour des servomoteurs, un site en anglais où il y a tout : <http://www.servocity.com>

8. Servomoteurs : les servomoteurs à rotation continue : fiche technique.

Description

- Un servomoteur à rotation continue est comparable à un servomoteur classique : c'est un boîtier plastique contenant un moteur associé à une mécanique et une électronique internes contrôlant la rotation continue de l'axe.

Il est possible de transformer un servomoteur standard en servomoteur à rotation continue... mais ça existe aussi « tout prêt » !

Type de mouvement

- La différence majeure avec le servomoteur standard réside dans le type de mouvement de l'axe : ici, **l'axe tourne de façon continue en fonction de l'impulsion reçue par le servomoteur : le sens ET la vitesse de rotation sont contrôlés simplement par la largeur d'impulsion !**

Brochage

- Un servomoteur dispose d'un connecteur de 3 broches :
 - 2 broches d'alimentation +5V et 0V,
 - 1 broche de commande de type numérique.

Principe de contrôle du servomoteur

- Un servomoteur à rotation continue est contrôlé par 1 broche numérique par impulsion de type PWM : la largeur de l'impulsion va fixer **le sens ET la vitesse de rotation !**

Caractéristiques mécaniques

- Un servomoteur à rotation continue est caractérisée par sa vitesse de rotation maximale et son couple.

Caractéristiques électriques

- L'alimentation du servomoteur nécessite une tension entre 4,8 et 6V typiquement et une intensité de 100mA pour un modèle de base, voire beaucoup plus.
- La broche de commande du servomoteur est de type numérique (0V / +5V) et consomme quelques mA.

Maintien de position « hors tension »

- Un servomoteur est capable de maintenir raisonnablement une position bloquée « hors alimentation » (résistance mécanique modérée)

Codage

- Facile à mettre en oeuvre à l'aide de la librairie Servo du langage Arduino. Mise en position voulue en 1 ligne seulement !

Interface de « puissance »

- AUCUNE INTERFACE « de puissance » n'est nécessaire** pour une utilisation avec une carte Arduino, jusqu'à 3-4 servomoteurs (500mA maximum)
- Au-delà, utiliser simplement une alimentation externe adaptée.

Un shield avec connecteurs droits 3 broches sera pratique !

Principe d'alimentation

- Jusqu'à 3 voire 4 servomoteurs, utiliser le +5V/500mA de la carte Arduino pour alimenter les servomoteurs standards
- Au-delà de 4 servomoteurs, utiliser une alimentation 5V (régulée) externe capable de fournir une intensité suffisante en fonction du nombre de servomoteurs (alimentation de PC par exemple)

Prix unitaire

- Des modèles à moins de 10€ existent

Coût global unitaire de mise en oeuvre

- Jusqu'à 3 ou 4 servomoteurs standards, **uniquement le prix du servomoteur, soit moins de 12€ pour 1 servomoteur seul.** Au-delà, prix de l'alimentation externe en plus.

Utilisation type

- Les servomoteurs rotation continue sont très utiles pour réaliser un petit robot mobile roulant à moindre coût. Idéal pour initiation !

Avantages

- Faible de coût de mise en oeuvre
- Contrôle du sens de la rotation et de la vitesse avec une seule broche de la carte Arduino et sans interface !**

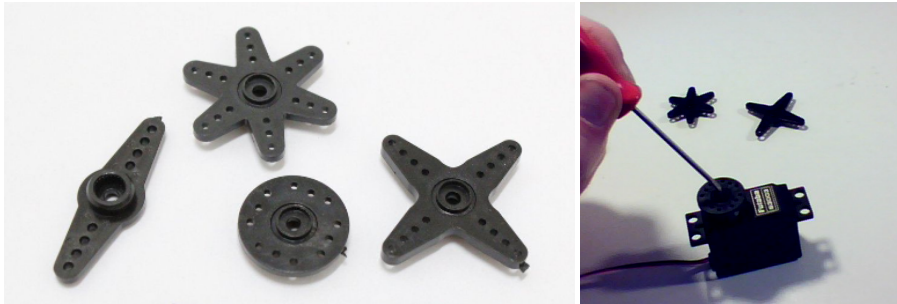
Inconvénients

- Peu de modèles au choix

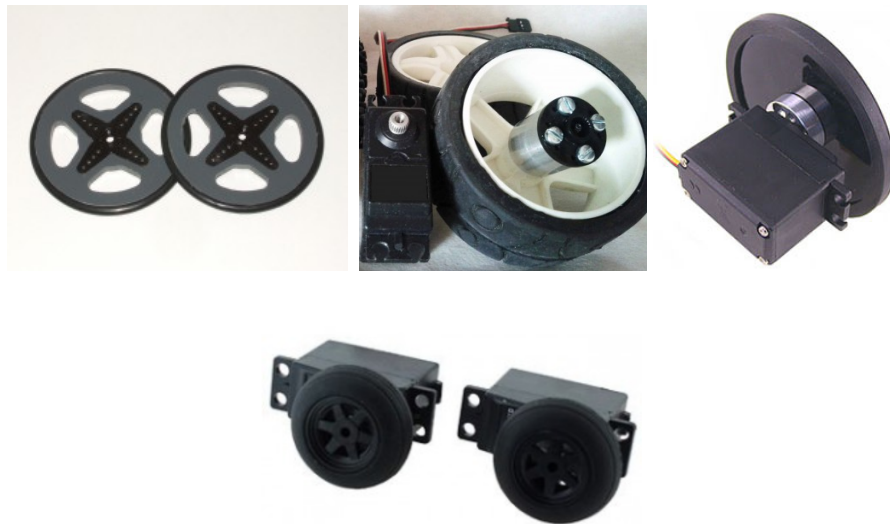
9. Infos techniques utiles pour les servomoteurs à rotation continue

Les roues

Les servomoteurs à rotation continue vont pouvoir être utilisés directement avec des roues. Pour fixer la roue sur le servomoteur, on pourra utiliser les palonniers standards fournis et à fixer sur l'axe mobile du servomoteur. La fixation se fait très simplement à l'aide d'une vis à visser dans l'axe :



Des modèles de roues existent prêtes à être utilisées avec un palonnier pour être fixée simplement sur le servomoteur. Des adaptateurs pour roues de modélisme existent aussi (voir notamment <http://store.easyrobotics.fr>) :



True pratique :

le palonnier tient correctement en place sans vis et il est souvent plus pratique de simplement l'enfiler sur l'axe sans le visser pour pouvoir l'enlever tout aussi facilement : c'est intéressant en phase de mise au point d'un robot par exemple.

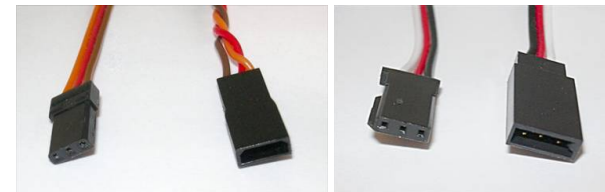
Accessoires de fixation

- Il existe également divers accessoires de fixations pour les servomoteurs, notamment les équerres, des supports équerres, ..
- Une mention spéciale pour un accessoire de fixation particulièrement intéressant : la **cage Easy** de chez EasyRobotics qui permet de fixer très facilement des servomoteurs à rotation continue sur un châssis.



Connectique

- Un servomoteur à rotation continue dispose d'un connecteur dits « mâle » 3 broches PWM / +5V / 0V (mais qui est en fait un triple connecteur « femelle »... ne pas se tromper quand on commande !)
- Il existe plusieurs types de connecteurs en fonction des fabricants : les connecteurs JR ou UNI et les connecteurs Futaba sont les 2 principaux (« mâle » à gauche et « femelle » à droite sur chaque photo) :



Connecteur JR

Connecteur Futaba

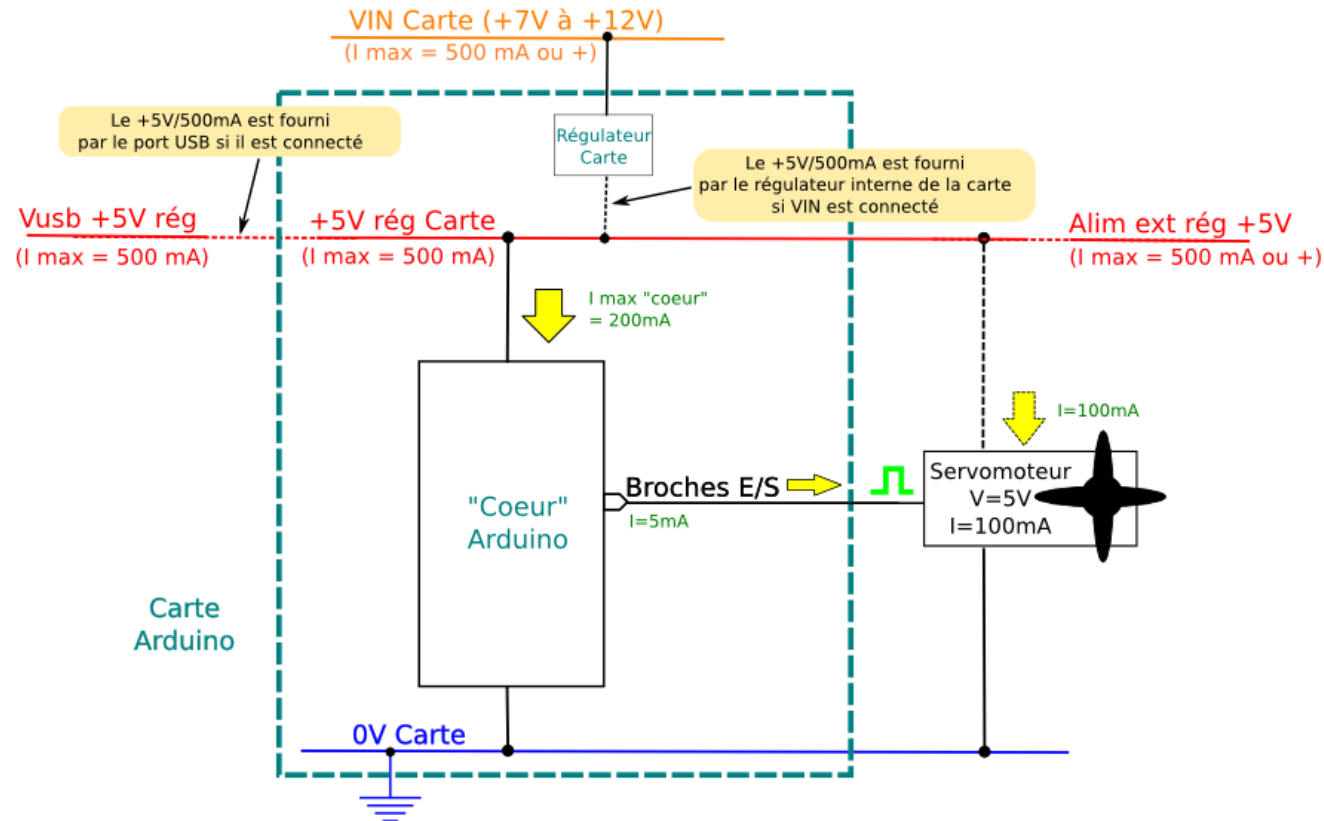
Bon à savoir : la façon la plus simple de connecter un servomoteur avec une carte Arduino passera par l'utilisation d'un connecteur droit pour CI - 3 broches et **qui sera utilisable indifféremment avec les connecteurs JR ou Futaba.**

Note : les servomoteurs Futaba sont utilisables avec les rallonges JR si on coupe le détrompeur latéral du connecteur Futaba, détrompeur qui ne sert à rien pour une utilisation avec Arduino.

10. Servomoteurs : les servomoteurs à rotation continue : schéma électrique type d'utilisation avec Arduino

Identique au servomoteur standard. On connecte :

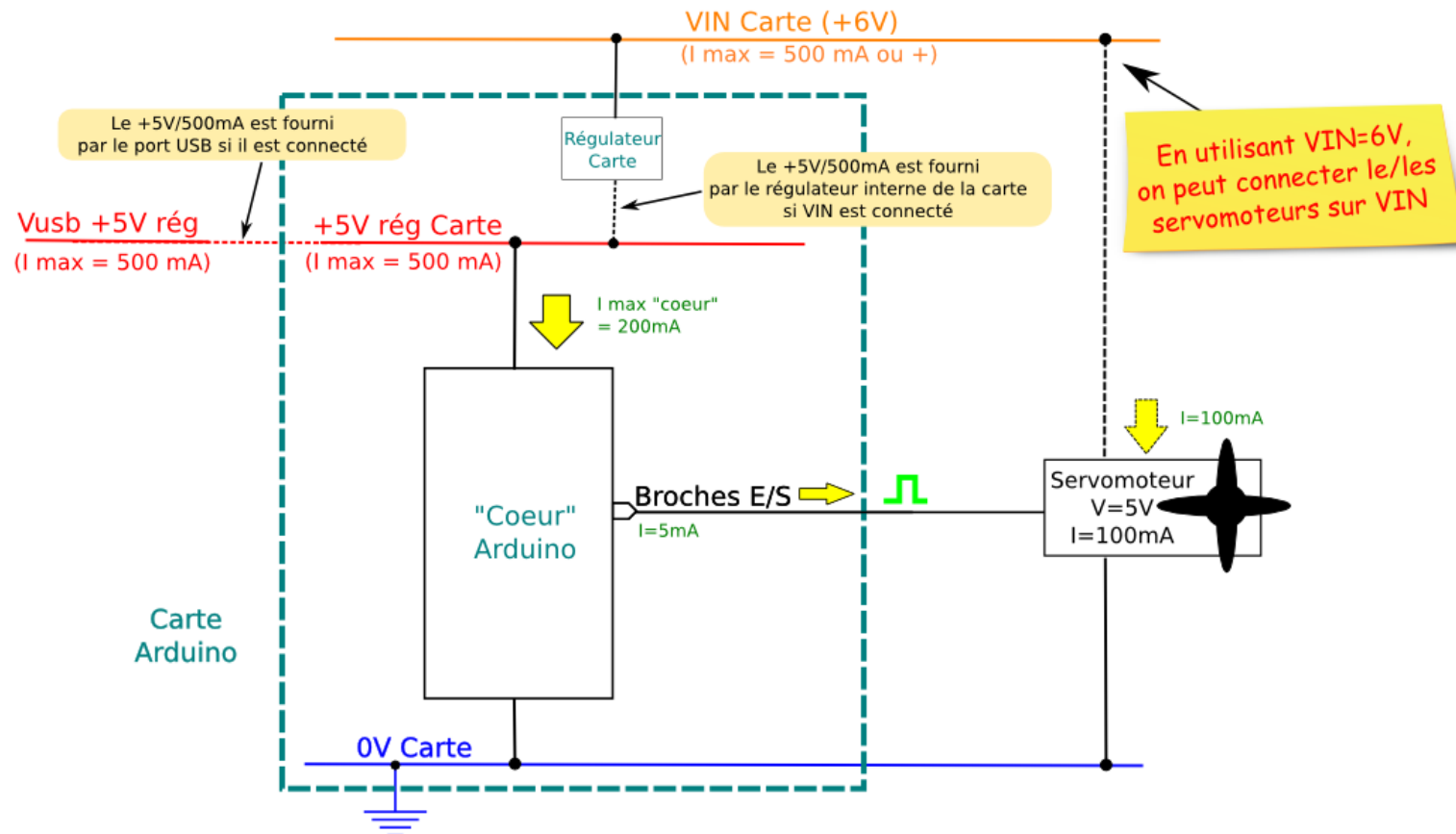
- le + (fil rouge) et le – (fil noir ou marron) respectivement au +5V et au 0V de la carte Arduino
- le fil de commande (fil blanc ou orange) à une broche numérique en sortie de la carte Arduino.



11. Servomoteurs : les servomoteurs à rotation continue : Variante schéma électrique type d'utilisation avec Arduino

- Pour éviter de multiplier les alimentations et si on utilise plus de 3 servomoteurs, sur un robot notamment, il est également possible d'alimenter directement les servomoteurs sur VIN si on utilise VIN de 6V (ou plus si les servomoteurs sont compatibles... mais les standards supportent 6V maxi).
- La carte Arduino devrait théoriquement être alimentée par VIN d'au moins 7V, mais à 6V, ça « passe »... Il devient ainsi possible d'alimenter les servomoteurs et la carte Arduino avec un paquets d'accus par exemple (**principe général de dimensionnement d'une alimentation type accu ou batterie : prévoir une capacité des accus à au moins 3 fois la consommation totale** des servomoteurs : par exemple 1500mA pour une conso de 500mA).

Cette variante sera utile si on utilise 4 servomoteurs ou plus... mais tant que l'on utilise que 2 à 3 servomoteurs, utiliser le 5V de la carte Arduino !

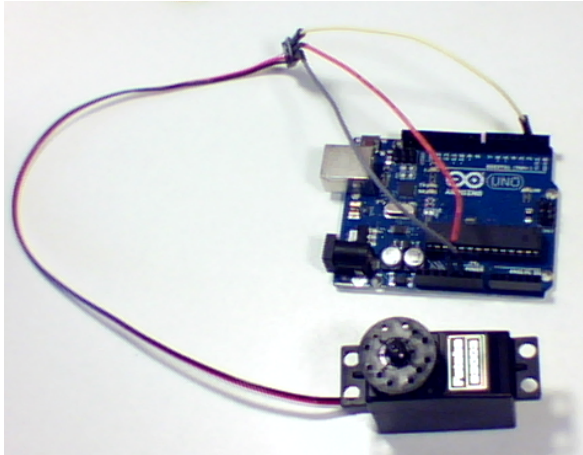


12. En pratique : utiliser un ou plusieurs servomoteurs avec une carte Arduino

Le plus simple pour commencer

Dans une première approche, la façon la plus simple d'utiliser un servomoteur avec une carte Arduino consiste à utiliser 3 jumpers mâle-mâle et à connecter :

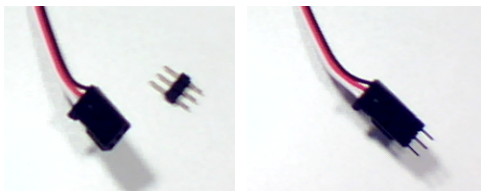
- la broche 0V (câble noir) du servomoteur au 0V de la carte Arduino (broches GND)
- la broche 5V (câble rouge) du servomoteur au +5V de la carte Arduino
- la broche de commande (câble blanc) du servomoteur à la broche numérique de la carte Arduino que l'on souhaite utiliser.



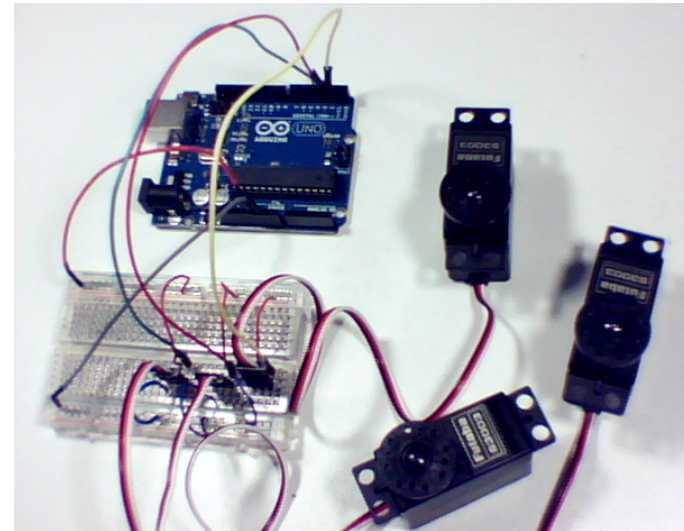
Utiliser plusieurs servomoteurs à l'aide d'une plaque d'essai

Dès que l'on va vouloir utiliser plusieurs servomoteurs, le même câblage va devoir être répété pour chaque servomoteur, ce qui complique un peu l'affaire et nécessite l'utilisation d'une plaque d'essai.

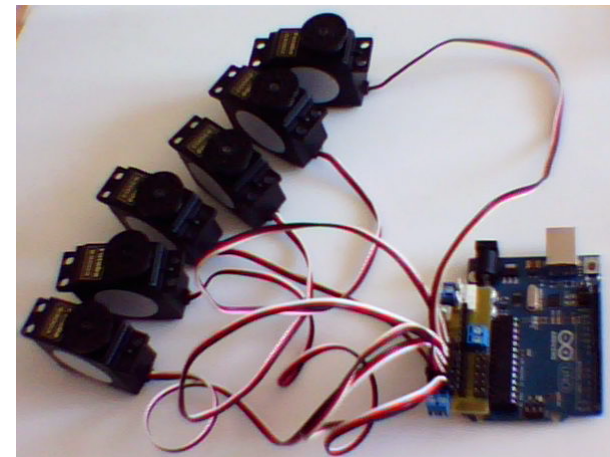
Pour que les choses soient plus simples, on commence par transformer les 3 broches femelles du connecteur du/des servomoteur(s) en 3 broches mâles à l'aide de connecteurs droits pour circuit imprimé en groupe de 3 contacts :



Une fois fait, la connexion sur la plaque d'essai devient assez facile et l'on réalisera le même câblage pour chaque servomoteur utilisé :



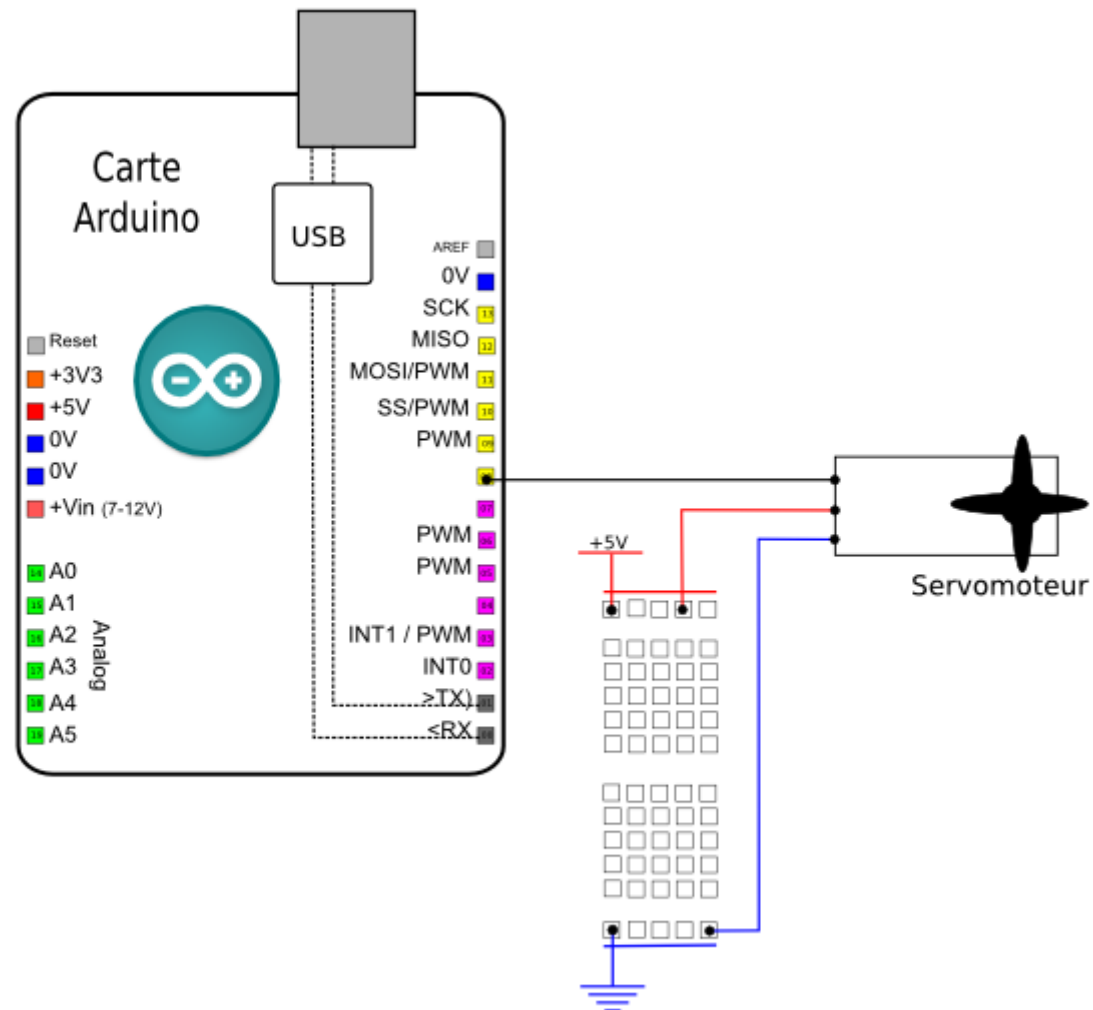
Lorsque le nombre de servomoteurs augmente, il sera plus pratique d'utiliser un shield qui dédoublera chaque broche de la carte Arduino sur un connecteur droit 3 contacts avec mise en parallèle du +5V et du 0V pour toutes les broches. On pourra ainsi utiliser très proprement 3, 6 ou 12 servomoteurs (utiliser une alimentation externe au-delà de 3 servomoteurs...) !



Exemple avec un mini-shield dont la fabrication est présentée sur www.mon-club-elec.fr

13. Servomoteurs : les servomoteurs à rotation continue : montage type avec une carte Arduino

Le montage est identique à celui d'un servomoteur standard



14. Servomoteurs : les servomoteurs à rotation continue : le principe de contrôle

Principe de contrôle du servomoteur

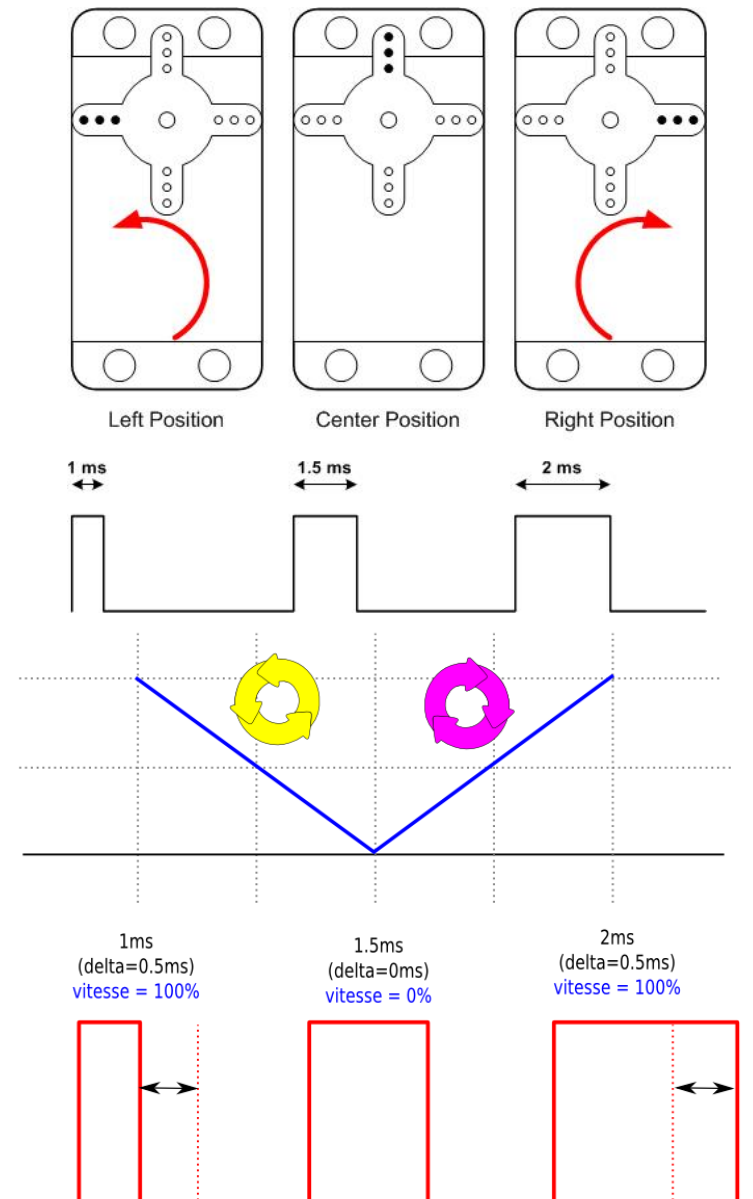
- Un servomoteur à rotation continue est contrôlé tout comme un servo standard par 1 broche numérique par impulsion de type PWM. Il existe cependant une différence notable : **la largeur de l'impulsion va fixer simultanément la vitesse de rotation ET le sens de rotation !**
- L'arrêt sera obtenu pour l'impulsion de « position » neutre.
- Le sens est déterminé par la largeur de l'impulsion par rapport à l'impulsion neutre :
 - $< 1,5\text{ms}$ ($1500\mu\text{s}$) = rotation dans un sens
 - $1,5\text{ms}$ ($1500\mu\text{s}$) pour la « position » médiane = arrêt
 - $> 1,5\text{ms}$ ($1500\mu\text{s}$) = rotation dans l'autre sens
- La vitesse variera dans un sens comme dans l'autre en fonction de l'écart entre la largeur de l'impulsion neutre et la largeur courante :
 - on aura 100% de la vitesse dans un sens à 1ms, 50% à 1,25ms, etc...
 - on aura 100% de la vitesse dans l'autre sens à 2ms, 50% à 1,75ms, etc...
- La période de cette impulsion PWM est de l'ordre de 20 ms.

Les valeurs données ici sont indicatives et peuvent varier selon le modèle de servomoteur utilisé : il faudra donc réaliser un test de position au préalable pour connaître la largeur de l'impulsion de la **position médiane d'arrêt**.

En pratique, avec Arduino

- Générer une telle impulsion est facile avec Arduino grâce à la **librairie Servo** dont nous allons voir l'utilisation :
 - Il sera ainsi possible de contrôler un servomoteur sur n'importe quelle broche numérique
 - et il sera possible de contrôler jusqu'à 20 servomoteurs si besoin !

Note : On n'utilisera pas l'instruction `analogWrite()` pour contrôler un servomoteur.

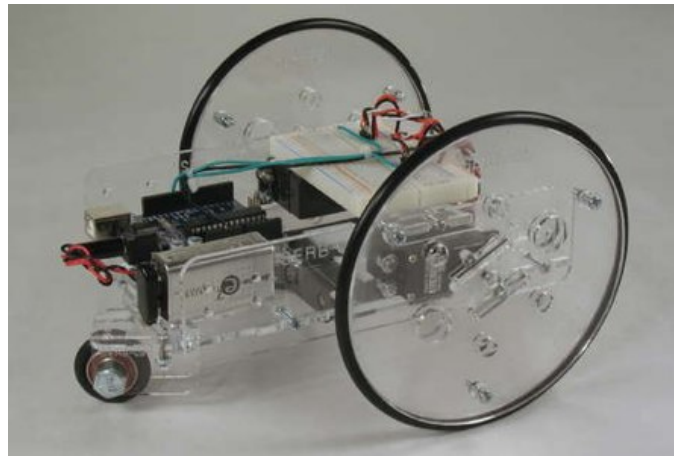


15. Servomoteurs : les servomoteurs à rotation continue : exemple d'utilisation

L'utilisation type des servomoteurs à rotation continue = réalisation d'un robot mobile d'initiation.

Le très grand avantage des servomoteurs à rotation continue :
permettre le contrôle de la vitesse ET du sens de rotation
avec une seule broche de la carte Arduino
et sans avoir besoin d'une interface supplémentaire !

On dispose ainsi d'une base ultra-simple et low-cost permettant de contrôler la marche avant / arrière et tourne droite / gauche.



16. Langage Arduino : Introduction aux librairies

C'est quoi une librairie Arduino ?

Le langage Arduino comporte de nombreuses instructions comme vous avez pu le constater, une quarantaine en tout. Ces instructions sont intégrées dans ce que l'on appelle le « noyau » ou « coeur » (core en Anglais) du langage Arduino. Ces instructions sont « générales » et servent souvent.

Le langage Arduino peut cependant être étendu à la demande avec des instructions dédiées à certaines applications particulières : afin de ne pas surcharger inutilement le « coeur », ces instructions spécifiques ont été intégrées dans des « paquets d'instructions » appelés librairies.

Comment ça marche ?

Par exemple, si on utilise un afficheur LCD, un servomoteur ou encore si l'on utilise un shield ethernet (réseau), on va intégrer dans notre programme la librairie dédiée correspondante.

Principe général d'utilisation

Pour intégrer une librairie dans un programme Arduino, c'est très simple : il suffit d'ajouter en début de programme une ligne de la forme :

```
#include <nomlibrairie.h> // librairie pour servomoteur
```

ATTENTION : l'instruction include est un peu particulière : la ligne commence par un # et il n'y a pas de point virgule de fin de ligne !

Ensuite, dans le code, au niveau de l'entête déclarative, là où vous déclarez vos variables, il va falloir déclarer un objet (une sorte de super variable) représentant la librairie. Cet objet est en fait une instance (= un exemplaire) d'une Classe (=le moule) qui regroupe les fonctions de la librairie. On a :

```
ClasseObjet monObjet; // déclare un objet
```

Généralement ensuite :

- au niveau de la fonction `setup()`, on initialise l'objet avec les paramètres voulus
- au niveau de la fonction `draw()`, on appelle les fonctions de la librairie sous la forme que vous connaissez déjà :

```
monobjet.fonction( param, param, ..);
```

Rappel : pour utiliser une fonction d'une classe du langage Arduino, on utilise le nom de la classe + un point + le nom de la fonction.

Il peut exister des variantes selon les librairies, mais grosso-modo, ça fonctionne de cette façon pour la plupart des librairies Arduino.

Vous avez déjà utilisé une librairie !

Si vous êtes attentifs à tout ce qu'on a déjà vu, vous me direz que ça ressemble étrangement à l'utilisation de la classe **Serial...** et vous aurez raison ! En fait, la classe Serial est une librairie qui est intégrée implicitement lorsque vous lancez Arduino : c'est pour ça que vous n'avez pas besoin d'utiliser **#include** pour l'utiliser.

Les librairies standards Arduino

Les librairies Arduino disponibles sont nombreuses, et disposent chacune de quelques fonctions à plusieurs dizaines... ce qui étend considérablement la puissance du langage Arduino et qui en fait aussi tout son intérêt. Voici la liste des librairies standards du langage Arduino ([le logiciel donne la liste...](#)) :

- [La librairie Serial](#) - pour les communications séries entre la carte Arduino et l'ordinateur ou d'autres composants
- [La librairie LCD](#) - pour l'utilisation et le contrôle d'un afficheur LCD alphanumérique standard.
- [La librairie Servo](#) - pour contrôler les servomoteurs.
- [La librairie Stepper](#) - pour contrôler les moteurs pas à pas (nécessite une interface de commande)
- [La librairie Ethernet](#) - pour se connecter à Internet en utilisant le module Arduino Ethernet
- [La librairie EEPROM](#) - référence - pour lire et écrire dans la mémoire EEPROM non volatile.
- [La librairie SD](#) - référence - pour utiliser une carte mémoire SD (utiliser des fichiers, stocker des données, ...)
- [La librairie SoftwareSerial \(Série Logicielle\)](#) - référence - pour communication série logicielle sur n'importe quelles broches de la carte Arduino
- [La librairie Wire / I2C](#) - référence - Interface "deux fils" (TWI/I2C) pour envoyer et recevoir des données sur un réseau de modules ou capteurs.
- [La librairie SPI \(Serial Peripheral Interface\)](#) - pour communication série avec des modules externes supportant le protocole SPI
- [Firmata](#) - pour communiquer avec des applications sur l'ordinateur utilisant un protocole série standard.

En jaune les plus utiles. Impressionnant non ? On les étudiera pas à pas...

Les librairies de la communauté

A côté de ces librairies standards, il existe toute une série de librairies proposées par les uns et les autres et qui concernent des matériels spécifiques, ou autre. Par exemple :

- [La librairie Keypad](#) - pour l'utilisation des claviers matriciels. ([hors référence](#))

Faites un tour ici pour voir ce qui existe : <http://arduino.cc/playground/Main/LibraryList>

17. Langage Arduino : la librairie **Servo** pour le contrôle des servomoteurs

Présentation

La librairie **Servo**, comme son nom l'indique, va servir à utiliser les servomoteurs ! Elle est utilisable aussi bien pour les servomoteurs standards, que pour les servomoteurs à rotation continue.

Cette librairie permet d'utiliser un servomoteur sur n'importe quelle broche de la carte Arduino !! (rappelez-vous : les impulsions PWM simples ne sont disponibles que sur 6 broches, donc ici c'est bien mieux !)

Inclusion

La librairie Servo s'intègre dans un programme avec la ligne (pas de ; !!) :

```
#include <Servo.h> // inclut la librairie Servo
```

Le constructeur de la classe

Le constructeur de la classe est le suivant :

```
Servo myservo; // déclare un objet servomoteur
```

Les fonctions de la librairie

La librairie dispose des fonctions suivantes :

- [attach\(\)](#) : attache le servomoteur à une broche
- [write\(\)](#) : positionne le servomoteur à l'angle voulu
- [writeMicroseconds\(\)](#) : génère une impulsion PWM de la largeur indiquée en microsecondes
- [read\(\)](#) : renvoie l'angle actuel du servomoteur
- boolean [attached\(\)](#) : renvoie true si servomoteur attaché à une broche
- [detach\(\)](#) : détache le servomoteur d'une broche

Utilisation type

- Inclusion de la librairie **Servo**
- Déclaration d'un ou plusieurs objets **Servo**
- Initialisation du servomoteur avec la fonction **attach()**
- Positionnement du servomoteur soit avec **write()**, soit avec **writeMicroseconds()**

La fonction **attach()**

Cette fonction est importante car elle initialise le servomoteur :

```
servo.attach(broche);  
servo.attach(broche, impuls_min, impuls_max);
```

Elle est étalonnée notamment à partir de 2 valeurs clés : les largeurs d'impulsion correspondant à 0° et à 180°, qu'on définira en début de programme.

```
// programme type utilisant un servomoteur
```

```
#include <Servo.h> // inclut la librairie Servo
```

```
Servo myservo; // déclare un objet représentant le  
servomoteur
```

```
void setup(){
```

```
    myservo.attach(9); // attache le servomoteur à la  
    broche 9
```

```
    myservo.write(90); // positionne le servomoteur à 90°  
    (position neutre)
```

```
}
```

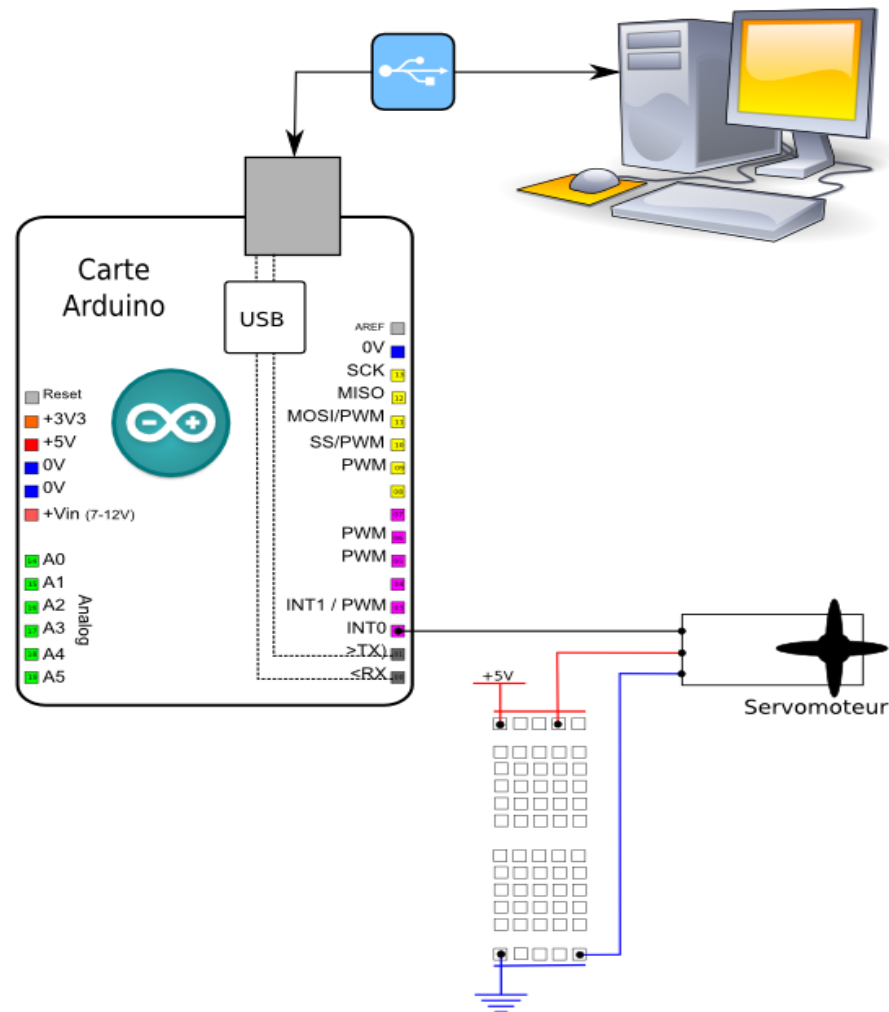
```
void loop() {
```

```
}
```

18. Calibrer un servomoteur à rotation continue via le port Série : le montage

Le montage est simple. On connecte :

- le **+** (**fil rouge**) et le **-** (**fil noir ou marron**) du servomoteur respectivement au **+5V** et au **0V** de la carte Arduino
- le **fil de commande** (fil blanc ou orange) à une **broche numérique en sortie** de la carte Arduino.
- La carte Arduino est par ailleurs connectée au PC pour la communication série



19. Calibrer la position d'arrêt d'un servomoteur à rotation continue via le port Série : le programme

On va reprendre un programme utilisé pour recevoir avec Arduino une valeur numérique en provenance du PC, en couplant à l'utilisation d'un servomoteur. Avec ce programme nous allons rechercher les largeurs d'impulsion correspondant :

- à la position d'arrêt ou « neutre », ou neutre,
- à la vitesse maximale sens Avant, ou maxAvant,
- à la vitesse maximale sens Arrière, ou maxArrière.

Ces 3 valeurs sont indispensables pour utiliser un servomoteur à rotation continue et peuvent varier selon les modèles.

Ce programme va vous permettre de les déterminer vous-mêmes : vous serez donc capable d'utiliser n'importe quel modèle de servomoteur à rotation continue.

Entête déclarative

- On commence par inclure la librairie **Servo**
- On déclare :
 - une variable **int** pour stocker l'octet en réception (code ASCII du caractère),
 - une variable **long** pour stocker le nombre reçu
 - une constante de broche pour le servomoteur
- On déclare également un objet **Servo**

```
//---- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo

//--- entete déclarative = variables et constantes globales
int octetReception=0; // variable de réception octet
long nombreReception=0; // déclare variable long stocker nombre reçu

const int brocheServo=2; // broche du servomoteur

Servo servo; // déclaration d'un objet servomoteur
```

Fonction **setup()**

- on initialise la communication série avec l'instruction **Serial.begin(vitesse)**. On utilisera 115200 bauds.
- on attache le servomoteur à la broche utilisée à l'aide de la fonction **attach()**

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

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

    servo.attach(brocheServo); // attache le servomoteur à la broche

} // 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 la valeur du nombre à la valeur courante x 10. *On réalise une petite pause entre 2 réceptions.*
- Et si c'est un saut de ligne que l'on reçoit :
 - on affiche le nombre reçu
 - on positionne le servomoteur
 - puis on sort de la boucle `while`.

Positionnement du servomoteur

- une fois la valeur reçue sur le port série obtenue, on contrôle la rotation du servomoteur à l'aide de la fonction `writeMicroseconds()` qui permet de fixer la largeur de l'impulsion PWM en micro-secondes.
- on place ce positionnement à l'intérieur de la gestion du saut de ligne pour que la nouvelle impulsion ne soit déclenchée que lorsqu'une nouvelle valeur est arrivée.

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

    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.print ("Nombre reçu = "); // affiche la le nombre reçu
            Serial.println (nombreReception);

            servo.writeMicroseconds(nombreReception);
            Serial.print ("Largeur impulsion servomoteur = ");
            Serial.print (nombreReception);
            Serial.println (" microsecondes");

            nombreReception=0; //RAZ le String de réception
            delay(10); // pause
            break; // sort de la boucle while
        } // fin if

        else { // si le caractère reçu n'est pas un saut de ligne

            octetReception=octetReception-48; // transfo valeur ASCII en
            valeur décimale

            // calcul du nombre à partir des valeurs reçues
            if ((octetReception>=0)&&(octetReception<=9)) nombreReception
            = (nombreReception*10)+octetReception;
            else Serial.println("La chaine n'est pas un nombre valide !");

            delay(1); // laisse le temps au caractères d'arriver

        } // fin else

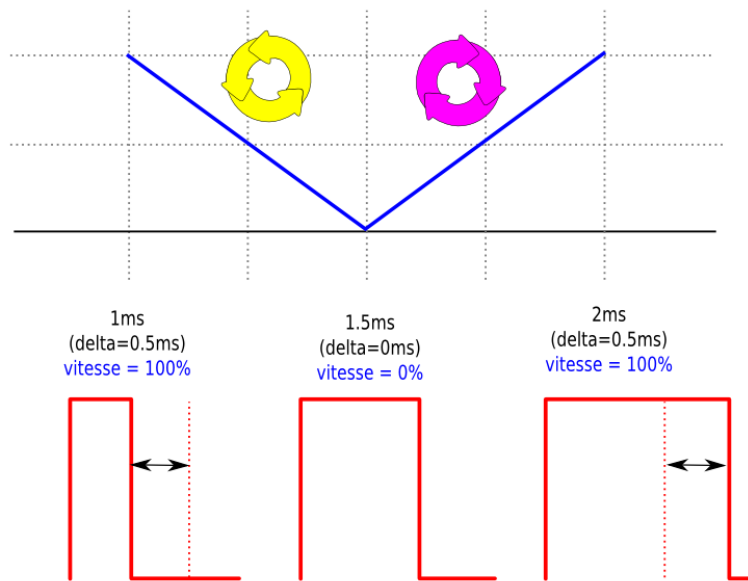
    } // fin while - fin de réception de la chaine

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

Note technique : il se peut que vous rencontriez des problèmes pour programmer la carte Arduino si le servomoteur tourne. Dans ce cas, débrancher la broche de commande et débrancher/rebrancher le câble USB, puis ré-essayer. D'une manière générale, mettez le servomoteur à l'arrêt avant de programmer la carte.

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 une valeur numérique en microsecondes (1000 pour 1 milliseconde) dans le champ de saisie et appui sur <send> : le servomoteur se met à tourner en conséquence. Essayer de trouver le neutre(arret), la vitesse maximale dans le sens positif et dans le sens négatif (**Voir page suivante pour plus de détails**).



```
/dev/ttyACM0
Saut de ligne recu : Nombre recu = 1480
Largeur impulsion servomoteur = 1480 microsecondes
Saut de ligne recu : Nombre recu = 1280
Largeur impulsion servomoteur = 1280 microsecondes
Saut de ligne recu : Nombre recu = 1230
Largeur impulsion servomoteur = 1230 microsecondes
Saut de ligne recu : Nombre recu = 1180
Largeur impulsion servomoteur = 1180 microsecondes
Saut de ligne recu : Nombre recu = 1380
Largeur impulsion servomoteur = 1380 microsecondes
Saut de ligne recu : Nombre recu = 1330
Largeur impulsion servomoteur = 1330 microsecondes
Saut de ligne recu : Nombre recu = 1430
Largeur impulsion servomoteur = 1430 microsecondes
Saut de ligne recu : Nombre recu = 1480
Largeur impulsion servomoteur = 1480 microsecondes
```

☒ Autoscroll Newline 115200 baud

Noter au passage que ce programme permet de générer une largeur d'impulsion à la micro-seconde près !

Vous êtes devenu un « as » de la précision, au millionième de seconde près !!



20. Technique : Méthode d'étalonnage d'un servomoteur à rotation continue

Ce que l'on veut connaître :

- Avant une première utilisation, la largeur d'impulsion du neutre (ou arrêt) et les largeurs d'impulsions des vitesses maximales dans chaque sens ne sont pas connues. Ce programme va permettre de les connaître avec précision.

Procédure à suivre

- Commencer par chercher le neutre** = la position d'arrêt. Avec un So4NF de DGServo, je trouve 1480µs, avec un servomoteur HK15138 modifié, j'obtiens 1430µs.
- Ensuite **chercher la vitesse maximale dans le sens négatif** en essayant des valeurs de plus en plus petites jusqu'à ce que la vitesse n'augmente plus. Avec un So4NF de DGServo, je trouve 1230µs, avec un servomoteur HK15138 modifié, j'obtiens 1130µs.
- Ensuite **chercher la vitesse maximale dans le sens positif** en essayant des valeurs de plus en plus grandes jusqu'à ce que la vitesse n'augmente plus. Avec un So4NF de DGServo, je trouve 1730µs, avec un servomoteur HK15138 modifié, j'obtiens 1730µs.

Remarques

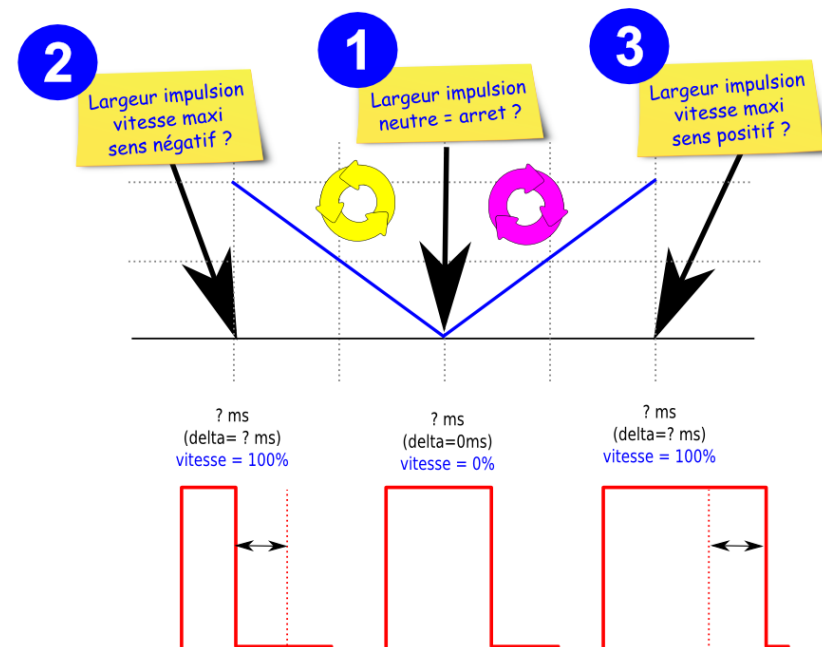
Vous constaterez d'une part que la **position neutre peut être légèrement fluctuante au fil du temps** et le moteur peut se mettre à tourner légèrement. Ce n'est pas très grave, ce problème sera corrigé par programmation comme nous allons le voir.

Vous devez utiliser au final des **valeurs centrées sur la valeur du neutre**. La précision n'est pas fondamentale, le centrage si.

Synthèse

- Une fois fait, on note ces valeurs (neutre, maxAvant et maxArrière)** et on les réutilisera ensuite dans les programmes utilisant ce servomoteur. **Voici un tableau avec quelques valeurs d'exemples obtenues :**

	vitesse Max sens -	Neutre	vitesse Max sens +
ex 1 : So4NF	1230 µs (-250)	1480 µs	1730 µs (+250)
ex 2 : HK15138	1130 µs (-300)	1430 µs	1730 µs (+300)
Le vôtre			



A noter que les servomoteurs d'un même modèle ont à priori les mêmes caractéristiques et n'ont pas besoin d'être étalonnés individuellement...

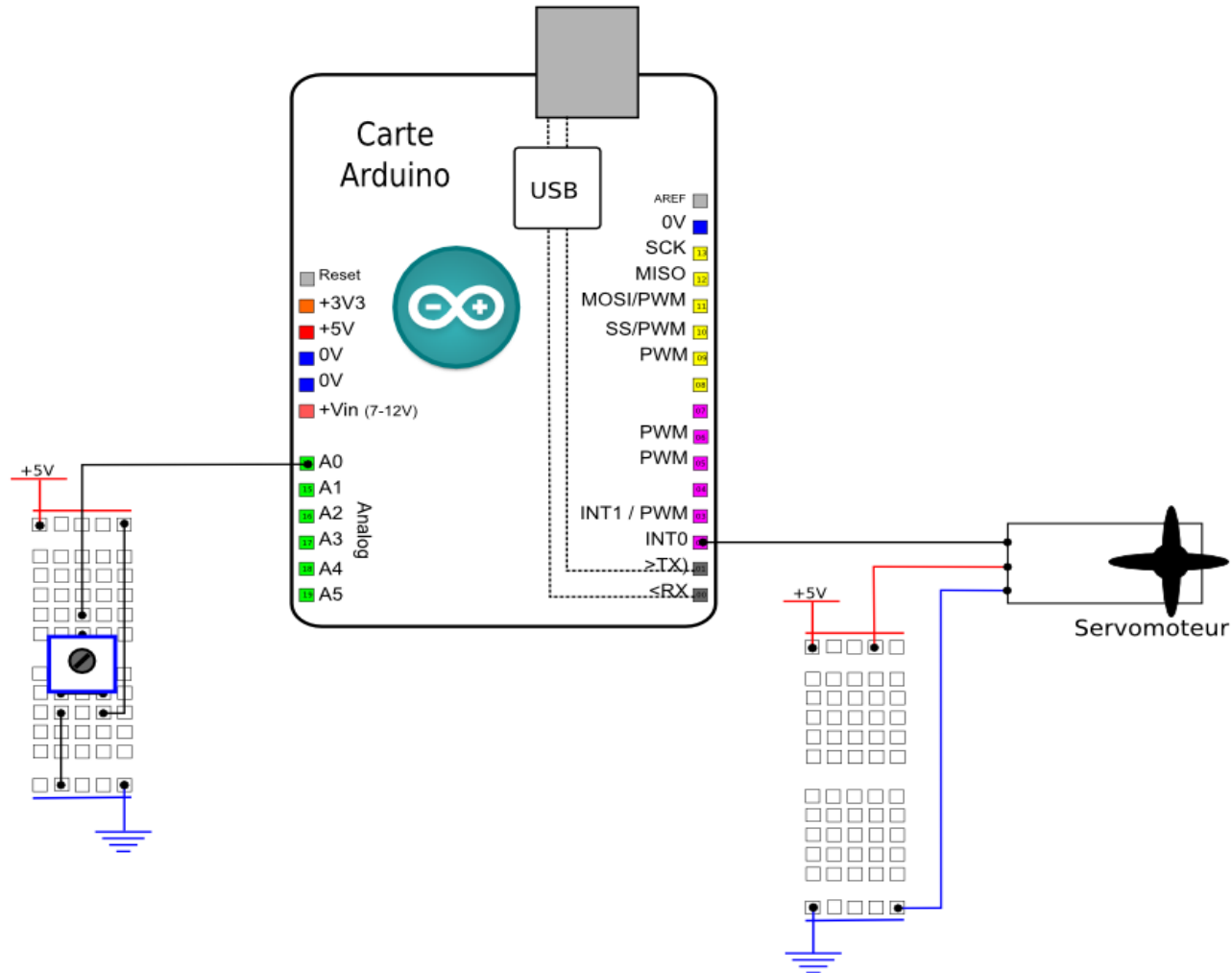


Une fois que vous connaissez ces 3 valeurs, votre servomoteur est étalonné et vous êtes prêt à l'utiliser ! Le plus dur est fait !

21. Contrôler un servomoteur à rotation continue à l'aide d'une résistance variable : le montage

Le montage est une fois de plus assez simple. On connecte :

- le + (**fil rouge**) et le – (**fil noir**) du servomoteur respectivement au **+5V** et au **0V** de la carte Arduino
- le **fil de commande** (fil blanc) à une **broche numérique en sortie** de la carte Arduino.
- la sortie d'une résistance variable (connectée entre 0V et +5V) sera connectée sur une broche analogique de la carte Arduino



22. Contrôler un servomoteur à rotation continue à l'aide d'une résistance variable : le programme

Nous allons donc reprendre la même structure de programme que celle vue précédemment, à la différence que nous n'utiliserons pas le port série.

Entête déclarative

- On commence par inclure la librairie **Servo**
- On déclare :
 - trois constantes **int** pour mémoriser les valeurs :
 - de la largeur d'impulsion de l'arrêt
 - de la largeur d'impulsion de la vitesse maxi en marche Avant
 - de la largeur d'impulsion de la vitesse maxi en marche Arrière
 - Initialiser ces constantes avec les valeurs obtenues précédemment en étalonnant votre servomoteur.
 - une constante de broche pour le servomoteur
 - une variable **int** pour stocker la largeur d'impulsion courante
 - une constante de broche analogique pour la résistance variable (en utilisant bien le numéro de broche analogique !)
 - une variable **int** pour mémoriser le résultat de la mesure
- On déclare également un objet **Servo**

Fonction **setup()**

- on attache le servomoteur à la broche utilisée à l'aide de la fonction **attach()** dans sa forme simple.

```
//--- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo

//--- entete déclarative = variables et constantes globales

const int neutre=1480; // largeur impulsion arrêt
const int maxAV=1730; // largeur impulsion vitesse max en avant
const int maxAR=1230; // largeur impulsion vitesse max en arriere

const int brocheServo=2; // broche du servomoteur
int impulsServo=0; // largeur impulsion servomoteur en usecondes
Servo servo; // déclaration d'un objet servomoteur

const int RVar=0; //declaration constante de broche analogique
int mesureBrute=0; // Variable pour acquisition résultat brut de
conversion analogique numérique
```

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

    servo.attach(brocheServo); // attache le servomoteur à la broche

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

Fonction `loop()`

Positionnement du servomoteur

- la première chose à faire est de mesurer la tension présente sur la broche analogique à l'aide de la fonction `analogRead()`, tension qui est fonction de la position de la résistance variable,
- ensuite, on convertit, à l'aide de la fonction `map()`, la valeur obtenue en la largeur d'impulsion voulue en se basant sur les deux valeurs des vitesses maximales dans les 2 sens correspondante de telle sorte que :
 - 0 mV correspond à la vitesse maximale dans un sens
 - 1023 mV correspond à la vitesse maximale dans l'autre sens

Remarquer qu'il suffit d'inverser les 2 valeurs `maxAR` et `maxAV` pour inverser l'effet de la résistance variable sur le sens de rotation du servomoteur. L'autre possibilité est également d'inverser les valeurs `maxAR` et de `maxAV` dans l'entête déclarative.

- 512mV (environ) correspondra par voie de conséquence à l'arrêt (impulsion médiane)
- on applique l'impulsion de largeur voulue sur le servomoteur à l'aide de la fonction `.write(angle)` de la classe `Servo`
- le programme boucle sans fin : tout changement de la valeur de la résistance variable se répercutera immédiatement sur la vitesse et le sens de rotation du servomoteur !

Fonctionnement du programme

- Tourner la résistance variable :
 - en position médiane, le servomoteur est à l'arrêt
 - vers la droite, le servomoteur tourne dans un sens de plus en plus vite,
 - vers la gauche, le servomoteur tourne dans l'autre sens de plus en plus vite...

Bravo !

Vous avez les bases pour contrôler un servomoteur à rotation continue simplement avec un potentiomètre : un bon début !

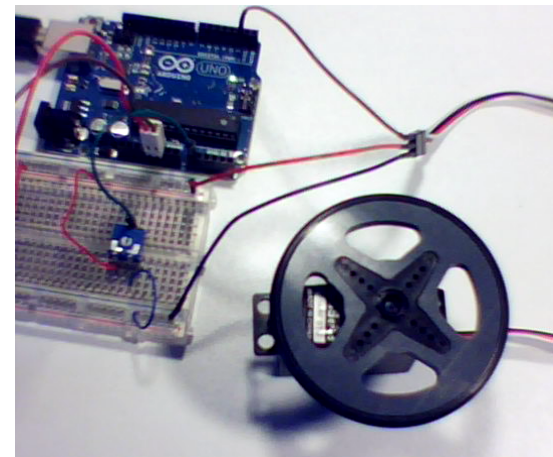
```
void loop() { //-- la fonction loop() : exécutée en boucle sans fin

    // acquisition conversion analogique-numerique (CAN) sur la voie
    analogique
    mesureBrute=analogRead(RVar);

    impulsServo=map(mesureBrute, 0, 1023, maxAR, maxAV); // calcul
    largeur impulsion correspondante
    servo.writeMicroseconds(impulsServo);

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

Remarquer une nouvelle fois la puissance du langage Arduino qui permet de réaliser cela en seulement 3 lignes de code !



23. Stopper ou faire tourner le servomoteur dans un sens ou dans l'autre à vitesse variable

Nous allons donc reprendre la même structure de programme que celle vue précédemment, sauf que nous n'utiliserons pas de résistance variable. Côté montage, connecter simplement le servomoteur sur la broche 2, sans utiliser de résistance variable.

Entête déclarative

- On commence par inclure la librairie **Servo**
- On déclare :
 - trois constantes **int** pour mémoriser les valeurs :
 - de la largeur d'impulsion de l'arrêt
 - de la largeur d'impulsion de la vitesse maxi en marche Avant
 - de la largeur d'impulsion de la vitesse maxi en marche Arrière
 - Initialiser ces constantes avec les valeurs obtenues précédemment en étalonnant votre servomoteur.
 - une constante de broche pour le servomoteur
 - une variable **int** pour stocker la largeur d'impulsion courante
- On déclare également un objet **Servo**

```
//--- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo

//--- entete déclarative = variables et constantes globales

const int neutre=1480; // largeur impulsion arrêt
const int maxAV=1730; // largeur impulsion vitesse max en avant
const int maxAR=1230; // largeur impulsion vitesse max en arriere

const int brocheServo=2; // broche du servomoteur

int impulsServo=0; // largeur impulsion servomoteur en µsecondes

Servo servo; // déclaration d'un objet servomoteur
```

Fonction **setup()**

- rien à ce niveau, le servomoteur sera attaché à la broche uniquement lorsque cela sera utile afin qu'il reste à l'arrêt par défaut.

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

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

Fonction `loop()`

Contrôle de la vitesse dans le sens positif

- on commence par attacher le servomoteur à la broche utilisée à l'aide de la fonction `attach()` la forme simple de cette fonction.
- ensuite, à l'aide d'une boucle `for`, on incrémente une variable de façon à défiler les pourcentages de 10% à 100%
- à chaque pas, on calcule la largeur de l'impulsion à utiliser à l'aide de la fonction `map()` qui va ré-échelonner la valeur en pourcentage en une valeur proportionnelle en microsecondes comprise entre la valeur neutre et la valeur de la vitesse maximale positive.
- on applique cette largeur d'impulsion au servomoteur à l'aide de la fonction `writeMicroseconds()`
- Une pause est appliquée entre chaque changement de vitesse

Mise à l'arrêt du servomoteur

- spontanément, à partir de tout ce qu'on a dit sur l'impulsion neutre qui met le servomoteur à l'arrêt, on aurait envie d'appliquer cette impulsion sur la broche pour stopper le servomoteur. Mais en pratique, ça ne marche pas bien : **la largeur d'impulsion du neutre est légèrement « fluctuant », ce qui entraîne une rotation légère du servomoteur....**
- **la seule façon de vraiment mettre le servomoteur à l'arrêt, c'est de ne plus appliquer aucune impulsion sur la broche en question.** Ceci se fait à l'aide de l'instruction `detach()` de la librairie `Servo` qui stoppe l'impulsion de commande pour le servomoteur en question. De cette façon, le servomoteur est parfaitement à l'arrêt !

Contrôle de la vitesse dans le sens négatif

- on reprend une séquence identique à celle utilisée pour la progression dans le sens positif mais ici, on se base sur la valeur de la vitesse maximale négative pour le ré-échelonnement avec `map()`

Mise à l'arrêt du servomoteur

- à nouveau on déconnecte le servomoteur avec `detach()` suivi d'une pause
- le programme boucle ensuite sans fin

C'est un petit peu plus compliqué qu'avec un servomoteur standard, mais la simplicité de contrôle du sens et de la vitesse par une seule broche compense largement les quelques lignes de codes supplémentaires...

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

    //-- marche avant progressive ----
    servo.attach(brocheServo); // attache le servomoteur à la broche

    for (int i=10; i<=100; i++) { // progression - en pourcentage %

        impulsServo=map(i, 0, 100, neutre, maxAV); // calcul largeur
        //impulsion AV correspondante
        servo.writeMicroseconds(impulsServo);
        delay(100); // entre 2 changements
    } // fin for

    //-- mise à l'arrêt réel du servomoteur
    servo.detach(); // détache le servomoteur de la broche = mise à
    //l'arrêt

    delay(2000); // pause

    //-- marche arriere progressive ----
    servo.attach(brocheServo); // attache le servomoteur à la broche

    for (int i=10; i<=100; i++) { // progression - en pourcentage %

        impulsServo=map(i, 0, 100, neutre, maxAR); // calcul largeur
        //impulsion AR correspondante en µs

        servo.writeMicroseconds(impulsServo);
        delay(100); // entre 2 changements
    } // fin for

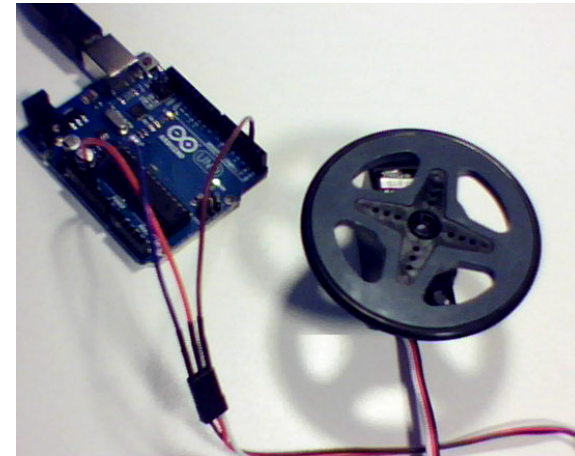
    //-- mise à l'arrêt réel du servomoteur
    servo.detach(); // détache le servomoteur de la broche = mise à
    //l'arrêt

    delay(2000); // pause

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

Fonctionnement du programme

- une fois la carte Arduino programmée, le servomoteur à rotation continue se met à tourner dans un sens de plus en plus vite puis après une pause, se met à tourner dans l'autre sens de plus en plus vite et ainsi de suite...



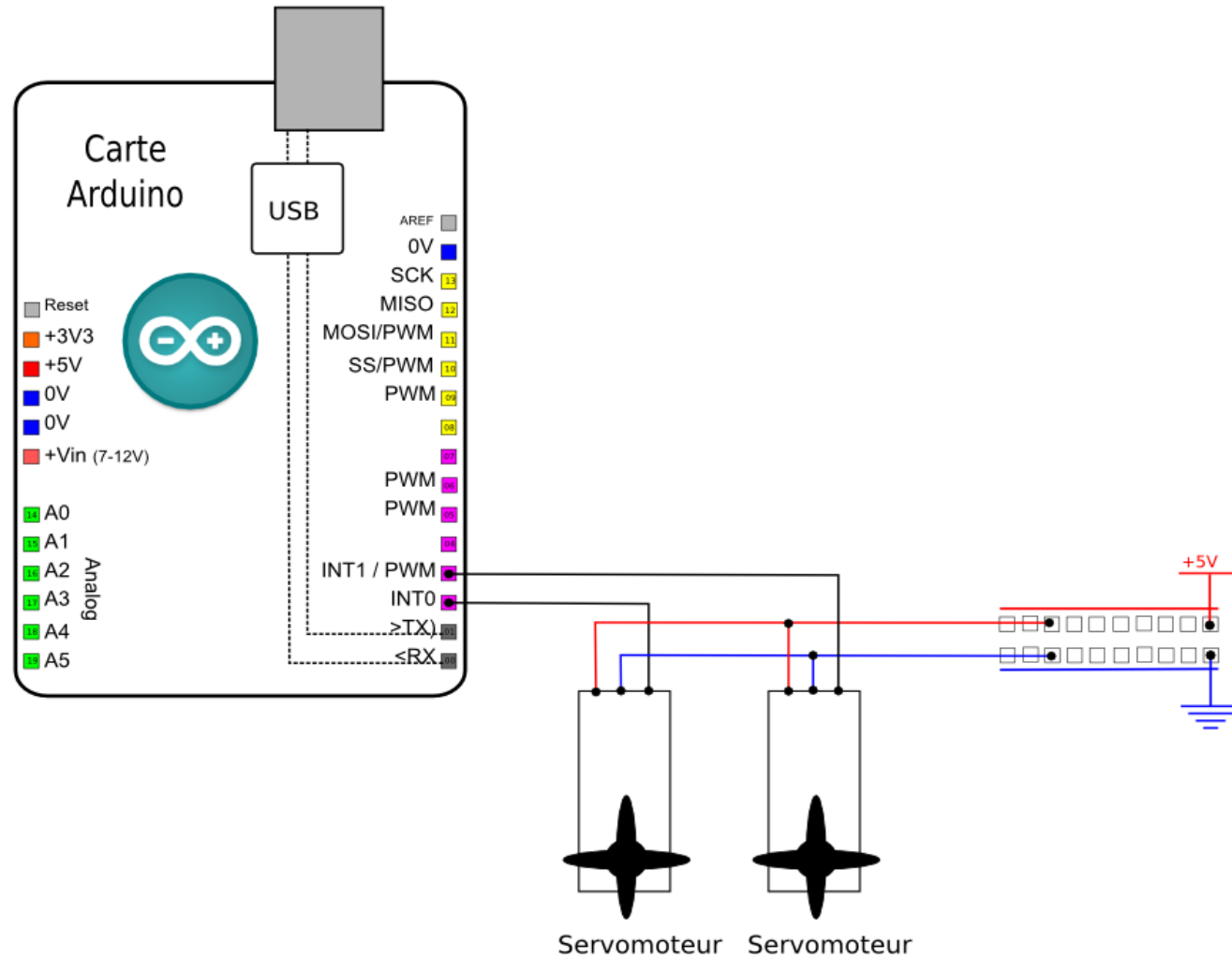
Sympa non ?

A ce stade, vous avez les bases pour réaliser un petit robot assez facilement. Dans les pages qui suivent, nous allons continuer nos explorations autour du servomoteur à rotation continue, mais concrètement, vous êtes déjà potentiellement opérationnel !



24. Contrôler 2 servomoteurs à rotation continue depuis le Terminal Série : le montage

- Le montage est relativement facile, même si les choses se compliquent un peu. Pour les 3 servomoteurs, on connecte :
 - le + (**fil rouge**) et le – (**fil noir ou marron**) du servomoteur respectivement au **+5V** et au **0V** de la carte Arduino
 - le **fil de commande** (fil blanc ou orange) à une **broche numérique en sortie** de la carte Arduino.



25. Contrôler 2 servomoteurs à rotation continue depuis le Terminal Série : le programme

Un des objectifs premiers à l'utilisation des servomoteurs à rotation continue est la réalisation d'un robot mobile d'initiation. Avec 2 servomoteurs à rotation continue, 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 2 servomoteurs, notamment la marche avant, la marche arrière, la bifurcation vers la droite ou vers la gauche et bien sûr l'arrêt ! **Le tout, rappelons-le, avec seulement 2 broches de la carte Arduino et aucune interface moteur !!** Dans ce programme, je vous propose de poser les bases du contrôle d'un tel robot à partir du Terminal Série. Non, non, vous ne rêvez pas... !

Entête déclarative

- On commence par inclure la librairie **Servo**
- 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.
- En ce qui concerne les servomoteurs, on déclare :
 - 2 constantes pour désigner les moteurs droit et gauche,
 - les constantes de la valeur neutre et des vitesses maximales sous forme d'un tableau (valeurs inversées pour chaque servomoteur)
 - un tableau de constantes des broches pour les servomoteurs
- On déclare également un tableau de 2 objets **Servo**

```
//--- Programme pour le controle de 2 servomoteurs à rotation continue
// par le Terminal Serie
// par X. HINAULT - Tous droits réservés - licence GPL v3 - www.mon-club-elec.fr

//--- entete déclarative = variables et constantes globales

//--- inclusion de librairie
#include <Servo.h> // inclut la librairie Servo

//--- 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

//--- variables et constantes pour les servomoteurs
const int Droit=0; // servo Droit a l'indice 0
const int Gauche=1; // servo Droit a l'indice 1

const int neutre=1480; // largeur impulsion arrêt
const int maxAV[2]={1730,1230}; // largeur impulsion vitesse max en avant
const int maxAR[2]={1230,1730}; // largeur impulsion vitesse max en arrière

const int brocheServo[2]={2,3}; // broches des servomoteur

//int impulsServo=0; // largeur impulsion servomoteur en µsecondes

Servo servo[2]; // déclaration d'un objet servomoteur
```

Fonction **setup()**

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() {

    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("arriereD");
    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`.

Remarquer ici comment il est très facile de créer son propre « code » de contrôle de la carte Arduino par l'association d'une chaîne à l'exécution d'une fonction donnée du programme : très simple et efficace !

Noter que l'on se contente ici de « passer » seulement des chaînes de caractères, mais il est également possible d'aller plus loin en « décodant » littéralement la chaîne reçue suivie de un ou plusieurs paramètres. Ceci sera traité dans un autre atelier consacré au port Série.



```
//--- la fonction loop() : exécutée 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") servoAvant(Droit, 100);

            if (chaîneReception=="avantG") servoAvant(Gauche, 100);

            if (chaîneReception=="arriereD") servoArriere(Droit, 100);

            if (chaîneReception=="arriereG") servoArriere(Gauche, 100);

            if (chaîneReception=="enAvant") enAvant(100);

            if (chaîneReception=="enArriere") enArriere(100);

            if (chaîneReception=="tourneD") tourneDroite(100);

            if (chaîneReception=="tourneG") tourneGauche(100);

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

            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 de la fonction loop()
```

Fonctions gérant les mouvements individuels des servomoteurs

Fonction de gestion du sens de rotation avant

- Cette fonction reçoit l'indice du servomoteur 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 :
 - attacher le servomoteur à la broche correspondante
 - à calculer la largeur de l'impulsion à utiliser à l'aide de la fonction `map()` qui va ré-échelonner la valeur en pourcentage en une valeur proportionnelle en microsecondes comprise entre la valeur neutre et la valeur de la vitesse maximale positive.
 - on applique cette largeur d'impulsion au servomoteur à l'aide de la fonction `writeMicroseconds()`
 - Une courte pause est appliquée.

Fonction de gestion du sens de rotation arrière

- Fonction identique avec comme différence l'utilisation de la valeur de la vitesse maximale arrière au lieu de la vitesse maximale avant au niveau de la fonction `map()`

```
//----- fonctions des mouvements de base des servomoteurs -----  
  
//----- fonction de controle de la vitesse avant d'un servomoteur  
void servoAvant(int indiceServo, int vitesseIn) {  
    // -- vitesse entre 10 et 100% --  
  
    int impulsServo; // variable locale  
  
    servo[indiceServo].attach(brocheServo[indiceServo]); // attache le  
    servomoteur à la broche  
  
    impulsServo=map(vitesseIn, 0, 100, neutre, maxAV[indiceServo]); //  
    calcul largeur impulsion AV correspondante  
    servo[indiceServo].writeMicroseconds(impulsServo);  
    delay(10); // pause  
  
} // fin servoAvant  
  
//----- fonction de controle de la vitesse arriere d'un servomoteur  
void servoArriere(int indiceServo, int vitesseIn) {  
    // -- vitesse entre 10 et 100% --  
  
    int impulsServo; // variable locale  
  
    servo[indiceServo].attach(brocheServo[indiceServo]); // attache le  
    servomoteur à la broche  
  
    impulsServo=map(vitesseIn, 0, 100, neutre, maxAR[indiceServo]); //  
    calcul largeur impulsion AV correspondante  
    servo[indiceServo].writeMicroseconds(impulsServo);  
    delay(10); // pause  
  
} // fin servoArriere
```

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 servomoteurs 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 servomoteurs 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 servomoteurs 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 servomoteurs à l'aide de la fonction `detach()`

```
//--- fonctions de mouvements synchrones des 2 moteurs

void enAvant(int vitesseIn) { // met les 2 moteurs en avant
// vitesse entre 10 et 100%

    servoAvant(Droit, vitesseIn); // moteur Droit
    servoAvant(Gauche, vitesseIn); // moteur Gauche

} // fin en avant

void enArriere(int vitesseIn) { // met les 2 moteurs en arriere
// vitesse entre 10 et 100%

    servoArriere(Droit, vitesseIn); // moteur Droit
    servoArriere(Gauche, vitesseIn); // moteur Gauche

} // fin en arriere

void tourneDroite(int vitesseIn) { // tourne à droite
// vitesse entre 10 et 100%

    servoArriere(Droit, vitesseIn); // moteur Droit
    servoAvant(Gauche, vitesseIn); // moteur Gauche

} // fin en tourne droite

void tourneGauche(int vitesseIn) { // tourne à gauche
// vitesse entre 10 et 100%

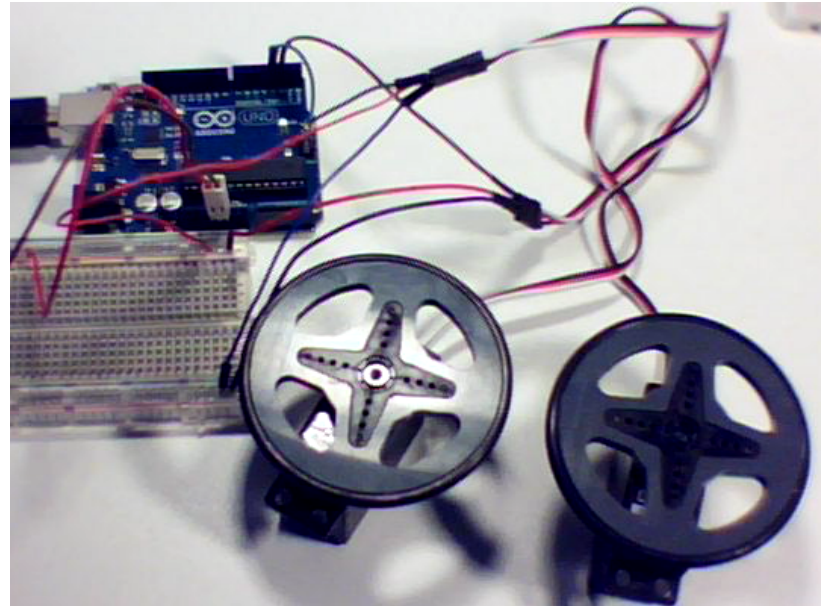
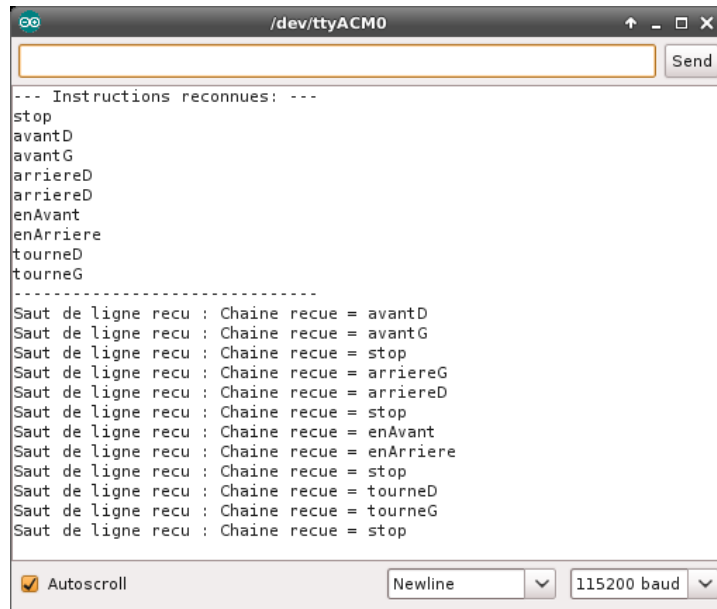
    servoAvant(Droit, vitesseIn); // moteur Droit
    servoArriere(Gauche, vitesseIn); // moteur Gauche

} // fin tourne gauche

void servosStop() { // arret des 2 moteurs
    servo[Droit].detach(); // met le servomoteur à l'arret
    servo[Gauche].detach(); // met le servomoteur à l'arret
}
```

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 servomoteurs à rotation continue !



Pour voir la vidéo associée à ce code et ce montage :

Bravo !

Si vous êtes arrivé jusqu'ici, c'est que vous êtes un vrai mordu !!

C'est un univers passionnant qui s'ouvre à vous !

26. Les éléments du langage Arduino étudiés dans cet atelier

Instructions du langage Arduino

Structure

- [#include](#)

Variables et constantes

Fonctions

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

Fonctions de la librairie **Servo**

Dans cet atelier, les fonctions de la librairie Servo ont été abordées :

- [attach\(\)](#)
- [write\(\)](#)
- [writeMicroseconds\(\)](#)
- int [read\(\)](#)
- boolean [attached\(\)](#)
- [detach\(\)](#)

La documentation de la librairie **Servo** en français est disponible ici :
http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.LibrairieServo

27. A présent, vous devriez être capable :

- d'utiliser des servomoteurs à rotation continue avec une carte Arduino afin d'être en mesure d'en contrôler la vitesse, le sens de rotation et de réaliser facilement un petit robot didactique.

Table des matières

Moteurs : Apprendre à utiliser des servomoteurs à rotation continue avec une carte Arduino.
Intro
Matériel nécessaire pour les ateliers Arduino
Matériel spécifique nécessaire pour cet atelier
Remarque
Rappel : Technique : l'alimentation de la carte Arduino
Rappel : Technique : utiliser un dispositif 5V / < 300mA avec la carte Arduino
Servomoteurs : les servomoteurs à rotation continue : concrètement
Servomoteurs : les servomoteurs à rotation continue : fiche technique.
Infos techniques utiles pour les servomoteurs à rotation continue
Servomoteurs : les servomoteurs à rotation continue : schéma électrique type d'utilisation avec Arduino
Servomoteurs : les servomoteurs à rotation continue : Variante schéma électrique type d'utilisation avec Arduino
En pratique : utiliser un ou plusieurs servomoteurs avec une carte Arduino
Servomoteurs : les servomoteurs à rotation continue : montage type avec une carte Arduino
Servomoteurs : les servomoteurs à rotation continue : le principe de contrôle
Servomoteurs : les servomoteurs à rotation continue : exemple d'utilisation
Langage Arduino : Introduction aux bibliothèques
Langage Arduino : la bibliothèque Servo pour le contrôle des servomoteurs
Calibrer un servomoteur à rotation continue via le port Série : le montage
Calibrer la position d'arrêt d'un servomoteur à rotation continue via le port Série : le programme
Technique : Méthode d'étalonnage d'un servomoteur à rotation continue
Contrôler un servomoteur à rotation continue à l'aide d'une résistance variable : le montage
Contrôler un servomoteur à rotation continue à l'aide d'une résistance variable : le programme
Stopper ou faire tourner le servomoteur dans un sens ou dans l'autre à vitesse variable
Contrôler 2 servomoteurs à rotation continue depuis le Terminal Série : le montage
Contrôler 2 servomoteurs à rotation continue depuis le Terminal Série : le programme
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