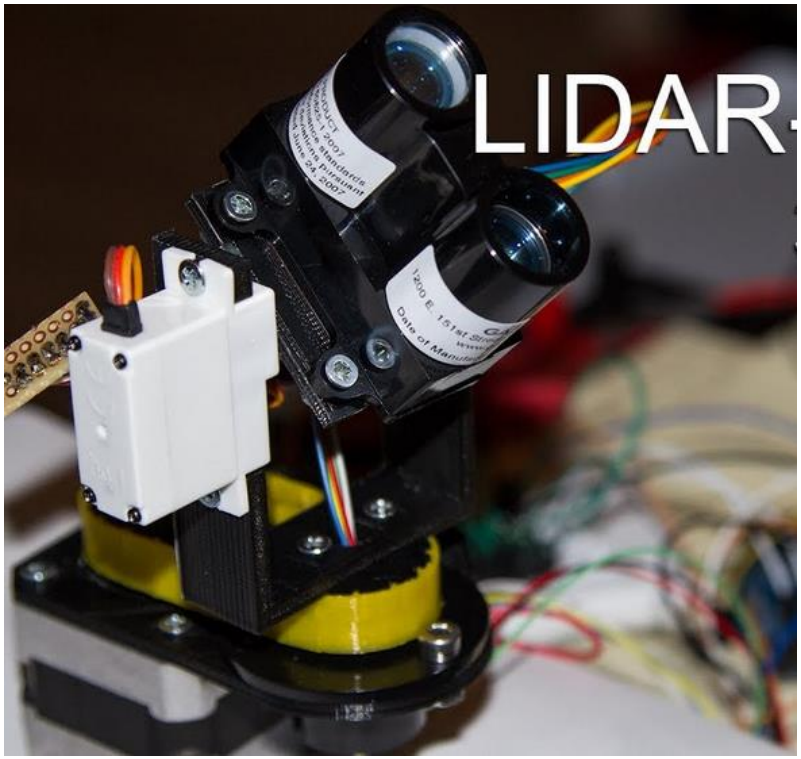
Eléments de robotique avec Arduino



Pascal MASSON

(pascal.masson@unice.fr)

Version projection
Edition 2018-2019-V10

light detection and ranging

POLYTECH NICE-SOPHIA

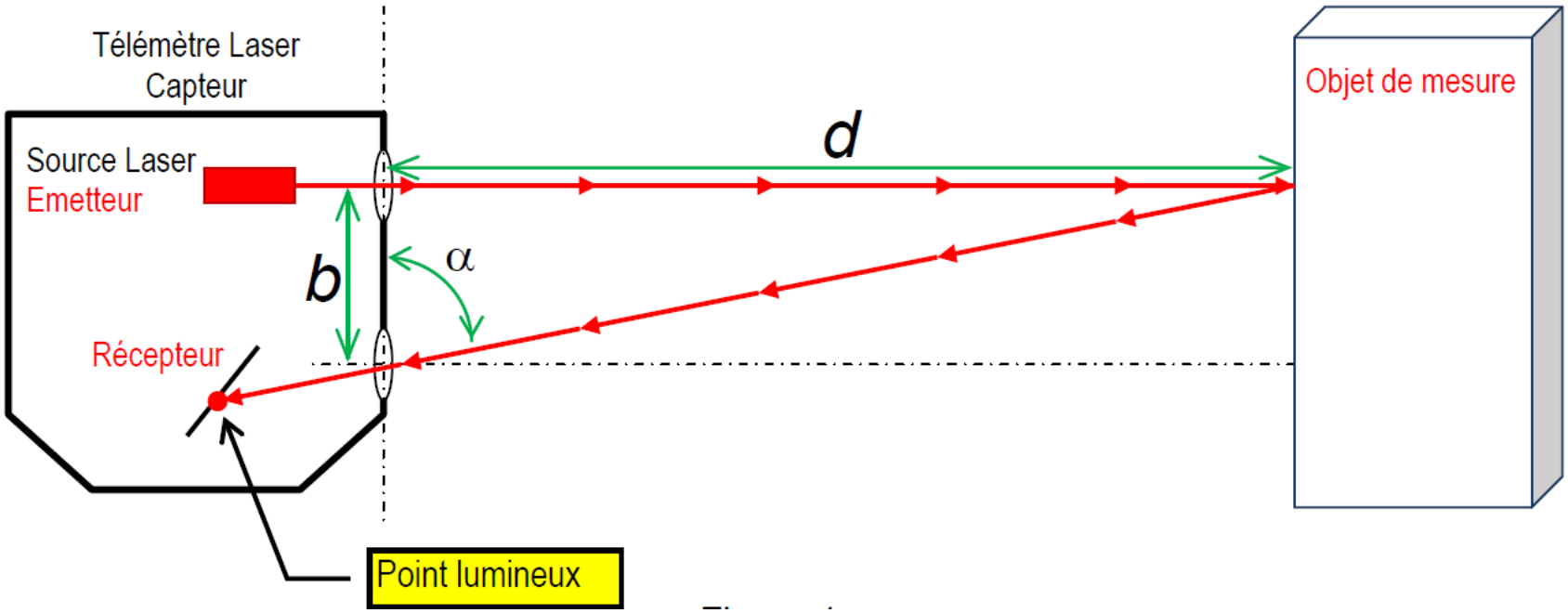
École Polytechnique Universitaire de Nice Sophia-Antipolis
Parcours des écoles d'ingénieurs Polytech
1645 route des Lucioles, 06410 BIOT

Université Nice SOPHIA ANTIPOLIS

Sommaire

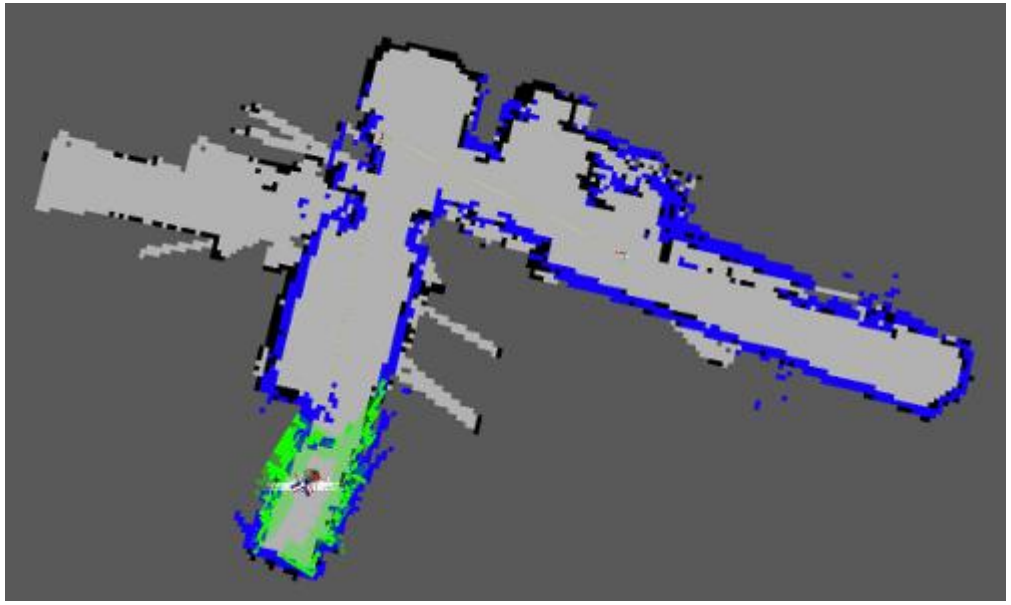
Introduction

Télémesure laser

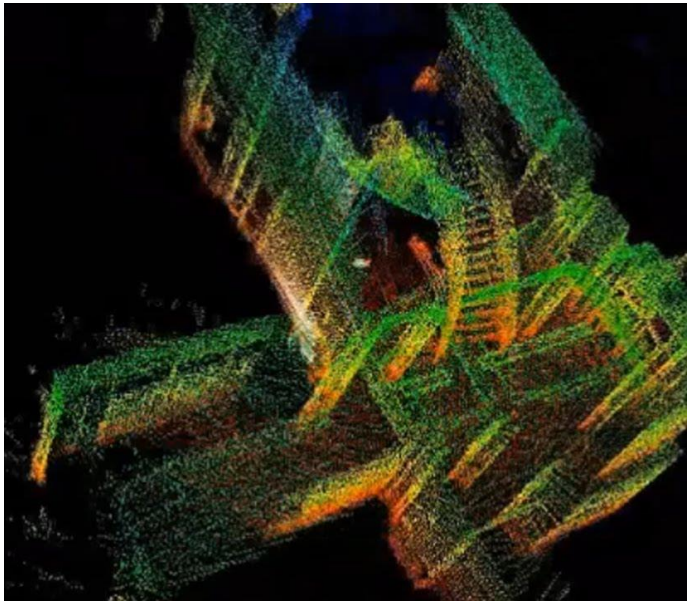


Introduction

- **Mapping**



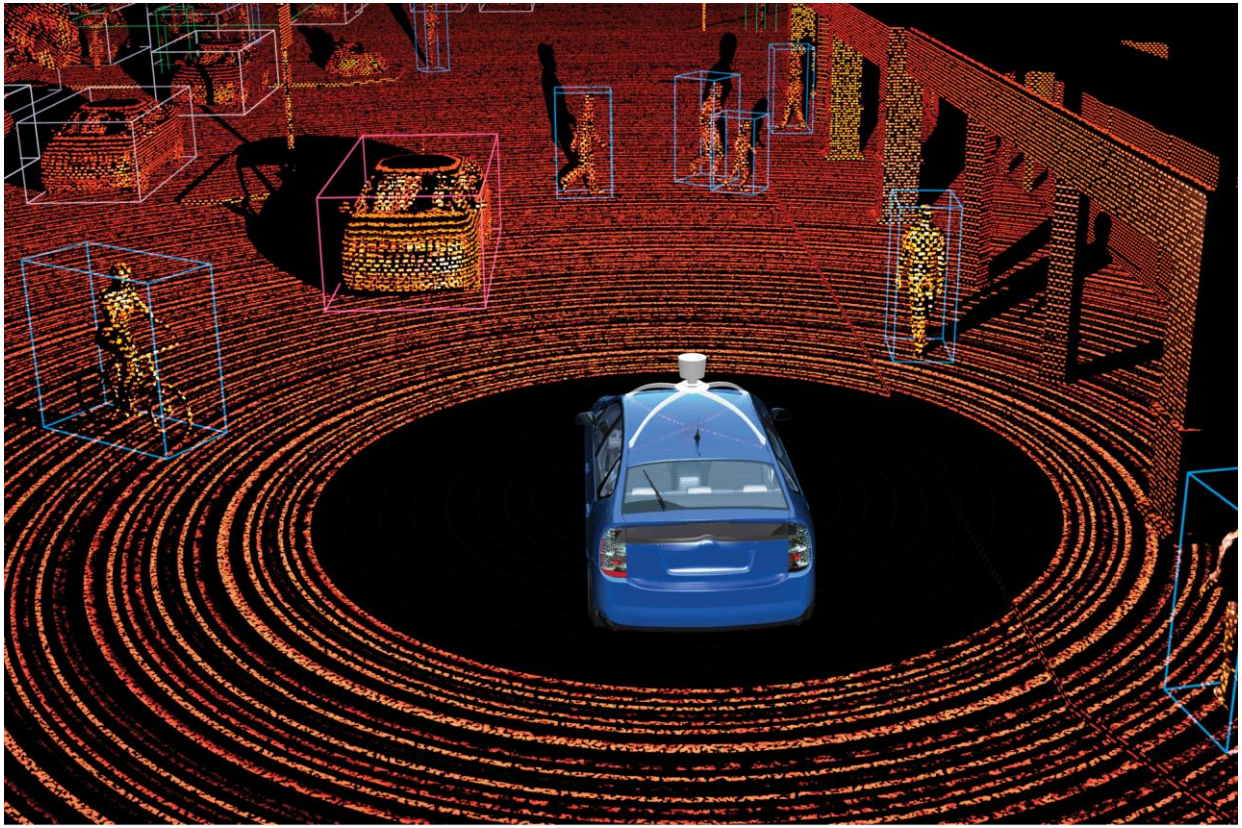
2D



3D

Introduction

- **Voiture autonome**



1. Moteur pas à pas

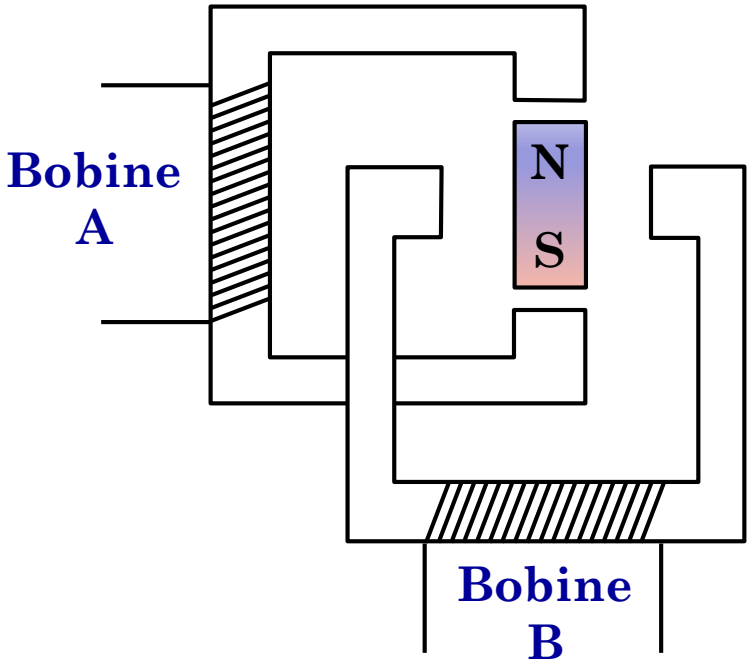
1.1. Description

- **Qu'est ce qu'un moteur pas a pas ?**
  - C'est un moteur qui est un compromis entre le moteur CC et le servomoteur
  - Il est possible de le faire tourner à des vitesses variables et on contrôle précisément sa position si on connait sa position initiale
  - Conrairement au servomoteur, il n'y a pas un capteur qui permet de connaitre la position angulaire du moteur
  - Il en existe plusieurs types en fonction du nombre de fils qui en sortent : 4 (moteur bipolaire), 5 et 6 (moteurs unipolaires ou à réluctance variable)
- **Dans quoi les utilise t'on ?**
  - On les trouve dans tous les objets qui nécessitent une rotation précise comme par exemple les imprimantes quelles soient pour le papier (ou pour le plastique (3D)

1.2. Type de moteur et fonctionnement

Moteur bipolaire

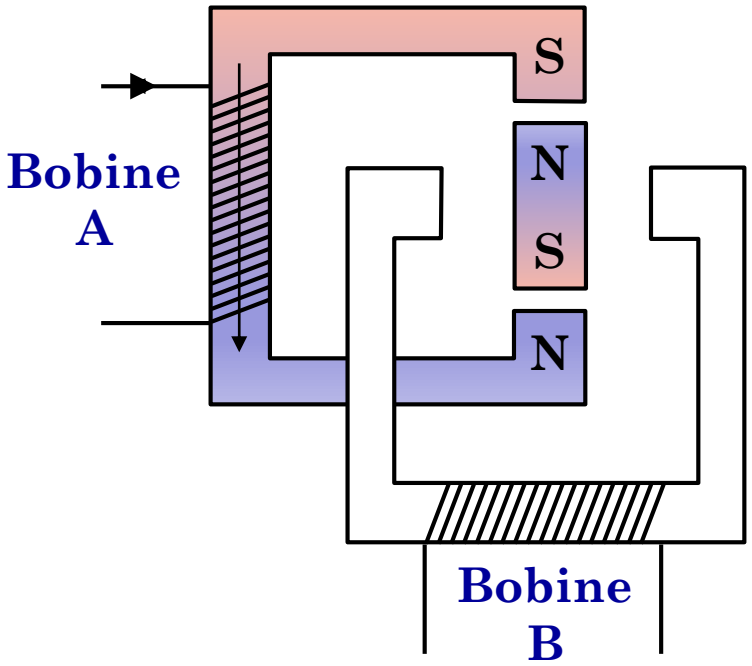
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement



1.2. Type de moteur et fonctionnement

Moteur bipolaire

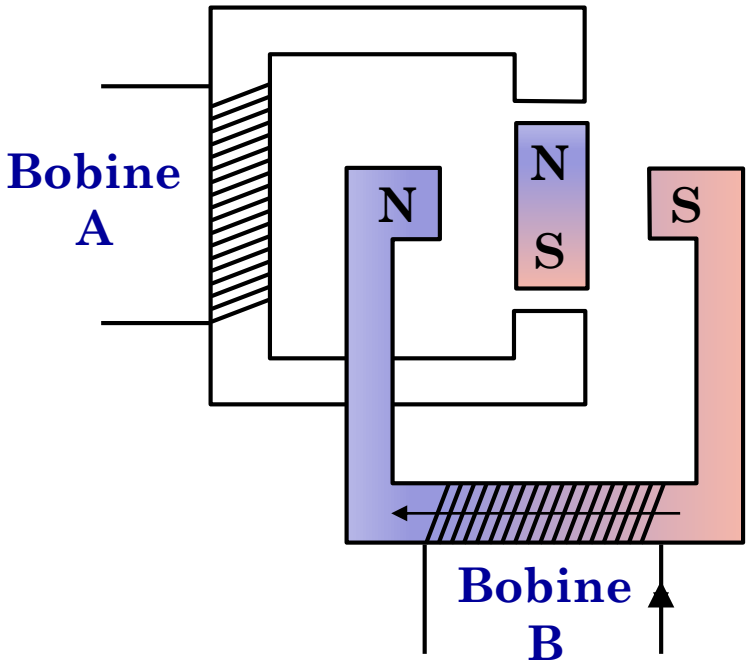
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud



1.2. Type de moteur et fonctionnement

Moteur bipolaire

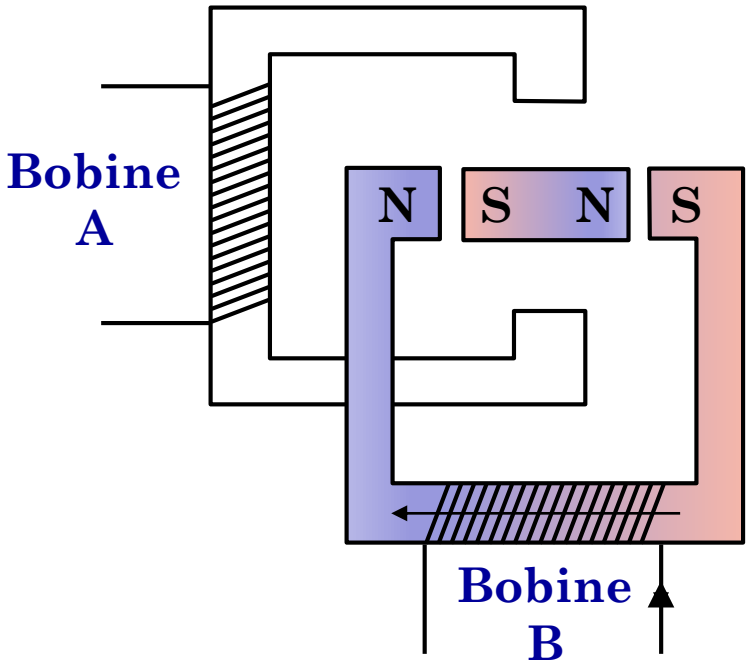
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud



1.2. Type de moteur et fonctionnement

Moteur bipolaire

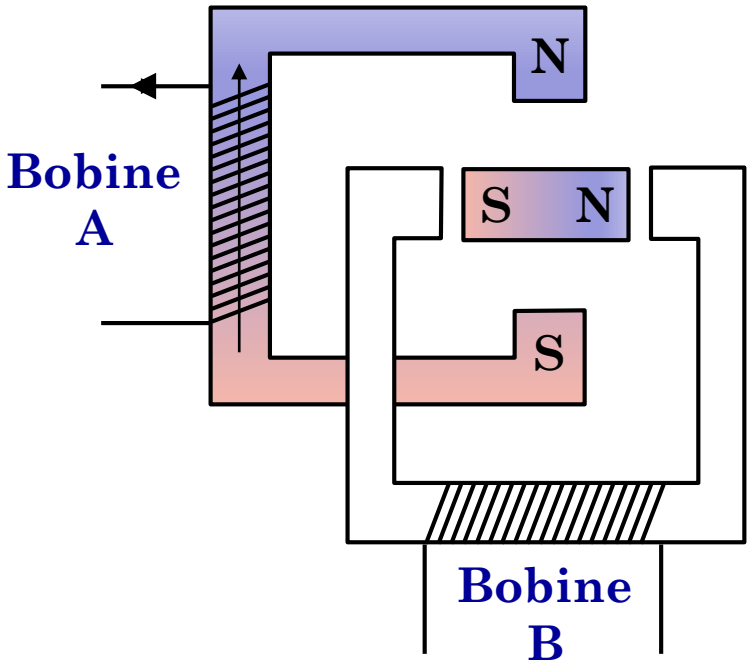
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas



1.2. Type de moteur et fonctionnement

Moteur bipolaire

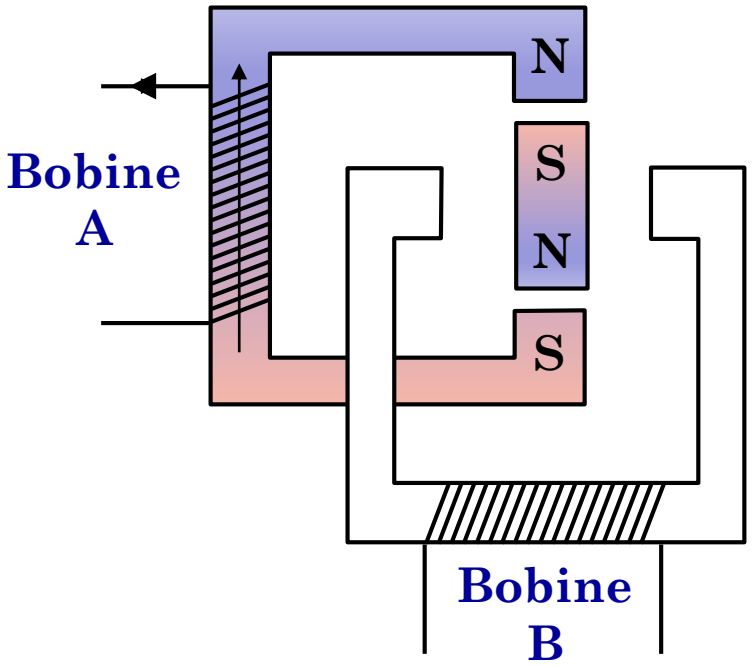
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas



1.2. Type de moteur et fonctionnement

Moteur bipolaire

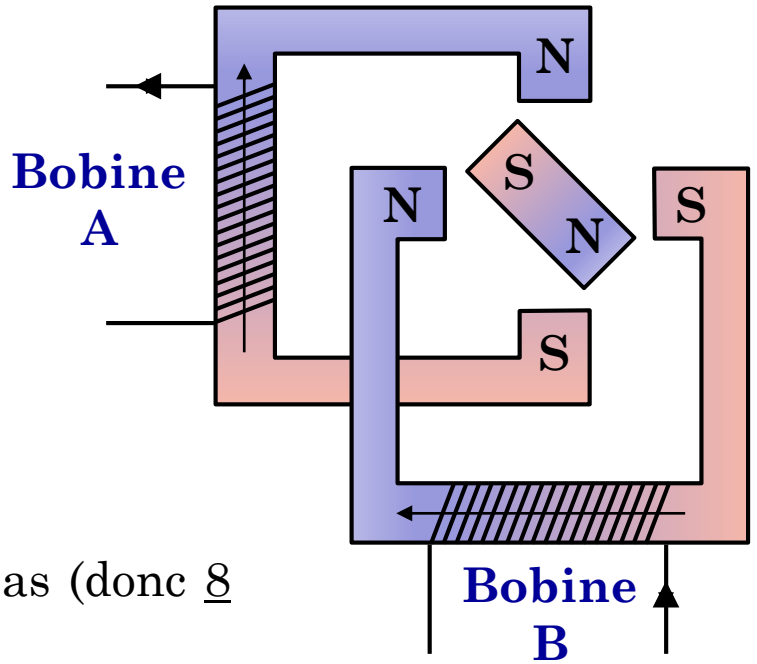
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas
- Dans cet exemple il faut 4 séquences (donc 4 pas) pour que le rotor fasse un tour complet.



1.2. Type de moteur et fonctionnement

Moteur bipolaire

- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas
- Dans cet exemple il faut 4 séquences (donc 4 pas) pour que le rotor fasse un tour complet.

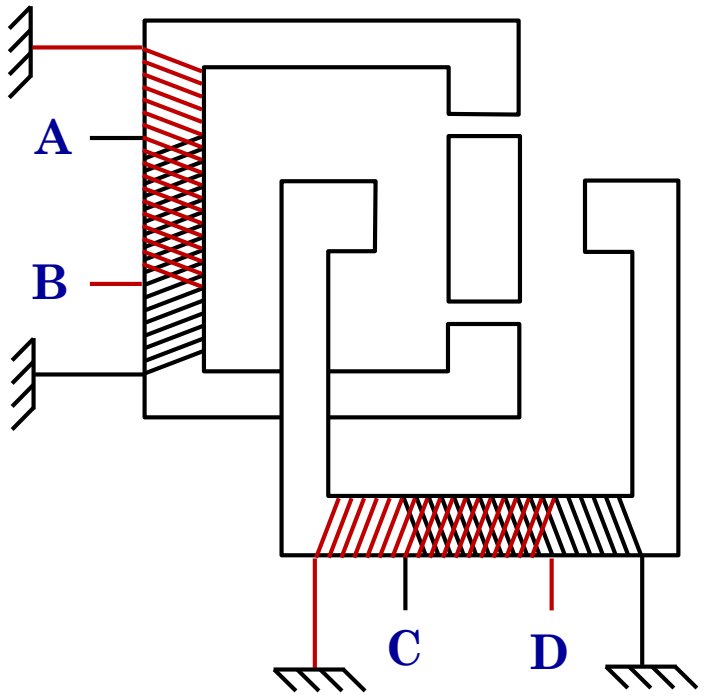


- On peut multiplier par 2 le nombre de pas (donc 8 séquences) en alimentant les 2 bobines
- Il existe des moteurs qui ont beaucoup plus de pas : 24, 48 …

1.2. Type de moteur et fonctionnement

Moteur unipolaire

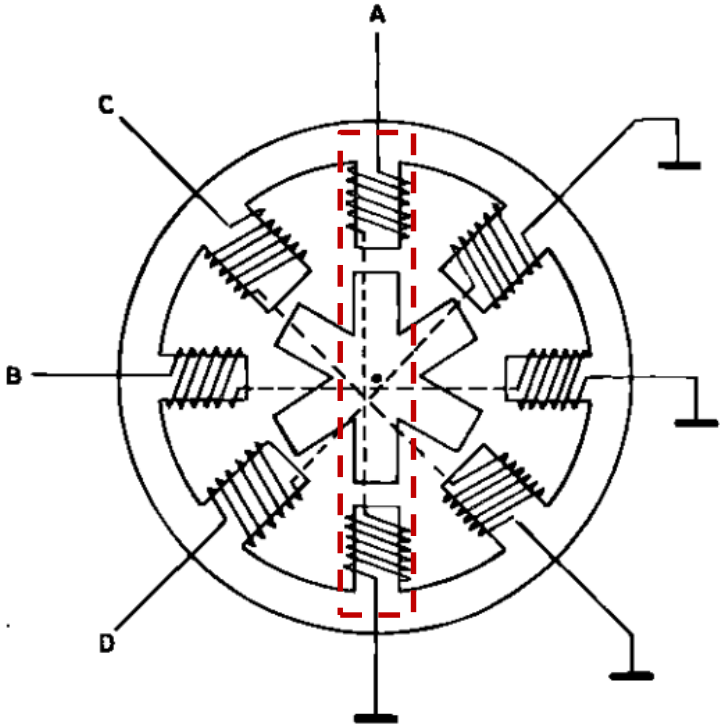
- Ce moteur possède 4 fils pour alimenter les 4 bobines et 1 aimant
- On alimente les 4 bobines successivement et le courant ne passe que dans un seul sens (unipolaire)



1.2. Type de moteur et fonctionnement

Moteur à réluctance variable

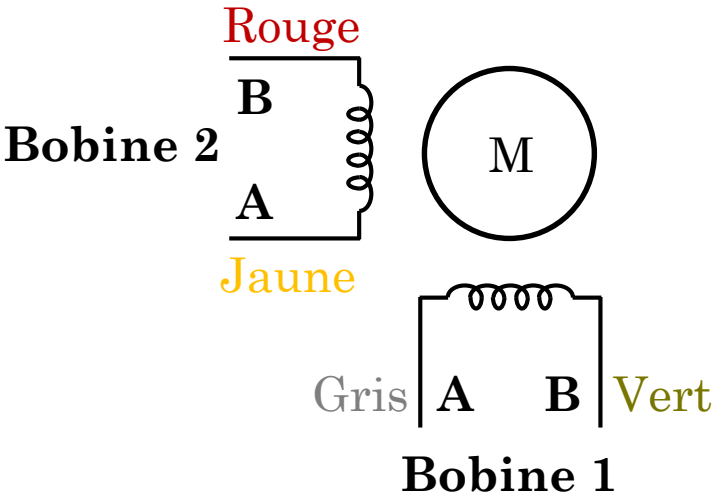
- Il est constitué de 4 bobines et de fer doux pour le rotor en remplacement de l'aimant
- Le fer doux conduit très bien le champ magnétique et cherche toujours à s'aligner avec lui



1.3. Exemple du moteur NEMA17

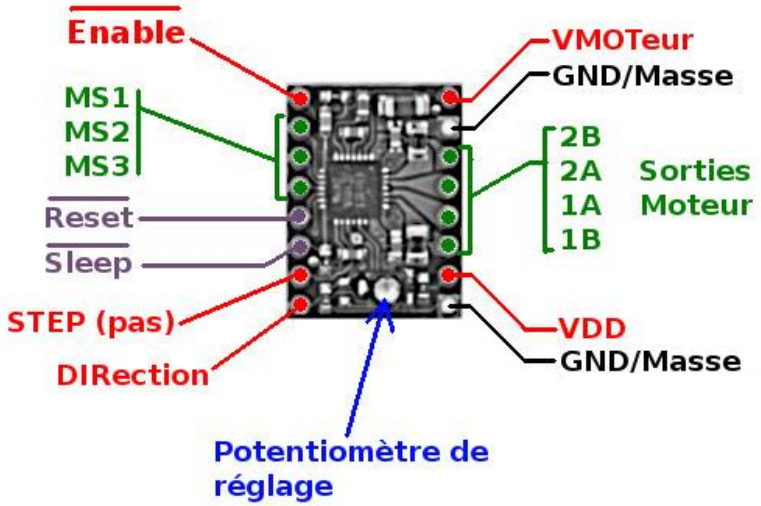
Présentation du moteur

- NEMA est l'abréviation de « National Electrical Manufacturers Association»
- Il mesure 1.7" de côté (soit 4.3 cm). Il existe aussi les NEMA 23
- C'est un moteur bipolaire qui s'alimente en 12 V pour obtenir un couple de 20 N.cm. On peut utiliser une tension plus faible mais le couple chute
- Le pas est de 1.8 °



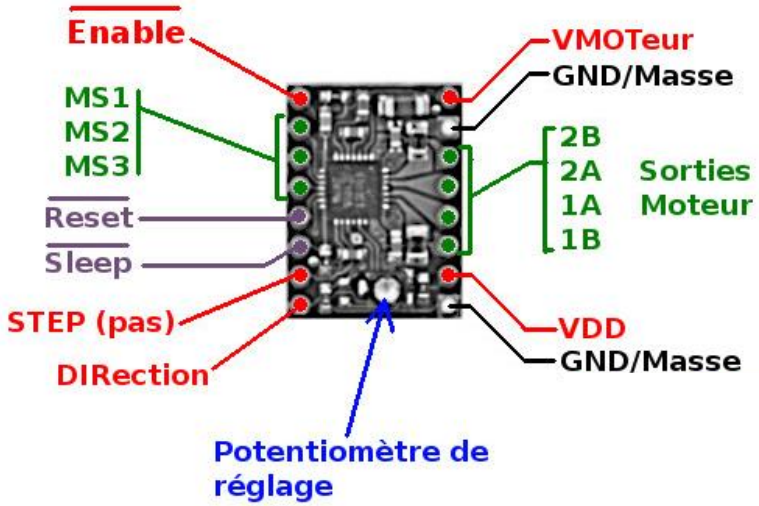
1.3. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : description**
  - Ce driver (ou contrôleur de moteur pas-à-pas) permet un contrôle très simple du moteur
  - Avec un radiateur, il peut faire passer jusqu'à 2 A
  - Il faut lui connecter l'alimentation de l'Arduino (VDD et GND) et celle du moteur (VMOTeur et GND)
  - L'entrée Enable est active à l'état bas et par défaut (sans rien connecter) elle est à l'état bas
  - L'entrée Reset est active à l'état bas et par défaut (sans rien connecter) elle est à l'état haut



1.3. Exemple du moteur NEMA17

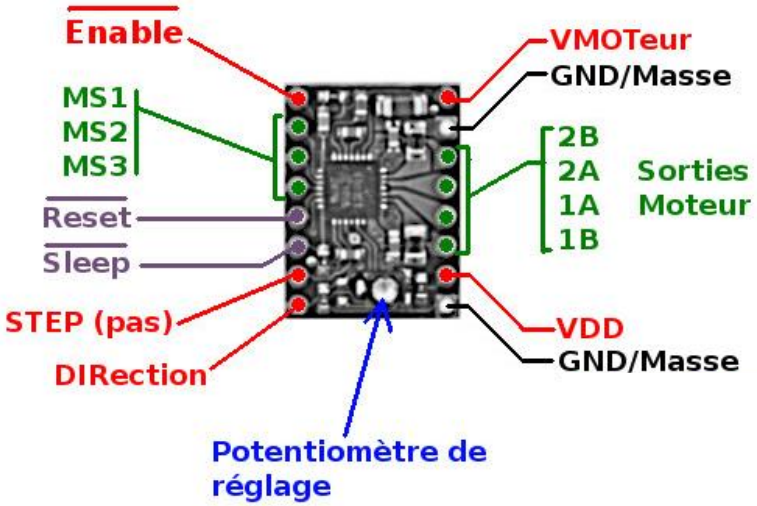
- **Mise en œuvre avec le driver A4988 : description**
  - L'entrée Sleep est active à l'état bas et permet de faire rentrer le module en mode basse consommation. On peut la connecter à l'entrée Reset pour qu'elle soit toujours inactive
  - L'entrée STEP permet de faire tourner le moteur d'un pas a chaque passage à l'état haut. Cela signifie qu'un simple générateur de signal carré permet de faire tourner le moteur
  - L'entrée DIR permet de sélectionner le sens de rotation (état haut pour le sens horaire)



1.3. Exemple du moteur NEMA17

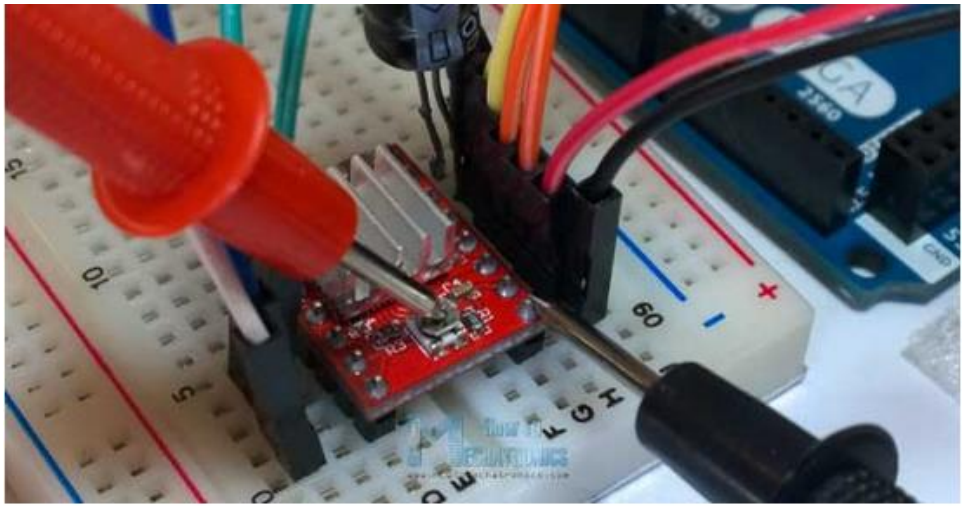
- **Mise en œuvre avec le driver A4988 : description**
  - Les entrées MS1 à 3 permettent de sélectionner la résolution du pas en accord avec la table de vérité
  - Par défaut les entrées MS1 à 3 sont à l'état haut ce qui sélectionne le mode « Full »

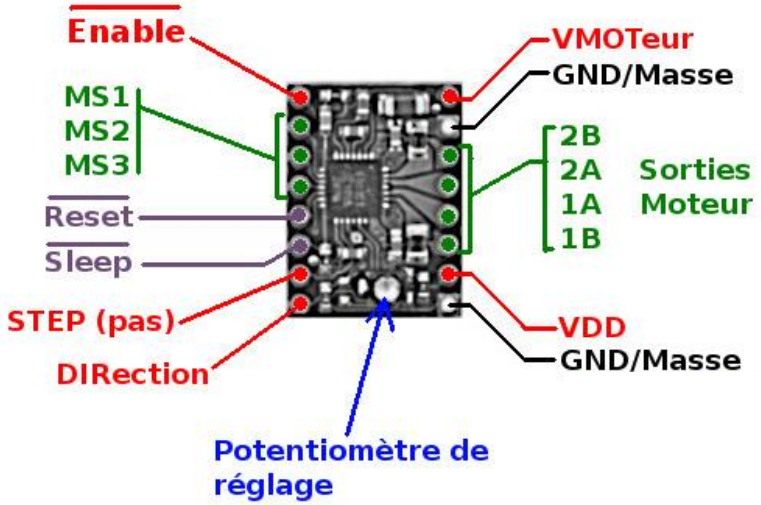
MS1	MS2	MS3	STEPS
L	L	L	FULL
H	L	L	HALF
L	H	L	QUARTER
H	H	L	EIGTH
H	H	H	SIXTEENTH



1.3. Exemple du moteur NEMA17

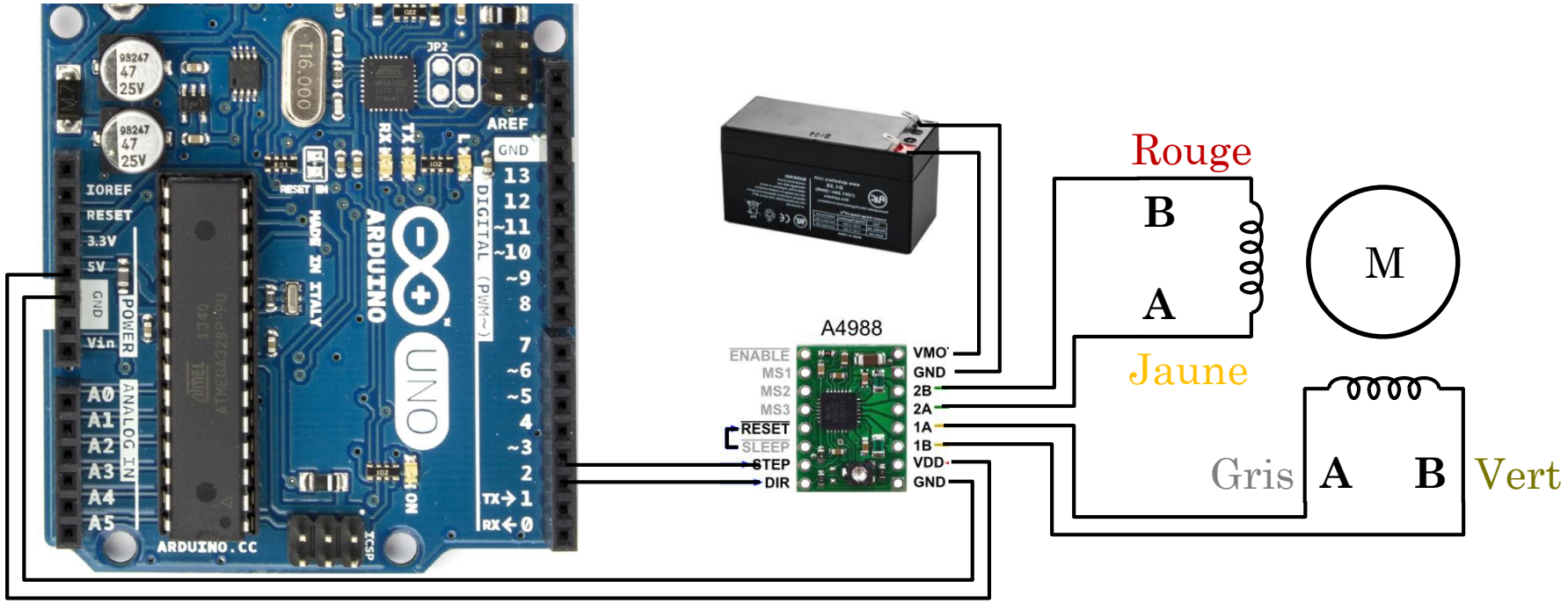
- **Mise en œuvre avec le driver A4988 : description**
  - Un potentiomètre permet d'ajuster le courant maximum qui passera dans les bobines et ainsi ne pas dépasser le courant limite du moteur
  - Dans le cas du NEMA 17, ce courant limite est de 350 mA
  - La tension entre la borne « + » du potentiomètre et la masse est donnée par : courant = 2.5 × tension. Pour courant = 0.35 A il faut obtenir tension = 0.14 V





1.3. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : montage**
  - Le montage nécessite une capacité de découplage pour l'alimentation du moteur afin de lisser les appels de courant des bobines



1.3. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : programme**
  - Avec ce programme, le moteur faire un tour complet et attend 1 s avant de recommencer

```
const int Pas = 3;
const int Dir = 2;
void setup() {
  Serial.begin(115200);
  pinMode(Pas,OUTPUT);
  pinMode(Dir,OUTPUT);
  digitalWrite(Dir,HIGH);
}
```

```
void loop() {
  for(int x = 0; x < 200; x++) {
  digitalWrite(Pas,HIGH);
  delayMicroseconds(500);
  digitalWrite(Pas,LOW);
  delay(4);
  }
delay(1000);
}
```

2.1. Présentation du lidar lite 3

- Il permet de mesurer des distances jusqu'à 40 m avec une précision de 1 cm
- Il peut effectuer jusqu'à 500 mesures par seconde (i.e. 500 Hz) et 1 kHz pour le LLV3HP
- Laser de 940 nm de classe 1
- Il est commandable par bus I^2C ou en PWM
- Il est utilisé pour les drones, la robotique générale, la détection industrielle …

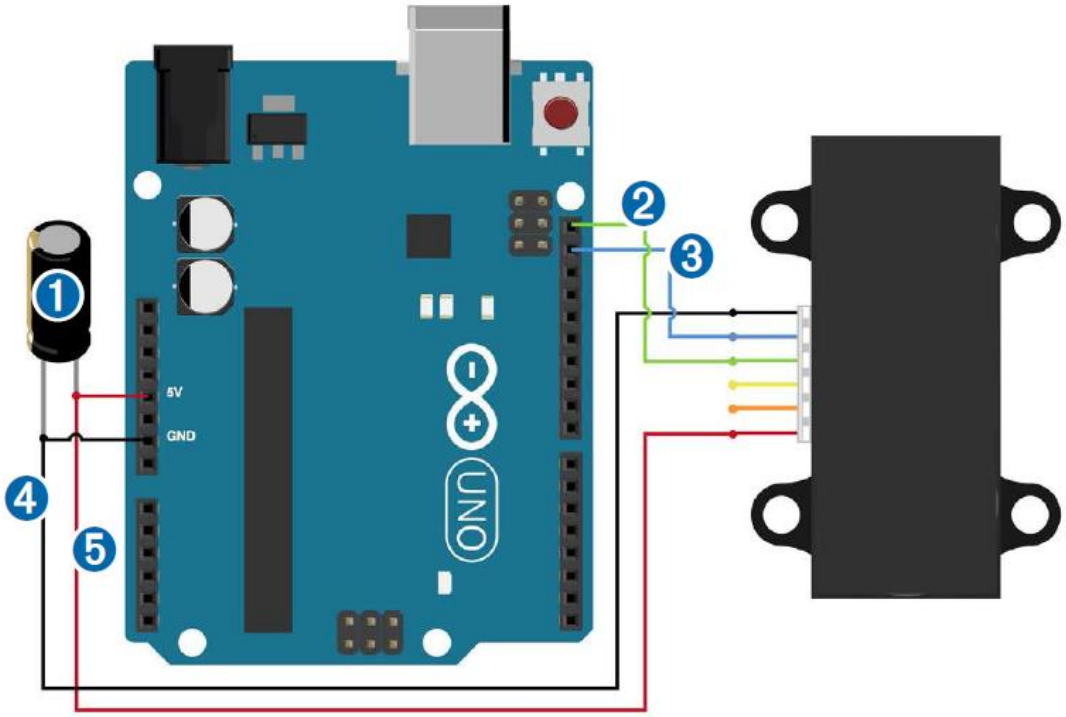
Lidar lite 3

LIDAR-Lite 3 (LLV3HP)

2.2. Mise en œuvre via I²C

Montage

- Un condensateur permet de filtrer les appels de courant du lidar (capacité de découplage)



2.2. Mise en œuvre via I²C

Librairie LIDARlite.h

- Cette librairie s'occupe de tout le protocole communication avec le lidar

```
#include <Wire.h>
#include <LIDARLite.h>
// Globals
LIDARLite lidarLite;
int cal_cnt = 0;
void setup()
{
Serial.begin(9600); // Initialize serial connection to display distance readings
lidarLite.begin(0, true); // Set configuration to default an d I2C to 400 kHz
lidarLite.configure(0); // Change this number to try out alt ernate configurations
}
```

2.2. Mise en œuvre via I²C

Librairie Lidar lite

- Toute les 100 mesures, il faut effectuer une calibration du lidar

```
void loop()
{
int dist;
// At the beginning of every 100 readings,
// take a measurement with receiver bias correction
if ( cal_cnt == 0 ) {
dist = lidarLite.distance(); // With bias correction
} else {
dist = lidarLite.distance(false); // Without bias correction
}
// Increment reading counter
cal_cnt++;
cal_cnt = cal_cnt % 100;
// Display distance
Serial.print(dist);
Serial.println(" cm");
delay(10);
}
```