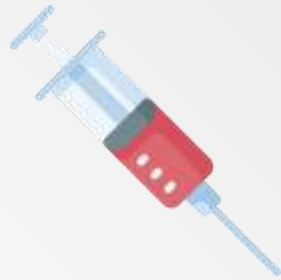


# Seringue Auto-Pousseuse (SAP)



ILIAS RAFIK

SCEI : 13024






# LE PLAN DE LA PRESENTATION

 Introduction

 Cahier des charges et le fonctionnement global,

 Les objectifs

- ✓ Commande de perfusion précis et positionnement du piston,
- ✓ Alimentation des différents composants du système,
- ✓ Contrôler la rotation du moteur en fonction des instructions reçues par l'électronique de puissance,
- ✓ piloter l'injection à distance

 Conclusion



# Présentation **du système**

Les pous­ses seringues élec­triques sont des appa­reils effi­caces pour une perfu­sion des pro­duits mé­dicame­nteux injectable, de façon pré­cise et ré­gu­lière, sur une longue pé­riode, avec un fonc­tion­ne­ment op­ti­mal grâce à l'in­tel­ligence de ces der­niers;

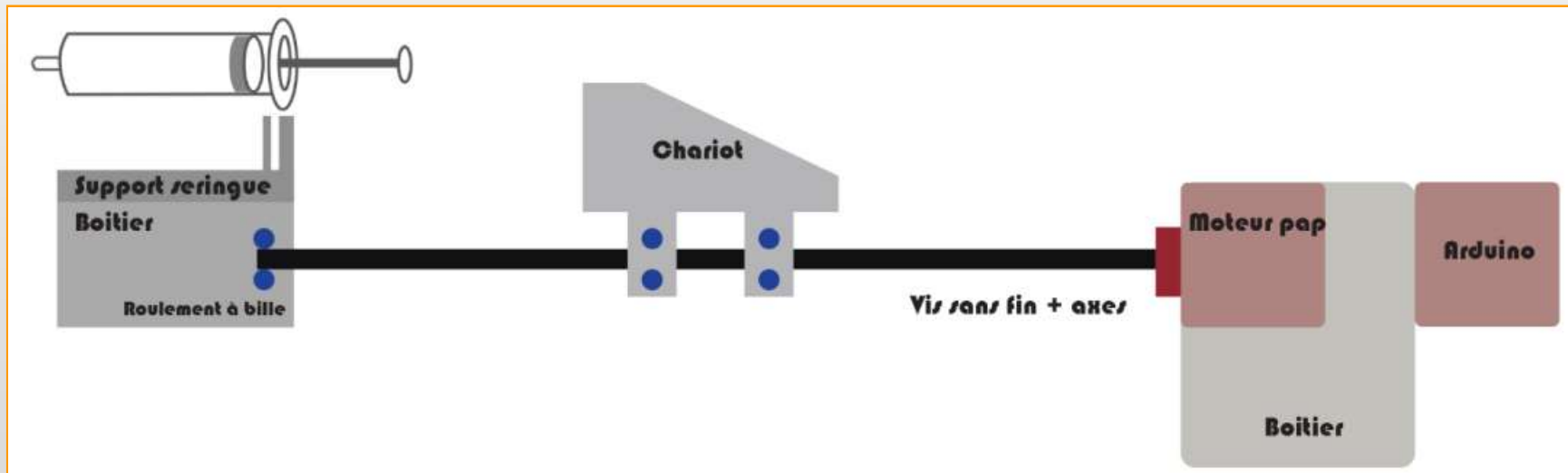


Figure 1 : Illustration du fonctionnement

## Problématique retenue :

Comment peut-on assurer une perfusion précise des solutés ou des médicaments d'une manière autonome ?

## Schéma synoptique du système :

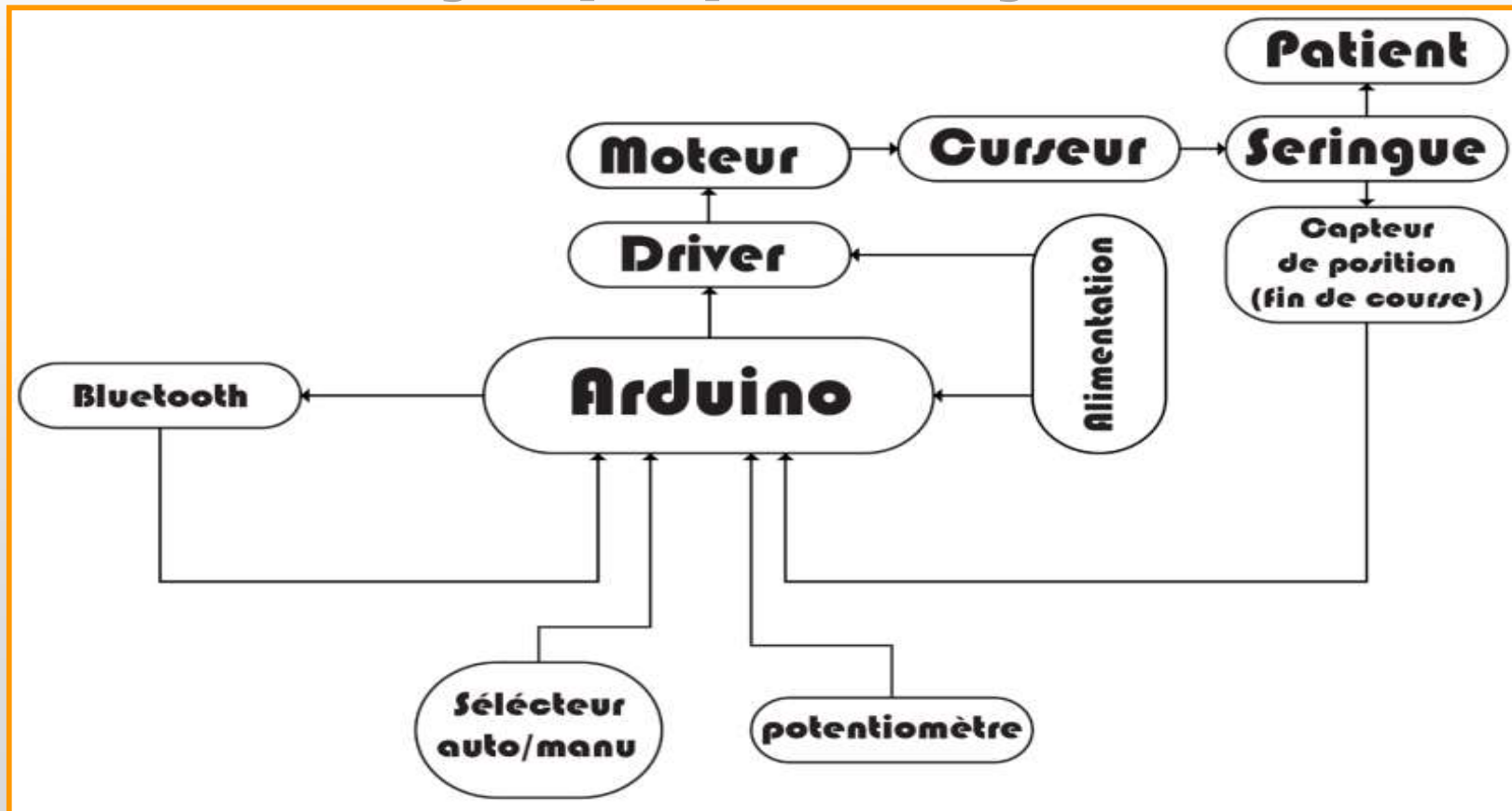


Figure 2 :Schéma synoptique



# Cahier des **charges**

## Diagramme de cas d'utilisation (UC)

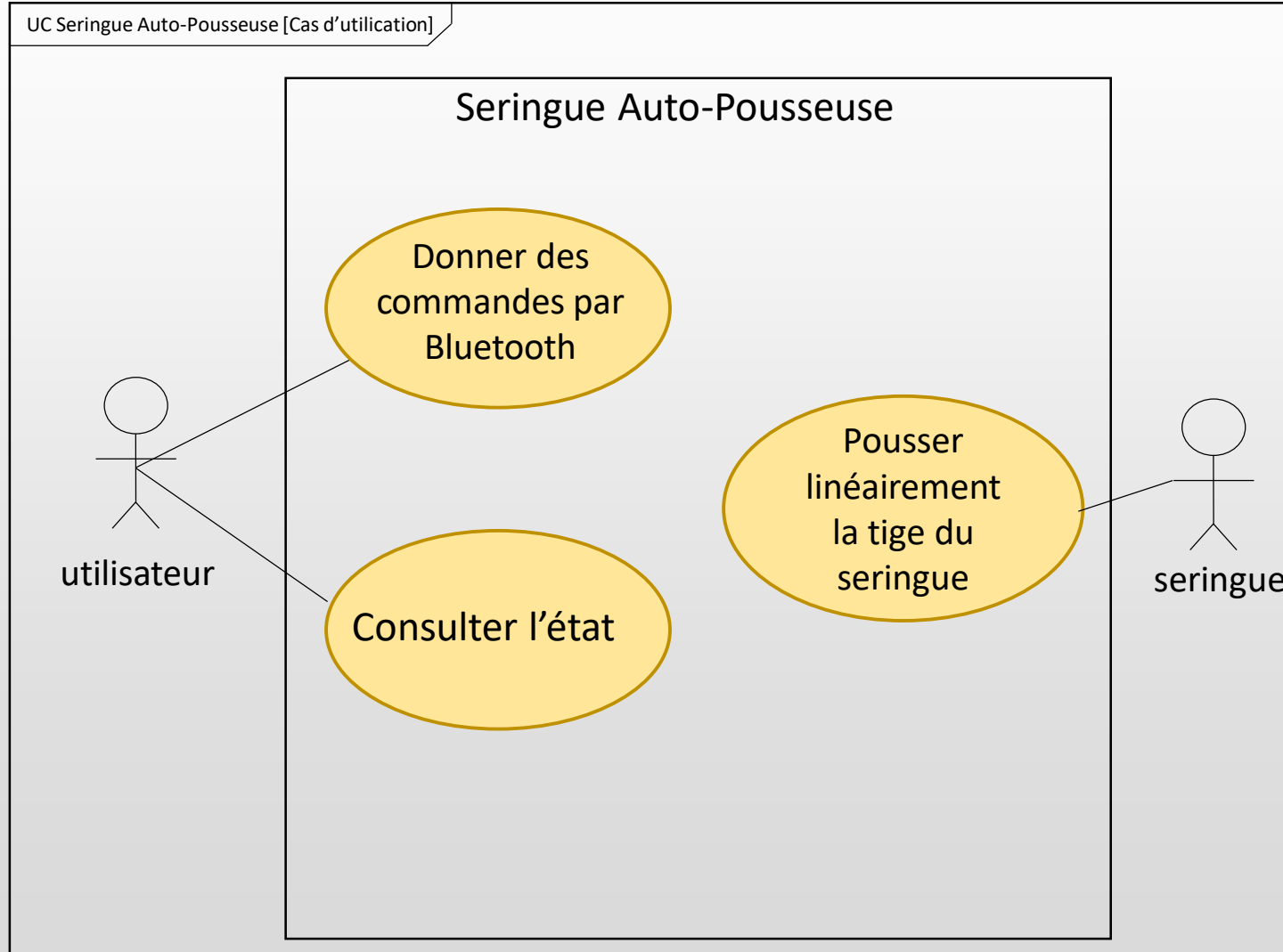


Figure 3 : Diagramme de cas d'utilisation du Seringue Auto-Pousseuse



# Cahier des charges

## Diagramme d'exigences (REQ)

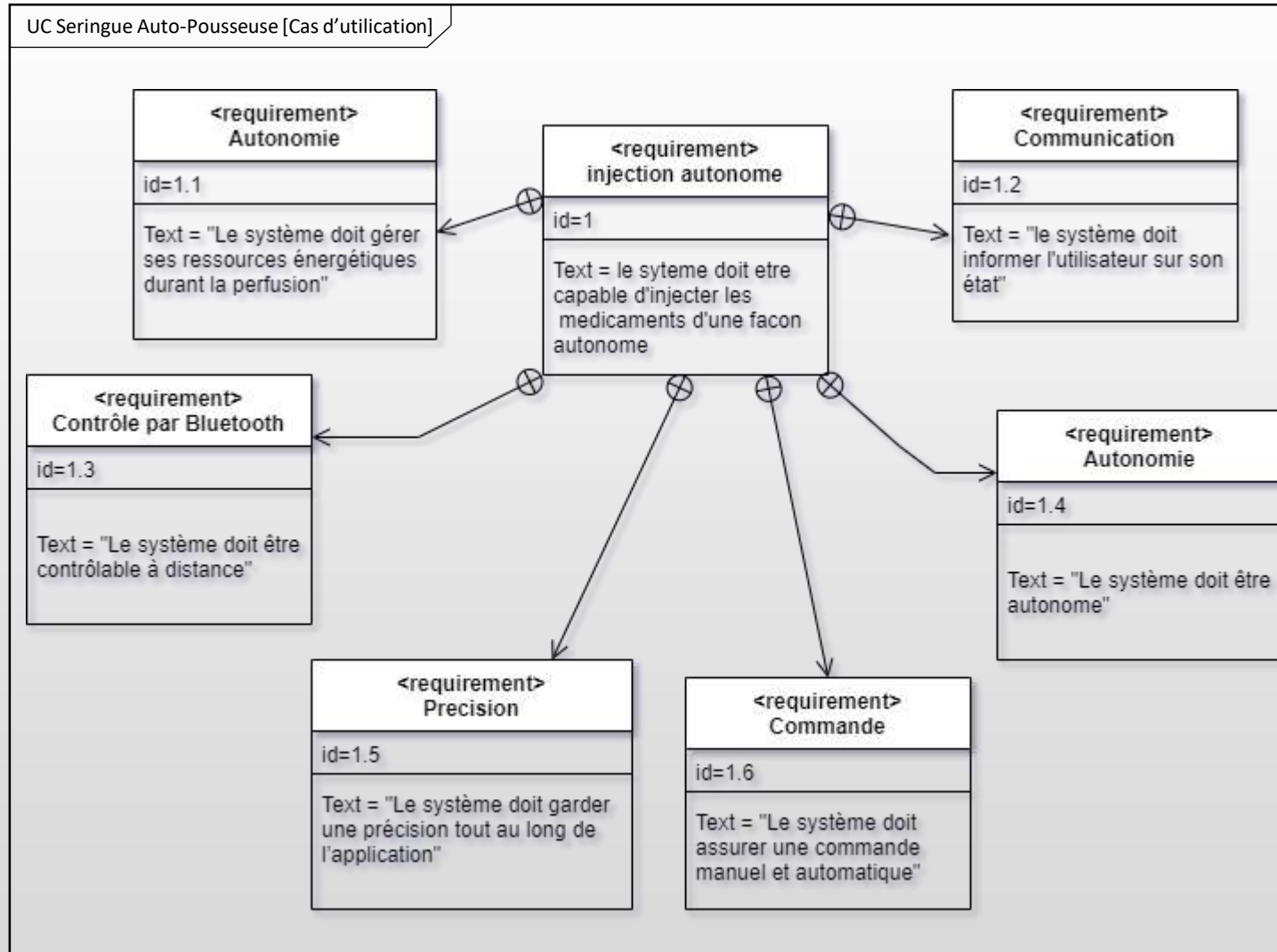
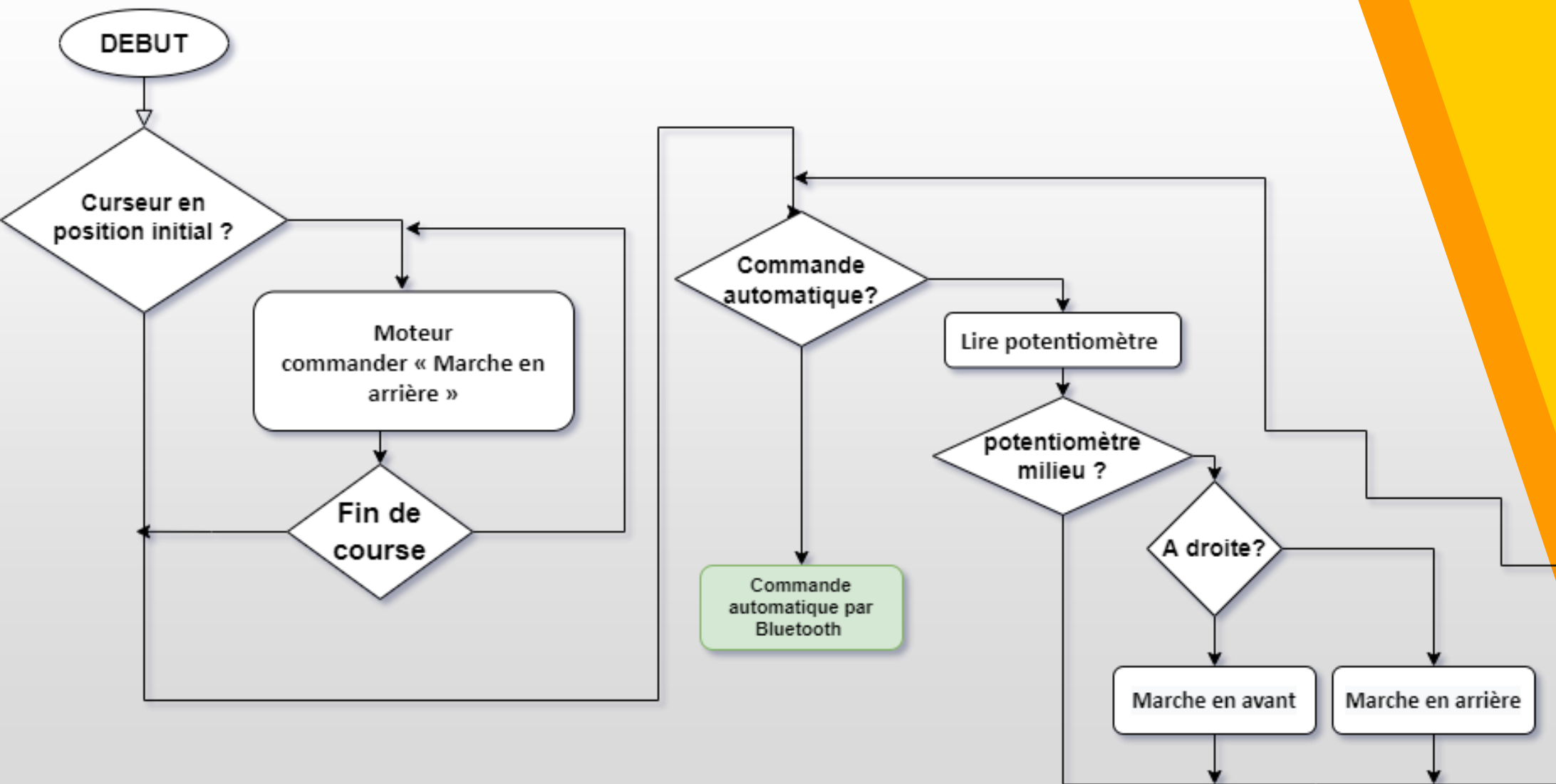


Figure 4 : Diagramme d'exigences seringue auto-posseuse





# FONCTIONNEMENT GLOBALE DU **SYSTEME**



# COMMANDE MANUEL DE SAP

## OBJECTIF 1:

Comment commander  
l'injection manuellement  
avec précision?



# UTILISATION DE CAPTEUR DE POSITION

## 1 – COMPARAISON, ET CHOIX D'UN CAPTEUR,

TYPE DE CAPTEUR	SANS CONTACT	AVANTAGES	INCONVÉNIENTS
Potentiométrique		<ul style="list-style-type: none"> <li>• Grande précision</li> <li>• Peu onéreux</li> </ul>	<ul style="list-style-type: none"> <li>• Usure importante</li> </ul>
LVDT ou RVDT		<ul style="list-style-type: none"> <li>• Grande précision</li> <li>• Robuste</li> <li>• Peu sensible aux environnements sévères</li> </ul>	<ul style="list-style-type: none"> <li>• Assez onéreux</li> <li>• Encombrant et lourd</li> </ul>
Optique		<ul style="list-style-type: none"> <li>• Grande précision</li> <li>• Résolution élevée</li> </ul>	<ul style="list-style-type: none"> <li>• Fragile</li> </ul>
Magnétique Effet Hall	X	<ul style="list-style-type: none"> <li>• Robuste</li> <li>• Peu sensible aux liquides</li> </ul>	<ul style="list-style-type: none"> <li>• Sensible aux chocs</li> <li>• Perturbé par les matériaux magnétiques et les fils électriques</li> <li>• Présente une hystérésis</li> </ul>
Magnétostrictif	X	<ul style="list-style-type: none"> <li>• Robuste</li> <li>• Précis sur les grandes longueurs</li> </ul>	<ul style="list-style-type: none"> <li>• Manque de précision sur les faibles longueurs</li> <li>• Assez onéreux</li> </ul>

# UTILISATION DE CAPTEUR DE POSITION

## 2 – Utilisation du Joystick :

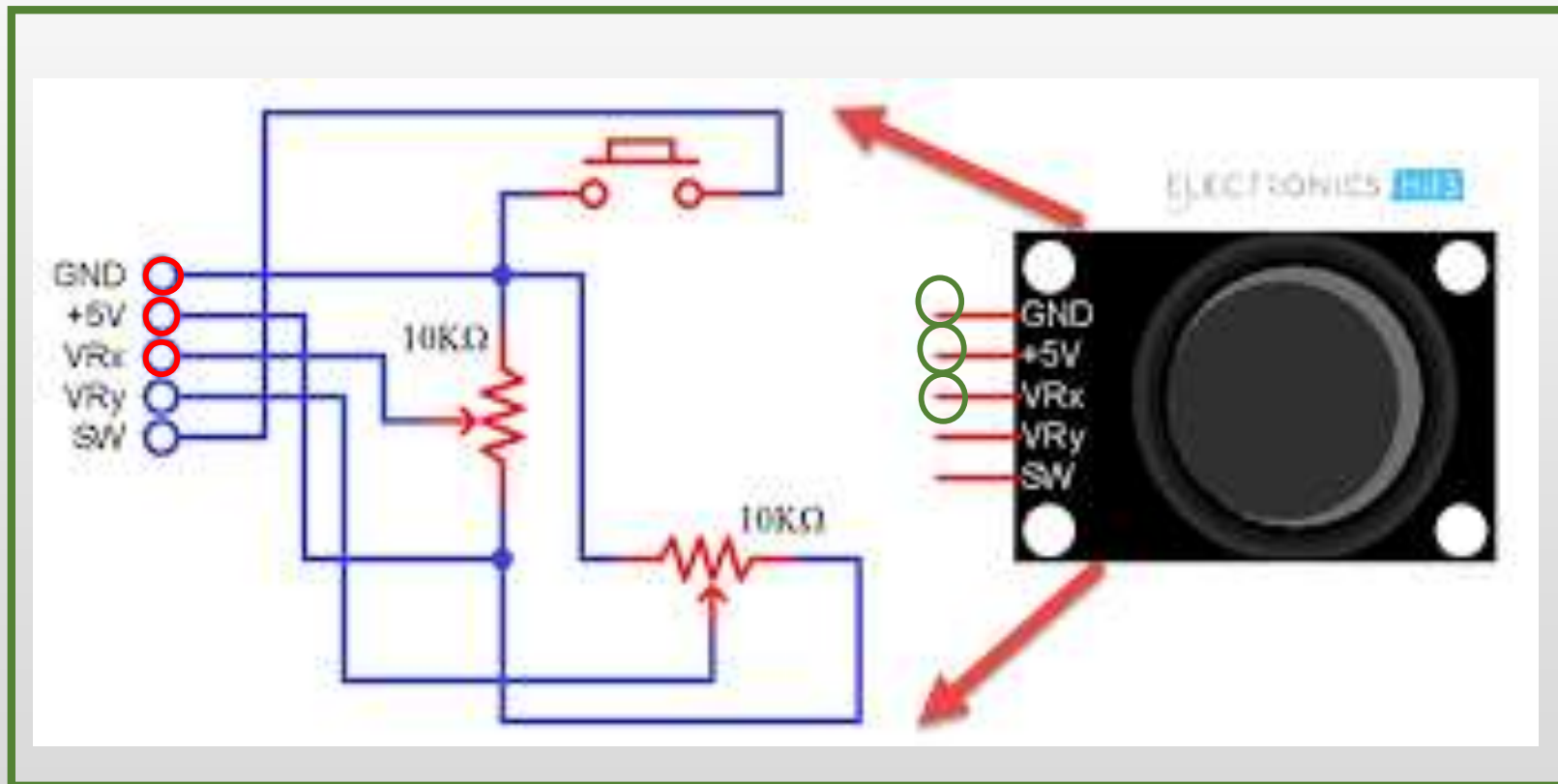


Figure 5 : schématisation du Joystick

# UTILISATION DE CAPTEUR DE POSITION

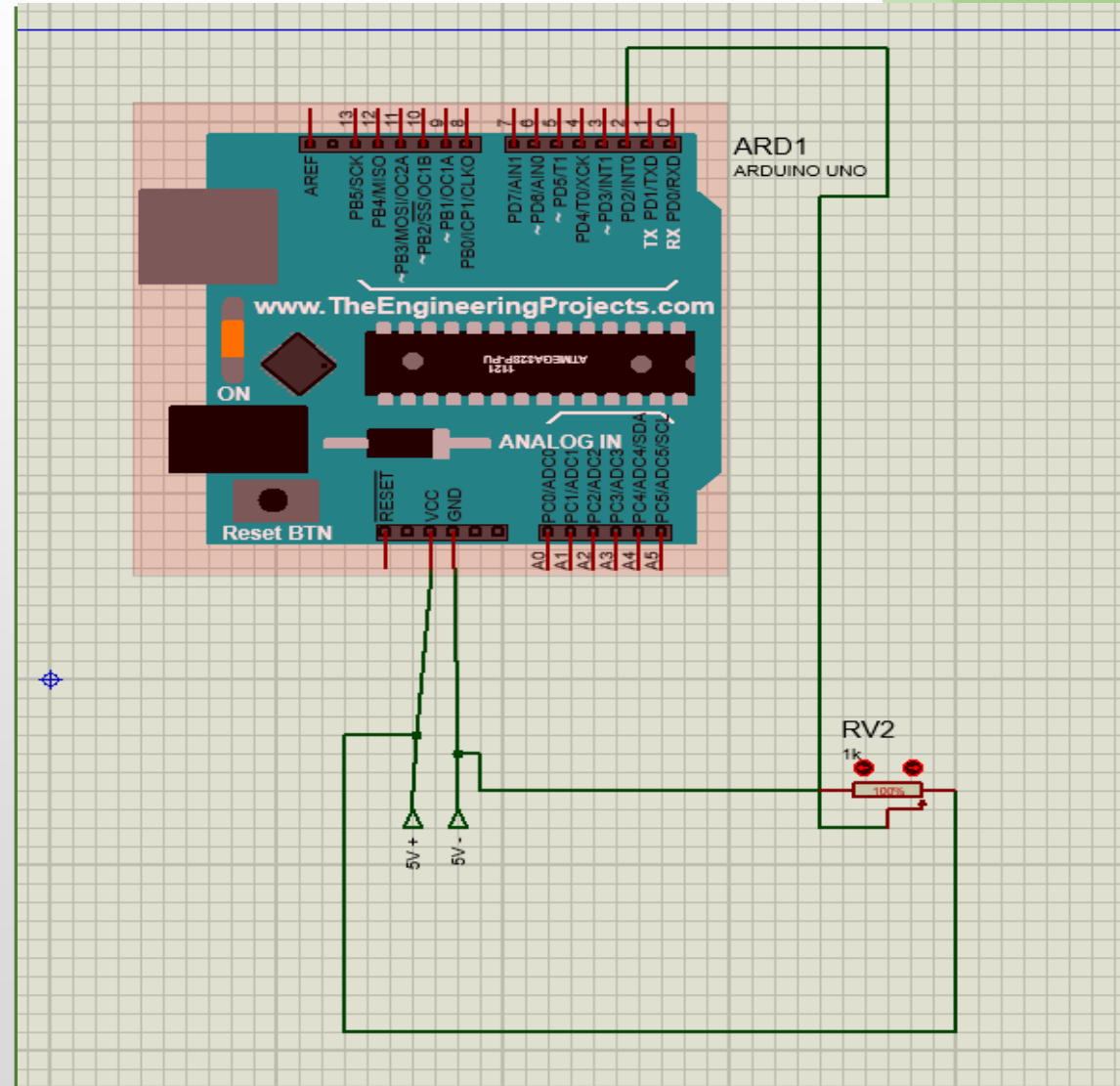
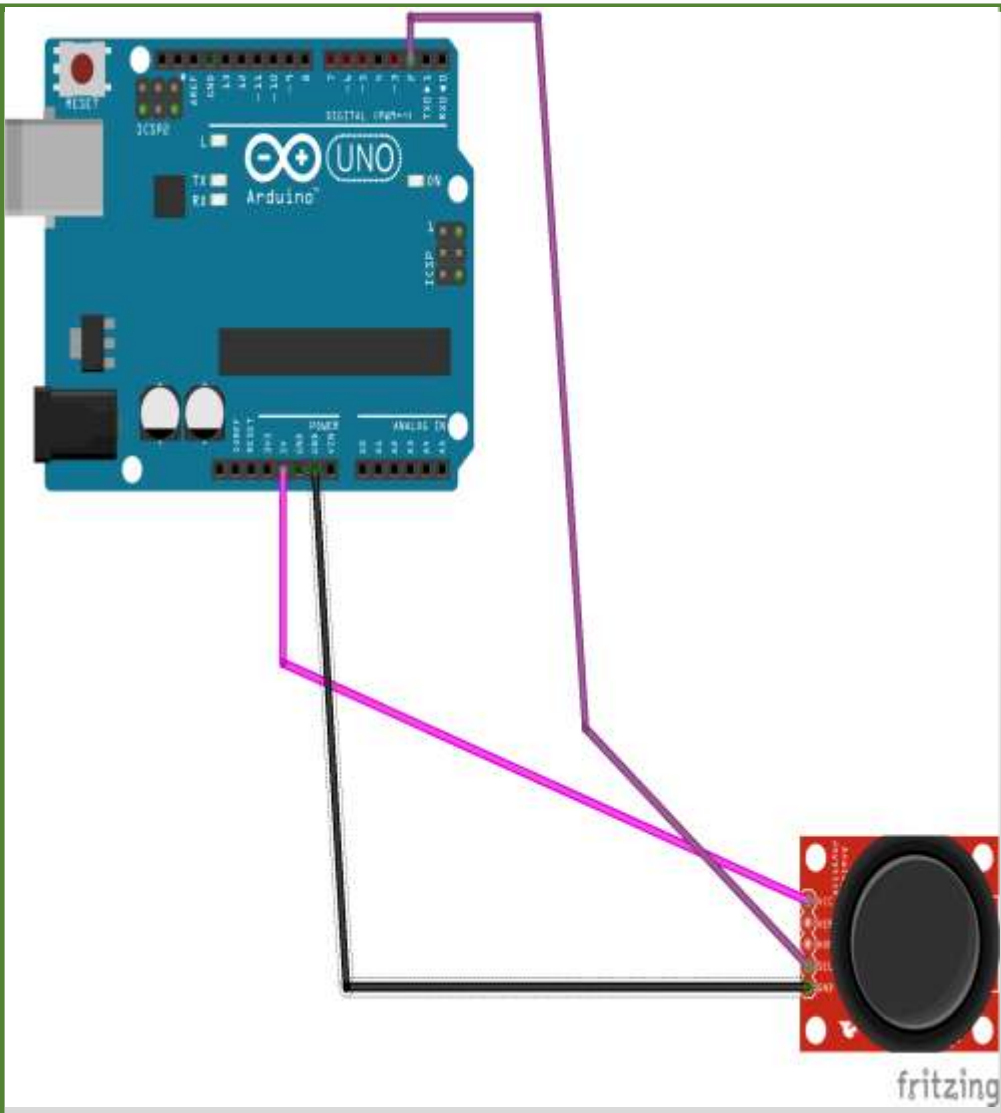


Figure 6 : Montage du Joystick avec Arduino UNO

# UTILISATION DE CAPTEUR DE POSITION

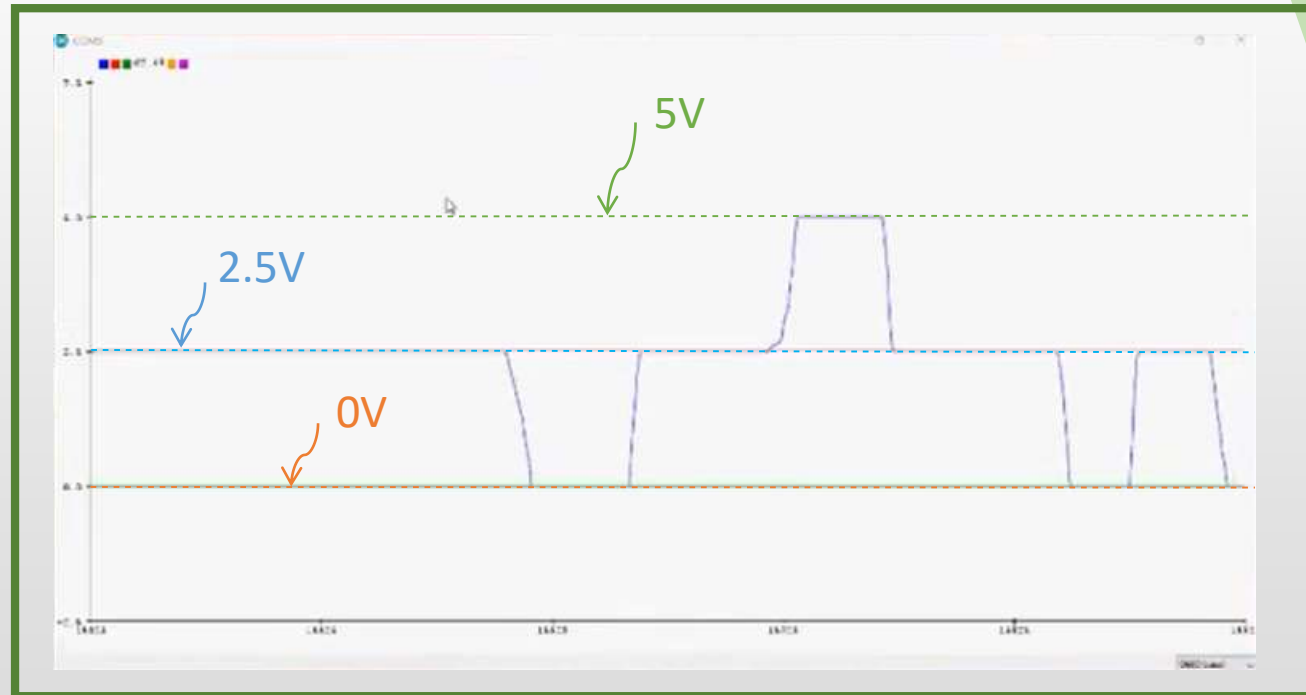
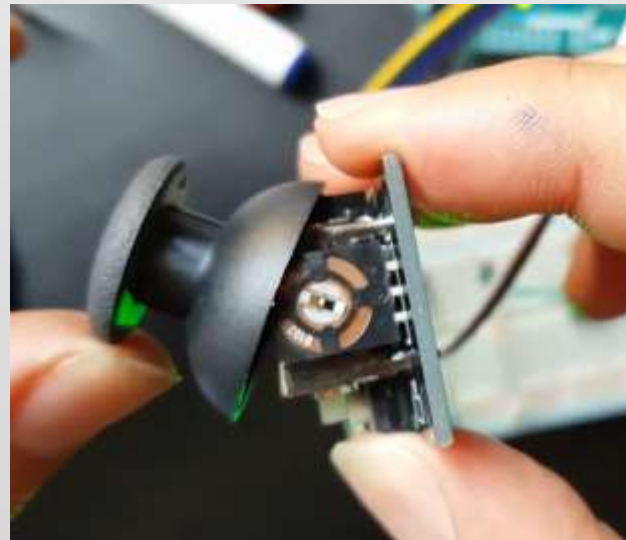
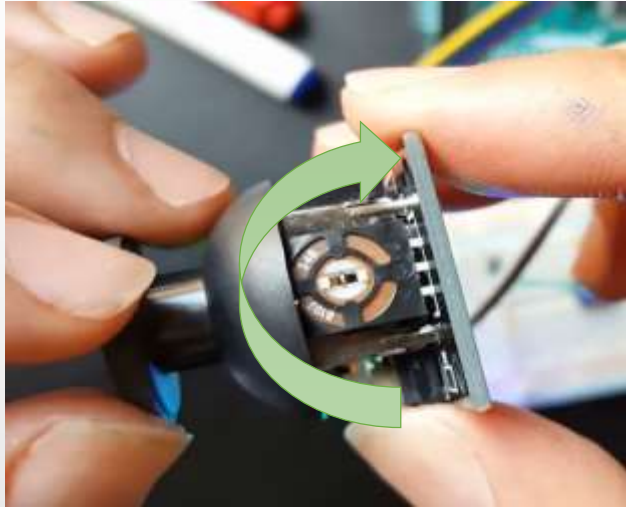
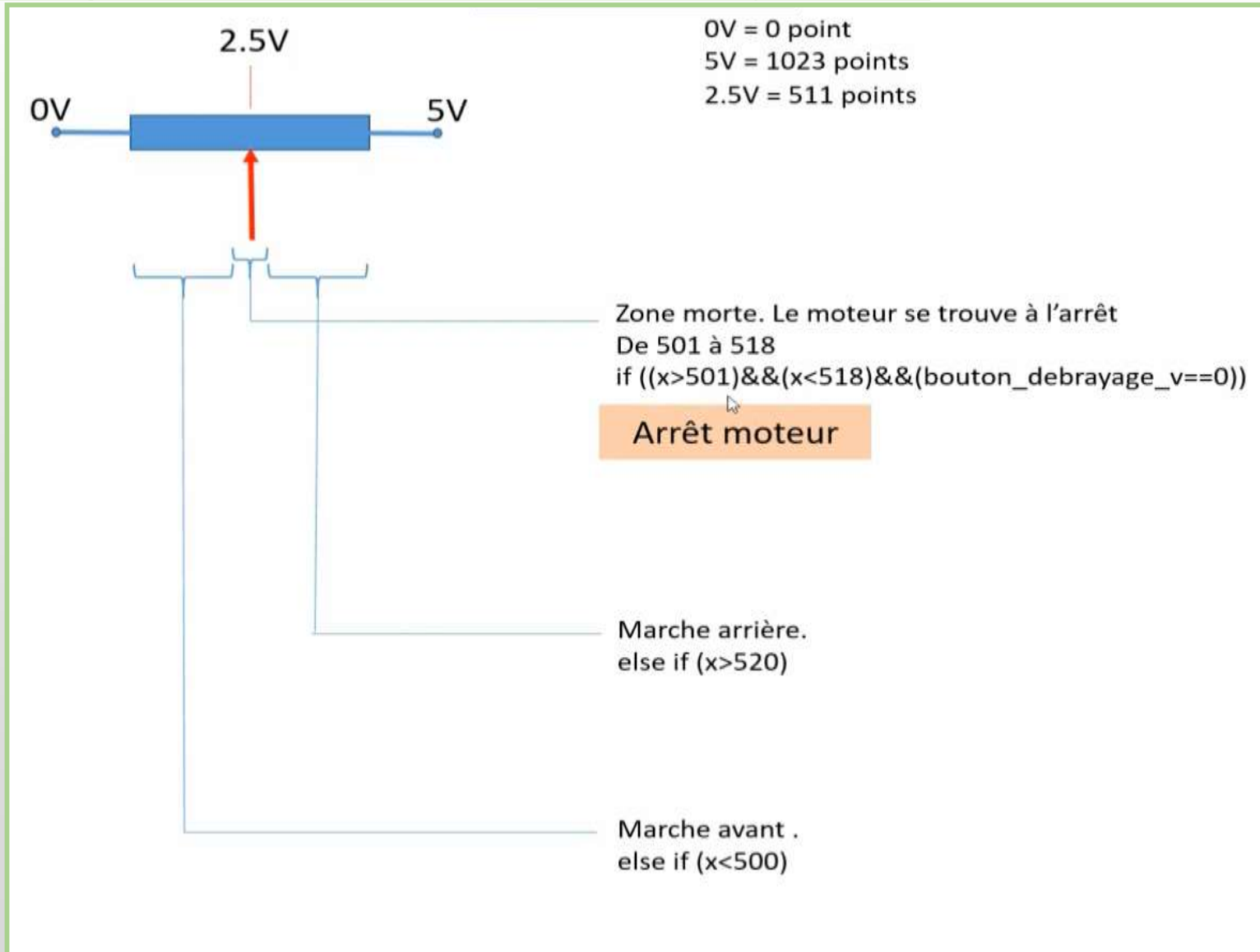


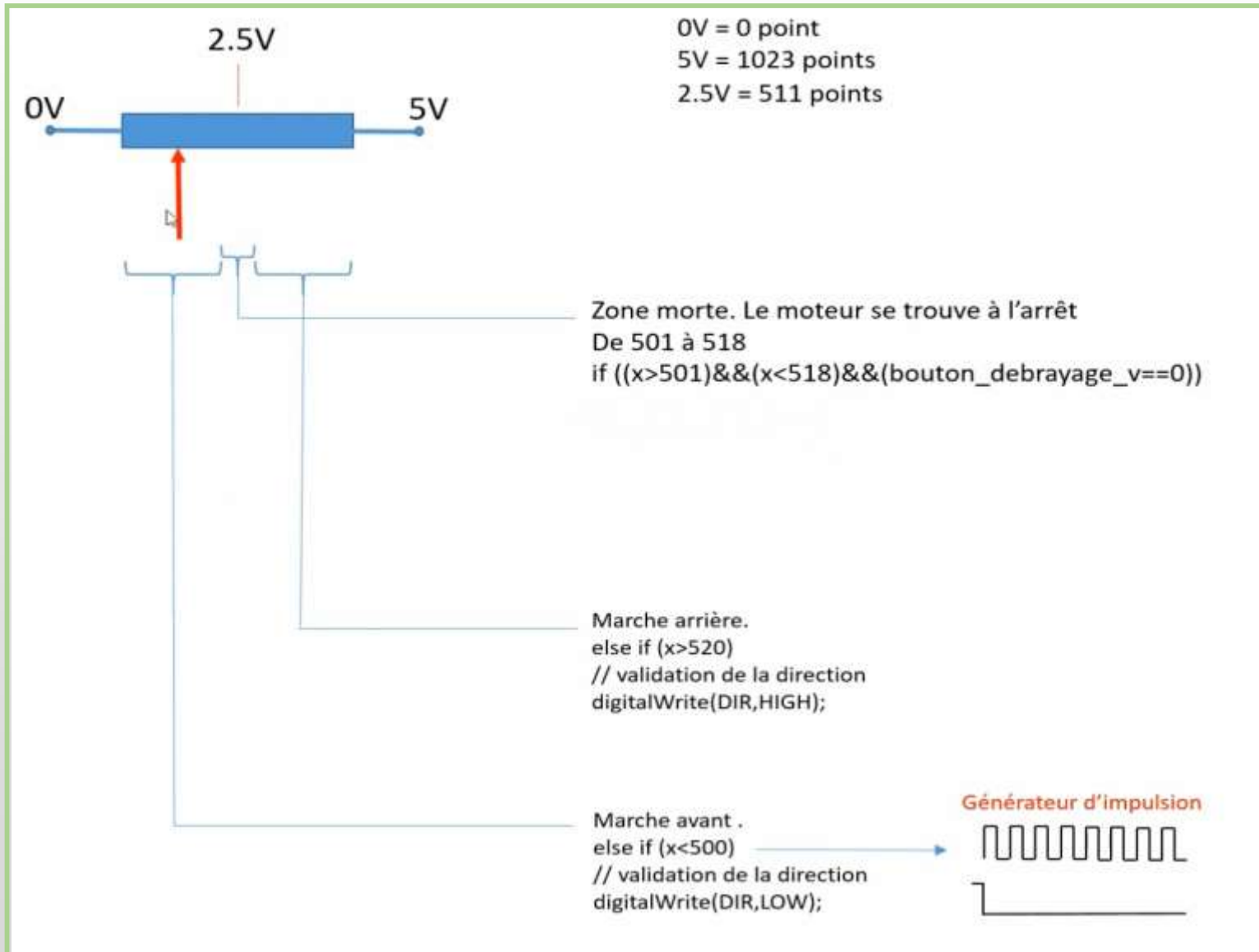
Figure 7 : Allure des signaux lors du mouvement du bras du seringue

# UTILISATION DE CAPTEUR DE POSITION

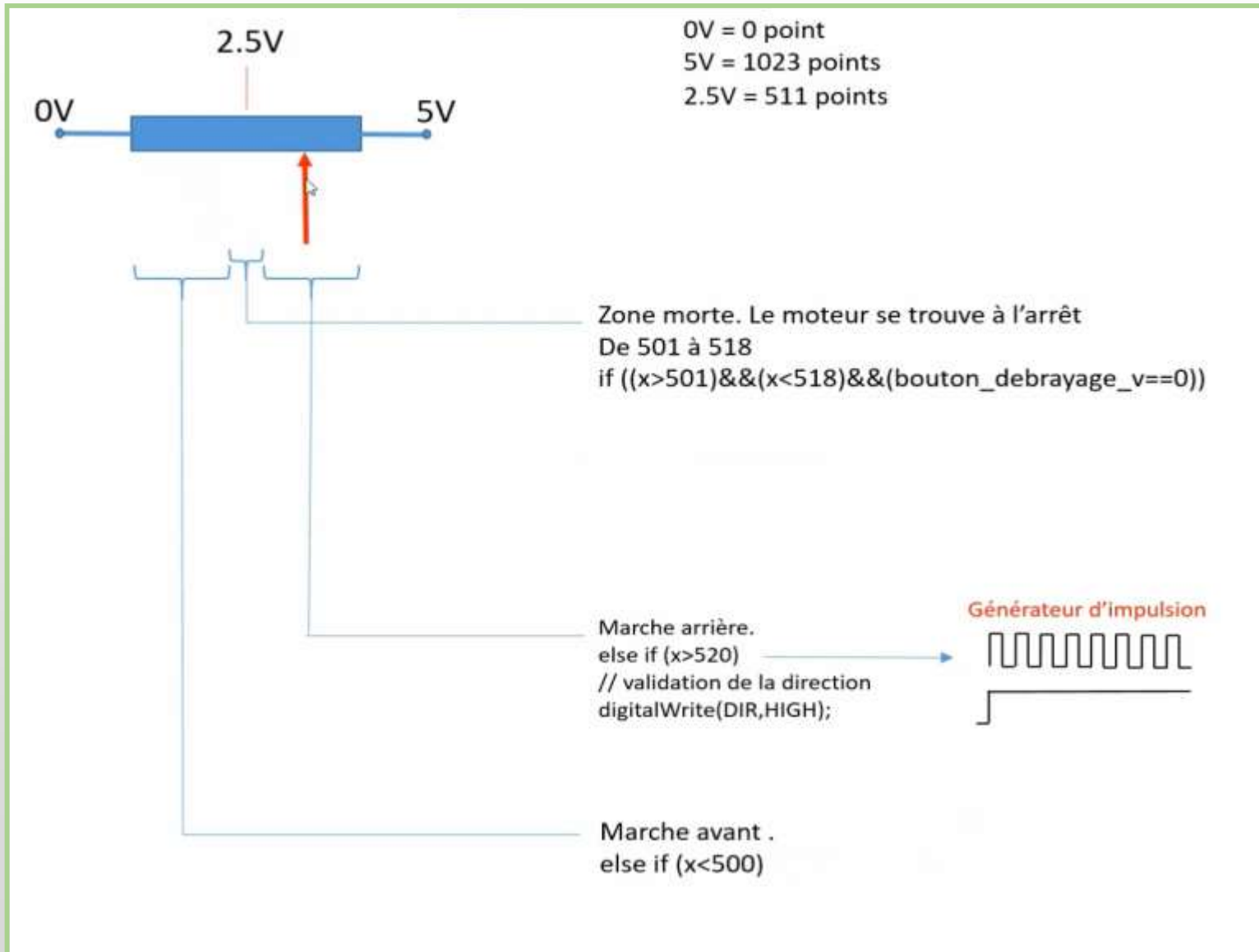
## 3- Analogie avec la fonction voulue du système :



# UTILISATION DE CAPTEUR DE POSITION



# UTILISATION DE CAPTEUR DE POSITION





# SCHEMA D'ALIMENTATION

## OBJECTIF 2:

Comment peut-on  
alimenter les différents  
composants du système?

# SCHEMA D'ALIMENTATION

## 1 - Alimentation des différents composants du circuit :



~ 12V



~ 12V & 5V



~ 5V



~ 5V



~ 5V

Figure 8 : Quelques composants du système

# SCHEMA D'ALIMENTATION

## 2- Les 3 Blocs du circuit d'alimentation :

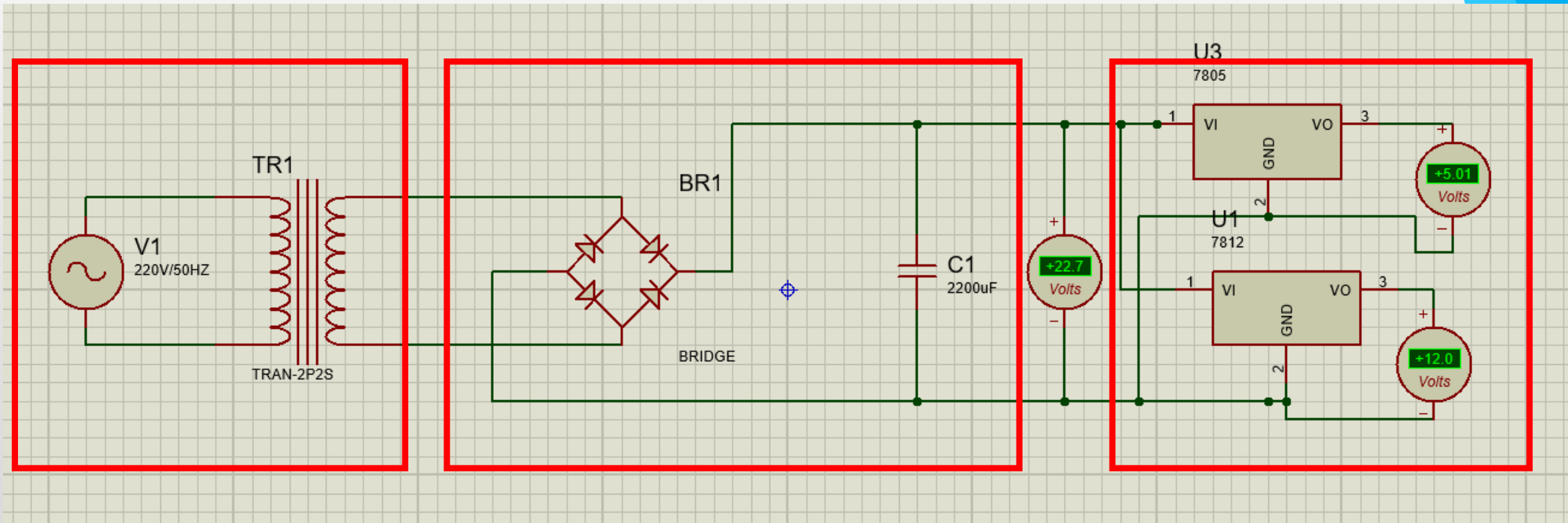
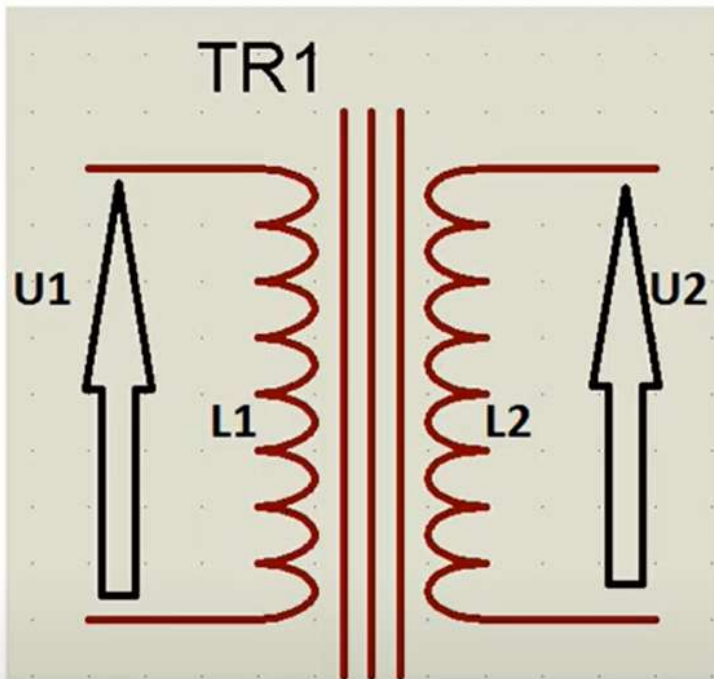


Figure 9 : schéma électrique d'alimentation

# SCHEMA D'ALIMENTATION

## 3- Choisir L2 pour la simulation ISIS :



### Theoriquement :

$$m = \frac{U_2}{U_1} = \frac{I_1}{I_2} \text{ (rapport de transformation)}$$

$$Z_1 = X \cdot L_1 = \frac{U_1}{I_1} \quad \text{ET} \quad Z_2 = X \cdot L_2 = \frac{U_2}{I_2}$$

$$XL_1 = XL_2 \cdot m^2$$

$$\frac{L_1}{L_2} = \left( \frac{U_2}{U_1} \right)^2$$

$$L_2 = L_1 \left( \frac{U_2}{U_1} \right)^2$$

### Transformateur 220V/24V :

$$U_1 = 220V \quad U_2 = 24V$$

Si je vais opter pour la valeur de  $L_1 = 0.1 \text{ H}$

$$\text{Alors } L_2 = 0.1 \left( \frac{24}{220} \right)^2 = 1.19 \text{ mH}$$

# CONTROLE DU MOTEUR

## OBJECTIF 3:

Comment **contrôler la rotation du moteur** en fonction **des instructions** reçues par **l'électronique de puissance**?

# CONTROLE DU MOTEUR

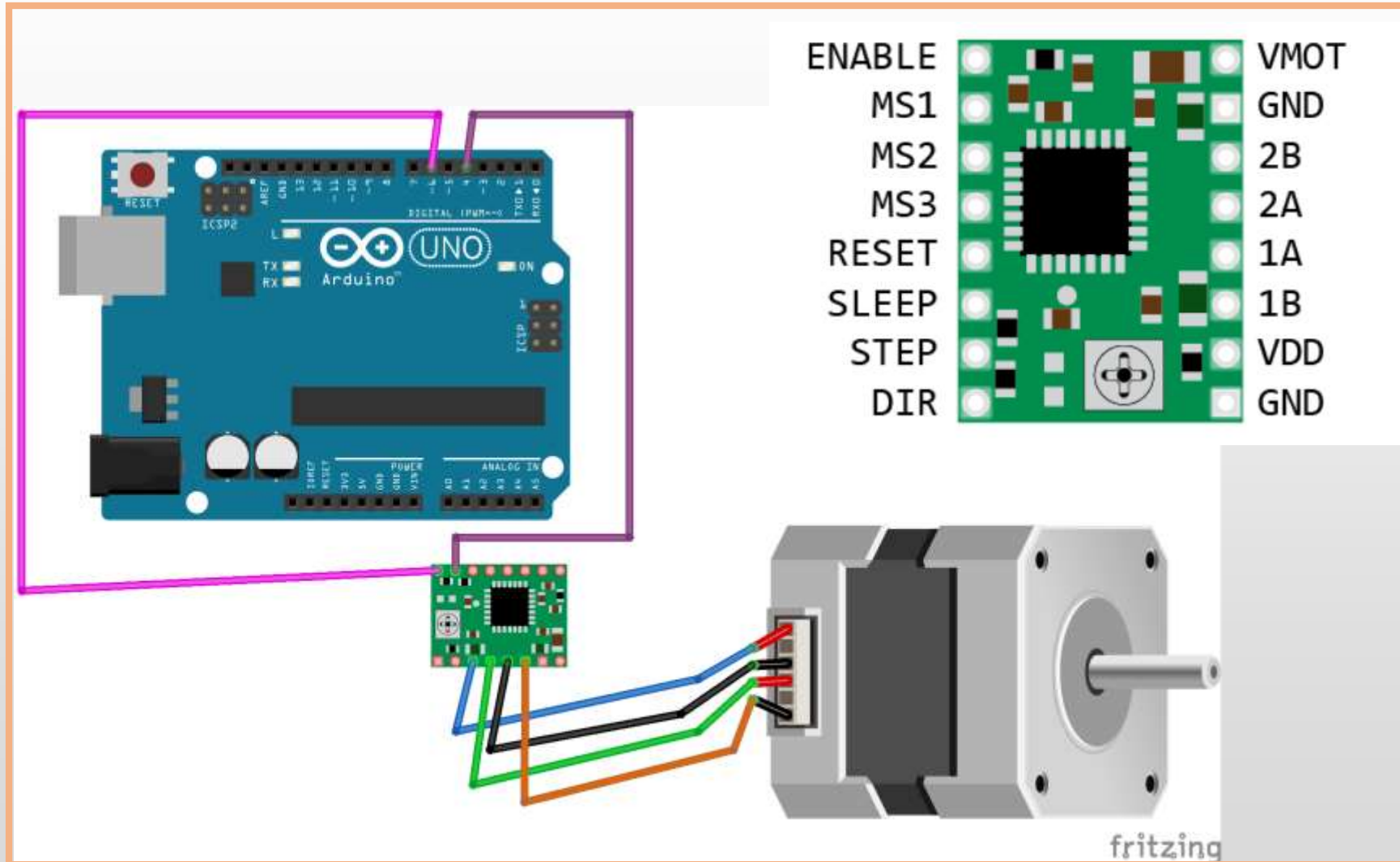
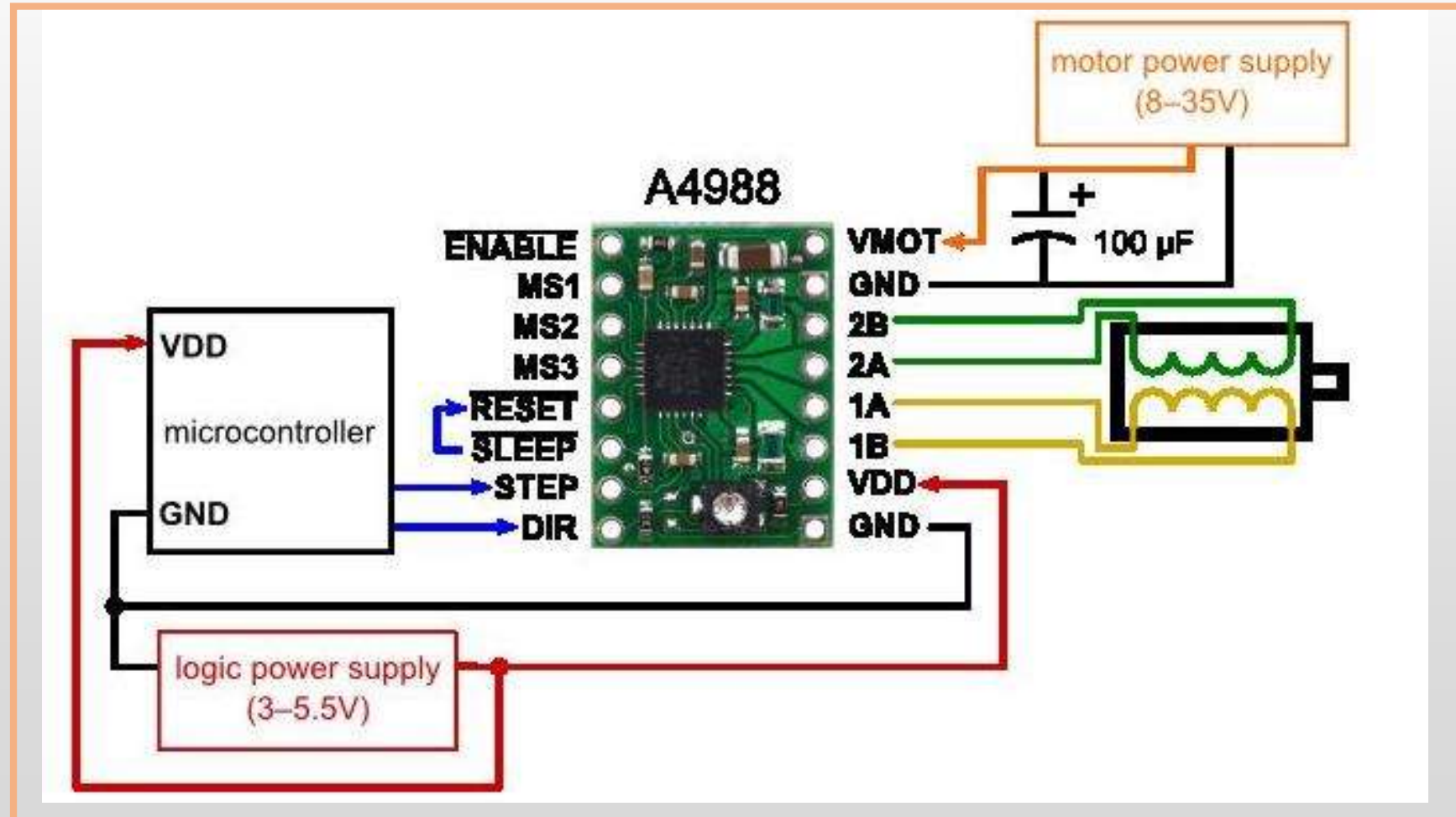


Figure 10 : Montage du controleur avec Arduino UNO

# CONTROLE DU MOTEUR

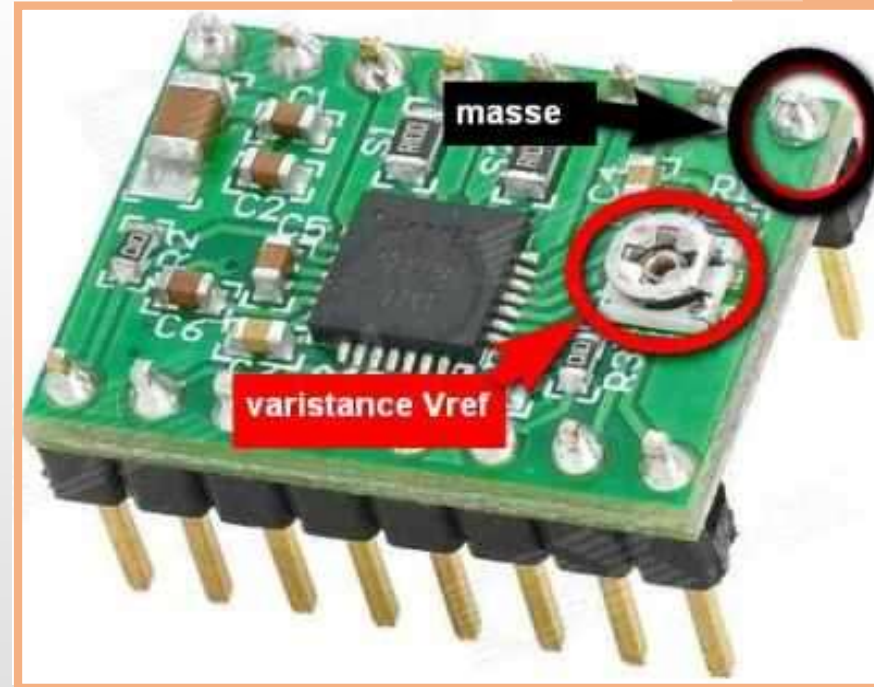
## 1- Alimentation du A4988 DRIVER :





# CONTROLE DU MOTEUR

QU'EST CE QUE  $V_{ref}$  ET  
POURQUOI FAUT-IL L'AJUSTER  
POUR UN BON  
FONCTIONNEMENT DE NOTRE  
POUSSE SERINGUE?



# CONTROLE DU MOTEUR

## 2- Réglage de Vref :

$$V_{\text{ref}} = \frac{I_{\text{max}}}{2.5}$$

Inominal est de 70% du Imax,

$$\text{soit } I_{\text{max}} = \frac{I_{\text{nominal}}}{0.7}$$

$$V_{\text{ref}} = \frac{I_{\text{nominal}}}{0.7 \times 2.5}$$

Dans Notre Cas Pour Le Moteur Pas A Pas [ NEMA17 ] :  
Le courant nominal est de **1.68 A**

$$V_{\text{ref}} = (1.7 / 0.7) / 2.5 = 0.97 \text{ V}$$

Dimensions	42.3x42.3x48 mm
Poids	350g
Diamètre de l'axe	Ø5 x ~24mm
Nombre de phase	2 phase
Voltage standard	2,8V
Nombre de pas	200 pas
Pas angulaire	1,8° (±5%)
Connexion	4 file
Résistance/phase	1.65 Ω
Inductance/phase	2.8mH
Courant/phase	1.68A

Tableau 2.5 : Fiche technique de NEMA17

# CONTROLE DU MOTEUR

## OBJECTIF 3:

Comment peut-on  
piloter l'injection à  
distance?



# UTILISATION DE **CAPTEUR DE POSITION**

1- Communication avec l'utilisateur :

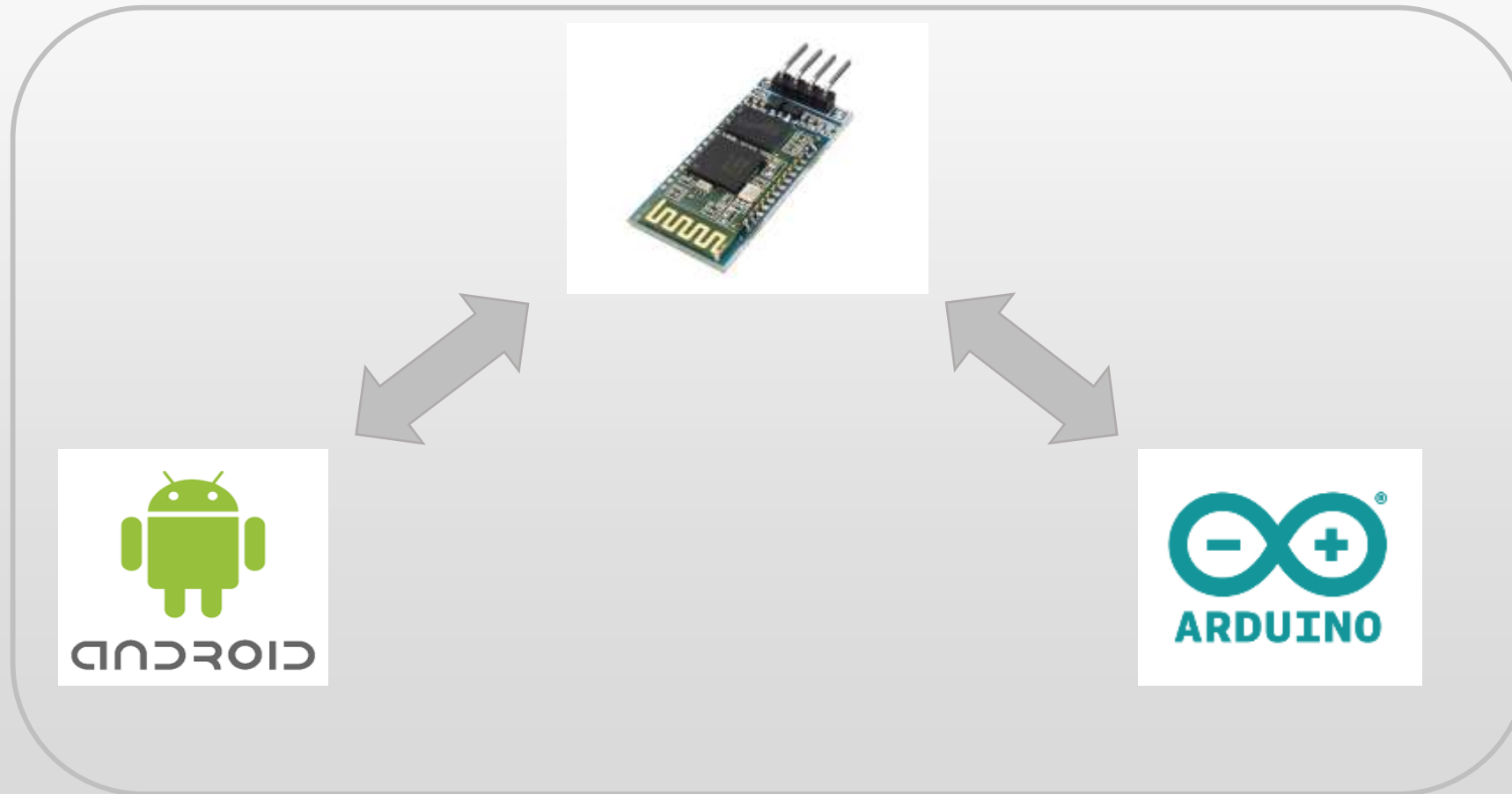
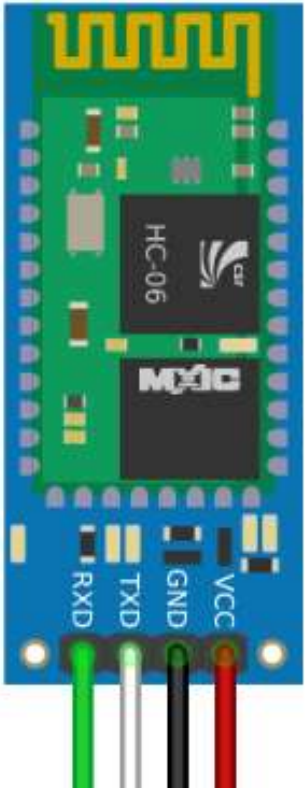


Figure 11 : Principe de fonctionnement de carte bluetooth



# UTILISATION DE **CAPTEUR** DE POSITION

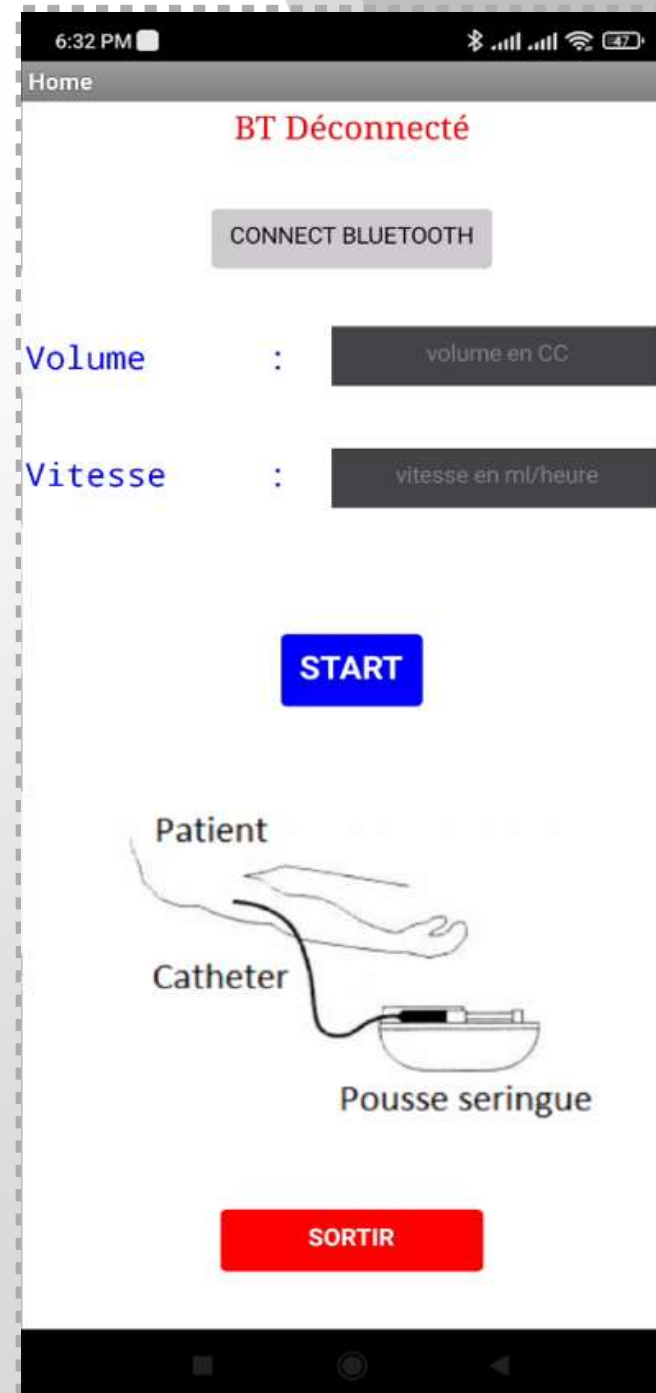


Ce module a quatre pins:

- VCC, source d'alimentation, fil rouge.
- GND, connecté au pin GND de la plaque Arduino, fil noir.
- TXD, transmission des données, fil blanc.
- RXD, réception des données, fil vert.

# APPLICATION ANDROID

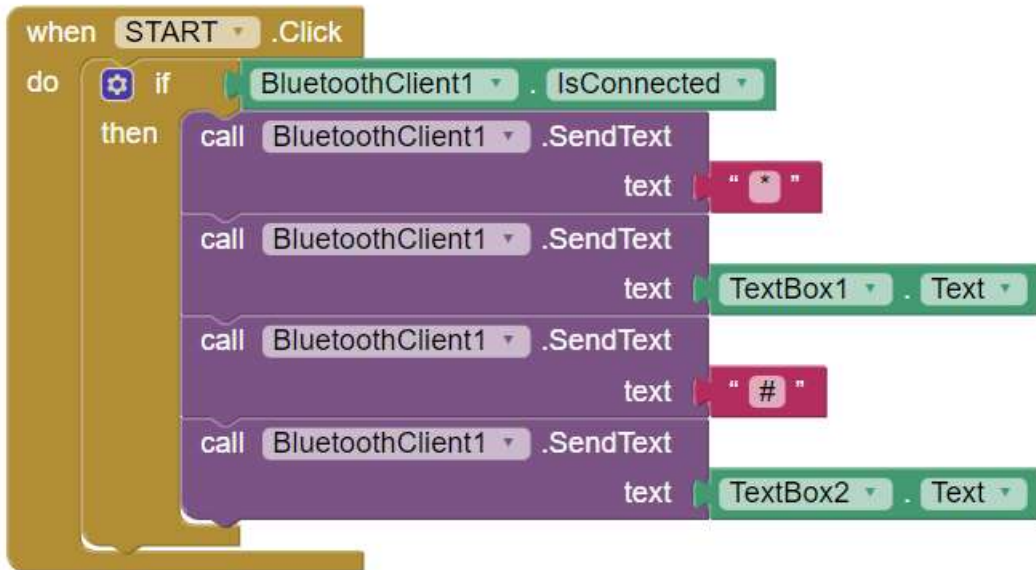
- ▶ Une interface simple,
- ▶ Possibilité de recevoir des notification en cas de panne,
- ▶ Pas de contrainte de distance,
- ▶ Bibliothèque et application gratuites,



# APPLICATION ANDROID

## 1- relation entre Arduino et l'application :

Premièrement l'application envoie un caractère '\*' ensuite un byte vitesse et un autre caractère '#' et enfin une byte de volume, l'Arduino prend la byte après le caractère '\*' pour définir la vitesse, ensuite il prend la byte après le caractère '#' pour définir le volume.



```
while (!digitalRead(A2)==0) {
while (Serial.available()) {
  type=Serial.read();
  if(type=='*') {
    p=Serial.parseInt();
    ss.setSpeed((p/2.76)/60);
  }
  if (type=='#') {
    v=Serial.parseInt();
    if (v>20) {
      Serial.print("donner une capacite moins de 20cc");
      goto ignore;
    }
  }
}
```



# PAS, VOLUME

## OBJECTIF 4:

Comment calculer le  
**volume voulu** en  
fonction des **pas** de  
notre moteur?

# PAS, VOLUME

## 1- Le schéma de la mesure mécanique :

Il faut calculer la distance entre point et autre dans 200 pas (tourne complet).



Figure 12 : Le schéma de la mesure mécanique

# PAS, VOLUME

## 1- La conversion du volume en pas :

Notre programme utilise une bibliothèque pour les moteurs pas à pas qui fonctionne avec le pas donc on a convertie le volume (ml) en pas.

200 pas → 1.25 mm

1ml → 3.45 mm

1ml →  $\frac{3.45 \times 200}{1.25}$

1ml → 552 pas

Donc pour injecté 1ml il faut que le moteur fait 552 pas.

# PAS, VOLUME

## 1- La conversion du volume en pas :

Conversion tr/min en ml/heure.

Ainsi que la bibliothèque fonctionne avec la vitesse tr/min , donc on va convertir la vitesse tr/min en ml/heure

200 pas/min  $\longrightarrow$  1 tr/min

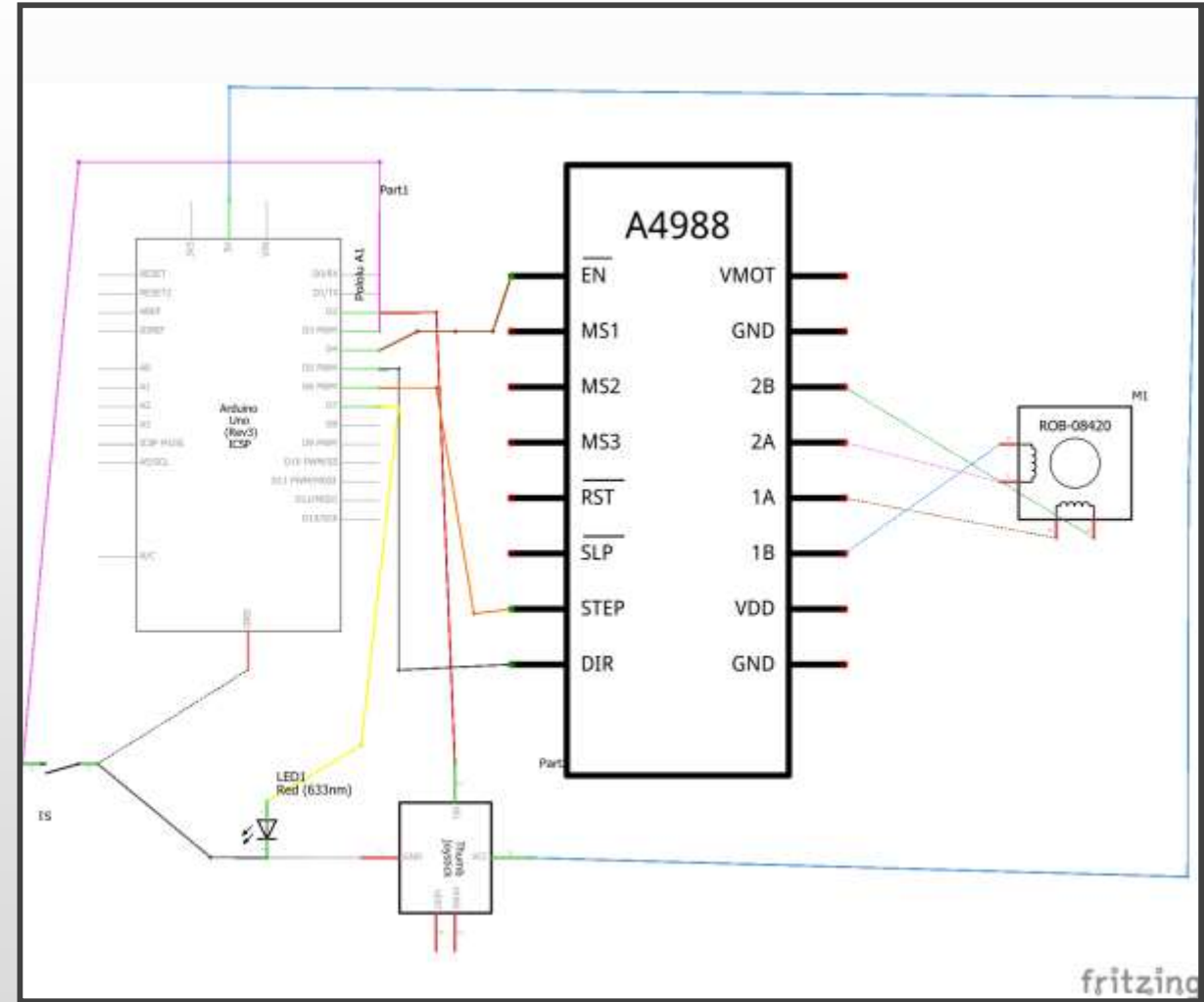
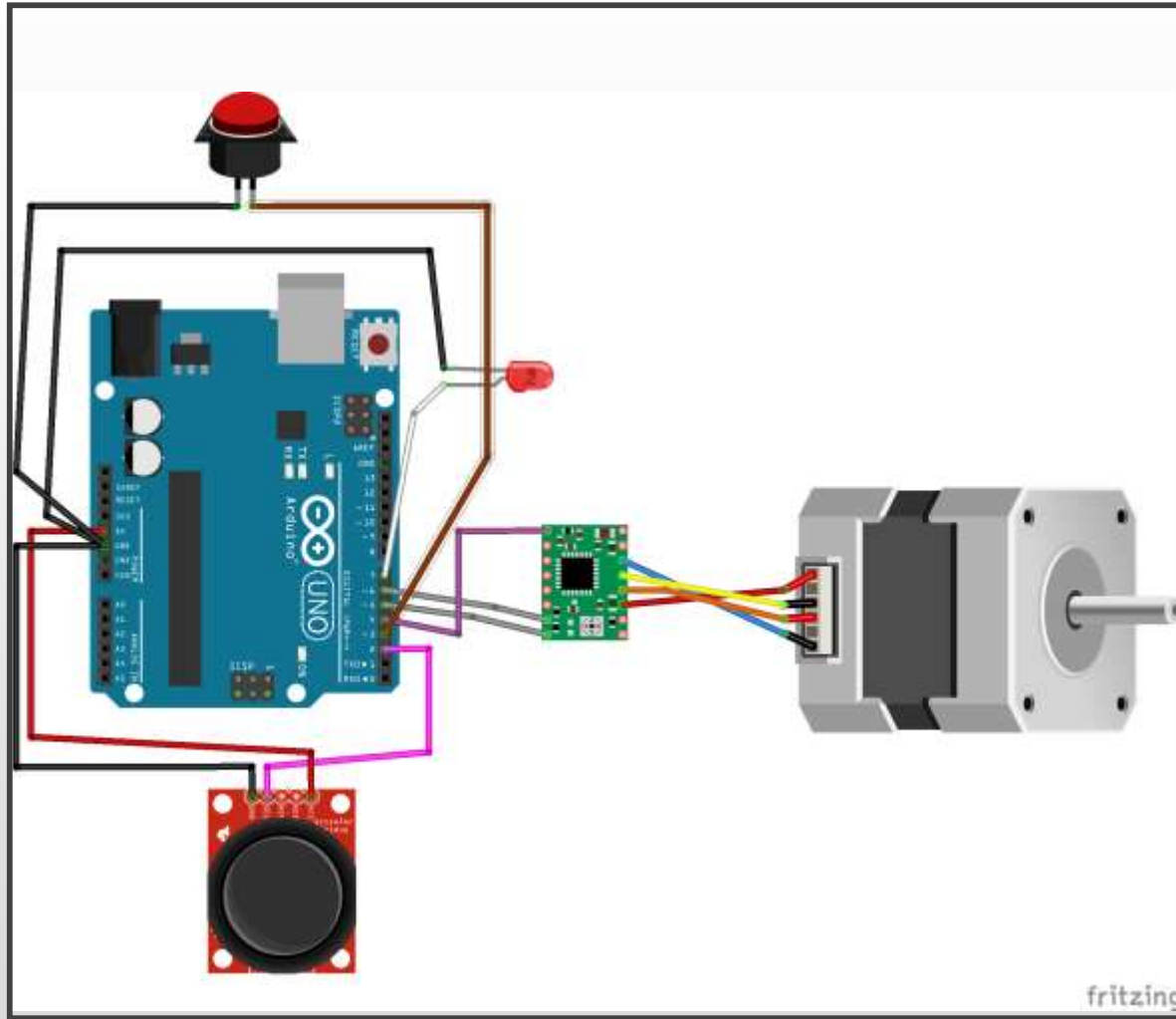
552 pas/min  $\longrightarrow$  2.76 tr/min

1 ml/min  $\longrightarrow$  2.76 tr/min

1 ml/heure  $\longrightarrow$  2.76/60 tr/min

Donc pour injecté avec une vitesse 1ml/heure il faut que le moteur tourne avec une vitesse 2.76/60 tour/min.

# MONTAGE FINAL:





# ANNEXES :

```
#include <StepperDriver.h>
unsigned long demarrage = micros();
int motor_steps = 200;
int base_de_temps1 = 1;
int PUL = 5;
int DIR = 6;
int ENA = 4;
int bouton_debrayage = 3;
int bouton_debrayage_v;
int voyant_vert_mar = 7;
int JoyStick_X = 2;
int vitesse=0 ;
int x;
int p;
int v;
char type;
char m='m';
char a='a';
StepperDriver ss(motor_steps, base_de_temps1,PUL ,DIR ,ENA );
```

Figure 14 : Initialisation

```
void setup() {
  pinMode (PUL, OUTPUT);
  pinMode (DIR, OUTPUT);
  pinMode (ENA, OUTPUT);

  pinMode(bouton_debrayage,INPUT);
  pinMode(voyant_vert_mar,OUTPUT);

  pinMode(A4,INPUT);
  int x=digitalRead(A4);
  while (!x==0){
    x=digitalRead(A4);
    ss.setSpeed(30);
    ss.step(-1);
  }
  Serial.begin(9600);
}
```

Figure 15 : Retour du seringue au position initial



# ANNEXES :

```
void loop() {  
  while (!digitalRead(A2)==0){  
    while (Serial.available()){  
      type=Serial.read();  
      if(type=='*'){  
        p=Serial.parseInt();  
        ss.setSpeed((p/2.76)/60);  
      }  
      if (type=='#'){  
        v=Serial.parseInt();  
        if (v>20){  
          Serial.print("donner une capacite moins de 20cc");  
          goto ignore;  
        }  
        Serial.print(m);  
        ss.step(v*552);  
        delay(2000);  
        x=digitalRead(A4);  
        ss.setSpeed(150);  
        ss.step((-v*552)+200);  
        ss.setSpeed(20);  
        while (!x==0);  
        x=digitalRead(A4);  
        ss.step(-1);  
      }  
      Serial.print(a);  
      ignore:break;  
    }  
  }  
}
```

Figure 16 : Traitement des données du commande Bluetooth

```
while (digitalRead(A2)==0){  
  bouton_debrayage_v = digitalRead(bouton_debrayage);  
  if ( bouton_debrayage_v == 1 )  
  {  
    digitalWrite(ENA,LOW);  
  }  
}
```

Figure 17 : Débrayage

# ANNEXES :

```
int x;
x=analogRead(JoyStick_X);
if ((x>501)&&(x<518)&&(bouton_debrayage_v==0))
{
digitalWrite(voyant_vert_mar,LOW);
digitalWrite(ENA,HIGH);
digitalWrite(DIR,HIGH);
digitalWrite(PUL,HIGH);
}

else if (x>520)
{
digitalWrite(voyant_vert_mar,LOW);
digitalWrite(ENA,HIGH);
digitalWrite(DIR,HIGH);
vitesse = map ( x,520,1023,3500,5);
deplacement();
}

else if (x<500)
{
digitalWrite(voyant_vert_mar,HIGH);
digitalWrite(DIR,LOW);
digitalWrite(ENA,HIGH);
vitesse = map ( x,500,0,3500,5);
deplacement();
}
```

```
void deplacement()
{
if (micros()-demarrage>vitesse)
{
digitalWrite(PUL,base_de_temps1!=base_de_temps1);
demarrage=micros();
}
}
```

# ANNEXES :

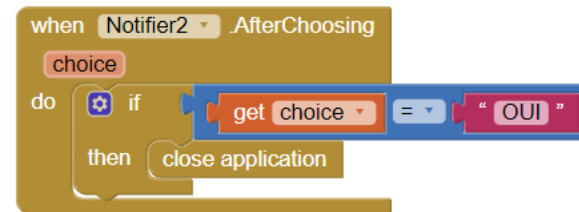
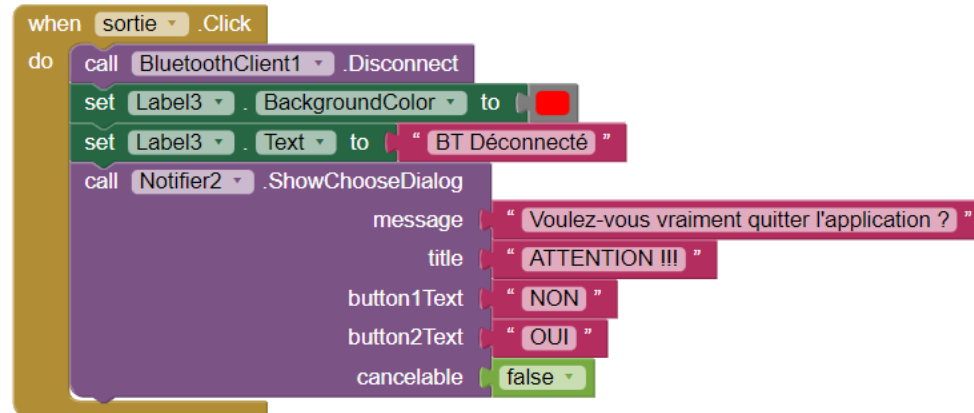
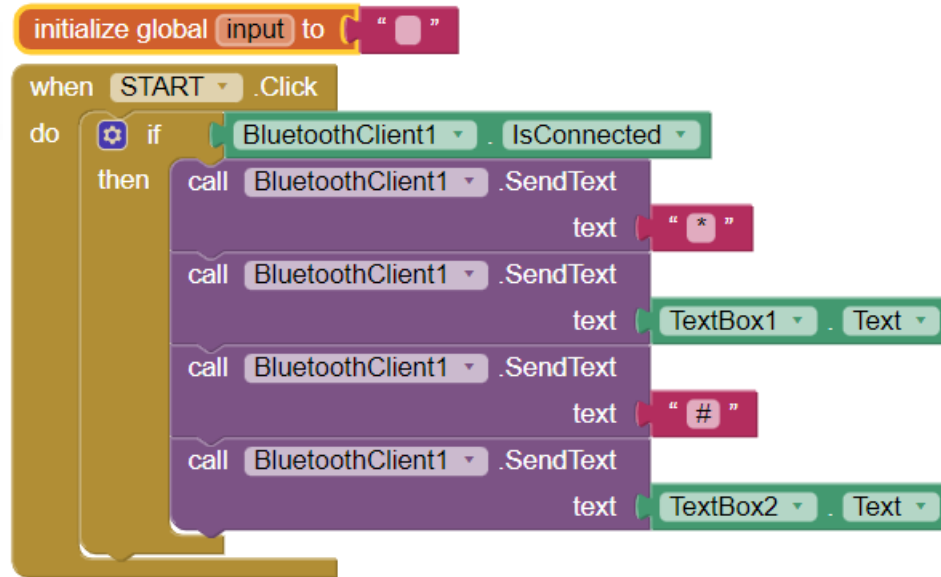
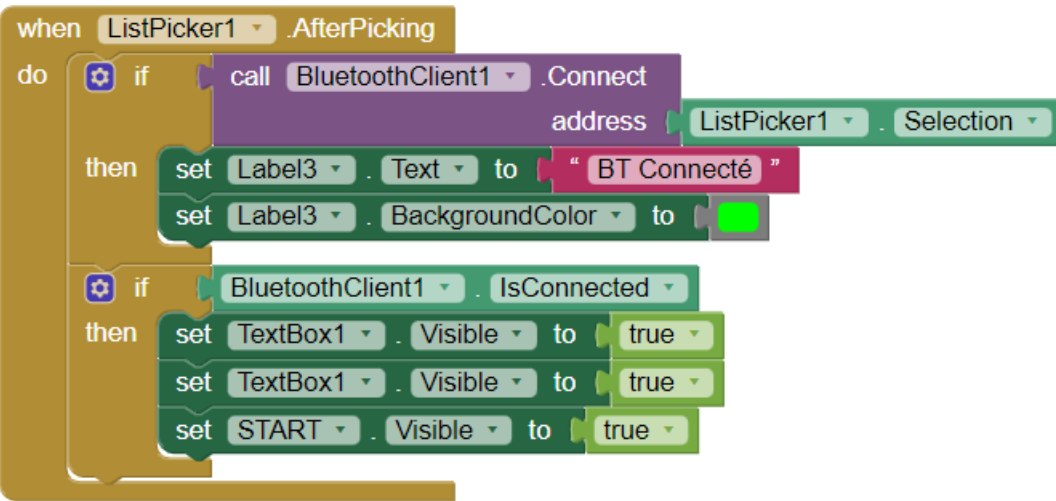
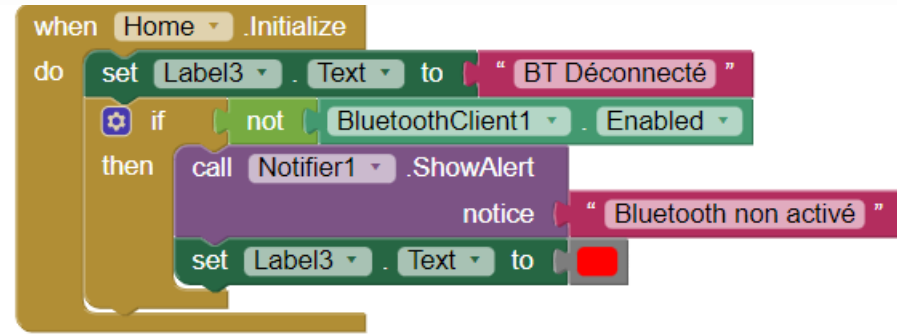


Figure 19 : Blocs de commande par Bluetooth

**MERCI DE VOTRE  
ATTENTION !**