LIVRET STAGIAIRE

PROGRAMMER ET DÉVELOPPER DES SYSTÈMES AVEC ARDUINO, PERFECTIONNEMENT

Date : 21 juillet 2022

Auteur(s) : Sébastien Charles

Table des matières

Introduction

L'objectif de cette ressource est de vous former à l'utilisation d'Arduino pour développer des systèmes interactifs qui intègrent des interfaces, des capteurs et des actionneurs. Le fonctionnement des effecteurs utilisés sera détaillé.

Déroulement

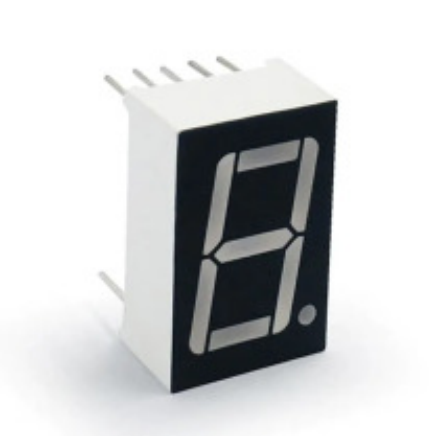
Durée : 12 heures

1. Contrôleur d'afficheur 7 segments

1.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de réaliser un afficheur à 7 segments.



Afficheur à LEDs 1 digit 7 segments

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un afficheur à 7 segments
- un résistor de 220 Ohms
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

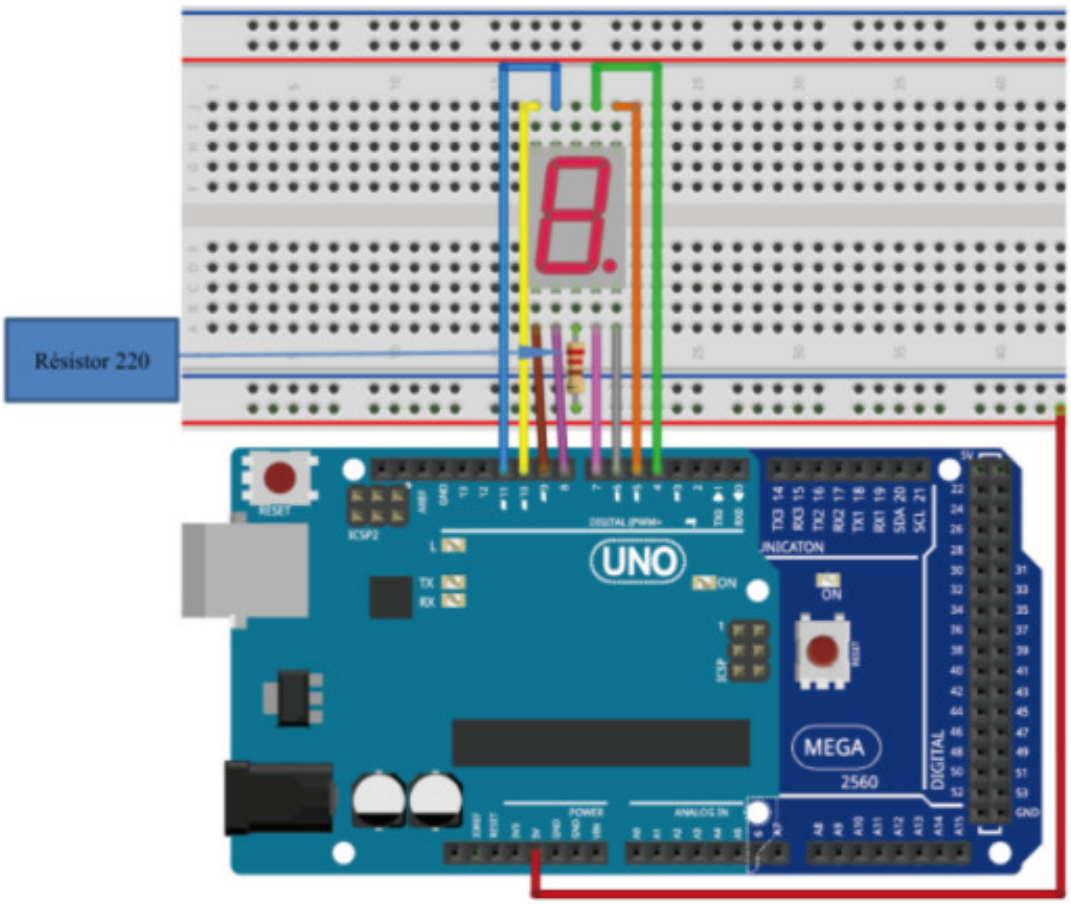
1.2. Afficheur à 7 segments

Le programme ci-dessous permet de contrôler un afficheur 7 segments pour lui faire afficher le chiffre 0 :

```
1 int a=4;
2 int b=5;
3 int c=7;
4 int d=8;
5 int e=9;
6 int f=11;
7 int g=10;
8 int dp=6;
```

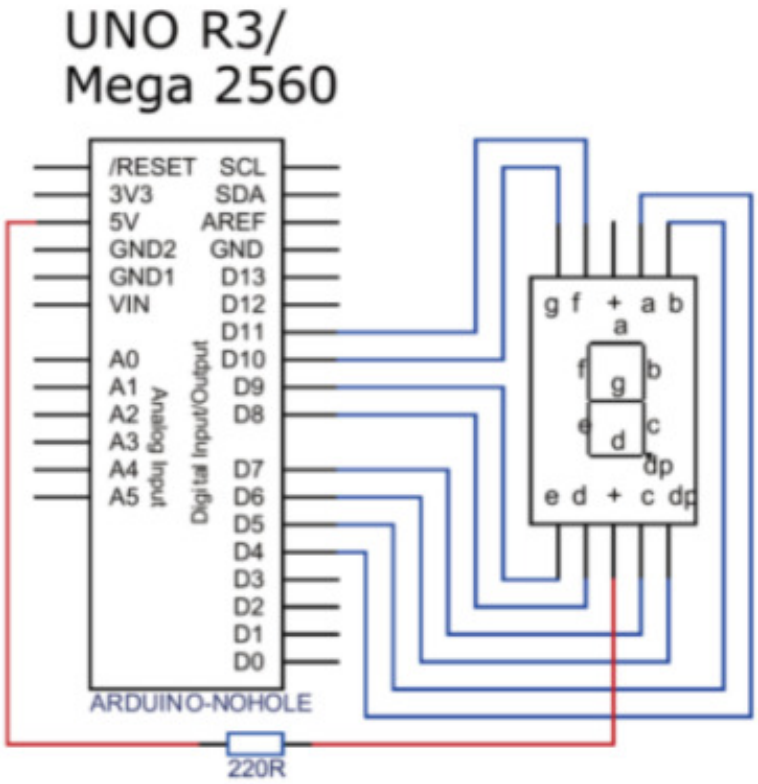
```
 9
10 void digital_0(void) // Fonction qui affiche ‘0’ sur l’afficheur 7 segments
11 {
12   digitalWrite(a,LOW);
13   digitalWrite(b,LOW);
14   digitalWrite(c,LOW);
15   digitalWrite(d,LOW);
16   digitalWrite(e,LOW);
17   digitalWrite(f,LOW);
18   digitalWrite(g, HIGH);
19   digitalWrite(dp,HIGH);
20 }
21
22 void setup()
23 {
24   int i;
25   for(i=4;i<=11;i++)
26   pinMode(i,OUTPUT); // Règle les connecteurs 4 à 11 en mode « sortie »
27 }
28
29 void loop()
30 {
31   digital_0(); //Segment display digital 0
32 }
```

Le montage à réaliser est le suivant :



Le schéma ci-dessous présente la correspondance entre les broches de la carte Arduino et celles de

l'afficheur à 7 segments.



Question n°1

Dans le programme ci-dessus, comment faut-il piloter les sorties de la carte Arduino pour allumer un segment ?

Question n°2

Réalisez le montage, téléversez le code et constatez son fonctionnement.

Question n°3

Modifiez le programme pour lui faire afficher des chiffres de 1 à 3 puis de 3 à 1 en opérant une pause d'une seconde entre 2 chiffres puis constatez le fonctionnement.

2. Utilisation d'un registre pour contrôler un afficheur 4 digits à 7 segments

2.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de piloter un afficheur à 7 segments à l'aide d'un registre à décalage.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un afficheurs à 4 digits de 7 segments SMA420564L
- un registre à décalage de 8 bits 74HC595
- 4 résistors de 220 Ohms
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

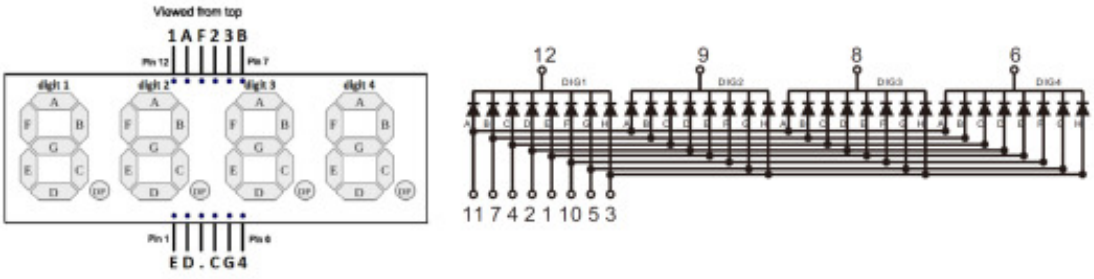
2.2. Pilotage d'un afficheur 7 segments à l'aide d'un registre à décalage.

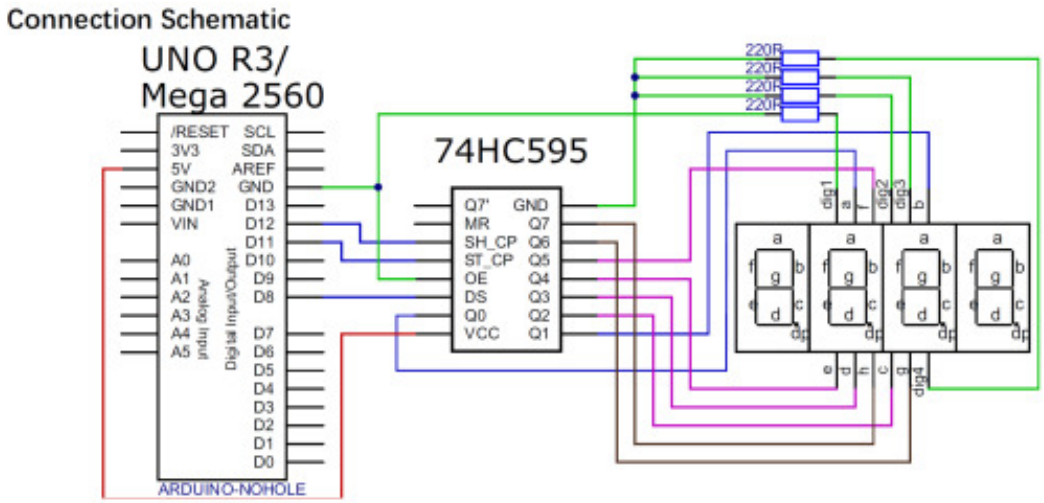
Le programme ci-dessous permet d'afficher les caractères 1 à 9 puis de `a` à `f` sur chaque afficheur en changeant d'afficheur toutes les 0.5 s :

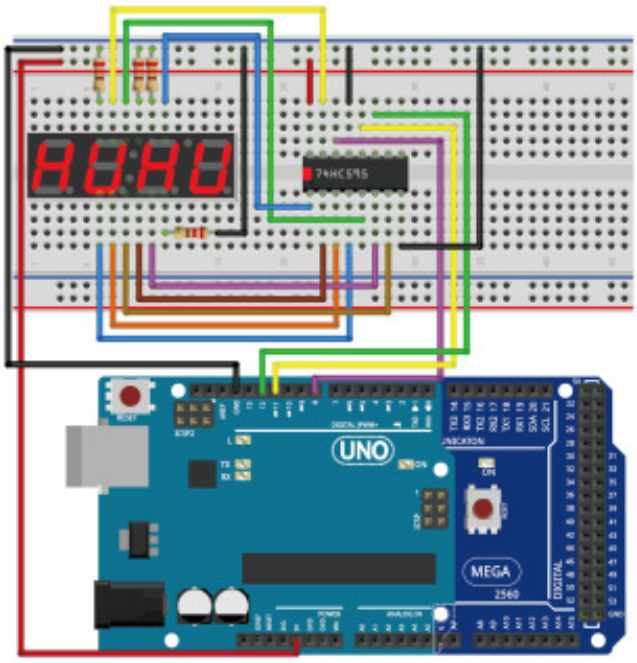
```
1  // broche ST_CP du registre, quand elle passe à 1 (5v) : copie les valeurs
   présentes dans le registre vers les sorties du registres de Q0 à Q7
2  int latch=11;
3  //broche SH_CP du registre, quand elle passe à 1 (5v) : décalle toutes les valeurs
   des bits vers la droite et copie la valeur de DS dans le bit de gauche du regsitre
4  int clock=12;
5  //broche DS du registre, accueille la valeur du bit : 0 (0V) ou 1 (5V) qui sera
   recopiée lors de la prochaine activation de la broche SH_CP
6  int data=8;
7
8  // tableau qui contient des octets qui seront traités par le registre à décalage,
   "unsigned char" peut être remplacé par "byte"
9  unsigned char table[]= {0b11111111,0b01100000,0b11011010,0b11110010};
10
11 void setup()
12 {
13   pinMode(latch,OUTPUT);
```

```
14    pinMode(clock,OUTPUT);
15    pinMode(data,OUTPUT);
16  }
17
18  void Display(unsigned char num)
19  {
20    digitalWrite(latch,LOW);
21    // la fonction shiftOut écrit vers la broche data (=8), les bits d'une variable
      un par un et active la broche clock (=12) à chaque envoi de bit
22    // MSBFIRST or LSBFIRST : (Most Significant Bit First = the leftmost or Least
      Significant Bit First = rightmost)
23    // ici c'est LSBFIRST qui est choisi, soit le bit le plus à droite en premier
24    // l'octet lu bit par bit est celui du tableau table[num]
25    shiftOut(data,clock,LSBFIRST,table[num]);
26    digitalWrite(latch,HIGH);
27  }
28
29  void loop()
30  {
31    Display(0); //active tous les segments des 4 digits
32    delay(1000);
33    Display(1); //active 2 segments pour afficher le chiffre 1 sur chaque digit
34    delay(1000);
35    Display(2); //active 5 segments pour afficher un chiffre 2 sur chaque digit
36    delay(1000);
37    Display(3); //active 5 segments pour afficher un chiffre 3 sur chaque digit
38    delay(1000);
39  }
```

Le montage à réaliser est le suivant :







Question n°1

Qu'est ce qu'un registre à décalage de 8 bits ?

Question n°2

En quoi est-il intéressant d'utiliser un registre de ce type dans ce montage ?

Question n°3

Réalisez le montage, téléversez le code et constatez le résultat.

Question n°4

Ecrire un programme qui affiche 9999 sur les 4 digits puis constatez le fonctionnement.

2.3. Pilotage d'un afficheur 7 segments à l'aide d'un registre à décalage, bibliothèque SevSeg.h.

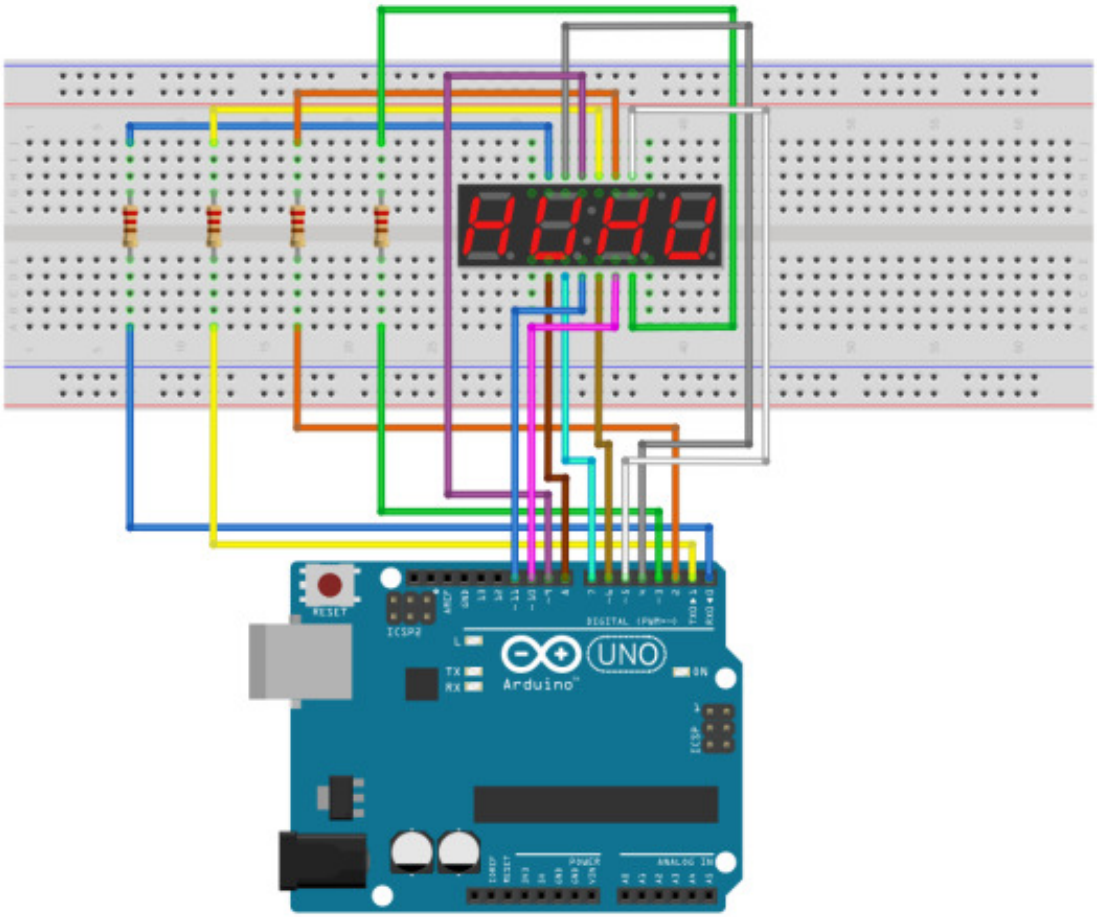
Il est possible de piloter plus facilement les 4 digits d'un afficheur par le biais de la bibliothèque SevSeg.h, le programme est le suivant :

```
#include "SevSeg.h"

SevSeg sevseg;

void setup() {
  // Pour le montage il faut connecter les broches 0 à 11 de l'Arduino sur les
  // broches 1,2,3,4,A,B,C,D,E,F,G,DP de l'afficheur
  // mettez une résistance 220Ohm entre les broches de l'Arduino et de l'afficheur
0 et 1, 1 et 2, 2 et 3, 3 et 4
  sevseg.Begin(0,0,1,2,3,4,5,6,7,8,9,10,11); // Le premier 0 indique qu'il s'agit
d'une cathode commune
}
```

```
11
12 void loop() {
13   // les 4 chiffres de la fonction NewNum seront affichés,
14   // le dernier chiffre après la virgule donne la position du "." de 0 à 3, un
   chiffre supérieur à 3 indique qu'on ne veut pas de virgule
15   sevseg.NewNum(1234,0);
16   sevseg.PrintOutput();
17 }
```

La bibliothèque est disponible ici : Bibliothèque SevSeg[https://docs.google.com/file/d/0Bwrp4uluZCpNdE9oWTY0M3BncTA/edit?resourcekey=0-94SXXrPEy4vslERsFfiFzQ], téléchargez le fichier Zip puis dans l'IDE Arduino cliquez sur `Croquis/Inclure une bibliothèque/Ajouter la bibliothèque .ZIP ...`

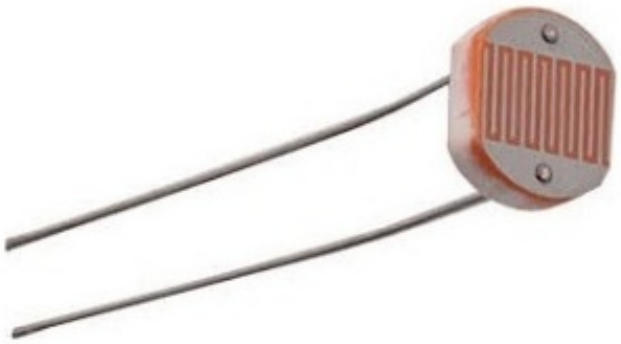
Le montage est le suivant :



3. Utilisation d'un photoresistor

3.1. Introduction

L'objectif est d'apprendre à contrôler un bandeau de LED qui indique la luminosité mesurée à l'aide d'un photo-résistor en utilisant un registre à décalage.



Photoresistor

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un photo-résistor
- un registre à décalage de 8 bits 74HC595
- 8 LED rouges
- 8 résistors de 220 Ohms
- 1 résistor de 10kOhm
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

3.2. Utilisation d'un photoresistor

Le programme ci-dessous permet de contrôler des LED avec un photoresistor :

```
1 // Broche reliée au photo-résistor
2 int lightPin = 0;
3 // Broche reliée au ST_CP du 74HC595
4 int latchPin = 11;
5 // Broche reliée au SH_CP du 74HC595
```

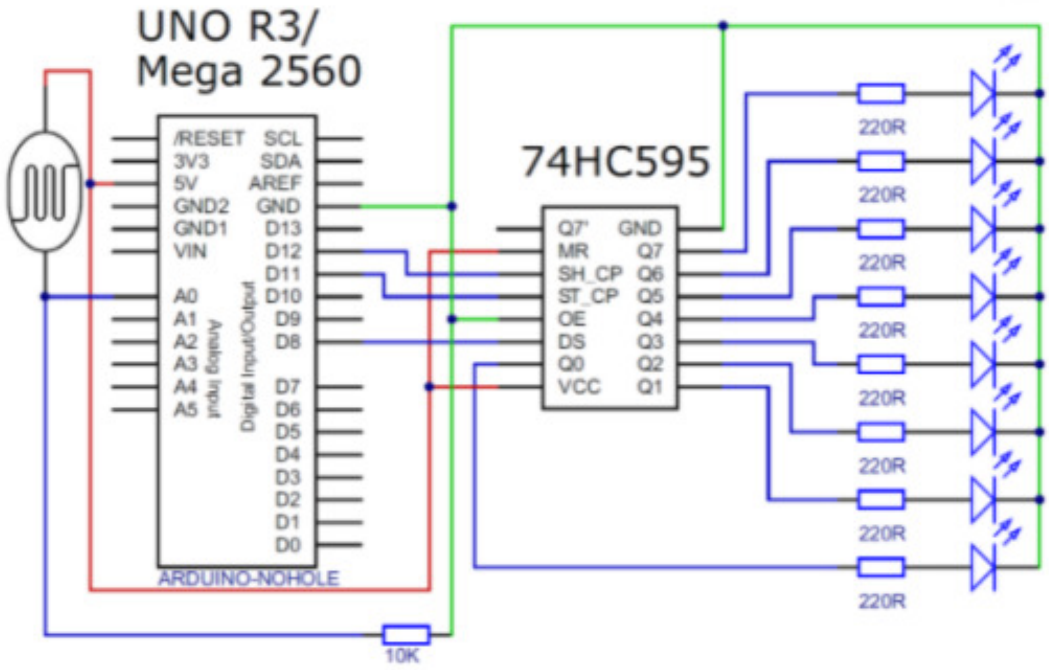
```
int clockPin = 12;
// Broche reliée au DS du 74HC595
int dataPin = 8;

int leds = 0;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(lightPin, INPUT);
}

void updateShiftRegister()
{
   digitalWrite(latchPin, LOW);
   shiftOut(dataPin, clockPin, LSBFIRST, leds);
   digitalWrite(latchPin, HIGH);
}

void loop()
{
  int reading  = analogRead(lightPin);
  int numLEDSLit = reading / 57;  // pour calibrer la sensibilité
  if (numLEDSLit > 8) numLEDSLit = 8;
  leds = 0;
  for (int i = 0; i < numLEDSLit; i++)
  {
    leds = leds + (1 << i);
  }
  updateShiftRegister();
  delay (500);
}
```

Photorésistance

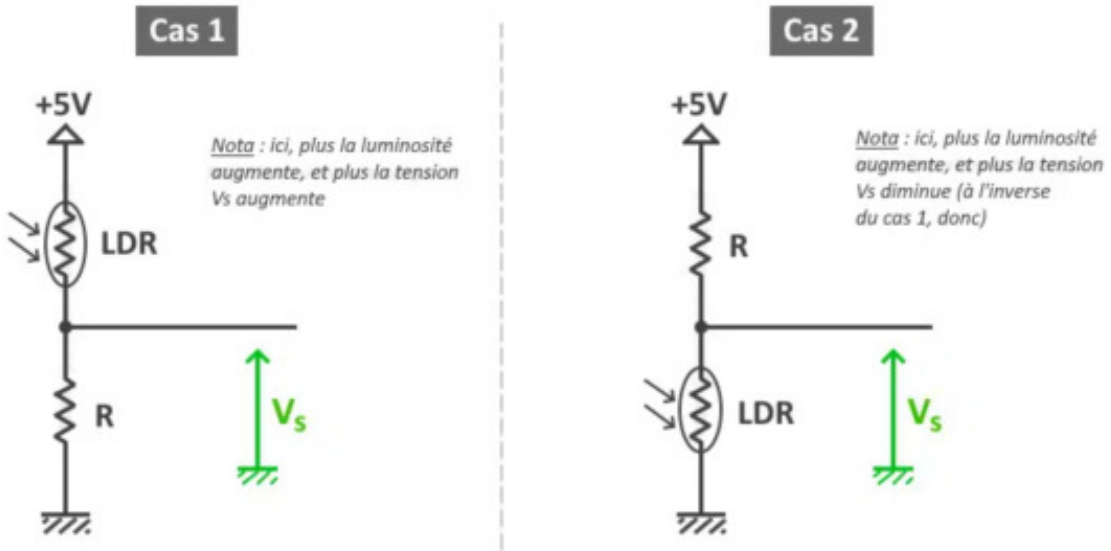
≃ *résistance photogénique, cellule photoconductrice ou cellule photoéléctrique*

Une photorésistance est un composant électronique dont la résistance varie en fonction de la quantité de lumière incidente : plus elle est éclairée, plus sa résistance baisse et réciproquement.

Le montage à réaliser est le suivant :



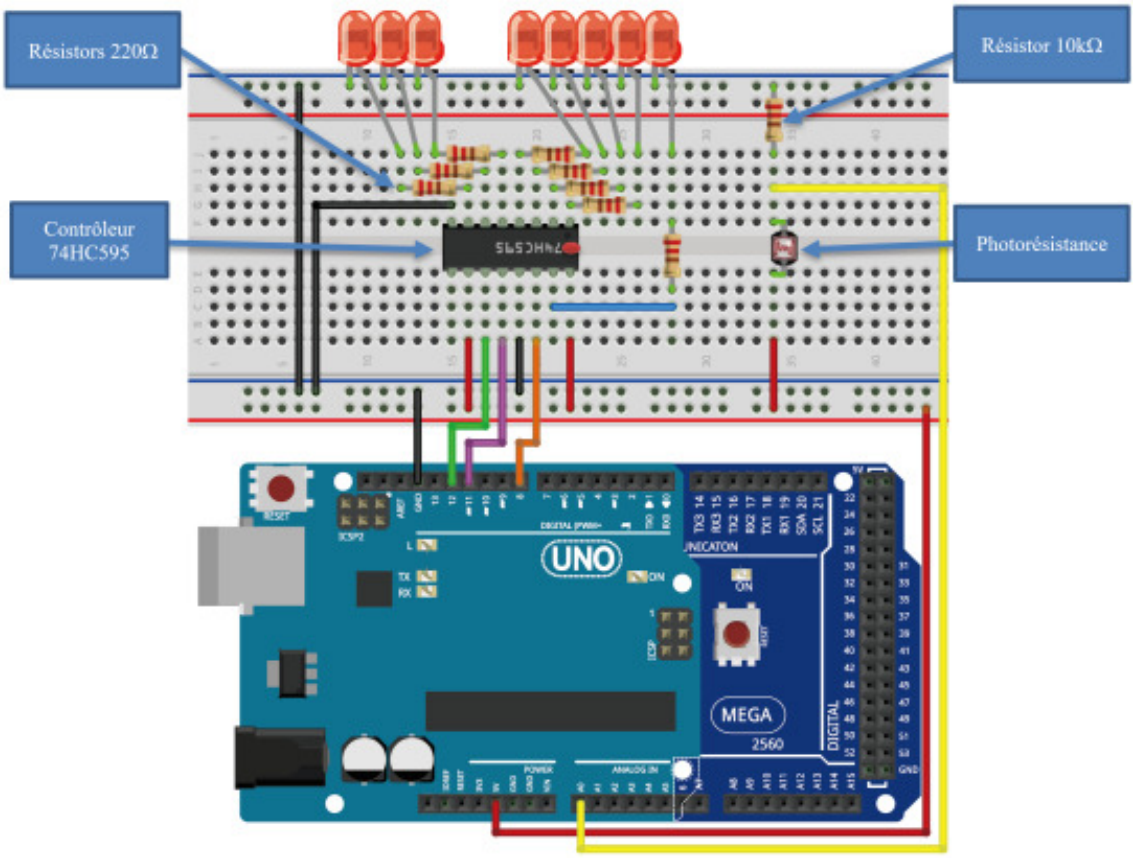
Les photo-résistors sont généralement implantés dans les circuits au moyen d'un pont diviseur de tension selon un de ces deux schémas :



Dans le montage nous avons choisi le cas 1 avec une résistance R de 10kOhm et une tension Vs mesurée sur l'entrée analogique A0. Dans ce cas, plus la luminosité augmente plus la tension mesurée sur l'entrée analogique se rapproche de 5V.

Pour plus d'information : Photo-résistor*[https://passionelectronique.fr/photoresistance/]*

Dans ce montage, nous utilisons un registre à décalage pour limiter l'utilisation des broches de la carte Arduino.



Question n°1

Dans le programme, expliquez ce qu'apporte la boucle FOR.

Question n°2

Réalisez le montage, téléversez le code et constatez le fonctionnement.

Question n°3

Modifiez le programme et le montage de façon à afficher l'intensité lumineuse sur un afficheur à 7 segments (0 : aucune luminosité, 8 : luminosité maximale) puis constatez son fonctionnement.

4. Mesure de distance par ultrason

4.1. Introduction

L'objectif du montage est de mesurer une distance à l'aide d'un capteur à ultrason et de l'afficher sur un écran LCD.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un capteur à ultrason HC-SR04
- un potentiomètre
- un écran LCD
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

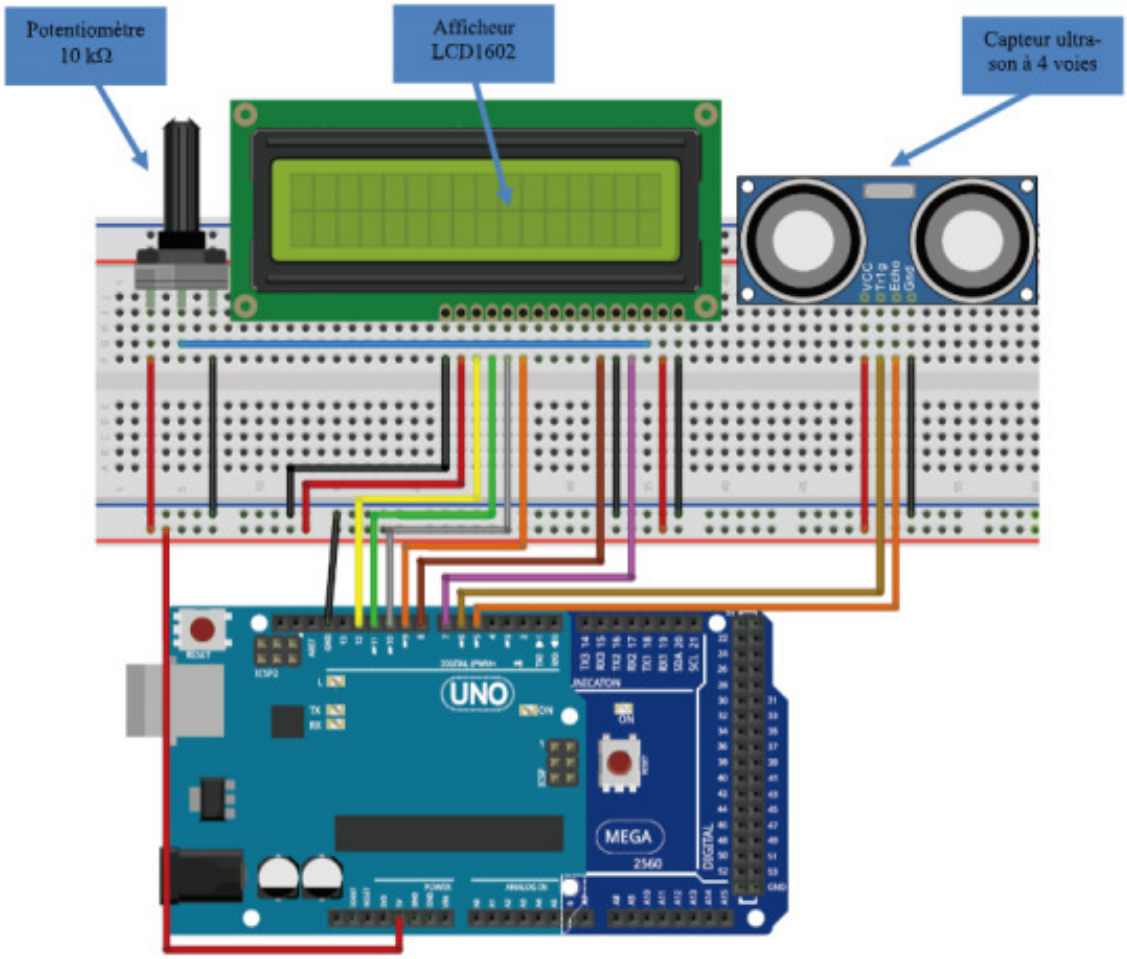
4.2. Mesure de distance par ultrason

Le programme ci-dessous permet d'afficher la distance séparant le capteur à ultrason de sa cible :

```
#include <LiquidCrystal.h>
const int echoPin = 5;
const int trigPin  = 6;

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
/*
 * LCD RS au connecteur 7
 * LCD Enable au connecteur 8
 * LCD D4 au connecteur 9
 * LCD D5 au connecteur 10
 * LCD D6 au connecteur 11
 * LCD D7 au connecteur 12
*/

void setup()
{
   pinMode(echoPin, INPUT);
   pinMode(trigPin, OUTPUT);
   lcd.begin(16, 2);  // Règle le nombre de colonnes et de lignes du LCD
   lcd.clear(); // Efface l'écran du LCD et positionne le curseur en haut à gauche
   delay(1000);
}

```

```
24 void loop()
25 {
26    int cm = ping(echoPin) ;
27    lcd.setCursor(0, 0); // Place le curseur à la colonne 0 et ligne 0
28    lcd.print("Distance: "); // Affiche le Texte « Distance » sur le LCD
29    lcd.print(cm); // Affiche la distance mesurée par le capteur à ultra son
30    lcd.print(" cm    ");
31    //les espaces après " cm____" servent à effacer une éventuelle mesure résiduelle
32    // précedente, ceci permet d'éviter d'avoir à effacer l'écran entre deux
affichages
33    delay(500);
34 }
35
36 int ping(int echoPin)
37 {
38    // Cette fonction mesure la distance à la cible à partir du temps de réponse
mesuré
39    long duration, cm;
40    // Un ping sonore est généré quand une pulsation HIGH est générée pendant 2
microsecondes ou plus.
41    // Une courte pulsation LOW est envoyée avant la commande du trigger pour
nettoyer le signal
42    pinMode(trigPin, OUTPUT);
43    digitalWrite(trigPin, LOW);
44    delayMicroseconds(2);
45    digitalWrite(trigPin, HIGH);
46    delayMicroseconds(5);
47    digitalWrite(trigPin, LOW);
48
49    pinMode (echoPin, INPUT);
50    // la fonction pulseIn (soit HIGH ou LOW) lit une impulsion sur une broche
51    // par exEmple, si la valeur est HIGH, pulseIn() attends que la broche passe de
LOW à HIGH,
52    // lance un timer, attend que la broche revienne à LOW et arrête le timer,
53    // la fonction renvoie ensuite le temps écoulé en microsecondes ou retourne 0 si
aucune impulsion n'est mesurée avant le timeout.
54    duration = pulseIn(echoPin, HIGH);
55
56    // convertit le temps en distance en cm
57    cm = microsecondsToCentimeters(duration);
58    return cm ;
59 }
60
61 long microsecondsToCentimeters(long microseconds)
62 {
63    // La vitesse du son est de 340 m/s ou 29 microsecondes par centimètre.
64    // Le signal du ping part et revient suite à l'impact sur l'objet pendant la
mesure ce qui fait qu'il
65    // faut diviser la distance mesurée par 2.
66    return microseconds / 29 / 2;
67 }
```

Le montage à réaliser est le suivant :



Question n°1

Comment fonctionne le capteur à ultrason ?

Question n°2

Comment fonctionne un écran LCD ?

Question n°3

A quoi sert le potentiomètre sur le montage ?

Question n°4

Réalisez le montage, téléversez le code et constatez le fonctionnement.

Question n°5

Modifiez le programme pour afficher la distance sur la première ligne et le temps de réponse sur la deuxième.

5. Mesure de distance par télémètre laser

5.1. Introduction

Objectifs pédagogiques

L'objectif du montage est de mesurer une distance à l'aide d'un télémètre laser et de l'afficher sur un écran LCD.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un télémètre laser VL53L0X QT
- un potentiomètre
- un écran LCD
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

5.2. Mesure de distance par télémètre laser

Le programme ci-dessous permet d'afficher la distance séparant le télémètre laser de sa cible :

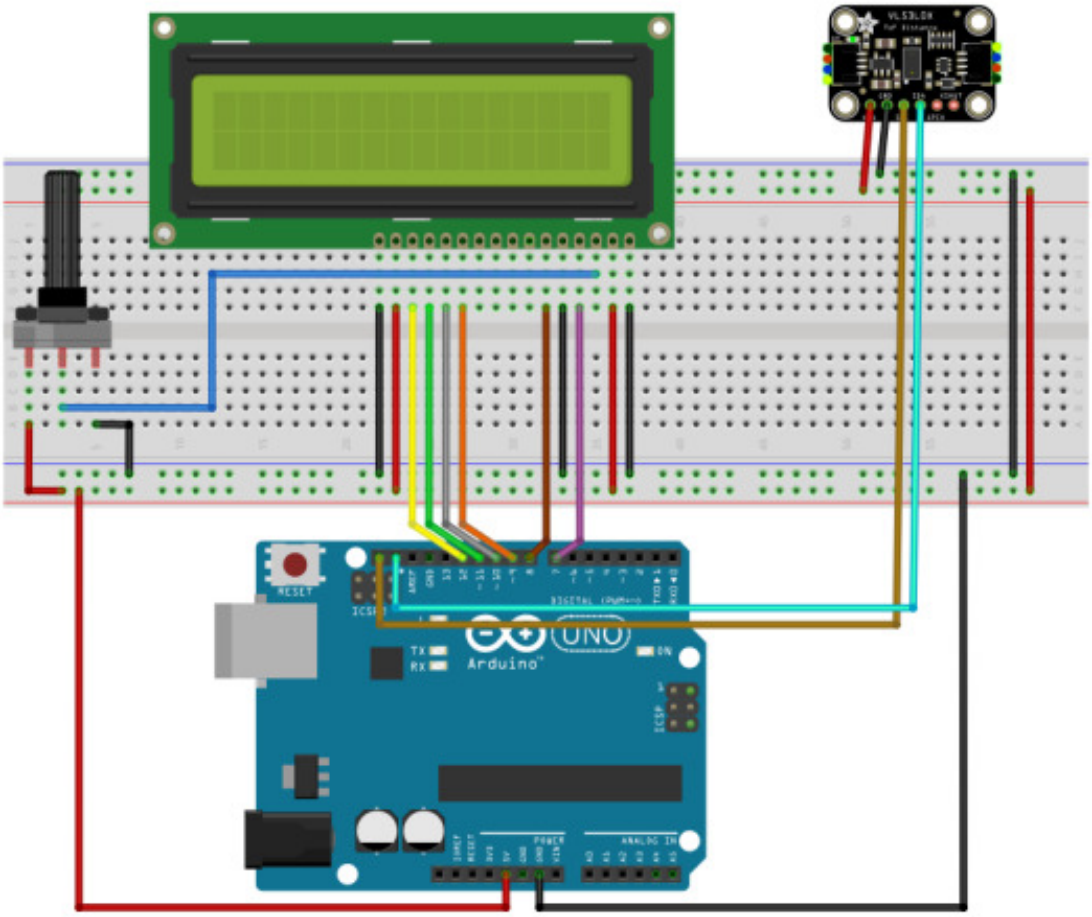
```
#include <LiquidCrystal.h>
#include "Adafruit_VL53L0X.h"

LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
/*
 * LCD RS au connecteur 7
 * LCD Enable au connecteur 8
 * LCD D4 au connecteur 9
 * LCD D5 au connecteur 10
 * LCD D6 au connecteur 11
 * LCD D7 au connecteur 12
*/

Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  Serial.begin(115200);
  // Pour attendre que le port soit prêt pour communiquer en série avec le capteur
  while (! Serial) {
    delay(1);
```

```
  }
  lcd.begin(16, 2);
  lcd.clear();
  delay(1000);
  lcd.setCursor(0, 0);
  lcd.print("Adafruit VL53L0X test");
  // Vérification du fonctionnement du capteur
  if (!lox.begin()) {
    lcd.print("Failed to boot VL53L0X");
    while(1);
  }
  lcd.setCursor(0, 1);
  lcd.print("VL53L0X Ready");
  delay(1000);
  lcd.clear();
}

void loop() {
  VL53L0X_RangingMeasurementData_t measure;
  lcd.setCursor(0, 0);
  // Mesure et vérification qu'elle n'est pas en dehors de la zone de mesure
  lox.rangingTest(&measure, false);
  if (measure.RangeStatus != 4) {
    lcd.print("Dist : "); lcd.print(measure.RangeMilliMeter); lcd.print(" mm   ");
  } else {
    lcd.print("Out of range !    ");
  }
  delay(100);
}
```

La bibliothèque Adafruit_VL53L0X est téléchargeable sur le site : Adafruit_VL53L0X*[https://github.com/adafruit/Adafruit_VL53L0X]* en cliquant sur `Code/Download ZIP`.

Ensuite, pour intégrer cette bibliothèque dans l'IDE Arduino, il faut aller dans : `Croquis/Inclure une bibliothèque/Ajouter la bibliothèque .ZIP ...`

Le montage à réaliser est le suivant :



Question n°1

Comment fonctionne ce télémètre laser ?

Question n°2

Réalisez le montage, téléversez le code et constatez le fonctionnement.

Question n°3

Modifiez le programme pour que la distance soit affichée dans le moniteur série.

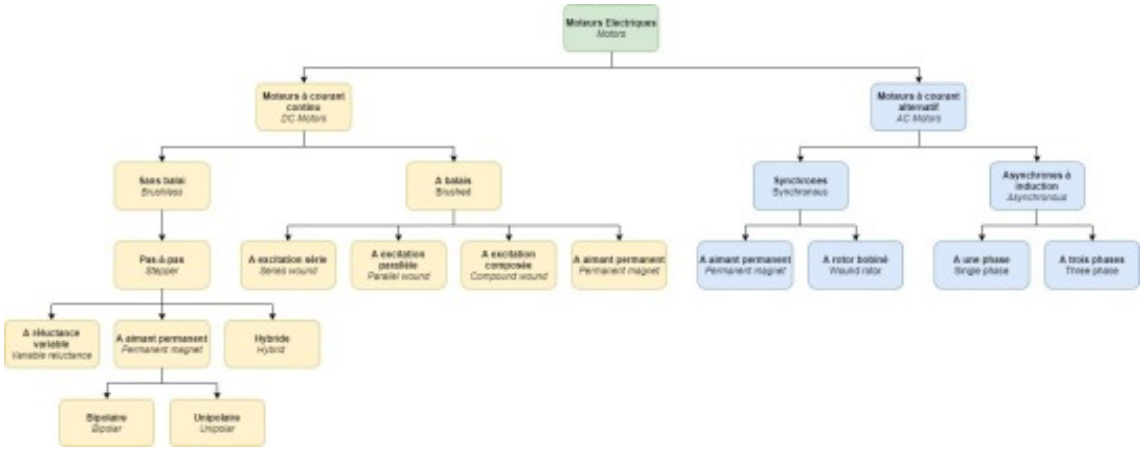
6. Les moteurs électriques

6.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de présenter de manière synthétique les principales familles de moteurs et leurs spécificités.

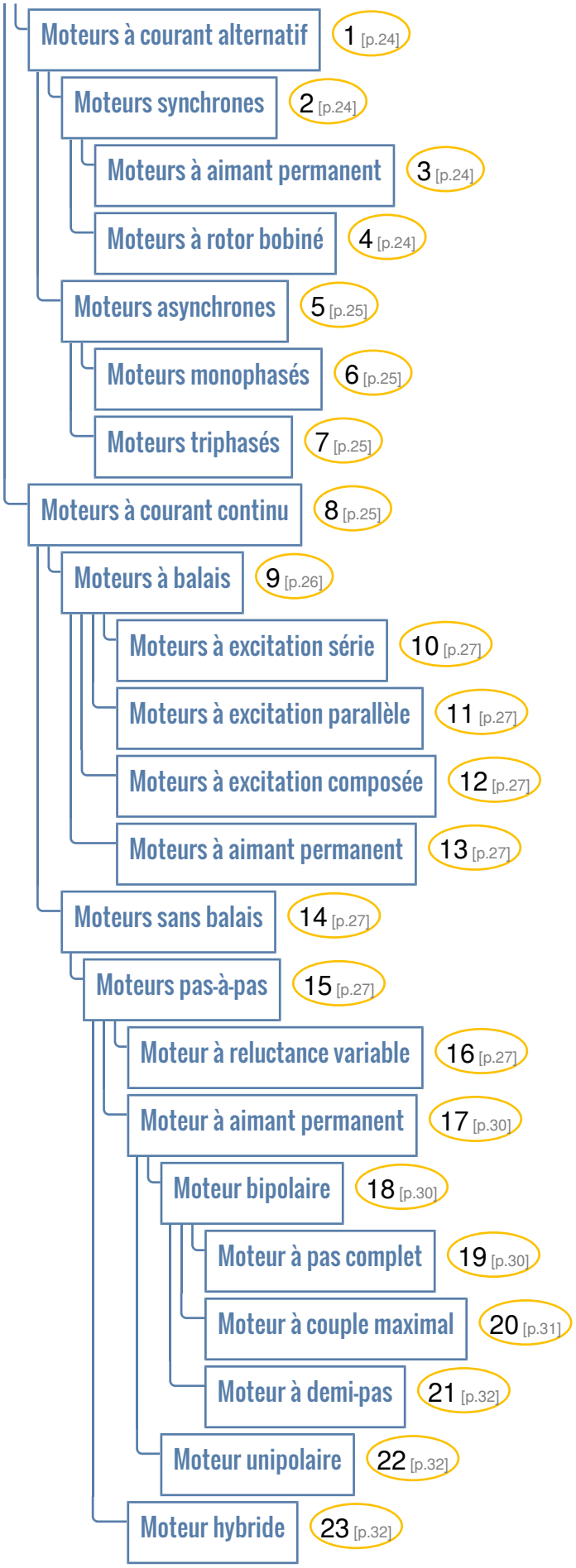
Les moteurs électriques se divisent en 2 grandes familles, les moteurs à courant continu et les moteurs à courant alternatifs.



Les familles de moteurs électriques

6.2. Classification des moteurs électriques

Les moteurs électriques


Moteurs à courant alternatif 1 [p.24]
Moteurs synchrones 2 [p.24]
Moteurs à aimant permanent 3 [p.24]
Moteurs à rotor bobiné 4 [p.24]
Moteurs asynchrones 5 [p.25]
Moteurs monophasés 6 [p.25]
Moteurs triphasés 7 [p.25]
Moteurs à courant continu 8 [p.25]
Moteurs à balais 9 [p.26]
Moteurs à excitation série 10 [p.27]
Moteurs à excitation parallèle 11 [p.27]
Moteurs à excitation composée 12 [p.27]
Moteurs à aimant permanent 13 [p.27]
Moteurs sans balais 14 [p.27]
Moteurs pas-à-pas 15 [p.27]
Moteur à reluctance variable 16 [p.27]
Moteur à aimant permanent 17 [p.30]
Moteur bipolaire 18 [p.30]
Moteur à pas complet 19 [p.30]
Moteur à couple maximal 20 [p.31]
Moteur à demi-pas 21 [p.32]
Moteur unipolaire 22 [p.32]
Moteur hybride 23 [p.32]

1 Moteurs à courant alternatif

Ces moteurs sont les plus polyvalents et trouvent de nombreux usages, notamment pour les fortes charges.

Par rapport aux moteurs à courant continu, ils présentent quelques avantages :

- Meilleur rapport poids/encombrement/puissance
- Faible consommation au démarrage
- Accélération, courant de démarrage, vitesse de fonctionnement, limite de couple contrôlables
- Perturbations réduites de la ligne de puissance

2 Moteurs synchrones

La rotation du rotor de ces moteurs est synchronisée avec la fréquence du courant d'alimentation, et la vitesse reste constante en cas de variation de la charge. Ce type de moteur est donc idéal pour les systèmes qui nécessitent une vitesse constante.

Applications habituelles : appareils de positionnement de grande précision comme les robots, l'instrumentation, les machines spéciales et le contrôle de processus

Pour plus d'informations

Les moteurs synchrones*[https://energieplus-lesite.be/techniques/ascenseurs7/moteur-synchrone/]*

3 Moteurs à aimant permanent

Dans cette configuration, le stator est bobiné et le rotor est à aimants permanents.

Ce sont des moteurs qui peuvent accepter des courants de surcharge importants pour démarrer rapidement.

Applications habituelles : motorisation d'ascenseurs compactes ayant une accélération rapide (immeuble de grande hauteur par exemple).

4 Moteurs à rotor bobiné

Dans cette configuration, le stator et le rotor sont bobinés et combinés à un système de balais et de collecteurs.

Ce type de machines est réversible car elles peuvent fonctionner en régime moteur comme en régime alternateur.

Applications habituelles : pour des machines performantes en moyennes et fortes puissances.

5 Moteurs asynchrones

Ces moteurs utilisent un champs magnétique tournant qui induit un courant dans des conducteurs en court circuit situés sur le rotor. La combinaison du champs magnétique tournant et du courant passant dans les conducteurs du rotor produit une force électromagnétique qui se traduit mécaniquement par un couple et une vitesse de rotation. Ce type de moteur est intéressant pour sa capacité de charge et pour son couple relativement constant quelque soit sa vitesse.

Applications habituelles : systèmes industriels telles que les compresseurs, pompes, systèmes de convoyeurs et le matériel de levage.

Pour plus d'informations

Les moteurs asynchrones*[https://energieplus-lesite.be/techniques/ascenseurs7/moteur-asynchrone/#:~:text=Dans%20un%20moteur%20asynchrone%2C%20c,%C3%A0%20rattraper%20le%20champ%20tournant.]*

6 Moteurs monophasés

Ils sont utilisés pour les faibles charges.

7 Moteurs triphasés

Ils sont utilisés pour les fortes charges.

8 Moteurs à courant continu

Ces moteurs sont très utilisés pour des applications à faibles charges.

Par rapport aux moteurs alternatifs, ils présentent quelques avantages :

- Installation facile
- Commande de vitesse simple
- Démarrage, arrêt, marche arrière et accélération rapides
- Couple de démarrage élevé
- Courbe couple-vitesse linéaire

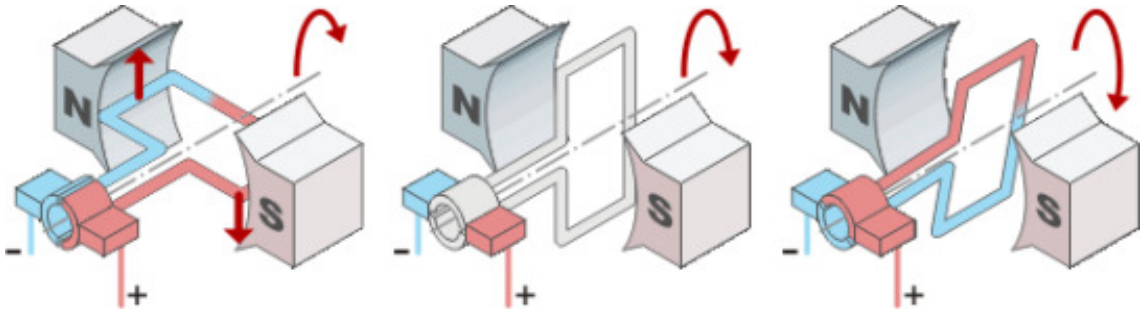
Pour plus d'informations

Les moteurs à courant continu*[https://energieplus-lesite.be/techniques/ascenseurs7/moteur-a-courant-continu/]*

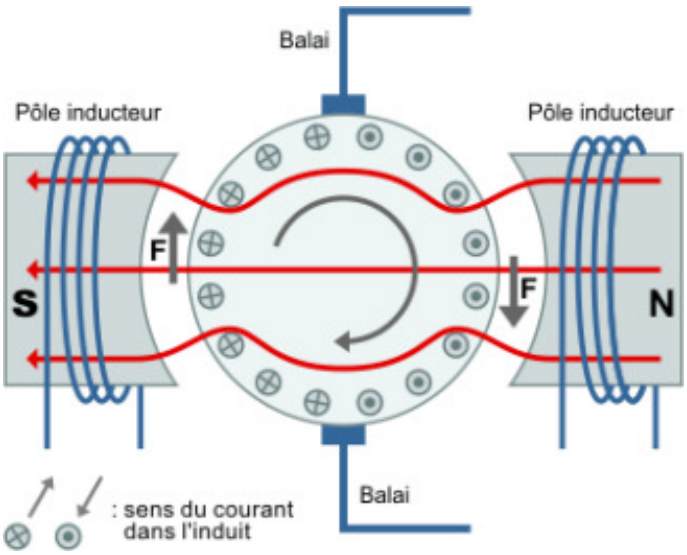
9 Moteurs à balais

Famille de moteur à courant continu la plus répandue car offrant l'avantage d'être peu coûteuse à l'achat et simple à commander puisqu'il suffit de faire varier leur tension d'alimentation pour contrôler leur vitesse de rotation.

Ces moteurs utilisent un stator inducteur (bobiné ou à aimant permanent) et un rotor induit et des collecteurs et des balais qui servent à inverser alternativement le sens de circulation du courant dans le rotor afin de le faire tourner en fonction de son orientation par rapport aux pôles du stator.



Si le système balais-collecteurs n'était pas présent (simple spire alimentée en courant continu), la spire s'arrêterait de tourner en position verticale sur un axe appelé communément "ligne neutre". Le système balais-collecteurs a pour rôle de faire commuter le sens du courant dans les deux conducteurs au passage de la ligne neutre. Le courant étant inversé, les forces motrices sur les conducteurs le sont aussi permettant ainsi de poursuivre la rotation de la spire.



Dans la pratique, la spire est remplacée par un induit (rotor) de conception très complexe sur lequel sont montés des enroulements (composés d'un grand nombre de spires) raccordés à un collecteur "calé" en bout d'arbre. Dans cette configuration, l'induit peut être considéré comme un seul et même enroulement semblable à une spire unique.

Pour plus d'informations

Les moteurs à courant continu *[https://energieplus-lesite.be/techniques/ascenseurs7/moteur-a-courant-continu/]*

10 Moteurs à excitation série

Dans cette configuration, le bobinage du stator est connecté en série au bobinage du rotor. La contrôle de la vitesse est réalisé en faisant varier la tension d'alimentation. Ce type de moteur offre un contrôle peu efficace de la vitesse puisque lorsque le couple résistant augmente, la vitesse du moteur chute.

Applications habituelles : systèmes exigeant un couple de démarrage élevé comme les automobiles, les palans, ascenseurs et grues.

11 Moteurs à excitation parallèle

Dans cette configuration le bobinage du stator est connecté en parallèle au bobinage du rotor afin de fournir un couple plus élevé, sans réduction de vitesse lors d'une augmentation du courant délivré au moteur. Son couple de démarrage est moyen avec une vitesse constante.

Applications habituelles : tours d'usinage, aspirateurs, convoyeurs et meuleuses.

12 Moteurs à excitation composée

Dans cette configuration, le bobinage du stator est connecté en parallèle et en série au bobinage du rotor. Ainsi la polarité du bobinage parallèle s'ajoute aux champs en série. Ce type de moteur possède un couple de démarrage élevé et offre un large variation de vitesse.

Applications habituelles : compresseurs, pompes centrifuges à tête variable, presses rotatives, scies circulaires, machines de cisaillement, ascenseurs et carrousels à bagages.

13 Moteurs à aimant permanent

Dans cette configuration, l'électroaimant du stator est remplacé par un aimant permanent. Ces moteurs peuvent être commandés précisément mais le couple fourni est faible.

Applications habituelles : robotique et servo-systèmes.

14 Moteurs sans balais

Les moteurs sans balais réduisent les problèmes d'usure liés à l'utilisation de balais et leur conception mécanique est beaucoup plus simple. Les contrôleurs de ce type de moteur utilisent des capteurs à effet Hall pour détecter la position du rotor et ainsi mieux réguler sa vitesse. Les avantages de cette technologie sont une longue durée de vie, peu d'entretien et un haut rendement (85 à 90%), tandis que les inconvénients sont des coûts élevés liés à des contrôleurs plus compliqués.

Applications habituelles : contrôle de positionnement et de vitesse avec des applications telles que les ventilateurs, propulsion de drone, pompes et compresseurs, qui nécessitent fiabilité et robustesse.

15 Moteurs pas-à-pas

Ces moteurs sont principalement utilisés dans le contrôle de position et se prêtent bien à l'asservissement en boucle ouverte (pas d'adaptation de la commande à la position réelle du système). Généralement lents, leur vitesse maximale est de l'ordre de 3000 tr/min. Leur durée de vie est grande.

Applications habituelles : imprimantes 3D, équipements de positionnement.

16 Moteur à reluctance variable

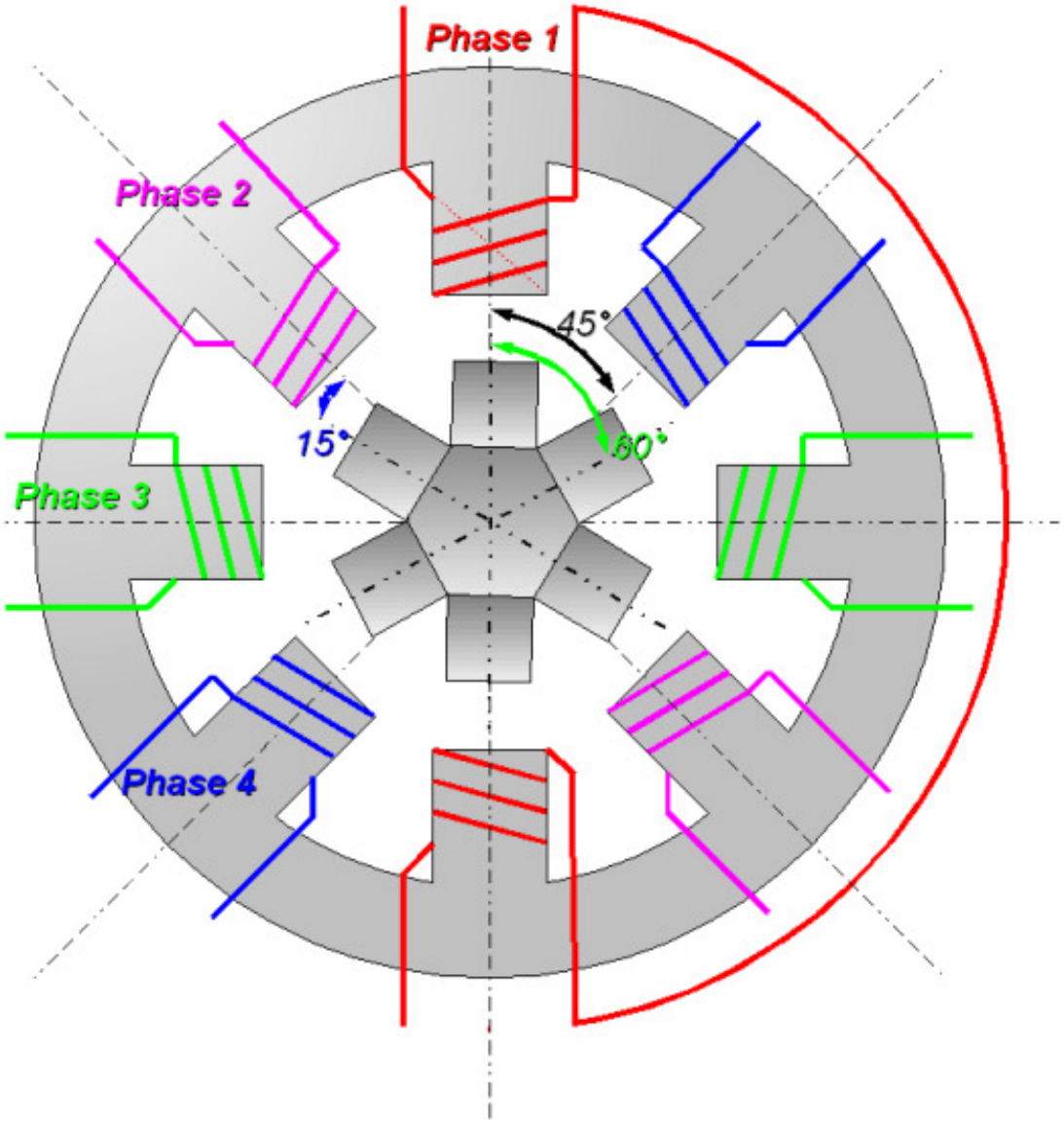
Les moteurs à réluctance variable (autrement dit à résistance magnétique variable) doivent leur nom au fait que le circuit magnétique qui les compose s'oppose de façon variable à sa pénétration par un champ magnétique.

Ces moteurs sont composés d'un barreau de fer doux et d'un certain nombre de bobines. Lorsqu'on alimente une bobine, le champ magnétique cherche à minimiser le passage dans l'air. Ainsi l'entrefer entre la bobine et le barreau se réduit. Le barreau s'aligne avec le champ magnétique pour obtenir une réluctance minimale. Pour faire tourner le moteur on alimente successivement les phases qui se suivent, dans un sens ou dans l'autre. Une phase alimente généralement deux bobines diamétralement opposées afin de générer un couple sur le barreau de ferrite.

Dans la pratique, le barreau de ferrite a plusieurs dents.

Pour connaitre le nombre de pas par tour de ce type de moteur, il faut compte le nombre de bobine X, le nombre de dents de la ferrite Y. L'angle de rotation entre deux alimentations successives de phase est alors (360/Y) - (360/X) = Z si Y < X. Le nombre de pas est alors 360/Z.

Par exemple dans le cas d'un moteur avec 8 bobines (soit 4 phases), 6 dents pour la ferrite, le moteur tourne de 60°-45°=15° par changement de phase, soit 360/15=24 pas par tour.



Moteur à reluctance variable à 4 phases et ferrite à 6 dents

Les inconvénients de ce moteur est qu'il nécessite au moins trois bobinages, qu'il n'a pas de couple résiduel (hors tension, le rotor est libre) et que sa fabrication est difficile puisque les entrefers doivent être très faibles. Ses avantages sont qu'il est peu coûteux, d'une bonne précision (par exemple, avec 4 bobinages on obtient 24 pas, il peut monter jusqu'à 360 pas) et que le sens du courant dans la bobine n'a aucune importance.

17 Moteur à aimant permanent

Les moteurs à aimants permanents le rotor possède des pôles NORD et SUD. En raison des aimants permanents, le rotor reste freiné à sa dernière position lorsque le bloc d'alimentation cesse de fournir des impulsions.

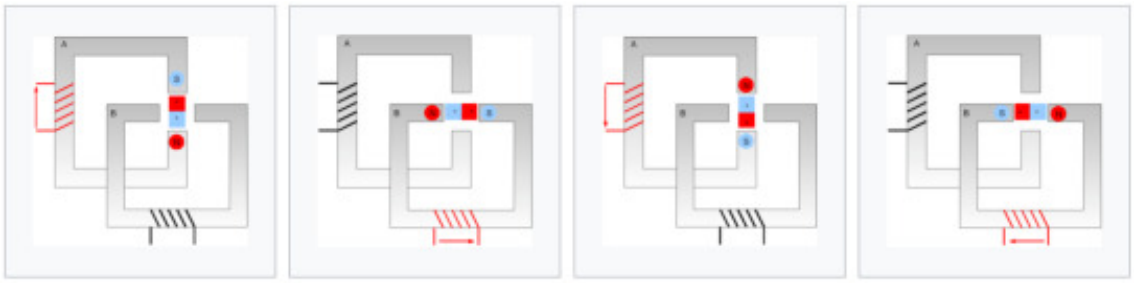
Ces moteurs peuvent être bipolaire ou unipolaire.

18 Moteur bipolaire

Dans cette configuration le courant circule dans les deux sens dans les enroulements des bobines.

19 Moteur à pas complet

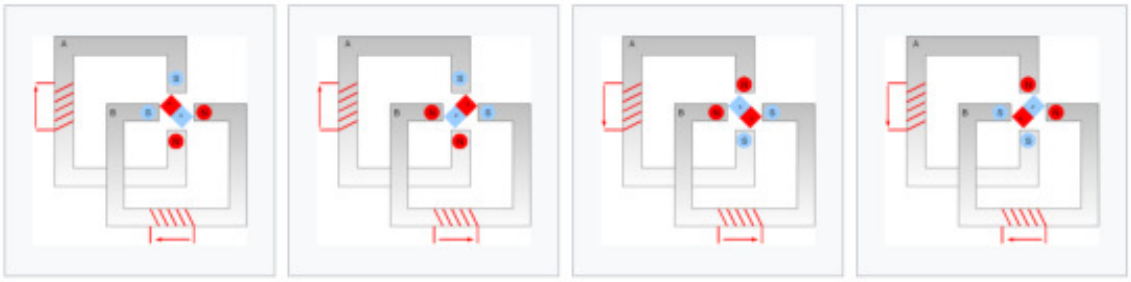
Dans cette configuration, une seule bobine est pilotée à la fois et le courant change de sens de circulation dans la même bobine à chaque demi-révolution.



Configuration à pas complet

20 Moteur à couple maximal

Dans cette configuration, deux bobines sont alimentées en même temps, ce qui fait que le couple est deux fois plus important que dans le cas précédent.



Pas n° 1 Pas n° 2 Pas n° 3 Pas n° 4

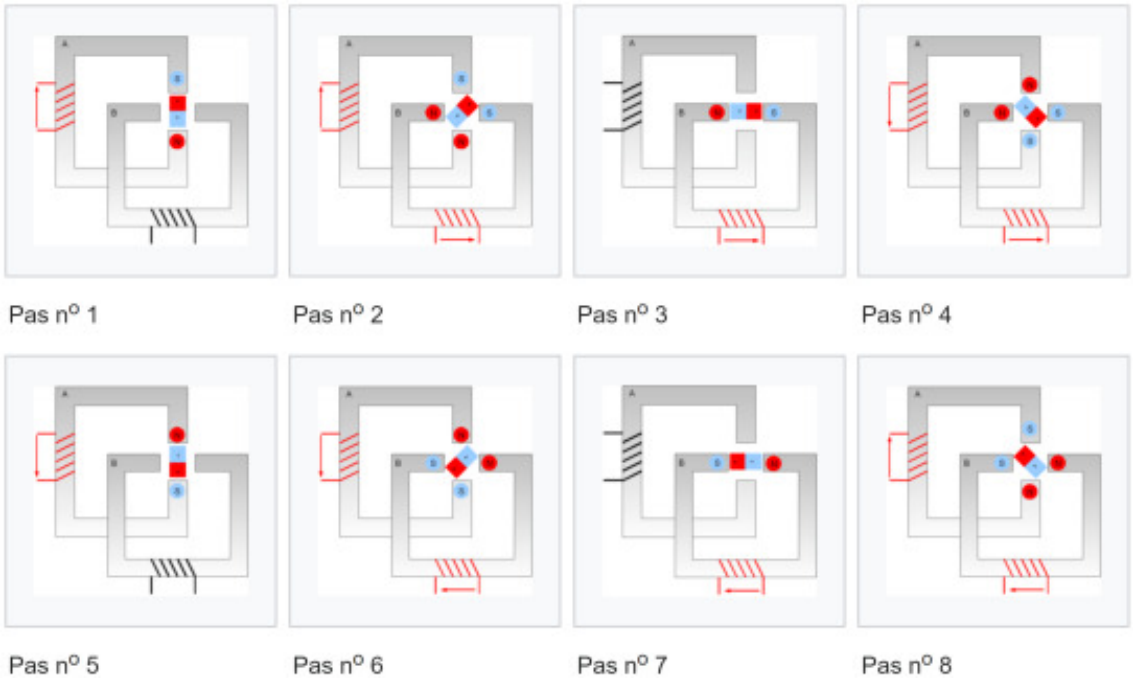
Configuration à fort couple

Impulsion	Bobine A	Bobine A	Bobine B	Bobine B
T1	+	–	+	–
T2	+	–	–	+
T3	–	+	–	+
T4	–	+	+	–

Alimentation des bobines au cours du cycle

21 Moteur à demi-pas

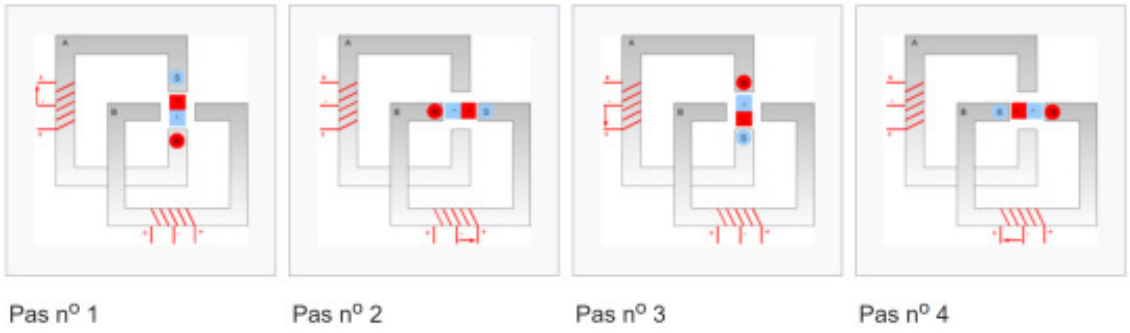
Dans cette configuration, le pas complet et le couple maximal sont combinés, ce qui augmente le nombre de pas par tour mais génère un couple variable.



Configuration en demi-pas

22 Moteur unipolaire

Dans cette configuration, on utilise des demi-bobines (bobines avec un point milieu) afin de ne jamais inverser le sens du courant, ce qui simplifie la commande. Cette configuration nécessite de doubler le nombre d'enroulements, ce qui fait que le moteur est plus coûteux et encombrant.



Configuration unipolaire

23 Moteur hybride

Le moteur pas à pas hybride emprunte du moteur à aimant permanent et de la machine à réluctance variable. Il est donc à réluctance variable mais avec un rotor à aimants permanents. L'avantage est un nombre de pas très élevé.

7. Contrôle de servomoteur

7.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de contrôler un servomoteur en utilisant la bibliothèque Servo.h.

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un servomoteur
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

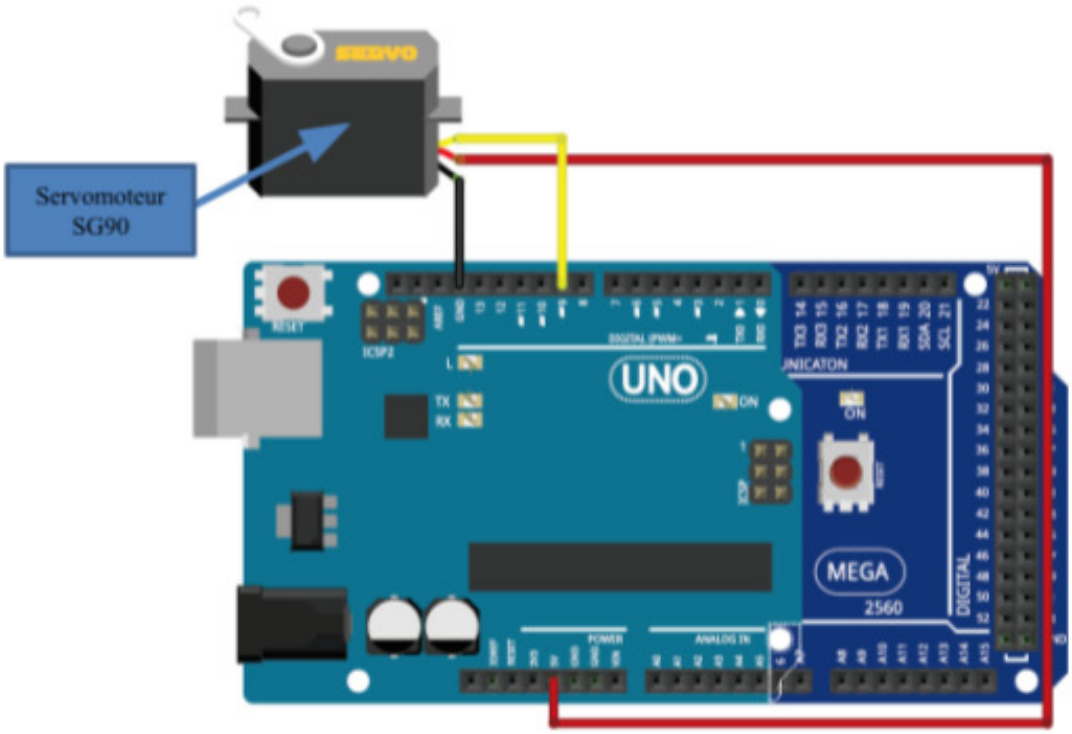
7.2. Contrôle de servomoteur avec la bibliothèque Servo.h

Le programme ci-dessous permet d'afficher les caractères 1 à 9 puis de `a` à `f` sur chaque afficheur en changeant d'afficheur toutes les 0.5s.

```
#include <Servo.h> // Inclut la bibliothèque Servo.h dans le programme
Servo myservo; // Produit un objet Servo à partir de la bibliothèque

void setup()
{
  myservo.attach(9); // Attache le servomoteur au connecteur 9
  myservo.write(0); // Positionne le palonnier à 0 degré
  delay(1000);
}

void loop()
{
  myservo.write(15); // Tourne de 15 degrés
  delay(1000);
  myservo.write(30);
  delay(1000);
  myservo.write(45);
  delay(1000);
  myservo.write(60);
  delay(1000);
  myservo.write(75);
  delay(1000);
  myservo.write(90);
```

```
  delay(1000);
  myservo.write(75);
  delay(1000);
  myservo.write(60); // Retourne à 60 degrés
  delay(1000);
  myservo.write(45);
  delay(1000);
  myservo.write(30);
  delay(1000);
  myservo.write(15);
  delay(1000);
  myservo.write(0);
  delay(1000);
  for(int num=0;num<=180;num++)
  {
     myservo.write(num); // Fait tourner le servomoteur jusqu'à la position angulaire indiquée (0-180)
     delay(10); // Contrôle la vitesse de rotation
  }
  for(int num=180;num>=0;num--)
  {
     myservo.write(num);
     delay(10);
  }
}
```

Le montage à réaliser est le suivant :



Question n°1

Qu'est ce qu'un servo-moteur ?

Question n°2

Réalisez le montage, téléversez le code et constatez le fonctionnement.

Question n°3

Modifiez le montage et le programme de façon à ce que le servo-moteur soit contrôlé par 2 boutons : un qui le fait tourner avec un pas de 10° dans le sens horaire et un autre dans l'autre sens.

8. Contrôle de 2 moteurs à courant continu

8.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est d'apprendre à piloter deux moteurs à courant continu avec un motor shield Arduino

Pour réaliser ce montage, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- deux moteurs à courant continu fonctionnant sous 9V, 18W max
- un Arduino Motor Shield Rev 3
- une alimentation externe de 9V

8.2. Contrôle de 2 moteurs à courant continu

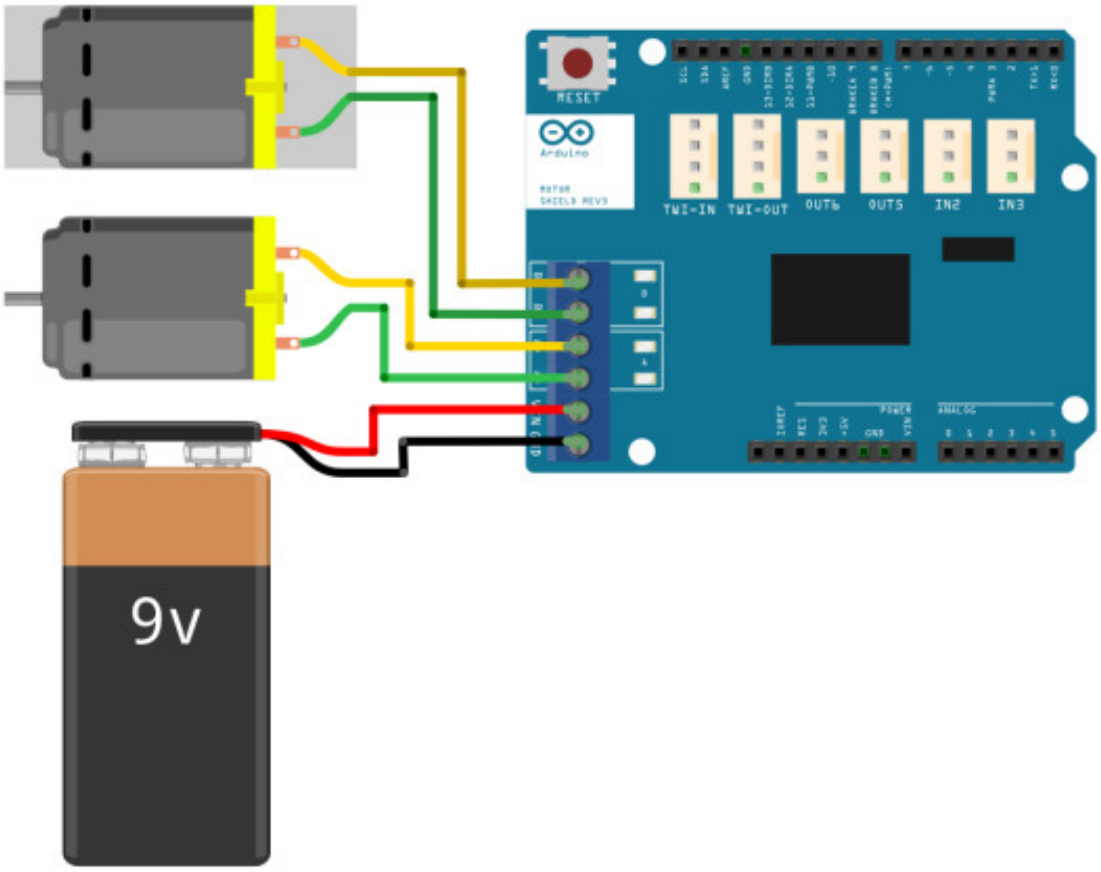
Le programme ci-dessous permet de contrôler deux moteurs à courant continu :

```
1  const int vitesseMotA=3; // Constante pour la broche 3
2  const int sensMotA=12; // Constante pour la broche 12
3  const int freinMotA=9; // Constante pour la broche 9
4  const int intensiteMotA=A0; // intensité du moteur 0
5
6  const int vitesseMotB=11; // Constante pour la broche 11
7  const int sensMotB=13; // Constante pour la broche 13
8  const int freinMotB=8; // Constante pour la broche 8
9  const int intensiteMotB=A1; // intensité du moteur 1
10
11 void setup()
12 {
13     Serial.begin(115200);
14     pinMode (vitesseMotA,OUTPUT); // Broche vitesseMotA configurée en sortie
15     pinMode (freinMotA,OUTPUT); // Broche freinMotA configurée en sortie
16     pinMode (vitesseMotB,OUTPUT); // Broche vitesseMotB configurée en sortie
17     pinMode (sensMotA,OUTPUT); // Broche sensMotA configurée en sortie
18     pinMode (sensMotB,OUTPUT); // Broche senMotB configurée en sortie
19
20     digitalWrite(vitesseMotA,LOW); // a l'arret
21     digitalWrite(sensMotA,LOW);
22     digitalWrite(freinMotA,LOW); // frein off
23
```

```
    digitalWrite(vitesseMotB,LOW); // à l'arret
    digitalWrite(sensMotB,LOW);
    digitalWrite(freinMotB,LOW); // frein off
}

void loop()
{
    //------ test initial du moteur A ----
    //- sens 1
    digitalWrite(sensMotA,LOW); // sens 1
    digitalWrite(vitesseMotA, HIGH); // vitesse maximale
    delay(2000); // 2 secondes
    digitalWrite(vitesseMotA, LOW); // vitesse maximale
    //- sens 2
    digitalWrite(sensMotA,HIGH); // sens 2
    digitalWrite(vitesseMotA, HIGH); // vitesse maximale
    delay(1000); // 2 secondes
    Serial.println(analogRead(intensiteMotA));
    delay(1000); // 2 secondes
    digitalWrite(vitesseMotA, LOW); // vitesse maximale

    //------ test initial du moteur B ----
    //- sens 1
    digitalWrite(sensMotB,LOW); // sens 1
    digitalWrite(vitesseMotB, HIGH); // vitesse maximale
    delay(2000); // 2 secondes

    digitalWrite(vitesseMotB, LOW); // vitesse maximale
    //- sens 2
    digitalWrite(sensMotB,HIGH); // sens 2
    digitalWrite(vitesseMotB, HIGH); // vitesse maximale
    delay(1000); // 2 secondes
    Serial.println(analogRead(intensiteMotB));
    delay(1000); // 2 secondes
    digitalWrite(vitesseMotB, LOW); // vitesse maximale

    //---- test vitesse variable moteur A ---
      for (int i=0; i<=255; i++)
   {
      analogWrite(vitesseMotA,i); // PWM croissant
      delay(50); // pause
      Serial.println(analogRead(intensiteMotA));
    }
      for (int i=0; i<=255; i++)
   {
      analogWrite(vitesseMotA,255-i); // PWM décroissant
      delay(50); // pause
      Serial.println(analogRead(intensiteMotA));
    }

    //---- test vitesse variable moteur B ---
      for (int i=0; i<=255; i++)
    {
      analogWrite(vitesseMotB,i); // PWM croissant
      delay(50); // pause
      Serial.println(analogRead(intensiteMotB));
    }
       for (int i=0; i<=255; i++)
   {
      analogWrite(vitesseMotB,255-i); // PWM décroissant
```

```
84        delay(50); // pause
85        Serial.println(analogRead(intensiteMotB));
86    }
87    while(1); // stop loop
88 }
```

Le montage à réaliser est le suivant. Ce montage utilise un motor shield Arduino qui vient s'emboiter sur le dessus de la carte Uno (branchez une alimentation extérieure sur les bornes Vin et Ground) :



Question n°1

A quoi sert un motor shield ?

Question n°2

Réalisez le montage, téléversez le code, et constatez le fonctionnement.

Question n°3

En imaginant que chaque moteur est en prise avec une des deux roues motrices droite et gauche d'un modèle réduit de voiture, écrivez le code permettant à la voiture d'avancer en ligne droite, de tourner à droite, de tourner à gauche puis de reculer en ligne droite.

9. Contrôle de moteur à courant continu avec les drivers L293D et L298N

9.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de piloter des moteurs à courant continu avec des drivers utilisant des ponts H, plus compacts et beaucoup moins coûteux que des Motor Shield Arduino.

Pour réaliser ces deux montages, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- deux moteurs à courant continu fonctionnant sous 9V, 18W max
- un driver L293D
- un driver L298N
- une alimentation externe de 9V
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

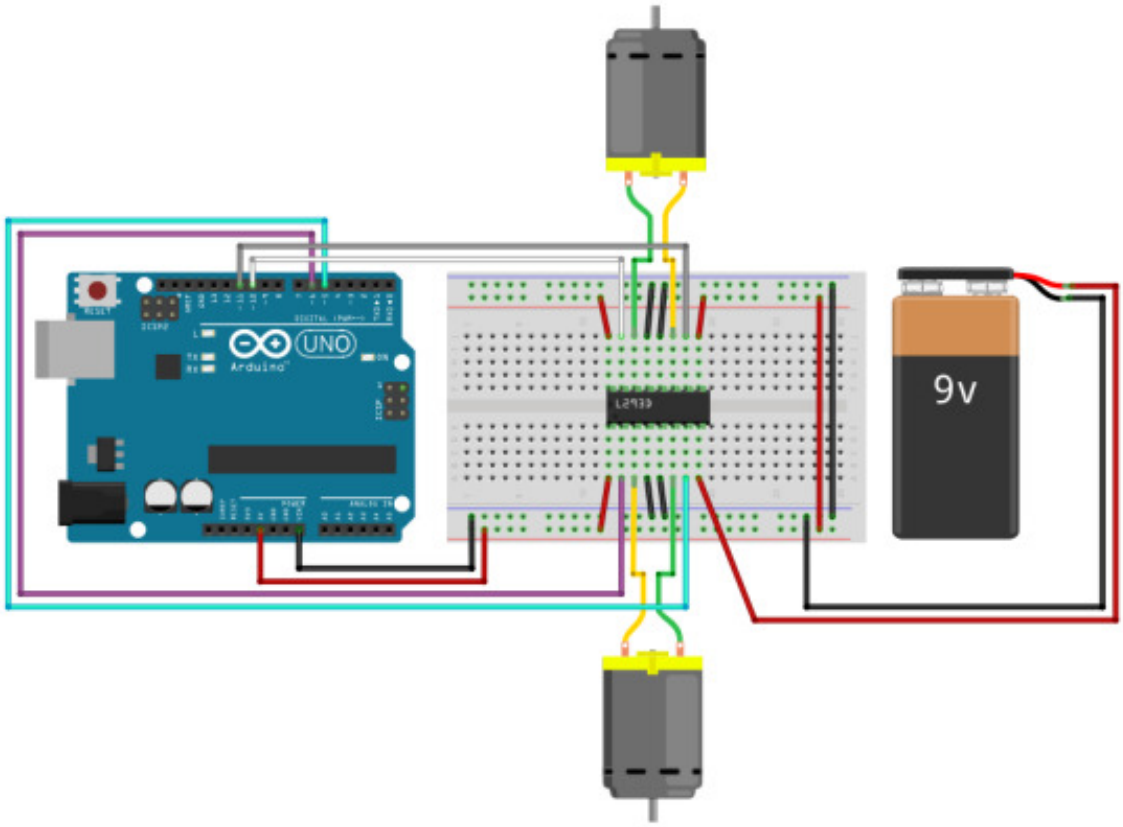
9.2. Contrôle de moteur à courant continu avec les drivers L293D et L298N

Le programme ci-dessous permet de contrôler deux moteurs à courant continu avec un contrôleur L293D :

```
1  //initialisation des broches : GA & GB pour le moteur de gauche, DA & DB pour le moteur de droite
2  int GA=11,GB=10,DA=5,DB=6;
3
4  void setup() {
5    Serial.begin(9600);
6    pinMode(DA,OUTPUT);
7    pinMode(DB,OUTPUT);
8    pinMode(GA,OUTPUT);
9    pinMode(GB,OUTPUT);
10 }
11
12 void arriere() {
13   digitalWrite(DA,HIGH);
14   digitalWrite(DB,LOW);
```

```
15    digitalWrite(GA,HIGH);
16    digitalWrite(GB,LOW);
17  }
18  void avant() {
19    digitalWrite(DA,LOW);
20    digitalWrite(DB,HIGH);
21    digitalWrite(GA,LOW);
22    digitalWrite(GB,HIGH);
23  }
24  void gauche() {
25    digitalWrite(DA,LOW);
26    digitalWrite(DB,HIGH);
27    digitalWrite(GA,HIGH);
28    digitalWrite(GB,LOW);
29  }
30  void droite() {
31    digitalWrite(DA,HIGH);
32    digitalWrite(DB,LOW);
33    digitalWrite(GA,LOW);
34    digitalWrite(GB,HIGH);
35  }
36
37  void loop() {
38    avant(); delay(1000);
39    arriere(); delay(1000);
40    gauche(); delay(1000);
41    droite(); delay(1000);
42  }
43
```

Le montage à réaliser est le suivant :



Question n°1

Comment fonctionne le driver L293D ?

Question n°2

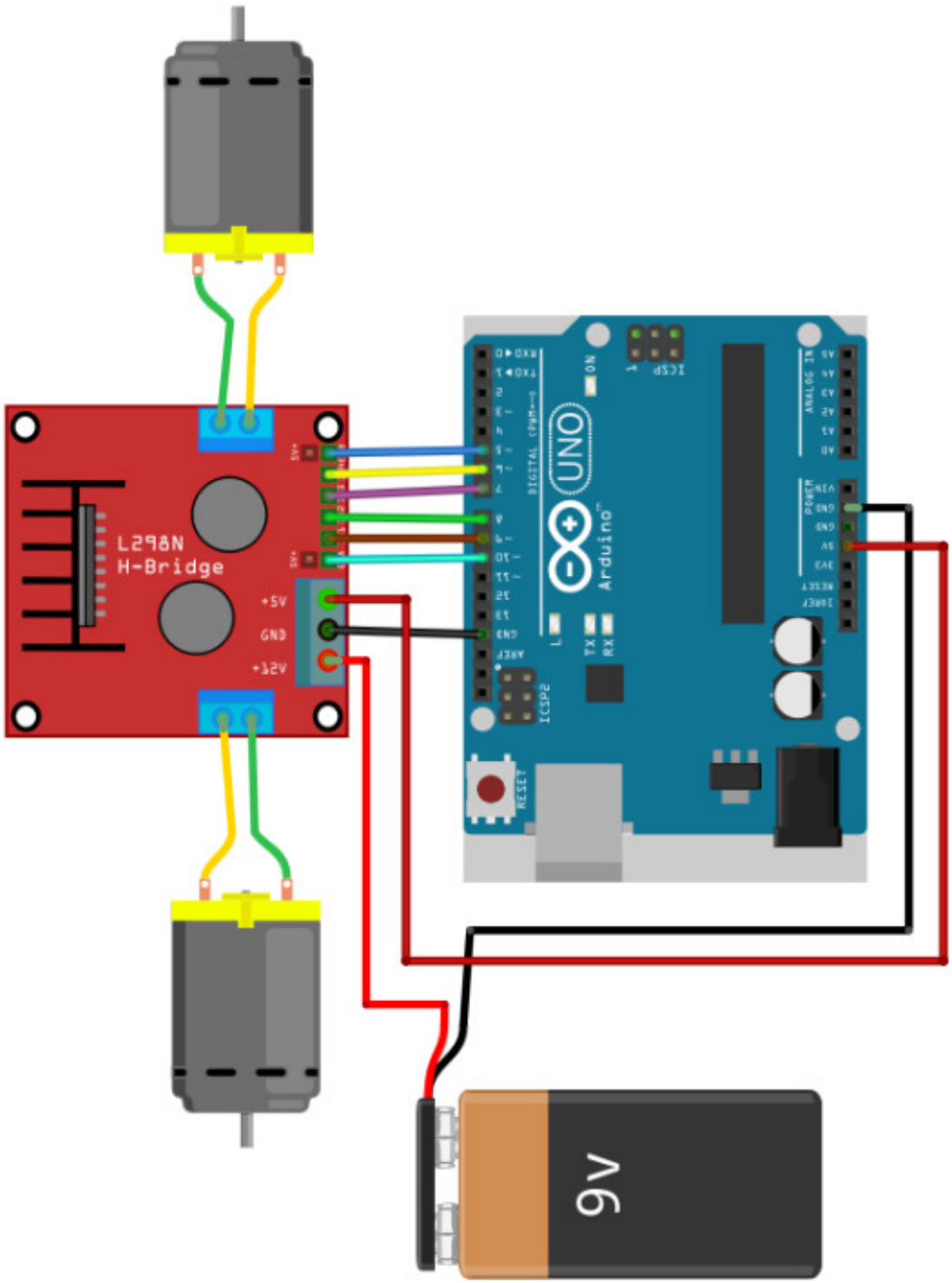
Réalisez le montage, téléversez le code, et constatez le fonctionnement.

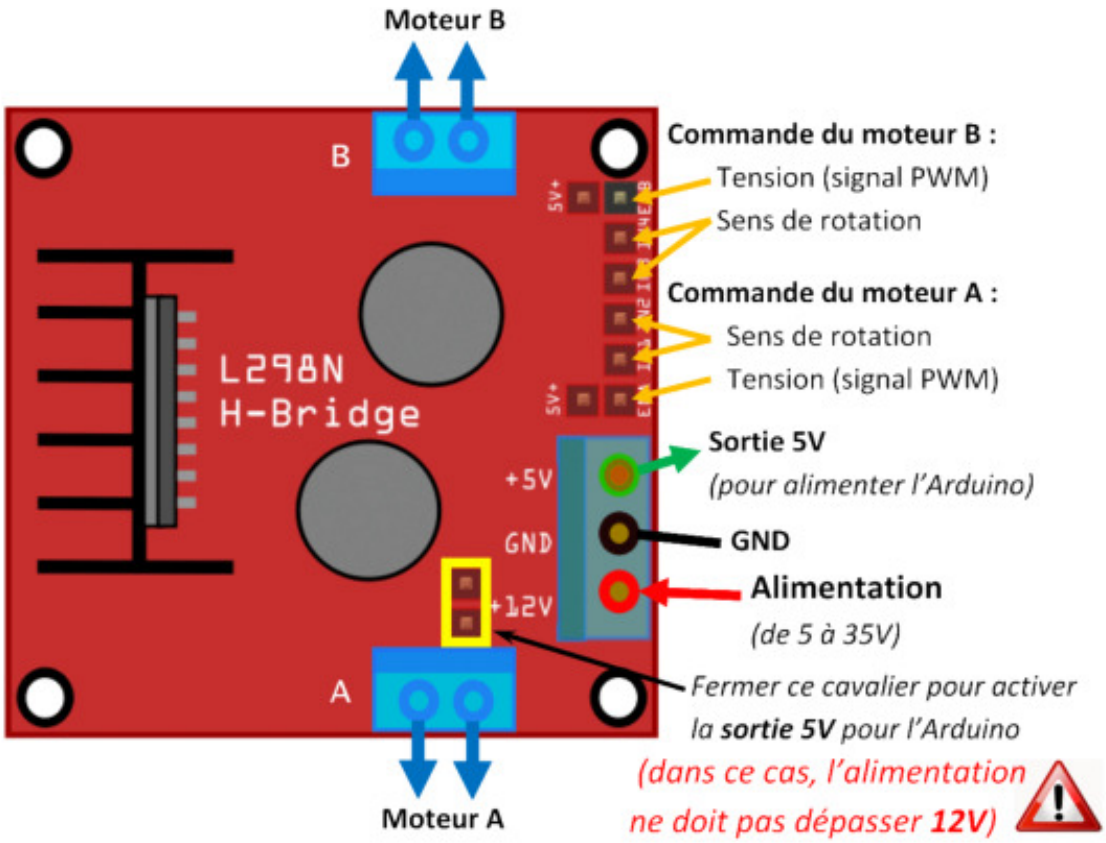
Question n°3

Modifiez le code et le montage pour qu'un moteur soit piloté par deux boutons poussoir, un pour la rotation dans un sens et l'autre pour la rotation dans le sens contraire.

9.3.

Vous pouvez également utiliser un driver L298N pour piloter un moteur à courant continu (ce driver peut également contrôler un moteur pas-à-pas), le montage est alors le suivant :


L298N
H-Bridge
+5V
GND
+12V
UNO
Arduino
9v



Le programme est alors le suivant :

```
1  //Ports de commande du moteur A
2  int motorA1 = 9; //In2
3  int motorA2 = 8; //In1
4  int enableA = 10; //ENA
5
6  //Ports de commande du moteur B
7  int motorB1 = 7; //In3
8  int motorB2 = 6; //Int4
9  int enableB = 5; //ENB
10
11 // Vitesse du moteur
12 int state = 0;
13
14 void setup() {
15
16     pinMode(motorA1, OUTPUT);
17     pinMode(motorA2, OUTPUT);
18     pinMode(enableA, OUTPUT);
19     pinMode(motorB1, OUTPUT);
20     pinMode(motorB2, OUTPUT);
21     pinMode(enableB, OUTPUT);
22
23     Serial.begin(9600);
24 }
25
26 void loop() {
27     if (Serial.available() > 0)
28     {
29       // Lecture de l'entier passé au port série
30       state = Serial.parseInt();
```

```
31
32      if (state > 0) // avant
33      {
34        digitalWrite(motorA1, HIGH);
35        digitalWrite(motorA2, LOW);
36        digitalWrite(motorB1, HIGH);
37        digitalWrite(motorB2, LOW);
38        Serial.print("Avant ");
39        Serial.println(state);
40      }
41      else if (state < 0) // arrière
42      {
43        digitalWrite(motorA1, LOW);
44        digitalWrite(motorA2, HIGH);
45        digitalWrite(motorB1, LOW);
46        digitalWrite(motorB2, HIGH);
47        Serial.print("Arrière ");
48        Serial.println(state);
49      }
50      else // Stop (freinage)
51      {
52        digitalWrite(motorA1, HIGH);
53        digitalWrite(motorA2, HIGH);
54        digitalWrite(motorB1, HIGH);
55        digitalWrite(motorB2, HIGH);
56        Serial.println("Stop");
57      }
58
59      // Vitesse du mouvement
60      analogWrite(enablePin, abs(state));
61    }
62    delay(100);
63 }
```

Question n°1

En vous aidant d'internet, déterminez la puissance maximale des moteurs pouvant être piloté par ce driver

Question n°2

A quoi sert la pièce noire dotée d'ailettes directement vissée sur le pont H L298N ?

Question n°3

Réalisez le montage, téléversez le code et constatez le fonctionnement

10. Contrôle de moteur pas-à-pas

10.1. Introduction

Objectifs pédagogiques

Le but de cette ressource est d'apprendre à piloter un petit moteur pas à pas, pratique et bon marché avec un driver ULN2003.

Pour réaliser ces deux montages, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un moteur pas à pas unipolaire 28BYJ-48
- un driver ULN2003
- des fils Dupont Mâle/Mâle

10.2. Contrôle de moteur pas-à-pas par commande simplifiée

Le programme ci-dessous permet de contrôler un moteur pas-à-pas 28BYJ-48 avec un contrôleur ULN2003 de la manière la plus simple possible mais dans ce cas le moteur ne développera pas tout son couple :

```
int IN1 = 8;      // pin digital 8 relié à IN1
int IN2 = 9;      // pin digital 9 à IN2
int IN3 = 10;     // pin digital 10 à IN3
int IN4 = 11;     // pin digital 11 à IN4
int temps = 20;  // temps entre les pas, minimum 10 ms.

void setup(){
  // tous les pins se configurent comme sorties
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop(){
  // 512*4 = 2048 pas pour un tour complet
  for (int i = 0; i < 512; i++) {
    digitalWrite(IN1, HIGH);  // pas 1
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
```

```
    delay(temps);

    digitalWrite(IN1, LOW);   // pas 2
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(temps);

    digitalWrite(IN1, LOW);   // pas 3
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    delay(temps);

    digitalWrite(IN1, LOW);   // pas 4
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(temps);
  }
  // pause de 5 secondes
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  delay(5000);
}
```

Ce programme peut être réduit en utilisant un tableau pour piloter les sorties du driver :

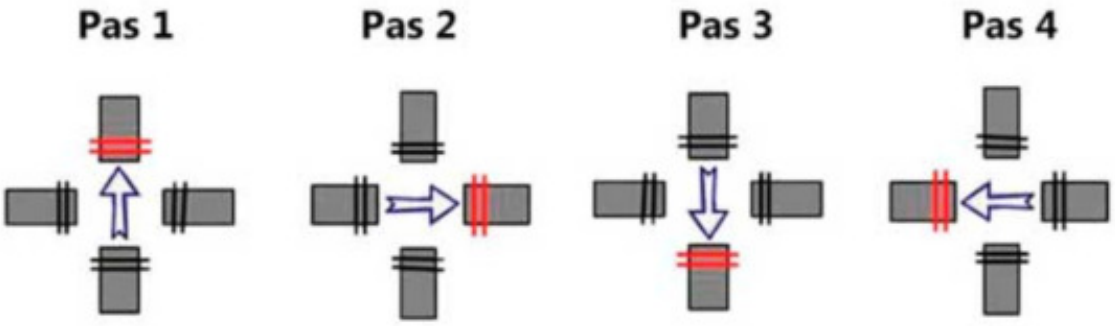
```
int IN1 = 8;      // pin digital 8 relié à IN1
int IN2 = 9;      // pin digital 9 à IN2
int IN3 = 10;     // pin digital 10 à IN3
int IN4 = 11;     // pin digital 11 à IN4
int temps = 20;  // temps entre les pas, minimun 10 ms.

// Tableau bidimensionnel avec la séquence des pas
int pas [4][4] = {
  {1, 0, 0, 0},
  {0, 1, 0, 0},
  {0, 0, 1, 0},
  {0, 0, 0, 1}
};

void setup(){
  // tous les pins se configurent comme sorties
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop(){
  // 512 cycles * 4 pas par cycle = 2048 pas par tour d'axe en sortie de réducteur
  for (int i = 0; i < 512; i++){
    // boucle qui lit le tableau ligne par ligne
    for (int i = 0; i < 4; i++){
      digitalWrite(IN1, pas[i][0]);
      digitalWrite(IN2, pas[i][1]);
      digitalWrite(IN3, pas[i][2]);
      digitalWrite(IN4, pas[i][3]);
```

```
32        delay(temps);
33      }
34    }
35    // pause
36    digitalWrite(IN1, LOW);
37    digitalWrite(IN2, LOW);
38    digitalWrite(IN3, LOW);
39    digitalWrite(IN4, LOW);
40    delay(temps);
41  }
```

Dans ce cas, les bobines sont pilotées ainsi :

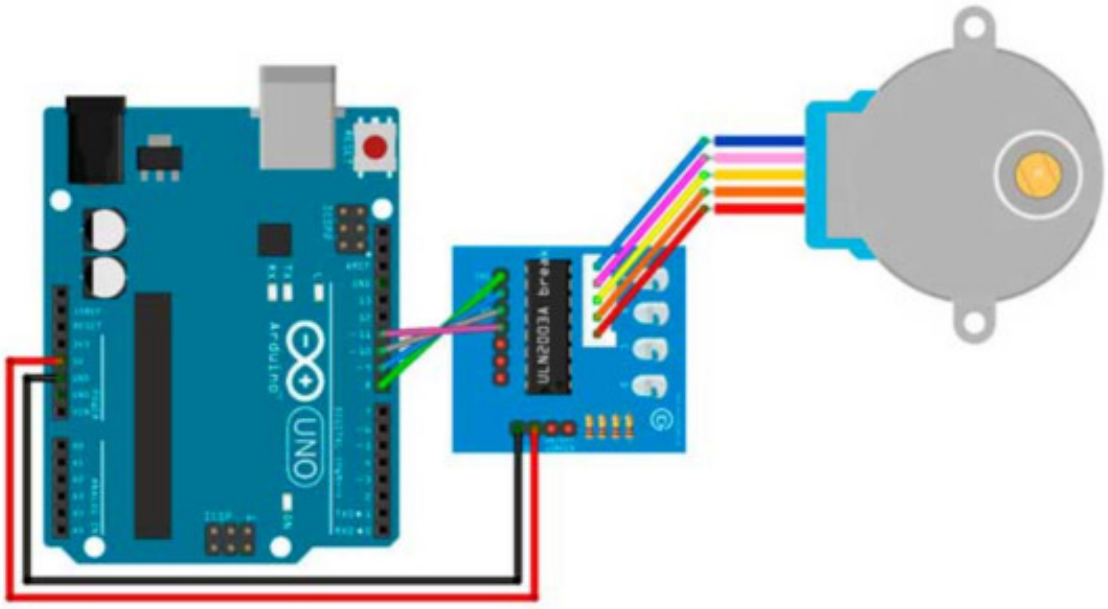
Séquence pas à pas Unipolaire

Pas	A	B	C	D
pas 1	1	0	0	0
pas 2	0	1	0	0
pas 3	0	0	1	0
pas 4	0	0	0	1



Pour plus de détails : Commander un moteur pas-à-pas*[https://ledisrupteurdimensionnel.com/arduino/connexion-du-moteur-pas-a-pas-28byj-48-et-le-module-unl2003-a-la-carte-darduino/]*

Le montage à réaliser est le suivant :



Ce moteur nécessite en théorie une alimentation de 12V pour fonctionner correctement mais vous pouvez l'alimenter avec la carte Arduino si vous le faites tourner à vide. Si vous possédez une alimentation 12V, branchez le + sur le Vin du driver et reliez les masses de l'alimentation et de la carte Arduino pour éviter les phénomènes de masse flottante.

Question n°1

Réalisez le montage, téléversez le code et constatez son fonctionnement.

Question n°2

Modifiez le programme pour faire faire au moteur 3 tours dans un sens et 1 tour dans l'autre.

10.3. Contrôle de moteur pas-à-pas par commande permettant de développer le couple maximal

Le programme ci-dessous permet de contrôler un moteur pas-à-pas 28BYJ-48 avec un contrôleur ULN2003 de la manière à ce qu'il développe tout son couple :

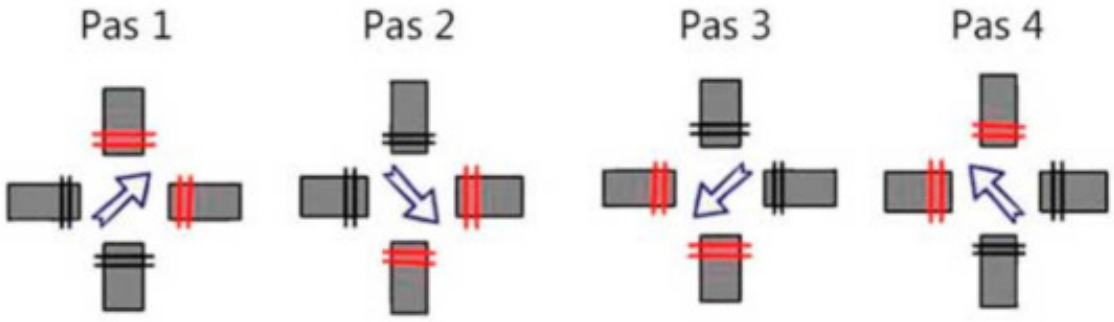
```
int IN1 = 8;      // pin digital 8 relié à IN1
int IN2 = 9;      // pin digital 9 à IN2
int IN3 = 10;     // pin digital 10 à IN3
int IN4 = 11;     // pin digital 11 à IN4
int temps = 20;  // temps entre les pas, minimum 10 ms.
// Array bidimensionnel indiquant la séquence des pas
int pas [4][4] ={
  {1, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 1, 1},
  {1, 0, 0, 1}
};

void setup(){
  // tous les pins se configurent comme sorties
```

```
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop(){
  // 512*4 = 2048 pas
  for (int i = 0; i < 512; i++){
    // boucle repasse le array ligne à ligne
    for (int i = 0; i < 4; i++){
      // valeurs qu'on va appliquer
      digitalWrite(IN1, pas[i][0]);
      digitalWrite(IN2, pas[i][1]);
      digitalWrite(IN3, pas[i][2]);
      digitalWrite(IN4, pas[i][3]);
      delay(temps);
    }
  }
  // pause de 5 secondes
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  delay(temps);
}
```

Dans ce cas, les bobines sont pilotées ainsi :

Séquence pas à pas unipolaire avec deux bobines

Pas	A	B	C	D
Pas 1	1	1	0	0
Pas 2	0	1	1	0
Pas 3	0	0	1	1
Pas 4	1	0	0	1



Question

Téléversez le code, et constatez le fonctionnement

10.4. Contrôle de moteur pas-à-pas par commande demi-pas

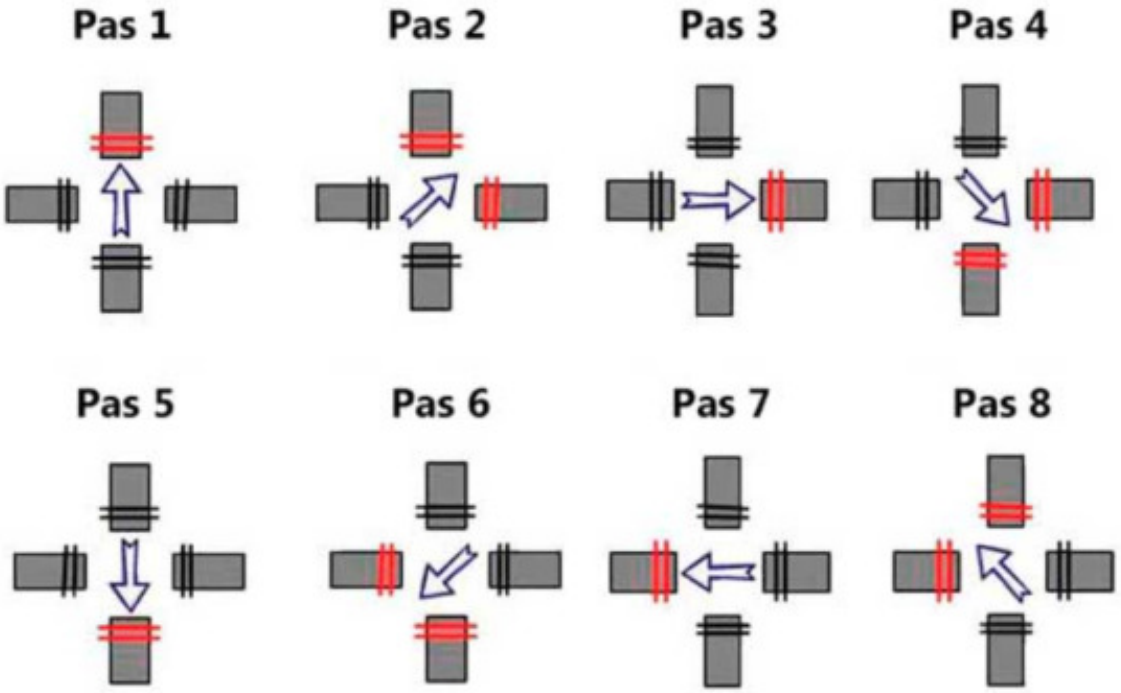
Le programme ci-dessous permet de contrôler un moteur pas-à-pas 28BYJ-48 avec un contrôleur ULN2003 en étant plus précis (commande par demi-pas) :

```
int IN1 = 8;      // pin digital 8 relié à IN1
int IN2 = 9;      // pin digital 9 à IN2
int IN3 = 10;     // pin digital 10 à IN3
int IN4 = 11;     // pin digital 11 à IN4
int temps = 20;  // temps entre les pas, minimum 10 ms.
// Array bidimensionnel qu'indique la séquence des pas
int pas [8][4] ={
  {1, 0, 0, 0},
  {1, 1, 0, 0},
  {0, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 1, 0},
  {0, 0, 1, 1},
  {0, 0, 0, 1},
  {1, 0, 0, 1}
};

void setup(){
  // tous les pins se configurent comme sorties
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop(){
  // 512*8 = 4096 pas
  for (int i = 0; i < 512; i++){
    // boucle repasse le array ligne à ligne
    for (int i = 0; i < 8; i++){
      // valeurs qu'on va appliquer
      digitalWrite(IN1, pas[i][0]);
      digitalWrite(IN2, pas[i][1]);
      digitalWrite(IN3, pas[i][2]);
      digitalWrite(IN4, pas[i][3]);
      delay(temps);
    }
  }
  // pause de 5 secondes
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  delay(temps);
}
```

Dans ce cas, les bobines sont pilotées ainsi :

Séquence pas à pas avec deux bobines, mi-pas

Pas	A	B	C	D
Pas 1	1	0	0	0
Pas 2	1	1	0	0
Pas 3	0	1	0	0
Pas 4	0	1	1	0
Pas 5	0	0	1	0
Pas 6	0	0	1	1
Pas 7	0	0	0	1
Pas 8	1	0	0	1



Question

Téléversez le code, et constatez le fonctionnement.

10.5. Contrôle de moteur pas-à-pas avec bibliothèque stepper Arduino

Le programme précédent peut être considérablement simplifié en utilisant la bibliothèque stepper d'Arduino :

```
1 // charge une bibliothèque qui permet de piloter des moteurs pas à pas simplement
2 #include <Stepper.h>
3
```

```
// fixe le nombre de pas du moteur à 513 par tour
const int stepsPerRevolution = 513;

// fixe la vitesse du moteur : 0 = min, 10 = max
int motorSpeed=10;

// initialise la bibliothèque en fonction des broches du driver
// dans le cas présent : (step,1N2,1N4,1N3,1N1)
Stepper myStepper(stepsPerRevolution, 9, 11, 10, 8);

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

void loop()
{
  // définit de la vitesse de révolution du moteur
  myStepper.setSpeed(motorSpeed);
  Serial.print("Vitesse : ");
  Serial.println(motorSpeed);

  // fait tourner le moteur de 513 pas dans le sens trigonométrique
  Serial.println("Rotation en sens trigonométrique");
  myStepper.step(stepsPerRevolution);
  delay(1500);

  // fait tourner le moteur de 513 pas dans le sens horaire
  Serial.println("Rotation en sens horaire");
  myStepper.step(-stepsPerRevolution);
  delay(1500);
}
```

Question n°1

Téléversez le code, et constatez le fonctionnement.

Question n°2

Modifiez le programme pour demander à l'utilisateur le nombre de pas qu'il souhaite faire faire au moteur et à quelle vitesse pendant 2s. Limitez le programme à un seul sens de rotation.

11. Contrôle de moteur pas-à-pas avec driver TB6600

11.1. Introduction

Objectifs pédagogiques

L'objectif de cette ressource est de piloter un moteur pas-à-pas qu'on trouve généralement dans les imprimantes 3D avec un driver de qualité

Pour réaliser ces deux montages, il faudra vous munir des composants suivants :

- une carte Arduino
- un câble USB
- un moteur pas à pas bipolaire Nema 17
- un driver TB6600
- une alimentation externe de 12V
- des fils Dupont Mâle/Mâle
- une platine de prototypage rapide à trous de type Labdec

11.2. Contrôle de moteur pas-à-pas avec driver TB6600

Le programme ci-dessous permet de contrôler un moteur pas-à-pas Nema 17 avec un contrôleur TB6600 :

```
1  const int ena = 2;
2  const int dir = 3;
3  const int pul = 4;
4
5  void setup()
6  {
7    pinMode(ena, OUTPUT);
8    pinMode(dir, OUTPUT);
9    pinMode(pul, OUTPUT);
10   // Pour activer le driver, si HIGH : le moteur passe en rotation libre et le
     driver ne consomme plus de courant
11   digitalWrite(ena, LOW);
12 }
13
14 void loop()
15 {
16   digitalWrite(dir,HIGH); // Définit le sens de rotation
17
18   // Envoie 6400 pulsations pour faire tourner le moteur
```

```
  for(int x = 0; x < 6400; x++)
  {
    digitalWrite(pul,HIGH);
    delayMicroseconds(300);
    digitalWrite(pul,LOW);
    delayMicroseconds(300);
  }
  delay(1000);

  // Change le sens de rotation
  digitalWrite(dir,LOW);
  for(int x = 0; x < 6400; x++)
  {
    digitalWrite(pul,HIGH);
    delayMicroseconds(300);
    digitalWrite(pul,LOW);
    delayMicroseconds(300);
  }
  delay(1000);
}
```

Pour rappel, la séquence de pilotage des bobines d'un moteur bipolaire est :

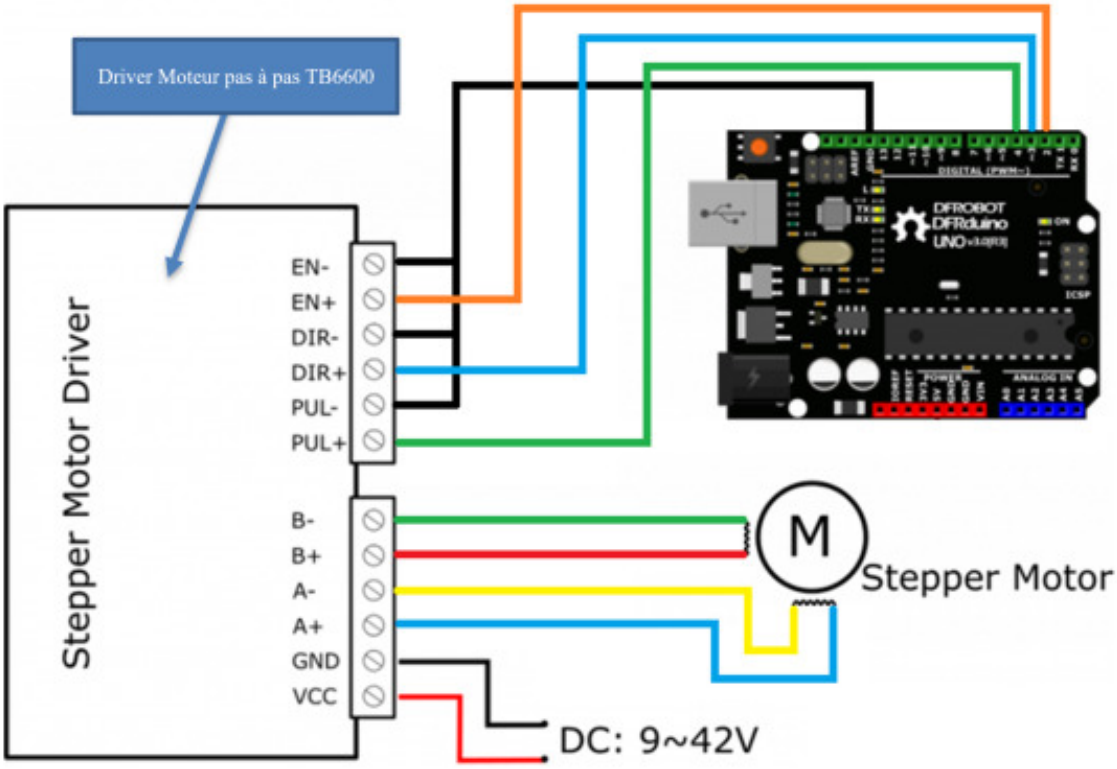
Impulsion	Bobine A	Bobine A	Bobine B	Bobine B
T1	+	–	+	–
T2	+	–	–	+
T3	–	+	–	+
T4	–	+	+	–

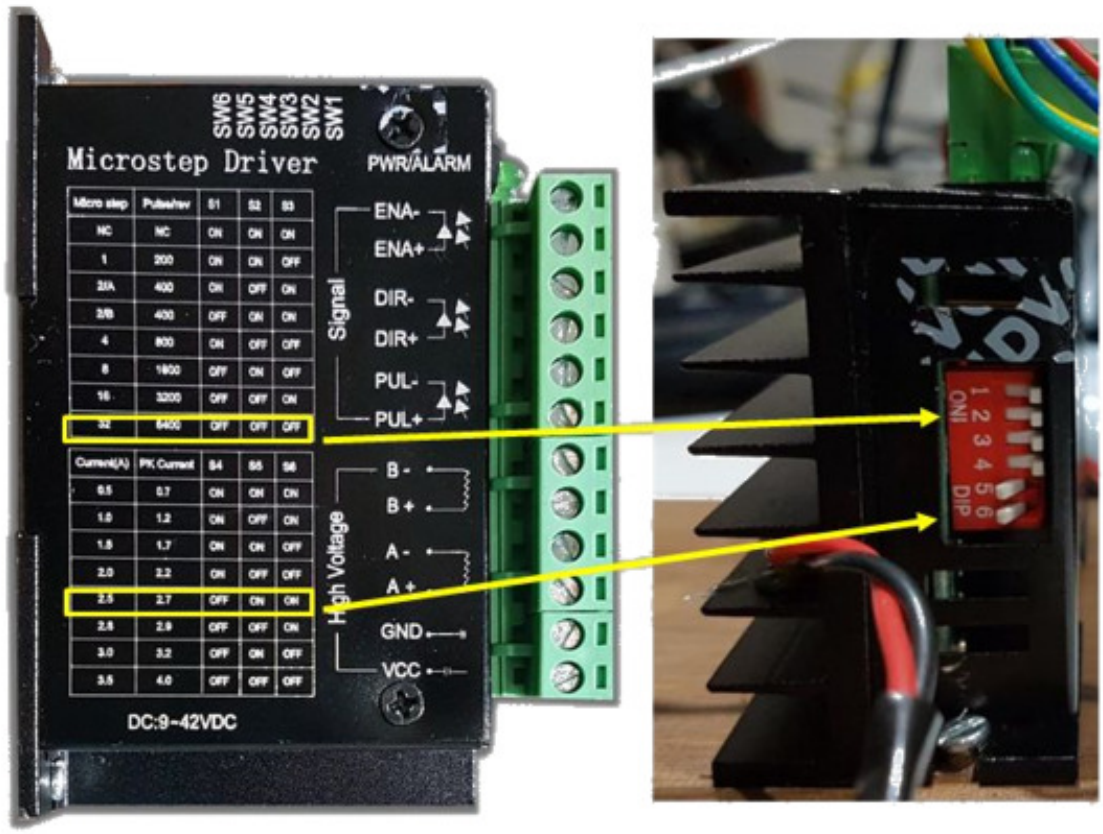
Alimentation des bobines au cours du cycle

Le montage à réaliser est le suivant (réglez les micro interrupteurs du driver de sorte à ce que le moteur reçoive un courant de 2A et 6400 pulsations par tour).

Branchez le Vin et le Ground sur une alimentation 12V.

Si vous ne disposez pas d'alimentation 12V, vous pouvez recycler une veille alimentation de PC : Transformer une alimentation de PC en alimentation d'atelier électronique *[https://www.latelierdugeek.fr/2013/05/11/transformer-une-alimentation-de-pc-en-alimentation-datelier/]*





Truc & astuce

Pour identifier sans la datasheet les paires de bobines sur votre moteur pour savoir comment les relier au A+A- ou B+B-, débranchez le moteur, faites tourner son axe pour ressentir sa résistance naturelle, prenez un des fils, reliez-le à un autre, faites tourner l'axe, si vous sentez une résistance

> plus grande c'est qu'il s'agit de deux fils de la même bobine (car elle est alors en court-circuit et s'oppose donc à la rotation du moteur), sinon essayez avec un autre fil jusqu'à sentir une résistance. Une fois les paires identifiées, mettez une paire sur les broches A-A+ et une autre sur les broches B-B+.

Question n°1

Quels sont les avantages de ce driver ?

Question n°2

Réalisez le montage, téléversez le code, et constatez le fonctionnement

Glossaire

Photorésistance *≃ résistance photogénique, cellule photoconductrice ou cellule photoéléctrique*	Une photorésistance est un composant électronique dont la résistance varie en fonction de la quantité de lumière incidente : plus elle est éclairée, plus sa résistance baisse et réciproquement.