# SERRURE SONORE

## Réalisez votre propre système de fermeture pour garder les curieux à distance de vos secrets !

Découverte : Utiliser un piezo en entrée, écrire vos propres fonctions

Durée : 1 heure

Difficulté : ■■■■■

Basé sur les projets : 1,2,3,4,5

***Le piezo que vous avez utilisé pour émettre des sons dans les projets 'Thérémine photonique' (#06) et 'Synthétiseur' (#07) peut également être utilisé comme un périphérique d'entrée. Lorsqu'il est alimenté sous 5V, le capteur peut détecter des vibrations dont l'intensité peut être lue via les entrées analogiques de l'Arduino. Vous aurez besoin de connecter une résistance de forte valeur (typiquement 1 MΩ) comme référence à la masse pour que cela fonctionne correctement.***

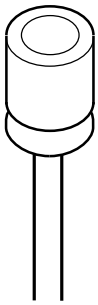
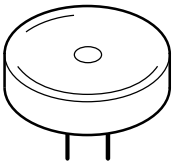
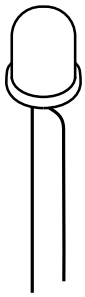
Lorsque le piézo est plaqué contre une surface solide qui peut vibrer, comme un dessus de table en bois, votre Arduino peut détecter l'intensité d'un coup donné contre cette surface. En utilisant cette information vous pourrez déterminer si un nombre de coups frappé tombe dans une plage donnée à l'avance. Ainsi, dans le programme, vous pourrez compter et suivre ce nombre de coups afin de voir s'il correspond à votre paramétrage.

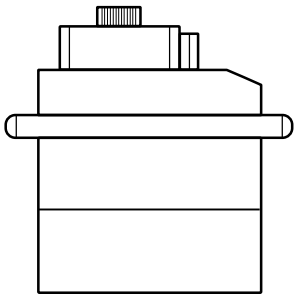
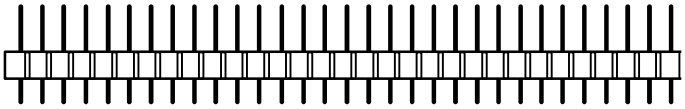
Un interrupteur vous permettra déplacer le bras du servomoteur pour verrouiller votre serrure. Quelques LEDs vous permettront de connaitre la position du verrou : une LED rouge indiquera que la boîte est verrouillée, tandis qu'une LED verte indiquera que la boîte est déverrouillée. Enfin, une LED jaune vous permettra de savoir si un coup frappé sur la table a bien été pris en compte par votre montage.

Vous écrirez également votre propre fonction. Elle vous permettra de savoir si un coup a été frappé trop violemment ou trop faiblement. Ecrire vos propres fonctions vous permet de réduire le temps passé à la programmation. En effet, cela permet de réutiliser du code au lieu de l'écrire à de nombreuses reprises. Les fonctions peuvent prendre des arguments en entrée et retourner des valeurs en sortie. Dans le cas de ce projet, vous passerez à une fonction la mesure de l'intensité d'un coup. Si ce coup est dans la bonne plage d'intensité, ni trop fort, ni trop faible, vous incrémenterez une variable pour comptabiliser ce coup.

Il est possible de construire seulement le circuit, mais cela sera beaucoup plus amusant si vous l'utilisez comme un outil pour verrouiller quelque chose. Si vous avez une boite en bois ou en carton dans laquelle vous pouvez percer quelques trous, utiliser le bras du servomoteur, telle une serrure que vous verrouillerez ou déverrouillerez, afin d'empêcher les autres personnes d'approcher vos précieuses affaires.

## INGREDIENTS

| | | | | | | |
|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |
| RESISTANCE 10 kΩ | RESISTANCE 220 Ω | RESISTANCE 1 MΩ | CONDENSATEUR 100 µF | PIEZO | LED | INTERRUPTEUR |
| x1 | x3 | x1 | x1 | x1 | x3 (rouge, vert, jaune) | x1 |

| | |
|---|---|
|  |  |
| SERVOMOTEUR | CONNECTEUR A BROCHES MALES |
| x1 | 3 broches |

## CONSTRUIRE LE CIRCUIT

*Fig.1* - [Circuit]

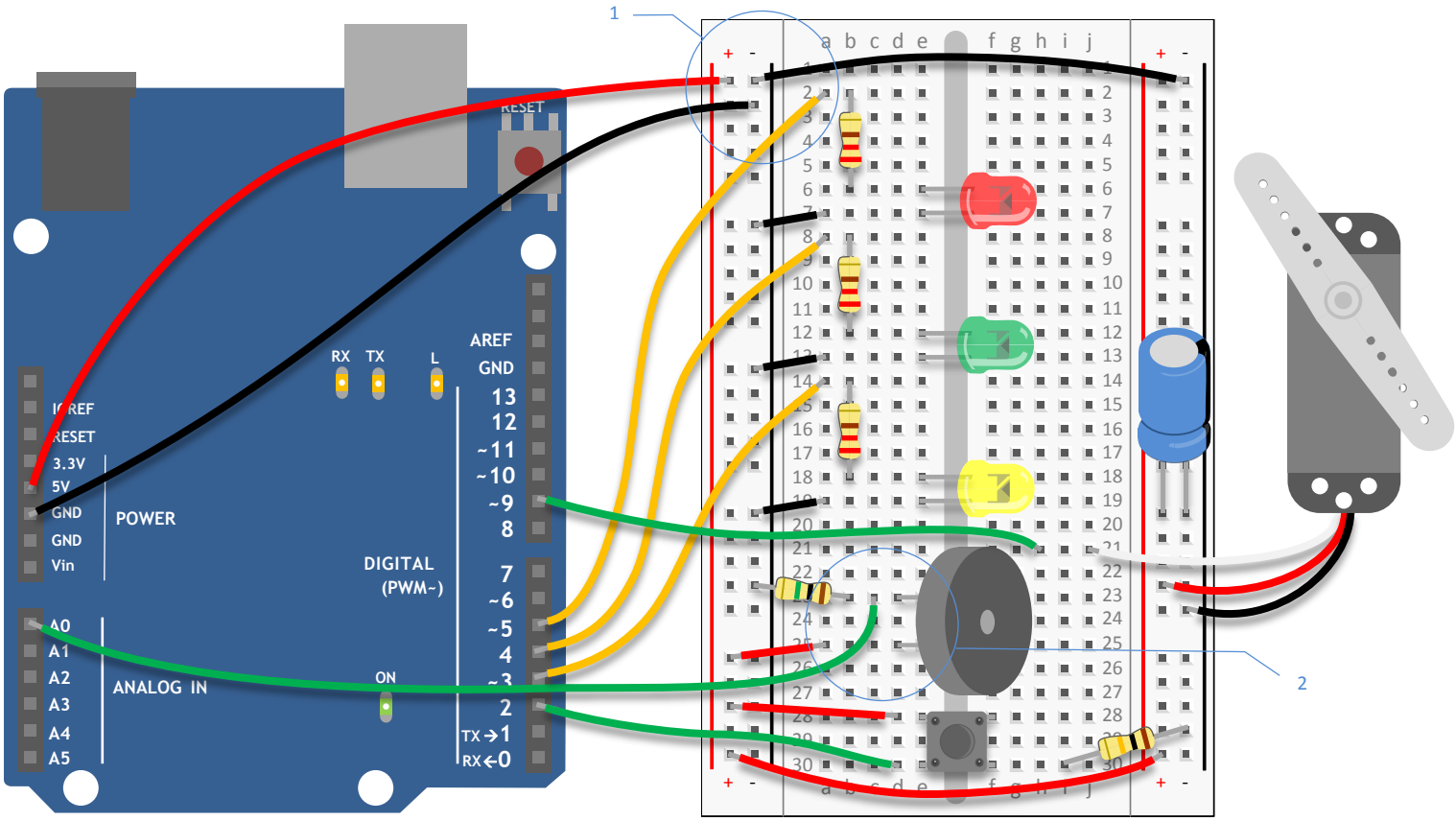
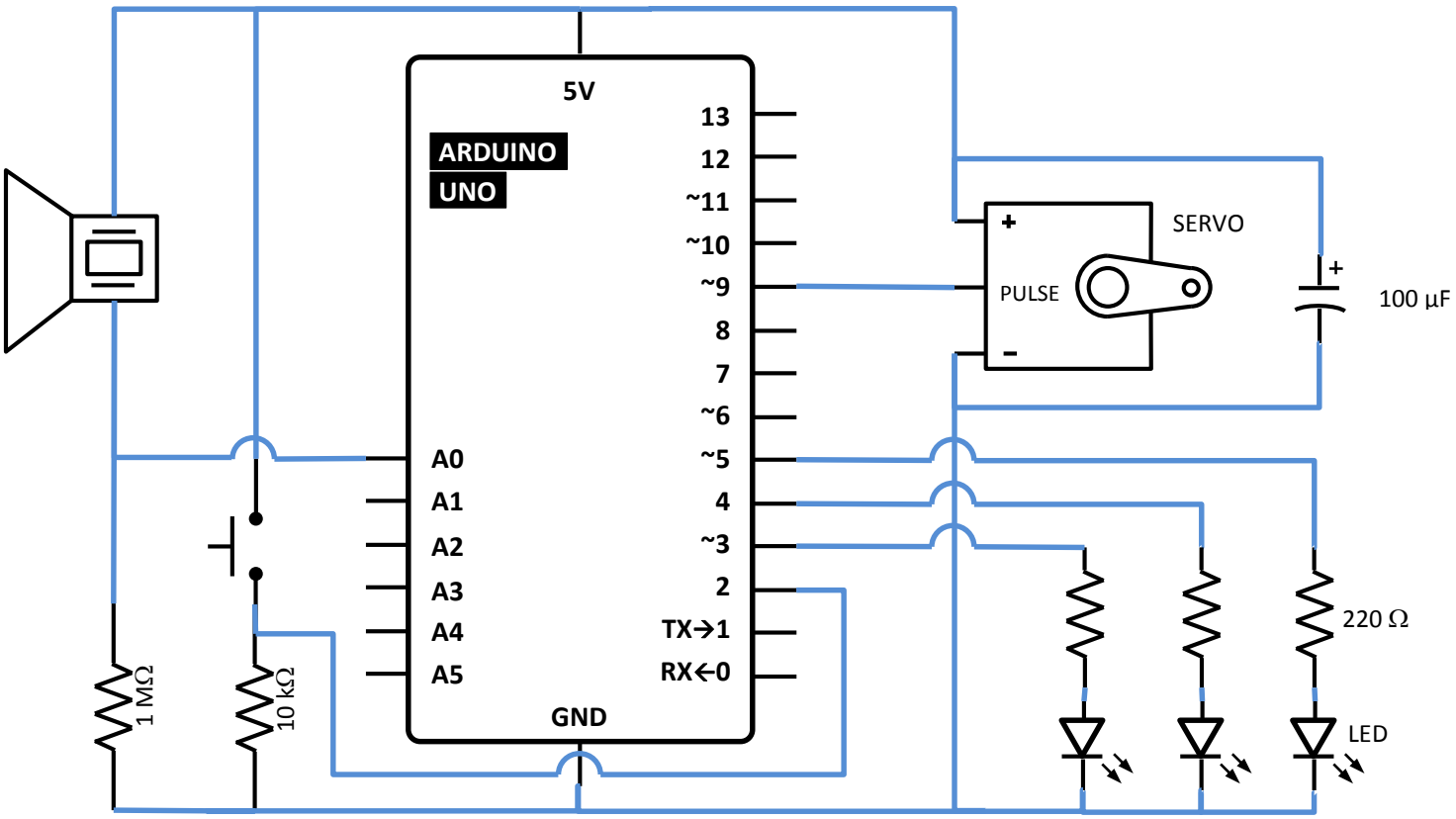



*Fig.2* - [Schéma]



Il y a de nombreuses connections sur la platine d'expérimentation, soyez attentif et veillez à bien suivre la manière dont les éléments sont connectés entre eux.

1) Connecter l'alimentation et la masse des deux côtés de la platine d'expérimentation. Placer le bouton poussoir et connecter un coté au 5V. Sur l'autre côté de l'interrupteur, connecter la masse via une résistance de 10 kΩ. Connecter aussi ce côté de l'interrupteur à la borne 2 de l'Arduino.

2) Enficher le piezo sur la platine d'expérimentation. Connecter une des bornes à l'alimentation. Si votre piezo a un fil rouge ou une borne marquée d'un signe "+", c'est cette borne qui doit être connectée à l'alimentation. Si votre piezo n'indique pas de polarité, alors vous pouvez le brancher dans le sens que vous voulez. Connectez l'autre borne du piezo à l'entrée Analogique 0 de votre Arduino. Enfin, placer la résistance de 1 MΩ entre la masse et la borne du piezo connectée à A0. Des valeurs de résistance moins élevées rendraient le piezo moins sensible aux vibrations.

3) Câbler les LEDs, en connectant leurs cathodes (broche courte) à la masse, et en plaçant une résistance de 220 Ω en série avec chacune des anodes. A travers ces résistances, connecter la LED jaune à la borne numérique 3 de l'Arduino, la LED verte à la borne numérique 4, et la LED rouge à la borne numérique 5.

4) Insérer le connecteur à broches mâles dans le connecteur femelle du servomoteur (voir Fig. 3). Connecter le fil rouge au +5 V, et le fil noir à la masse. Placer le condensateur chimique entre le +5 V et la masse, en s'assurant que la polarité du condensateur est respectée, afin de lisser les irrégularités de tension. Enfin, connecter la broche de donnée du servomoteur à la broche 9 de l'Arduino.

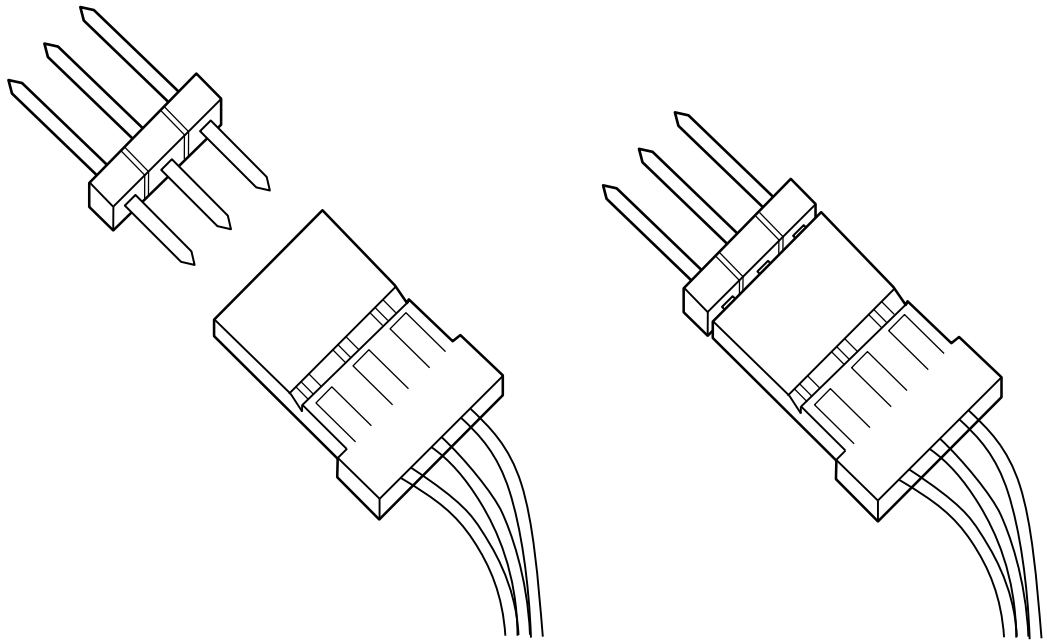


*Fig 3. Votre servomoteur est fourni avec un connecteur femelle, vous aurez donc besoin d'ajouter un bornier à broches mâles pour le connecter à votre platine d'expérimentation.*

## LE PROGRAMME

### *Déclaration de la bibliothèque **Servo***

Comme vous l'avez fait dans le projet '*Indicateur d'humeur*' (#05), vous avez besoin d'importer la bibliothèque **`Servo`** et de créer une instance pour utiliser le servomoteur.

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

### *Constantes utiles*

Créez les constantes qui désignent vos broches d'entrées et de sorties.

```
const int brochePiezo        = A0;
const int brocheInterrupteur = 2;
const int brocheLEDJaune     = 3;
const int brocheLEDVerte     = 4;
const int brocheLEDRouge     = 5;
```

### *Variables stockant les états de l'interrupteur et du piezo*

Créez les variables qui contiendront les états de votre interrupteur et du capteur piezo.

```
int etatInterrupteur;
int intensiteCoup;
```

### *Seuils limites des coups*

Déclarez des constantes qui définiront des seuils d'intensité minimum et maximum, limites permettant de déterminer la validité des coups.

```
const int coupFaible = 10;
const int coupFort   = 100;
```

### *Variable stockant l'état de la serrure et le nombre de coups enregistré*

La variable **estVerrouille** permettra de savoir si la serrure est en position "fermé" ou "ouvert". Un booléen (**boolean**) est un type de donnée qui peut être soit 'vrai' (**true**,1) or 'faux' (**false**,0). Au démarrage, la serrure devra être en position "ouvert".

La dernière variable globale contiendra le nombre de coups valides que vous aurez donné.

```
boolean estVerrouille = false;
int     nombreDeCoups = 0;
```

### *Définition des directions des broches numériques et initialisation du port série et de l'objet représentant le servomoteur*

Dans la fonction d'initialisation **setup()**, reliez le servomoteur à la broche 9.
Désignez les broches des LED comme des sorties et celle de l'interrupteur comme une entrée.

```
void setup() {
       monServo.attach(9);
       pinMode(brocheLEDJaune, OUTPUT);
       pinMode(brocheLEDVerte, OUTPUT);
       pinMode(brocheLEDRouge, OUTPUT);
       pinMode(brocheInterrupteur, INPUT );
```

### *Déverrouillage*

Initialisez une communication série avec l'ordinateur. Ainsi, vous pourrez contrôler l'intensité des coups reçus, connaître la position actuelle de la serrure, et savoir combien de coups il reste à donner pour déverrouiller la boite.

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

Allumez la LED verte, déplacez le servomoteur afin de placer la serrure en position "ouvert". Enfin, affichez l'état courant de la serrure dans le moniteur série, en indiquant que la boite est déverrouillée.

```
       digitalWrite(brocheLEDVerte, HIGH);
```

```
	monServo.write(0);
	Serial.println("La boite est déverrouillée!");
}
```

### *Vérification de l'interrupteur*

Dans la boucle **loop()**, vous allez commencer par vérifier si la boite est fermée ou non. Le résultat va déterminer ce qui va se passer dans la suite du programme. Si elle n'est pas fermée, lisez l'état de l'interrupteur.

```
void loop() {
	if (estVerrouille == false) {
		etatInterrupteur = digitalRead(brocheInterrupteur);
```

### *Verrouillage*

Si l'interrupteur est fermé (vous êtes en train d'appuyer dessus !), passez la variable **estVerrouille** à **true** (vrai), pour indiquer que la serrure est verrouillée. Eteignez la LED verte, et allumez la LED rouge. Si vous n'avez pas ouvert le moniteur série sur votre écran d'ordinateur, l'état de ces LEDs sera une aide précieuse qui vous permettra de connaître la position de la serrure. Placez le servomoteur en position "fermé", et affichez un message dans le moniteur série indiquant que la boite est maintenant fermée. Ajoutez une courte pause afin que le servomoteur ait suffisamment de temps pour se placer dans la nouvelle position.

```
		if (etatInterrupteur == HIGH) {
			estVerrouille = true;
			nombreDeCoups = 0;
			digitalWrite(brocheLEDVerte, LOW );
			digitalWrite(brocheLEDRouge, HIGH);
			monServo.write(90);
			Serial.println("La boite est verrouillée!");
			delay(1000);
		}
	}
```

### *Vérification du capteur de coups*

Si la variable **estVerrouille** est **true** (vrai), la serrure est normalement en position "fermé", lisez la valeur de la vibration actuellement mesurée par le capteur piezo et enregistrez la valeur dans la variable **intensiteCoup**.

```
	if (estVerrouille == true) {
		intensiteCoup = analogRead(brochePiezo);
```

### *Comptage des coups valides*

La prochaine étape est de vérifier si vous avez reçu moins de 3 coups valides, et qu'une vibration a été mesurée par le capteur. Si ces 2 conditions sont vraies, vérifiez que le coup reçu est valide, et alors incrémentez la variable **nombreDeCoups**. C'est ici que vous allez appeler votre propre fonction **verifierCoup()**. Vous écrirez cette fonction dès que vous aurez terminé d'écrire le code de la fonction **loop()**, mais vous savez déjà que vous allez demander à cette fonction de dire si le coup est valide, donc passez lui la variable **intensiteCoup** en argument. Après avoir appelé votre fonction de vérification, affichez le nombre de coups encore nécessaire pour déverrouiller votre serrure.

```
		if (nombreDeCoups < 3 && intensiteCoup > 0) {
			if (verifierCoup(intensiteCoup) == true) {
				nombreDeCoups++;
			}
			Serial.print(3-nombreDeCoups);
			Serial.println(" supplémentaire(s) à donner");
		}
```

### *Déverrouillage*

Vérifiez que vous avez reçu 3 coups valides ou plus. Si c'est vrai, positionnez la variable **estVerrouille** à **false** (faux), et placez le servomoteur en position "ouvert". Attendez quelques millisecondes afin de lui laisser le temps d'initier le mouvement, puis modifiez l'état des LEDs verte et rouge. Affichez un message dans le moniteur série, indiquant que la boite est ouverte.

```
            if (nombreDeCoups >= 3) {
                  estVerrouille = false;
                  monServo.write(0);
                  delay(20);
                  digitalWrite(brocheLEDVerte, HIGH);
                  digitalWrite(brocheLEDRouge, LOW);
                  Serial.println("La boite est déverrouillée!");
            }
```

Fermez l'instruction **if()** (si) et la fonction **loop()** avec une paire d'accolades.

```
      }
}
```

*Définition d'une fonction pour vérifier la validité d'un coup*

Il est maintenant temps d'écrire la fonction **verifierCoup()**. Quand vous écrivez vous-même des fonctions, vous avez besoin d'indiquer si elle retournera une valeur ou pas. Si la fonction ne doit pas retourner de valeur, vous la déclarez comme **void** (vide), comme c'est le cas pour les fonctions **loop()** et **setup()**. Si elle doit retourner une valeur, vous devez déclarer de quel type (**int**, **long**, **float**, etc.) est la valeur renvoyée. Dans notre cas, vous souhaitez vérifier si un coup est valide (vrai, **true**) ou pas (faux, **false**). Ainsi, déclarez la fonction comme étant du type **boolean** (booléen).

Cette fonction traitera un nombre (votre variable **intensiteCoup**) et déterminera si sa valeur est valide ou pas. Pour transmettre la valeur à la fonction, vous créez un paramètre nommé par exemple **intensite** lorsque vous déclarez la fonction.

*Vérification de la validité d'un coup*

Dans votre fonction, lorsque vous ferez référence au paramètre **intensite**, sa valeur correspondra au nombre passé comme argument dans le programme principal. Dans notre cas, la variable **intensite** contiendra la valeur contenue dans la variable **intensiteCoup**, et ce, quelle qu'elle soit. Vérifiez que le contenu de **intensite** est plus grand que la limite basse (**coupFaible**, indiquant un coup trop faible), et plus petite que la limite haute (**coupFort**, indiquant un coup trop violent).

```
boolean verifierCoup(int intensite) {
```

*Indication qu'un coup est valide*

Si le contenu de la variable **intensite** est compris entre ces 2 valeurs, alors le coup est valide, c'est-à-dire ni trop faible, ni trop fort. Faites clignoter la LED jaune une fois et affichez la valeur de l'intensité du coup reçu dans le moniteur série.

```
      if (intensite > coupFaible && intensite < coupFort) {
            digitalWrite(brocheLEDJaune, HIGH);
            delay(50);
            digitalWrite(brocheLEDJaune, LOW );
            Serial.print("Coup VALIDE, d'intensité ");
            Serial.println(intensite);
```

*La fonction renvoie 'vrai' (**true**)*

Afin d'informer le programme appelant la fonction du résultat de la comparaison, vous utiliserez la commande **return**, pour renvoyer **true** (vrai). Lorsque vous utilisez la commande **return**, vous terminez aussi l'exécution de la fonction : Une fois la commande **return** exécutée, vous revenez dans le programme ayant appelé la fonction, ici le programme principal.

```
            return true;
      }
```

*Indication d'un coup invalide; La fonction renvoie 'faux' (**false**)*

Si **intensite** est soit trop faible, soit trop forte, affichez la valeur de l'intensité du coup reçu dans le moniteur série et renvoyez **false** (faux) au programme principal.

Terminez votre fonction avec une dernière accolade.

```
      else {
            Serial.print("Coup INVALIDE, d'intensité ");
            Serial.println(intensite);
```

```
                return false;
        }
}
```

## UTILISEZ LE MONTAGE

Lorsque vous branchez pour la première fois votre circuit à Arduino, ouvrez le moniteur série sur votre ordinateur. Il est possible qu'à la mise sous tension, vous entendiez le capteur piezo émettre un petit "clic". Vous devriez aussi voir la LED verte s'allumer et le servomoteur se placer en position "déverrouillé". Enfin, le moniteur série devrait afficher "La boite est déverrouillée !".

Appuyez brièvement sur l'interrupteur, la LED verte s'éteint, la LED rouge s'allume et le servomoteur se place en position "verrouillé". Enfin, le moniteur série devrait afficher "La boite est verrouillée !".

Comme on frappe à une porte, essayer de frapper doucement puis de plus en plus fort sur la surface à laquelle le capteur piezo est fixé (par exemple, la table ou le couvercle de la boite), afin de voir quelle force est nécessaire pour déclencher votre fonction. Vous saurez que cela fonctionne lorsque la LED jaune clignotera et que le moniteur série vous dira que vous avez donné un coup valide, ainsi que son intensité. Il affichera aussi le nombre de coups supplémentaires à donner afin de déverrouiller la boite.

Lorsque vous aurez atteint le nombre de coups 'valides' attendu, La LED rouge s'éteindra, la LED verte s'allumera, le servomoteur pivotera de 90 degrés, et le moniteur série vous indiquera que la serrure est déverrouillée.

Appuyez à nouveau sur l'interrupteur pour démarrer un nouveau cycle.

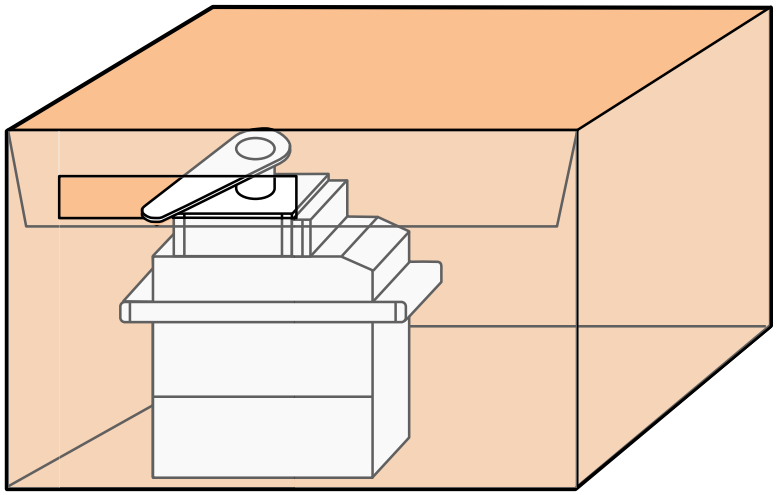
Les valeurs minimale (**coupFaible**) et maximale (**coupFort**) de l'intensité de votre "toc" idéal peuvent être différentes de celles données dans cet exemple. Cela dépend en effet d'un grand nombre de paramètres, comme le type de surface sur laquelle le capteur piezo est placé (carton, bois, …), la manière dont le capteur est fixé sur cette surface (colle, ruban adhésif, élastique,…), et bien entendu de la force avec laquelle le capteur est plaqué contre la paroi. Utilisez le moniteur série et l'exemple *AnalogInSerialOut* dans l'interface de développement (aussi appelée *IDE*) de Arduino, afin de trouver les valeurs limites qui conviendront le mieux à votre montage. Vous pourrez trouver une explication détaillée de cet exemple à l'adresse : http://www.arduino.cc/analogtoserial

Si vous placez ce projet dans une boite, vous devrez y faire des trous afin de faire apparaitre les LEDs et rendre accessible l'interrupteur. Vous aurez aussi besoin de pratiquer une fente pour permettre au bras du servomoteur de pivoter et ainsi verrouiller la boite. Il sera aussi certainement utile de faire un trou pour passer votre câble USB relié à votre ordinateur : Il permettra au projet de communiquer avec le moniteur série et vous aidera à déterminer à quel point ce nouvel environnement modifie la sensibilité du montage et nécessite des ajustements des valeurs limites.

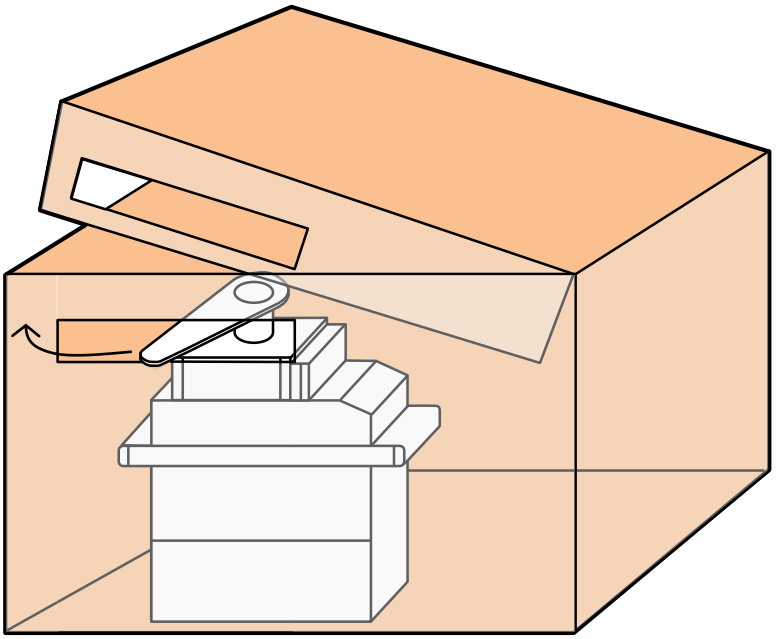
Vous pourriez avoir besoin de modifier la disposition des éléments et de Arduino, voire de souder les LEDs et l'interrupteur afin de les rendre accessible depuis l'extérieur de la boite. Le brasage ou soudure, est un procédé d'assemblage de deux ou plusieurs composants métalliques au moyen d'un adhésif fondu au niveau du joint entre les composants. Si vous n'avez jamais soudé auparavant, demandez à quelqu'un ayant de l'expérience en soudure de vous aider, ou bien essayer de vous entraîner sur une chute de fil avant de tenter la soudure sur un des éléments utilisé dans ce projet. Lorsque vous soudez quelque chose, c'est habituellement pour que la connexion soit définitive, c'est pourquoi il est important de s'assurer qu'il s'agit bien d'un élément que vous êtes prêt à assembler de cette façon.

Connectez-vous à l'adresse http://www.arduino.cc/soldering pour une explication sur la manière de souder.

1°) Pratiquez 2 trous dans sur votre boite: un sur le côté de la boite, et un autre sur le côté du couvercle. Placez le servomoteur dans la boite de telle manière que le bras puisse sortir et entrer librement au travers de ces 2 trous lorsque la boite est fermée.



2°) Maintenez le servomoteur à cette place, par exemple à l'aide de ruban adhésif, en prenant encore une fois garde à ce que le bras puisse aisément pivoter à l'intérieur de la fente que vous avez pratiqué sur la boite.



Ecrire vos propres fonctions ne vous permet pas seulement de contrôler plus facilement le flot d'exécution de vos programmes. Elles les rendent aussi plus lisibles alors qu'ils deviennent de plus en plus complexes. Au court du temps, comme vous écrirez plus de programmes, vous pourrez vous rendre compte que vous avez écrit un grand nombre de fonctions qui pourront être réutilisées dans différents projets, rendant l'étape de la rédaction du programme plus rapide et en parfaite adéquation avec à votre style de programmation.

Cette exemple compte seulement le nombre de coups donnés, sans tenir compte du temps que cela prend. Vous pouvez commencer par complexifier cet exemple en créant un chronomètre avec l'instruction **millis()**. Utilisez le chronomètre pour déterminer si les coups sont reçus dans une période de temps donnée. Revenez au projet '*Montre Numérique*' (#08) pour un exemple d'utilisation des chronomètres. Vous n'êtes pas limité à cette simple évolution. Vous pouvez

chercher à reconnaître des combinaisons complexes de coups, basées sur l'intensité des vibrations et le temps : Par exemple, rechercher une combinaison de coups faibles et forts, en mesurant aussi le temps séparant deux coups successifs, et ceci pendant une période maximale déterminée. Il y a de nombreux exemples en ligne qui expliquent comment procéder. Vous pouvez chercher "Arduino knock lock" sur un moteur de recherche pour découvrir plus d'exemples sur ce type de projet.

*Les éléments piezo peuvent être utilisés comme des entrées lorsqu'ils sont câblés en ponts diviseur de tension avec une résistance de forte valeur. Créer une fonction est un moyen simple d'écrire du code qui pourra être réutilisé pour une tâche spécifique.*

# 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