

LE BLOG D'ESKIMON

Tutoriels Arduino (ou pas) et articles divers

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

Posté par Eskimon le 12 décembre 2013, 12:22

Dans ce chapitre, nous allons parler d'un moteur que nos amis modélistes connaissent bien : le **Servomoteur** (abrégé : "servo"). C'est un moteur un peu particulier, puisqu'il confond un ensemble de mécanique et d'électronique, mais son principe de fonctionnement reste assez simple. Les parties seront donc assez courtes dans l'ensemble car les servomoteurs contiennent dans leur "ventre" des moteurs à courant continu que vous connaissez à présent. Cela m'évitera des explications supplémentaires. 😊

Sommaire

- Principe du servo-moteur
- La commande d'un servo-moteur
- Arduino et les servomoteurs
- L'électronique d'asservissement
- Un peu d'exercice !
- Tester un servomoteur "non-standard"



Principe du servo-moteur

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

Vue générale

LE SERVO, UN DRÔLE DE MOTEUR

^
ençons en image, avec la photographie d'un servomoteur :



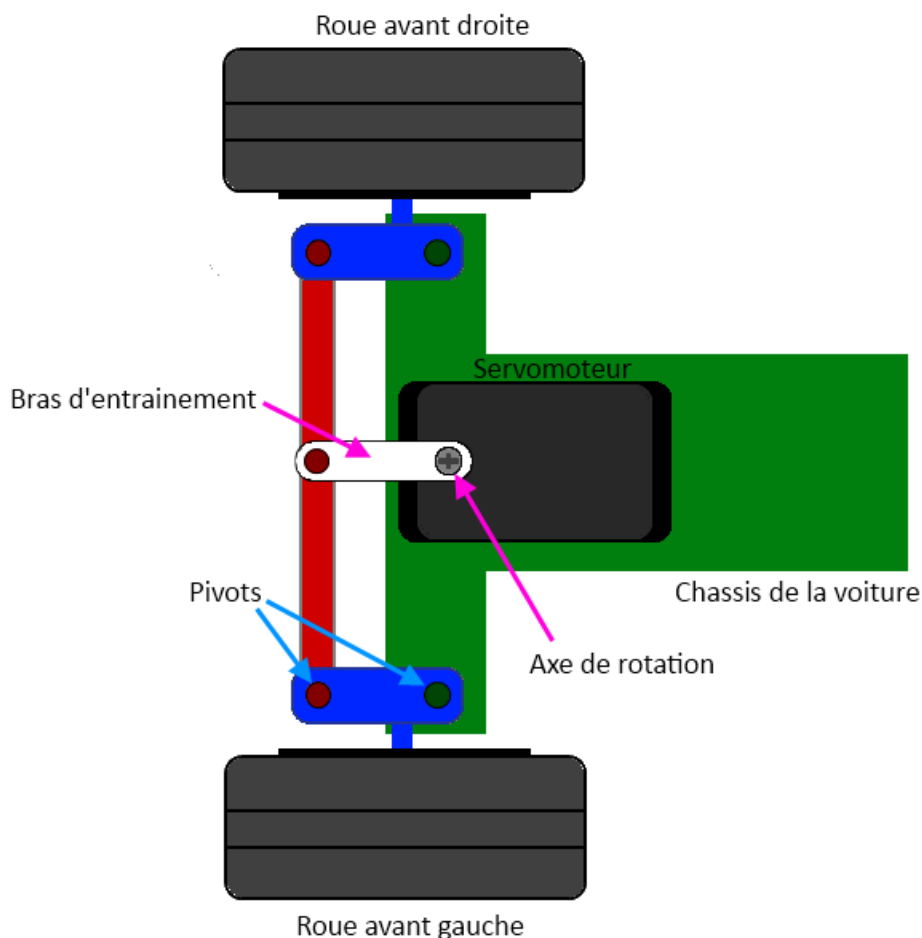
C'est, en règle générale, à quoi ils ressemblent, variant selon leur taille.

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

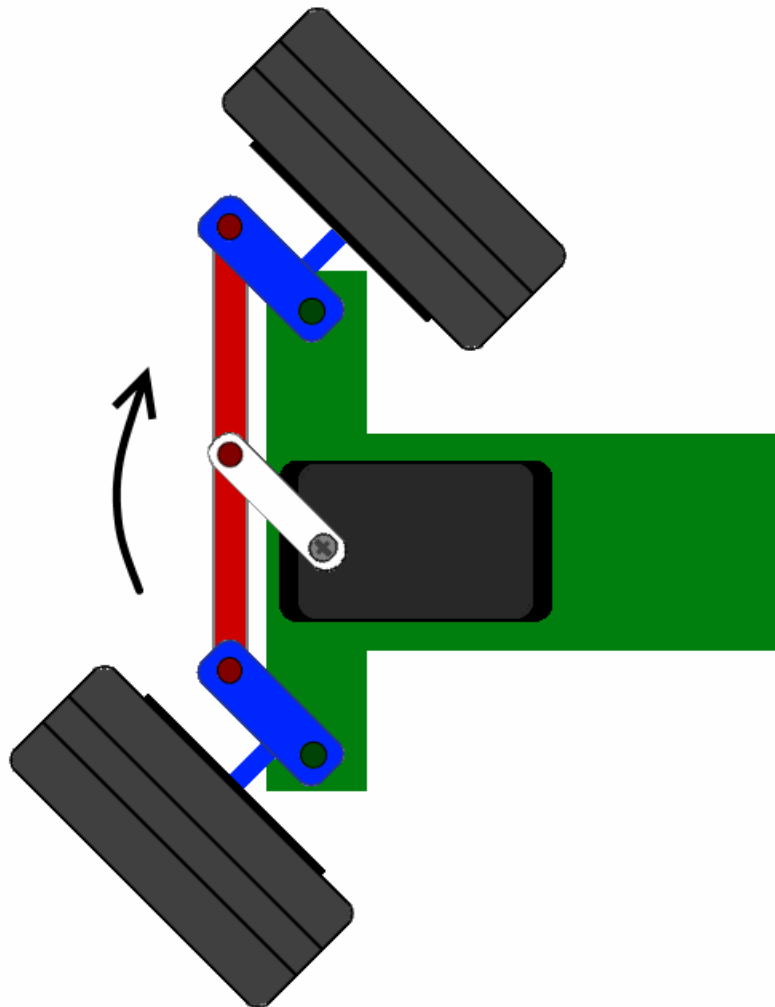
J'avais dit que c'était des moteurs particuliers ! En détail, voyons à quoi ils servent. De manière semblable aux moteurs à courant continu, les servomoteurs disposent d'un axe de rotation. Sur la photo, il se trouve au centre de la roue blanche. Cet axe de rotation est en revanche entravé par un système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

L'EXEMPLE DE LA VOITURE RADIOCOMMANDÉE

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



Vue de dessus Représentation schématique du système de guidage des roues d'une voiture radiocommandée. Chaque roue est positionnée sur un axe de rotation (partie bleue) lui-même monté sur un pivot sur le châssis de la voiture (en vert). La baguette (rouge) permet de garder la parallélisme entre les roues. Si l'une pivote vers la gauche, l'autre en fait de même (ben ouais, sinon la voiture devrait se couper en deux pour aller dans les deux directions opposées 😊). Cette baguette est fixée, par un pivot encore, au bras de sortie du servomoteur. Ce bras est à son tour fixé à l'axe de rotation du servomoteur. Ainsi, lorsque le servomoteur fait tourner son axe, il entraîne le bras qui entraîne la baguette et fait pivoter les roues pour permettre à la voiture de prendre une direction dans son élan (tourner à gauche, à droite, ou aller tout droit). Il n'y a rien de bien compliqué. Ce qu'il faut retenir est que le servomoteur va entraîner la baguette pour orienter les roues dans un sens ou dans l'autre. Elles auront donc un angle d'orientation par rapport au châssis de la voiture. Voyez plutôt :



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

Composition d’un servomoteur

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

APPARENCE

On en trouve de toutes les tailles et de toutes les puissances. La plupart du temps la sortie peut se positionner entre 0

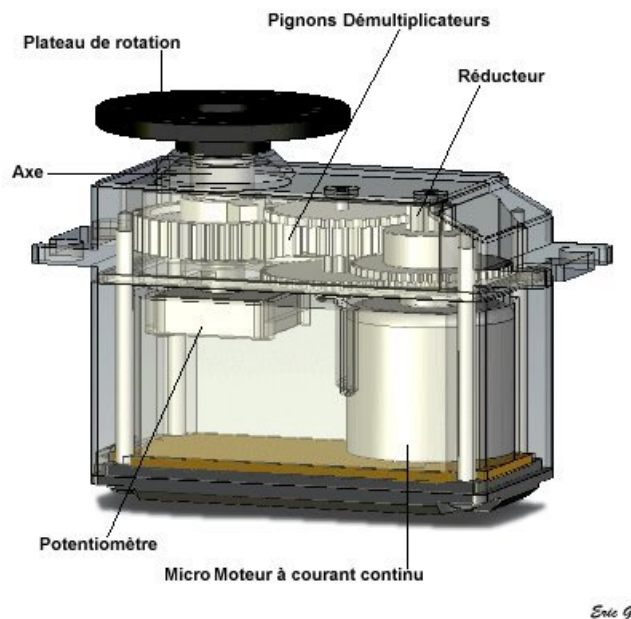
et 180°. Cela dit, il en existe également dont la sortie peut se débattre sur seulement 90° et d'autres, ayant un plus grand débattement, sur 360°. Ceux qui ont la possibilité de faire plusieurs tours sont souvent appelés **servo-treuil**. Enfin, les derniers, qui peuvent faire tourner leur axe sans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

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

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

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

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



CONNECTIQUE

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

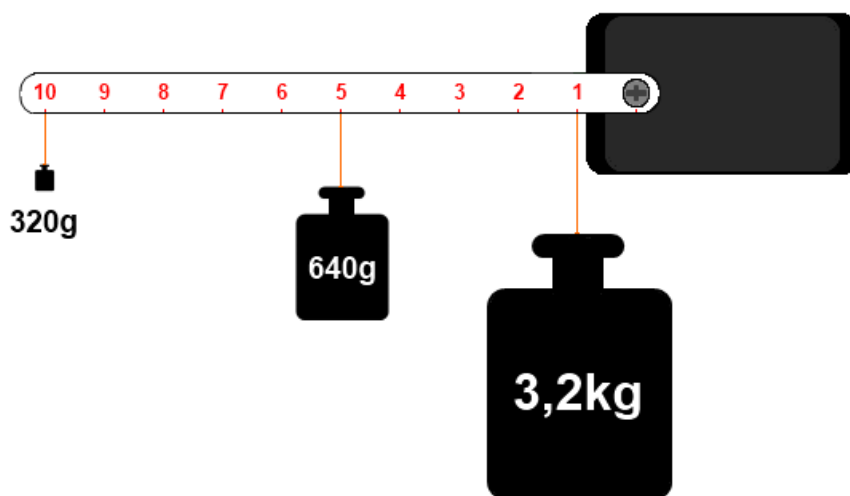
- **rouge** : pour l'alimentation positive (4.5V à 6V en général)
- **noir** ou **marron** : pour la masse (0V)
- **orange**, **jaune**, **blanc**, ... : entrée du signal de commande

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

LA MÉCANIQUE

Comme on le voit dans l'image précédente, le servomoteur possède plusieurs pignons (engrenages) en sortie du petit moteur CC. Cet ensemble est ce qui constitue le **réducteur**. Ce réducteur fait deux choses : d'une part il réduit la vitesse de rotation en sortie de l'axe du servomoteur (et non du moteur CC), d'autre part il permet d'augmenter le couple en sortie du servomoteur (là encore non en sortie du moteur CC). Alors, à quoi ça sert de réduire la vitesse et d'augmenter le couple ? Eh bien les moteurs CC se débrouillent très bien pour tourner très vite mais lorsqu'ils font une si petite taille

ils sont bien moins bons pour fournir du couple. On va donc utiliser ce réducteur qui va réduire la vitesse, car nous n'avons pas besoin d'avoir une vitesse trop élevée, et augmenter le couple pour ainsi pouvoir déplacer une charge plus lourde. Ceci est prouvé par la formule que je vous ai donnée dans le chapitre précédent : $R = \frac{\omega_{entree}}{\omega_{sortie}} = \frac{C_{sortie}}{C_{entree}}$. Le rapport de réduction (R) du réducteur définit le couple et la vitesse de sortie (en sortie du réducteur) selon la vitesse et le couple d'entrée (en sortie du moteur CC). Ces données sont souvent transparentes lorsque l'on achète un servomoteur. Dans la quasi totalité des cas, nous n'avons que la vitesse angulaire (en degré par seconde °/s), le couple de sortie du servomoteur et le débattement maximal (s'il s'agit d'un servomoteur ayant un débattement de 0 à 90°, 180, 360 ou autre). Et c'est largement suffisant étant donné que c'est ce qui nous intéresse dans le choix d'un servomoteur. Il y a cependant une unité qui pourra peut-être vous donner quelques doutes ou une certaine incompréhension. Cette caractéristique est celle du couple du servomoteur et a pour unité le **kg.cm** (kilogramme-centimètre). Nous allons tout de suite rappeler ce que cela signifie. Avant tout, rappelons la formule suivante : $C = F \times r$ qui donne la relation entre le couple C du servomoteur (en kilogramme mètre), F la force exercée sur le bras du servomoteur (en kilos) et r la distance (en m) à laquelle s'exerce cette force par rapport à l'axe de rotation du servomoteur. Disséquons dans notre langage la signification de cette formule : le couple (C) exercé sur un axe est égal à la force (F) appliquée au bout du levier accroché à ce même axe. À force identique, plus le levier est long et plus le couple exercé sur cet axe est important. En d'autres termes, si votre servomoteur dispose d'un bras d'un mètre de long (oui c'est très long) eh bien il aura beaucoup plus de difficultés à soulever une charge de, disons 10g, que son homologue qui supporte la même charge avec un bras nettement raccourci à 10 centimètres. Prenons l'exemple d'un servomoteur assez commun, le Futaba s3003. Sa documentation nous indique que lorsqu'il est alimenté sous 4.8V (on reviendra dessus plus tard), il peut fournir un couple (*torque* en anglais) de **3,2kg.cm**. C'est à dire, qu'au bout de son bras, s'il fait 1 centimètre, il pourra soulever une charge de 3,2kg. Simple, n'est-ce pas ? 😊 Si le bras fait 10 centimètres, vous aurez compris que l'on perd 10 fois la capacité à soulever une masse, on se retrouve alors avec un poids de 320g au maximum (sans compter le poids du bras lui-même, certes négligeable ici, mais parfois non).



Voilà une image qui permet d'illustrer un peu ce que je vous raconte depuis tout à l'heure (ça commençait à être ennuyeux, non ?). Bref. Ici, chaque poids représenté est celui maximum que peut soulever le servomoteur selon la distance à laquelle il est situé. Et ne vous avisez pas de les mettre tous car votre pauvre servo serait bien dans l'incapacité de les soulever en même temps. Et oui, malgré le fait qu'il n'y ait que 320g au bout du bras, le servo voit comme s'il y avait un poids de 3,2kg ! Dans cette situation on aurait trois fois 3,2kg, ce qui ferait un poids total de 9,6kg ! Impossible pour le servo de ne bouger ne serait-ce que d'un millimètre (vous risqueriez fort de le détruire d'ailleurs).

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

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

distance voulue :

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

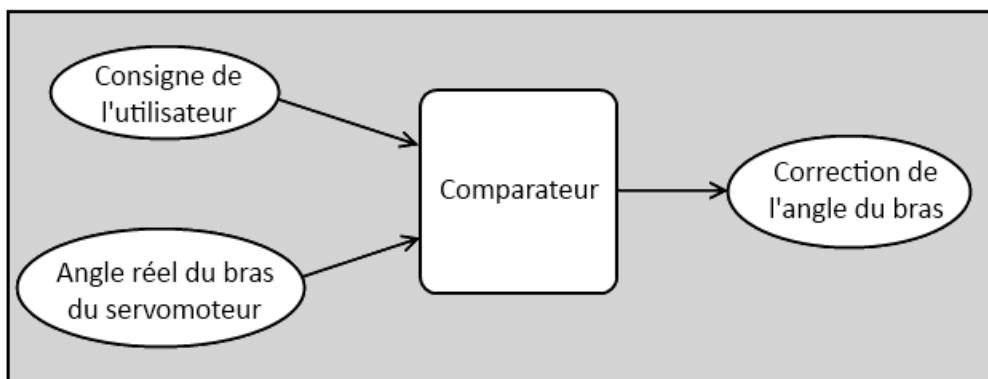
Avec :

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

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

L'ÉLECTRONIQUE D'ASSERVISSEMENT

“Qu'est-ce que l'asservissement ?”, vous demandez-vous sans doute en ce moment. Malgré la signification peu intuitive que ce terme porte, il se cache derrière quelque chose de simple à comprendre, mais parfois très compliqué à mettre en œuvre. Heureusement, ce n'est pas le cas pour le servomoteur. Toutefois, nous n'entrerons pas dans le détail et nous nous contenterons de présenter le fonctionnement. L'asservissement n'est ni plus ni moins qu'un moyen de gérer une consigne de régulation selon une commande d'entrée. Euuuh, vous me suivez ? 😊 Prenons l'exemple du servomoteur : on l'alimente et on lui envoie un signal de commande qui permet de définir à quel angle va se positionner le bras du servomoteur. Ce dernier va s'exécuter. Essayez de forcer sur le bras du servomoteur... vous avez vu ? Quelle que soit la force que vous exercez (dans les limites du raisonnable), le servo va faire en sorte de toujours garder la position de son bras à l'angle voulu. Même si le poids est largement supérieur à ce qu'il peut supporter, il va essayer de remettre le bras dans la position à laquelle il se trouvait (à éviter cependant). Ainsi, si vous changez l'angle du bras en forçant dessus, lorsque vous relâchez le bras, il va immédiatement reprendre sa position initiale (celle définie grâce au signal de commande). Pour pouvoir réaliser le maintien de la position du bras de manière correcte, le servo utilise une **électronique de commande**. On peut la nommer **électronique d'asservissement**, car c'est elle qui va gérer la position du bras du servomoteur. Cette électronique est constituée d'une zone de comparaison qui compare (étonnamment 😊) la position du bras du servo au signal de commande. Le deuxième élément qui constitue cette électronique, c'est le capteur de position du bras. Ce capteur n'est autre qu'un potentiomètre couplé à l'axe du moteur. La mesure de la tension au point milieu de ce potentiomètre permet d'obtenir une tension image de l'angle d'orientation du bras. Cette position est ensuite comparée, je le disais, à la consigne (le signal de commande) qui est transmise au servomoteur. Après une rapide comparaison entre la consigne et valeur réelle de position du bras, le servomoteur (du moins son électronique de commande) va appliquer une correction si le bras n'est pas orienté à l'angle imposé par la consigne.



Synoptique de fonctionnement de l'asservissement du servomoteur

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



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

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

La commande d'un servo-moteur

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

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

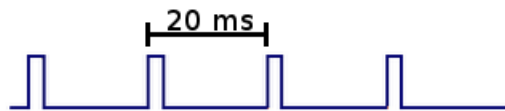
Le signal de commande

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

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

LA FRÉQUENCE FIXE

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



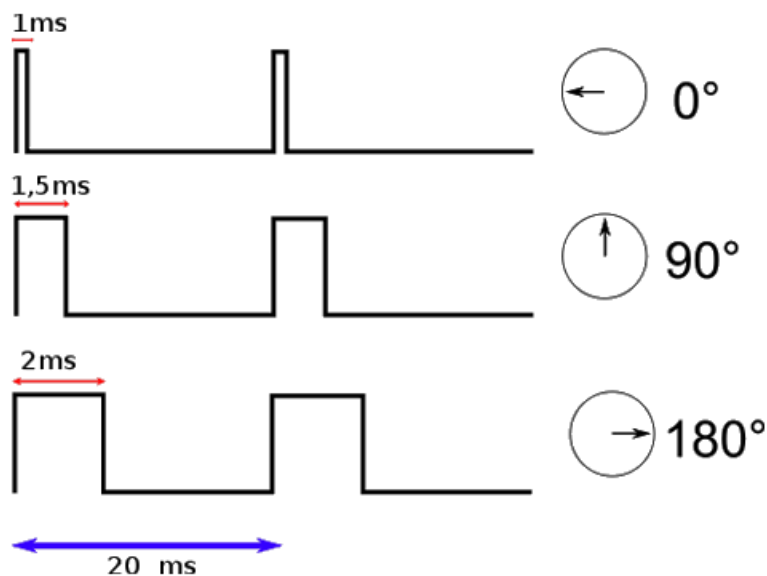
Malheureusement, la fonction `analogWrite()` de Arduino ne possède pas une fréquence de 50Hz, mais dix fois plus élevée, de 500Hz environ. On ne pourra donc pas utiliser cette fonction.

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

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

LA DURÉE DE L'ÉTAT HAUT

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



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

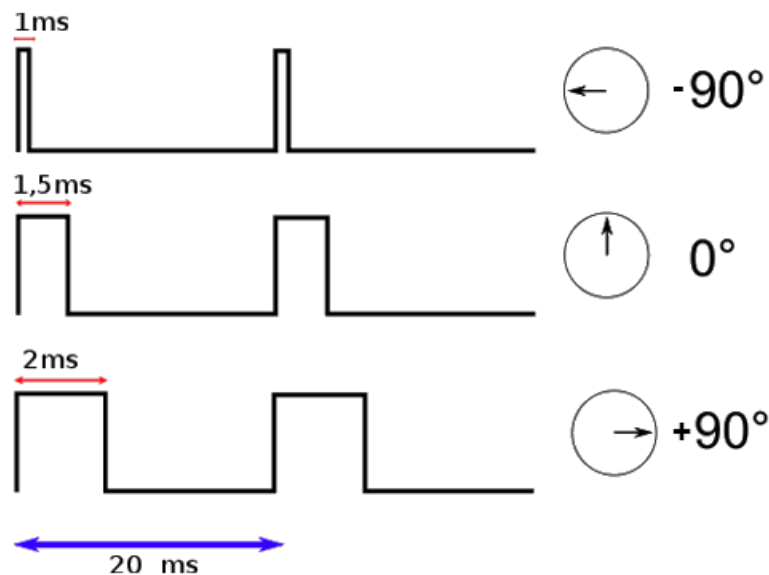
Et comment je fais si je veux que mon servomoteur face un angle de 45° ? Ça marche pas ? o_O

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

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

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

-90° à +90° :



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

Arduino et les servomoteurs

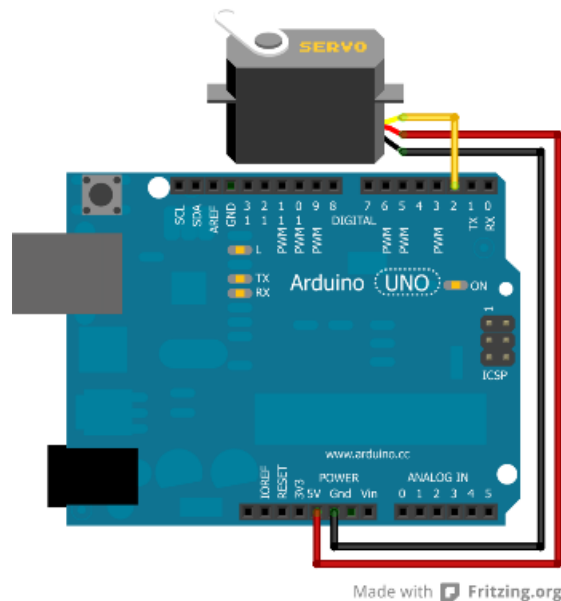
Bon, eh bien à présent, voyons un peu comment utiliser ces moteurs dont je vous vante les intérêts depuis tout à l'heure. Vous allez le voir, et ça ne vous surprendra même plus, la facilité d'utilisation est encore améliorée grâce à une bibliothèque intégrée à l'environnement Arduino. Ils nous mâchent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

Câblage

Nous l'avons vu plus haut, la connectique d'un servomoteur se résume à trois fils : deux pour l'alimentation positive et la masse et le dernier pour le signal de commande. Rappelons qu'un servomoteur accepte généralement une plage d'alimentation comprise entre 4.5V et 6V (à 6V il aura plus de couple et sera un peu plus rapide qu'à 4.5V). Si vous n'avez besoin d'utiliser qu'un ou deux servomoteurs, vous pouvez les brancher sur la sortie 5V de la carte Arduino. Si vous voulez en utiliser plus, il serait bon d'envisager une alimentation externe car le régulateur de l'Arduino n'est pas fait pour délivrer trop de courant, vous risqueriez de le cramer. Dans ce cas, n'oubliez pas de relier la masse de l'alimentation externe et celle de l'Arduino afin de garder un référentiel électrique commun. Le câble permettant le transit du signal de commande du servo peut-être branché sur n'importe quelle broche de l'Arduino. Sachez cependant que lorsque nous utiliserons ces derniers, les sorties 9 et 10 ne pourront plus fournir un signal PWM (elles pourront cependant être utilisées comme de simples entrées/sorties numériques). C'est une des contraintes de la bibliothèque que nous allons utiliser.

Ces dernières contraintes s'appliquent différemment sur les cartes MEGA. Cette page vous dira tout !

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



La librairie Servo

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

PRÉPARER LE TERRAIN

Tout d'abord, il nous faut inclure la librairie dans notre sketch. Pour cela vous pouvez au choix écrire vous même au début du code `#include <Servo.h>` ou alors cliquer sur *library* dans la barre de menu puis sur "Servo" pour que s'écrive automatiquement et sans faute la ligne précédente. Ensuite, il vous faudra créer un objet de type Servo pour chaque servomoteur que vous allez utiliser. Nous allons ici n'en créer qu'un seul que j'appellerai "monServo" de la manière suivante : `Servo monServo;`. Nous devons lui indiquer la broche sur laquelle est connecté le fil de commande du servo en utilisant la fonction `attach()` de l'objet Servo créé. Cette fonction prend 3 arguments :

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

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

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

UTILISER LE SERVO

Une fois ces quelques étapes terminées, notre servo est fin prêt à être mis en route. Nous allons donc lui donner une consigne d'angle à laquelle il doit s'exécuter. Pour cela, nous allons utiliser la fonction prévue à cet effet : `write()`. Tiens, c'est la même que lorsque l'on utilisait la liaison série ! Eh oui. 😊 Comme son nom l'indique, elle va *écrire*

quelque chose au servo. Ce quelque chose est l'angle qu'il doit donner à son axe. Cette fonction prend pour argument un nombre, de type `int`, qui donne la valeur en degré de l'angle à suivre. Si par exemple je veux placer le bras du servo à mi-chemin entre 0 et 180°, j'écrirais :

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

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

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

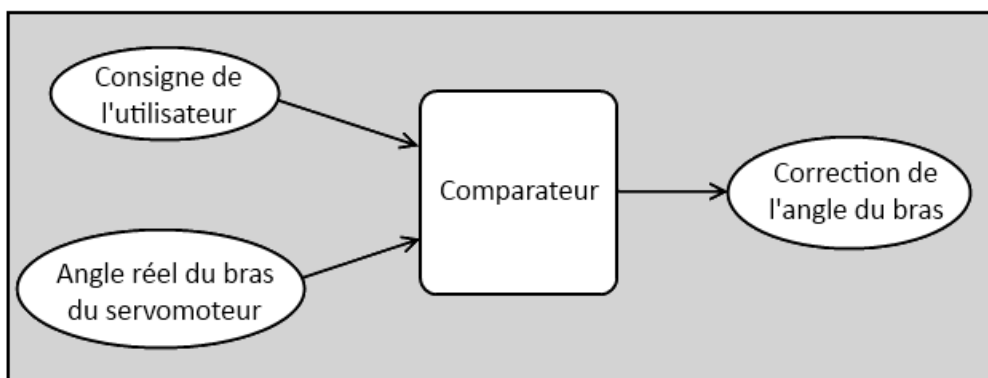
J'ai mis l'ordre de l'angle dans la fonction `setup()` mais j'aurais tout autant pu la mettre dans la `loop()`. En effet, lorsque vous utilisez `write()`, la valeur est enregistrée par Arduino et est ensuite envoyée 50 fois par seconde (rappelez vous du 50Hz du signal 😊) au servo moteur afin qu'il garde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

L'électronique d'asservissement

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

Principe de fonctionnement

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



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

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

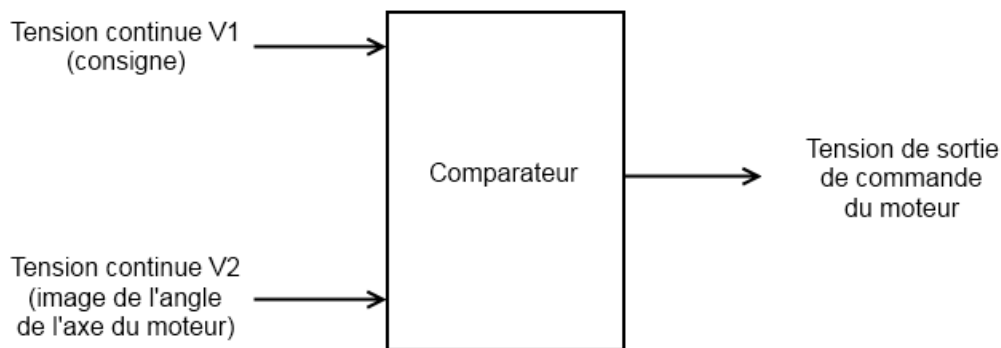
Avant d'aller plus loin, il faut savoir que les servomoteurs analogiques du commerce emploient en fait, dans leur électronique de commande, un microcontrôleur. Je ne vais donc pas vous expliquer comment ceux-là fonctionnent, mais je vais prendre le montage le plus basique qui soit. D'ailleurs, à l'issue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

Électronique à consigne manuelle

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

SYNOPTIQUE DE L'ÉLECTRONIQUE INTERNE

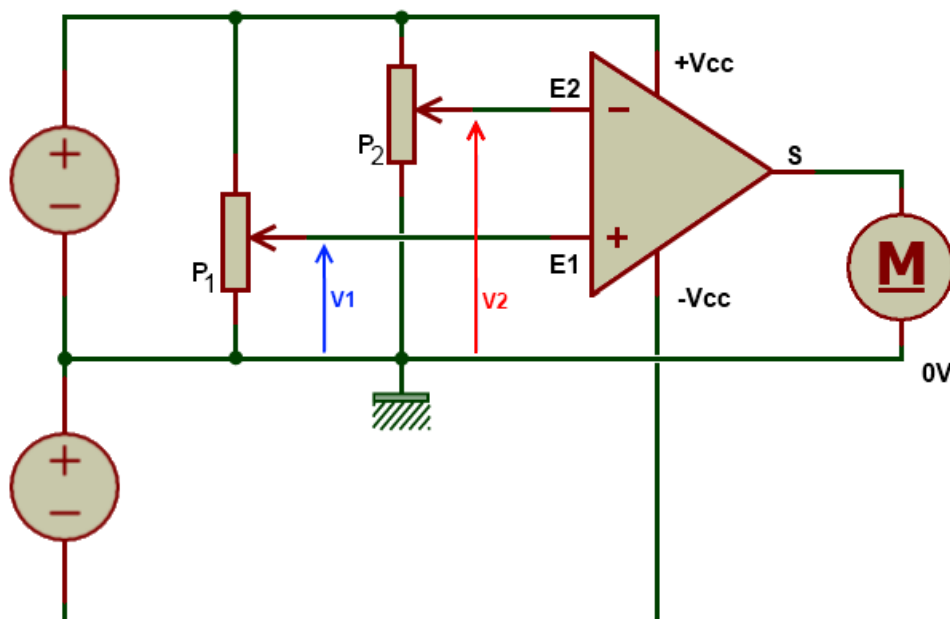
Commençons par un petit synoptique qui établit le fonctionnement de l'électronique de contrôle :



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

SCHÉMA DE PRINCIPE

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



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

FONCTIONNEMENT DU COMPARATEUR

Un comparateur est un composant électronique de la famille des circuits intégrés car, il contient en vérité d'autres composants, essentiellement des semi-conducteurs (diodes, transistors) et des résistances. **Ce composant a toujours besoin d'une alimentation externe pour fonctionner, c'est à dire qu'on ne peut lui mettre des signaux à son entrée que s'il est alimenté.** Autrement il pourrait être endommagé (c'est pas souvent le cas, mais mieux vaut être prudent). Vous

le constatez par vous-même, le comparateur est un composant qui possède deux entrées et une sortie. Et, de la manière la plus simple qui soit, en fait il n'y a rien de plus simple qui puisse exister, son fonctionnement réside sur le principe suivant :

- Si la tension (je me base par rapport au schéma) $V1$ qui arrive sur l'entrée $E1$ du comparateur est supérieure à la tension $V2$ qui entre sur l'entrée $E2$ du comparateur, alors la tension en sortie S du comparateur est égale à $+V_{cc}$ (l'alimentation du comparateur).
- Tandis que dans le cas opposé où la tension $V2$ va être supérieure à $V1$, la sortie S du comparateur aura une tension égale à $-V_{cc}$.

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

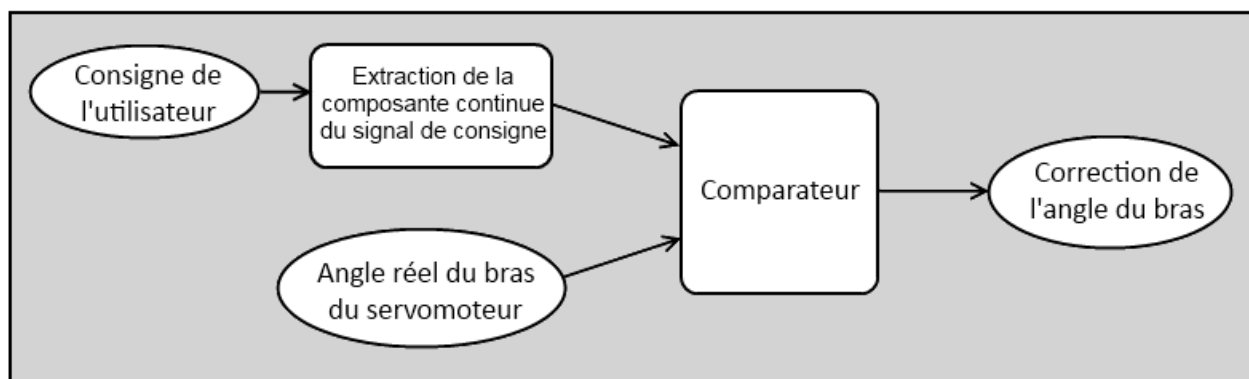
- Si $V1 > V2$, alors $V_s = +V_{cc}$
- Si $V1 < V2$, alors $V_s = -V_{cc}$

Comment s'en rappeler ? Eh bien grâce aux petits symboles "+" et "-" présents dans le triangle représentant le comparateur. La sortie du comparateur prendra $+V_{cc}$ si la tension sur l'entrée "+" du comparateur est supérieure à celle sur l'entrée "-" et inversement. Voyez, j'avais dit que c'était ultra simple. 😊 Il y a encore quelque chose à savoir : Il est impossible que les tensions $V1$ et $V2$ soient égales ! Oui, car le comparateur ne peut pas fournir une tension positive ET une tension négative en sa sortie, c'est pourquoi, même si vous reliez $E1$ et $E2$ avec un fil, la tension en sortie du comparateur sera toujours OU $+V_{cc}$ OU $-V_{cc}$.

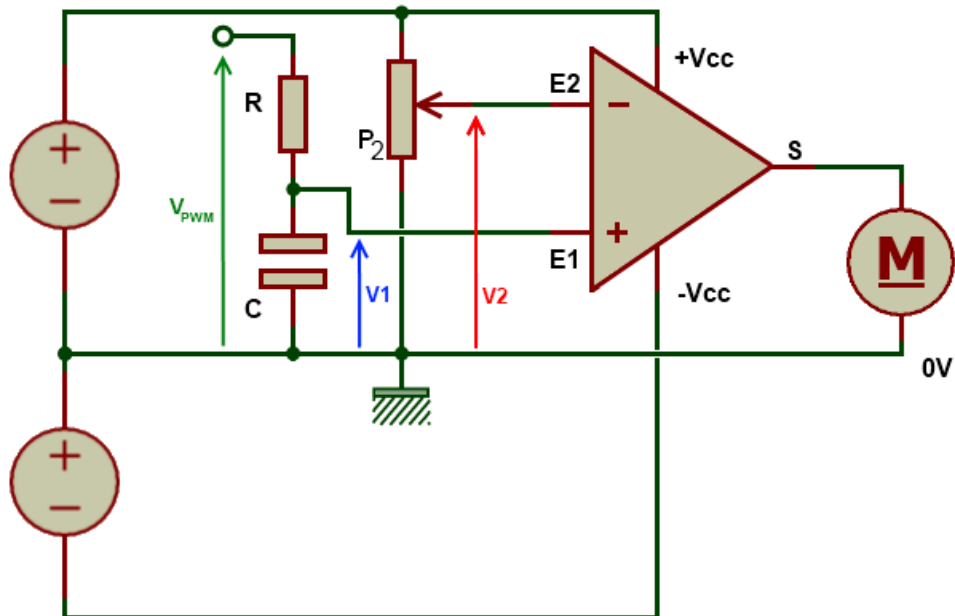
Électronique à consigne PWM

SYNOPTIQUE DE PRINCIPE

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



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



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

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

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

Et pour l'électronique à consigne PPM alors ? o_O

Pour ce type d'électronique de commande (présent dans tous les servos du commerce), je vous l'ai dit : il y a utilisation d'un microcontrôleur. Donc tout se fait par un programme qui scrute la position réelle du bras du moteur par rapport à la consigne PPM qu'il reçoit. Je n'ai donc rien d'intéressant à vous raconter. 😊

Un peu d'exercice !

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

CONSIGNE

Nous allons utiliser trois éléments dans cet exercice :

- un servomoteur (évidemment)

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

OBJECTIF

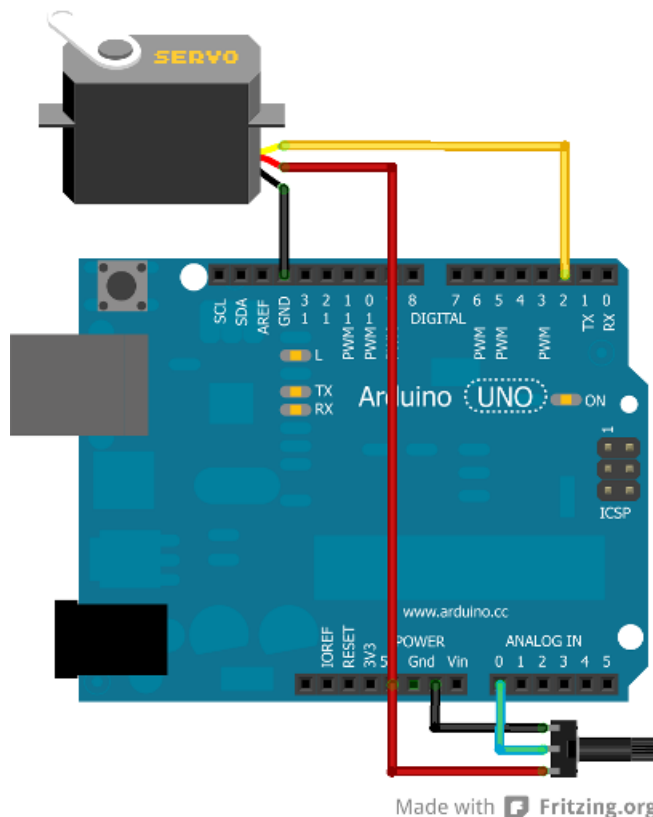
Le servo doit “suivre” le potentiomètre. C’est à dire que lorsque vous faites tourner l’axe du potentiomètre, le bras du servomoteur doit tourner à son tour et dans le même sens. Pour ce qui est de l’utilisation de la liaison série, je veux simplement que l’on ait un retour de la valeur donnée par le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

Utiliser un servo-moteur - Tutoriel Arduino



Correction

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



Made with Fritzing.org

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

```

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

```

Voilà qui est fait pour les préparatifs, il n'y a plus qu'à travailler un tout petit peu pour faire la logique du code. Commençons par la lecture analogique que nous allons renvoyer sur le servo ensuite. Le potentiomètre délivre une tension variable de 0 à 5V selon sa position. La carte Arduino, elle, lit une valeur comprise entre 0 et 1023. Ce nombre est stocké au format `int`. Il faut ensuite que l'on donne à la fonction qui permet d'envoyer la consigne au servo une valeur comprise entre 0 et 180°. On va donc utiliser une fonction dédiée à cela. Cette fonction permet de faire le rapport entre deux gammes de valeurs ayant chacune des extrêmes différents. Il s'agit de la fonction `map()` (nous en avons parlé dans le chapitre sur les lectures analogiques) :

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

Avec pour correspondance :

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

Nous utiliserons cette fonction de la manière suivante :

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


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

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

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

FONCTION `LOOP()`

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

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

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

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

Ou bien la version utilisant le produit en croix :

Secret: Réponse

Selectionner

Afficher

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

ET LA LIAISON SÉRIE

Pour renvoyer la valeur, rien de bien sorcier :

```
1 Serial.println(angle);
```

CODE FINAL

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

Secret: Réponse

Selectionner

Afficher

Je vous laisse mixer avec les différents codes que l'on vous a donné pour que vous fassiez celui qui vous convient le mieux (avec la fonction `map()`, ou bien celui qui est tout condensé, etc.). Dorénavant, vous allez pouvoir vous amuser avec les servomoteurs ! 😊

Tester un servomoteur “non-standard”

C'est déjà la fin ? o_O

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

WRITEMICROSECONDS()

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

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

READ()

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

Programme de test

En préparant ce chapitre, j'ai pu commencer à jouer avec un servomoteur issu de mes fonds de tiroirs. N'ayant bien entendu aucune documentation sur place ou sur internet, j'ai commencé à jouer avec en assumant qu'il utiliserait des valeurs "standards", donc entre 1000 et 2000µs pour l'état haut de la PPM. J'ai ainsi pu constater que mon servo fonctionnait, mais on était loin de parcourir les 180° attendu. J'ai donc fait un petit code utilisant une des fonctions précédentes pour tester le moteur en mode "pas à pas" et ainsi trouver les vrais timings de ces bornes. Pour cela, j'ai utilisé la liaison série. Elle m'a servi pour envoyer une commande simple ('a' pour augmenter la consigne, 'd' pour la diminuer). Ainsi, en recherchant à tâtons et en observant le comportement du moteur, j'ai pu déterminer qu'il était borné entre 560 et 2130 µs. Pas super proche des 1 et 2ms attendues ! 😞 Comme je suis sympa (😊), je vous donne le code que j'ai réalisé pour le tester. Les symptômes à observer sont : aucun réaction du servo (pour ma part en dessous de 560 il ne se passe plus rien) ou au contraire, du mouvement sans changement de la consigne (de mon côté, si on augmente au dessus de 2130 le servo va continuer à tourner sans s'arrêter).

```

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

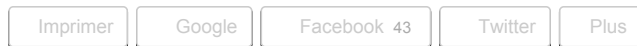
```

```
36 }  
37 }
```

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

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

Partager :



WordPress:



Soyez le premier à aimer cet article.

Archivé sous Partie 6 - Les moteurs | Taggé Arduino, Moteur, Tutoriel | 82 commentaires | Permalink

82 commentaires



DK

22/07/2014 at 00:11

Super Tuto... pardon, à ce niveau-là, on dit "cours"... Donc : "Super cours".

Pour l'aniv' de mon fiston, je me suis mis à l'Arduino et au servomoteur. J'ai entrepris (à mes risques) de lui confectionner un viseur (avant-bras) d'Ironman. J'ai hésité entre le capteur de pression, d'étirement, le bouton et le potard... De tous les tutos que j'ai trouvé, la version potard que tu propose semble être la meilleure. Il a 7 ans dans quelques jours et il m'aide à réaliser son gadget. Je suis (je peux l'affirmer) de dire que ça fonctionne et que j'ai impressionné mon bambin avant même d'avoir fini la conception. Mais au passage, je crois que le cadeau est plus de participer que de recevoir le truc tout fini. Il adore. Il m'aide dans le dessin et dans le calibrage. On est passé de 90° à 45° puis 55° pour une "visée plus précise" ^_^

Le fait de se contenter de plier le poignet pour sortir le viseur le fascine. J'adore le voir dans cet état.

Big Up to Eskimon en plus pour un cours super bien réalisé.

Les formules distillées aideront même dans d'autres domaines...

...IJK...



Eskimon

22/07/2014 at 08:31

Merci de ton retour, ça fait franchement plaisir de voir des applications concrètes dans la vraie vie et de beau récit comme le tien 😊



chafik

17/08/2014 at 12:30

Petit erreur sur la partie traitant de la relation liant le couple et la distance au poids, je pense que la formule devrait être $Poids(max) = C/(d \cdot g)$, g la constante gravitationnelle

**chafik**

17/08/2014 at 12:39

Autant pou moi je viens de voir les usitées attribuée au couple

Pingback: Pilotage d'un moteur avec arduino et raspberry pi - les notes de thierry

**Philippe**

12/10/2014 at 19:04

Bonjour

Que de travail !!!!

Bravo

J'aurai besoin d'un petit complément. Lors de la déclaration du potar, tu utilises "0" et pas "A0".

Pour quelle(s) raison(s) ?

Merci d'avance

**Eskimon**

13/10/2014 at 08:18

C'est simplement une question de nomenclature. Les noms symboliques des entrées analogiques sont Ax plutôt que 0, 1, 2...

**Philippe**

13/10/2014 at 19:22

Merci beaucoup

**Matt**

03/11/2014 at 17:37

Bonjour,

Tout d'abord un GRAND merci pour ce cours; vraiment chapeau c'est très complet et pourtant compréhensible pour un débutant comme moi.

Puis-je me permettre 2 questions?

1- Au "repos" (donc pas de commande) le servo consomme t-il?

2- Si je souhaite mettre le servo à 90° (par ex) pendant 2sec, il suffit de configurer l'état haut à 1.5ms pendant 2sec puis à 1ms (ou moins) pour qu'il retourne à 0° angle?

L'idée est de s'en servir pour "réarmer" un mécanisme à ressort.

Merci 😊

**Eskimon**

04/11/2014 at 09:31

2- Si je souhaite mettre le servo à 90° (par ex) pendant 2sec, il suffit de configurer l'état haut à 1.5ms pendant 2sec puis à 1ms (ou moins) pour qu'il retourne à 0° angle?

Yep ! C'est tout bon

1- Au "repos" (donc pas de commande) le servo consomme t-il?

Il faudra mieux définir "repos". Si par repos tu veux dire "a une position x sans bouger" alors la reponse sera "ca depend, est-ce qu'il y a une force qui le contraint a exercer une force contraire pour rester a cette position ?" . En soi, tant que le servo force pas il ne consomme pas vraiment.



Matt

04/11/2014 at 09:51

Merci beaucoup pour cette réponse!

Effectivement je veux dire "sa position de départ (0°) sans bouger" et sans force appliqué dessus.
L'idée est donc d'activer le servo le temps de réarmer le mécanisme puis revenir à 0° et attendre sans consommer (ou le moins possible).

Encore merci, c'est super un blog vivant et réactif comme ça 😊



Géryko

04/11/2014 at 15:52

Bonjour,

Très bien ce site.

Seul problème, ce genre de bouton à gauche, en milieu de page qui gêne la lecture.

Il faut absolument le déplacer, c'est insupportable.

Le bouton bouton bleu pour retourner en haut de page est limite.

Cordialement



Eskimon

04/11/2014 at 18:29

Ah mince, moi qui voulait faire plaisir 😞 Pour le coup c'est délicat, c'est le premier retour négatif que j'ai a ce sujet, d'habitude les lecteurs le trouve pratique pour naviguer dans les chapitres.



Jean-Louis

05/11/2014 at 16:41

Bonjour

Très bon tuto !

Pour une application permettant de motoriser une tête panoramique sphérique pour réaliser des photos sphériques, j'aimerais que le changement d'angle se fasse en douceur pour ne pas brutaliser la mécanique et l'appareil photo.

Pour cela il faudrait que la vitesse de changement d'angle suit une rampe d'accélération et de décélération. Cette possibilité est-elle incluse dans le code "servo", ou doit-on la programmer de toute pièce.

Merci d'avance

Cordialement

**Eskimon**

05/11/2014 at 16:49

Et non, par défaut cette fonction n'existe pas, par contre je serais prêt à mettre ma main au feu en disant que ça a sûrement dû être fait par qqun et que du coup une bibliothèque doit le permettre 😊 (comme `AccelStepper` par exemple 😊)

**Matt**

14/11/2014 at 10:04

Re bonjour,

Est-il possible de régler la vitesse de rotation d'un servo? Je ne suis pas sûr de comprendre tout le chapitre...

J'ai acheté un servo "pas cher" et souhaiterais le ralentir... y'a-t-il une solution? (via arduino MEGA)

D'avance merci et encore bravo pour ce cours!

Matt

**Eskimon**

14/11/2014 at 10:57

Pour un servo "pas cher" (que je suppose analogique donc) non ce n'est pas possible. Seules les servos numériques (plus cher) le permettent.

**Matt**

14/11/2014 at 11:35

Ah mince 😞

Ok merci, il me semblait avoir lu une astuce qui consiste à faire des "micro pauses" entre chaque 1° angle?

Je vais voir si je trouve un compromis...

Encore merci 😊

**Eskimon**

14/11/2014 at 12:18

Oui effectivement, pour le ralentir faire des mini-pauses entre chaque degré (ou carrément en écrivant des tout petits pas avec `writeMicroseconds()`) peut fonctionner, mais ça demandera un peu de code ^^ .

**Matt**

14/11/2014 at 12:43

Je vais tester tout ça dès que possible et viendrais faire un retour si ça aide? 😊

Merci!

Pingback: Un moteur qui a de la tête : le Servo-Mot...

**marion**

28/12/2014 at 19:13

Bonjour,

Je ne comprends pas très bien comment obtenir un signal à partir de la carte arduino pour le mettre en consigne à l'entrée du comparateur ...

je comprends comment commander le servomoteur avec la carte arduino mais je vois pas trop comment commander le servomoteur à partir de la carte arduino lorsque l'on rajoute le comparateur...

Pourriez-vous m'expliquer ?

Cordialement,

Marion

**Eskimon**

29/12/2014 at 10:22

Toute cette histoire de comparateur est une tambouille interne au servo-moteur. L'important point de vue Arduino est "simplement" de générer le signal de position (état haut de 1 à 2ms etc) puis ensuite le servo le récupère et s'en occupe pour l'utiliser comme consigne à atteindre/respecter.

(Mais j'ai peut être mal compris la question)

**s3**

29/12/2014 at 18:05

Salut, tout d'abord merci pour ce cours bien complet!

par contre j'ai un petit soucis, je suis en train de monter un verrou biometrique à l'aide d'un lecteur d'empreinte digitale (le GT511C3) et le servo (S3003) fait office de verrou, au repos il est à 0° et lorsque l'empreinte digitale est reconnue le servo se place à 90°.

Mais au repos, que ce soit à 0° ou 90° le servo "tremble", j'ai tenté d'utiliser une alim externe sur mon arduino mais le tremblement est toujours là, j'ai aussi tenté sans utiliser la bibli servo, à l'aide de la fonction analogWrite, toujours le même problème...

T'as une idée?

P.S.: j'ai testé le servo seul et il fonctionne très bien

**Eskimon**

30/12/2014 at 09:25

Si le servo seul fonctionne bien, alors c'est que la consigne est bien applique. Du coup s'il tremble "en application", c'est peut-être qu'un effort trop important est applique dessus et du coup il a du mal à maintenir sa correction.

**s3**

30/12/2014 at 12:58

C'est bien le soucis, car rien ne force sur le servo car il y a juste un crochet qui se met en position

**Eskimon**

30/12/2014 at 13:13

Si rien ne force dessus et que le servo isole fonctionne bien alors il y a une part de magie quelques part... 😊
(ou ton signal est pas propre à cause de quelque chose de nouveau dans le système)

(sinon analogWrite ne sert a rien dans le cas d'un servo)

**s3**

31/12/2014 at 14:57

```
j'ai partiellement résolu le problème avec la fonction detach()
j'attache le servo pour faire sa rotation et le detache dès que c'est fait.
if (id == 0)
{
verrou.attach(6, 550, 2330);
digitalWrite(ledv, LOW);
digitalWrite(ledr, HIGH);
verrou.write(90);
delay(1000);
verrou.detach();
digitalWrite(ledv, HIGH);
digitalWrite(ledr, HIGH);
}
ça permet aussi "d'économiser" l'énergie de l'accu dans le cas d'un circuit autonome
```

**beneddaman**

18/02/2015 at 17:11

Bonjour;

Tout d'abord je vous félicite et je vous remercie pour ce cours; vraiment rien à dire c'est très très complet et compréhensible. je veux commander un bra robotique a l'aide de 4 servomoteurs HS-422 avec Labview !! donc je voudrais savoir comment puis je crée le beau signal de commande sous Labview ?

**Eskimon**

19/02/2015 at 09:13

Aucune idée 😞 ! Je n'ai jamais utilise LabView ou d'autres outils de programmation graphique avec Arduino.

**Sam**

21/02/2015 at 16:55

Bonjour, super cette page!!!

J'ai une question assez précise: serait-il possible, grâce au arduino, d'enregistrer des mouvements de plusieurs servos pilotés simultanément par une radio commande comme celle utilisée dans le modélisme. Pour ensuite refaire ces mouvements automatiquement? J'espère avoir été assez clair 😊

Merci d'avance

**Sam**

21/02/2015 at 17:12

(via un récepteur multi-voies de modélisme)

**Eskimon**

23/02/2015 at 08:41

L'idée serait de "lire" les commandes reçues sur le récepteur ? C'est possible mais ça sera pas forcément simple (parce que les histoires de timing c'est jamais marrant)

**Sam**

23/02/2015 at 11:27

Oui, exactement. Et de pouvoir ensuite lire la "séquence" sans moi. J'avais acheté un mini maestro usb servo controller de chez Pololu mais on ne peut qu'écrire un script ou enregistrer position par position (sans utiliser de radiocommande) ce qui ne convient pas pour ce que je veux faire. J'ai trouvé une vidéo qui illustre bien mon besoin:

<https://www.youtube.com/watch?v=39dR1flPooA>

Je ne suis pas électronicien, donc je suis un peu limité 😊

Merci

**Lilyan**

06/03/2015 at 22:38

Et pour programmer un potentiomètre linéaire avec retour de force sur un moteur à CC, vous avez une idée ?

**Eskimon**

09/03/2015 at 15:07

C'est à dire ? Je ne vois pas très bien de quoi l'on parle... (je ne suis pas sûr de voir à quoi le retour de force fait référence... Car programmer en utilisant un potentiomètre et un moteur CC c'est dans le cours, après il *suffit* d'adapter aux différentes situations 😊)

**Lilyan**

09/03/2015 at 18:59

Bonjour, merci pour la réponse et merci pour le cours. Par retour de force je voulais dire retour au point de départ (comme un volant)...

**Eskimon**

16/03/2015 at 13:09

Donc le moteur est montée *sur* le potentiomètre et s'oppose un peu au mouvement ? Comment est piloté le moteur etc ? Je te conseille d'ouvrir un sujet sur le forum "Systeme et Materiel" de Zeste de Savoir, ça sera plus simple pour en parler, illustrer etc...

**Jules Playa**

13/03/2015 at 15:15

Salut Eskimon,

Pour commencer c'est un super tuto que tu as fais là, je n'y connaissais rien en servomoteur et en programmation et maintenant je m'en sort un peu près merci à toi !

Alors voila, avec un copain nous arrivons à allumer/éteindre deux leds via un smartphone (par Bluetooth/arduino) ainsi que changer le sens de variation du servomoteur lorsque l'utilisateur appuie sur le bouton 1 ou le bouton 2.

Le problème c'est que je n'arrive pas à faire tourner le servomoteur pendant un certain temps.. Il tourne à l'infini. Ou même ordonner de tourner avec un certain nombres de degrés, nous utilisons un servomoteur a rotation continue !

Tout ce qui est câblage, programme, carte arduino ainsi que la liaison Bluetooth marchent très bien, mais nous

sommes bloqué juste par ce que nous ne savons et n'arrivons pas à faire tourner le Servomoteur sur une courte distance, il tourne seulement à l'infini ce qui ne va pas.

Voilà si tu as une petite idée pour résoudre ce problème merci d'avance 😊

**Eskimon**

16/03/2015 at 13:18

Les servo moteurs à rotation continue ne sont pas faits pour comprendre qu'il faut tourner seulement de quelques degrés. Il faut donc que tu calcul le temps qu'il faut laisser tourner le servo (et à quelle vitesse) pour déterminer la rotation faite.

Pingback: Arduino & Raspberry | Pearltrees

Pingback: Techno | Pearltrees

**Jules Playa**

16/03/2015 at 13:37

C'est donc ce que j'ai fait, mais je ne sais pas comment l'arrêter.

Je n'arrive pas et ne sais pas comment intégrer la rotation à effectuer..

J'ai essayé avec Delay()

Je veux juste le faire tourner une seconde avec le bouton 1 et de trois secondes avec le bouton 2 à une vitesse de 1 tout par seconde.

**Eskimon**

19/03/2015 at 08:22

Si on fait tourner le moteur à une vitesse de 1 tour en 2 secondes, alors cela signifie que l'on fait 360 degrés en 2 secondes, ou 180 degrés en 1 seule seconde. Donc pour faire par exemple 45 degrés il nous faudra $1/(180/45) = 0.25$ seconde.

Ça aide un petit peu ou pas ?

**Léo**

19/03/2015 at 17:04

Merci Monsieur pour ce travail .

Je n'arrive pas à comprendre comment , en écrivant la commande `monservo.attach` dans 'setup' ça me génère un signal à une fréquence de 50Hz???

**Léo**

21/03/2015 at 09:29

svp j'ai pas compris ça

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

merci

**Eskimon**

23/03/2015 at 09:14

Ce que je veux dire c'est qu'une fois qu'un angle est réglé, ce dernier est envoyé *automatiquement* 50 fois par secondes, tant qu'on ne le change pas.

**meid meir**

29/03/2015 at 19:06

Bonsoir, je tiens à vous remercier pour le partage de votre savoir et j'en profite pour vous posez ces questions:
1/ Comment procède-t-on au dimensionnement d'un servomoteur(c'est à dire son choix en fonction de l'application à laquelle est destiné);
2/ comment faire le choix de la résistance de limitation du courant dans un circuit (par exemple celle branchée avec une LDR)

**Eskimon**

30/03/2015 at 08:11

1. Il faut avoir une idee des contraintes finales (couple a déployer, vitesse a atteindre, encombrement, alimentation disponible...). Le plus important étant surement le couple (plus il faut de couple, plus il faudra un servo costaud).
 2. Pour faire un diviseur de tension avec la LDR ? Alors il faut appliquer les calculs du pont diviseur de tension pour obtenir l'intervalle de mesure que l'on désire.
-

**Tournie**

30/03/2015 at 10:19

Bonjour,

J'ai pris tes lignes de code pour m'aider dans ma mon projet ou je dois faire pivoter une lame séparatrice de 180° pour celà j'utilise un servo moteur où la lame est monté sur son axe de rotation.

Je récupère la valeurs de la position du servo moteur en degrés, suivant la position le logiciel hésite dans sa position et du coup sur un déplacement d'un degrés il fluctut, comment je peux resoudre ce problème.

#include

```
const int potar = 0; //notre potentiomètre
Servo monServo;
```

```
void setup()
{
  //on déclare le servo sur la broche 2 (éventuellement régler les bornes)
  monServo.attach(2);
  //on oublie pas de démarrer la voie série
  Serial.begin(9600);
}
```

```
void loop()
{
  monServo.write(map(analogRead(potar), 0, 1023, 0, 171
```

```
));  
int val = analogRead(pot);  
//on converti la valeur lue en angle compris dans l'intervall [0;180]  
int angle = val / 5.7;  
Serial.println(angle);  
  
}
```

**Eskimon**

30/03/2015 at 12:48

Si je comprend bien, tu as un potentiomètre sur l'axe de pivot de la scie mais la mesure de ce dernier manque de précision ?

La solution simple mais pas garanti : Au lieu de faire une seule mesure, en faire plusieurs (5 ? 10 ? 20 ?) et en faire la moyenne.

Une solution moins simple serait d'augmenter la précision en décuplant mécaniquement la course du potentiomètre (en prenant un potentiomètre multi-tour et faisant une réduction pour que les 180 degrés s'étale sur tout la course du potentiomètre.

**Faoro**

31/03/2015 at 07:52

Que de travail ! bravo

Je dois pour une de mes constructions développer un test de torsion. Je voudrais savoir si l'utilisation d'un servo-moteur pourrait être utile ?

Dans mon cas, je pense que oui, pour contrôler l'angle de torsion.

Quelle est la plage de moment que peux fournir un servo-moteur ?

Merci d'avance pour votre réponse

**yreful9**

03/04/2015 at 14:58

Bonjour! j'ai trouvé ce tuto très utile!

Merci beaucoup!

Je veux juste savoir si je peux alimenter 2 servo moteur :

(l'un un MG 995 et l'autre un SG90) directement avec le V+ (5V) de Arduino sans utiliser une alimentation externe pour mon projet?

Merci d'avance pour vos réponses!

**Eskimon**

03/04/2015 at 15:29

Difficile à dire sans savoir quel est le courant qu'ils vont consommer. Si c'est pour des déplacements demandant peu d'efforts ça passera, sinon mieux vaut prévoir l'alimentation externe (le régulateur internet 5V arduino pouvant supporter environ 1A max)

**samsam2**

08/04/2015 at 08:19

merci pour le cour

Pingback: Motoriser une fenêtre et commander son ouverture | Vahagn's Blog

**Fred**

22/04/2015 at 15:09

Extremement interessant, j'ai l'impression d'être retourné à l'école...

**Eskimon**

30/04/2015 at 10:52

Dans le bon sens du terme j'espère 😊

**koumo**

01/05/2015 at 11:19

Extremement interessant et instructif

**Azar**

26/05/2015 at 10:45

s'il vous plait, je veux savoir si je peux commander mon Servomoteur avec les ports PWM et sans utiliser la bibliothèque

**Eskimon**

27/05/2015 at 09:21

Un servomoteur ne se pilote pas avec une PWM classique, il faut un signal particulier.

**Christophe**

26/05/2015 at 11:05

Bonjour Eskimon as tu déjà essayé LIFA ?

**Eskimon**

27/05/2015 at 09:21

LIFA...Ca me dit quelque chose mais je n'arrive pas à me rappeler quoi...

**Christophe**

27/05/2015 at 16:22

Labview Interface For Arduino

**Eskimon**

28/05/2015 at 12:25

Ah bah non ca ne me disait rien finalement. Donc non, je connais le nom et le but mais pas plus que ca (et n'aime pas particulièrement ce système avec LabView)

**Olivier**

01/06/2015 at 19:03

Merci pour ces explications précises et précieuses pour les débutants!

Je suis en train de tester divers montages très simples avec une carte arduino uno et sans vos explications j'aurais bien peiné à comprendre pourquoi mon servo ne fonctionnait pas avec l'exemple "knob"!

Grâce à cette page, j'ai pu comprendre le problème et déterminer les durées min/max du signal de commande propre à mon servo (MPX Nano-S: de 630 à 2350) pour les "mapper" avec l'intervalle [0, 1023] (correspondant aux valeurs récupérées sur l'entrée analogique, liées à la position d'un potar).

**Eskimon**

02/06/2015 at 08:04

Super !! C'est vrai que le fait que les standards ne soient pas toujours respectées des constructeurs n'aide pas forcément non plus 😊

**Florian**

07/06/2015 at 18:02

ouah ! Super ! Merci beaucoup ! J'ai toutes les informations dont j'ai besoin sur les servo-moteurs cette page 😊

Pour ma part, je réalise un projet dans le cadre de concours pour rentrer en école d'ingénieur.

Je réalise un système anti-débordement, en gros, je veux marcher avec un récipient ouvert sans jamais renverser le liquide.

J'utilise donc un servo-moteur pour incliner une plaque sur laquelle repose le récipient et c'est là que vous m'avez aidé !

Un grand merci 😊

**Joris**

02/07/2015 at 13:11

Bonjour, j'ai récupéré un servomoteur composé de 6 fils, 3 pour le potentiomètre, 3 pour le moteur, (d'ailleurs je ne sais pas à quoi sert le 3e). Ce servo ne comporte pas une carte électronique. Je voudrais savoir s'il serait possible de l'asservir facilement avec arduino ou s'il faut que j'envisage d'acheter un servomoteur standard à 3 fils.

**Eskimon**

03/07/2015 at 15:10

à 6 fils, ne serait-ce pas un servo-moteur numérique (qui se pilote avec un protocole genre I2C) ? Il y a une référence dessus ?

**Joris**

03/07/2015 at 15:23

je l'ai récupéré sur un voiture télécommandé nikko, non rien d'écrit dessus.

**meidmeir**

13/07/2015 at 19:19

salut, dans une application j'ai besoin d'utiliser plus qu'un servomoteur; lors de la recherche d'un circuit d'alimentation pour mes servo j'ai trouvé celui la "voir le lien", mais est ce que vous pouvez m'expliquer comment le choix de la capacité de 220 uF a été fait? ainsi que le rôle de la résistance R1 connecté avec l'un des deux servo

Lien de l'image: <http://www.instructables.com/file/FQ2Y408GWZMIIL9>

**Eskimon**

14/07/2015 at 17:14

La résistance semble être là pour limiter le courant dans la led (mais j'ai des doutes sur le schéma). Les condensateurs sont là pour filtrer les fluctuations de tensions lorsque le servo se met à bouger.

**meidmeir**

14/07/2015 at 21:51

Merci, pouvez vous me proposer un autre schéma ? mais comment est ce que le choix de la valeur de 220 uF a été fait ??

**martin**

23/08/2015 at 11:59

Très intéressant ! Merci

je me permet de vous signaler une petite erreur de copié/collé :

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

Si $V1 > V2$, alors $V_s = +V_{cc}$

Si $V1 < V2$, alors $V_s = -V_{cc}$

"

il faut inverser le comparateur sur la seconde ligne.

**quant**

24/08/2015 at 17:42

Bonjour,

j'ai 2 servomoteurs Parallaxes connectés à une Board of Education Shield (toujours Parallaxe).

Quand j'essaie de les faire tourner vers l'arrière, ou à gauche, ou à droite, rien ne marche.

Par exemple, ce programme est sensé faire rouler mon Board of Education Shield en avant puis tourner à gauche, puis tourner à droite et enfin aller à reculons :

```
// Robotics with the BOE Shield – ForwardLeftRightBackward
```

```
// Move forward, left, right, then backward for testing and tuning.
```

```
#include // Include servo library

Servo servoLeft; // Declare left and right servos
Servo servoRight;

void setup() // Built-in initialization block
{
  tone(4, 3000, 1000); // Play tone for 1 second
  delay(1000); // Delay to finish tone

  servoLeft.attach(13); // Attach left signal to pin 13
  servoRight.attach(12); // Attach right signal to pin 12

  // Full speed forward
  servoLeft.writeMicroseconds(1700); // Left wheel counterclockwise
  servoRight.writeMicroseconds(1300); // Right wheel clockwise
  delay(2000); // ...for 2 seconds

  // Turn left in place
  servoLeft.writeMicroseconds(1300); // Left wheel clockwise
  servoRight.writeMicroseconds(1300); // Right wheel clockwise
  delay(600); // ...for 0.6 seconds

  // Turn right in place
  servoLeft.writeMicroseconds(1700); // Left wheel counterclockwise
  servoRight.writeMicroseconds(1700); // Right wheel counterclockwise
  delay(600); // ...for 0.6 seconds

  // Full speed backward
  servoLeft.writeMicroseconds(1300); // Left wheel clockwise
  servoRight.writeMicroseconds(1700); // Right wheel counterclockwise
  delay(2000); // ...for 2 seconds

  servoLeft.detach(); // Stop sending servo signals
  servoRight.detach();
}

void loop() // Main loop auto-repeats
{ // Empty, nothing needs repeating
}
```

Mais évidemment ça ne marche pas, les servos tournent sans s'arrêter.

J'ai pourtant bien réglé les potentiomètres et les branchements sont parfaits, le programme l'est aussi.

Quelqu'un pourrait-il m'aider ?

Pingback: Actuateurs | Pearltrees

Pingback: Arduino | Pearltrees



Antonin

08/11/2015 at 17:46

Merci beaucoup pour ce cours très complet

étant en term S-SI j'ai un projet à réaliser pour le BAC ce cours va vraiment m'aider dans la compréhension des servomoteurs et de leur programmation sous Arduino.

Notre projet consiste à reproduire le mouvement du bras humain grâce à un bras robotisé (constitué de servomoteur). ce cours risque d'être ma Bible dans le début de la programmation. 😊

Et je risque de devoir trouver un cours sur les capteurs de flexion y a-t-il un cours sur ses derniers sur ce site ? (j'avoue ne pas avoir cherché mais ça ne va tarder).

Encore une fois Merci.

**Eskimon**

10/11/2015 at 11:55

Nop, je n'ai rien écrit au sujet des capteurs de flexion. Il me semble (sans certitude) que ces derniers se comportent comme une résistance variable non ?

**CHAHED Lyes**

16/11/2015 at 14:07

Merci, c'est trop top

j'ai une petite manipulation à réaliser qui consiste à stabiliser une balle sur une table actionnée par deux servomoteurs avec une webcam comme capteur de position, je voudrais la commander directement avec matlab et ssc 32, puis-je utiliser les commandes d'arduino sur matlab? es-ce que vous avez des astuces pour cette manipulation?

**nico59777**

20/12/2015 at 17:26

bonjour .

Merci pour toute cette instruction, très claire.

J'aurai 2 questions :

– comment peut-on gérer linéairement le mouvement ? (j'ai compris que l'on termine la position de chaque angle), mais peut-on gérer linéairement les 0-5V d'entrée et les traduire en 0-180 degrés ? sans décomposer.

– j'ai bien compris le servomoteur, petit électronique.

Peut-on utiliser la fonction servomoteur avec un actionneur linéaire ? mais du coup qui, je pense n'a pas la gestion de la copie de la position en interne ?

(type : [http://fr.aliexpress.com/item/Free-shipping-12V-50mm-2inch-stroke-1000N-Load-10mm-s-linear-actuator-with-feedback-potentiometer/1575014939.html?](http://fr.aliexpress.com/item/Free-shipping-12V-50mm-2inch-stroke-1000N-Load-10mm-s-linear-actuator-with-feedback-potentiometer/1575014939.html?spm=2114.06010108.3.1.KdFq6o&ws_ab_test=searchweb201556_1,searchweb201644_3_79_78_77_82_80_62,searchweb201560_8)

[spm=2114.06010108.3.1.KdFq6o&ws_ab_test=searchweb201556_1,searchweb201644_3_79_78_77_82_80_62,searchweb201560_8](http://fr.aliexpress.com/item/Free-shipping-12V-50mm-2inch-stroke-1000N-Load-10mm-s-linear-actuator-with-feedback-potentiometer/1575014939.html?spm=2114.06010108.3.1.KdFq6o&ws_ab_test=searchweb201556_1,searchweb201644_3_79_78_77_82_80_62,searchweb201560_8))

merci de votre retour ! et bravo !