Montage

14 Le moteur pas-à-pas

Au sommaire :

- comprendre ce qu'est un moteur pas-à-pas et savoir le commander ;
- le sketch complet ;
- l'analyse du schéma ;
- la réalisation du circuit ;
- un exercice complémentaire.

Encore plus de mouvement

Un servomoteur permet de transformer le courant électrique en mouvement. Cependant, son rayon d'action demeure limité, même si des modifications peuvent être entreprises pour combler ce manque. Mais il s'avère suffisant pour la plupart des applications.

Le moteur pas-à-pas s'impose en revanche si une plus grande liberté d'action est nécessaire. Voyez par souci d'économie si vous ne pouvez pas en récupérer un sur un vieil appareil quelconque :

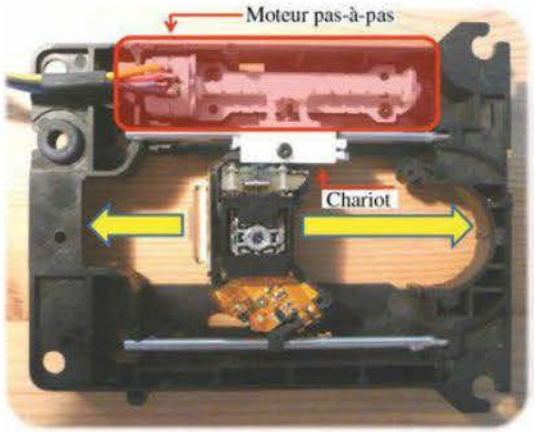
- imprimante ;
- scanner à plat ;
- lecteur de CD/DVD ;
- ancien lecteur de disquettes (de 3,5 pouces).

Sur la figure 14-1, vous pouvez voir un lecteur de disquettes de 3,5 pouces, encore parfois utilisé dans les ordinateurs actuels.

Figure 14-1 ▶
Lecteur de disquette de 3,5 pouces

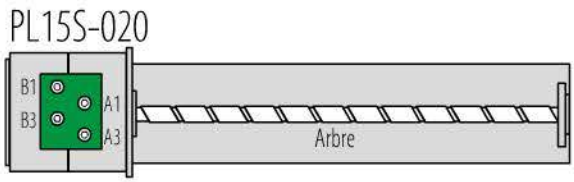
Ces lecteurs possèdent un petit moteur pas-à-pas, de type PL15S-020 la plupart du temps, qui entraîne un petit chariot sur lequel se trouve la tête d'écriture/lecture. La figure 14-2 montre une unité de ce genre provenant d'un ancien lecteur de CD-Rom.

Figure 14-2 ▶
Moteur pas-à-pas PL15S-020 provenant d'un ancien lecteur de CD-Rom



Le moteur pas-à-pas dispose de 4 bornes, sur lesquelles nous allons revenir en détail. Comme vous pouvez le voir sur la figure, j'ai déjà soudé deux fils de couleur pour qu'il soit plus facile de le commander à la main avec la carte Arduino. Le graphique 14-3 montre les noms utilisés pour ces bornes.

Figure 14-3 ▶
Branchements du moteur pas-à-pas PL15S-020



Vu qu'il dispose de 4 bornes, il s'agit d'un moteur pas-à-pas bipolaire. Pour le mettre en marche, les bornes en question doivent recevoir certaines impulsions dans un ordre chronologique bien déterminé.

dans le sens des aiguilles d'une montre

	Bornes			
PAS	A1	A2	B1	B3
1	LOW	HIGH	HIGH	LOW
2	LOW	HIGH	LOW	HIGH
3	HIGH	LOW	LOW	HIGH
4	HIGH	LOW	HIGH	LOW

Figure 14-4
Séquences de commande pour moteur pas-à-pas PL15S-020

Quand on écrit un sketch pour traiter successivement les pas 1 à 4 et envoyer les niveaux `LOW` ou `HIGH` correspondant au moteur pas-à-pas, ce dernier tourne dans le sens horaire. Quand l'ordre des pas est inversé, le sens est antihoraire. Il y a une chose importante que je n'ai pas encore dite : on ne peut pas se contenter de raccorder le moteur pas-à-pas aux sorties numériques, car elles seraient alors sollicitées jusqu'à temps que la carte en pâtisse. Aussi utilise-t-on ici un circuit de commande de moteur de type L293 (voir figure 14-5).

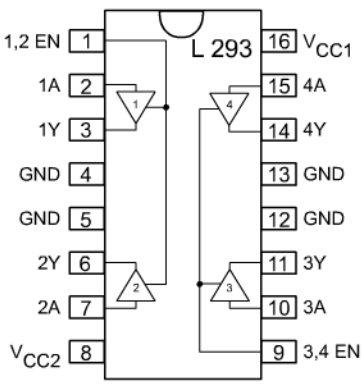


Figure 14-5
Commande de moteur de type L293DNE

Les petits triangles sont le symbole de l'interface nécessaire pour fournir la puissance dont un moteur raccordé a besoin pour fonctionner. Les bornes A du circuit intégré sont les entrées et celles Y sont les sorties. Chaque paire de drivers a une borne de validation commune, libellée 1,2EN ou 3,4EN (EN pour *enable*, ou permettre). Ce circuit de commande de moteur peut fournir un courant de 600 mA par sortie. Les circuits de commande suivants sont capables de délivrer un courant plus élevé :

- SN754410 (1 ampère) ;
- L298 (2 ampères).

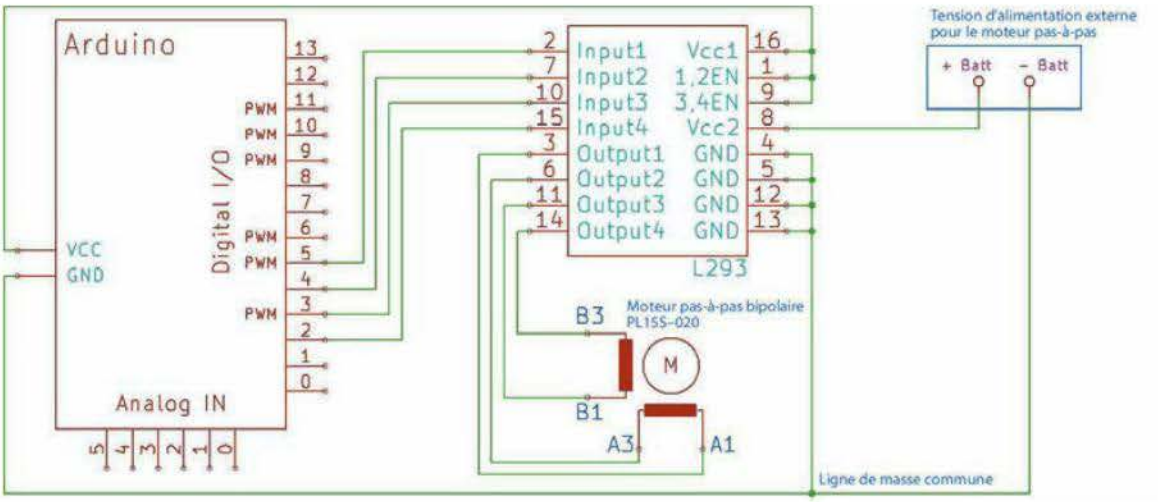


Figure 14-6▲
Commande du moteur pas-à-pas par le circuit L293DNE

Vous ne remarquez rien ? Eh bien, à droite du schéma se trouve une source de tension supplémentaire, qui est nécessaire pour alimenter le moteur pas-à-pas sous une tension distincte. En présence de deux sources de tension ou plus, il est toujours nécessaire d'interconnecter les lignes de masse pour obtenir un point de référence commun.

Attention !
Le pôle + de la carte Arduino ne doit jamais être raccordé à la source de tension externe. Sinon, destruction de la carte assurée !

La fiche technique indique que le moteur pas-à-pas a besoin de 5 V pour fonctionner. Si la tension d'alimentation du moteur pas-à-pas est inférieure, le positionnement ne sera ni précis ni reproductible. Atteindre une position déterminée avec précision et de manière répétitive tiendra alors plus du jeu de hasard que d'autre chose. Voici quelques données importantes sur le moteur pas-à-pas PL16S-020 :

- nombre de pas par tour : 20 ;
- type : bipolaire ;
- tension d'alimentation : 5 V ;
- résistance de la bobine par phase : 10 ohms.

Je pense que vous avez oublié quelque chose de capital ! Je crois me souvenir qu'une commande de moteur nécessite une diode de protection. Ne l'avez-vous pas dit tout au début ?

Effectivement, Ardus ! N'empêche que je ne l'ai pas oubliée. La mention DNE derrière L293 signifie que les diodes de protection, appelées également diodes de roue libre, sont déjà intégrées dans le circuit. C'est naturellement très pratique ! Si vous avez encore un ancien circuit L293 (sans mention DNE derrière), il faut absolument brancher les diodes de protection en externe. Faute de quoi, la carte Arduino sera endommagée.

Composants nécessaires

	1 circuit de commande de moteur L293DNE
	1 moteur pas-à-pas bipolaire (par exemple, PL15S-020 provenant d'un ancien lecteur de CD/DVD)
	Plusieurs cavaliers flexibles de couleurs et de longueurs diverses

Qu'est-ce que je fais si je ne trouve pas de lecteur de disquette ou CD/DVD ? Je ne peux pas faire l'expérimentation alors.

Pas de panique, Ardus ! Vous pouvez en fait prendre pratiquement n'importe quel moteur pas-à-pas bipolaire. Il vous suffit de trouver la fiche technique correspondante sur Internet pour en connaître les spécifications. Attention cependant au courant consommé par le moteur pas-à-pas en service ; comparez-le à celui pour le circuit de commande de moteur utilisé ici. Il ne doit en aucun cas dépasser 600 mA par borne. Faute de quoi, vous devez prendre soit un autre moteur pas-à-pas, soit un autre circuit de commande.

Quand je regarde le schéma, j'ai comme un léger problème avec la source de tension externe qui, d'après les données du moteur pas-à-pas, doit être de 5 V. Où voulez-vous que je la trouve ?

Vous devez utiliser soit une alimentation de laboratoire réglable, soit – et c'est encore moins cher – un bloc d'alimentation secteur (voir chapitre 8).

Code du sketch

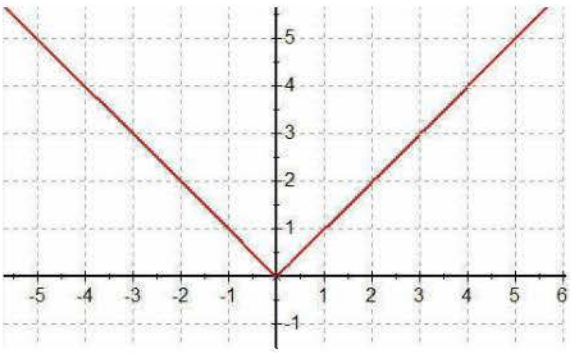
```
#define Stepper_A1 5  //Broche pour connexion A1
#define Stepper_A3 4  //Broche pour connexion A3
#define Stepper_B1 3  //Broche pour connexion B1
#define Stepper_B3 2  //Broche pour connexion B3

byte stepValues[5][4] = {{LOW, LOW, LOW, LOW},   //Moteur à l'arrêt
                         {LOW, HIGH, HIGH, LOW},  //Pas 1
                         {LOW, HIGH, LOW, HIGH},  //Pas 2
                         {HIGH, LOW, LOW, HIGH},  //Pas 3
                         {HIGH, LOW, HIGH, LOW}}; //Pas 4

void setup(){
  pinMode(Stepper_A1, OUTPUT);
  pinMode(Stepper_A3, OUTPUT);
  pinMode(Stepper_B1, OUTPUT);
  pinMode(Stepper_B3, OUTPUT);
  for(int i = 0; i < 10; i++){
    action(30, 2);   //30 pas vers la droite avec pause de 2 ms
    action(-30, 10); //30 pas vers la gauche avec pause de 10 ms
  }
  action(0, 0);     //Mise hors courant
}

void loop() {/* vide */}

void action(int count, byte delayValue){
  if(count > 0)    //Rotation vers la droite
    for(int i = 0; i < count; i++)
      for(int sequenceStep = 1; sequenceStep <=4; sequenceStep++)
        moveStepper(sequenceStep, delayValue);
  if(count < 0)    //Rotation vers la gauche
    for(int i = 0; i < abs(count); i++)
      for(int sequenceStep = 4; sequenceStep > 0; sequenceStep--)
        moveStepper(sequenceStep, delayValue);
  if(count == 0)   //Mise hors courant
    moveStepper(0, delayValue);
}

void moveStepper (byte s, byte delayValue){
  digitalWrite(Stepper_A1, stepValues[s][0]);
  digitalWrite(Stepper_A3, stepValues[s][1]);
  digitalWrite(Stepper_B1, stepValues[s][2]);
  digitalWrite(Stepper_B3, stepValues[s][3]);
  delay(delayValue); //Pause
}
```

Il me semble que vous utilisez dans ce sketch une fonction inconnue appelée `abs`. Pouvez-vous m'en dire un peu plus ?

Ah oui, c'est vrai. Si vous appliquez la fonction `abs` – qui est la forme abrégée de *absolute* – à un nombre réel, le signe n'est tout simplement pas pris en compte. Le résultat est toujours positif. Les mathématiciens formulent cet état comme suit :

$$|x| = \begin{cases} x[\text{pour } x \geq 0] \\ -x \text{ pour } x < 0 \end{cases}$$

Les deux barres verticales de chaque côté du x signifient « valeur absolue de x ». Le fonctionnement est plus clair grâce à cette représentation graphique que j'ai créée pour la fonction `abs` (voir figure ci-après).



Revue de code

D'un point de vue logiciel, les variables suivantes sont nécessaires à notre expérimentation du moteur pas-à-pas.

Tableau 14-1
Variable nécessaire et son rôle

Variable	Rôle
`stepValues[5][4]`	Tableau bidimensionnel pour stocker les informations de pas servant à déplacer le moteur

Le contenu du tableau correspond exactement aux valeurs du tableau des séquences de commande. Seule une ligne avec des valeurs `LOW` a été ajoutée au début, laquelle sert à mettre le moteur pas-à-pas hors courant, une fois la position demandée atteinte.

Si je ne le faisais pas, le moteur s'immobiliserait bien à la fin, mais nous aurions affaire à un blocage à la dernière position occupée. Un tel moteur ne peut plus être bougé à la main, car il est encore sous

tension. Par ailleurs, cela signifie qu'il devient vite très chaud voire brûlant.

```
byte stepValues[5][4] = {{LOW, LOW, LOW, LOW},   //Moteur à l'arrêt
                         {LOW, HIGH, HIGH, LOW},  //Pas 1
                         {LOW, HIGH, LOW, HIGH},  //Pas 2
                         {HIGH, LOW, LOW, HIGH},  //Pas 3
                         {HIGH, LOW, HIGH, LOW}}; //Pas 4
```

Voyons tout d'abord la fonction `moveStepper` qui déplace le moteur pas-à-pas. Elle comporte deux arguments.

- Le premier indique le pas dans la séquence, soit de 1 à 4 pour une rotation à droite et de 4 à 1 pour une rotation à gauche.
- Le second fixe un temps d'attente entre les pas. Vous pouvez ainsi influer un peu sur la vitesse du moteur pas-à-pas. Cette valeur ne doit cependant pas être inférieure à `2`, car la commande électrique est alors si rapide que le moteur n'a plus le temps de réagir mécaniquement. Il se contente de bourdonner et de vibrer.

```
void moveStepper(byte s, byte delayValue){
  digitalWrite(Stepper_A1, stepValues[s][0]);
  digitalWrite(Stepper_A3, stepValues[s][1]);
  digitalWrite(Stepper_B1, stepValues[s][2]);
  digitalWrite(Stepper_B3, stepValues[s][3]);
  delay(delayValue);   //Pause
}
```

À l'intérieur de la fonction, le pas est transmis comme index pour la première dimension du tableau de séquence `stepValues`. La deuxième dimension indique les niveaux de tension `LOW` ou `HIGH`. Ils sont dûment appelés au moyen des valeurs d'index de `0` à `3` et sont transmis aux sorties numériques qui, à leur tour, contrôlent le moteur pas-à-pas via le circuit de commande. Passons maintenant à la fonction `action`, qui appelle la fonction `moveStepper` :

```
void action(int count, byte delayValue){
  if(count > 0)       //Rotation vers la droite
    for(int i = 0; i < count; i++)
      for(int sequenceStep = 1; sequenceStep <=4; sequenceStep++)
        moveStepper(sequenceStep, delayValue);
  if(count < 0)       //Rotation vers la gauche
    for(int i = 0; i < abs(count); i++)
      for(int sequenceStep = 4; sequenceStep > 0; sequenceStep--)
        moveStepper(sequenceStep, delayValue);
  if(count == 0)       //Mise hors courant
    moveStepper(0, delayValue);
}
```

Le nombre de pas et le temps de pause après chaque pas lui sont communiqués. Le moteur tourne vers la droite si la valeur du pas est positive, et vers la gauche si elle est négative. Le moteur est mis hors courant si la valeur est `0`. Deux boucles `for` imbriquées travaillent toujours main dans la main pour faire tourner le moteur. La boucle extérieure régit le nombre de pas et la boucle intérieure fixe le sens de rotation. La boucle intérieure traite les pas de 1 à 4 si la valeur du pas est positive, et de 4 à 1 si elle est négative. Cette séquence sert d'index à la fonction `moveStepper`, index avec lequel le tableau `stepValues` détermine les valeurs `LOW` ou `HIGH` correspondantes. La demande proprement dite de mouvement du moteur pas-à-pas se fait par l'appel de la fonction `action`*n* avec une ligne de code comme suit :

```
action(30, 2);
```

Elle dit au moteur pas-à-pas : « Tourne de 30 pas vers la droite et observe une pause de 2 ms entre chaque pas ! » La ligne :

```
action(-30, 10);
```

dit en revanche : « Tourne de 30 pas vers la gauche et observe une pause de 10 ms entre chaque pas ! »

Le moteur pas-à-pas peut ainsi être déplacé à l'endroit souhaité. Cela étant, pensez aux limites mécaniques, car plus à gauche que le minimum ou plus à droite que le maximum n'est tout simplement pas possible. Dans ces cas-là, rien n'y fera, pas même une tension plus élevée.

Bibliothèque pour moteurs pas-à-pas prête à l'emploi

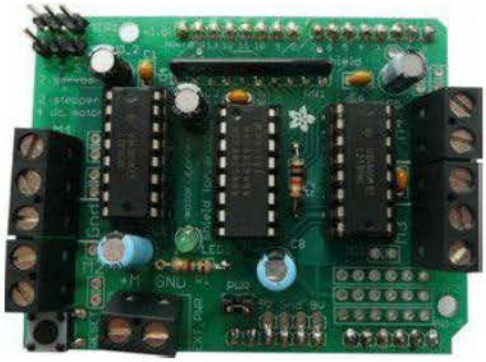
Il existe une bibliothèque prête à l'emploi, avec laquelle vous pouvez commander des moteurs pas-à-pas sans avoir à vous préoccuper de la programmation. Elle a pour nom `Stepper` et figure dans le pack de téléchargement Arduino. Vous trouverez toutes les informations nécessaires sur *http://www.arduino.cc/en/Reference/stepper*.

Shield pour moteur prêt à l'emploi

Vous pouvez acheter un shield pour moteur prêt à l'emploi, qui utilise deux circuits de commande de moteur L293DNE dont je vous ai parlé. La commande passe par le registre à décalage 74HC595 pour ne pas utiliser trop de broches numériques. Vous n'avez donc pas à

vous en faire car toute la logique se trouve dans la bibliothèque mise à disposition, que vous trouverez sur le site Internet correspondant.

Figure 14-7 ▶
Shield pour moteur



Vous pouvez raccorder des composants moteur les plus variés sur ce shield :

- 2 servomoteurs de loisir ;
- jusqu'à 4 moteurs à courant continu ;
- jusqu'à 2 moteurs pas-à-pas (unipolaire ou bipolaire).

Vous trouverez d'autres informations sur : *http://www.ladyada.net/make/mshield/*.

Problèmes courants

Si le moteur pas-à-pas ne bouge pas ou ne fait que bourdonner ou vibrer, vérifiez :

- que le câblage est correct ;
- qu'il n'y a pas de court-circuit éventuel ;
- que le moteur pas-à-pas ne change pas de position ou qu'il ne bourdonne pas ou vibre en début de sketch. Si tel est le cas, il y a de fortes chances que vous ayez interverti les quatre branchements ;
- que la connexion de masse commune est bien établie entre la carte Arduino et la source de tension externe ;
- que vous n'avez pas raccordé ensemble les deux pôles de tension d'alimentation de la carte et de la source de tension externe, qui sont marqués d'un +. Sinon, destruction de la carte Arduino assurée !

Qu'avez-vous appris ?

- Vous avez découvert comment commander un moteur pas-à-pas bipolaire.
- La commande a été réalisée au moyen du circuit de commande de moteur L293DNE.

Exercice complémentaire

La figure 14-8 montre une construction en Lego, sur laquelle j'ai monté un moteur pas-à-pas provenant d'un vieux scanner à plat.

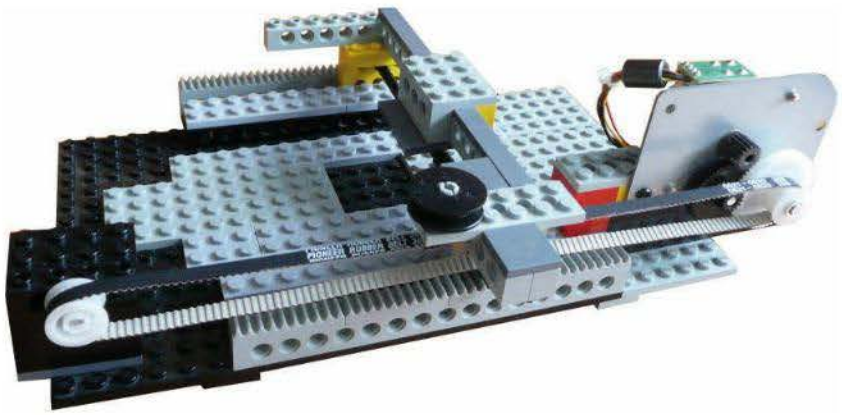


Figure 14-8
Moteur pas-à-pas sur une construction en Lego

La courroie dentée et la poulie de renvoi ont également été récupérées sur le scanner. Le chariot se déplace de gauche à droite et inversement sur des crémaillères sitôt le moteur entraîné. Avec un peu d'adresse et de créativité, vous pouvez ainsi vous construire un enregistreur X-Y.