Moulinet motorisé

Faire tourner une roue multicolore grâce à un moteur piloté par Arduino

Découverte : Les transistors et les charges à haute intensité/tension

Durée : 45 minutes

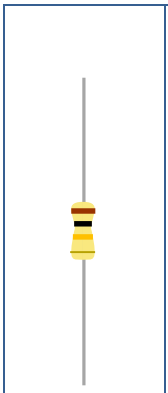
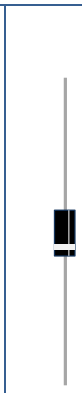
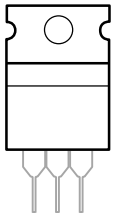
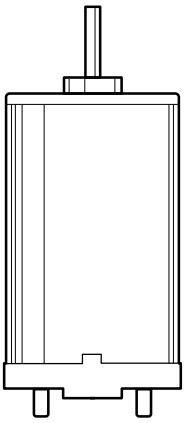
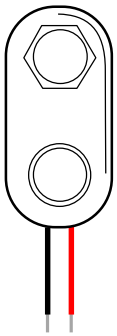
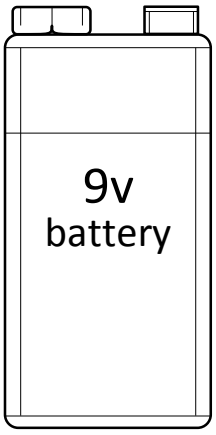
Difficulté : ■■■□□

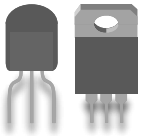
Basé sur les projets : 1, 2, 3, 4

Contrôler des moteurs avec un Arduino est plus compliqué que contrôler de simples LEDs pour 2 raisons. Premièrement, les moteurs nécessitent plus de courant que ce que peuvent fournir les broches de sortie de l'Arduino, ensuite, les moteurs sont capables de générer leur propre courant grâce à un processus appelé l'induction, ce qui peut endommager votre circuit si vous n'avez pas tenu compte de cette spécificité. Cependant, les moteurs permettent de mettre en mouvement des objets physiques, rendant vos projets beaucoup plus excitant. Cela vaut bien quelques difficultés!

Mettre en mouvement des choses consomme beaucoup d'énergie. Les moteurs nécessitent généralement plus de courant que ce que l'Arduino peut fournir. Certains moteurs nécessitent aussi une tension plus élevée. Pour commencer à tourner, en particulier quand il a une charge lourde attachée, un moteur consommera autant de courant que possible. L'Arduino ne peut fournir 40 milliampères (mA) à partir de ses broches numériques, beaucoup moins que ce que la plupart des moteurs ont besoin pour travailler.

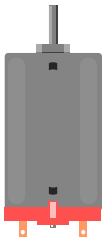
INGREDIENTS

						
RESISTANCE 10 kΩ	DIODE 1N4007	MOSFET	INTERRUPTEUR	MOTEUR DC	CONNECTEUR PILE 9V	PILE
x1	x1	x1	x1	x1	x1	x1



Les transistors sont des composants qui vous permettent de contrôler les sources d'alimentation courant et tension élevée à partir des sorties à faible courant de l'Arduino. Il existe beaucoup de types différents, mais ils fonctionnent tous sur le même principe. Vous pouvez imaginer que les transistors sont des interrupteurs numériques. Lorsque vous fournissez une tension à une des broches du transistor, appelée la **Grille**, il ferme le circuit entre les deux autres broches, appelé la **Source** et le

Drain. De cette façon, vous pouvez mettre en marche ou arrêter un moteur nécessitant une grande valeur de courant/tension avec votre Arduino.



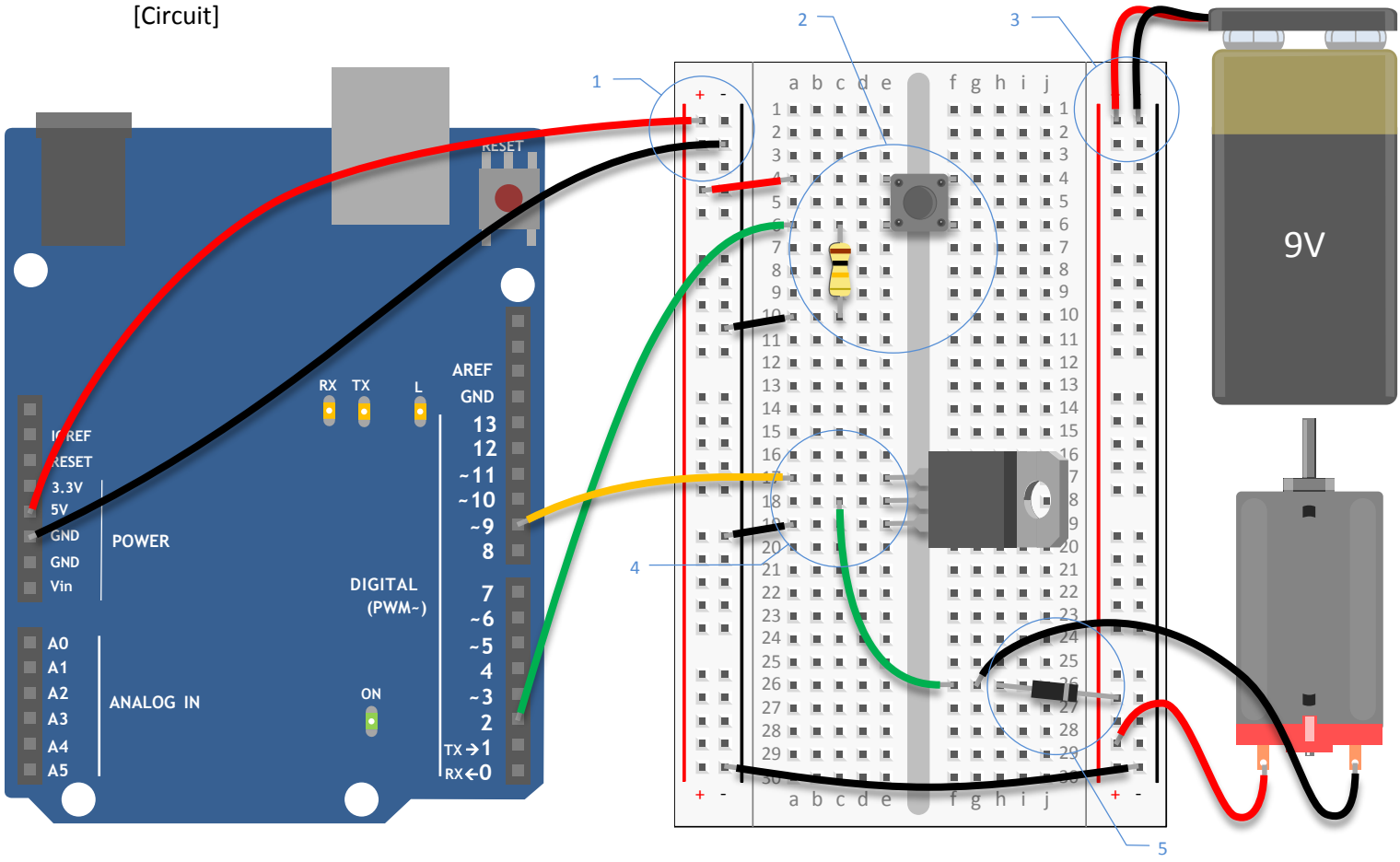
Les moteurs sont des dispositifs de type *inductif*. L'induction est un processus par lequel la variation du courant électrique dans un fil entraîne une variation du champ magnétique autour de ce fil. Lorsqu'un moteur électrique est alimenté en électricité, une bobine de cuivre étroitement enroulée à l'intérieur du boitier crée le champ magnétique. Ce champ entraîne alors la rotation de l'arbre du moteur (la partie qui dépasse du boîtier).

L'inverse est également vrai : un moteur peut produire de l'électricité lorsque l'arbre est mis en rotation. Essayez de connecter une LED entre les deux bornes de votre moteur, puis tournez l'arbre à la main. Si rien ne se passe, faites tourner l'arbre dans l'autre sens. La LED doit s'allumer. Vous avez créé un mini générateur de courant avec votre moteur.

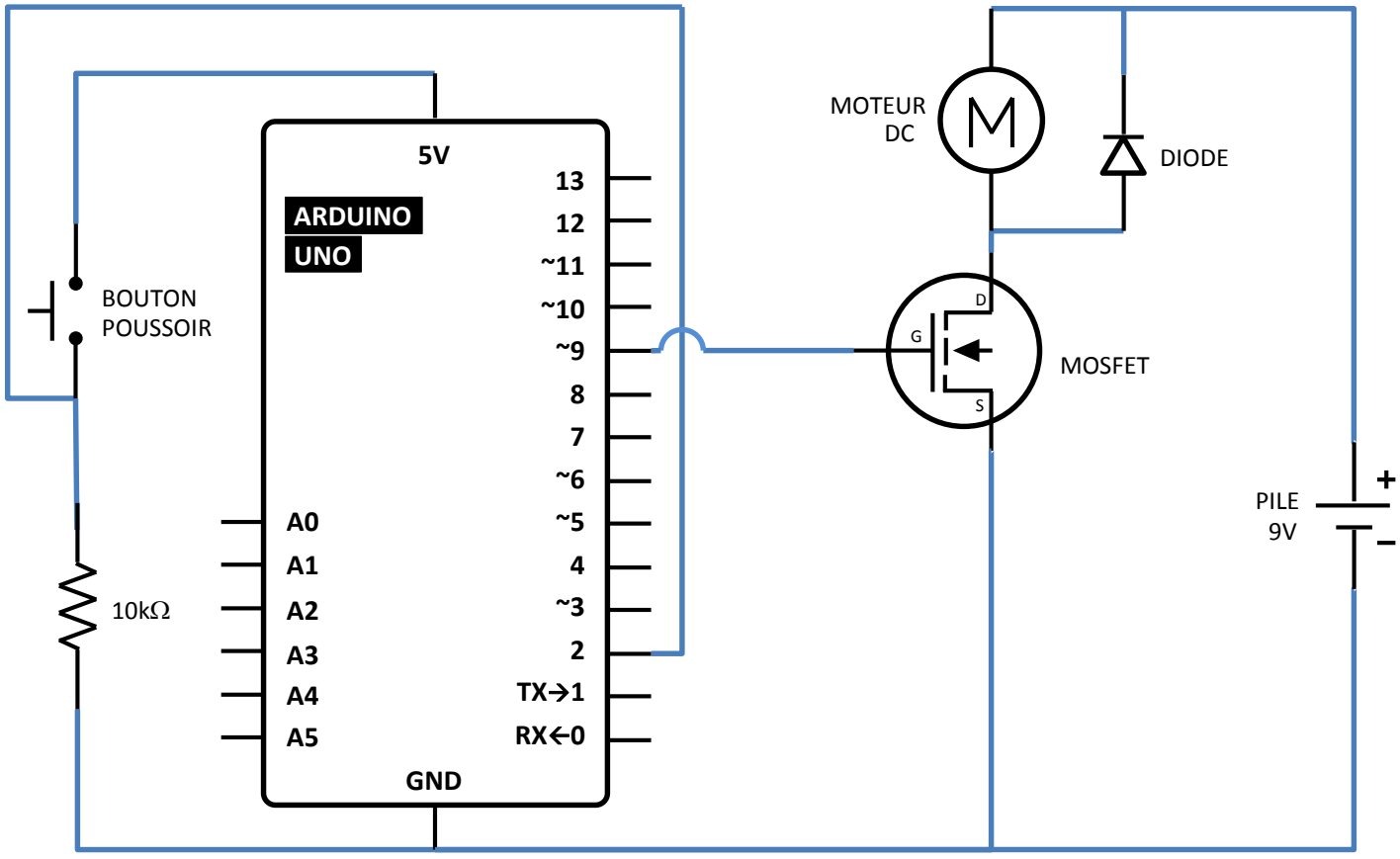
Lorsque vous arrêtez de fournir de l'énergie à un moteur, il va continuer à tourner, car il a de l'inertie. En tournant, il va générer une tension orientée dans le sens opposé à la tension que vous lui avez donné. Vous avez vu cet effet lors vous avez utilisé le moteur pour allumer une LED. Cette tension inverse parfois appelée **tension inverse**, peut endommager votre transistor. Pour cette raison, vous devriez mettre une diode en parallèle avec le moteur, afin que la **tension-inverse** passe à travers la diode. La diode ne permettra à l'électricité de ne circuler que dans une seule direction, en protégeant le reste du circuit.

CONSTRUIRE LE CIRCUIT

[Circuit]



[Schéma]



1) Branchez l'alimentation et la masse à votre platine d'expérimentation à partir de l'Arduino
2) Ajoutez un bouton poussoir à la carte, en connectant un côté à l'alimentation et l'autre côté à la broche numérique 2 de l'Arduino. Ajouter une résistance "pull-down" de 10 kΩ entre la masse et la broche de l'interrupteur connectée à l'Arduino.
3) Lorsque vous utilisez des circuits avec différentes tensions, vous devez vous connecter leur masse ensemble pour créer une masse commune. Branchez le connecteur de pile 9V à la platine d'expérimentation. Connectez la masse de la pile (pole -) à la masse de votre Arduino sur la platine d'expérimentation avec un cavalier, comme le montre la Fig. 1. Ensuite, brancher l'autre borne de la pile (pole +) au 9V sur la platine d'expérimentation.
4) Placez le transistor sur la carte. Placez le composant de sorte que la languette métallique est face à vous, à l'opposé de l'Arduino. Connectez la broche numérique 9 à la broche à gauche sur le transistor. Cette borne est appelée la **Grille**. Une variation de la tension sur la **Grille** assure une liaison entre les deux autres broches. Connecter une extrémité du moteur à la broche intermédiaire du transistor. Cette borne est appelé le **Drain**. Lorsque l'Arduino active le transistor en appliquant une tension sur la Grille, cette borne sera connectée à la troisième borne, appelée la **Source**. Connectez la source à la masse.
5) Ensuite, connectez les fils d'alimentation du moteur au moteur à la platine d'expérimentation. Le dernier composant à ajouter est la diode. La diode est un composant polarisé, il ne peut aller que dans un sens dans le circuit. Remarquons que la diode possède une bande à une extrémité. Cette extrémité est la broche négative, ou cathode, de la diode. L'autre extrémité est la broche positive, ou anode. Connecter l'anode de la diode à la masse du moteur et la cathode de la diode à l'alimentation 9V du moteur. Voir Fig. 1. Cela peut sembler être à l'envers, et en fait, ça l'est. La diode servira à empêcher que la tension inverse générée par le moteur retourne dans votre circuit. Rappelez-vous, la tension inverse circule dans le sens inverse de la tension que vous fournissez.

Les LEDs sont aussi des diodes, au cas où vous vous demanderiez pourquoi leurs broches sont aussi appelées anodes et cathodes. Il y a quantité de types de diodes, mais elles partagent toutes la même caractéristique. Elles permettent au courant de circuler de l'anode vers la cathode, mais pas l'inverse.

LE PROGRAMME

Définissez les constantes et les variables

Le programme est relativement similaire au 1er programme que vous avez créé pour allumer une LED. Avant tout, définissez les constantes correspondant à la broche d'état de l'interrupteur **switchPin** `(interrupteurBroche)` et celle de commande du moteur **motorPin** `(moteurBroche)`, ainsi qu'une variable **switchState** (`interrupteurEtat`) qui stockera l'état de l'interrupteur.

```
const int switchPin   = 2;
const int motorPin    = 9;
      int switchState = 0;
```

Déclarer le mode d'utilisation des broches

Dans le **setup**(), déclarer le **pinMode**() des broches du moteur (**OUTPUT**) et de l'interrupteur (**INPUT**).

```
void setup() {
       pinMode(switchPin, INPUT);
       pinMode(motorPin, OUTPUT);
}
```

Lire l'entrée, et placer la sortie à l'état haut si l'interrupteur est appuyé

La fonction **loop**() est simple. Vérifier l'état de la broche **switchPin** avec **digitalRead**().

Si l'interrupteur est appuyé, mettre la broche **motorPin** à l'état haut (**HIGH)**. S'il n'est pas appuyé, mettre la broche à l'état bas (**LOW**). A l'état **HIGH**, le transistor est activé, on dit qu'il est *passant*, le circuit alimentant le moteur est fermé, donc il tourne. A l'état **LOW**, le transistor n'est pas activé, on dit qu'il est *bloquant*, le circuit alimentant le moteur est ouvert, il ne tourne donc pas.

```
void loop() {
       switchState = digitalRead(switchPin);

       if (switchState == HIGH) {
              digitalWrite(motorPin, HIGH);
       } else {
              digitalWrite(motorPin, LOW);
       }
}
```

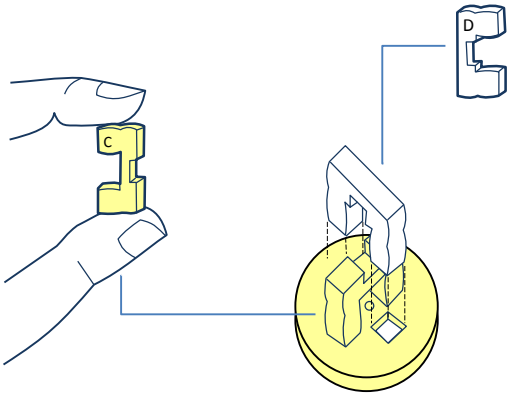
Les moteurs ont une valeur optimale de leur tension de fonctionnement. Ils pourront fonctionner dès que la tension atteint 50% de cette valeur et jusqu'à 50% au-delà de cette valeur. Si vous faites varier la tension, vous pourrez modifier la vitesse à laquelle le moteur tourne. Cependant, ne le faites pas varier au-delà de la limite, ou vous pourriez griller votre moteur.

Lorsqu'ils sont pilotés par des micro-contrôleurs, les moteurs nécessitent des dispositions particulières. Habituellement, un micro-contrôleur ne peut pas fournir assez de courant et/ou de tension pour alimenter directement un moteur. A cause de cela, vous devez utiliser des transistors pour faire l'interface entre les deux. C'est aussi une bonne chose d'utiliser des diodes pour éviter tout dommage au circuit.

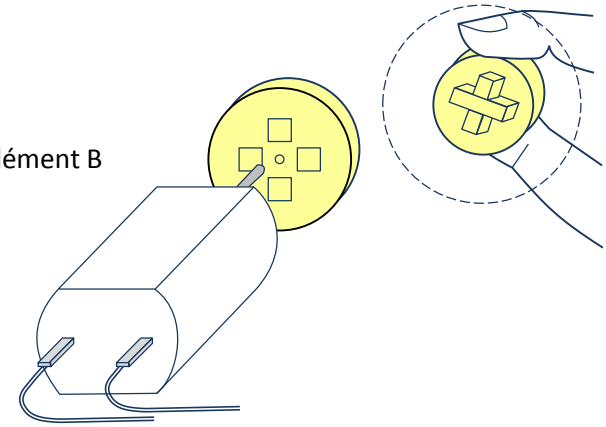
Les transistors sont des composants statiques, c'est à dire qu'ils n'ont pas de parties mobiles. Grâce à cette caractéristique vous pouvez les activer et les éteindre très rapidement. Essayer de brancher un potentiomètre à une entrée analogique et d'utiliser la valeur lue pour piloter la broche PWM qui pilote le transistor. Que pensez-vous qu'il arrivera à la vitesse de rotation du moteur si vous faites varier la tension qu'il reçoit ? Avec les illustrations sur votre disque tournant, pouvez-vous obtenir différents effets visuels ?

UTILISER LE MONTAGE

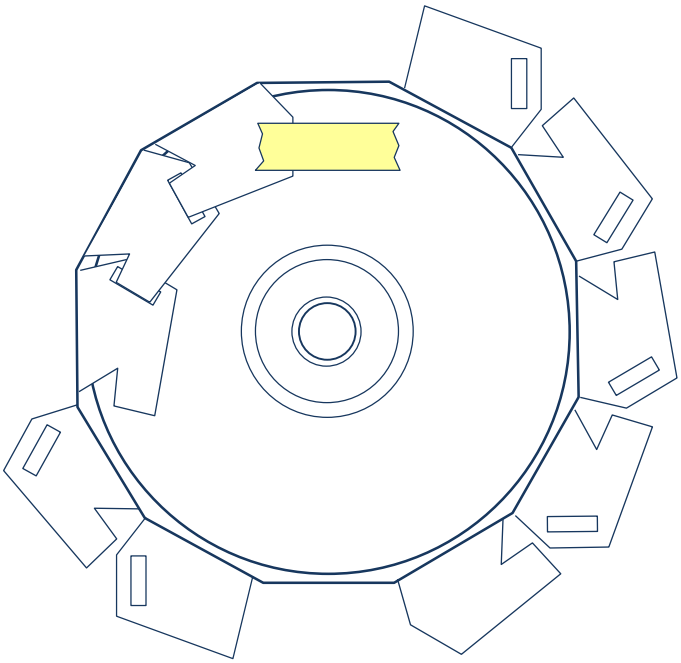
Assemblez le support CD comme illustré à l'étape 1, et fixez le au moteur comme illustré à l'étape 2. Fixez la forme prédécoupée à un CD comme indiqué à l'étape 3. Clipsez le CD au support et sécurisez le tout avec une pointe de colle. Laissez sécher avant de continuer. Branchez une pile 9V au connecteur de pile 9V. Alimentez l'Arduino via le port USB. Lorsque vous appuierez sur l'interrupteur situé sur la platine d'expérimentation, le moteur se mettra à tourner très rapidement.



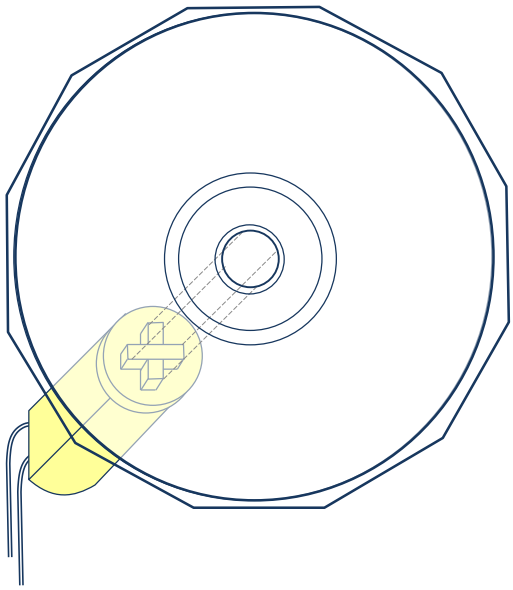
1. Insérer l'élément C dans l'élément B, et enficher doucement l'élément D par-dessus.



2. Insérer l'axe du moteur dans le trou situé à l'arrière de l'élément B



3. Placer la forme prédécoupée en papier sur un CD et rabattez les volets vers l'arrière pour solidariser l'ensemble.

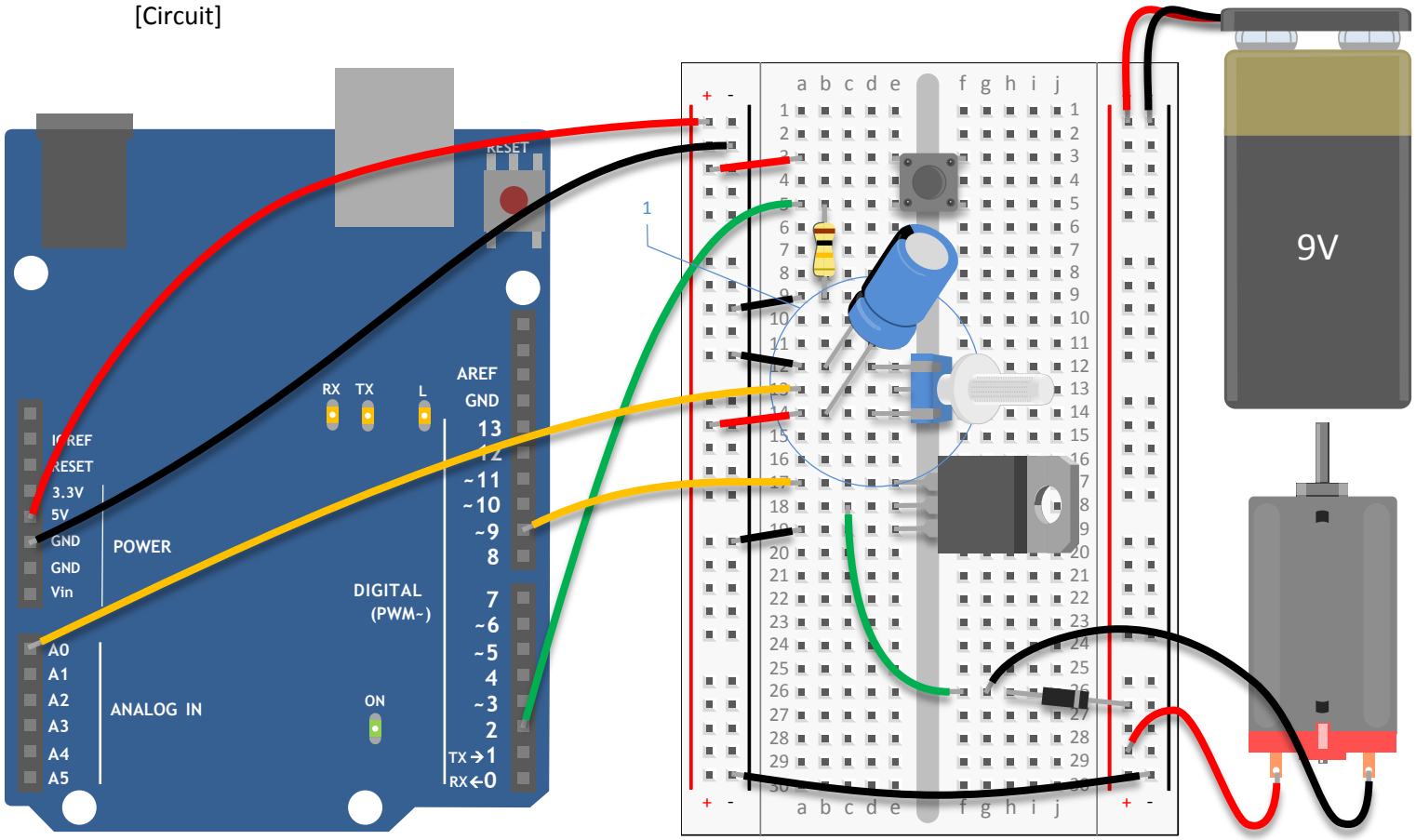


4. Fixer le CD à la croix formée par les éléments C et D. Utilisez une pointe de colle pour empêcher le CD de se détacher.

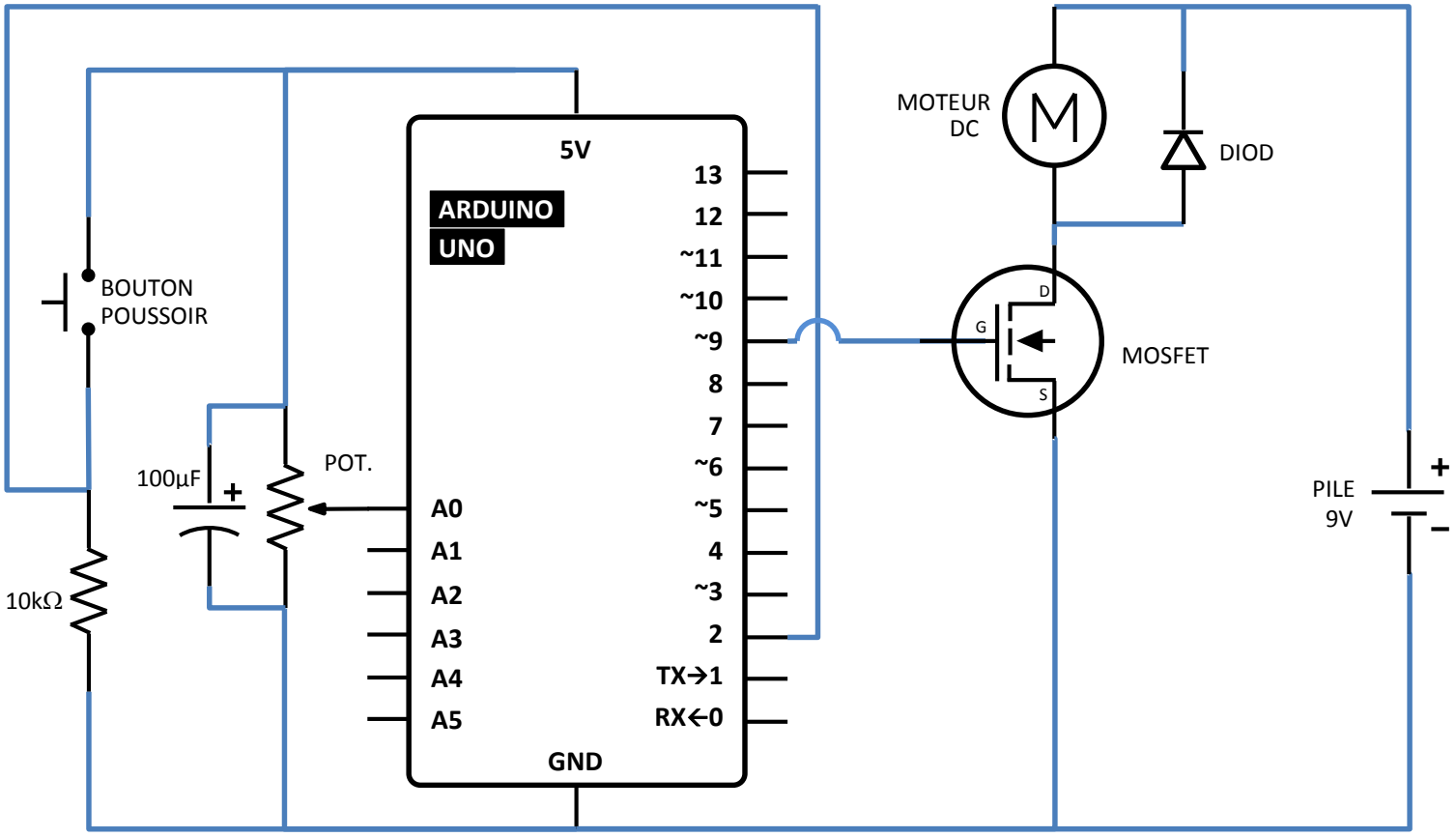
Le moteur tournant très rapidement, vous pourrez probablement aussi fabriquer une grande hélice. Attention cependant à ce qu'elle ne se détache pas et blesse quelqu'un. Essayez différentes formes et illustrations et comparer les effets visuels obtenus.

CONSTRUIRE LE CIRCUIT

[Circuit]



[Schéma]



1) Ajoutez un potentiomètre sur la plaque d'expérimentation, et connectez une des broches sur le coté sur la masse et celle de l'autre coté sur le +5V. Un potentiomètre est en quelque sorte un pont-diviseur de tension. Lorsque vous tourner le bouton, vous modifiez le ratio de tension entre la broche centrale du potentiomètre et l'alimentation. Vous pouvez lire cette modification sur une entrée analogique. Connectez la broche centrale à la broche de l'entrée analogique A0. Cela permettra de contrôler la vitesse de rotation du moteur.
   En plaçant un condensateur de 100µF entre les broches du potentiomètre reliées à la masse et l'alimentation. Ce condensateur est appelé "condensateur de découplage" parce qu'il réduit, ou découple, les modifications causées par les composants du reste du circuit. Soyer très prudent et assurez-vous que vous avez connecté la cathode à la masse (c'est le côté avec une bande noir) et l'anode à l'alimentation. Si vous mettez le condensateur à l'envers, il pourrait exploser.

LE PROGRAMME

Définissez les constantes et les variables

Le programme est similaire au programme précédent, vous allez simplement modifier l'utilisation de l'interrupteur et ajouter le pilotage de la vitesse du moteur via le potentiomètre.

Le moteur restera allumé ou éteint entre deux pressions sur l'interrupteur permettant de libérer vos mains pour manipuler le potentiomètre.

Le potentiomètre, permettra de mettre en marche le moteur à partir d'une valeur limite permettant de délivrer une tension minimale au moteur, compatible avec sa tension minimale de démarrage.

Comme lors des projets précédents, le programme permet de calibrer les positions 'Min' et 'Max' du potentiomètre pendant 5 secondes après le démarrage du montage. La LED 'L' reste allumée pendant ce laps de temps.

Enfin, les valeurs lues sur les broches du potentiomètre et la valeur correspondante transmise au moteur sont affichées dans la console Série

Avant tout, définissez les constantes correspondant à la broche d'état de l'interrupteur **switchPin** `(interrupteurBroche)`, celle de commande du moteur **motorPin** `(moteurBroche)`, celle de la LED 'L' **LEDPin** (`LEDBroche`), et celle du potentiomètre **potPin** (`potentiometreBroche`).

```
// Pins
const int switchPin = 2;
const int motorPin  = 9;
const int LEDPin    = 13; // Built-in 'L' LED
const int potPin    = A0;
```

Ensuite, définissez les constantes correspondant à la vitesse du moteur lorsque le potentiomètre est en position 'Max' **motMax** (`moteurMax`), à la valeur de démarrage du moteur **motThreshold** (`moteurSeuil`), et à la durée en milliseconde (ms) de la période de calibrage du potentiomètre **calibrationTime** (`calibrageTemps`).

```
// Constants
const int motMax          = 150;  // max motor speed [0 - 255]
const int motThreshold    = 20;   // min motor speed [0 - 255]
const int calibrationTime = 5000; // in ms
```

Enfin initialisez les autres variables nécessaires au fonctionnement du programme.

Premièrement, les valeurs analogiques **potMin** et **potMax** qui contiendront, après le calibrage du potentiomètre, les valeurs des positions 'Min' et 'Max' du potentiomètre, puis **potValue** et **motorValue**, qui contiendront respectivement la valeur lue sur le potentiomètre et celle transmise au moteur.

```
// Variables
int potMin     = 1023; // Calibration minimum sensor value
int potMax     = 0;    // Calibration maximum sensor value
int potValue   = 0;
int motorValue = 0;
```

Deuxièment, les valeurs numériques **LEDstate** (`LEDEtat`), qui déterminera si la LED 'L' est allumée ou éteinte, **switchState** (`interrupteurEtat`) qui contient l'état actuel de l'interrupteur, et **previousSwitchState** (`precedentInterrupteurEtat`) qui contient l'état de l'interrupteur lors de la précédente lecture.

```
int LEDState            = LOW;
int previousSwicthState = LOW;
int switchState         = LOW;
```

Déclarer le mode d'utilisation des broches

Dans le **setup**(), ouvrez la liaison série et déclarer le **pinMode**() des broches du moteur (**OUTPUT**), de la LED (**OUTPUT**), du potentiomètre (**INPUT**) et de l'interrupteur (**INPUT**).

```
void setup() {

    Serial.begin(9600);

    pinMode(switchPin, INPUT );
    pinMode(motorPin , OUTPUT);
    pinMode(potPin   , INPUT );
    pinMode(LEDPin   , OUTPUT );
```

Ensuite, effectuez le calibrage du potentiomètre

```
// Calibrate during the first five seconds

    // Light on the LED 'L'
    digitalWrite(LEDPin, HIGH);

    while (millis() < calibrationTime ) {
        potValue = analogRead(potPin);

        // record the maximum sensor value
        if (potValue > potMax) {
            potMax = potValue;
        }

        // record the minimum sensor value
        if (potValue < potMin) {
            potMin = potValue;
       }
    }

    // Light off the LED 'L'
    digitalWrite(LEDPin, LOW);

}
```

Lire l'entrée sur le potentiomètre, et placer la sortie à l'état correspondant si l'interrupteur a été appuyé

Dans la fonction **loop**(), vérifier l'état de la broche **switchPin** avec **digitalRead**() afin de déterminer si l'interrupteur a changé de position depuis la dernière vérification.

Si l'interrupteur est enfoncé, changer l'état du moteur. S'il était arrêté, mettre la broche **motorPin** à l'état haut (**HIGH**), s'il était démarré, mettre la broche à l'état bas (**LOW**). A l'état **HIGH**, le transistor est activé, on dit qu'il est *passant*, le circuit alimentant le moteur est fermé, donc il tourne. A l'état **LOW**, le transistor n'est pas activé, on dit qu'il est *bloquant*, le circuit alimentant le moteur est ouvert, il ne tourne donc pas. La LED 'L' reflète l'état du moteur : Allumée, le moteur est en marche, éteinte le moteur est arrêté.

```
void loop() {
    switchState = digitalRead(switchPin);

    // If the switch changes its state (ON/OFF), change the motor state (STOP/START)
    if (previousSwicthState != switchState) {
        previousSwicthState = switchState;
        if (switchState == HIGH) { // Button is pressed
            if (LEDState == HIGH) {
                LEDState = LOW;    // Motor is now OFF
                analogWrite(motorPin,0);
            } else {
                LEDState = HIGH;   // Motor is now ON
            }
            digitalWrite(LEDPin, LEDState);
        }
    }
```

Ainsi, la variable d'état de la LED est aussi utilisée pour déterminer si le moteur est en marche.

Dans ce cas, vérifier la valeur de la broche **potPin** avec **analogRead**() afin de déterminer la position du potentiomètre potValue entre [potMin,potMax], et calculer la valeur de la tension à envoyer au moteur sur la plage [0, motMax].

Ces 2 valeurs sont affichées sur la liaison série.

Par précaution, afin de n'endommager ni la pile 9V, ni le moteur, si la valeur calculée pour alimenter le moteur est inférieure à la valeur minimale permettant au moteur de démarrer, la valeur est forcée à 0 afin de laisser le moteur à l'arrêt. De même, si la valeur calculée est en dehors des limites admises par Arduino [0, 255], la valeur est repositionnée dans l'intervalle.

Enfin, la valeur est envoyée via **analogWrite**() à la broche **motorPin**.

```
    // PWM
    if (LEDState == HIGH) {
        potValue   = analogRead(potPin);
        motorValue = map(potValue,potMin,potMax,0,motMax);

        Serial.print("Potentiometre : ");
        Serial.print(potValue);
        Serial.print(" / Moteur : ");
        Serial.println(motorValue);

        // in case the motor value is less than the min value necessary to move
        if (motorValue<motThreshold) {
            motorValue = 0;
        }

        // in case the sensor value is outside the range seen during calibration
        motorValue = constrain(motorValue, 0, 255);

        analogWrite(motorPin,motorValue);
    }
}
```

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