

## Moteurs :

**Apprendre à utiliser des servomoteurs standards 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 :

- de découvrir le principe de contrôle d'un servomoteur standard,
- d'apprendre à étalonner un servomoteur standard,
- d'apprendre à générer l'impulsion de contrôle d'un servomoteur standard,

... afin d'être en mesure d'utiliser des servomoteurs avec une carte Arduino.

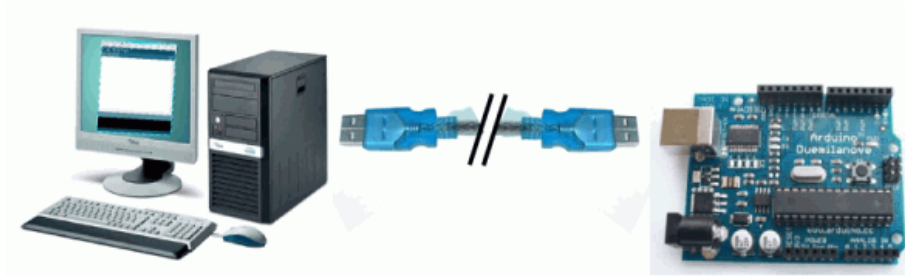


**Prêt ? C'est parti !**

## 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

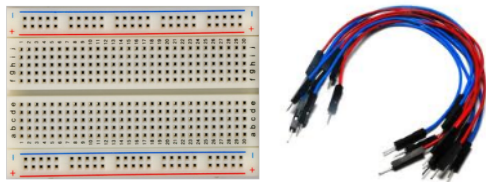


L'espace de développement Arduino associe :

- 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

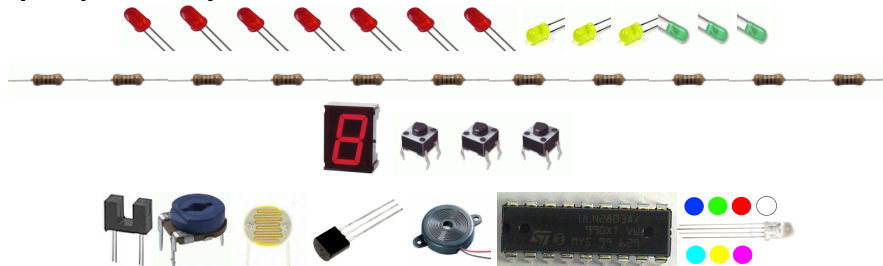


Pour réaliser des montages sans soudure, vous aurez besoin :

- 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 :

#### 1 à 2 Servomoteurs à rotation continue



Avec un servomoteur standard, **une seule broche de la carte Arduino suffit pour contrôler le 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 ici : <http://store.easyrobotics.fr/servomoteur/12-servomoteur-futaba-s3003.html>
- ou tout autre modèle de servomoteur standard...

#### 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 !

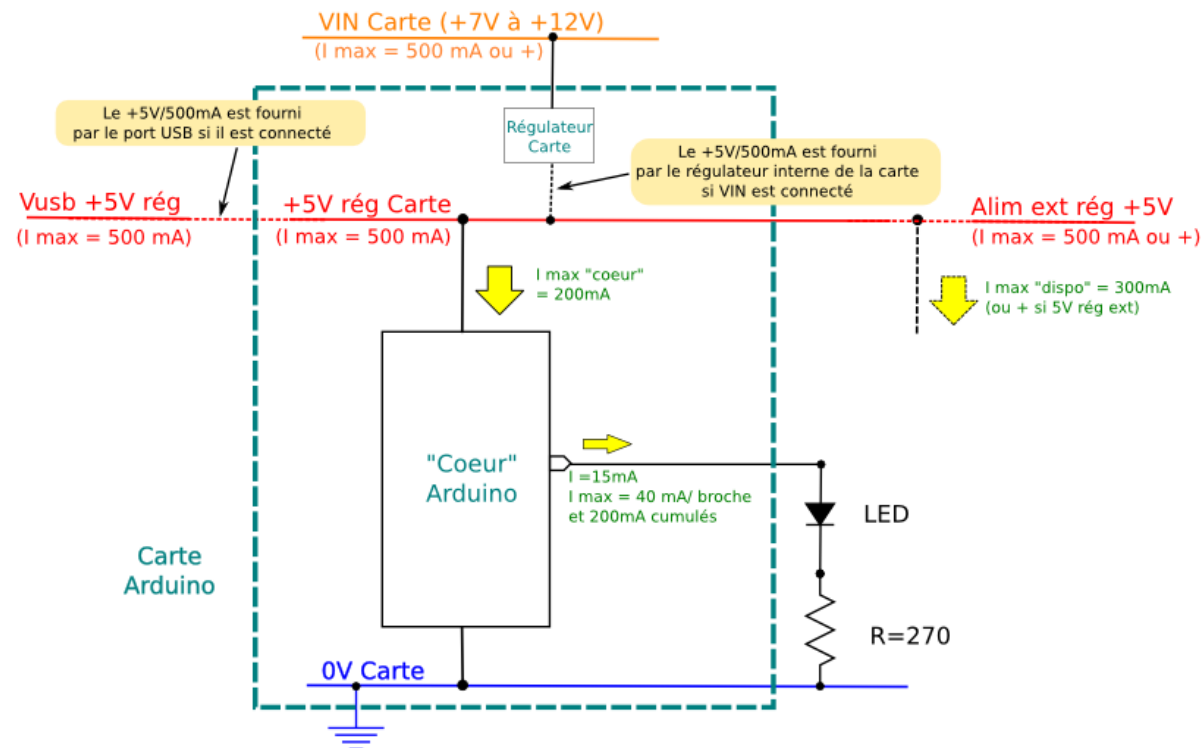
## 5. Technique : l'alimentation de la carte Arduino

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 +)

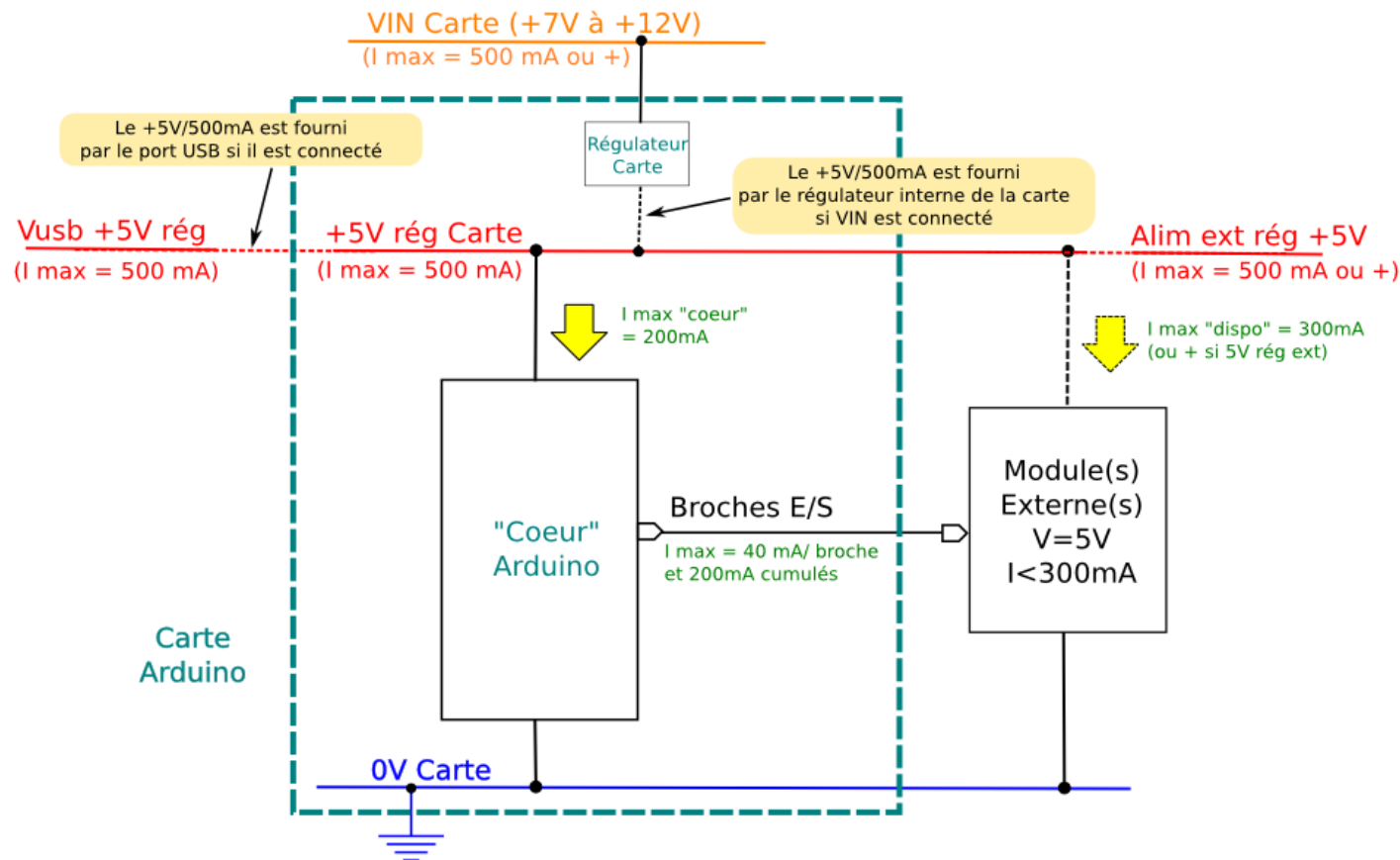
Il est essentiel de distinguer :

- **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. 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.

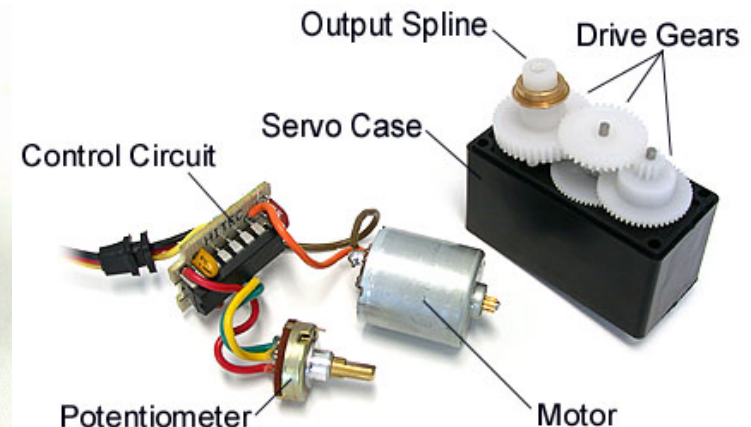
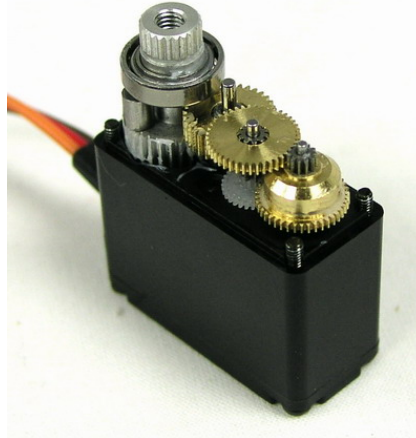
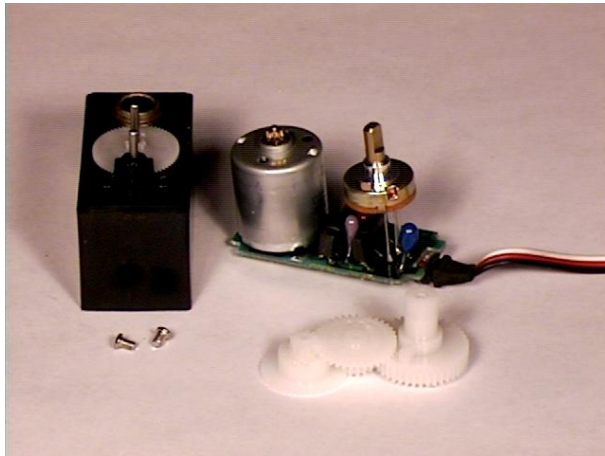
**La bonne nouvelle :** un servomoteur standard est contrôlé par une broche de type numérique et consomme 100mA environ. On pourra donc l'utiliser directement avec une carte Arduino ! (jusqu'à 3 voire 4 à la fois en pratique sans interface ni alimentation externe...)



## 7. Servomoteurs : les servomoteurs standards : concrètement



Servomoteur vu de l'intérieur (en pratique, on n'a pas besoin de le démonter... juste pour comprendre) :



### Quel modèle utiliser ?

Il existe de très nombreux modèles de servomoteurs standards, de caractéristiques variées. Les principaux fabricants sont Hitec, Graupner, Futaba... Dans une première approche, je conseille d'opter pour un servomoteur polyvalent et peu coûteux.

Le **Futaba S3003** répond à cette mission et vous le trouverez facilement à un prix avoisinant les 10€ / pièce, livré avec visserie et palonniers.

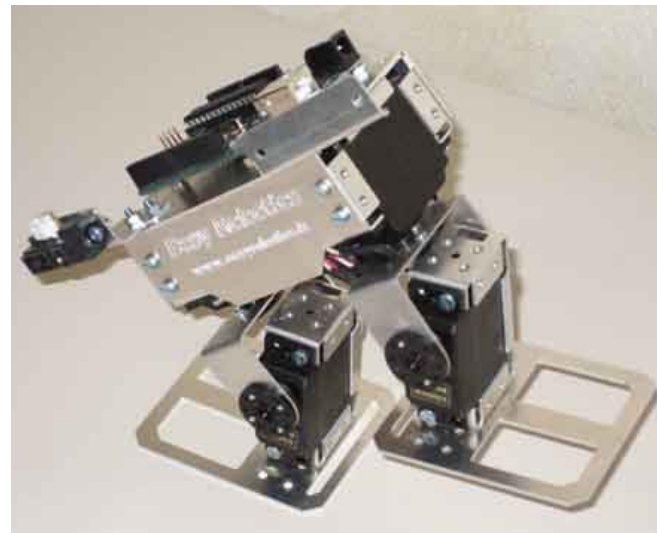
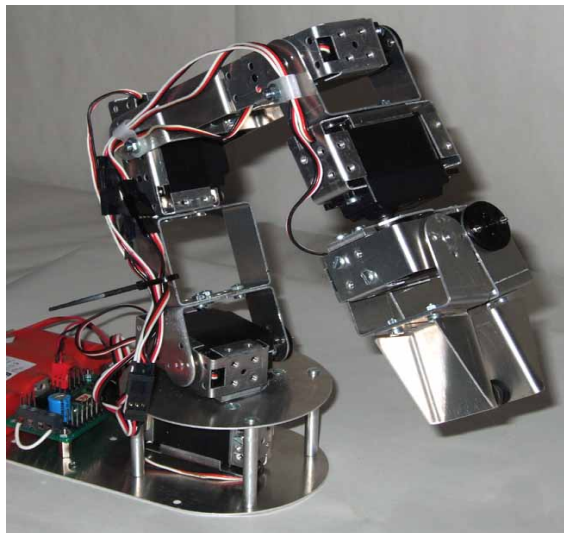
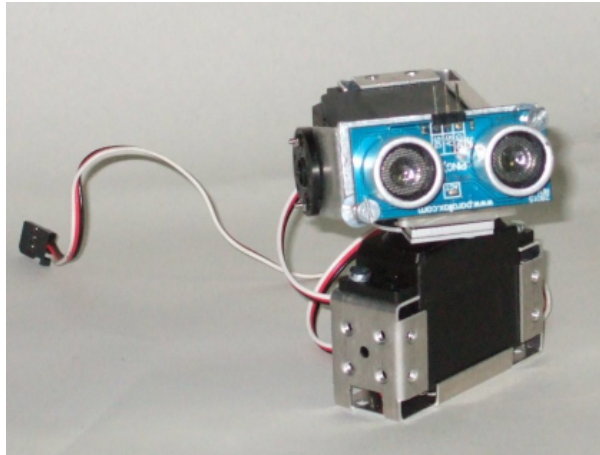
Pour découvrir tout le matériel existant autour des servomoteurs, un site en anglais où il y a tout : <http://www.servocity.com>



## 8. Servomoteurs : les servomoteurs standards : exemples d'utilisation

Voici quelques exemples d'utilisation de servomoteurs contrôlés avec Arduino et basés sur une mécanique de chez <http://www.easyrobotics.fr/>

# EasyRobotics



**Une carte Arduino associée à une simple alimentation adaptée peut permettre de contrôler jusqu'à 20 servomoteurs à la fois !**

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

Atelier Arduino : Moteurs : Apprendre à utiliser des servomoteurs standards avec une carte Arduino.

## 9. Servomoteurs : les servomoteurs standards : fiche technique.

### Description

- Un servomoteur est un boîtier plastique contenant un moteur associé à une mécanique et une électronique internes permettant d'assurer le maintien de position à l'angle voulu.

Typiquement, les servomoteurs sont utilisés en radio-modélisme.

### Type de mouvement

- Un servomoteur standard assure un **maintien de position dans un angle voulu** de l'axe du servomoteur (un peu à la manière d'un gouvernail d'un bateau...)

### 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 standard est **contrôlé par 1 broche numérique par impulsion de type PWM** : la largeur de l'impulsion va fixer la position de l'axe du servomoteur entre 0° ou 180° voire 360° selon les modèles.

### Caractéristiques mécaniques

- Un servomoteur est caractérisé par :
  - son couple (ex: 3.2 kg/cm)
  - et sa vitesse de positionnement (ex : 0.23 sec / 60°)

### 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 à l'usage, surtout si on utilise plusieurs servomoteurs !

### 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 (le S3003 de Futaba)

### Coût global unitaire de mise en oeuvre

- Jusqu'à 3 ou 4 servomoteurs standards, **uniquement le prix du servomoteur, soit moins de 10€ pour 1 servomoteur seul**. Au-delà, prix de l'alimentation externe en plus.

### Utilisation type

- Les servomoteurs standards sont très utiles pour toutes les mécaniques nécessitant des maintiens de position précis : tourelles pan/tilt (pour capteur, webcam, etc..), pinces, bras robotisé, robot bipède, robot hexapode, etc...

### Avantages

- Faible de coût de mise en oeuvre comparativement à des moteurs CC
- Existe en toute petite taille

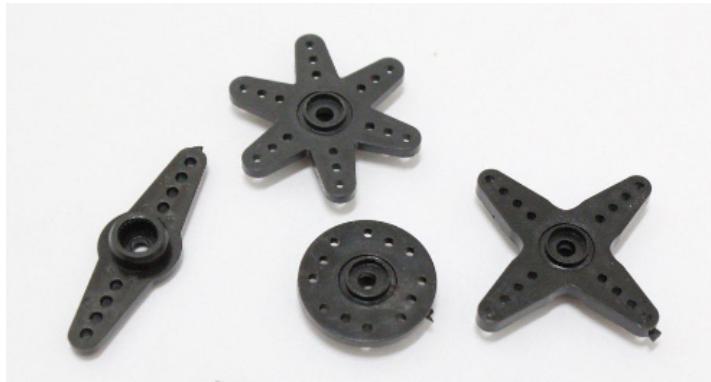
### Inconvénients

- Pas toujours facile à fixer. **Très simple cependant avec une cage métallique Easy** : <http://store.easyrobotics.fr/pièces/11-cage.html>

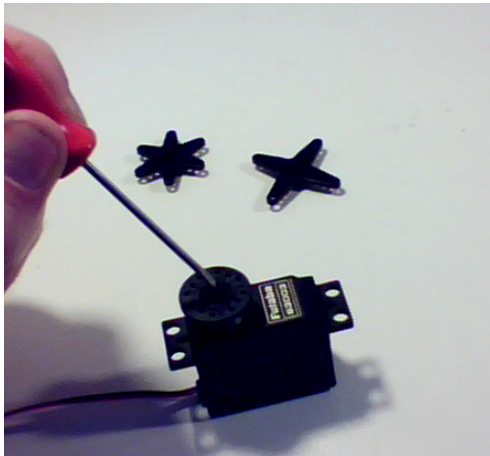
## 10. Infos techniques utiles pour les servomoteurs standards

### Palonniers

Les servomoteurs standards sont fournis avec différents types de palonniers 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 :



#### Truc 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, notamment pour caler le palonnier en fonction de la position angulaire de l'axe du servomoteur.

### Accessoires pour pignon

Il existe de nombreux accessoires de modélisme prévu pour être utilisés avec les servomoteurs standards, notamment chape, tringle filetée, articulation, etc... A adapter au cas par cas. Juste retenir que ça existe.

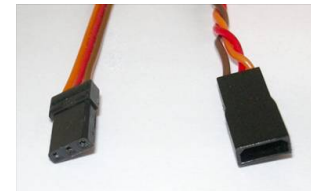


### Accessoires de fixation

- 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 <http://www.easyrobotics.fr/> qui permet un maximum de fixations et de combinaisons des servomoteurs entre eux (voir page suivante).

### Connectique

- Un servomoteur standard 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



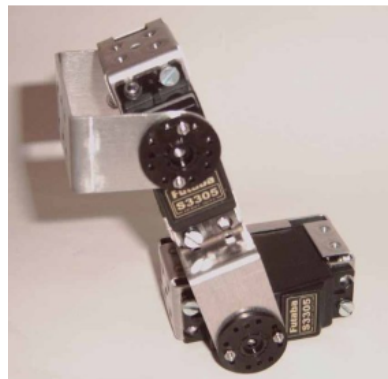
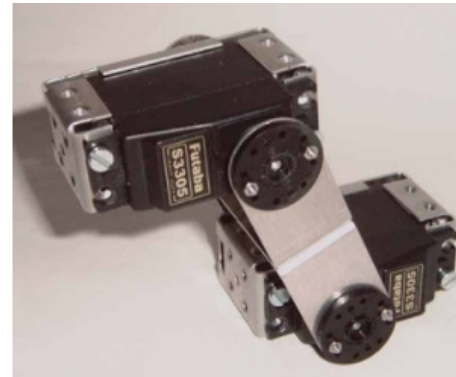
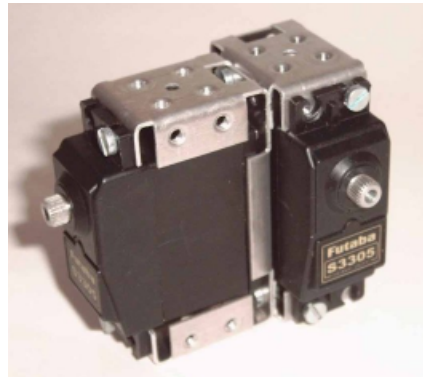
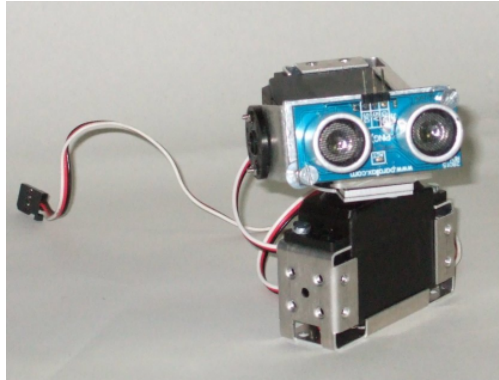
Connecteur Futaba

**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 « mâle » Futaba, détrompeur qui ne sert à rien pour une utilisation avec Arduino.

## 11. Infos techniques utiles pour les servomoteurs : combiner des servomoteurs facilement !

- Une mention spéciale pour un accessoire de fixation particulièrement intéressant : la **cage Easy** de chez [EasyRobotics](http://EasyRobotics.com) , low-cost (3€ à 4€ environ la cage), en aluminium, et qui permet un maximum de fixations (trous filetés au pas de vis 3mm) et de combinaisons des servomoteurs entre eux. Très pratique pour se fabriquer toutes sortes de robots et mécaniques. De la tourelle pan/tilt au robot hexapode en passant par le bras robotisé avec pince : tout est possible !



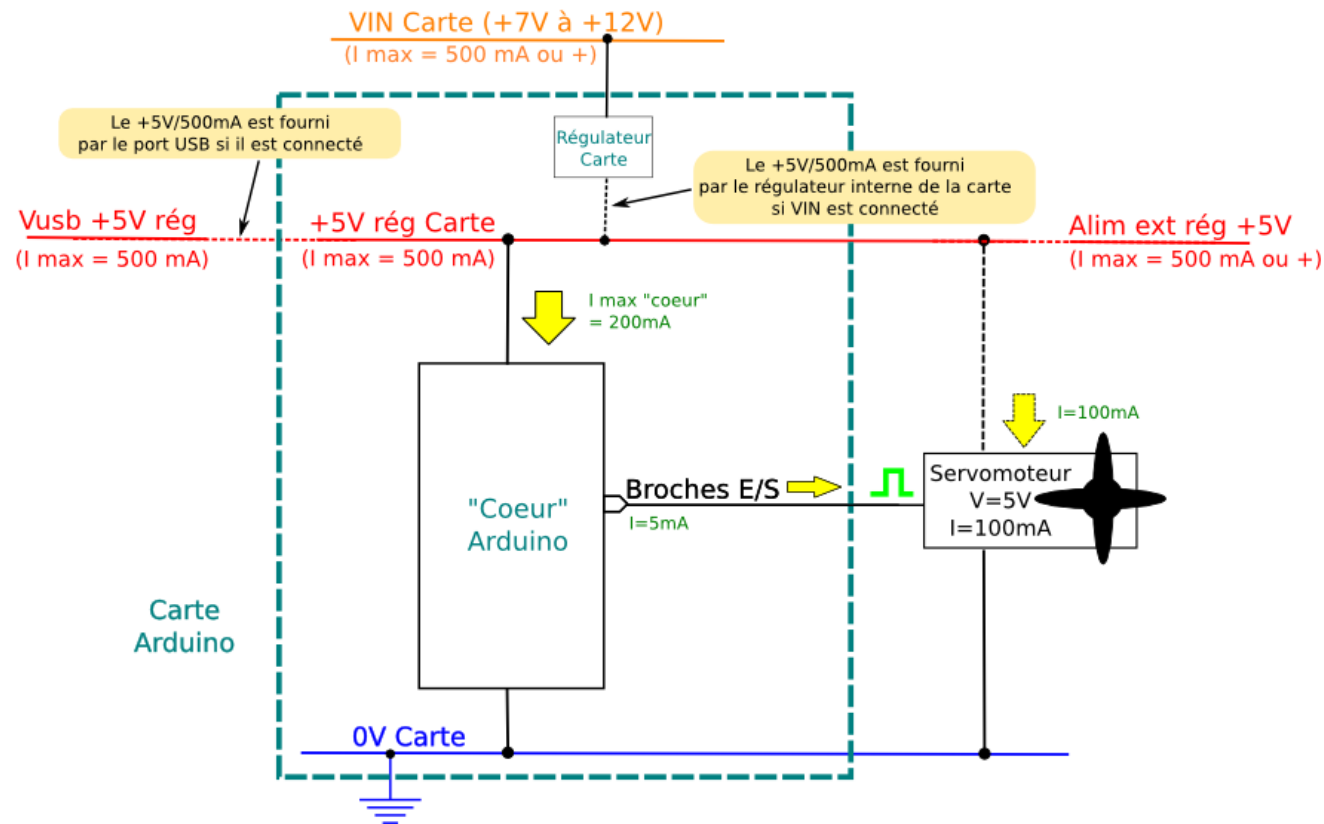
Purchased by Franck Ourion, [franck.ourion@univ-lorraine.fr](mailto:franck.ourion@univ-lorraine.fr) #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des servomoteurs standards avec une carte Arduino.

## 12. Servomoteurs : les servomoteurs standards : schéma électrique type d'utilisation avec Arduino

Le plus simple est d'utiliser le 5V rég / 500mA de la carte (possible jusqu'à 3 servomoteurs standards). 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.

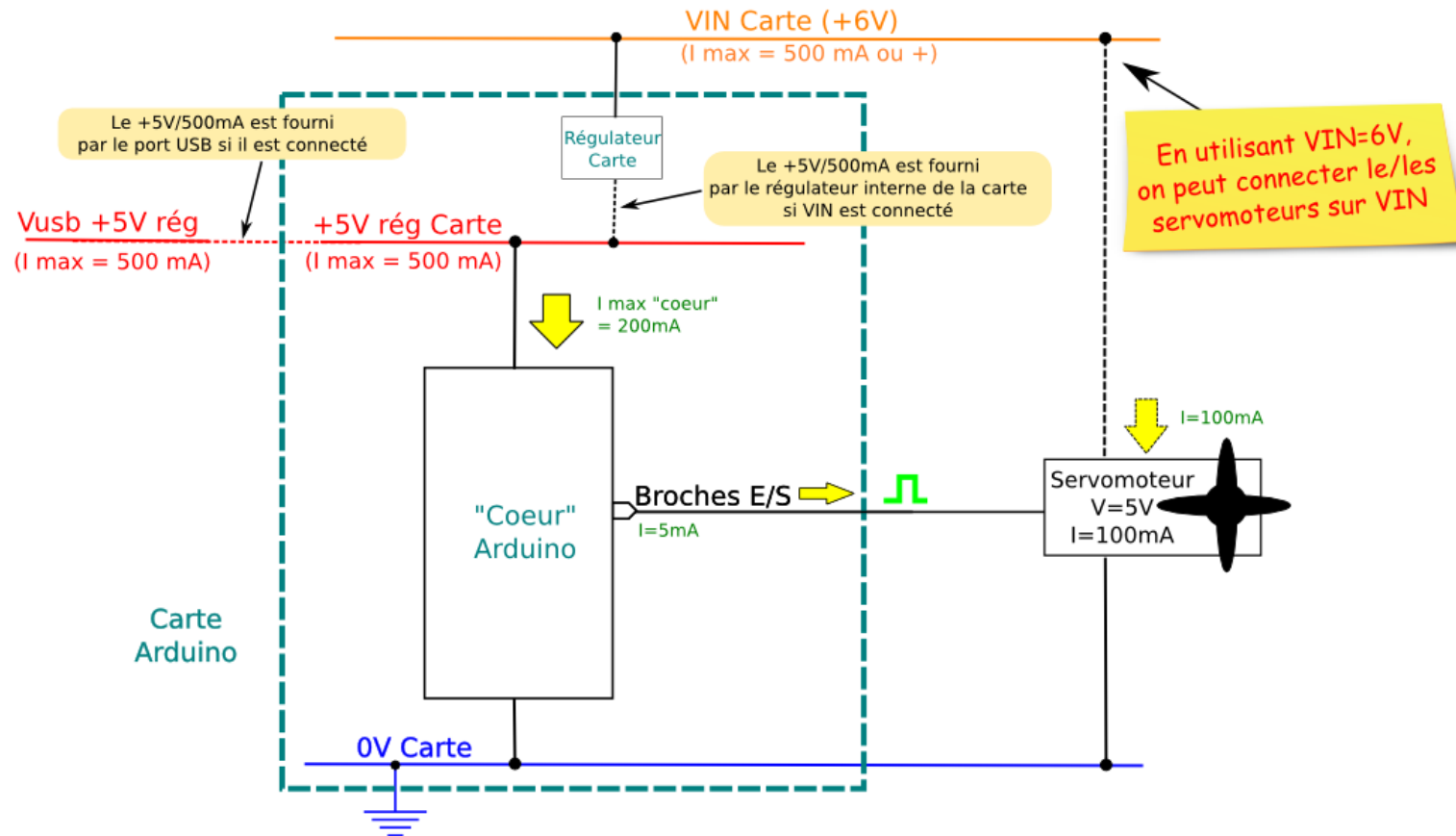




### 13. Servomoteurs : les servomoteurs standards : Variante schéma électrique type d'utilisation avec Arduino

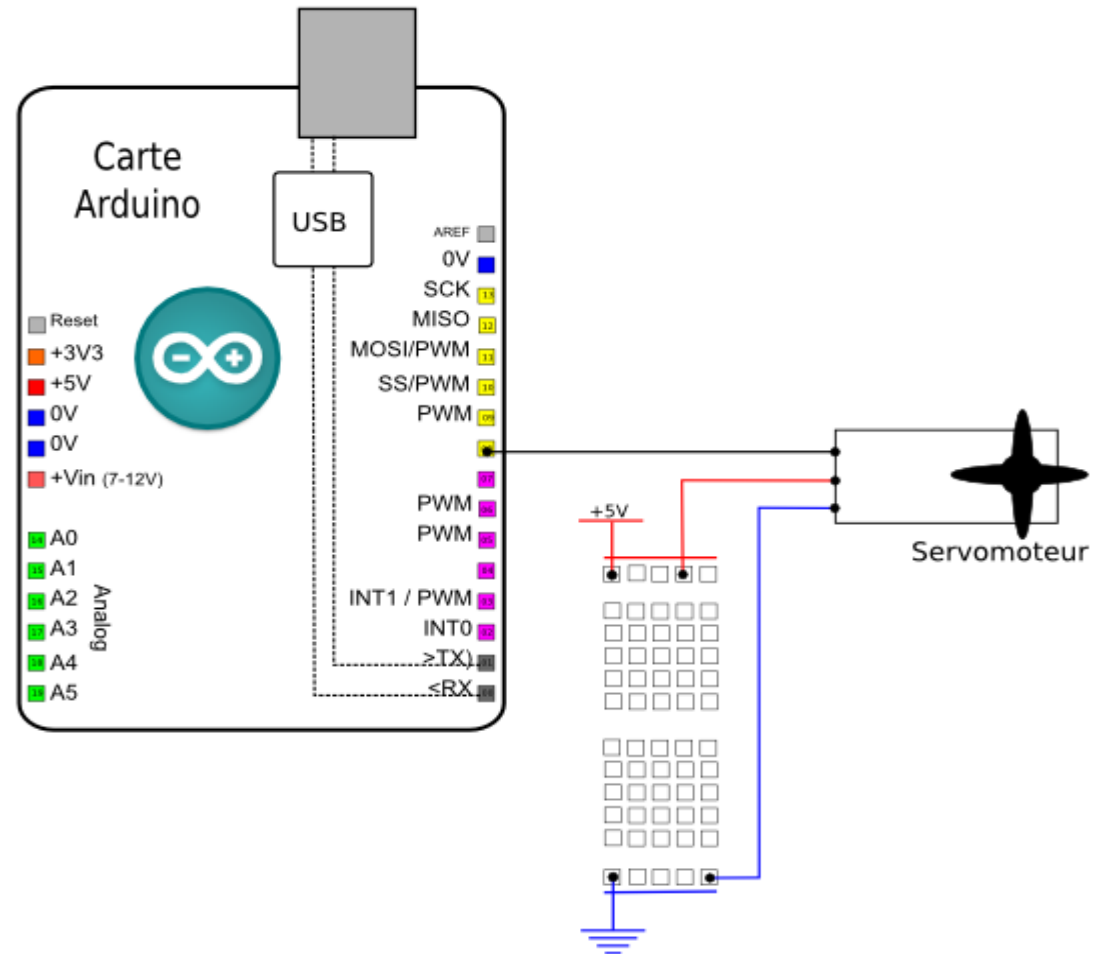
- Pour éviter de multiplier les alimentations et si on utilise plus de 3 servomoteurs, sur un robot notamment, il est également possible d'alimenter directement les servomoteurs sur VIN si on utilise VIN de 6V (ou plus si les servomoteurs sont compatibles... mais les standards supportent 6V maxi).
- 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 !



## 14. Servomoteurs : les servomoteurs standards : montage type avec une carte Arduino

Difficile de faire plus simple !



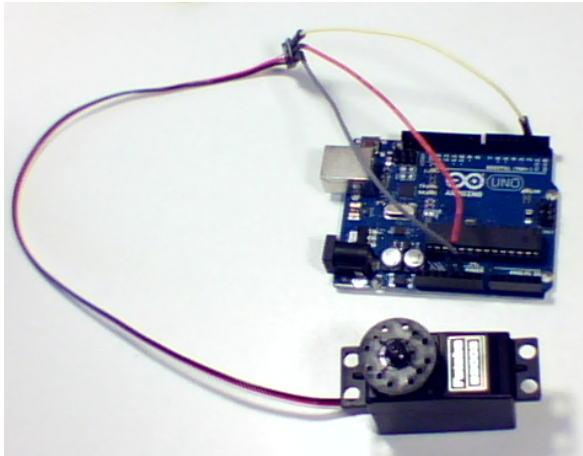


## 15. En pratique : utiliser un ou plusieurs servomoteurs avec une carte Arduino

### 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 :

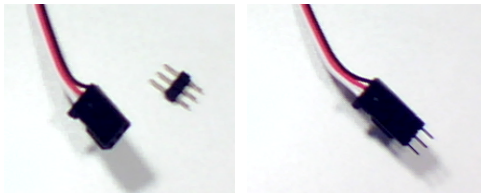
- la broche 0V (câble noir) du servomoteur au 0V de la carte Arduino (broches GND)
- la broche 5V (câble rouge) du servomoteur au +5V de la carte Arduino
- la broche de commande (câble blanc) du servomoteur à la broche numérique de la carte Arduino que l'on souhaite utiliser.



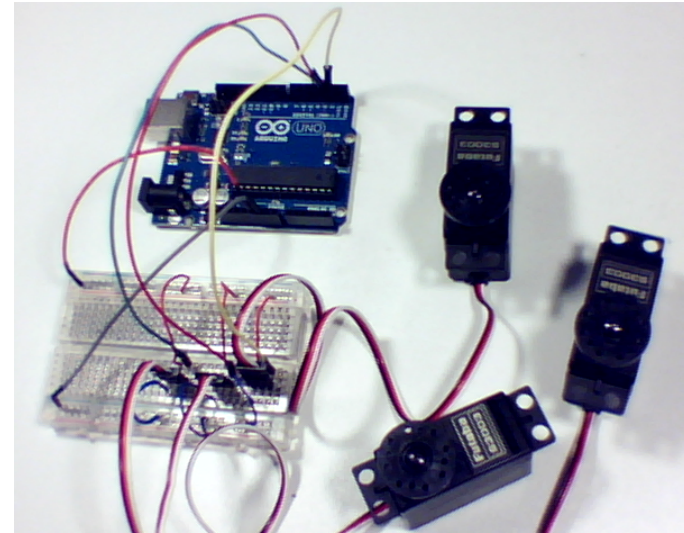
### Utiliser plusieurs servomoteurs à l'aide d'une plaque d'essai

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.

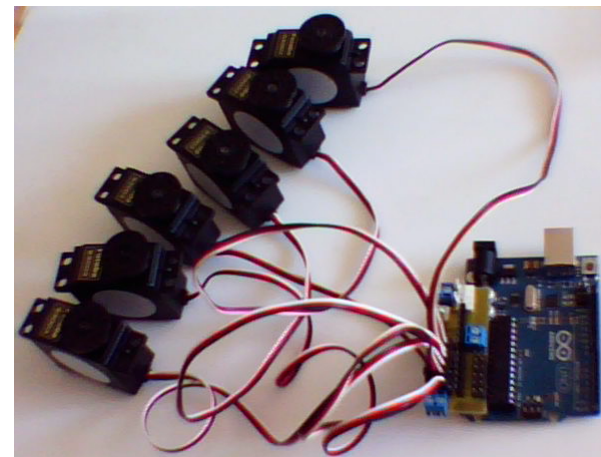
Pour que les choses soient plus simples, on commence par transformer les 3 broches femelles du connecteur du/des servomoteur(s) en 3 broches mâles à l'aide de connecteurs droits pour circuit imprimé en groupe de 3 contacts :



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

## 16. Servomoteurs : les servomoteurs standards : le principe de contrôle

### Principe de contrôle du servomoteur

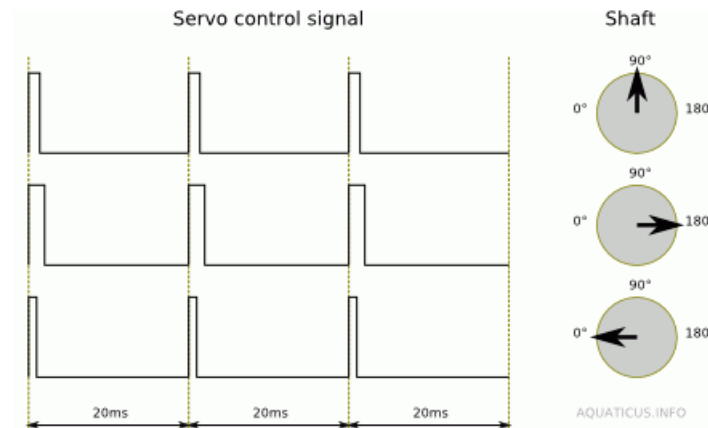
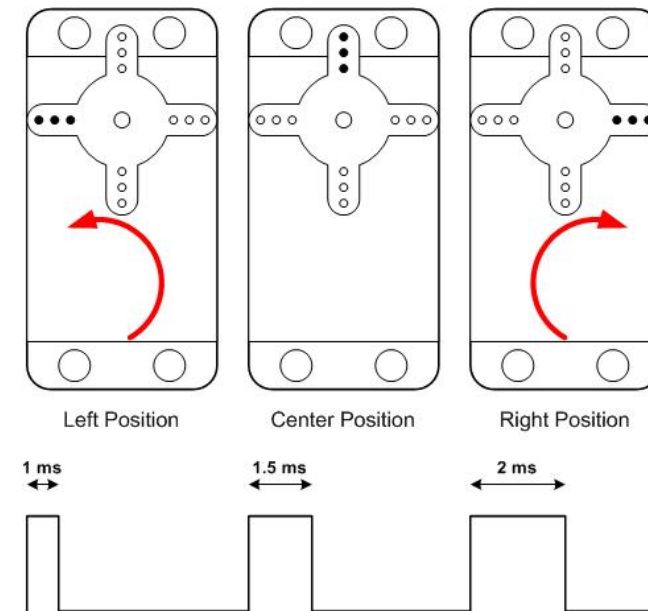
- Un servomoteur standard est contrôlé par 1 broche numérique par impulsion de type PWM : la largeur de l'impulsion va fixer la position de l'axe du servomoteur entre 0° ou 180° voire 360° selon les modèles.
- La rotation de l'axe se fait typiquement entre 0° et 180°. On appelle aussi position neutre la position médiane (90°).
- Pour orienter l'axe du servomoteur à l'angle voulu, il faut appliquer sur la broche de contrôle, de type numérique, une impulsion PWM un peu particulière ayant une largeur de durée précise (et pas un % précis..), typiquement :
  - 1 ms (1000µs) pour la position 0°
  - 1,5 ms (1500µs) pour la position médiane 90°
  - 2 ms (2000 µs) pour la position 180°
  - toutes les positions intermédiaires sont possibles
- 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 des positions extrêmes.

### En pratique, avec Arduino

- Générer une telle impulsion est facile avec Arduino grâce à la librairie **Servo** dont nous allons voir l'utilisation ultérieurement :
  - 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.**



## 17. 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>

## 18. 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)
```

```
}
```

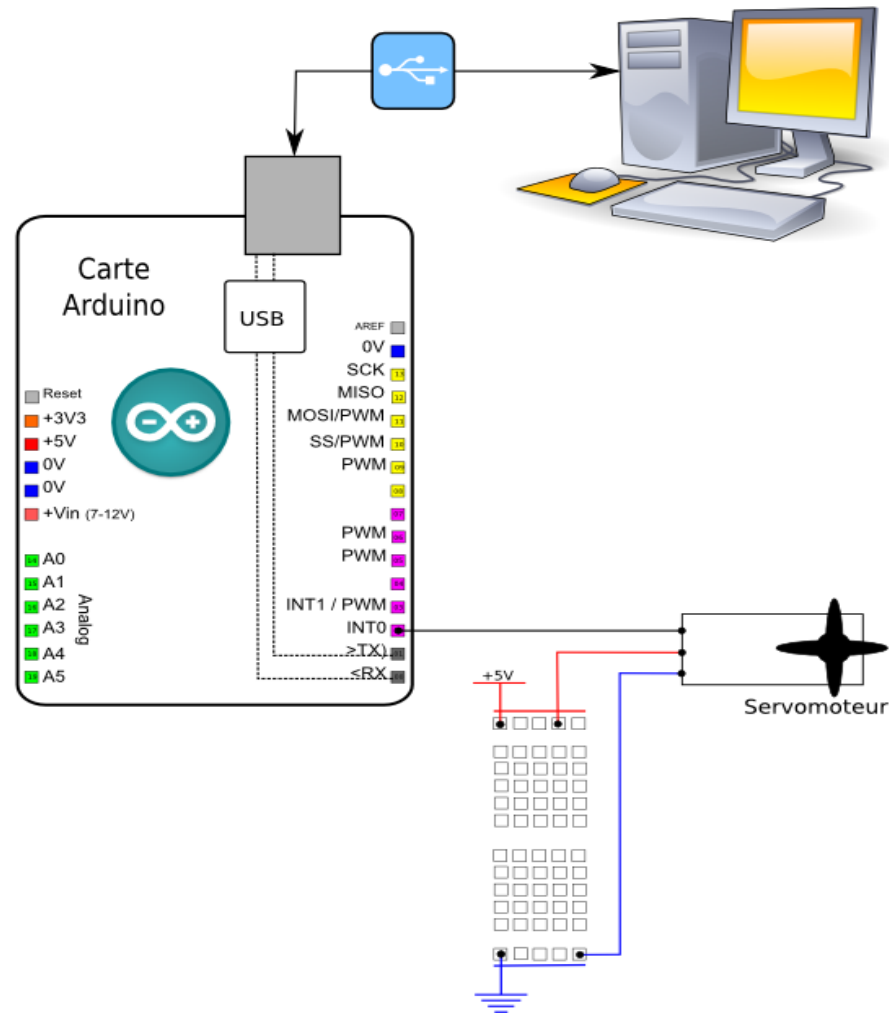
```
void loop() {
```

```
}
```

## 19. Calibrer un servomoteur 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



## 20. Calibrer un servomoteur via le Terminal 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.

### 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 positionne le 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.

### Etalonnage d'un servomoteur

Avant une première utilisation, les impulsions correspondant aux positions extrêmes d'un servomoteur ne sont pas précisément connues. Ce programme va permettre de les connaître avec précision.

Pour cela, tester les valeurs extrêmes : vous savez qu'elles sont atteintes lorsque le servomoteur vibre un peu et ne bouge plus lorsque vous modifiez la valeur utilisée.

**Une fois fait, on note ces valeurs (posMin et posMax) et on les réutilisera ensuite dans les programmes utilisant ce servomoteur. A titre d'exemple, avec un Futaba S3003, j'obtiens posMin=550 et posMax=2350.**

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

```
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*10; // 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

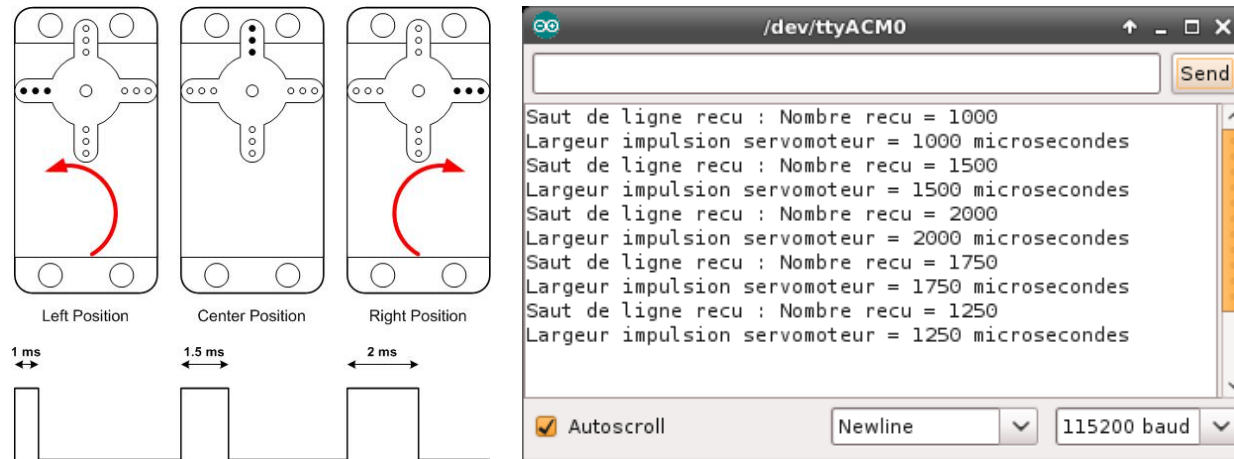
    } // fin while - fin de réception de la chaine

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



## 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 déroulante. **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 doit se positionner en conséquence. Essayer des valeurs entre 800 et 2200. Lorsque le servomoteur ne bouge plus, c'est que l'on a atteint la limite.



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 !!**



## 21. Positionner l'axe d'un servomoteur via le Terminal Série : le programme

Une fois que le servomoteur est calibré, il va devenir possible de fixer très simplement la position de l'axe. Ici nous allons le faire à partir du Terminal Série car c'est la méthode la plus sympathique pour prendre les choses en main. En plus, ça pose les bases pour un contrôle du servomoteur à partir du PC... Ce programme va donc permettre de saisir une valeur entre 0 et 180 dans le Terminal Série et de l'envoyer vers Arduino : le servomoteur se positionnera à l'angle voulu. On utilise ici le même montage que précédemment. Allez, on se lance !

### 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
  - on va déclarer 2 constantes correspondant aux 2 valeurs obtenues lors du calibrage effectué précédemment :
    - **posMin** sera la largeur de l'impulsion de la position 0°
    - **posMax** sera la largeur de l'impulsion de la position 180°
    - pour mémoire, dans le cas d'un Futaba S3003, les valeurs sont **posMin=550** et **posMax=2350**.
  - une constante de broche pour le servomoteur
- On déclare également un objet **Servo**

### 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()** mais en utilisant ici la forme de cette fonction qui initialise les positions extrêmes du servomoteur sous la forme :

```
servo.attach(brocheServo, posMin, posMax); // attache le servomoteur à la broche et initialisation des positions extremes
```

Pour plus de détails, voir :

[http://www.mon-club-elec.fr/pmwiki\\_reference\\_arduino/pmwiki.php?n=Main.Servoattach](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Servoattach)

```
//---- 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

//----- constantes de paramétrage du servomoteur ----
const int posMin=550; // largeur impulsion en µs correspondant à la position 0° du servomoteur
const int posMax=2350; // largeur impulsion en µs correspondant à la position 180° du servomoteur
//----- valeur pour un Futaba S3003 - à adapter à votre situation

const int brocheServo=2; // broche du servomoteur
Servo servo; // déclaration d'un objet servomoteur

int angleServo=0; // variable de position du servomoteur
```

```
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, posMin, posMax); // attache le servomoteur à la broche
    // et initialisation des positions extremes

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

## Fonction loop()

### Gestion du port Série

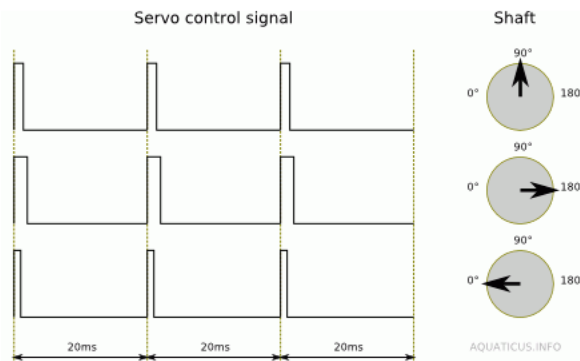
- A ce niveau, on va « écouter » le port Série en testant l'arrivée d'un caractère à l'aide d'une boucle **while** pour tester la présence d'un octet dans la file d'attente du port série avec la fonction **Serial.available()**
- Tant qu'un octet différent du saut de ligne est présent on ajoute la valeur 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 positionne le servomoteur à l'aide de la fonction **write(angle)** qui permet de positionner le servomoteur simplement à partir de la valeur de l'angle en degrés comprise entre 0 et 180° : simple non ?
- 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.

### Important

Bien distinguer **writeMicroseconds()** vue précédemment qui permet de fixer la largeur de l'impulsion PWM en micro-secondes et **write(angle)** qui permet de positionner le servomoteur simplement à partir de la valeur de l'angle en degrés comprise entre 0 et 180°.



```
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);

            angleServo=nombreReception;

            servo.write(angleServo); // positionne le servomoteur à l'angle voulu

            Serial.print ("Angle = ");
            Serial.print (angleServo); // --- affiche la valeur de l'angle utilisé
            Serial.println (" degres");

            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 chaîne n'est pas un nombre valide !");

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

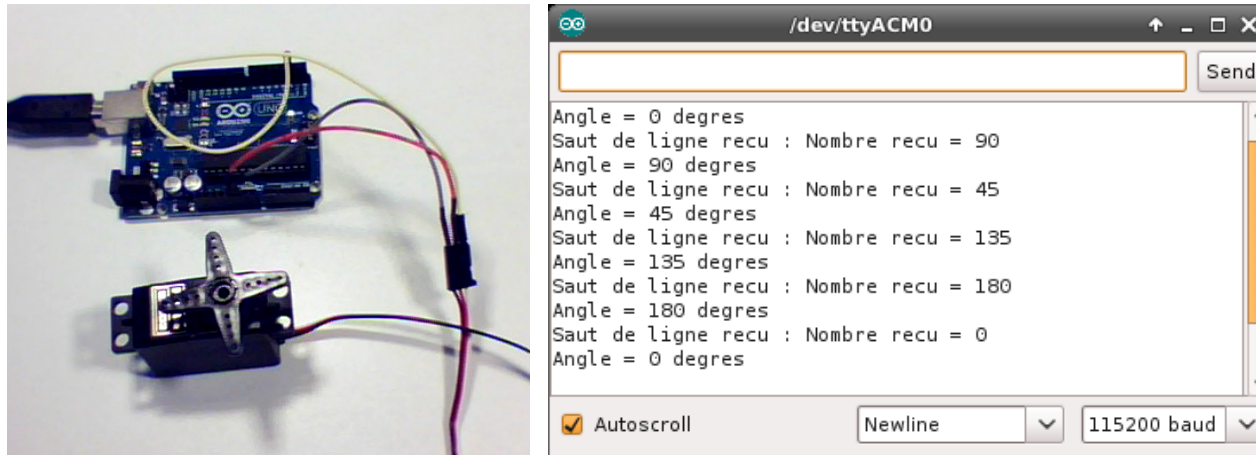
        } // fin else

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

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

## 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 déroulante. **Mettre sur « New Line »** pour ajout du « saut de ligne » après la chaîne saisie.
- saisir une valeur numérique en degrés comprise entre 0 et 180° dans le champ de saisie et appui sur <send> : le servomoteur doit se positionner en conséquence. Tester plusieurs valeurs : cool non ?



Le servomoteur se positionne à l'angle voulu...



Pour voir la vidéo associée à ce code et ce montage :  
<https://vimeo.com/42262509>

**Bravo !**

Cette fois, vous êtes capable de fixer la position du servomoteur au degré près...Pas mal pour un début !

## 22. Positionner l'axe d'un servomoteur à vitesse lente via le Terminal Série : le programme

- Bon, c'est déjà bien de pouvoir positionner le servomoteur à l'angle voulu... mais la vitesse du servomoteur n'est pas réglable à ce stade. Ceci n'est pas sans poser quelques problèmes : si vous voulez « scanner » les 180° avec une webcam fixée sur le servomoteur, si vous voulez ouvrir une pince pour lâcher un objet ou si vous bougez les pattes d'un robot hexapode, les mouvements vont être secs et ça ne sera pas satisfaisant.
- Je vous propose donc ici une solution pour contourner ce problème : nous allons voir comment fluidifier à la demande le mouvement du servomoteur de manière à ce que le passage d'une position à l'autre se fasse plus lentement. Le code est ici un peu plus conséquent, mais ça vous servira tout le temps par la suite. Tourelle pan/tilt, pince, bras robotisé 5 servomoteurs, ou encore robot hexapode à 12 servos : la fonction de positionnement progressif présentée ici vous sera indispensable ! On utilise toujours le même montage. Accrochez vos ceintures... !

### 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
  - on va déclarer 2 constantes correspondant aux 2 valeurs obtenues lors du calibrage effectué précédemment :
    - **posMin** sera la largeur de l'impulsion de la position 0°
    - **posMax** sera la largeur de l'impulsion de la position 180°
    - pour mémoire, dans le cas d'un Futaba S3003, les valeurs sont **posMin=550** et **posMax=2350**.
  - une constante de broche pour le servomoteur
- On déclare également un objet **Servo**
- On déclare également :
  - 2 variables pour stocker l'angle du servomoteur
  - une variable pour fixer la vitesse entre 2 pas, en millisecondes

```
// Programme Arduino : Rendre fluide le mouvement d'un servomoteur
//---- par X. HINAULT - 2012 - licence GPL v3 - www.mon-club-elec.fr

//---- 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

//----- constantes de paramétrage du servomoteur ----
const int posMin=550; // largeur impulsion en µs correspondant à la
position 0° du servomoteur
const int posMax=2350; // largeur impulsion en µs correspondant à la
position 180° du servomoteur
//----- valeur pour un Futaba S3003 - à adapter à votre situation

const int brocheServo=2; // broche du servomoteur
Servo servo; // déclaration d'un objet servomoteur

int angleServo=0; // variable de position du servomoteur
int angleServo0=0; // variable pour mémoriser la dernière position du
servomoteur

int vitesse=50; // vitesse de positionnement entre 2 pas - en ms - entre
5 et 100ms
```

## Fonction **setup()**

### Initialisation du port Série

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

### Initialisation du servomoteur

- on attache le servomoteur à la broche utilisée à l'aide de la fonction **attach()** mais **en utilisant ici la forme de cette fonction qui initialise les positions extrêmes du servomoteur** sous la forme :

```
servo.attach(brocheServo, posMin, posMax); // attache le servomoteur à la broche et initialisation des positions extremes
```

Pour plus de détails, voir :

[http://www.mon-club-elec.fr/pmwiki\\_reference\\_arduino/pmwiki.php?n=Main.Servoattach](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.Servoattach)

- enfin, on positionne le servomoteur à 0 degrés par défaut (position initiale) et on mémorise cette position.

```
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, posMin, posMax); // attache le servomoteur à la broche
    // et initialisation des positions extremes

    servo.write(angleServo); // positionne le servomoteur à l'angle voulu
    - initial =0°
    angleServo0=angleServo; // mémorise la position courante du servomoteur

} // 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 positionne le servomoteur à l'aide d'une fonction dédiée écrite pour cela et qui est décrite page suivante. Cette fonction **servoTo()** reçoit en paramètre :
  - le servomoteur à positionner (objet **Servo**)
  - l'angle de départ et l'angle cible
  - la vitesse à utiliser (pause en ms)
- **après chaque mouvement, on mémorise le nouvel angle courant, qui sera utilisé lors du nouvel appel de la fonction servoTo().**
- 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);

            angleServo=nombreReception;

            // servo.write(angleServo); // positionne le servomoteur à
            l'angle voulu
            servoTo(servo, angleServo0, angleServo, vitesse);
            angleServo0=angleServo; // mémorise la nouvelle position
            Serial.print ("Angle = ");
            Serial.print (angleServo); // --- affiche la valeur de l'angle
            utilisé
            Serial.println (" degres");

            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()
```



## Fonction servoTo() pour positionnement progressif d'un servomoteur

### Description

- La fonction s'appelle **servoTo()**
- Cette fonction ne renvoie rien et est donc de type **void**
- La fonction reçoit en paramètres :
  - un objet **Servo**
  - la valeur de l'angle courant et la valeur de l'angle cible (type **int**)
  - la vitesse (**int**) qui correspond en fait à la pause à réaliser entre 2 positionnements. Utiliser une valeur entre 10ms(rapide) et 100ms (lent)

### Principe

- On commence par calculer l'écart entre les 2 angles que l'on mémorise.
- on teste à l'aide d'une condition **if** si la progression est positive.
- ensuite, à l'aide d'une boucle **for()**, on fait défiler au servomoteur tous les crans intermédiaires par cran de 1 degré jusqu'à atteindre la position voulue.
- Si la progression est négative (**else**), on fait la même chose mais à l'envers...
- Des messages sont affichés pour s'assurer que tout se déroule bien.

```
//--- fonction de positionnement progressif du servomoteur par pas fixe
de n degrés-----

void servoTo( Servo toServo, int fromAngle, int toAngle, int
toVitesse ) {

    int delta=toAngle-fromAngle; // variation d'angle

    Serial.print(F("delta = ")), Serial.println(delta);

    if (delta>=0) { // si variation positive

        for (int i=0; i<delta; i++) { // defile n positions pour atteindre
angle final dans sens positif

            fromAngle=fromAngle+1; // ajoute cran
            toServo.write(fromAngle); // crée impulsion à partir valeur
angle - plus précis que write()
            Serial.print("angle courant servo = "),
            Serial.println(fromAngle);
            delay(toVitesse); // pause entre chaque positionnement

        } // fin for

    } // fin if

    else { // si variation négative

        for (int i=-delta; i>0; i--) { // defile n positions pour
atteindre angle final dans sens négatif

            fromAngle=fromAngle-1; // ajoute cran
            toServo.write(fromAngle); // crée impulsion à partir valeur
angle - plus précis que write()
            Serial.print("angle courant servo = "),
            Serial.println(fromAngle);
            delay(toVitesse); // pause entre chaque positionnement

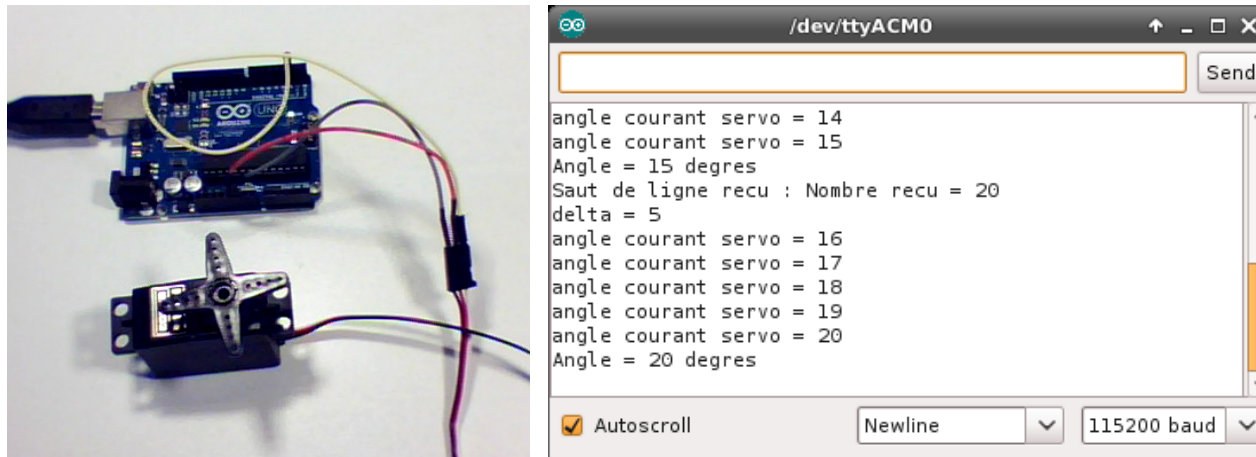
        } // fin for

    } // fin else

} // fin fonction servoTo
```

## 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 degrés comprise entre 0 et 180° dans le champ de saisie et appui sur <send> : le servomoteur doit se positionner en conséquence. Comparativement au programme précédent, le positionnement se fait de façon plus lente, permettant des mouvements adaptés au besoin. Reprogrammer Arduino en modifiant la valeur de la variable vitesse (entre 10ms = rapide et 100ms = lent).

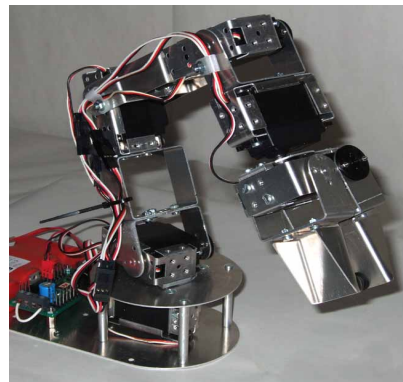


Le servomoteur se positionne à l'angle voulu à une vitesse plus lente...

### A vous de jouer !

A ce stade vous avez les bases pour utiliser les servomoteurs standards en mode simple ou en contrôlant la vitesse de mouvement.

A vous les tourelles pan/tilt, pinces et autre bras robotisés ! Voir notamment le site [www.EasyRobotics.fr](http://www.EasyRobotics.fr) ou la rubrique mecatronique sur [www.mon-club-elec.fr](http://www.mon-club-elec.fr)



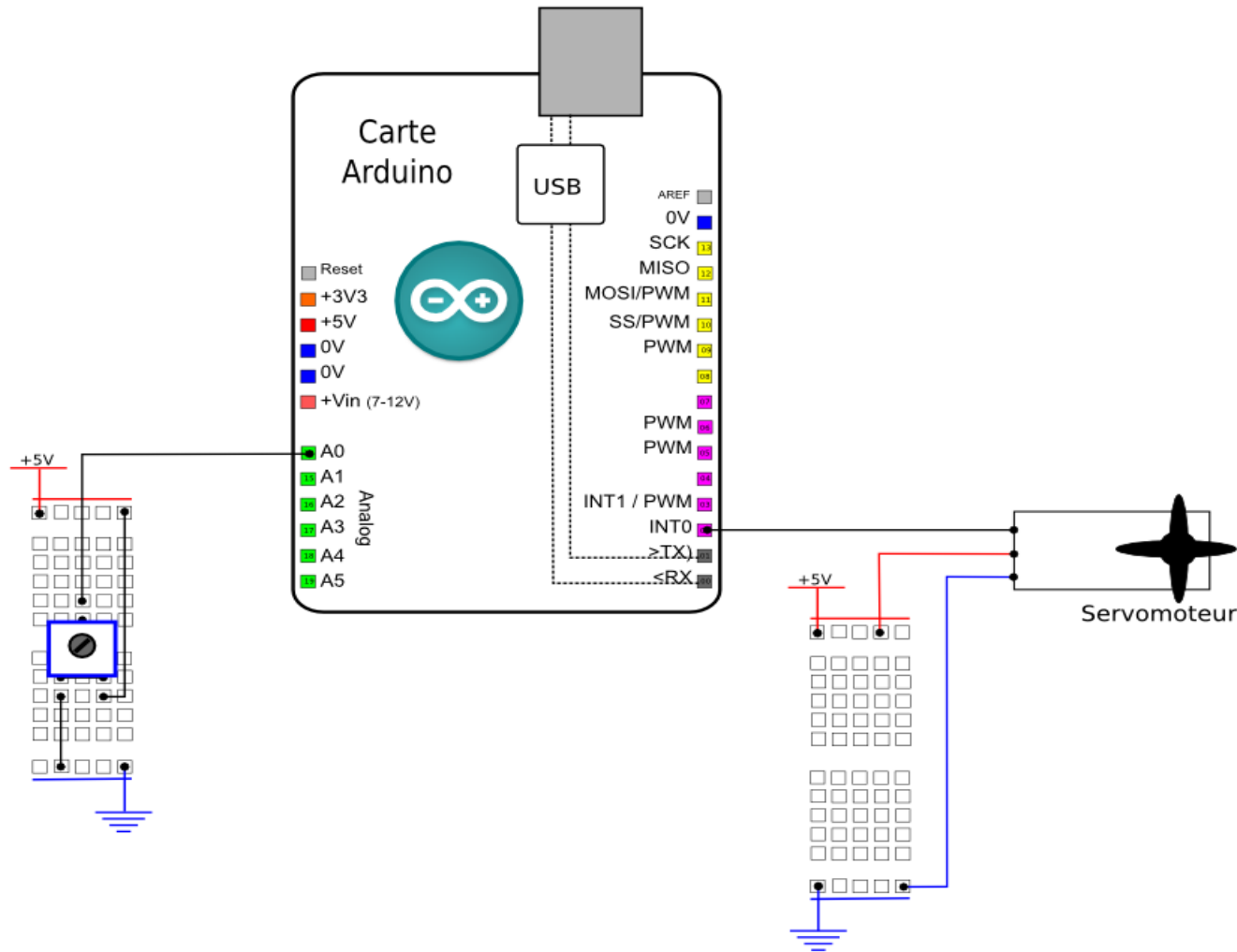
Purchased by Franck Ourion, [franck.ourion@univ-lorraine.fr](mailto:franck.ourion@univ-lorraine.fr) #6280170

Atelier Arduino : Moteurs : Apprendre à utiliser des servomoteurs standards avec une carte Arduino.

## 23. Contrôler le positionnement d'un servomoteur 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



## 24. Contrôler le positionnement d'un servomoteur 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 :
  - deux constantes **int** pour mémoriser les valeurs posMin et posMax du servomoteur utilisé. Initialiser ces constantes avec les valeurs obtenues précédemment en étalonnant votre servomoteur.
  - une constante de broche pour le servomoteur
  - une constante de broche analogique pour la résistance variable (en utilisant bien le numéro de broche analogique !)
  - deux variables **int** pour mémoriser le résultat de la mesure et la position du servomoteur (en degrés)
- On déclare également un objet **Servo**

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

//--- entete déclarative = variables et constantes globales
const int posMin=550; // largeur impulsion en µs correspondant à la
position 0° du servomoteur
const int posMax=2350; // largeur impulsion en µs correspondant à la
position 0° du servomoteur

const int brocheServo=2; // broche du servomoteur

const int brocheRVar=A0; // broche de la résistance variable

int positionServo=0; // variable de positionnement du servomoteur
int mesureRVar=0; // variable pour la mesure de la résistance variable

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

### Fonction **setup()**

- on attache le servomoteur à la broche utilisée à l'aide de la fonction **attach()** mais en utilisant ici la forme de cette fonction qui initialise les positions extrêmes du servomoteur sous la forme :

```
servo.attach(brocheServo, posMin, posMax); // attache le servomoteur à
la broche en initialisant les positions extrêmes du servomoteur
```

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

    servo.attach(brocheServo, posMin, posMax); // 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 la valeur obtenue en la valeur angulaire correspondante à l'aide de la fonction `map()` de telle sorte que :
  - 0 mV correspond à la position 0° du servomoteur
  - 1023 mV correspond à la position 180° du servomoteur
- on positionne enfin 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 position du servomoteur !

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

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

    mesureRVar=analogRead(brocheRVar); // mesure la tension sur la broche
    analogique

    positionServo=map(mesureRVar, 0,1023, 0, 180); // convertit la valeur
    en degrés

    servo.write(positionServo); // positionne le servomoteur dans l'angle
    voulu

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

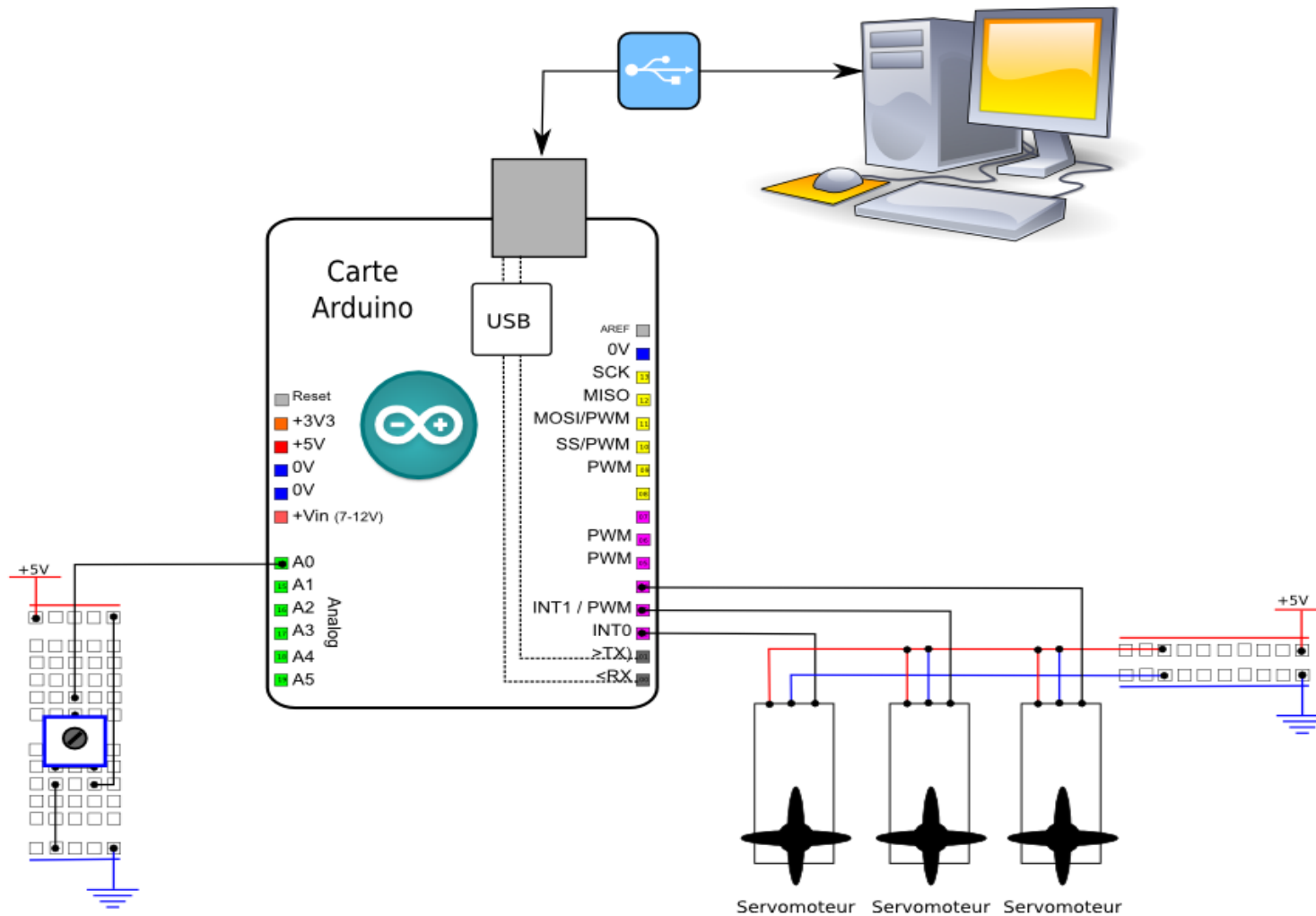
### Fonctionnement du programme

- Tourner la résistance variable : le servomoteur bouge en conséquence entre 0° et 180° !

Vous avez les bases pour contrôler un servomoteur simplement avec un potentiomètre : à vous de jouer !

## 25. Contrôler le positionnement de 3 servomoteurs à l'aide d'une résistance variable : 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.
- La sortie d'une résistance variable (connectée entre 0V et +5V) sera connectée sur une broche analogique de la carte Arduino et contrôlera l'ensemble des 3 servomoteurs.



## 26. Contrôler le positionnement de 3 servomoteurs à l'aide d'une résistance variable : le programme

Nous allons donc reprendre la même structure de programme que celle vue précédemment, mais cette fois en utilisant 3 servomoteurs synchronisés. L'intérêt ici est de montrer comment utiliser un tableau d'objets **Servo** ainsi que de revoir le principe des tableaux pour la désignation des broches.

### Entête déclarative

- On commence par inclure la librairie **Servo**
- On déclare :
  - deux constantes **int** pour mémoriser les valeurs posMin et posMax du servomoteur utilisé. Initialiser ces constantes avec les valeurs obtenues précédemment en étalonnant votre servomoteur. On suppose ici que ces valeurs sont identiques pour les 3 servomoteurs. Si ce n'est pas le cas, utiliser un tableau de constantes pour chaque valeur.
  - une constante fixant le nombre de servomoteurs.

Noter que ce programme est transposable à l'identique par simple changement de cette ligne, pour 4, 6, 12 servomoteurs ou + !

- un tableau de 3 constantes de broches pour les servomoteurs. On utilise ici les broches numériques 2,3 et 4.
  - une constante de broche analogique pour la résistance variable (en utilisant bien le numéro de broche analogique !)
  - deux variables **int** pour mémoriser le résultat de la mesure et la position du servomoteur (en degrés)
- On déclare ici non par un objet **Servo** mais un tableau d'objets **Servo** de la même façon qu'on le ferait pour une variable. Chaque servomoteur sera dès lors accessible sous la forme `servo[i]`, où *i* est l'index (**i=0 pour le premier servomoteur**).

### Fonction **setup()**

- à l'aide d'une boucle **for**, on passe en revue les 3 servomoteurs et on attache chaque servomoteur à la broche utilisée à l'aide de la fonction **attach()** mais en utilisant ici la forme de cette fonction qui initialise les positions extrêmes du servomoteur sous la forme :

```
servo.attach(brocheServo, posMin, posMax); // attache le servomoteur à la broche en initialisant les positions extrêmes du servomoteur
```

```
// Programme Arduino : Contrôler 3 servomoteurs simultanément avec 1
résistance variable
//---- par X. HINAULT - 2012 - licence GPL v3 - www.mon-club-elec.fr

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

//--- entete déclarative = variables et constantes globales
const int posMin=550; // largeur impulsion en µs correspondant à la
position 0° du servomoteur
const int posMax=2350; // largeur impulsion en µs correspondant à la
position 180° du servomoteur
//---- valeur pour un servomoteur Futaba S3003

const int nombreServo=3; // constante fixant le nombre de servomoteurs

const int brocheServo[nombreServo]={2,3,4}; // tableau des broches des
servomoteurs
const int brocheRVar=A0; // broche de la résistance variable

int positionServo=0; // variable de positionnement du servomoteur
int mesureRVar=0; // variable pour la mesure de la résistance variable

Servo servo[nombreServo]; // déclaration d'un tableau de 3 objets
servomoteur
```

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

for (int i=0; i<nombreServo; i++) {
    servo[i].attach(brocheServo[i], posMin, posMax); // attache le
servomoteur à la broche
} // fin for

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



## Fonction `loop()`

### Positionnement des servomoteurs

- 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 la valeur obtenue en la valeur angulaire correspondante à l'aide de la fonction `map()` de telle sorte que :
  - 0 mV correspond à la position 0° du servomoteur
  - 1023 mV correspond à la position 180° du servomoteur
- à l'aide d'une boucle `for` on passe en revue les 3 servomoteurs et à chaque fois, on positionne 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 position du servomoteur !

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

### Fonctionnement du programme

- Tourner la résistance variable : les 3 servomoteurs bougent simultanément en conséquence entre 0° et 180° !

Cette fois, vous avez les bases pour contrôler des dispositifs utilisant 4, 6, 12 (voire plus) servomoteurs : cool non ?

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

  mesureRVar=analogRead(brocheRVar); // mesure la tension sur la broche
  analogique

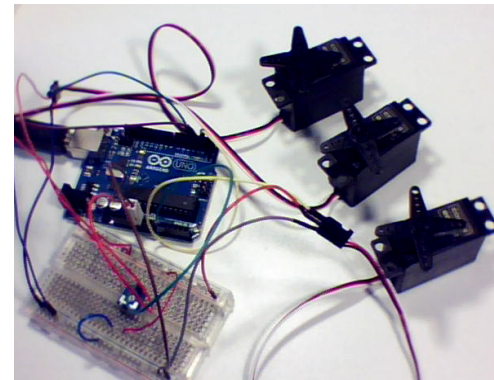
  positionServo=map(mesureRVar, 0,1023, 0, 180); // convertit la valeur
  en degrés

  for (int i=0; i<nombreServo; i++) {

    servo[i].write(positionServo); // positionne le servomoteur dans
    l'angle voulu

  } // fin for

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



Le câblage se surcharge un peu... utiliser un shield dédié pour simplifier les choses !

## 27. 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](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](http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.LibrairieServo)

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

- d'utiliser des servomoteurs standards avec une carte Arduino afin d'être en mesure de contrôler la position angulaire de l'axe du servomoteur. En sachant faire cela, vous avez les connaissances nécessaires pour mettre en oeuvre des mécaniques robotisées simples ou plus complexes (pince, bras à 6 servomoteurs, etc..).

# Table des matières

Moteurs : |

Apprendre à utiliser des servomoteurs standards 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|

Technique : utiliser un dispositif 5V / < 300mA avec la carte Arduino|

Servomoteurs : les servomoteurs standards : concrètement|

Servomoteurs : les servomoteurs standards : exemples d'utilisation |

Servomoteurs : les servomoteurs standards : fiche technique. |

Infos techniques utiles pour les servomoteurs standards|

Infos techniques utiles pour les servomoteurs : combiner des servomoteurs facilement !|

Servomoteurs : les servomoteurs standards : schéma électrique type d'utilisation avec Arduino|

Servomoteurs : les servomoteurs standards : Variante schéma électrique type d'utilisation avec Arduino|

Servomoteurs : les servomoteurs standards : montage type avec une carte Arduino|

|

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

Servomoteurs : les servomoteurs standards : le principe de contrôle |

Langage Arduino : Introduction aux bibliothèques|

Langage Arduino : la bibliothèque Servo pour le contrôle des servomoteurs|

Calibrer un servomoteur via le port Série : le montage|

Calibrer un servomoteur via le Terminal Série : le programme|

Positionner l'axe d'un servomoteur via le Terminal Série : le programme|

Positionner l'axe d'un servomoteur à vitesse lente via le Terminal Série : le programme|

Contrôler le positionnement d'un servomoteur à l'aide d'une résistance variable : le montage|

Contrôler le positionnement d'un servomoteur à l'aide d'une résistance variable : le programme|

Contrôler le positionnement de 3 servomoteurs à l'aide d'une résistance variable : le montage|

Contrôler le positionnement de 3 servomoteurs à l'aide d'une résistance variable : 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](http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.ATELIERS)