ZOOTROPE

Créer des images animées d'avant en arrière avec votre Arduino en connectant un moteur à un pont en H et quelques images fixes

Découverte : Les 'ponts en H'

Durée : 30 minutes

Difficulté : ■■■■■

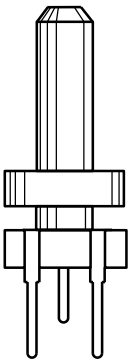
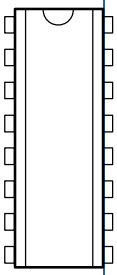
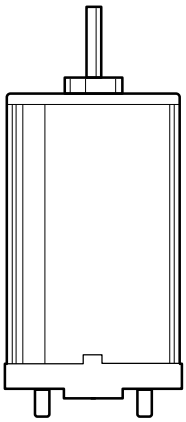
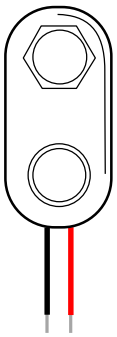
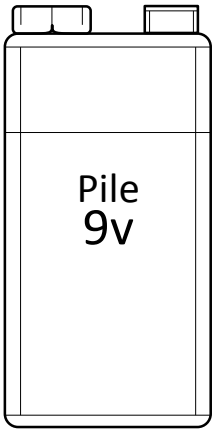
Basé sur les projets : 1,2,3,4,9

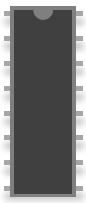
Avant Internet, la télévision, et même avant les films, quelques-unes des premières images animées étaient créées à l'aide d'un outil appelé un zootrope. Les zootropes créent l'illusion du mouvement à partir d'un ensemble d'images fixes ayant quelques petites différences entre elles. Les zootropes sont typiquement des cylindres possédant des fines fentes sur leur pourtour. Lorsque le cylindre tourne et que vous regardez à travers les fentes, vos yeux perçoivent les images fixes disposées sur la face opposée du cylindre comme une animation. Les fentes permettent d'empêcher les images d'être perçues comme une grosse tâche floue, et la vitesse à laquelle les images défilent permet de créer l'illusion du mouvement. A l'origine, ces instruments étaient mis en rotation à la main, ou grâce à un mécanisme à remontoir.

Dans ce projet, vous aller fabriquer votre propre zootrope permettant d'animer une plante carnivore. Le mouvement de rotation sera produit par un moteur. Afin de rendre ce système encore plus avancé, vous ajouterez 2 interrupteurs qui permettront pour l'un, d'allumer ou d'éteindre le dispositif, pour l'autre de contrôler le sens de rotation, et aussi un potentiomètre qui contrôlera la vitesse.

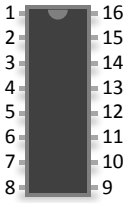
Dans le projet 'Moulinet motorisé' vous aviez un moteur ne tournant que dans un seul sens. Si vous aviez interverti les fils d'alimentation et de masse aux bornes du moteur, le moteur aurait alors tourné dans le sens opposé. Il n'est vraiment pas pratique de devoir effectuer cette modification à chaque fois que vous voulez faire tourner quelque chose dans un sens différent. C'est pourquoi nous allons utiliser un composant appelé 'Pont en H' qui inversera pour nous la polarité aux bornes du moteur.

INGREDIENTS

						
RESISTANCE 10 kΩ	POTENTIOMETRE	PONT en H	BOUTON POUSSOIR	MOTEUR DC	BORNIER PILE	PILE
x2	x1	x1	x2	x1	x1	x1



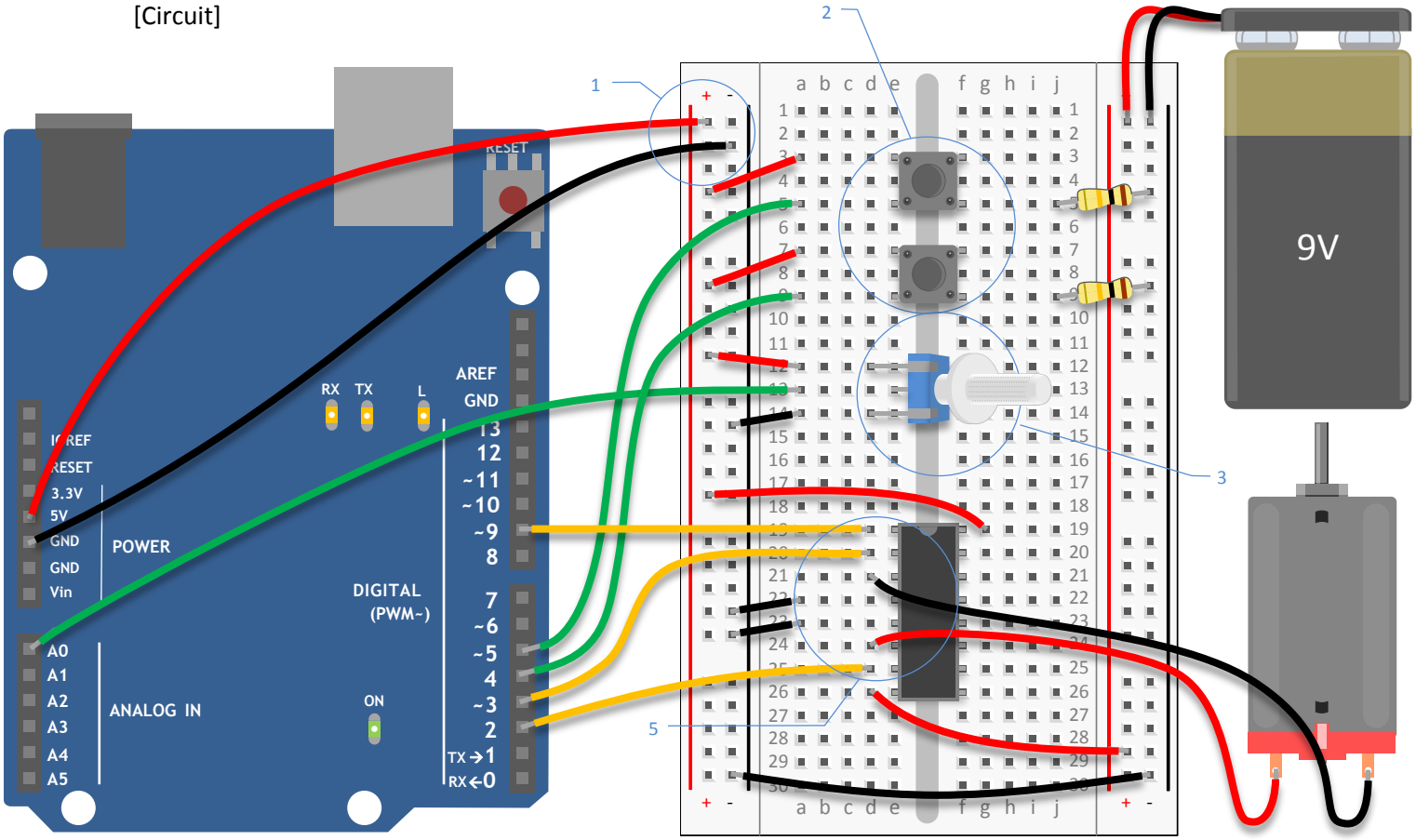
Les ***Ponts en H*** sont un type de composant appelés ***circuits intégrés (Integrated Circuits : IC)***. Les ICs sont des composants qui contiennent de grands circuits dans un volume minuscule. Ils peuvent aider à simplifier l'utilisation de circuits plus complexes en les plaçant dans des composants aisément remplaçables. Par exemple, le pont en H que vous allez utiliser dans ce projet contient de nombreux transistors. Pour construire le circuit contenu dans le pont en H, vous auriez probablement eu besoin d'une autre platine d'expérimentation.



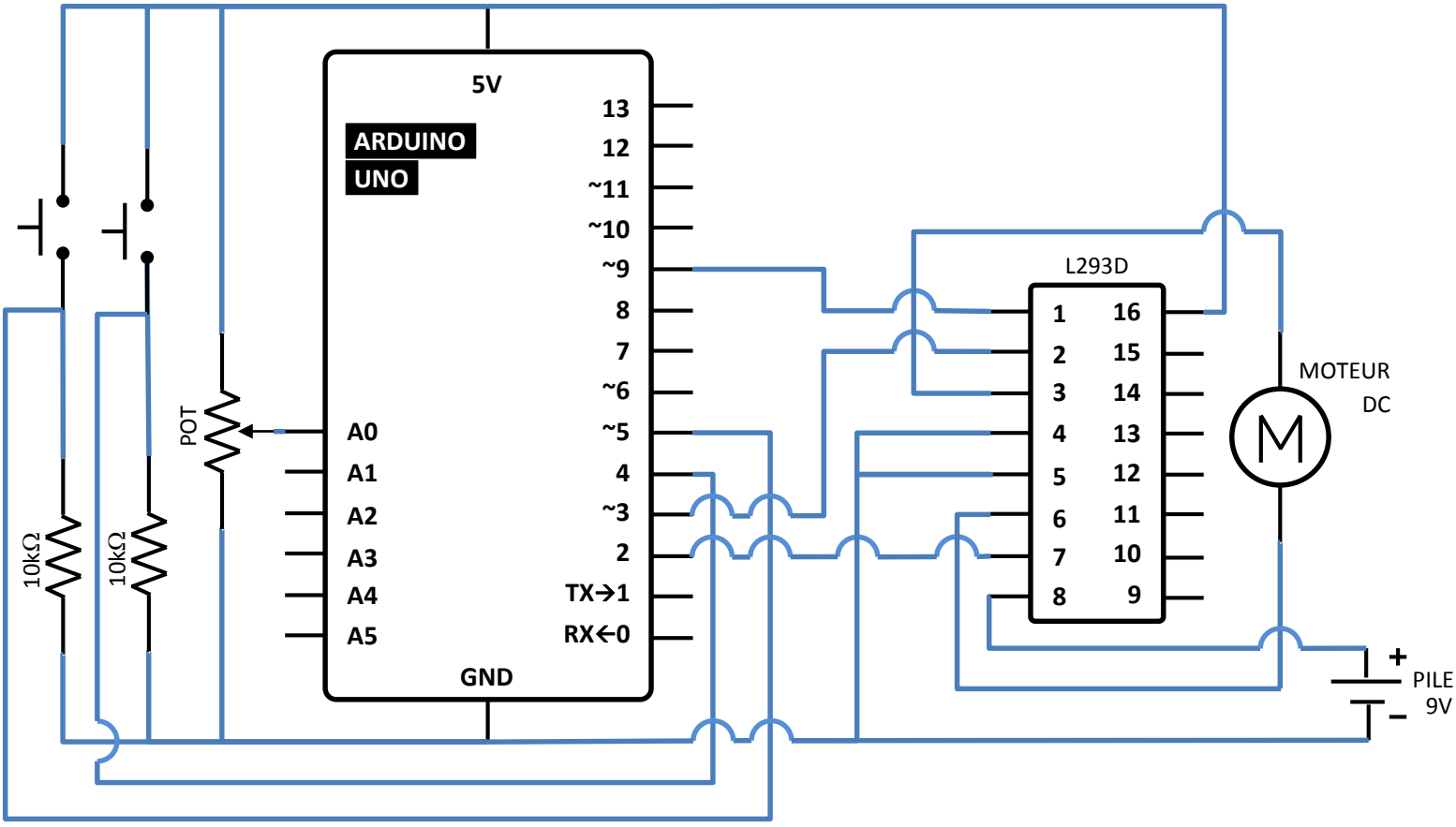
Avec un IC, l'accès aux circuits qu'il contient s'effectue grâce aux broches situées sur son pourtour. Différents ICs auront des nombres de broche différents, et toutes les broches ne seront pas nécessairement utilisées dans tous les circuits. Il est quelques fois plus pratique de désigner une broche par son numéro plutôt que par sa fonction. Lorsque vous regardez un IC, la partie possédant une encoche est considérée comme le haut. Vous pouvez alors identifier le numéro des broches en commençant à compter à partir du coin en haut à gauche puis en suivant le sens d'un "U" comme illustré sur la Fig. 1.

CONSTRUIRE LE CIRCUIT

[Circuit]



[Schéma]



1) Branchez l'alimentation et la masse à votre platine d'expérimentation à partir de l'Arduino.

2) Ajouter 2 boutons poussoir sur la platine, et connecter l'un des côtés de chaque interrupteur à l'alimentation. Ajouter une résistance 'pull-down' de 10kΩ en série avec la masse sur l'autre côté de chacun des interrupteurs.

   L'interrupteur relié à la broche 4 contrôlera la direction, tandis que l'interrupteur relié à la broche 5 permettra d'allumer ou d'éteindre le moteur.

3) Connecter le potentiomètre à la platine. Câbler le 5V d'un côté et la masse de l'autre. Relier la broche centrale à l'entrée analogique 0 de l'Arduino. Ce potentiomètre sera utilisé pour contrôler la vitesse de rotation du moteur.

4) Placer le pont en H sur votre platine de façon à ce qu'il enjambe le centre de la plaque (voir Fig. 2 pour le détail sur sa position). Connecter la broche 1 du pont en H à la sortie numérique 9 de l'Arduino. C'est la broche d'Activation du pont en H. Quand elle reçoit 5V, le moteur s'allume, quand elle reçoit 0 V, le moteur s'éteint. Vous utiliserez cette broche pour commander le pont en H en PWM, afin de régler la vitesse de rotation du moteur.

5) Connecter la broche 2 du pont en H à la sortie numérique 3 de l'Arduino. Connecter la broche 7 à la sortie numérique 2. Ce sont les broches que vous utiliserez pour communiquer avec le pont en H, et lui indiquer dans quelle direction faire tourner le moteur. Si la broche 3 est à l'état bas (LOW) et que la broche 2 est à l'état haut (HIGH), le moteur tournera dans un sens. Si la broche 2 est à l'état bas (LOW) et que la broche 3 est à l'état haut (HIGH), le moteur tournera dans l'autre sens. Si les 2 broches sont à l'état haut (HIGH) ou bas (LOW) en même temps, le moteur arrêtera de tourner.

6) Le pont en H est alimenté via sa broche 16, branchez la au 5V. Les broches 4 et 5 sont toutes les 2 connectées à la masse.

7) Relié les bornes du moteur aux broches 3 et 6 du pont en H. Ces 2 bornes seront alimentées en fonctions des signaux que vous enverrez sur les broches 2 et 7 du pont en H.

8) Brancher le bornier de la pile (sans que la pile y soit connectée!) à l'autre ligne d'alimentation de votre platine d'expérimentation. Connecter la masse de l'Arduino à la masse de la pile. Connecter la broche 8 du pont en H à l'alimentation de la pile. C'est via cette broche que le pont en H capte l'alimentation qu'il transmet au moteur. Assurez-vous que vous n'avez pas, par erreur, relié les lignes des alimentations en 5V et en 9V. Ces lignes doivent absolument rester séparées. Seules les lignes de masse doivent être connectées.

LE PROGRAMME

Définissez les constantes et les variables

Créer les constantes qui définissent les broches d'entrée/sortie

```
const int controlPin1 = 2;
const int controlPin2 = 3;
const int enablePin = 9;
const int directionSwicthPin = 4;
const int onOffSwitchStateSwitchPin = 5;
const int potPin = A0;
```

Créer les variables qui stockent les différents états du programme

Utiliser des variables pour conserver en mémoire les valeurs de vos entrées. Vous détecterez les changements de position des 2 interrupteurs, en comparant leurs états pendant une boucle à ceux pendant la boucle précédente, comme cela a déjà été fait pendant le projet 'Montre'. Aussi, en plus de stocker leur position courante dans les variables **onOffSwitchState** (*marcheArrêtInterrupteurEtat*) et **directionSwitchState** (*directionInterrupteurEtat)*, vous aurez besoin de conserver en mémoire l'état précédent de chacun des interrupteurs dans les variables **previousOnOffSwitchState** (*précédentMarche ArrêtInterrupteurEtat*) et **previousDirectionSwitchState** (*précédentDirectionInterrupteurEtat*)).

```
int onOffSwitchState = 0;
int previousOnOffSwitchState = 0;
int directionSwitchState = 0;
int previousDirectionSwitchState = 0;
```

Créer les variables de commande du moteur

motorDirection (*moteurDirection*) garde une trace du sens de rotation du moteur, tandis que **motorSpeed** (*moteurVitesse*) garde une trace de la vitesse de rotation et **motorEnabled** (*moteurActivé*) garde une trace de l'état d'alimentation du moteur (alimenté ou éteint), .

```
int motorEnabled = 0;
int motorSpeed = 0;
int motorDirection = 0;
```

Déclarer les broches numériques comme entrée ou sorties

Dans **setup()**, définissez les directions de chacun des broches d'entrée et de sortie.

```
void setup() {
       pinMode(directionSwitchPin, INPUT);
       pinMode(onOffSwitchStateSwitchPin, INPUT);
       pinMode(controlPin1, OUTPUT);
       pinMode(controlPin2, OUTPUT);
       pinMode(enablePin, OUTPUT);
```

Eteindre le moteur

Mettre la broche **enablePin** (*activerBroche*) à l'état bas (LOW) au démarrage de l'Arduino, afin que le moteur ne se mette pas à tourner.

```
       digitalWrite(enablePin, LOW);
}
```

Lire les informations des capteurs

Dans votre **loop()**, lisez l'état de l'interrupteur Marche/Arrêt et stocker le dans la variable **onOffSwitchState** (*marcheArrêtInterrupteurEtat*). De même, lisez l'état de l'interrupteur de

changement de direction et stocker le dans la variable **directionSwitchState** (*directionInterrupteurMoteur*). Enfin, lisez la valeur aux bornes du potentiomètre afin d'en déduire une consigne de vitesse pour le moteur que vous stockerez dans la variable **motorSpeed** (*moteurVitesse*).

```
void loop() {
	onOffSwitchState = digitalRead(onOffSwitchStateSwitchPin);
	delay(1);
	directionSwitchState = digitalRead(directionSwitchPin);
	motorSpeed = analogRead(potPin)/4;
```

Vérifier si l'état de l'interrupteur Marche/Arrêt a changé

Il y a seulement 2 états dans lequel un moteur peut être : Allumé ou Eteint. Ainsi, vous allez vouloir faire basculer la variable **motorEnabled** (*moteurActivé*) alternativement entre ces 2 états. Dans un test SI (**if**), le 0 est interprété comme étant la valeur FAUX et le 1 comme étant la valeur VRAI. Nous voulons donc faire basculer alternativement la valeur de **motorEnabled** (*moteurActivé*) de 0 à 1 et vice-versa. Une manière de parvenir à ce résultat est d'utiliser l'opérateur d'inversion **!** ainsi : **motorEnabled = !motorEnabled**. Cet opérateur **!** permet d'inverser la valeur d'une variable de VRAI à FAUX et vice-versa, un peu comme le signe **–** fait passer d'une valeur positive à une valeur négative.

S'il y a une différence entre l'état actuel de l'interrupteur et son état lu lors de la boucle précédente, et que l'état actuel est haut (HIGH), mettre la variable **motorEnabled** (*moteurActivé*) à 1 (VRAI, moteur Allumé). Si c'est bas (LOW), mettre la variable à 0 (FAUX, moteur Eteint).

```
	if (onOffSwitchState != previousOnOffSwitchState) {
		if (onOffSwitchState == HIGH) {
			motorEnabled = !motorEnabled;
		}
	}
```

Vérifier si l'état de l'interrupteur Direction a changé

Vérifier si l'interrupteur Direction est dans une position différente de sa position pendant la boucle précédente. Si elle est différente, modifier la valeur de la variable stockant la Direction. Comme pour Marche/Arrêt, il y a seulement 2 sens dans lequel un moteur peut tourner, ainsi, vous allez vouloir faire basculer la variable alternativement dans 2 états. Une manière de parvenir à ce résultat est d'utiliser l'opérateur d'inversion **!** ainsi : **motorDirection = !motorDirection**.

```
	if (directionSwitchState != previousDirectionSwitchState) {
		if (directionSwitchState == HIGH) {
			motorDirection = !motorDirection;
		}
	}
```

Modifier la valeur de sortie de la broche pour faire tourner le moteur dans le bon sens

La variable **motorDirection** (*moteurDirection*) détermine dans quel sens le moteur tourne. Pour régler la direction, vous définissez l'état des broches de contrôle, l'une à l'état haut (HIGH) et l'autre à l'état bas (LOW). Lorsque **motorDirection** change, inversez l'état des broches de contrôle.

Lorsque le bouton poussoir Direction est appuyé, vous voulez faire tourner le moteur dans l'autre sens, il faut donc inverser l'état des broches **controlPin1** et **controlPin2**.

```
	if (motorDirection == 1) {
		digitalWrite(controlPin1, HIGH);
		digitalWrite(controlPin2, LOW);
	} else {
		digitalWrite(controlPin1, LOW);
		digitalWrite(controlPin2, HIGH);
	}
```

PWM le moteur s'il est activé

Si la variable **motorEnabled** (*moteurActivé*) est à VRAI (1), définissez la vitesse de rotation du moteur en utilisant **analogWrite()** pour envoyer un signal PWM à la broche d'Activation. Si **motorEnabled** (*moteurActivé*) est à FAUX (0), alors éteignez le moteur en positionnant avec **analogWrite** la valeur sur cette broche à 0.

```
        if (motorEnabled == 1) {
                analogWrite(enablePin, motorSpeed);
        } else {
                analogWrite(enablePin, 0);
        }
```

*Enregistrer les états courants pour la prochaine boucle **loop()***

Avant de sortir de **loop()**, stockez les états actuels des interrupteurs dans les variables stockant les états précédents, pour préparer la prochaine exécution du programme dans la boucle **loop()** suivante.

```
        previousDirectionSwitchState = directionSwitchState;
        previousOnOffSwitchState = onOffSwitchState;
}
```

UTILISER LE MONTAGE

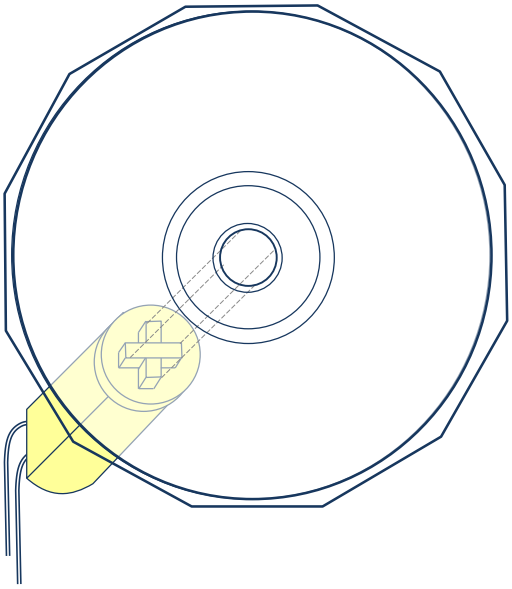
Branchez l'Arduino à votre ordinateur. Branchez la pile sur son bornier. Lorsque vous presserez le bouton poussoir Marche/Arrêt, le moteur devrait commencer à tourner. Si vous tourner la tige du potentiomètre, vous devriez constater que le moteur accélère ou ralentit en fonction de la valeur lue aux bornes du potentiomètre. En pressant le bouton Marche/Arrêt une seconde fois, le moteur s'arrêtera. Essayez alors l'interrupteur de Direction et vérifiez que le moteur peut tourner dans les 2 sens.

Une fois que vous aurez vérifié que le circuit fonctionne correctement, déconnectez la pile et le connecteur USB du circuit.

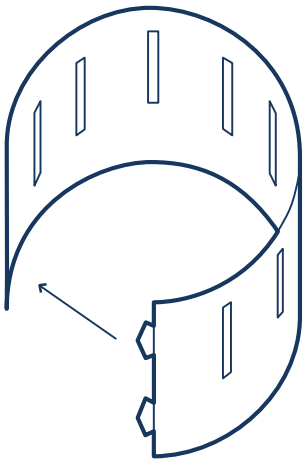
Afin de construire votre zootrope, vous devez prendre le 'moulinet' utilisé pendant le 'Projet 9' et découper les fentes verticales dans le bandeau fourni avec votre kit. Une fois que le CD est emboité de manière sécurisée sur le support en forme de croix fixé sur l'arbre du moteur, branchez à nouveau la pile et le connecteur USB. Tenez votre projet en hauteur, afin que vous puissiez regarder à travers les fentes (mais assurez-vous bien que le CD est suffisamment fixé sur le support en bois du moteur et qu'il ne risque pas de se détacher. Aussi ne vous tenez pas trop prêt du zootrope pour ne pas vous blesser). En activant l'interrupteur Marche/Arrêt, vous devriez voir la séquence d'images fixes s' "animer" !
Si l'animation est trop rapide ou trop lente, tournez la tige du potentiomètre afin d'ajuster la Vitesse de défilement des images.

Essayer d'appuyer sur le bouton poussoir de Direction afin de voir à quoi ressemble l'animation lorsque qu'elle est passée à l'envers. Le zootrope et les images fournis avec le kit sont simplement un point de départ: essayez de produire vos propres animations, en utilisant le bandeau fourni comme modèle.

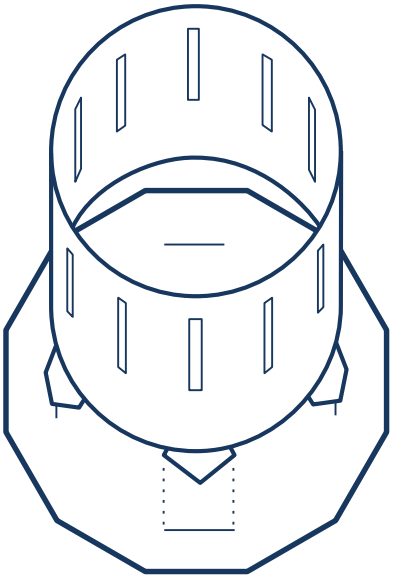
Pour cela, commencez par dessiner une image simple. Ensuite, effectuez de petites modifications entre chacune des images suivantes. Enfin, essayez de graduellement revenir au dessin de l'image originale afin que vous puissiez jouer l'animation en boucle, sans coupure, indéfiniment.



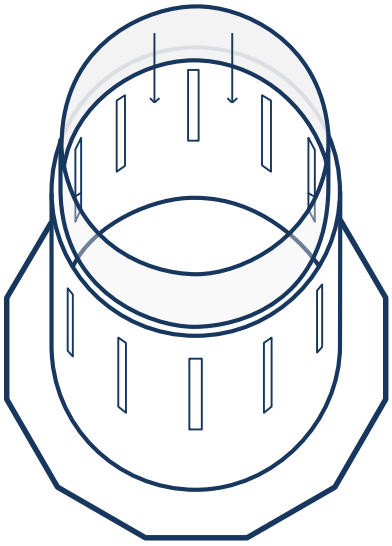
1. Fixez le CD sur le support en bois. Ajouter une goutte de colle afin que le CD ne glisse pas au démarrage du moteur.



2. Utilisez les languettes pour fermer le bandeau, et former un cercle



3. Insérer les 4 languettes sur la base du zootrope.



4. Insérer la bande de papier avec les images à l'intérieur du zootrope

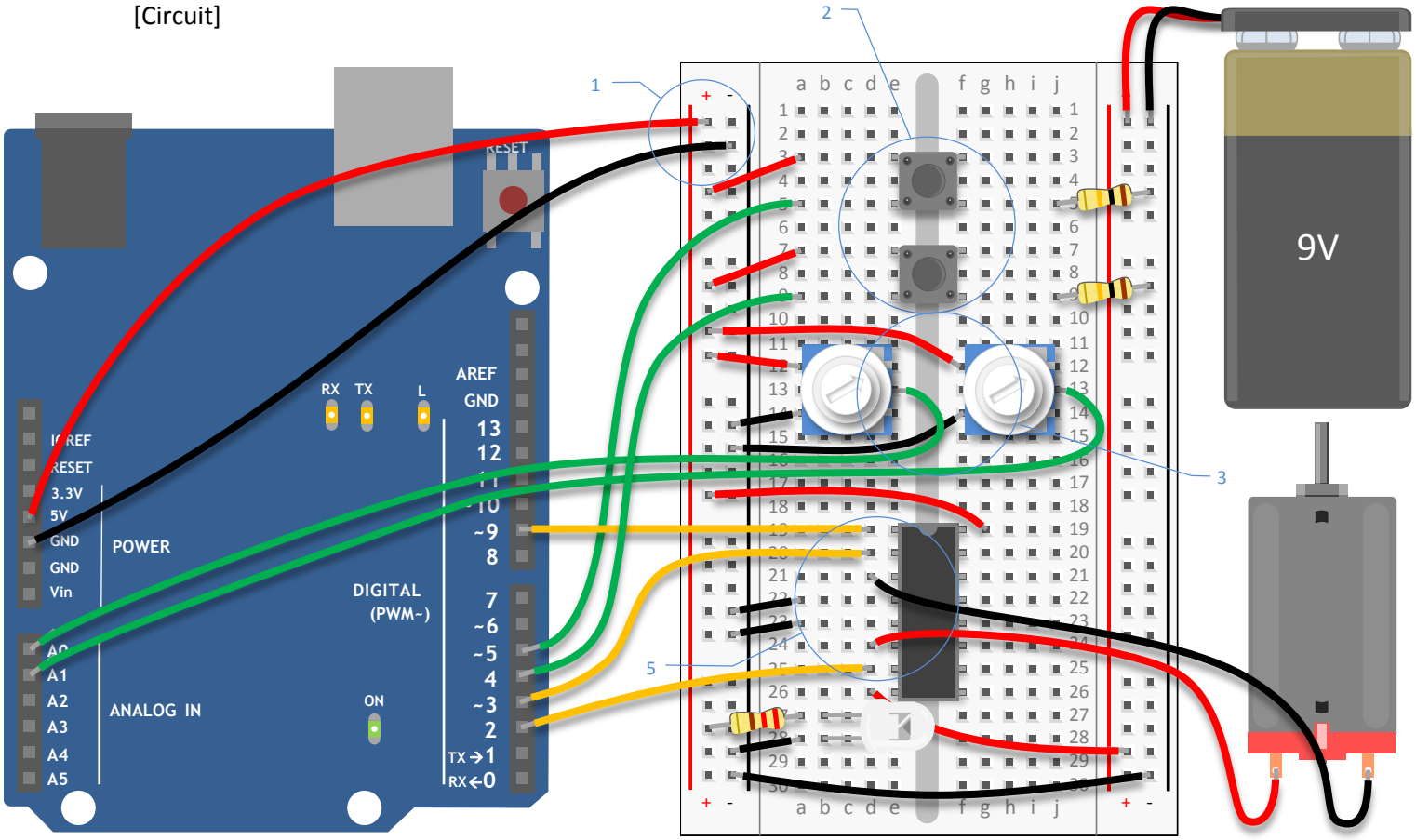
Les zootropes fonctionnent grâce à un phénomène appelé "persistance rétinienne" ('Persistence of Vision' en anglais, quelquefois abrégé en POV). La POV explique l'illusion du mouvement créée lorsque nos yeux observent une succession rapide d'images fixes comportant d'infimes variations entre elles. Si vous entrez "POV display" sur un moteur de recherche, vous trouverez de très nombreux projets créés par des gens qui exploitent cet effet, et souvent à base de LEDs et d'un Arduino.

Construisez une base pour soutenir le moteur. Une boite de cartes à jouer avec une ouverture pratiquée sur le dessus pourra très bien servir de base, permettant de libérer vos mains pour utiliser les boutons poussoirs et le potentiomètre. Cela rendra aussi plus simple la démonstration du résultat de votre travail aux autres personnes.

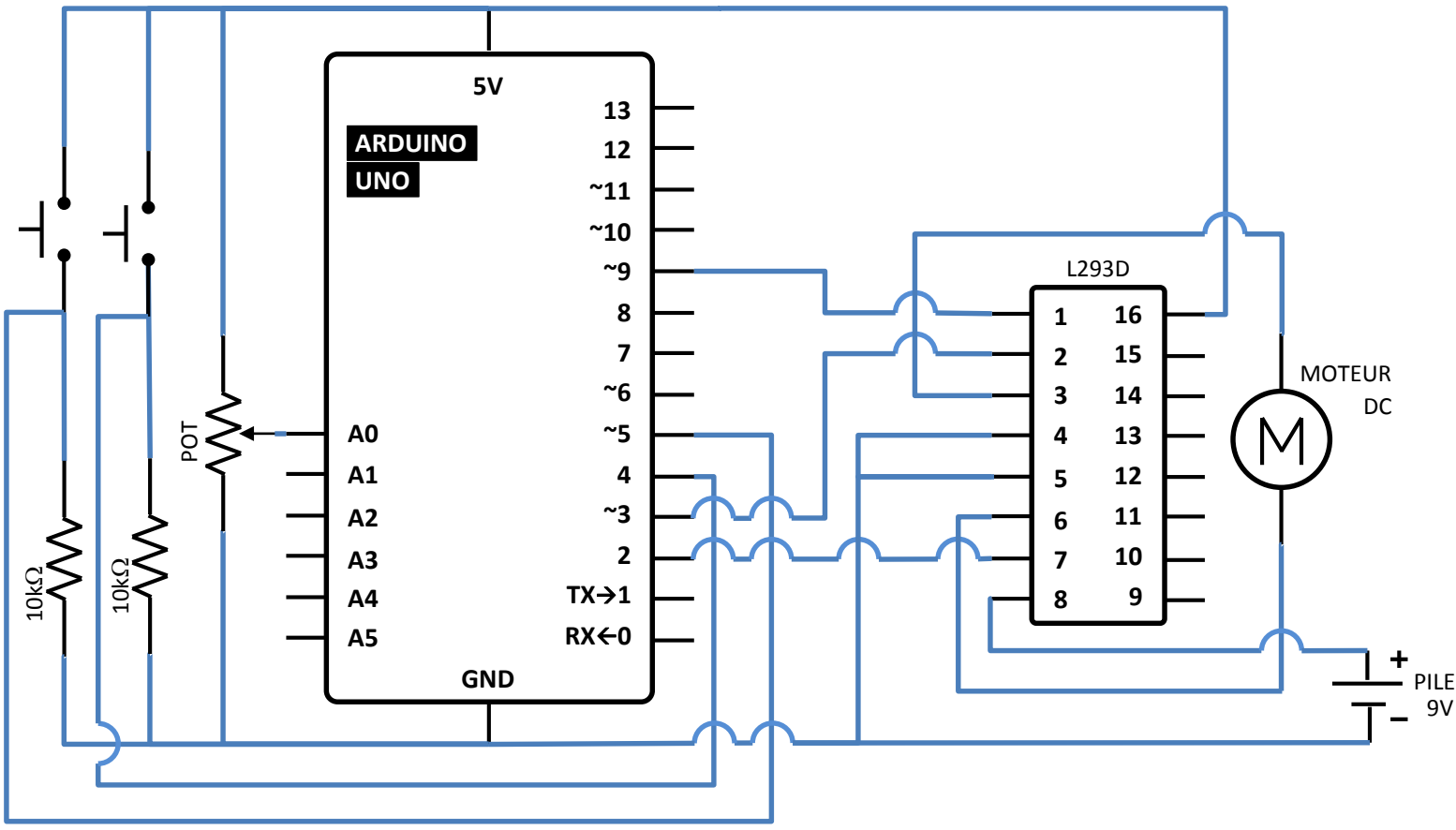
Avec un peu de travail supplémentaire, vous pourrez permettre à votre zootrope de fonctionner même dans la pénombre. Branchez une LED et une résistance à l'une des broches de sortie numérique disponible sur l'Arduino. Ajoutez aussi un second potentiomètre, et connectez-le à une entrée analogique. Positionnez la LED de façon à ce qu'elle éclaire directement les images. En utilisant l'entrée analogique pour définir la fréquence du flash produit par la LED, vous réglerez le système afin que ce flash soit produit juste au moment où l'image arrive devant vos yeux. Il faudra un peu de temps pour que vous réussissiez à régler correctement les 2 potentiomètres (Fréquence du flash, Vitesse du moteur), mais le résultat final sera vraiment spectaculaire!

CONSTRUIRE LE CIRCUIT

[Circuit]



[Schéma]



1) Branchez l'alimentation et la masse à votre platine d'expérimentation à partir de l'Arduino.

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.

Conçu, imprimé et relié à Turin, Italie
Septembre 2012

Première réimpression, Décembre 2012

Traduit de l'anglais par Nicolas PONCET