REGLEZ LE LOGO ARDUINO

Via une communication série, vous allez utiliser votre Arduino pour contrôler un programme sur votre ordinateur

Découverte : Communication série avec un programme sur ordinateur, logiciel Processing

Durée : 45 minutes

Difficulté : ■■■■■

Basé sur les projets : 1,2,3

Vous avez fait beaucoup de trucs sympas dans le monde physique, maintenant il est temps de contrôler votre ordinateur avec Arduino. Lorsque vous programmez votre Arduino, vous ouvrez une connexion entre l'ordinateur et le micro-contrôleur. Vous pouvez utiliser cette connexion pour recevoir ou transmettre des données à d'autres applications.

La carte de l'Arduino a une puce qui convertit les communications sur USB issues de l'ordinateur en communication série utilisée par le micro-contrôleur de l'Arduino. Une 'communication série' signifie que les deux ordinateurs, votre Arduino et votre PC, échangent des bits d'information en série, c'est-à-dire des bits d'information transmis l'un après l'autre dans le temps.

Lorsqu'ils communiquent en série, les ordinateurs ont besoin de se mettre d'accord sur la vitesse à laquelle ils vont parler à l'autre. Vous avez certainement remarqué lorsque vous avez utilisé le moniteur série qu'il y avait un nombre dans le coin en bas à droite de la fenêtre. Ce nombre, 9600 bits par seconde, ou baud, est identique à la valeur que vous avez déclaré en utilisant l'instruction **Serial.begin()**. C'est la vitesse à laquelle Arduino et l'ordinateur échangent des données. Un bit est la plus petite quantité d'information qu'un ordinateur peut comprendre et manipuler.

Vous avez utilisé le moniteur série pour observer les valeurs retournées par les entrées analogiques; vous allez utiliser une méthode similaire pour transmettre des valeurs à un programme que vous allez écrire dans un environnement de développement appelé **Processing**. **Processing** est écrit en Java, et l'environnement de développement d'Arduino est basé sur **Processing**. Ils se ressemblent beaucoup, vous devriez donc vous y sentir comme chez vous.

Avant de démarrer ce projet, téléchargez la dernière version de Processing depuis http://www.processing.org.
Il peut être utile de consulter les didacticiels "Getting started" (**Bien débuter** en français) et "Overview" (**Vue d'ensemble** en français) sur http://www.processing.org/learning.
Ils vous aideront à vous familiariser avec l'outil Processing avant que vous commenciez à écrire le croquis pour communiquer avec votre Arduino.

Le moyen le plus efficace de transmettre des données entre Arduino et Processing est d'utiliser la fonction **Serial.write()** de l'Arduino. Elle est similaire à la fonction **Serial.print()** que vous avez déjà utilisé, dans le sens où elle transmet des informations à l'ordinateur connecté à la carte, mais au lieu de transmettre des informations compréhensibles par un humain, comme des nombres et des lettres, elle envoie des valeurs de 0 à 255 sous forme d'octets bruts. Cela limite les valeurs que l'Arduino peut envoyer, mais cela permet une transmission rapide des informations.

Présent à la fois sur votre ordinateur et votre Arduino, il y a quelque chose appelé la 'mémoire tampon série' qui conserve les informations jusqu'à ce qu'elles soient lues par un programme. Vous allez transmettre des octets de votre Arduino à la mémoire tampon série de Processing. Processing

lira alors les octets depuis la mémoire tampon. A mesure que les informations sont lues par le programme, l'espace est libéré pour accueillir les suivantes.

Lorsque l'on utilise une communication série entre des périphériques et des programmes, il est important que des deux côtés, non seulement la vitesse de communication soit connue, mais aussi ce à quoi il faut s'attendre. Quand vous rencontrez quelqu'un, vous attendez probablement à un "Bonjour!"; Si à la place, il dit quelque chose comme "Le chat est flou", il y a des chances pour que vous soyez désorienté. Avec les logiciels, vous avez besoin d'accorder les deux côtés de la liaison sur ce qui sera envoyé et reçu, cela s'appelle un 'protocole de communication'.

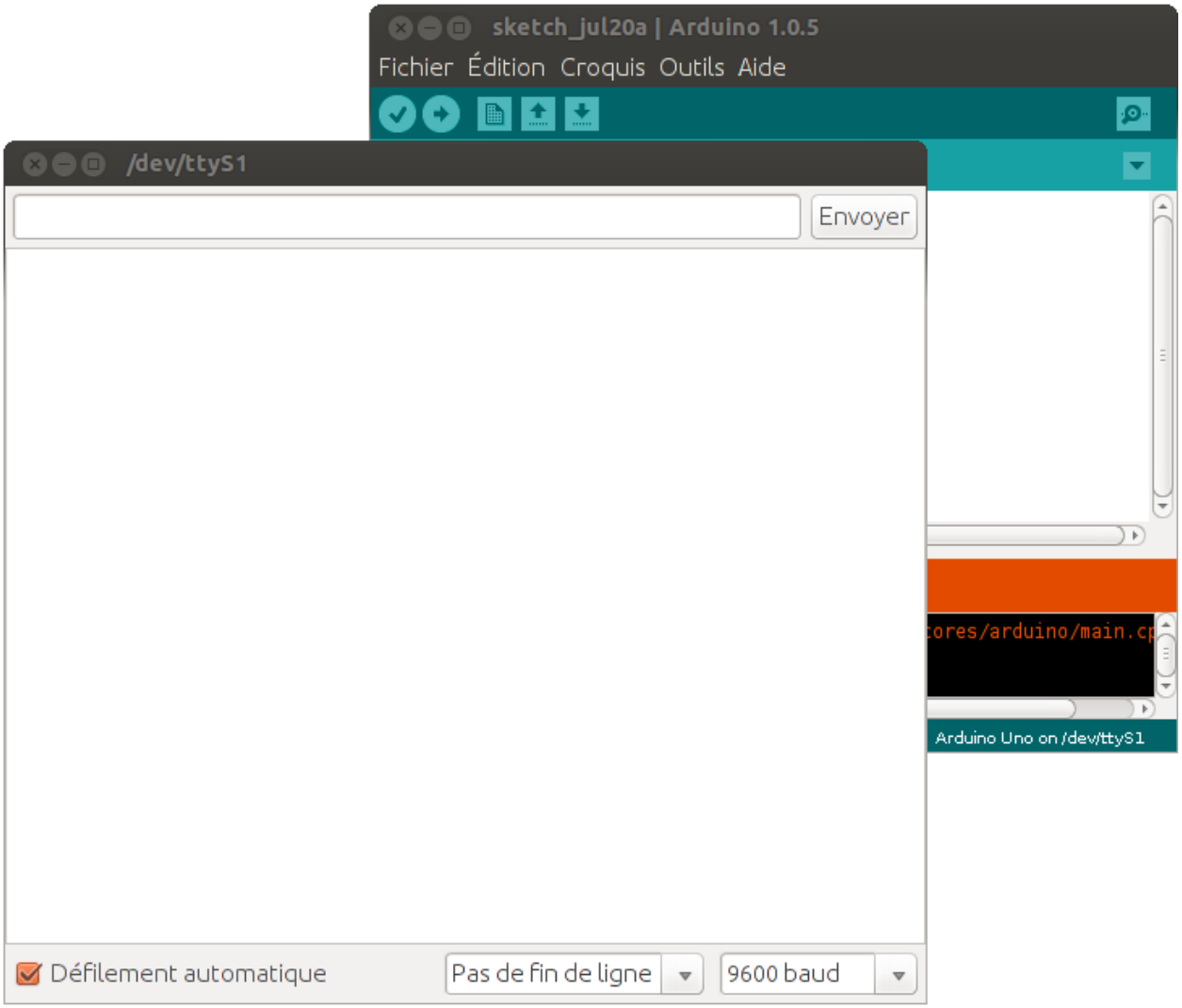
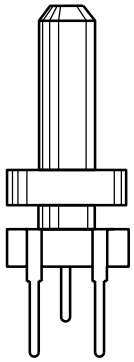


Fig.1

INGREDIENTS


POTENTIOMETRE
x1

CONSTRUIRE LE CIRCUIT

Fig.2 - [Circuit]

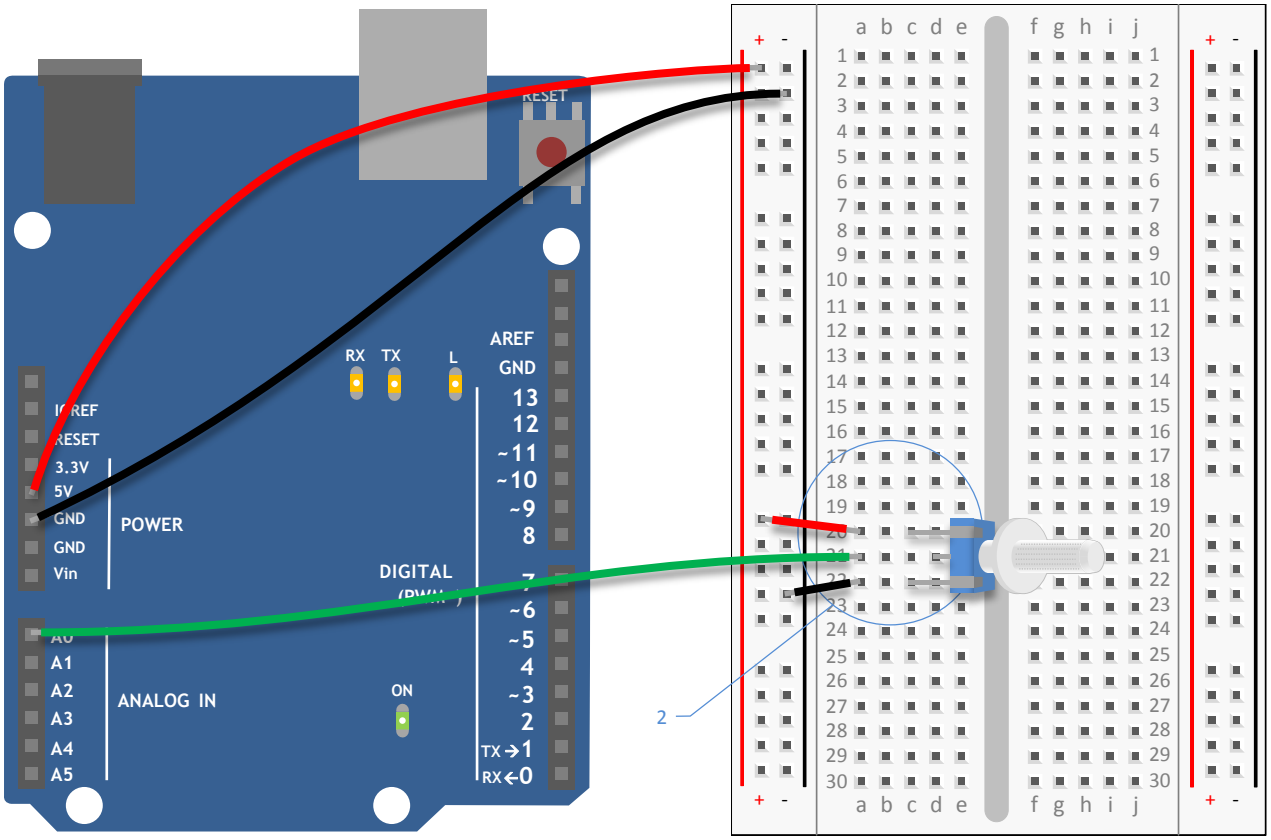
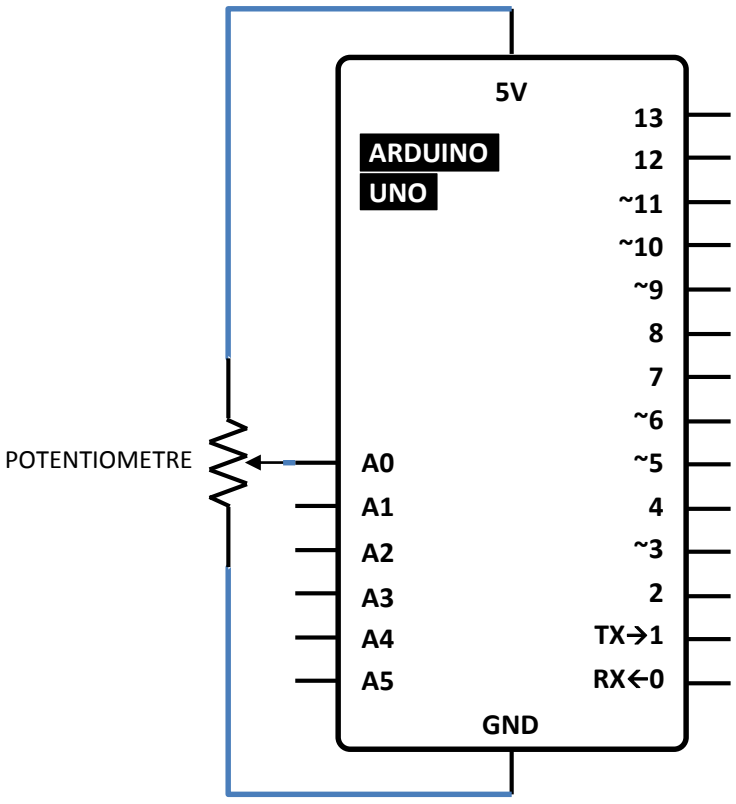


Fig.3 - [Schéma]



1) Connecter l’alimentation et la masse à votre platine de test.

2) Connecter les broches externes de votre potentiomètre, l’une à l’alimentation, l’autre à la masse. Connecter la borne du milieu à l'entrée Analogique 0 (*ANALOG IN - A0*) de votre Arduino.

LE CROQUIS ARDUINO

Ouverture d'une liaison série

Premièrement, programmez votre Arduino. Dans la fonction d'initialisation **setup()**, vous allez démarrer une communication série, exactement comme vous l'avez fait plus tôt lorsque vous cherchiez à afficher les valeurs lues sur un capteur. Le croquis Processing que vous allez écrire aura la même vitesse série, exprimée en 'baud', que votre Arduino.

```
void setup() {
	Serial.begin(9600);
}
```

Transmettre la valeur lue sur le capteur

Dans la boucle **loop()**, vous allez utiliser la commande **Serial.write()** pour envoyer de l'information au travers de la liaison série. **Serial.write()** peut seulement envoyer une valeur entre 0 et 255. Pour être certain que les valeurs que vous allez transmettre se trouvent dans cette plage, divisez la valeur analogique lue par 4. En effet, la valeur analogique lue sur A0 est convertie en une valeur entre 0 et 1023.

```
void loop() {
	Serial.write(analogRead(A0)/4);
```

Laisser le convertisseur ADC (Analog/Digital Converter=Convertisseur Analogique/Numérique) se stabiliser

Après avoir envoyé un octet, attendez 100 millisecondes afin de laisser un temps de repos au ADC (=*Analogic to Digital Converter*) et permettre à Processing de traiter l'information reçue.

Vous pourrez ajuster cette valeur, en l'augmentant ou en la diminuant, si Processing réagit avec du retard à vos commandes.

```
	delay(100);
}
```

Téléchargez le croquis sur l'Arduino et laisser le de côté pendant que vous écrivez le croquis pour Processing.

ENREGISTRER VOTRE CROQUIS ET FERMER L'INTERFACE DE DEVELOPPEMENT ARDUINO, DEMARRER L'INTERFACE DE DEVELOPPEMENT PROCESSING

LE PROGRAMME PROCESSING

Importer et définir l'objet Serial

Le langage Processing est similaire à celui de l'Arduino, mais il y a suffisamment de différences pour que vous ayez besoin de consulter quelques-uns des didacticiels et le guide "Bien débuter", cité plus haut, avant de vous familiariser avec ce langage.

Ouvrez une nouvelle fenêtre de développement Processing. Processing, contrairement à Arduino, ne connait pas les ports séries sans l'aide d'une bibliothèque externe. Importez la bibliothèque Série.

Vous avez besoin de créer une instance de l'objet série, tout comme vous l'avez fait avec Arduino pour la bibliothèque **Servo**. Vous utiliserez le nom unique donné à cet objet dès que vous voudrez utiliser la liaison série.

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

Créer un objet pour manipuler l'image

Pour utiliser des images dans Processing, vous avez besoin de créer un objet qui gérera cette image, et de lui donner un nom.

```
PImage logo;
```

Variable pour stocker la couleur d'arrière-plan

Créer une variable qui gérera la couleur de fond du logo Arduino. Le logo est un fichier *.png*, format d'image gérant l'effet de transparence, permettant de rendre visible le changement de couleur de l'arrière-plan.

```
int couleurFond = 0 ;
```

Processing a une fonction d'initialisation **setup()**, tout comme Arduino. C'est à cet endroit que vous allez ouvrir la liaison série et fournir au croquis quelques paramètres qu'il utilisera pendant son exécution.

```
void setup() {
```

Définir le mode de gestion des couleurs

Vous pouvez modifier la façon dont Processing travaille avec les informations de couleur. Par défaut, il travaille avec les couleurs en mode Rouge Vert Bleu (**RVB** en français = **RGB** (Red, Green, Blue) en anglais). C'est ce mode que vous avez utilisé lors du mélange des couleurs dans le projet '*Lampe mélangeuse de couleurs*' (#04), lorsque vous avez manipulé des valeurs entre 0 et 255 pour modifier la couleur d'une LED RGB. Dans ce croquis, vous allez utiliser un mode de gestion des couleurs appelé **HSB**, ce qui signifie **H**ue (**Teinte** en français), **S**aturation, et **B**rightness (**Luminosité** en français). Vous allez modifier la teinte lorsque vous tournerez le potentiomètre.

colorMode() prend deux arguments : Le type du mode de gestion des couleurs, et la valeur maximale qu'il pourra recevoir.

```
        colorMode(HSB, 255);
```

Chargement de l'image

Pour charger l'image du logo Arduino dans le croquis, lisez-la grâce l'objet `logo` créé plus tôt. Lorsque vous fournissez l'adresse d'une image sous forme d'URL (*Uniform Ressource Locator*), Processing la télécharge lors de l'exécution du croquis. Si cette URL ne fonctionne pas, n'ayez crainte, n'importe quelle autre adresse vers une image .PNG ayant un fond transparent pourra être utilisée.

Avec la fonction **size()**, vous indiquez à Processing quelle taille doit avoir la fenêtre d'affichage. Si vous utilisez les fonctions **logo.width** et **logo.height** comme arguments, le programme ajustera automatiquement la taille de la fenêtre à celle de l'image que vous utilisez.

```
        logo = loadImage("http://arduino.cc/logo.png");
        size(logo.width, logo.height);
```

Affichage des ports série disponibles

Processing a la capacité d'afficher des messages d'état en utilisant la commande **println()**. Si vous l'utilisez en conjonction avec la fonction **Serial.list()**, alors vous obtiendrez la liste de tous les ports série accessibles par votre ordinateur au moment où le croquis a été lancé pour la première fois. Vous utiliserez cela une fois que la phase de programmation sera terminée pour détecter sur quel port se trouve votre Arduino.

```
        println("Ports serie disponibles:");
        printArray(Serial.list());
```

Création de l'objet série

Vous avez besoin d'indiquer à Processing quelques informations à propos de la liaison série. Pour enrichir votre objet série nommé **monPort** avec les informations nécessaires, le programme a besoin de savoir qu'il s'agit d'une nouvelle instance de l'objet série. Les paramètres attendus lors de la création de l'objet sont l'application avec laquelle il va dialoguer, l'identifiant du port série au travers duquel il devra communiquer, et la vitesse de la liaison.

```
        monPort = new Serial(this, Serial.list()[0], 9600);
}
```

L'attribut **this** indique à Processing que vous allez utiliser la liaison série à l'intérieur de l'application en cours. L'argument **Serial.list()[0]** spécifie quel port série vous souhaitez utiliser. **Serial.list()** contient un tableau de tous les périphériques série connectés. L'argument **9600** devrait vous être familier, il définit à quelle vitesse le programme communiquera.

La fonction **draw()** est analogue à la boucle **loop()** d'Arduino dans le sens où elle s'exécute encore et encore, à l'infini. C'est là que les éléments sont dessinés par le programme dans la fenêtre d'affichage.

```
void draw() {
```

Lecture des données de l'Arduino au travers du port série

Vérifier qu'il y a des informations provenant de l'Arduino. La commande **monPort.available()** vous dira s'il y a des données en attente dans la mémoire tampon série. S'il y a quelques octets à cet endroit, lisez les valeurs dans la variable **couleurFond** et affichez les dans la fenêtre de déverminage.

```
        if (monPort.available() > 0) {
            couleurFond = monPort.read();
            println(couleurFond);
        }
```

Définition de l'arrière-plan de l'image et affichage de l'image

La fonction **background()** (**arrière-plan** en français) définie les couleurs d'arrière-plan de la fenêtre. Elle prend 3 arguments. Le premier est la Teinte (**Hue** en anglais), le suivant est la Luminosité (**Brightness** en anglais), et le dernier est la Saturation. Utilisez **couleurFond** comme valeur pour la Teinte, et positionnez les valeurs de la Luminosité et de la Saturation à la valeur maximale, 255.

Vous afficherez le logo avec la commande **image()**. Vous avez besoin d'indiquer à **image()** ce qu'il faut dessiner, et à quelle coordonnée dans la fenêtre débuter l'affichage de l'image. 0,0 est le point situé en haut à gauche, commencez donc là.

```
        background(couleurFond, 255, 255);
        image(logo, 0, 0);
}
```

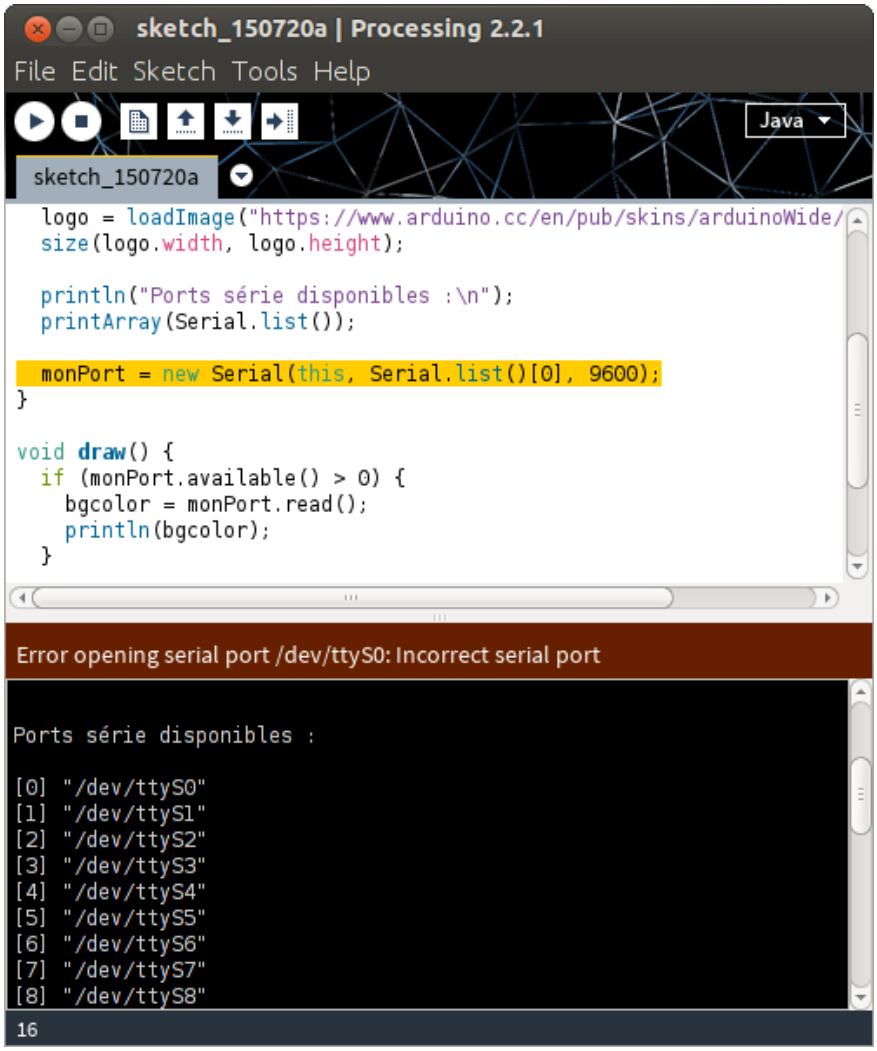
UTILISEZ LE MONTAGE

Connectez votre Arduino et ouvrez le moniteur série dans l'interface de développement (IDE) de Arduino. Tournez le potentiomètre sur votre platine de test. Vous devriez voir de nombreux caractères s'afficher et varier lorsque vous tourner le curseur du potentiomètre. Le moniteur série s'attend à recevoir des caractères ASCII, pas des octets bruts. La norme ASCII est un standard de codage de l'information utilisée pour représenter du texte sur ordinateur. Ce que vous apercevez dans la fenêtre est le moniteur série essayant d'interpréter les octets bruts selon la norme ASCII.

Lorsque vous utilisez la fonction **Serial.println()**, vous transmettez des informations formatées pour l'affichage dans l'interface série. Lorsque vous utilisez la fonction **Serial.write()**, comme dans l'application que vous exécutez maintenant, vous envoyez des informations brutes, c'est-à-dire sans formatage. Les programmes tels que Processing peuvent comprendre ces octets bruts.

Fermez le moniteur série de l'IDE de Arduino.

Exécutez le croquis dans Processing en pressant le bouton 'Lecture' en haut à droite dans l'IDE de Processing. Regarder dans la fenêtre de sortie de Processing. Vous devriez y voir une liste similaire à celle de la capture d'écran ci-dessous.



C'est une liste de tous les ports série disponibles sur votre ordinateur. Si vous utilisez OSX, recherchez un élément de la liste qui contient quelque chose comme "/dev/tty usbmodem411", ce sera certainement le premier élément de la liste. Sur Linux, il peut apparaître comme "/dev/ttyUSB0", ou quelque chose de similaire. Pour Windows, il s'appellera port 'COMx', où 'x' est un nombre, le même que celui que vous utilisez lorsque vous programmez votre carte Arduino. Le nombre affiché dans cet élément est le numéro d'index dans le tableau **Serial.list[]**. Modifier ce

nombre dans votre croquis Processing afin qu'il corresponde au port connecté à votre carte Arduino sur votre ordinateur.

Redémarrez le croquis Processing. Dès que le croquis démarre, tournez le potentiomètre relié à l'Arduino. Vous devriez voir la couleur affichée derrière le logo Arduino changer lorsque vous manipuler le potentiomètre. Vous devriez aussi voir des valeurs s'afficher dans la fenêtre de sortie de Processing. Ces nombres correspondent aux octets bruts que vous envoyez depuis la carte Arduino.

Lorsque vous en aurez assez de tourner le potentiomètre, essayez de le remplacer par un capteur analogique. Trouvez quelque chose dont il vous apparaît intéressant de suivre l'évolution grâce à un changement de couleur (luminosité, température, …). Quelles sont les sensations que vous ressentez en interagissant avec votre capteur ? C'est très probablement différent de ce que vous ressentez en utilisant un clavier ou une souris; L'interaction ne vous paraît-elle pas plus naturelle ?

Plutôt qu'une simple division par 4 de la valeur lue sur `A0`, et vous l'avez vu dans le projet '*Theremine Lumineux*' (#06), vous pouvez ajouter une phase de calibration des valeurs minimum et maximum dans le **setup()** et transposer ces valeurs sur la plage 0 à 255 grâce à la fonction **map()** avant de les envoyer à Processing via la liaison série. Cela pourra permettre d'augmenter la plage d'interaction entre le capteur et le résultat dans Processing.

Lors de l'utilisation d'une communication série, une seule application à la fois peut dialoguer avec l'Arduino. Ainsi, si vous exécutez un croquis Processing qui est connecté à votre Arduino, vous ne pourrez pas télécharger un nouveau croquis sur votre Arduino ou visualiser le moniteur série, jusqu'à ce que vous fermiez l'application active.

Avec Processing et d'autres environnements de programmation, vous avez de nouvelles et remarquables manières de contrôler des médias sur votre ordinateur. Si vous êtes intéressé par ces nouvelles possibilités de contrôler des éléments sur votre ordinateur, prenez un peu de temps pour poursuivre vos expérimentations avec Processing. Il y a plusieurs exemples de communication série à la fois dans les IDEs Processing et Arduino qui pourront vous aider à aller plus loin dans cette exploration.

La communication par liaison série permet à Arduino de parler avec des programmes s'exécutant sur un ordinateur. Processing est un environnement de programmation libre (open source) sur lequel l'IDE d'Arduino est basé. Il est possible de contrôler un croquis Processing avec l'Arduino via une liaison série.

EXTRAIT DU LIVRE DE PROJETS ARDUINO

EDITEURS

Projets et texte par Scott Fitzerald et Michael Shiloh
Revue de texte complémentaire par Tom Igoe

DESIGN ET DIRECTION ARTISTIQUE

TODO
Giorgio Olivero, Mario Ciardulli, Vanessa Poli, Michelle Nebiolo
todo.to.it

EDITION NUMERIQUE ET GESTION DE PROJET

Officine Arduino Torino
Katia De Coi, Enrico Bassi

CONSEILLERS ET SUPPORTERS

Massimo Banzi, Gianluca Martino, Smart Projects

TESTEURS DES PROJETS ET RELECTEURS

Michael Shiloh, Michelle Nebiolo, Katia De Coi, Alessandro Buat, Frederico Vanzati, David Mellis

REMERCIEMENTS

Un grand merci à toute la communauté des utilisateurs Arduino pour leurs contributions continues, leur soutien, et leurs retours.

Nous remercions particulièrement l'équipe Fritzing: Quelques-unes des illustrations de composants électroniques utilisés dans le livre sont issues ou inspirées par le projet open-source de Fritzing (www.fritzing.org).

Un grand merci à Paul Badger pour la bibliothèque *CapacitiveSensor* utilisée dans le projet 13.

Ce livre ne peut être vendu séparément du ‘Kit de Démarrage Arduino’.

Conçu, imprimé et relié à Turin, Italie
Septembre 2012

Première réimpression, Décembre 2012

Traduit de l’anglais par Nicolas PONCET