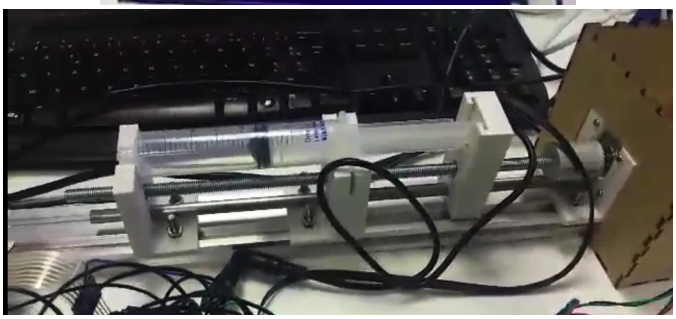
# Pousse - Seringue

UE : Atelier de Recherche Encadrée
Portail : PCGI

Membres:
Hadrien BAKKAR
Shelly CLEMENTE
Idel GABOLAEV
Karoliina KOPONEN
Jieyeon WOO

Enseignants :
Christian SIMON et Vincent DUPUIS

UPMC
SORBONNE UNIVERSITÉS

# Sommaire

# I. Introduction

Le but de notre projet est de créer une pousse-seringue. La fonction de cet objet est de pousser un volume donné de fluide contenu dans la seringue pendant un certain temps à débit constant. Cela est utilisé dans divers domaines tels que la santé ou dans le cadre d'autres machine (ex : cytomètre).

L'utilisation la plus répandue de la pousse-seringue est en soins palliatifs, à continuellement administrer les analgésiques, les antiémétiques et d'autres drogues. Ceci empêche les périodes pendant lesquelles ces médicaments dans le sang sont trop élevés ou trop bas, et évite l'utilisation de plusieurs comprimés. Les pousse-seringues sont aussi également utiles pour dispenser IV médicaments pendant plusieurs minutes. Dans le cas d'un médicament qui doit être lentement enfoncé pendant plusieurs minutes, ce dispositif économise du temps du personnel et réduit les erreurs.

# II. Répartition des tâches

Communication et coordination : Shelly CLEMENTE
Aspects physiques : Karoliina KOPONEN
Aspects mécaniques : Hadrien BAKKAR
Modélisation et programmation : Idel GABOLAEV et Jieyeon WOO

## 1) Communication et coordination

Cette personne est essentiellement le meneur du groupe et se concentre sur l'organisation des créneaux de travail en dehors des séances obligatoires chaque semaine. Il est aussi le responsable du page wiki, la documentation de l'avancement, de trouver le matériel pour le projet et la gestion du budget. Ce travail consiste à veiller sur les tâches de tous les membres du groupe et d'être sur qu'ils avancent. Cette personne aide également avec des tâches dans les autres aspects, tels que souder les fils et les essais du projet.

## 2) Aspects physiques

Cette personne est chargée de l'analyse et du calcul du débit volumique de la pousse-seringue. Cette tâche consiste à utiliser l'information de notre pousse-seringue et ce que nous savons sur le débit volumique pour calculer celui de notre projet. De plus, cette personne analysera les incertitudes et erreurs possibles qui pourraient poser des problèmes à notre pousse-seringue.

## 3) Aspects mécaniques

Cette tâche consiste à savoir comment toutes les pièces de la pousse-seringue travaillent ensemble, en détail, pour créer et modifier le débit volumique. En d'autre termes, comment l'Arduino DUE, l'écran LCD, le pavé numérique, le moteur et les tiges fonctionnent et travaillent étape par étape. Cette personne doit aussi comprendre les calculs du débit volumique et les transformations de l'énergie qui se produisent dans la machine.

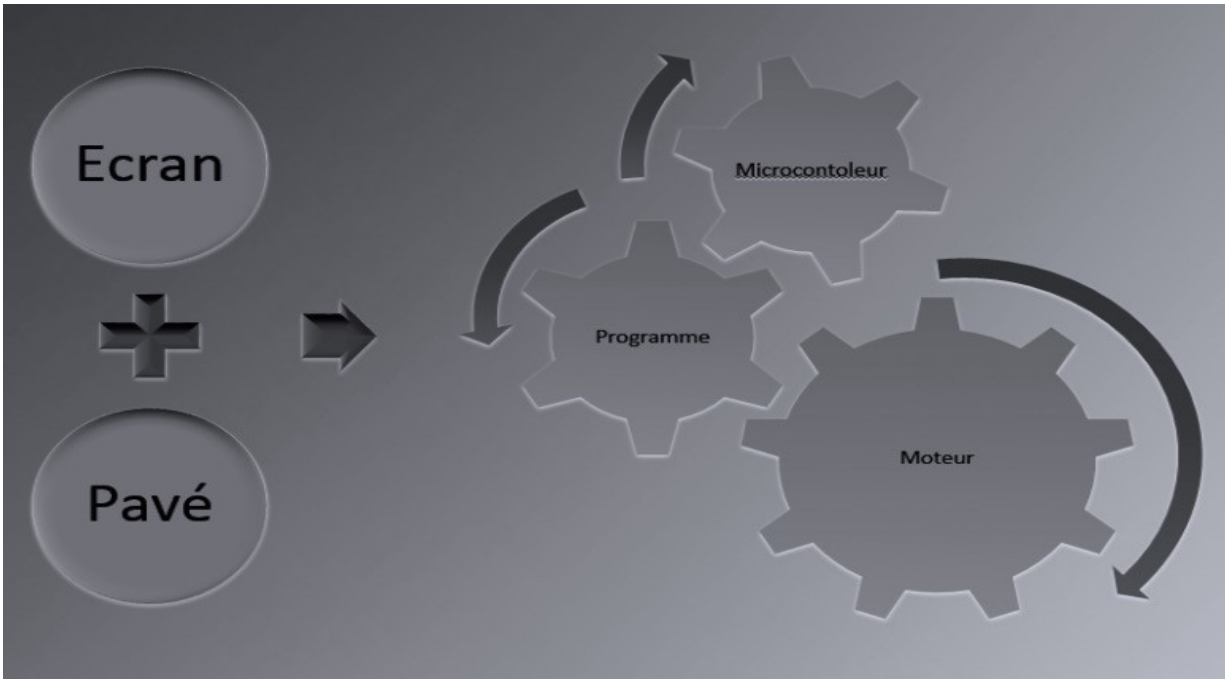
## 4) Modélisation 3D et programmation

Cette tâche est divisée en deux parties :

La première est l'impression des pièces 3D en utilisant les fichiers trouvés en ligne pour une pousse-seringue open source. Ces pièces 3D seront utilisées comme une partie du support et vont également tenir les tiges qui permet la seringue de pomper du liquide. En outre, ce travail comprend la création de la boîte en bois qui tiendra l'Arduino DUE, les fils et pavé numérique, ainsi que d'agir comme une cadre pour l'écran.

La deuxième consiste d'écrire les codes pour la programmation de l'écran LCD, le moteur et le pavé numérique. En premier, ils doivent veiller que les connexions sont correctes et que l'écran, le moteur et le pavé fonctionnement. Ensuite, ils doivent écrire les codes qui fassent en sorte que lorsque que vous entrez le débit volumique en utilisant le pavé numérique, il s'affichera sur l'écran, puis le moteur effectuera le débit volumique qui a été tapé.

# III. Aspect mécanique




Afin de pouvoir répondre aux besoins que doit combler ce type de machine, nous devons créer un mouvement translationnel avec une longueur et une vitesse choisis pour pousser la seringue. Pour pouvoir faire cela, on utilise un moteur pas à pas que l'on contrôlera grâce à un pavé numérique et un écran LCD. Ces éléments seront reliés par un microcontrôleur Arduino DUE pour fonctionner ensemble. Le mouvement de rotation du moteur sera ensuite convertit en mouvement de translation grâce à un système de « vis sans fin », d'écrou et d'une pièce imprimée en 3D. Cette pièce sera alors rattachée à la partie qui va pousser de la seringue. Le reste de la seringue est immobilisée grâce à d'autres pièces imprimées en 3D qui seront fixées sur un rail grâce à des vis et écrous.
Le mouvement effectué par le moteur, donc par la seringue, sera contrôlé par le code téléversé à l'Arduino, qui demandera à l'utilisateur le volume et le débit voulu pour pousser le fluide.

Nous devons donc calculer combien de pas il faut effectuer par le moteur pour 1ml afin de le contrôler précisément grâce au code.

On pose donc le calcul suivant :

- Sur la vis sans fin on mesure 1cm = 8tours or 1tour = 200pas (indiqué sur le moteur)
- Donc, 1cm = 1600 pas
- Sur la seringue on mesure 10ml= 2,6cm on a donc 1ml = 2,6/10cm

→ On obtient finalement 1ml = 1600*2,6/10pas = 416pas

Partie modélisation et programmation : En effectuant des essais, on remarque que l'on doit ajouter environ 10 pas à cette valeur trouvée par le calcul pour faire exactement 1ml. On choisit donc la valeur de 425pas pour faire 1ml.

Après que l'utilisateur ait choisi le volume de fluide désiré grâce au pavé numérique et à la vérification par l'écran LCD (les éléments connectés à l'Arduino et les informations traitées par la code), le code effectué par l'Arduino multipliera donc le nombre de ml par 425 et fera tourner le moteur, ce qui poussera la partie mobile de la seringue.

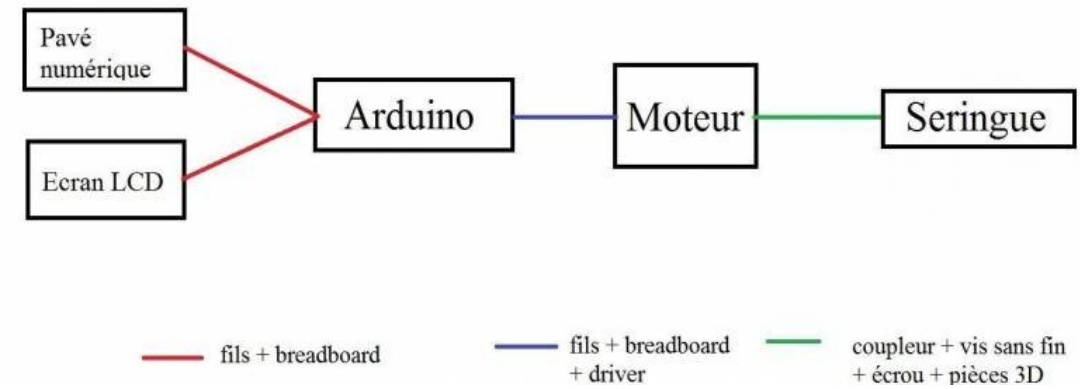



*Image 1 : Schéma fonctionnel*

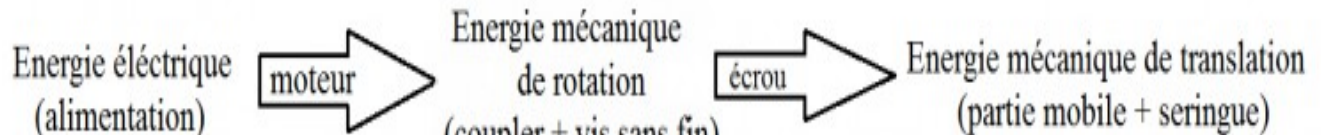



*Image 2 : Bilan énergétique*

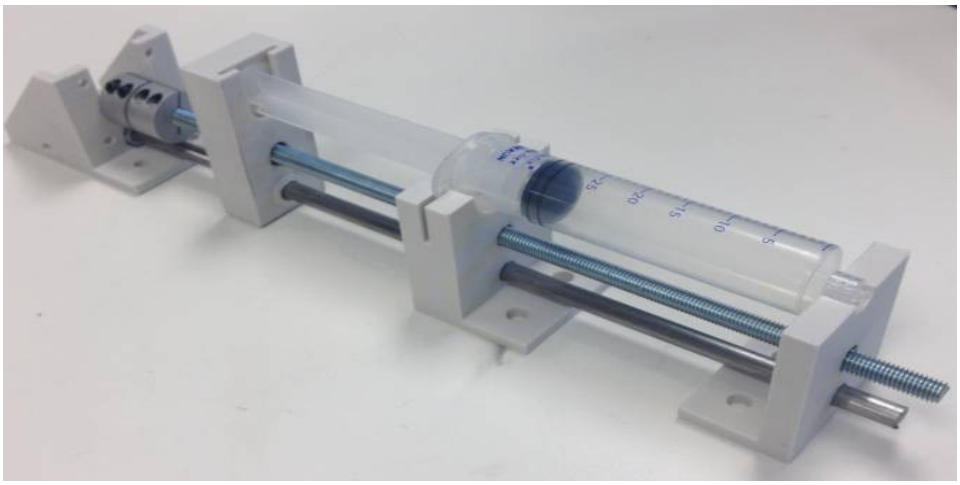


*Image 3 : Photo de la partie permettant de convertir le mouvement de rotation en mouvement de translation*

# IV. Aspect physique

Au cours du calcul des aspects physiques on a défini le débit massique du médicament mis dans le pousse-seringue en milligrammes de substance active (masse) par millilitres de liquide (volume). Connaissant le volume du liquide introduit et le temps mis à l'écoulement, on peut déterminer le débit volumique en ml/h. On peut modifier le débit à l'aide du pavé numérique qui entraîne une fonction modifiant la pression dans la seringue comme vu au-dessus.

- Débit de la seringue = volume / temps = vitesse x surface
- Volume maximal : 30 ml, diamètre de la seringue : 24 mm
- Vitesse = distance / temps → 8 tours de la vis du seringue = 1 cm ⇒ 1 tour = 1/8 cm
- Embout (diamètre) → 2 mm x Vf = Vi x 24 mm où Vf = vitesse finale et Vi = vitesse initiale

Les incertitudes en mesurant les distances sont les erreurs de mesure/lecture +/- 0,5 mm. Les incertitudes en mesurant les volumes sont les erreurs de mesure qui dépendent de la calibration du pavé numérique et de la précision d'affichage du débit.

Les erreurs possibles qui pourraient affecter notre pousse-seringue : l'expiration des parties cruciales pour le fonctionnement du pousse-seringue comme l'écran LCD, l'endommagement, le changement spontané du code si le moteur est mis en contact avec de l'eau par exemple.

# V. Modélisation 3D

La modélisation nous a permis d'assembler les différents composants du produit, ainsi que d'imprimer les supports en 3D et la connexion des mécanismes.

Pour les supports intérieurs en plastique, on a utilisé l'imprimante 3D. On a utilisé les fichiers « .stl » pour ses pièces ont été trouvés en ligne pour une pousse-seringue open source(voir annexe A pour la modélisation 3D).

La découpeuse laser a été utlisée pour fabriquer les supports extérieurs en bois. La base pour la boîte était faite par une application sur internet qui s'appelle "Makercase" avec les dimensions de notre pousse-seringue. Cette boîte est utilisée pour couvrir le moteur et les parties électroniques ainsi qu'être le cadre pour l'écran LCD et le pavé numérique.

# VI. Programmation

La partie de la programmation nous a donné la possibilité de faire marcher les mécanismes en utilisant des codes sur l'Arduino. Les connexions des fils, le breadboard, le moteur, l'écran et le pavé numérique aussi font partie de la programmation.

## 1) Le moteur

Composants:

- Moteur à pas, modèle : NEMA 17 42HS4013A4 SUMTOR
- Driver Pololu DRV8825
- Alimentation universelle à 12V
- Capacité à 100µF

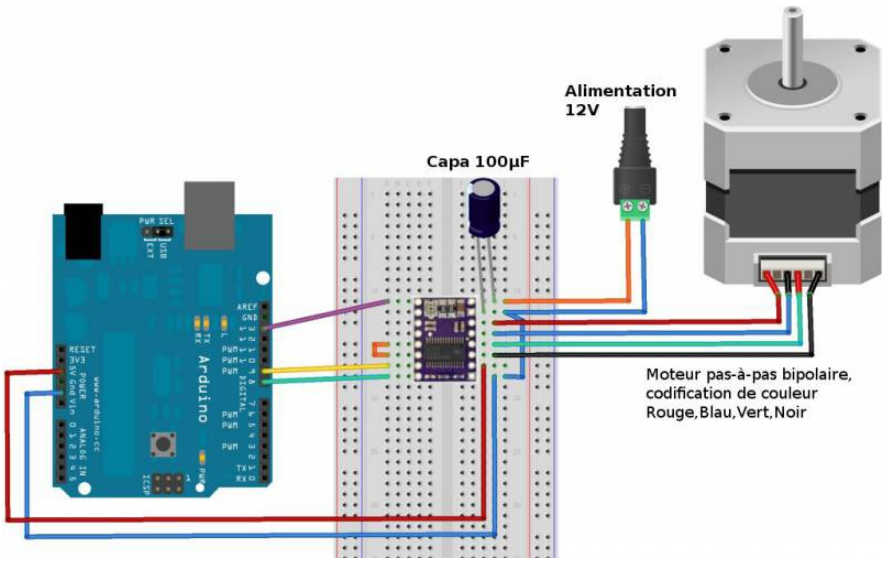



*Image 4 : Le montage du moteur (source : http://mchobby.be/wiki/index.php?title=DRV8825)*

## 2) L'écran LCD

L'écran LCD (modèle 1602) est connecté à l'Arduino en utilisant des fils et un breadboard. Un potentiomètre est nécessaire pour que l'écran LCD marche. Le téléchargement du dossier <LiquidCrystal.h> est aussi obligatoire pour que le code marche.

Composants:

- Ecran LCD : modèle 1602

- Potentiomètre

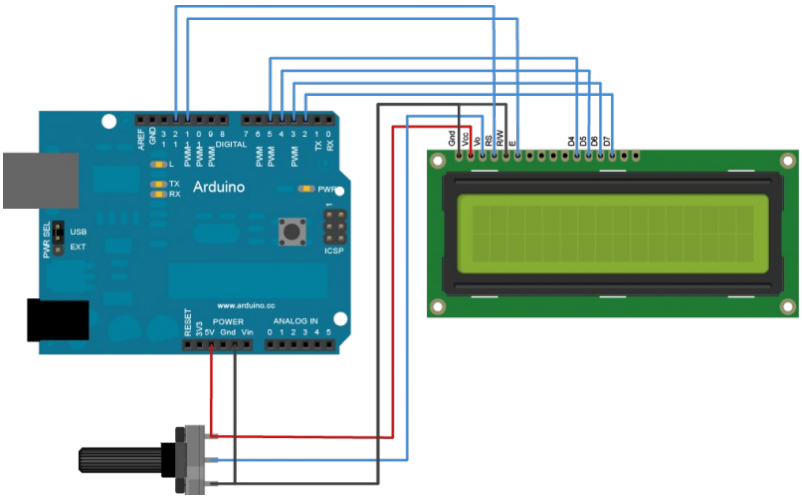



*Image 5 : Le montage de l'écran LCD (source : http://wiki.funlab.fr/Arduino_LCD_16x2ligne_parallèles)*

## 3) Le pavé numérique

Le pavé numérique (matrice 4×4) peut être connecté directement à l'Arduino. Pour la démarche du programme, on doit télécharger de dossier <Keypad.h> avec le "keypad.zip" et l'ajouter dans la bibliothèque du programme Arduino.

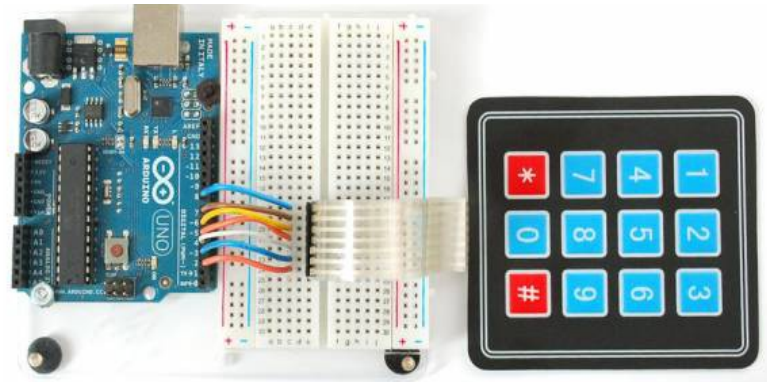


*Image 6 : Le montage du pavé numérique (matrice 4x4)*
*(source : https://cdn-shop.adafruit.com/large/membranekeypad34arduino_LRG.jpg)*

## 4) Code final

D'abord, on a fait un code d'une calculatrice en combinant les codes de l'écran LCD et le pavé numerique. Avec celle-ci on avait ajouté le code du moteur qui est spécifié pour notre driver. Avec ceci on a pu tourner le moteur avec la valeur que l'on a calculé en faisant soit l'addition, la soustraction, la multiplication ou la division. Ensuite, grâce au calcul fait dans les parties de l'aspect physique et mécanique, on a entré la valeur de pas pour avancer 1ml de notre seringue. Donc, on a fait un code qui demande initialement de mettre le débit volumique en saisissant le volume total à pousser et le temps réquis pour faire 1ml, le volume divisé par le temps est égal au débit volumique. C'est à dire, on entre le débit en saisissant le temps. Après, en affichant la valeur de pas que le moteur fera, il nous demande la

direction que le moteur va pousser, le sens positif pour avancer et le sens négatif pour réculer. L'action de la pousse-seringue est finit quand le écran devient vide et on peut recommencer tout de suite.

Du fait de la différence des composants de ceux utilisés sur le site open syringe (l'écran LCD comportait un pavé à 5 touches dans le projet open source), on a dû modifier le code final pour qu'il marche avec notre matériel.

Le code complet:

```
#include <LiquidCrystal.h>
#include <Keypad.h>

int x;
#define BAUD (9600)

LiquidCrystal lcd(7, 8, 9, 10, 11 , 12);
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','+'},
  {'4','5','6','-'},
  {'7','8','9','*'},
  {'X','0','=','/'},
};
byte rowPins[ROWS] = {53,51,49,47}; //connect to row pinouts
byte colPins[COLS] = {45,43,41,39}; //connect to column pinouts
Keypad myKeypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

//variables declaration
boolean valOnePresent = false;
boolean next = false;
boolean final = false;
boolean calcul_termine = false;
String num1, num2;
int ans = 200;
int temps = 0;
int direction = 0;
char op;

void setup(){
  pinMode(6,OUTPUT); // Enable
  pinMode(5,OUTPUT); // Step
  pinMode(4,OUTPUT); // Dir
  digitalWrite(6, HIGH); // Set Enable low & eliminate the motor's initial vibration noise
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("Debit Volumique");
  delay(2000);
```

```
 lcd.clear(); //clears the LCD screen and positions the cursor in the upper-left corner.
 Serial.begin(BAUD);
}

void loop(){
 char key = myKeypad.getKey();
  if ((calcul_termine == false) && key != NO_KEY && (key=='1'||key=='2'||key=='3'||key=='4'||key=='5'||key=='6'||
key=='7'||key=='8'||key=='9'||key=='0')){
    if (valOnePresent != true){
      num1 = num1 + key;
      int numLength = num1.length();
      lcd.setCursor(15 - numLength, 0); //to adjust one whitespace for operator
      lcd.print(num1);
    }
    else {
      num2 = num2 + key;
      int numLength = num2.length();
      lcd.setCursor(15 - numLength, 1);
      lcd.print(num2);
      final = true;
    }
  }
  else if ((calcul_termine == false) && (valOnePresent == false) && (key != NO_KEY) && (key == '/' || key == '*'
|| key == '-' || key == '+')){
    if (valOnePresent == false){
      valOnePresent = true;
      op = key;
      lcd.setCursor(15,0); //operator on right corner
      lcd.print(op);
    }
  }
if ((calcul_termine == false) && (final == true) && (key != NO_KEY) && (key == '=')) {
  if (op == '+'){

      ans = num1.toInt()*425  ;
      //1ml = 425steps
      temps = num2.toInt();
    }
      lcd.clear();
      lcd.setCursor(15,0);
      lcd.autoscroll();
      lcd.print(ans);
      lcd.noAutoscroll();
      calcul_termine = true;
      valOnePresent = false;
      final = false;
}
if (calcul_termine) {
    delay(2000);
```

```
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Direction");
    key = NO_KEY;
    while ( key == NO_KEY) {
    key = myKeypad.getKey();
    }
    if (key == '+') {
    direction = 1;
    }
    else if (key == '-') {
      direction = 2;
    }
}
if ((key != NO_KEY) && (key == 'X')){
    lcd.clear();
    valOnePresent = false;
    final = false;
    num1 = "";
    num2 = "";
    ans = 0;
    op = ' ';
  }
if (direction == 1){
     lcd.clear();
    lcd.print("Direction +ve");
      digitalWrite(6,LOW); // Set Enable low
        digitalWrite(4,HIGH); // Set Dir high
        for(int x = 0; x < ans; x++) // Loop ans times
      {
          digitalWrite(5,HIGH); // Output high
          delay(temps);
          digitalWrite(5,LOW); // Output low
          delay(temps);
       }
        delay(1000); // pause one second
     direction = 0;
     calcul_termine = false;
     lcd.clear();
     num1 = "";
    num2 = "";
    ans = 0;
    op = ' ';
    digitalWrite(6,HIGH); // Set Enable low
    }
else if (direction == 2){
     lcd.clear();
    lcd.print("Direction -ve");
      digitalWrite(6,LOW); // Set Enable low
```

```
      digitalWrite(4,LOW);  // Set Dir low
      for(int x = 0; x < ans; x++) // Loop ans times
     {
        digitalWrite(5,HIGH); // Output high
        delay(temps);
        digitalWrite(5,LOW); // Output low
        delay(temps);
      }
      delay(1000); // pause one second
      direction = 0;
      calcul_termine = false;
      lcd.clear();
      num1 = "";
   num2 = "";
   ans = 0;
   op = ' ';
      digitalWrite(6,HIGH); // Set Enable low

   }
}
```

# VII. Communication et coordination

## 1) Avancement des tâches

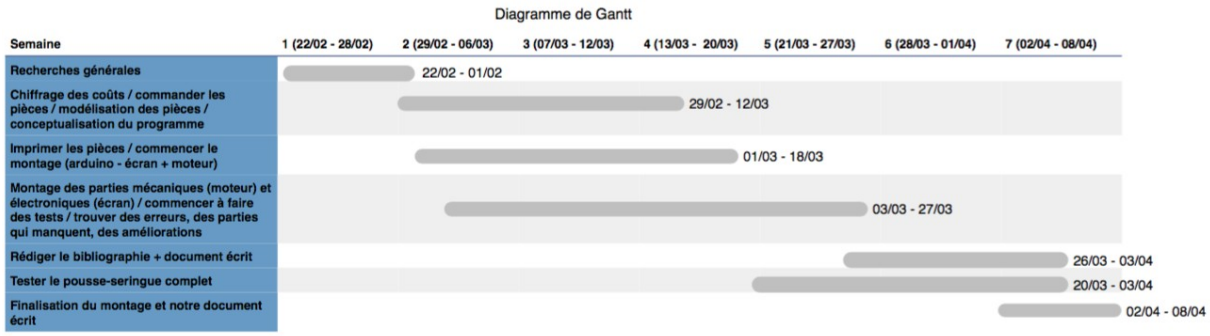



*Image 7 : Diagramme de Gantt*

Au début du projet nous avons bien avancé par rapport aux recherches générales et le budget car on a trouvé une pousse-seringue open source qu'on pouvait utiliser pour créer le notre.

Dans la deuxième semaine, on a pu imprimer quatres pièces 3D grâce à cet open source qui avait déjà la modélisation des pièces en ligne. En plus, pendant la deuxième et la troisième semaine on a réussi à faire marcher l'écran LCD, le pavé numérique et le moteur à pas avec les matériaux disponibles au FabLab. C'était pendant cette période de deux semaines qu'on a aussi commandé nos pièces pour notre projet.

En attendant nos commandes, on a commencé à faire des expériences avec la programmation du moteur, l'écran LCD et le pavé numérique. Dans la quatrième semaine, on a réussi à écrire un code de calculatrice

qui permettait d'afficher ce qu'on a tapé avec le pavé numériqe sur l'écran LCD et de faire des calculs simples, comme l'addition et la soustraction.

On a reçu une partie de nos commandes à partir de la cinquième semaine. Donc, on a pu commencer à monter notre pousse-seringue. On a soudé tous les fils qu'on avait besoin et on a connecté la tige, le shaft coupler et le moteur ensemble. Pour tester si les fils marchaient, on a utilisé notre Arduino DUE et des lumières LEDs pour être sur qu'ils marchaient.

Dans la sixième et la septième semaine on a reçu tous nos commandes manquant. On a monté les pièces imprimées sur notre rail avec les tiges et la seringue. De plus, on a créé une boîte en bois pour la partie électronique de la pousse-seringue, qui contient l'Arduino DUE, l'écran LCD, le breadboard et les fils. Pendant cette semaine on a fait les connexions de fils finales pour le moteur, le pavé numérique et l'écran LCD.

Tout ce qui restait à completer était le programme final dans la huitième semaine. On a fait beaucoup d'essais, mais on n'a pas pu réussir jusqu'a ce qu'un étudiant de master en informatique nous a aidé avec notre programme.

## 2) Difficultés rencontrés

### a. Trouver la(les) source(s) d'erreur(s)

Quand on a commencé à connecter les fils, le breadboard et le moteur on a eu des problèmes parce que le moteur ne marchais pas. De plus, on a fait exploser une capacité parce qu'elle n'était pas mise dans le bon sens. Donc, suite à ça, on a appris comment correctement placé la capacité et le driver. On a essayé plusieurs méthodes pour trouver ce qui ne marchait pas. On a changé les fils, on a changé de moteur, on a soudé les fils et on a changé de programme. On a réussi à le faire marcher à la fin. C'était une combinaison des fils et le programme qui ne marchait pas.

### b. Trouver les pièces exactes

Pour notre pousse-seringue, on a utiliser une pousse-seringue open source qu'on a trouver en ligne. On a utilisé les fichiers des pièces 3D à imprimer sur le site. Comme il y avait des trous pour mettre des tiges dedans, il fallait qu'on trouve des tiges de même taille que celles sur le site qu'on a trouvé. C'était un défi à les trouver car on a cherché sur plusieurs sites, mais à la fin on a utilisé les tiges trouvées dans le FabLab qui étaient exactement la taille que l'on cherchait.

### c. La modélisation et la programmation

D'abord, les connexions devraient être faites avec précaution. On avait fait exploser une capacité comme on ne savait pas qu'il fallait faire attention au sens de capacité. Ensuite, le dossier pour Arduino DUE n'est pas le même que celui de Arduino Uno. Il fallait qu'on télécharge le dossier qu'on a trouvé après des recherches et plusieurs essais avec le programme d'Arduino. D'ailleurs, notre moteur et driver ont chauffé plus que le cas d'utilisation normale. Il était également difficile de trouver le problème. L'intensité créée par notre alimentation externe était trop forte. D'où, on a utilisé un ampèremètre pour régler la tension et donc l'intensité avec le potentiomètre de notre driver. De plus, l'étape d'assemblage

des composants a été difficile, notamment la partie de programmation qui consistait de trouver la combinaison du programme de écran LCD et le pavé numérique avec celle de moteur.

# VIII. Conclusion

On a finalement réussi à répondre aux besoins auquel le pousse-seringue doit subvenir. En effet, l'utilisateur choisit un volume et un débit, en rentrant le temps pour pousser 1ml, pour lesquels le fluide va etre poussé.

Néanmoins, la précision de notre pousse seringue n'est pas optimale pour des quantités de fluide très faible (inférieur à 1/425 $\approx$ 0,0024ml = 2,4 $\mu$ l). Pour avoir une précision plus élevée, on aurais pu prendre un moteur avec plus de pas par tour (ex : avec 400 pas, la précision est doublée). On aurait aussi pu miniaturiser notre circuit en personalissant un circuit imprimé avec la machine C.I.F.

# IX. Bibliographie

## Général


- https://en.wikipedia.org/wiki/Syringe_driver
- https://hackaday.io/project/1838-open-syringe-pump
- https://www.wevolver.com/gerrit.niezen/openpump-an-open-source-hardware-syringe-pump/openpump/description/


## Modélisation


- https://github.com/naroom/OpenSyringePump/tree/master/3D%20Files
- https://productrealization.stanford.edu/resources/processes/laser-cutting


## Programmation

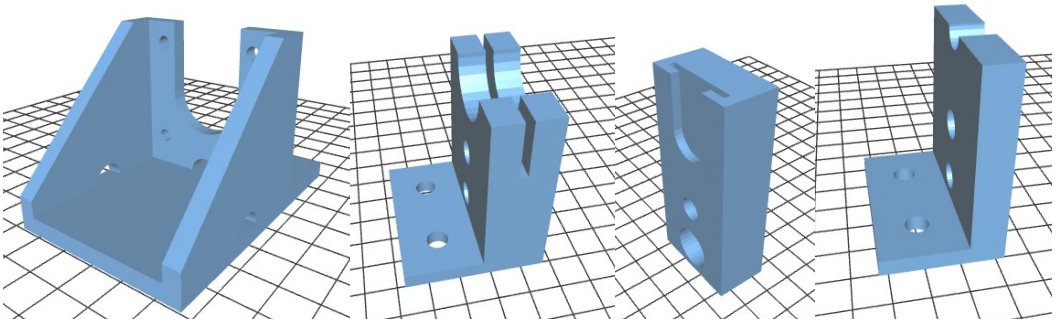

- http://mchobby.be/wiki/index.php?title=DRV8825
- http://www.dreamdealer.nl/tutorials/connecting_a_1602a_lcd_display_and_a_light_sensor_to_arduino_uno.html
- http://playground.arduino.cc/Main/LiquidCrystal
- http://wiki.funlab.fr/Arduino_LCD_16x2ligne_parallèles
- https://cdn-shop.adafruit.com/large/membranekeypad34arduino_LRG.jpg
- http://www.vathsav.com/post/arduino_calculator.html

# Annexe

## A. Budget

| Pièces | Quantité | Prix |
|---|---|---|
| Stepper Motor | 1 | 20€ |
| Alimentation (universelle) | 1 | 18.91€ |
| Arduino DUE | 1 | 43.02€ |
| Écran LCD (1602) | 1 | 3.89€ |
| Pavé numérique (Membrane 4×4 Matrix Keypad) | 1 | 3.74€ |
| Motor shield/driver (DRV8825) | 1 | 7.95€ |
| Fils (bobine) | 1 | 10.06€ |
| Seringue (30ml) | 1 | 6€ |
| Rail | 1 | 14.50€ |
| Breadboard | 1 | 9.19€ |
| Capacité = condensateur (100 µF) | 5 | 1.09€ |
| Shaft coupler | 1 | 6.67€ |
| Threaded rod | 1 | pris du FABLAB |
| Nuts, bolts, vis | | pris du FABLAB |
| Interrupteur | 1 | 1.40€ |
| Potentiometre | 1 | 6.59€ |
| | | |
| **TOTAL** | | 153€ |

## B. Modélisation des pièces 3D



*Les fichiers ".stl" des pièces de support intérieur en plastique (source : https://github.com/naroom/OpenSyringePump/tree/master/3D%20Files)*