COMMENT FAIRE LE CERVEAU D'UNE (PETITE) FUSÉE

AVEC THOMAS AKA PAPI ELEC

PLAN DU COURS

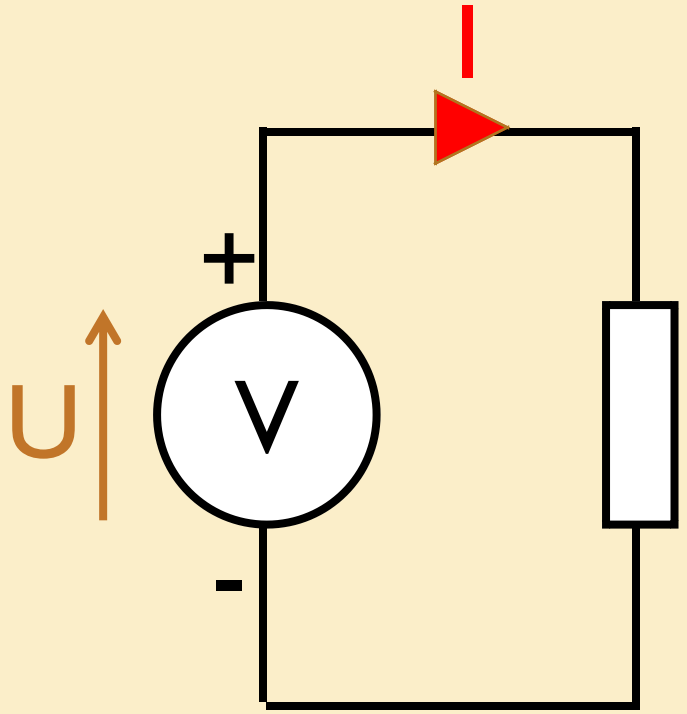
- Les bases de l'électronique
  - Les lois basiques
  - Les composants simple
  - Les composants complexes
  - Les actionneurs
  - Les micro contrôleurs
- Comment concevoir son « élec »
  - L'alimentation
  - Le séquenceur
  - L'expérience
  - L'interface

LES BASES DE L'ÉLECTRONIQUE

IL EN MANQUE UN PEU, MAIS ÇA DEVRAIT SUFFIRE

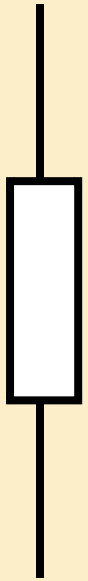
LES LOIS BASIQUES

- Tension : Différence de Potentiel
  - Mesuré en Volts (V)
  - DC (Direct Current): ~fixe
  - AC (Alternating Current): sinusoïde
- Courant : Flux d'électrons
  - Mesuré en Ampères (A)
  - Varie selon le besoin

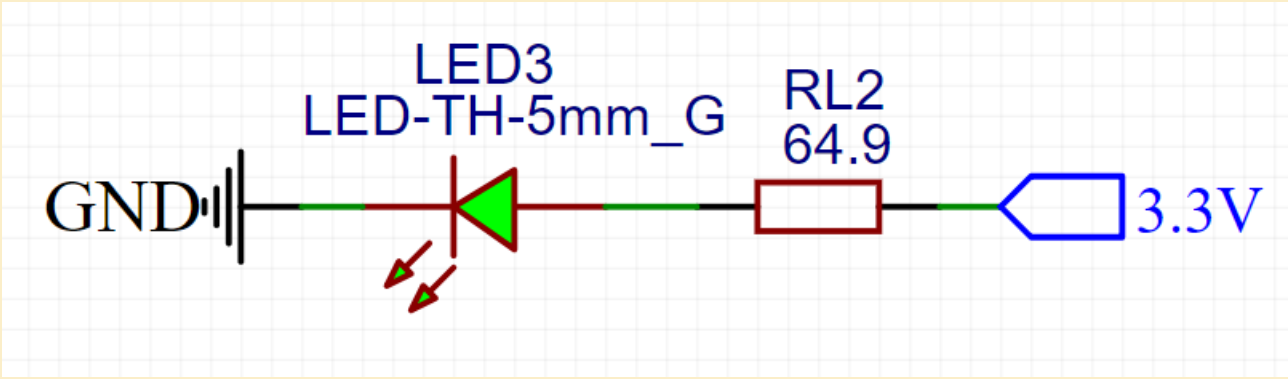


LES COMPOSANTS SIMPLES

- Resistance
  - Mesuré en Ohm (Ω)
  - U=RI ou I=U/R



- Exemple d'utilisation : LED



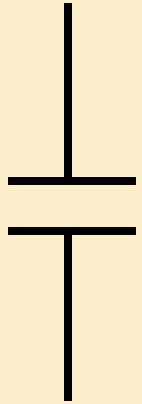
3,3V

2V ; 20mA

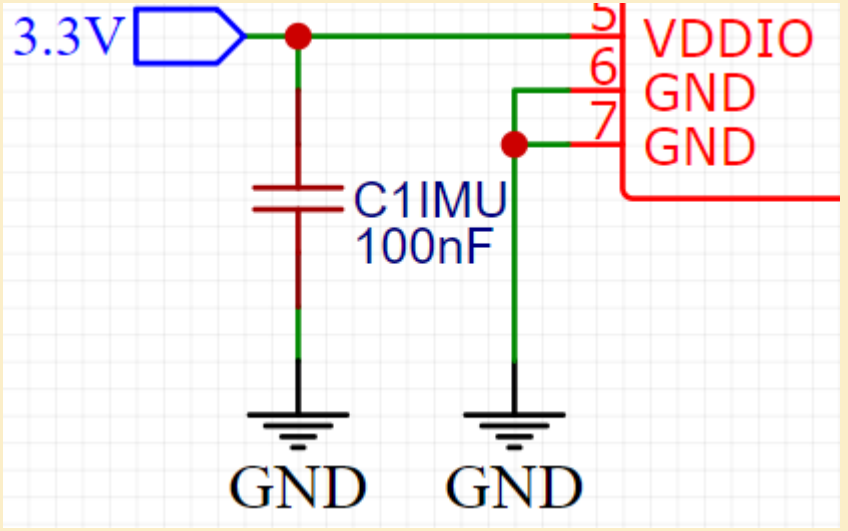
$$R = \frac{U}{I} = \frac{3{,}3V - 2V}{20mA} = 65\Omega$$

LES COMPOSANTS SIMPLES

- Condensateur
  - Mesuré en Farad (F)



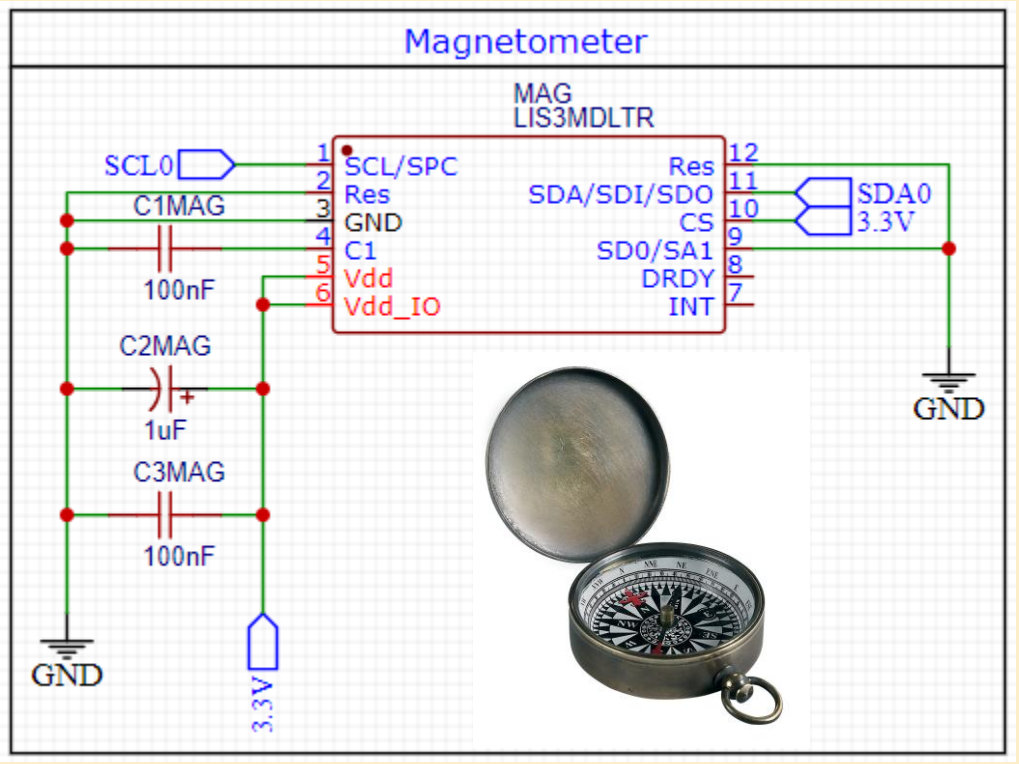
- Exemple d'utilisation : Découplage



LES COMPOSANTS COMPLEXES

- Capteurs
  - Alimenté par une tension fixe
  - Communique de différentes manière (I2C,SPI,UART,Analog,…)

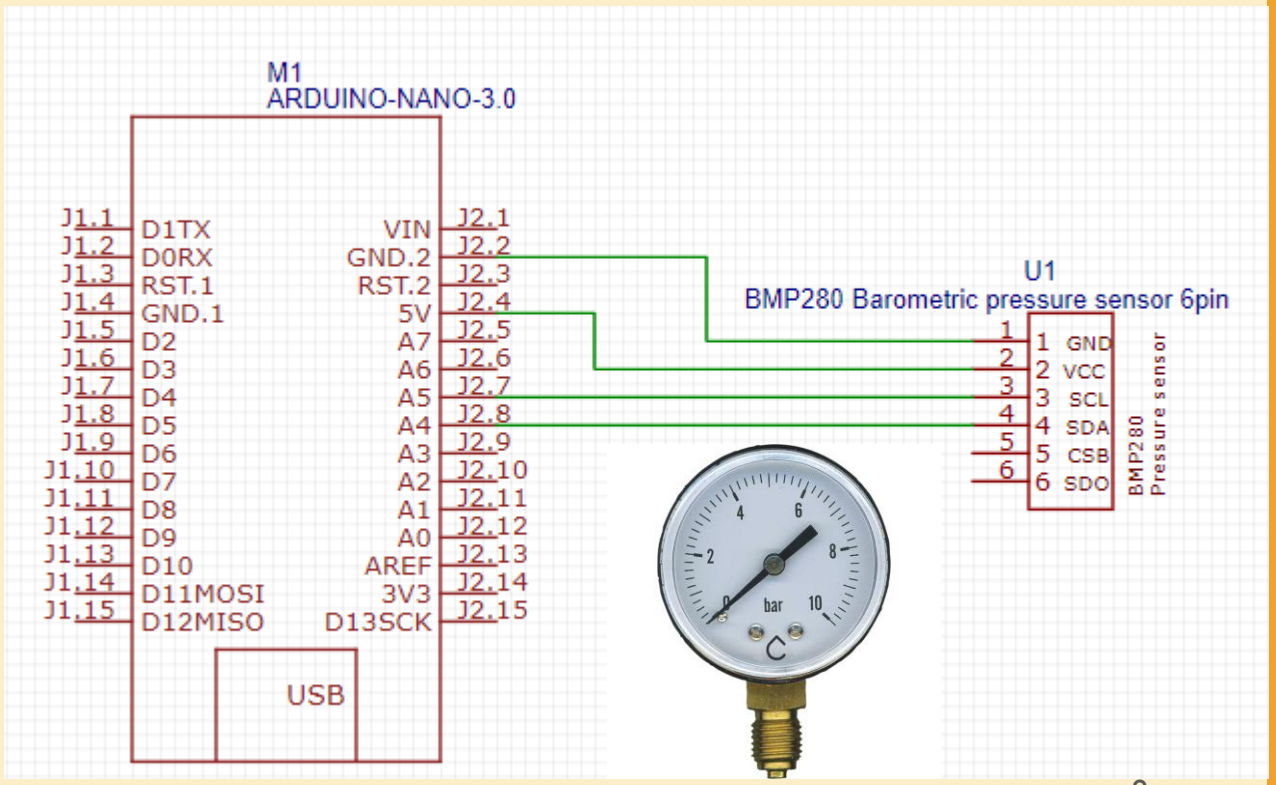
- Exemple de capteur : Magnétomètre



LES COMPOSANTS COMPLEXES

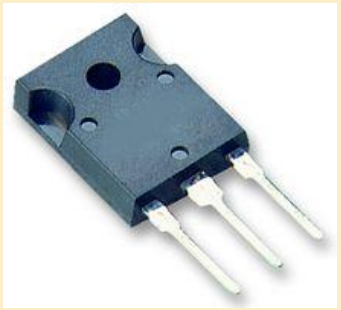
- Capteurs
  - Alimenté par une tension fixe
  - Communique de différentes manière (I2C,SPI,UART,Analog,…)

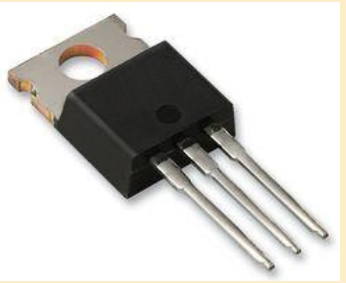
- Exemple de capteur : Capteur de Pression (breakout)

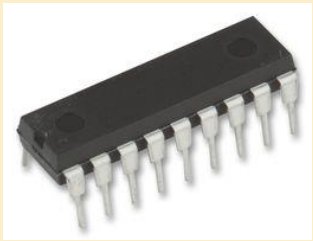


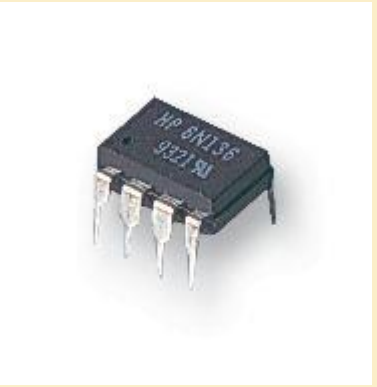
LES COMPOSANTS COMPLEXES

- Semi conducteurs simples
  - Convertir des tensions logiques (Transistor)
  - Convertir des tensions d'alimentations (Régulateur)
  - Piloter un moteur CC avec une tension logique et une tension d'alimentation (Driver)
  - Communiquer avec un système en étant isolé électriquement (Optocoupleur)
  - Générer une fréquence fixe (Quartz)









LES COMPOSANTS COMPLEXES

- Exemple de module : Driver de moteur DC

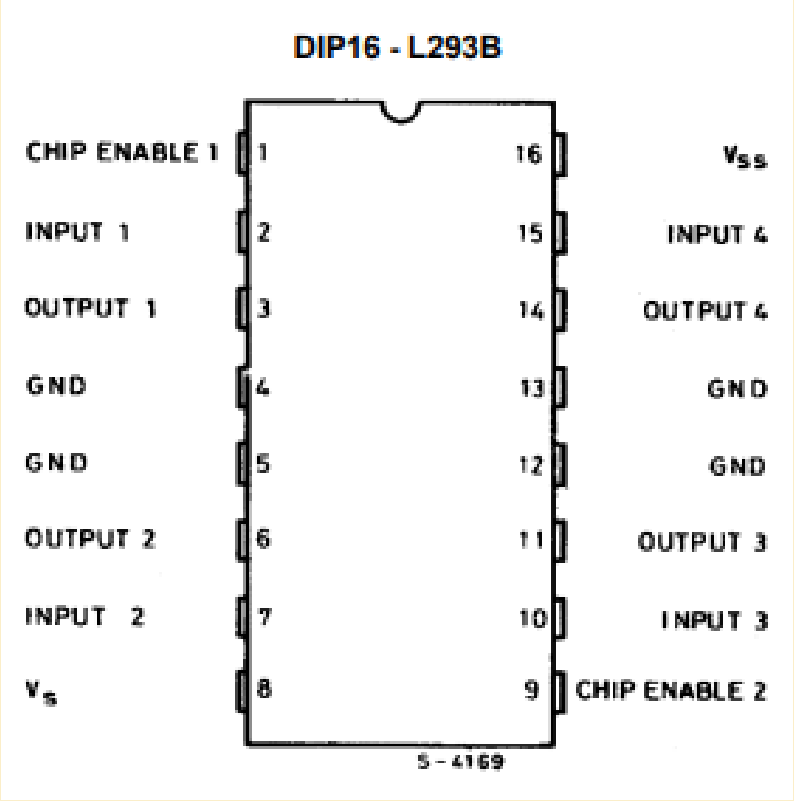
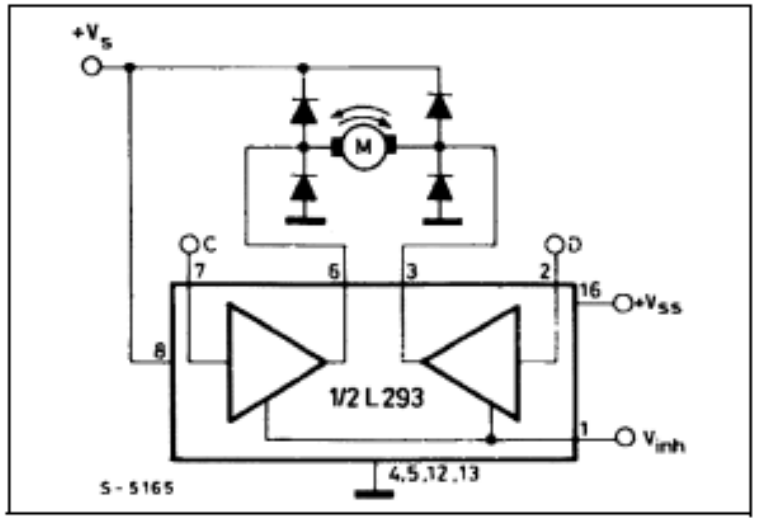


Figure 9. Bidirectional DC Motor Control



Inputs	Function	
V_{inh} = H	C = H ; D = L	Turn Right
	C = L ; D = H	Turn Left
	C = D	Fast Motor Stop
V_{inh} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't Care

L293B

Driver / Contrôleur de Moteur, Push Pull, Logique TTL, 4 Sorties, Alim. 4,5V à 36V, Sortie 1A DIP-16

Date/Lot Code

Fabricant :	STMICROELECTRONICS
Réf. Fabricant:	L293B
Code Commande :	2762683
Fiche technique:	L293B Fiche de données

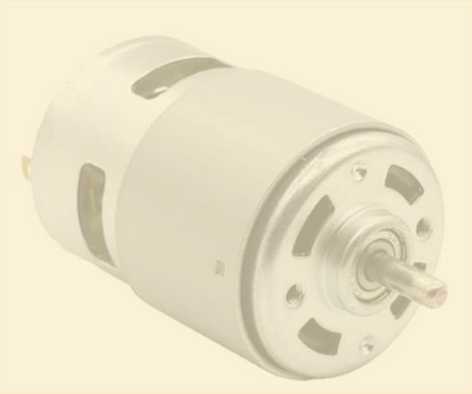
Découvrez tous les documents techniques

LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

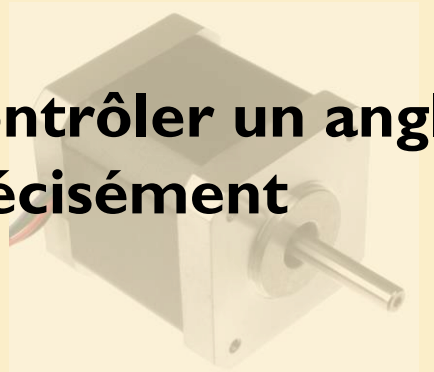
CONVERTIT UNE PUISSANCE ÉLECTRIQUE EN PUISSANCE MÉCANIQUE

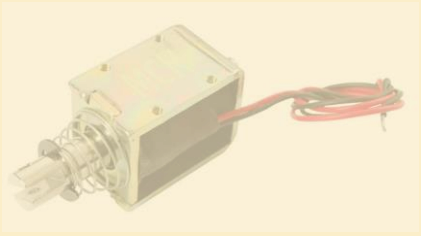
LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

- Convertit une puissance électrique en puissance mécanique
  - Servomoteur
  - Moteur DC (courant continu)
  - Moteur Brushless (courant alternatif)
  - Vérin électrique
  - Solénoïde
  - StepMotor



- **Contrôler un angle précisément**

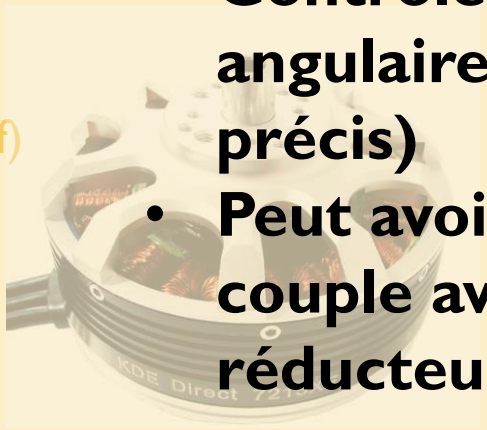


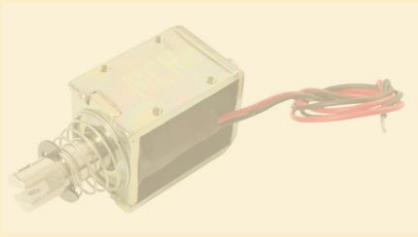


LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

- Convertit une puissance électrique en puissance mécanique
  - Servomoteur
  - Moteur DC (courant continu)
  - Moteur Brushless (courant alternatif)
  - Vérin électrique
  - Solénoïde
  - StepMotor

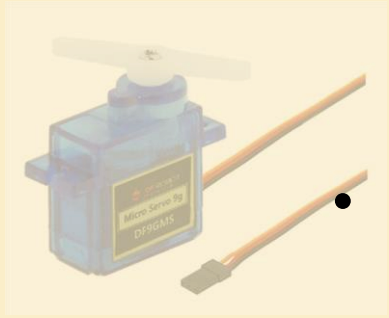
- **Contrôler une vitesse angulaire (pas très précis)**
- **Peut avoir un grand couple avec un réducteur**

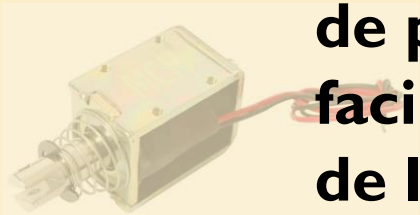




LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

- Convertit une puissance électrique en puissance mécanique
  - Servomoteur
  - Moteur DC (courant continu)
  - Moteur Brushless (courant alternatif
  - Vérin électrique
  - Solénoïde
  - StepMotor

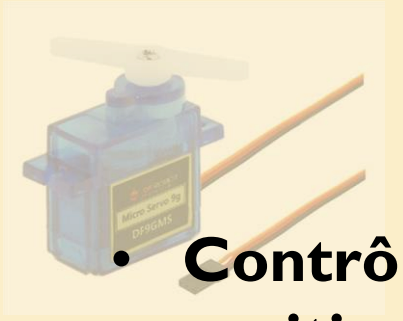




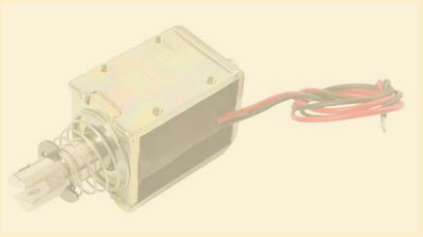
- **Contrôler une vitesse angulaire (plutôt précis)**
- **Peut avoir un couple important et une vitesse très importante**
- **Nécessite un circuit de pilotage (se trouve facilement mais prend de la place)**

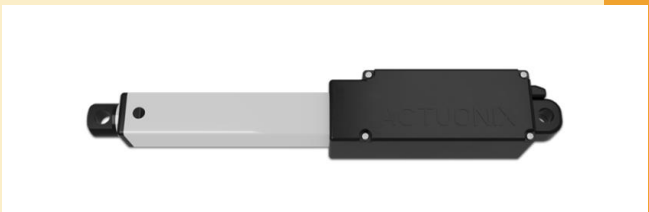
LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

- Convertit une puissance électrique en puissance mécanique
  - Servomoteur
  - Moteur DC (courant continu)
  - Moteur Brushless (courant alternatif)
  - Vérin électrique
  - Solénoïde
  - StepMotor



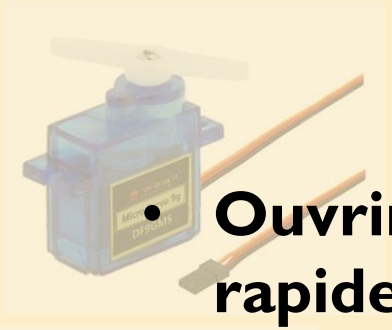
- **Contrôler une position linéaire précisément**
- **Peut avoir une grande force avec un réducteur**

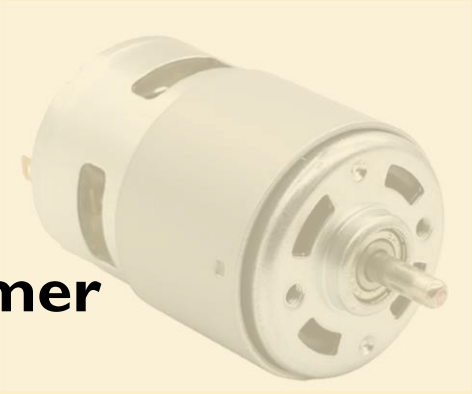




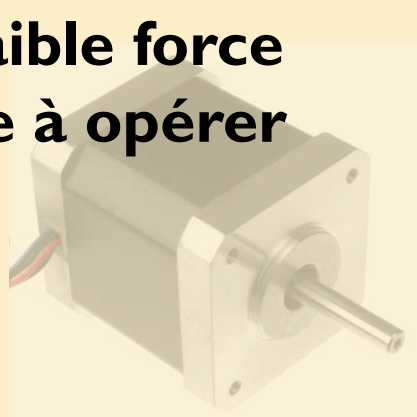
LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

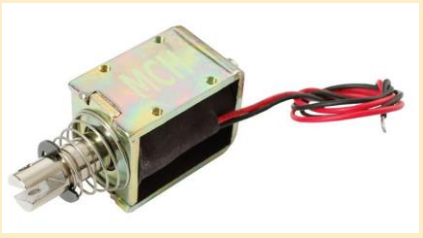
- Convertit une puissance électrique en puissance mécanique
  - Servomoteur
  - Moteur DC (courant continu)
  - Moteur Brushless (courant alternatif)
  - Vérin électrique
  - Solénoïde
  - StepMotor





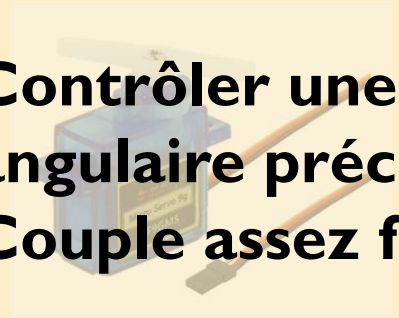
- **Ouvrir et fermer rapidement**
- **Très faible force**
- **Simple à opérer**



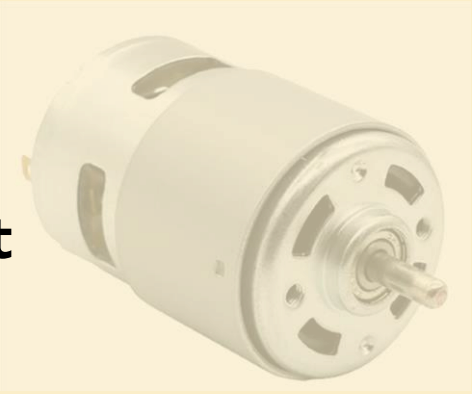


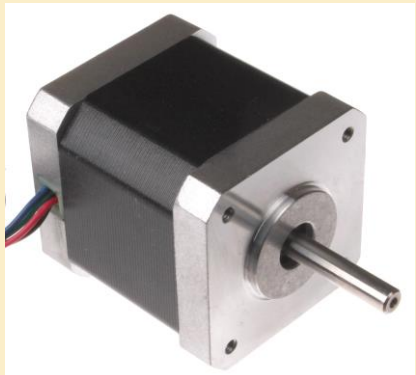
LES ACTIONNEURS (ÉLECTROMÉCANIQUES)

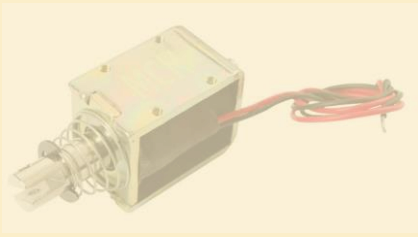
- Convertit une puissance électrique en puissance mécanique
  - Servomoteur
  - Moteur DC (courant continu)
  - Moteur Brushless (courant alternatif)
  - Vérin électrique
  - Solénoïde
  - StepMotor



- **Contrôler une vitesse angulaire précisément**
- **Couple assez faible**

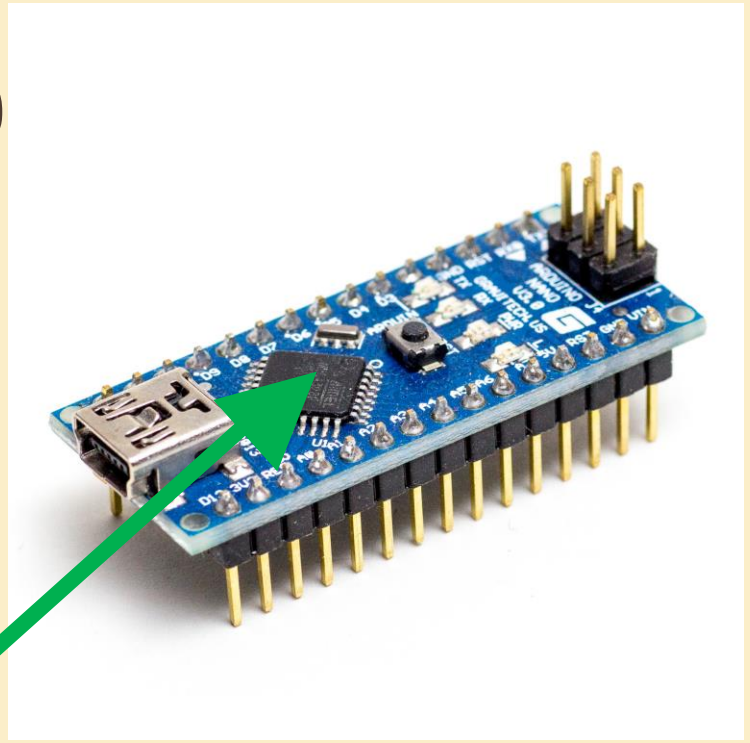


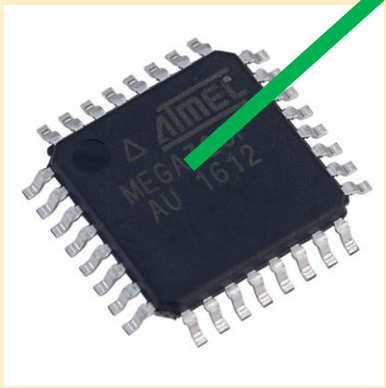


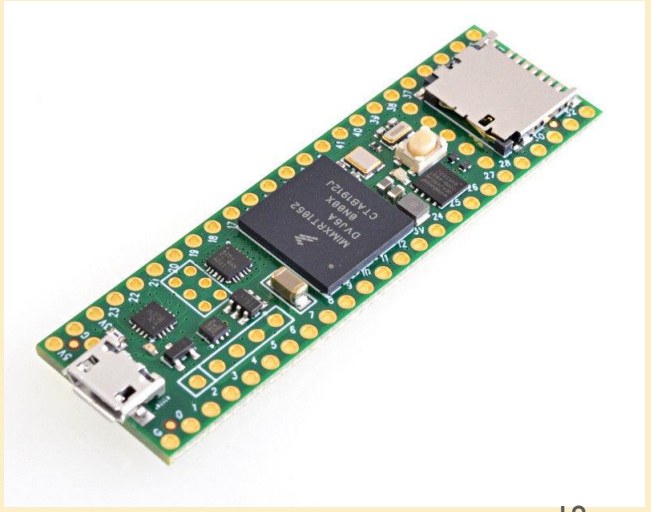


LES MICRO CONTRÔLEURS

- Composants Programmables
- Fait tourner un code unique en boucle
- Codé en C/C++ (ou python)
- Peut communiquer de plusieurs manières avec d'autres composants (I2C,SPI,UART,Digital,Analog,…)

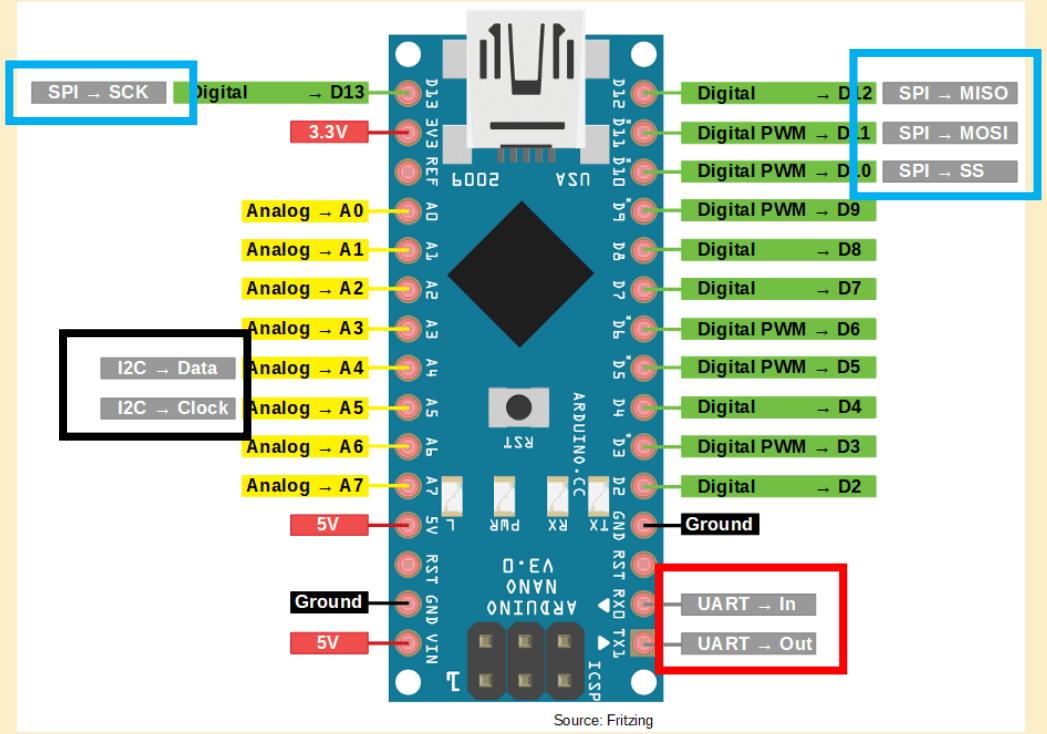






LES MICRO CONTRÔLEURS

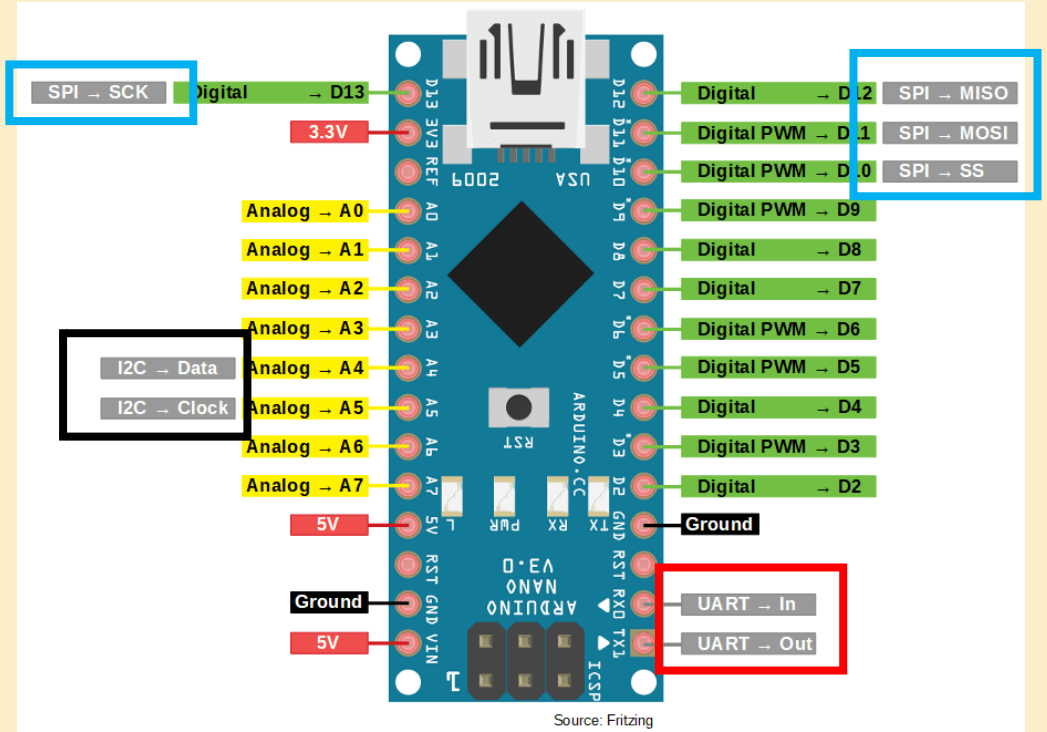
« Pinout »



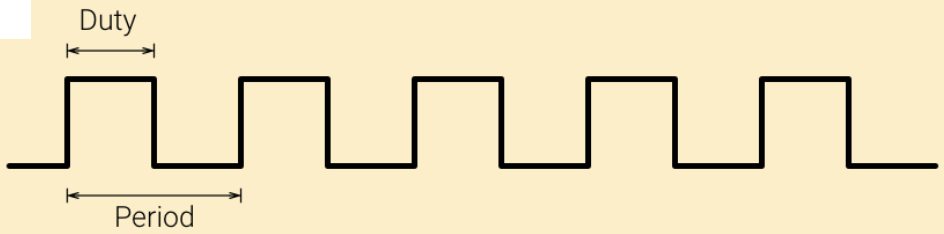
- Vin : Alimentation (5-12V, 9V recommandé)
- 5V, 3,3V : Sorties de tension fixe
- Ground : Potentiel 0

LES MICRO CONTRÔLEURS

« Pinout »

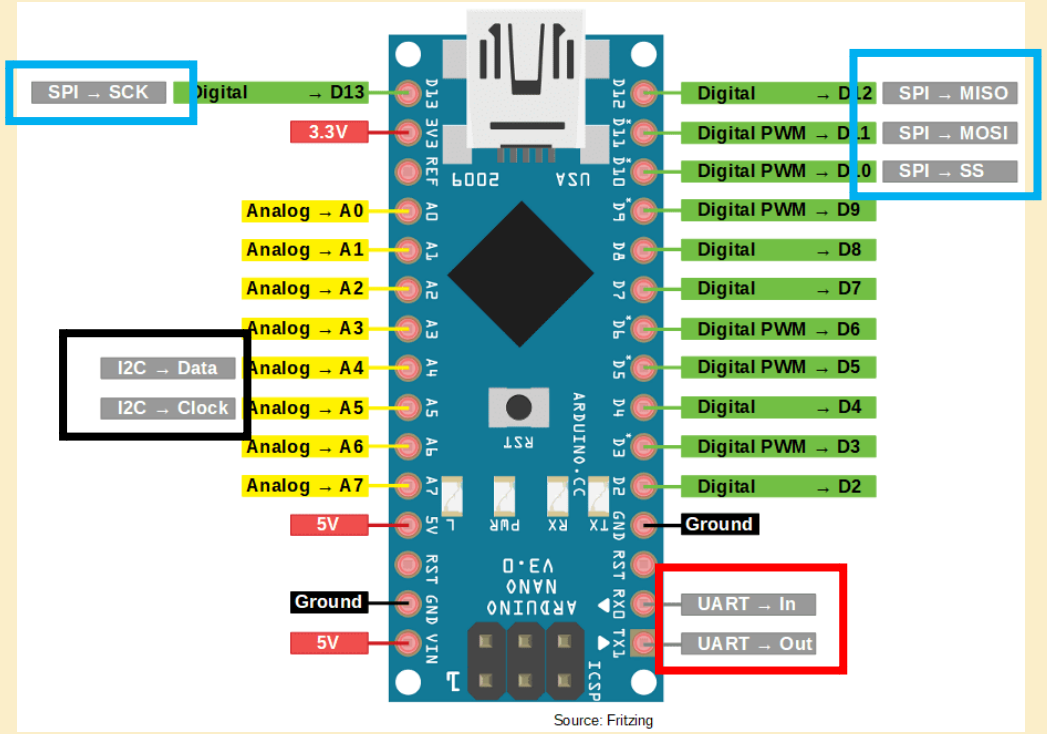


- Analog : pins analogiques
  - Capable de lire des tensions de 0 à 5V
- Digital : pins digitales
  - Capable de lire ou écrire 1 ou 0 logique (5V ou 0V)
- PWM : pins PWM
  - Pins capables de générer un signal en créneau, utile pour créer une (fausse) tension variable
  - Utile pour piloter les servomoteurs



LES MICRO CONTRÔLEURS

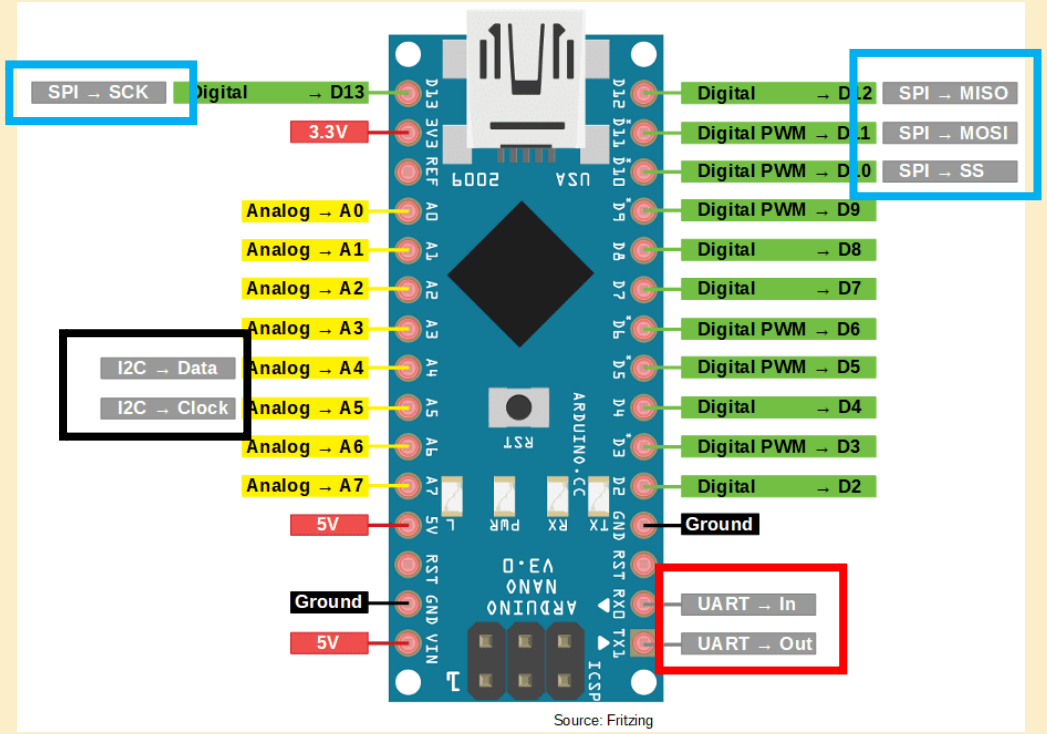
« Pinout »



- I2C :
  - communication de plusieurs modules sur une même ligne de 2 câbles
  - Flux de donné assez faible
  - Utilisé généralement pour les capteurs
- SPI :
  - communication d'un module avec 4 câbles
  - Flux de donné important
  - Utilisé généralement pour la carte SD et des capteurs avec une haute fréquence

LES MICRO CONTRÔLEURS

« Pinout »

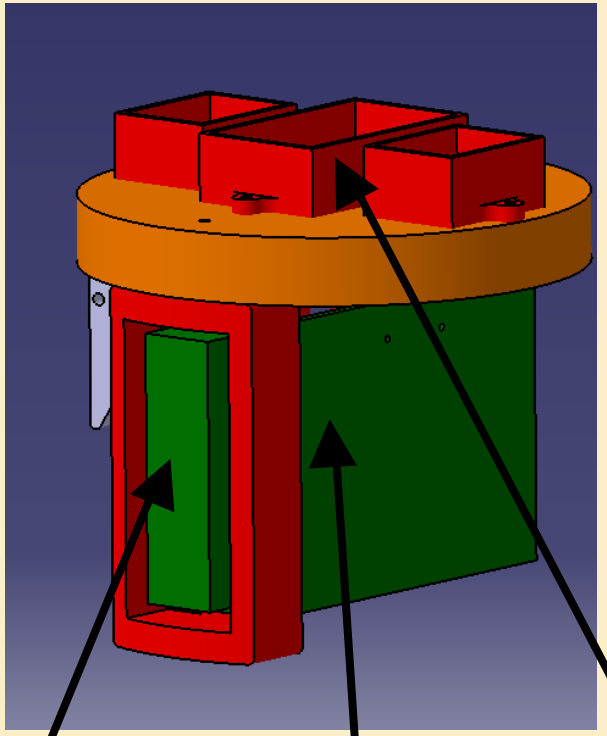


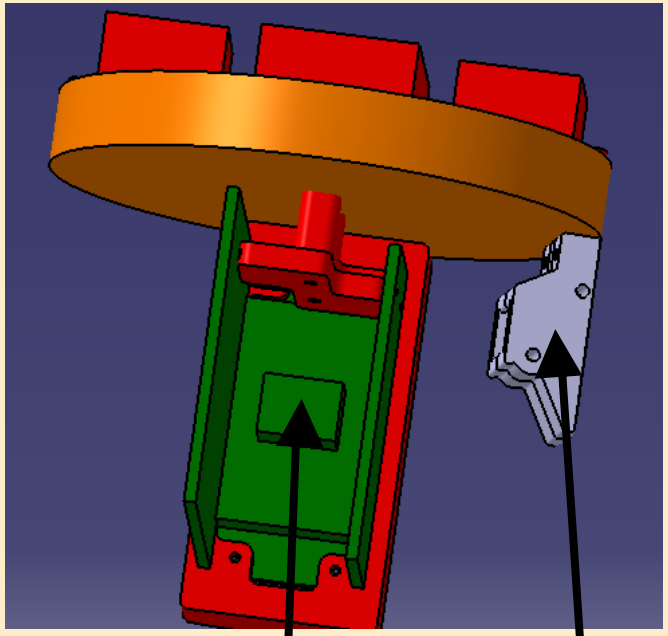
- UART:
  - Communication « en série » (Serial)
  - Pour communiquer avec un ordinateur ou d'autres microcontrôleurs
  - Flux de données important
  - Utilisé généralement pour les capteurs complexes, les GPS/GNSS et la télémétrie

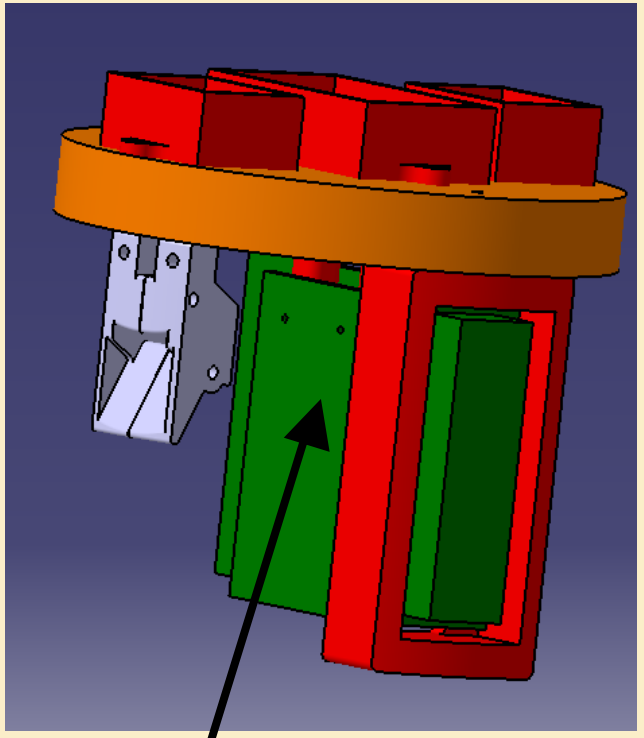
COMMENT CONCEVOIR SON « ÉLEC »

AVEC L'EXEMPLE D'AMORHC, FUSEX 2021

UNE DES FAÇONS D'ORGANISER SON ÉLEC







Interface

Expérience

Support Batterie

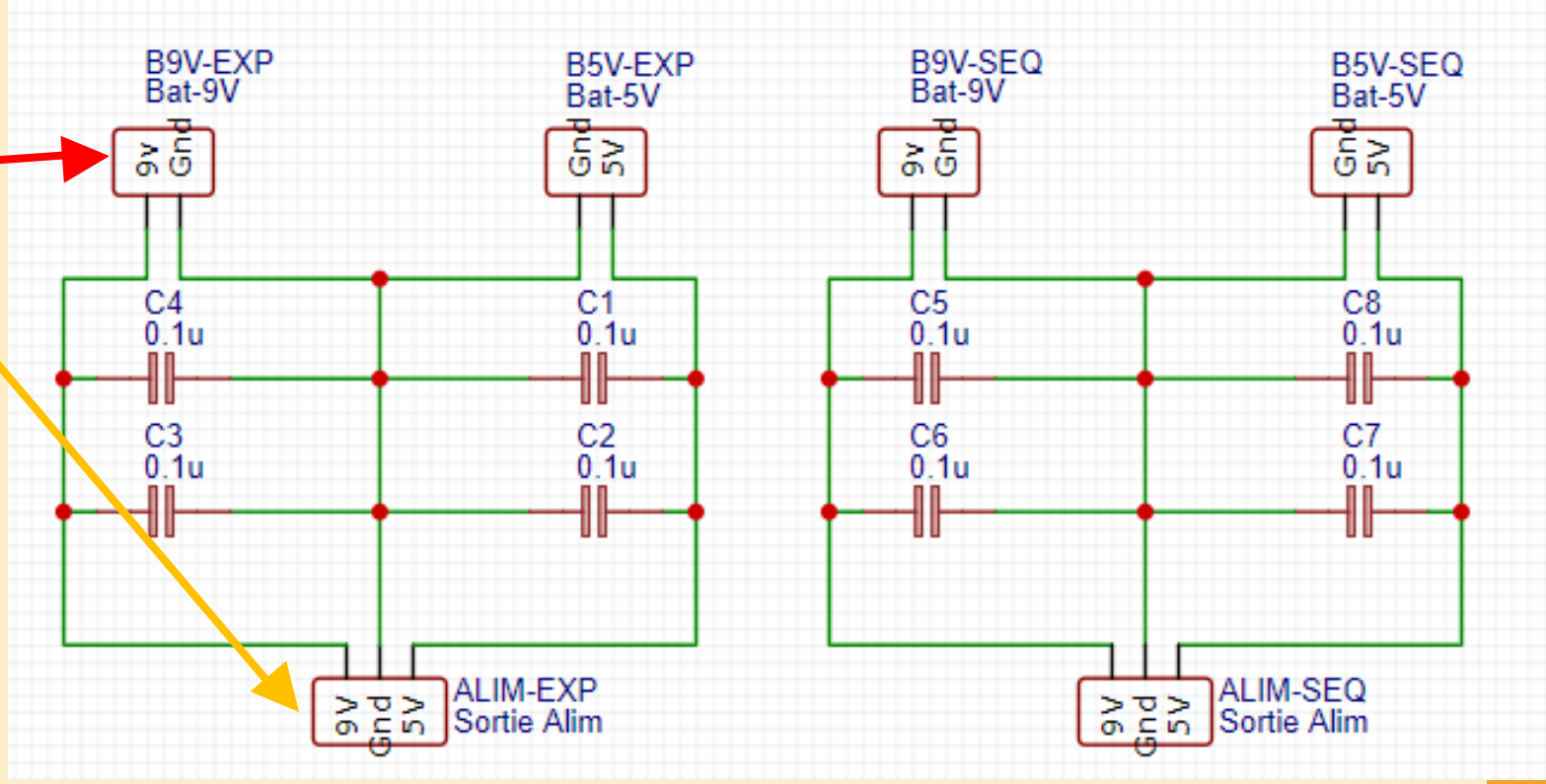
Alimentation

Support Jack

Séquenceur

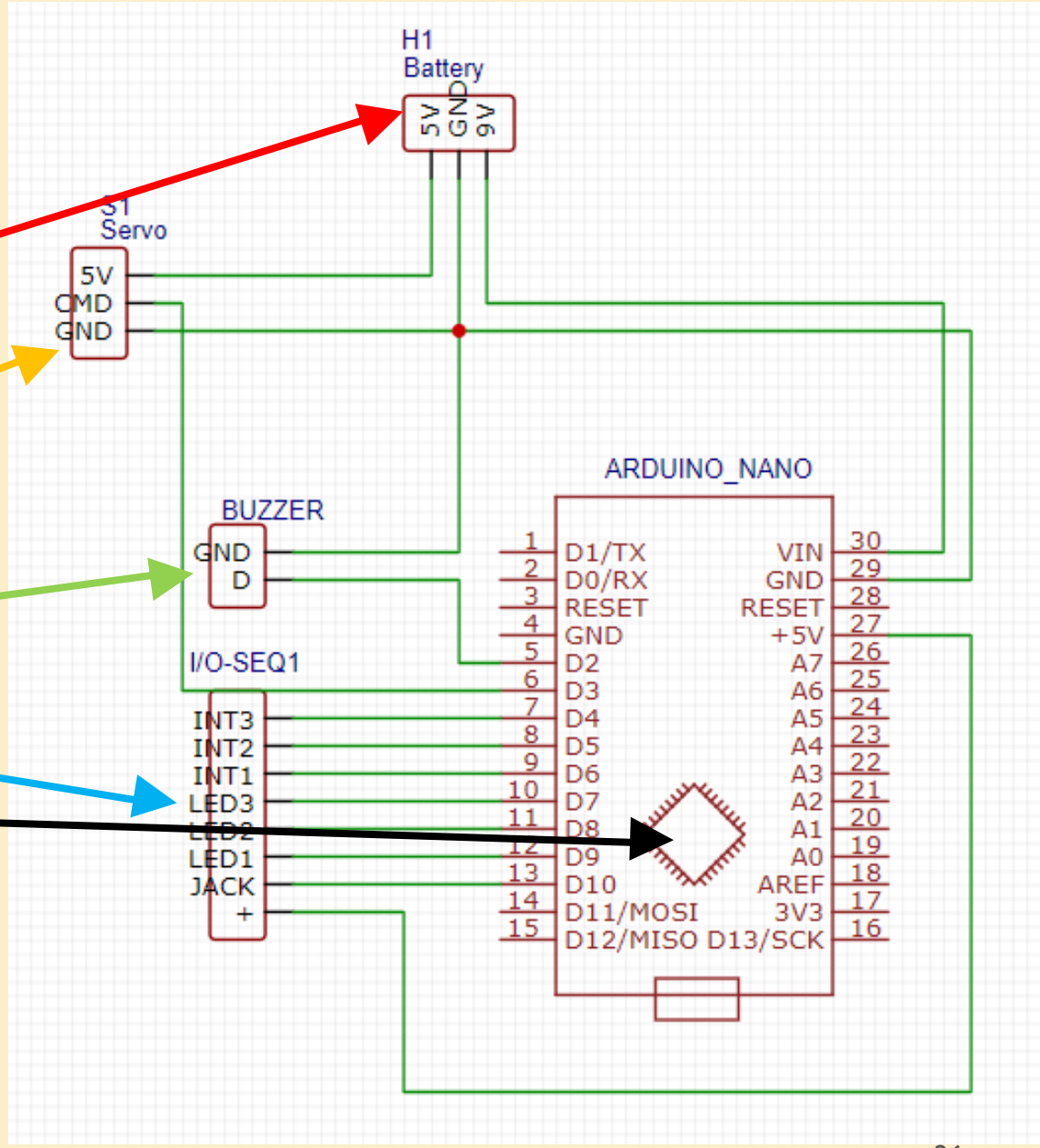
L'ALIMENTATION

- Pour alimenter les cartes électroniques et les actionneurs
- En entrée: des batteries
- En sortie: des tensions stabilisés



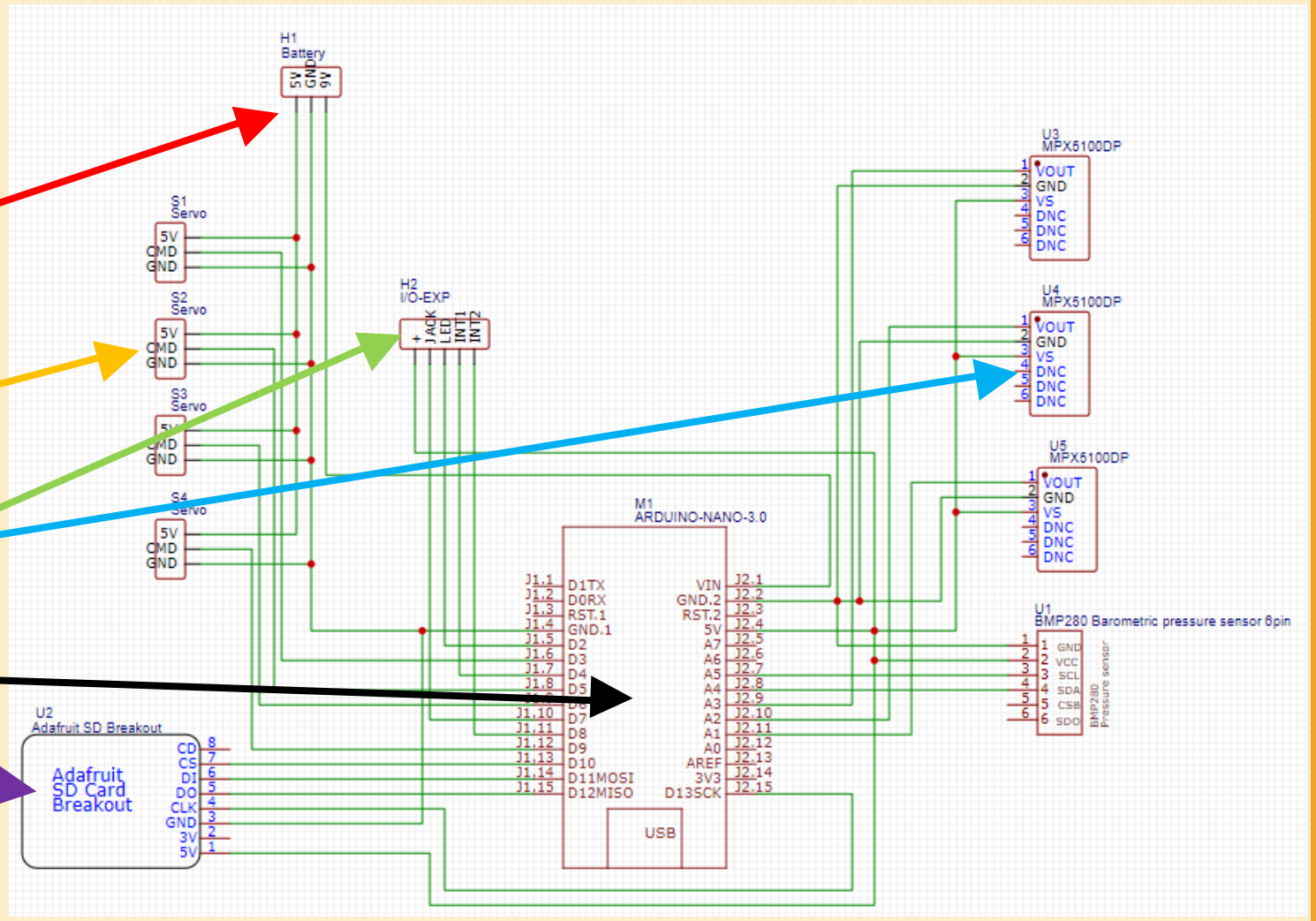
LE SÉQUENCEUR

- Pour piloter le système de libération du parachute
  - Alimentation
  - Servomoteur pour ouvrir la trappe
  - Buzzer pour repérer la fusée
  - Les entrés/sorties de l'interface
  - Le microcontrôleur



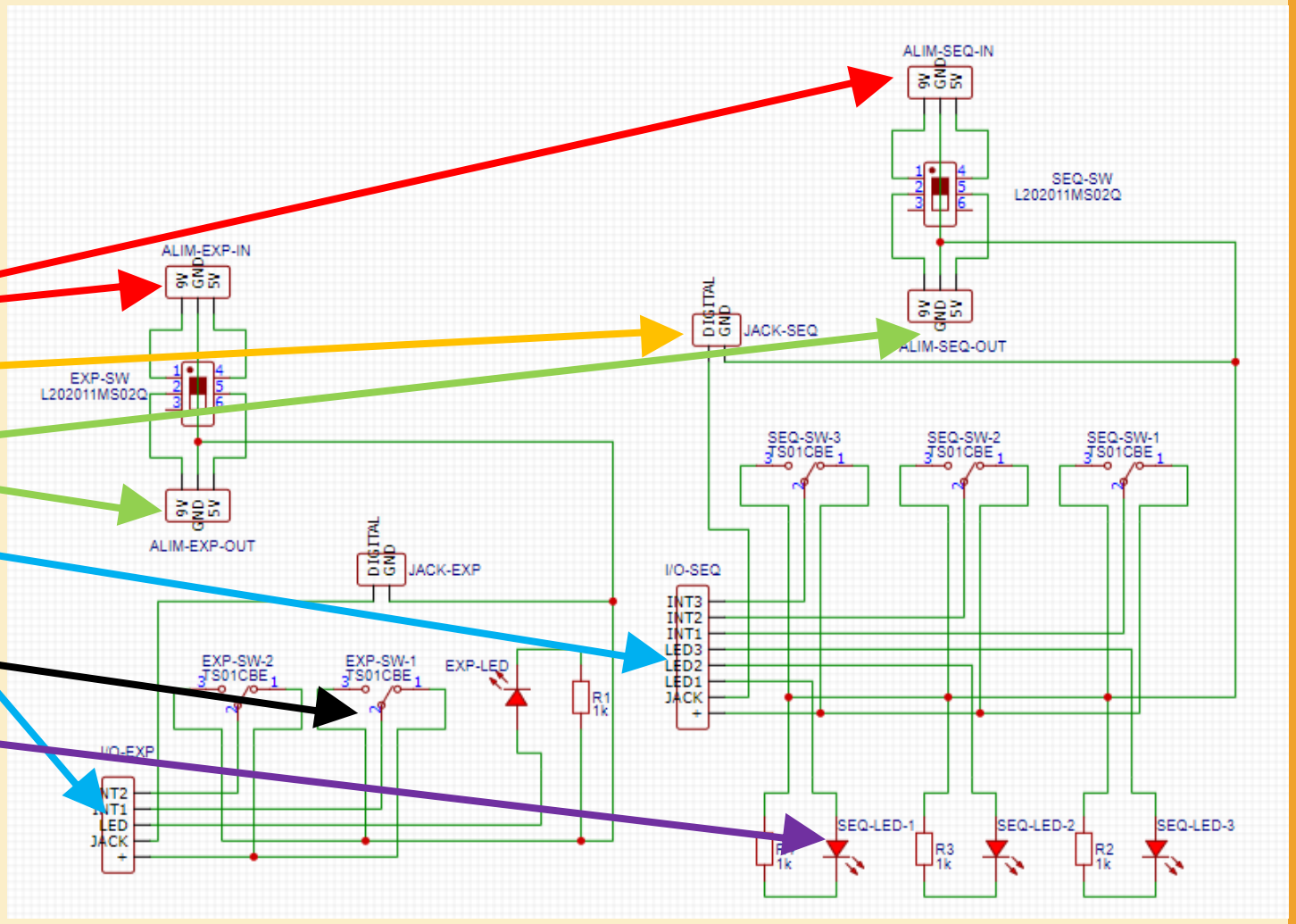
L'EXPÉRIENCE

- Pour piloter les expériences de la fusée
  - Alimentation
  - Servomoteur pour contrôler les sorties de poudre
  - Les entrés/sorties de l'interface
  - Les capteurs
  - Le microcontrôleur
  - Le module SD



L'INTERFACE

- Pour que l'on puisse interagir avec la fusée
  - Sorties de l'alimentation
  - Jack
  - Alimentation des cartes
  - Les entrés/sorties des cartes
  - Interrupteurs
  - LED



DERNIER CONSEIL : LISEZ LES DATASHEET !

PAS ***TOUTE*** LA DATASHEET, JUSTE LES INFOS QUI VOUS IMPORTENT

LE LANGAGE ARDUINO

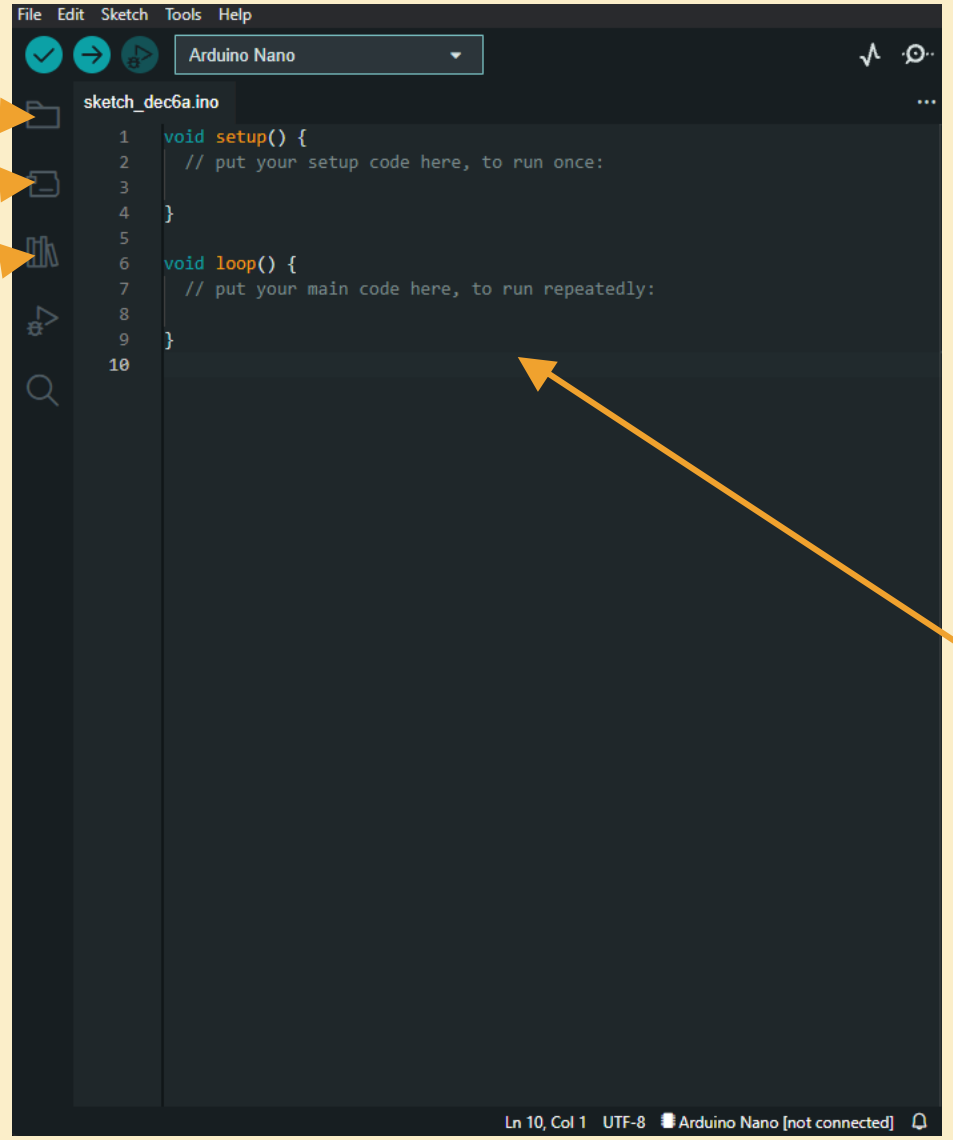
C MAGIQUE

L'INTERFACE

SKETCHBOARD

BOARDS MANAGER

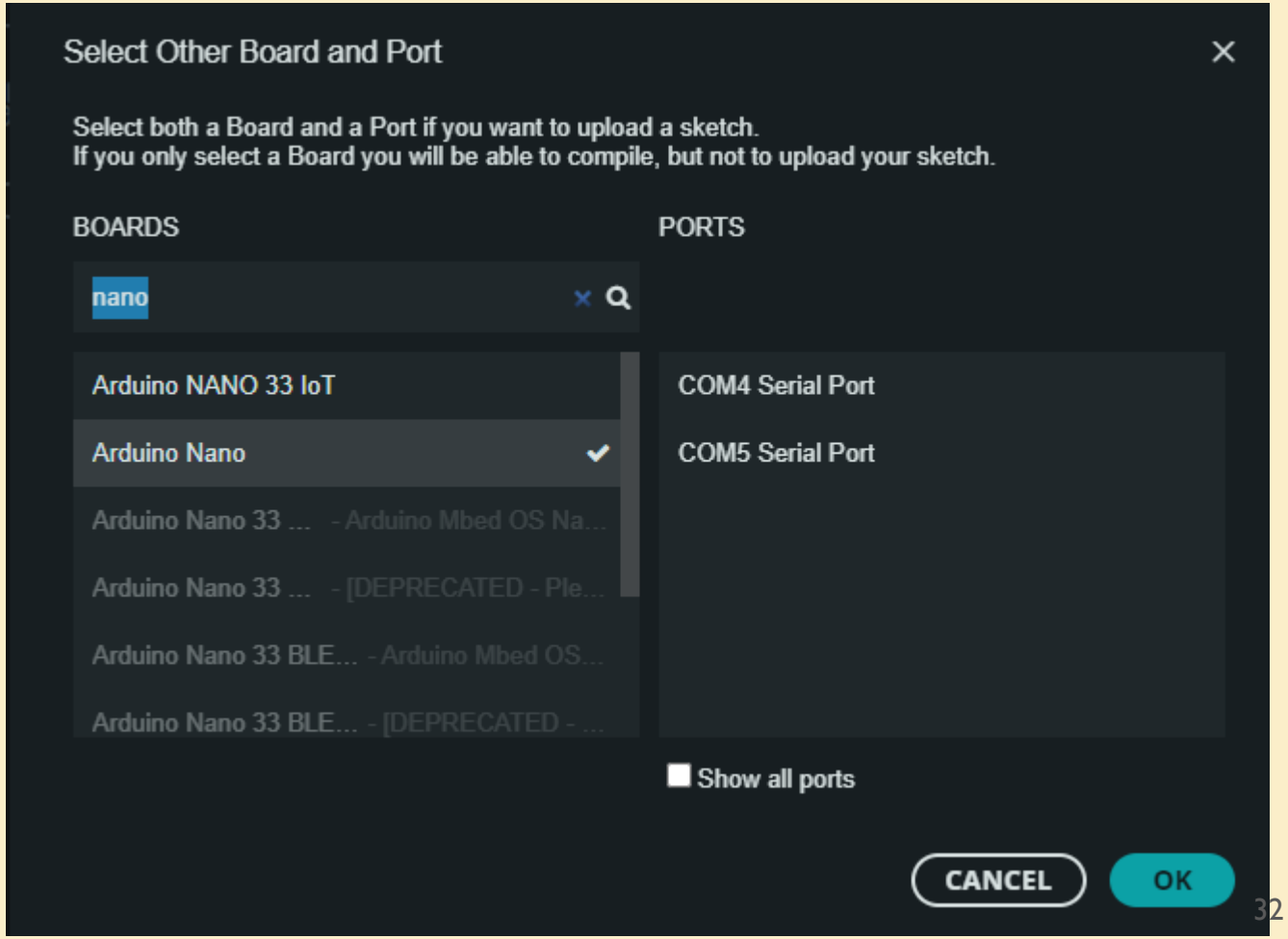
LIBRARY MANAGER



ZONE DE CODE

SÉLECTION DE LA CARTE

- Sélectionner la bonne carte et le bon port



SYNTAXE DE BASE

- Deux fonctions obligatoires :
- L'Arduino lit les instructions de haut en bas
- Chaque instruction DOIT être terminée par un « ; »
- Les commentaires : //blabla
- Ou : /* blabla blabla blabla */

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

SYNTAXE DE BASE

- Déclaration de variable avant les deux fonctions (tout en haut)
- Différents types de variable :
  - char
  - int
  - long
  - float
  - double
  - …

```
int a;

void setup() {
  // put your setup code here, to run once:

}
```

SYNTAXE DE BASE

- Opérateurs arithmétiques :
  - =
  - +
  - -
  - *
  - /
  - %

```
int a = 6;
int b = 8;
int c = a - b;
int d = a + b;
int e = a*b;
```

LES CONDITIONS ET BOUCLES

- If
- Boucle for
- Boucle while

- Opérateur de comparaison
  - ==
  - !=
  - <
  - >
  - <=
  - >=

FONCTION D'ARDUINO

- Entrées et Sorties

```
void setup() {
  // put your setup code here, to run once:
  pinMode(/*Numero de la pin*/, INPUT);
  pinMode(/*Numero de la pin*/, OUTPUT);
}
```

- Lecture et écriture

```
void setup() {
  // put your setup code here, to run once:
  digitalRead(/*Numero de la pin*/);
  digitalWrite(/*Numero de la pin*/, LOW); /*Passe la sortie à 0V*/
  digitalWrite(/*Numero de la pin*/, HIGH); /*Passe la sortie à 5V*/
```

FONCTIONS D'ARDUINO

- delay

```
delay(/*Nombre de millisecondes*/)
```

- Bibliothèque

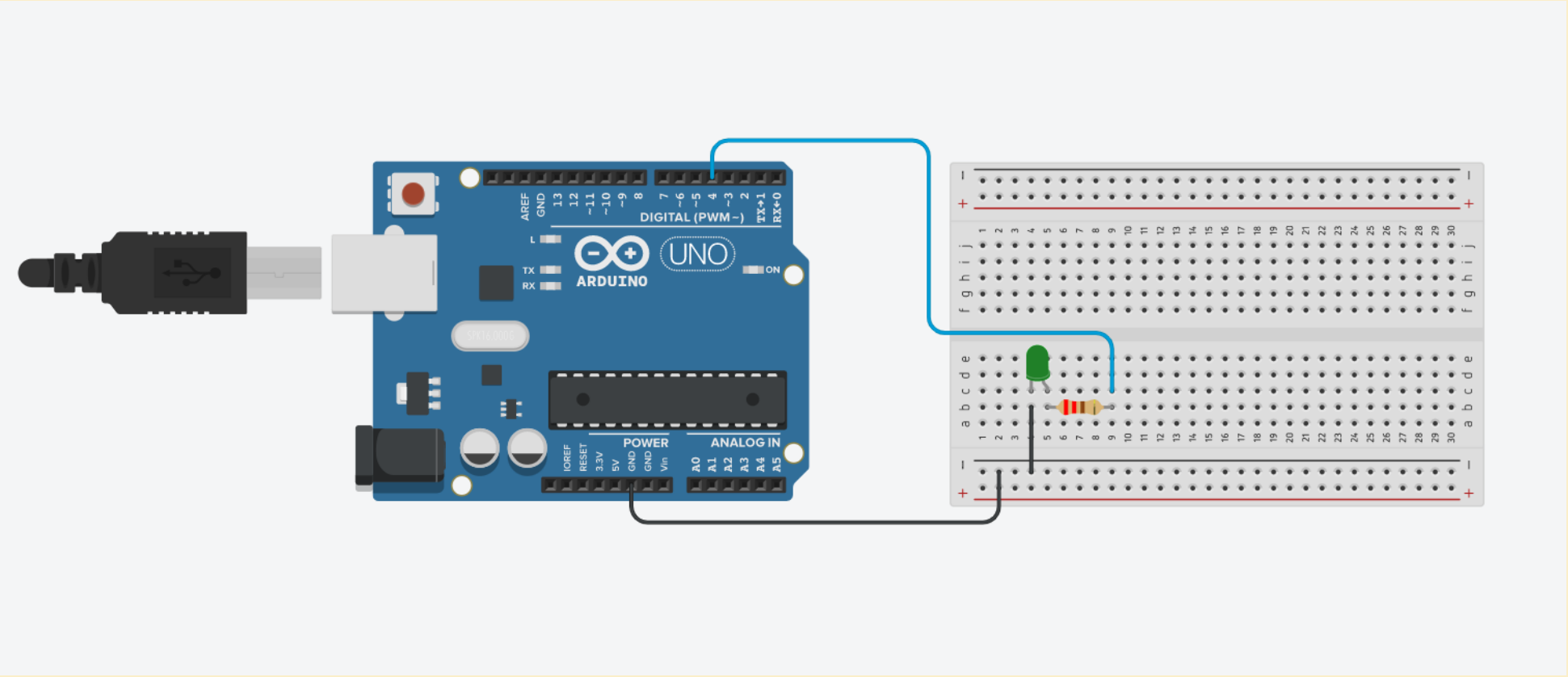
```
#include<Servo.h>

Servo servo;

void setup() {
  servo.attach(/*Numero de la pin*/)}
```

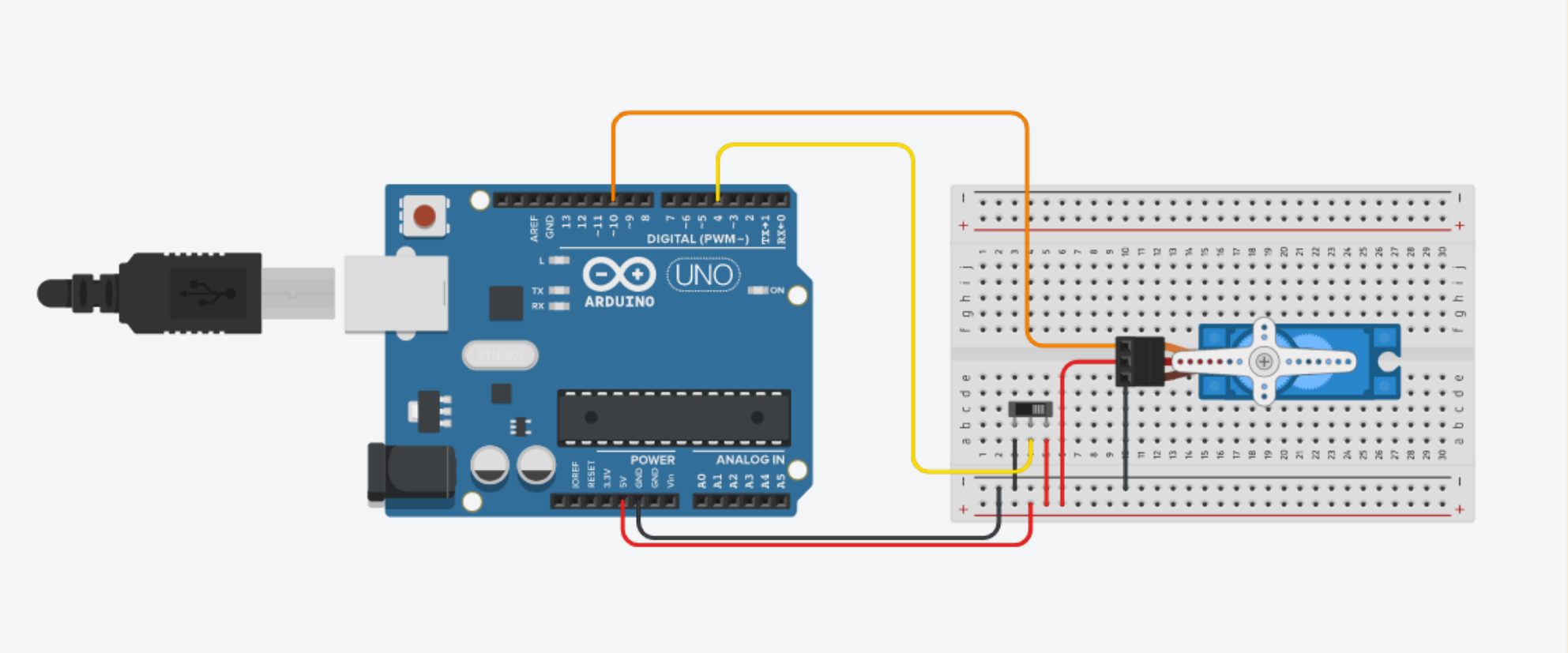
MAINTENANT LE TP

- Faire clignoter une LED toutes les seconde,
- Attention à ne pas oublier la résistance ;)



MAINTENANT LE TP

- Faire tourner in servomoteur de 90° quand l'interrupteur est fermé et rester à la position initial ouvert.



MAINTENANT, LE TP !

- Faire tourner in servomoteur de 90° et laisser la LED allumée quand l'interrupteur est fermé et rester à la position initial et faire clignoter la LED toutes les 250 ms lorsqu'il est ouvert.