Contrôle d'un servo-moteur depuis son PC

Bon il était tard quand j'ai fini l'article donc il contient quelques fautes... A ne pas trop prendre au pied de la lettre

Prérequis

- Processing, un logiciel de programmation ressemblant furieusement à Arduino

Lien pour le télécharger : https://processing.org/

- Un Arduino
- Un servo-moteur (??)
- Avoir des bases de C (créer des variables et les modifier) et un tout petit peu de programmation objet...
- Un tout petit peu de Java (mais ça ressemble à du C, que vous connaissez déjà)

I. Servo-moteur vous dites ?

Un servo-moteur est un moteur intelligent (il possède un servo...) qui est capable de se placer tout seul à une angle donné (compris entre 0 et 180). C'est avec ces composants que l'on peut faire des **panneaux solaires** suivant la trajectoire du Soleil ou des **caméras** balayant une pièce. Ici nous allons justement prendre le contrôle de l'un de ces engins avec l'aide du **clavier** de notre ordinateur, mais après ce tutoriel vous serez tout à fait capable de prendre le contrôle avec votre **souris** ou des **fruits** (c'est même plus simple !)

II. Ce que nous allons faire

Nous allons faire deux tutoriels en un seul :

1. Savoir contrôler un servo-moteur
2. Savoir envoyer depuis un ordinateur des données à un arduino

Avec comme but de contrôler un servo-moteur avec votre clavier

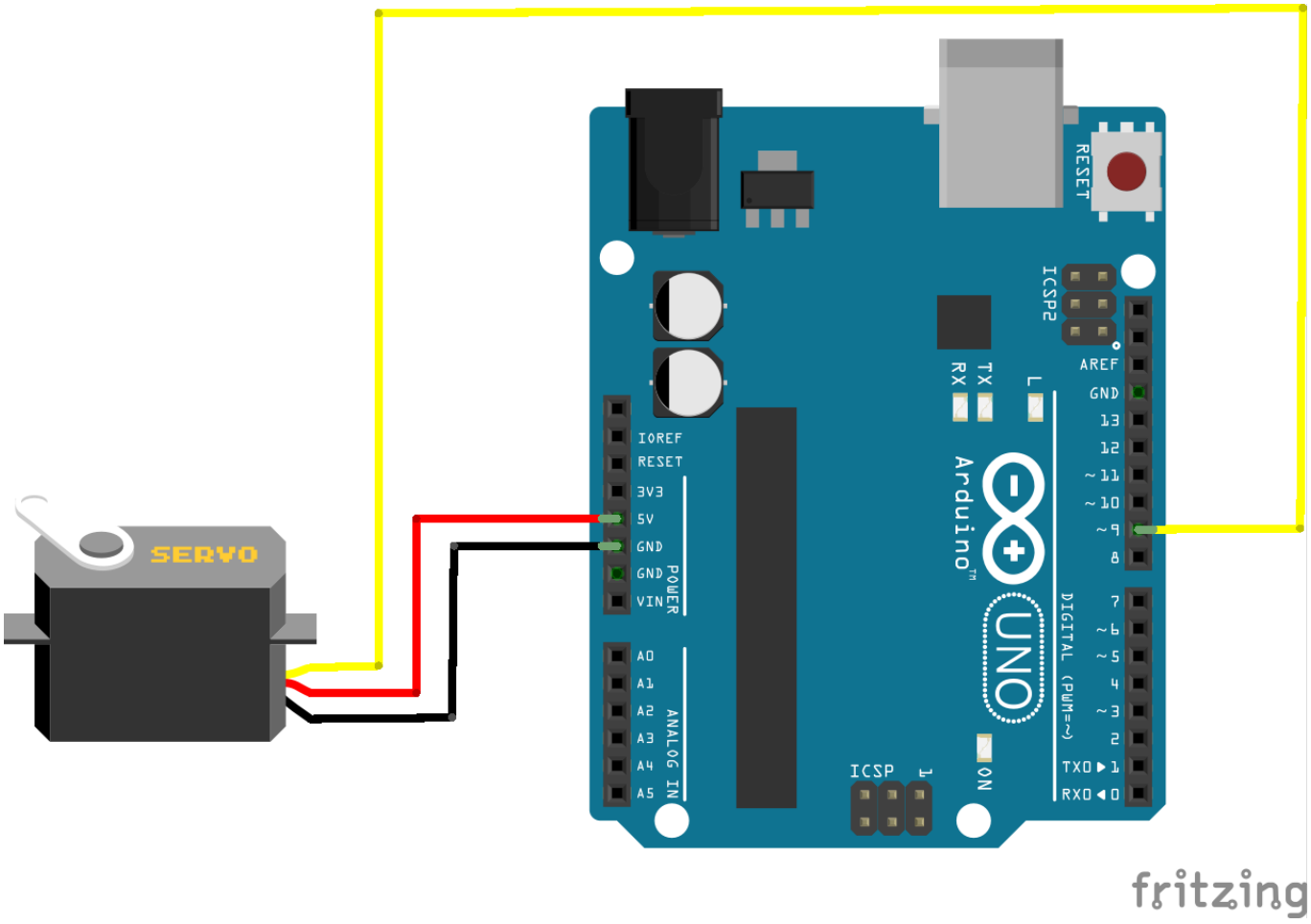
III. Le montage

Un servo-moteur possède trois entrées : deux correspondant à l'alimentation et une recevant spécialement les ordres. Vous pouvez reconnaître quelle entrée est laquelle en regardant la couleur du fil :

- Rouge : Alimenter en 5V
- Noir : Connecter la masse
- Autre (souvent jaune ou blanc) : C'est elle qui permet d'asservir le servo-moteur

Le montage (difficile (~~ironie~~)) :

Vous pouvez brancher la patte jaune à l'entrée que vous voulez, mais pour les besoins du tutoriel nous la placerons à l'entrée 9.



On peut désormais passer à la partie suivante :

IV. Le code arduino pour contrôler un servo-moteur

Nous avons besoin d'une bibliothèque pour utiliser le servo-moteur facilement : la bibliothèque Servo.h Cette bibliothèque permet d'utiliser très facilement un servo-moteur.

Cette bibliothèque désactive la possibilité d'utiliser les fonctions analogWrite() des sorties 9 et 10 (sauf sur une Mega...)

Pour utiliser les fonctions offertes par Servo.h vous devez implanter la ligne suivante en début de votre code Arduino :

```
#include <Servo.h>
```

Ici il n'y a pas de point virgule à la fin

Maintenant il faut créer un **objet** qui va permettre de commander votre servo-moteur. On va le nommer “monServo” et le créer de la façon suivante :

```
Servo monServo;
```

Un **objet** est une sorte de variable mais avec des fonctions plus puissantes. Si un variable est un verre contenant de l'eau, un objet est un distributeur de canettes capable de faire plein de choses (prendre la monnaie, vous donner votre boisson)

Dans votre boucle setup() vous aurez besoin de dire à votre objet “monServo” à quelle entrée est branché votre servo-moteur. Ici nous l'avons branché à l'entrée 9, donc nous utiliserons la fonction attach() qui s'utilise de la façon suivante :

```
monServo.attach(9);
```

Mais que se passe-t-il ?

Votre objet **monServo** est comme je le disais une variable avec des fonctions plus puissantes. C'est réellement une machine. Ici nous lui demandons, grâce à la fonction attach() de monServo, de se souvenir que monServo est l'objet représentant le servo-moteur branché à l'entrée 9.

Si vous voulez maintenant dire à votre servo-moteur de se placer à un angle de 45 degrés, vous n'avez qu'à utiliser la ligne suivante dans votre boucle loop :

```
monServo.write(45);
```

N'hésitez pas à tester d'autres valeurs ou à utiliser une variable pour contrôler votre servo-moteur...

Voalà le code entier :

```
#include <Servo.h>

Servo monServo;

void setup()
{
  monServo.attach(9);
}

void loop()
{
```

```
  monServo.write(45);
}
```

V. Le code pour interagir entre l'arduino et l'ordinateur

Nous savons commander un servo-moteur depuis l'arduino. Et si maintenant on le commandait depuis l'ordinateur ?

Le code s'organise en deux parties :

- Un code qui va être téléversé sur l'arduino, avec le logiciel Arduino
- Un code qui va être exécuté sur l'ordinateur, avec le logiciel Processing

La partie Arduino

Pour pouvoir faire communiquer un Arduino avec un PC, nous devons d'abord ouvrir une connexion. Cela se fait avec la fonction begin() de la bibliothèque Serial, de la façon suivante :

```
Serial.begin(9600);
```

Le fonctionnement de serial ressemble furieusement à celui du servo, et c'est normal : Serial est un objet qui représente tout simplement la connexion entre l'arduino et l'ordinateur. Contrairement à monServo de la partie précédente, Serial se crée tout seul dans tous vos codes Arduino. Le 9600 correspond en gros au nombre d'informations envoyées par seconde : il est important que l'ordinateur et l'arduino fonctionnent au même rythme.

Pour lire une valeur que l'on aurait envoyé à l'arduino depuis son ordinateur, il suffit d'enregistrer dans une variable ce que nous renvoie la fonction Serial.read(). Exemple d'utilisation :

```
int variable; // La variable pour sauvegarder ce que nous envoie
l'ordinateur
if(Serial.available()) //On vérifie qu'il y a quelque chose à lire
{
  variable = Serial.read(); // On enregistre dans variable ce que
l'ordinateur nous envoie.
}
```

Super ! On sait lire des données depuis un ordi !
Supposons maintenant que cette variable correspond à l'angle que l'on veut donner à notre servo-moteur, il suffit d'écrire dans la boucle loop() la ligne suivante :

```
monServo.write(variable);
```

Voici le code Arduino entier pour placer un servo-moteur à la position “variable” reçue depuis l'ordinateur, à téléverser dans votre carte Arduino avant de passer à la partie suivante :

```
#include <Servo.h>
```

```
Servo monServo;

void setup()
{
  monServo.attach(9);
  Serial.begin(9600);
}

void loop()
{
  int variable; // Je déclare la variable
  if(Serial.available()) // Je vérifie qu'il y a quelque chose à enregistrer
  {
    variable = Serial.read(); // Je lis ce que m'envoie l'ordinateur
  }
  monServo.write(variable); // Je dis au servo-moteur de se mettre à la
position "Variable"
}
```

Mais dis moi Jamie, comment on envoie des données vers un ordinateur ?
Minute papillon, j'y arrive

La partie Processing

Aaaahhhh on arrive enfin à la partie la plus intéressante, comment envoyer des ordres à un arduino ? Processing a le bon goût de fonctionner quasiment comme arduino, car Arduino est basé sur Processing. Cependant il faut faire attention : Processing tourne en **Java** et non pas en **C** (ouf, ces deux langages se ressemblent). Il y a aussi des petites différences de fonctionnement.

Du concret plutôt que du blabla :

```
void setup()
{
  size(400,400); // Crée une fenêtre de dimensions 400 * 400 pixels
}
void draw()
{
  ellipse(40, 40, 20, 20); // Dessiner une ellipse de coordonnés (40, 40) et
de rayon horizontal 20 et vertical 20
}
```

Ce code affiche une fenêtre avec un cercle à l'intérieur... Super. Ce qu'il faut comprendre du code :

- Le **setup()** de Processing est le strict équivalent du setup() de Arduino. Il sert notamment à définir la taille de la fenêtre ou que l'on veut écrire à l'arduino.
- Le **draw()** de Processing est le strict équivalent de loop() de Arduino. Pas besoin d'en dire beaucoup plus...

Revenons à nos moutons.
Nous allons voir comment dire à notre Arduino de se mettre à un angle de 45° en appuyant sur

gauche, et à un angle de 135 si nous appuyons à droite.

Il va falloir d'abord créer un objet en début de code qui va contenir le “port” de communication. C'est lui qui va représenter la connexion entre l'ordinateur et l'arduino. Tout d'abord il faut importer une bibliothèque grâce à la ligne de code suivante à placer en tout début de programme (avant la setup())

```
import processing.serial.*;
```

Puis ensuite créer un objet représentant notre connexion, ici nommé “monPort” (comme c'est original) :

```
Serial monPort;
```

Pour ouvrir le port placez le code suivant dans votre boucle **setup()**.

```
String nomPort = Serial.list()[0]; // Mettre 0 ou 1 ou 2 etc. jusqu'à ce que ça marche en compilant avec l'arduino branché.
monPort = new Serial(this, monPort, 9600); // 9600 comme avant ! Bon ok y'a d'autres trucs compliqués

<note important>Si faîtes un programme qui contient les deux lignes si dessus, il y a une probabilité non nulle qu'il ne fonctionne pas du premier coup.</note>
```

Explication :
Quand nous communiquons avec un Arduino, nous utilisons les ports COM1, COM2, COM3, etc. Le nombre que vous mettez dans Serial.list()[i] correspond au ième port COM ouvert. Il faut donc bien trouver celui correspondant à l'arduino, et il faut pas qu'il soit occupé par un autre programme (Arduino, ou un autre code processing, ou carrément un autre logiciel aléatoire...). Il faut donc recompiler plusieurs fois le code.
Le “this” correspond au fait qu'on communique entre cet ordinateur (this) et l'arduino... Mais c'est un petit peu compliqué

Cool, nous savons ouvrir un port ! Maintenant il faut envoyer des infos. Ce que nous allons faire, c'est envoyer 45 à l'arduino. Pour s'y faire on utilise la fonction **write()** de monPort de la façon suivante dans la boucle **draw()**

```
monPort.write(45);
```

Mais je veux envoyer des ordres à l'arduino !!! On parle de quoi depuis un moment ?.

Ok pas de panique, maintenant souvenez vous de la partie arduino : Nous avons fait un code qui attendait de l'ordinateur une valeur contenant l'angle à imposer au servo-moteur. Nous sommes en fait en train d'envoyer cette valeur qu'il attendait tant !!!

Voici un petit résumé du coup de tout ce qu'on a écrit, c'est-à-dire, comment :

- Connecter un ordinateur avec un Arduino
- Envoyer des valeurs à l'Arduino

```
import processing.serial.*;
Serial monPort;

void setup()
{
  // Connexion avec l'Arduino
  String nomPort = Serial.list()[0];
  monPort = new Serial(this, monPort, 9600);
}

void draw()
{
  monPort.write(45); // Envoie de la valeur 45 à l'Arduino
}
```

Si votre Arduino est bien connectée avec le code que je vous ai demandé de téléverser, et si vous êtes connectés au bon port de communication, alors lorsque vous exécuterez le code Processing vous devriez voir votre Arduino bouger.
Si ce n'est pas le cas tentez un autre paramètre pour la fonction Serial.list() jusqu'à ce que ça bouge.

Utiliser son clavier pour contrôler un Arduino

Nous voulons désormais contrôler l'Arduino avec le clavier. Plutôt que de faire un long discours regardez le code suivant, qui utilise la fonction keyPressed

```
void setup()
{
  size(400,400); // Crée une fenêtre de dimensions 400 * 400 pixels
}

void draw()
{
  ellipse(40, 40, 20, 20); // Dessiner une ellipse de coordonnés (40, 40) et
de rayon horizontal 20 et vertical 20
}

void keyPressed() // Si une touche est pressée
{
  if (keyCode == LEFT) // Si j'ai appuyé sur gauche
  {
    println(1); // Afficher 1 dans la console
  }
  if (keyCode == RIGHT) // Si j'ai appuyé sur droite
  {
    println(0); // Afficher 0 dans la console
  }
}
```

Si vous appuyez sur gauche ou droite, vous verrez des 0 ou des 1 apparaître dans la partie noire en

dessous de votre code sur Processing.
Nous allons maintenant créer une variable dans processing, que nous allons mettre égale à 45 si nous appuyons sur gauche, et égale à 135 si nous appuyons sur droite. Comme je suis trop gentil je vous ai écrit le code :

```
int variable = 0;

void setup()
{
  size(400,400); // Crée une fenêtre de dimensions 400 * 400 pixels
}

void draw()
{
  println(variable); // Permet de voir dans la console noire la valeur de
variable
}

void keyPressed() // Si une touche est pressée
{
  if (keyCode == LEFT) // Si j'ai appuyé sur gauche
  {
    variable = 45; // Mettre dans variable la valeur 45
  }
  if (keyCode == RIGHT) // Si j'ai appuyé sur droite
  {
    variable = 135; // Mettre dans variable la valeur 135
  }
}
```

Quand vous appuyez sur une touche, vous voyez dans la console noire que variable change effectivement de valeur.
Pour contrôler le servo-moteur , nous n'avons alors plus qu'à envoyer “variable” depuis la boucle draw() avec ce que nous avons vu avant. Je suis sûr que vous pouvez le faire vous même

…

Du mal ?

Bon…

Ok fin de l'interro !

Voici la correction :

```
import processing.serial.*; // Permet de créer un port de communication

Serial monPort; // Objet représentant notre port de communication

int nombre = 0; // Initialise la variable

void setup()
{
  String nomPort = Serial.list()[0]; // 0, ou 1 ou 2 jusqu'à ce que ça
marche
  monPort = new Serial(this, monPort, 9600); // Ouvre le port de
communication
}

void draw()
{
  monPort.write(nombre); // Envoie vers l'arduino la valeur nombre
}

void keyPressed()
{
  if (keyCode == LEFT) // Si on appuie sur gauche
  {
    nombre = 45;
  }
  if (keyCode == RIGHT) // Si on appuie sur droite
  {
    nombre = 135;
  }
}
```

Si vous compilez ce code, avec votre carte Arduino connectée à l'ordi avec le code que je vous ai demandé de téléverser auparavant, et que vous êtes connectés au bon port, et que vous appuyez sur les flèches du clavier, votre servo-moteur va se mettre à bouger !
C'est exactement ce que nous souhaitions faire.

Pour aller plus loin

Vous savez désormais prendre le contrôle d'un Arduino avec l'aide d'un servo-moteur. Vous pouvez faire beaucoup de choses désormais, pourquoi ne pas tenter les choses suivantes :

- Une caméra qui tourne à gauche quand on appuie sur gauche et inversement
- Des lumières qui s'allument quand on appuie sur les touches du clavier
- Récupérer la luminosité d'une pièce grâce à un Arduino (la connexion se fait dans l'autre sens…)
- Fabriquer une imprimante 3D…

- etc.

Bon pour faire tout ce que je vous ai dit au dessus il faut un peu chercher sur internet. Un tuto spécifiquement dédié aux connexions Ordinateur - Arduino arrive, il vous permettra de faire tout ce que vous voulez.

From:
https://wiki.centrale-med.fr/fablab/ - **WiKi fablab**

Permanent link:
https://wiki.centrale-med.fr/fablab/start:projet:arduino:controle_arduino_clavier

Last update: **2015/03/23 18:48**