Projet d'Arduino Peip 2

Année scolaire 2018-2019

Ball and Plate

Etudiants : Baptiste Schall & Yassine Waldane

Encadrants : Pascal Masson & Nassim Abderrahmane

Ecole Polytechnique Universitaire de Nice Sophia-Antipolis,

1645 route des Lucioles, Parc de Sophia Antipolis, 06410 BIOT

REMERCIEMENTS

Remerciements aux encadrants, Pascal Masson et Nassim Abderrahmane pour nous avoir fourni du matériel et nous avoir aidé pendant les séances d'Arduino. Bien évidemment merci au responsable du Fablab pour la mise à disposition d'outils très utiles à l'aboutissement de notre projet ainsi que pour sa disponibilité.

SOMMAIRE

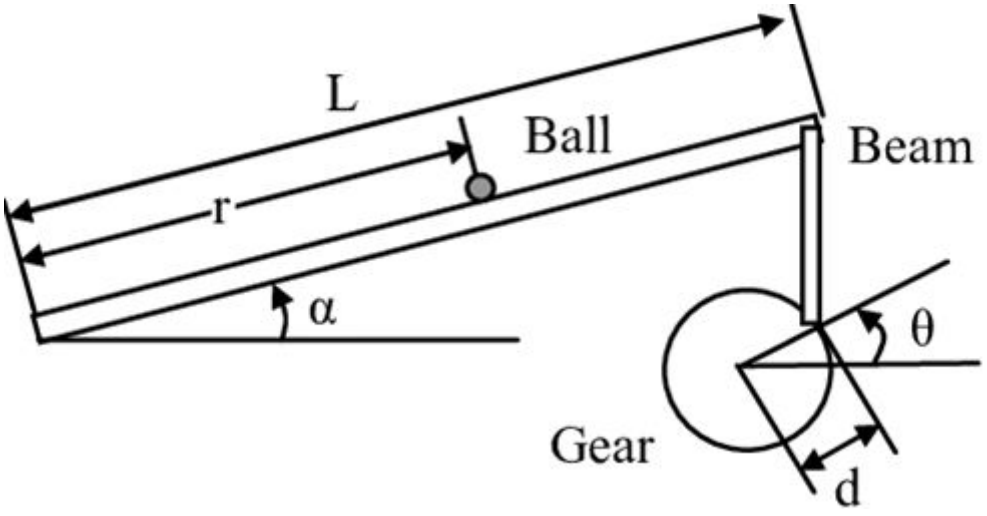
Introduction

Nous voulons réaliser une plateforme auto-stabilisatrice.
Ce système devrait être capable de déplacer n'importe quel élément sphérique vers n'importe quel point de la surface avec une précision de ± 1 mm.
L'intérêt de ce projet est la compréhension en profondeur de ce genre de système, qui trouvent leurs applications dans l'industrie de tous les jours comme dans les simulations par exemple .
Ce rapport est composé de trois parties. Tout d'abord nous allons présenter notre projet d'un point de vue technique, nos attentes/objectifs. Puis nous parlerons de la réalisation, notre organisation, des difficultés rencontrées ainsi que les solutions apportées. Nous finirons par, la partie résultat avec: l'aspect visuel et l'aspect code .

I. Cahier des charges

- Objectifs :

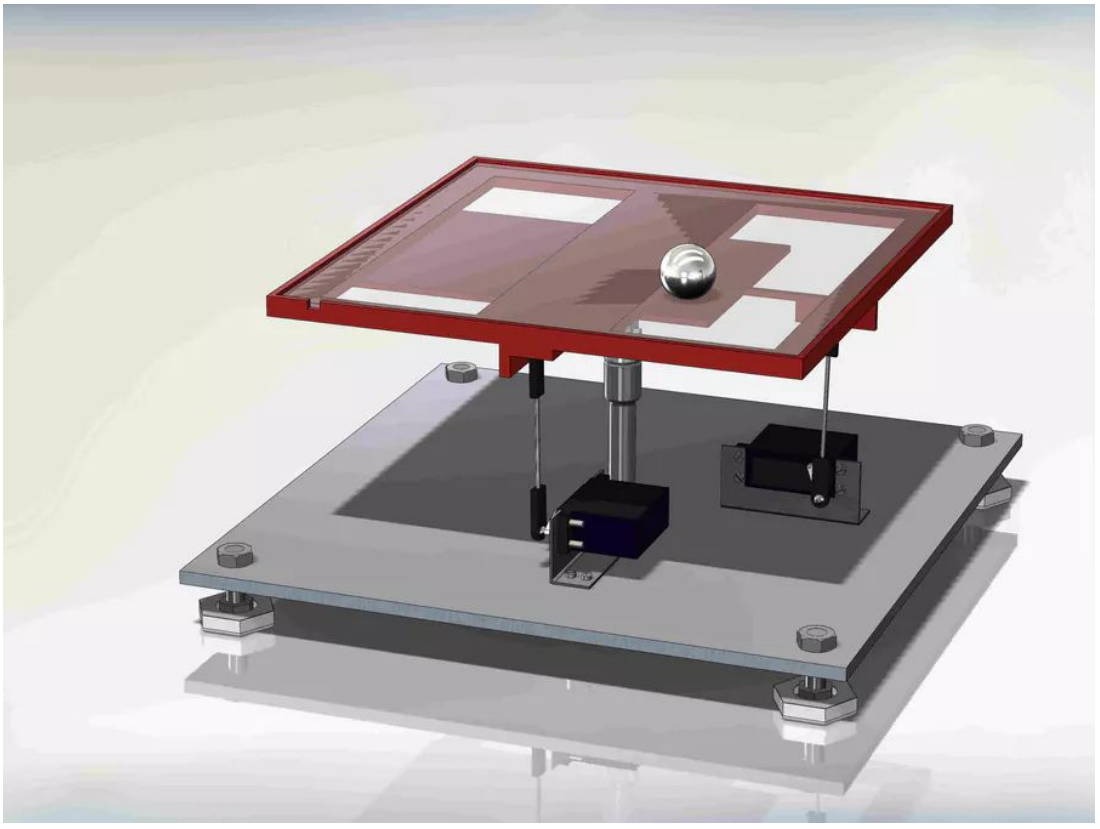
Le système Ball on Plate est l'extension en deux dimensions du système Ball on Beam qui est un système d'équilibrage, dont l'objectif est de positionner une bille à un endroit donné le long d'un axe.



- Le projet consiste en une plaque dirigée par deux moteurs, le but étant bien évidemment de conduire une bille qui roule librement sur la plaque, vers une position spécifique et de la maintenir dans cette position, ou faire suivre à la bille une trajectoire prédéfinie avec le moins d'erreur et le plus rapidement possible.
- Potentiellement si le projet est fini en avance l'écran capacitif (fonction tactile mais pas d'affichage) pourra être remplacé par un écran tactile classique, afin d'implémenter des jeux dans notre projet.
- Description fonctionnelle des besoins :
    - Fonction principale : Centrer une bille placée à un endroit donné.
    - Sous-fonctions :
        - Détection de la bille sur l'écran
        - Mise en place d'un système de contrôle manuel des moteurs via une manette Bluetooth
        - Utilisation du PID
        - Suivi de trajectoire

- Pour cela :
    - Détection de la plaque : écran capacitif
    - Mécanisme d'activation : Servos moteurs x2
    - Mécanisme de contrôle : PID

- Maquette finale escomptée:



- Ressources :
    - Deux Servomoteurs
    - Un écran capacitif de 7 pouces
    - Des bielles
    - Un pilier central
    - 3 Liaisons rotules
    - Module Bluetooth
    - Alimentation pour les Servos
    - Un support pour écran
    - Une carte Arduino, des câbles ….

- Bibliothèques:
    - <SoftwareSerial.h> : Pour la partie communication avec notre manette
    - <PID_v1.h> : Pour pouvoir utiliser toute les fonctions impliquants le PID

    (Dont la plus importante , PID.Compute qui effectue tous les calculs)
    - <TouchScreen.h> : Pour pouvoir utiliser les fonctionnalités du TouchScreen (comme .getPoint permettant de récupérer la position)
    - <Servo.h> : Pour pouvoir utiliser les fonctionnalités des servomoteurs

- Délais :

  Début de la recherche du projet, le 26 octobre, et rendu la semaine du 20 mars donc approximativement 5 mois.

I.2. Principe

LE PID

Afin de mener à bien ce projet nous avons dû comprendre le principe du PID, qui est un régulateur énormément utilisé de nos jours dans l'industrie. Celui-ci permet de grandement améliorer les systèmes en leur permettant de trouver leurs point d'équilibre le plus rapidement possible.
Le PID , autrement dit le "proportional, integral , derivative" est composé de trois modules comme l'indique son nom.

Nous allons aborder chacun de ces modules:

Proportional :

Est une composante proportionnelle à l'erreur qui sépare l'état actuel de notre système et son état d'équilibre.

Ce module permet donc d'adapter la correction à apporter par rapport à l'état de notre système.

Integral :

C'est la somme des écarts selon le temps où l'ajustement se fait. Plus le temps est long plus la correction apportée par unité de temps est grande.
Quand le système s'approche de l'état d'équilibre le module derivative intervient.

Derivative :

L'erreur est dérivée selon le temps. Cela permet de réduire les oscillations autour du point d'équilibre et donc de réduire le temps de stabilisation de notre système.

Les trois modules du PID sont complémentaires et permettent un gain de temps et d'efficacité pour tous les systèmes ayant besoin d'équilibre ou de répondre rapidement à des consignes.

L'écran:

L'écran capacitif est composée de deux plaques. Lorsqu'une pression est exercée, celle-ci met en contact les deux plaques. Cela referme un circuit et donne donc la représentation analogique de la position touchée. On peut assimiler l'écran à deux potentiomètres qui mesurent pour l'un la position en x et l'autre en y de la bille sur l'écran.

Le principe général:

Dans notre cas nous avons utilisé deux PID, l'un gère la position en x et l'autre la position en y de la bille, afin de recentrer la bille sur un point donné en paramètre.
La pression qu'exerce la bille sur l'écran capacitif envoie ses coordonnées aux entrées analogiques de l'arduino.
Le PID fait donc les calculs afin d'asservir les deux moteurs.
Selon les coordonnées transmises par l'écran, un certain angle est donné à chacun des moteurs afin que la bille s'approche de la position d'équilibre prédéfinie.

II Réalisation

II.1 répartitions des tâches

Il fallait déterminer un emploi du temps auquel on se fierait afin d'être le plus productif possible, chaque membre du groupe ayant des tâches précises à réaliser.
Dès la deuxième semaine un emploi du temps fut mis en place :

Planning											
Activité	Répartition	Semaines									
		1	2	3	4	5	6	7	8	9	
Recherche du matériel et des méthodes techniques afins de créer une maquette.	à deux										
Commandes (ebay)	à deux										
Création modèle de chassis	Yassine										
Comprendre le PID	Baptiste										
Asservissement des moteurs	Yassine										
Fabrication du chassis	Yassine										
Codage de l'écran	Baptiste										
Assemblage de la maquette	à deux										
Tests	à deux										

Un emploi du temps assez simpliste, où chaque tâche dure en moyenne deux séances, sauf la partie compréhension du PID, qui, au moment de la réalisation de l'emploi du temps était la partie qui nous paraissait la plus ardue. Au niveau de la répartition des tâches, nous avons opté de scinder le projet en deux parties, la partie maquette et la partie code, chaque membre du groupe ayant une partie de prédilection.

II.2 Rapports:

Dans cette partie nous allons résumer nos activités pendant les séances d'Arduino et en dehors de celle-ci:
Les détails de nos activités sont bien évidemment présents sur notre Github
(* = difficultés rencontrées à ce moment)

07/12/2018 :

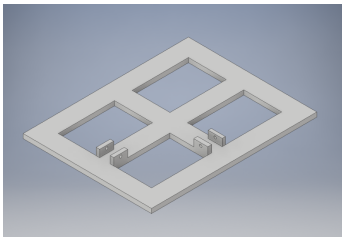
- Pris connaissance de la manière de réaliser le projet.
- Déterminé précisément, le matériel à utiliser afin de le commander dans la semaine (quel type/taille d'écran, type de moteurs: pas à pas ou servomoteurs).

14/12/2018 :

- Commande des servomoteurs et de l'écran.
- Planification, réalisation de l'emploi du temps global.

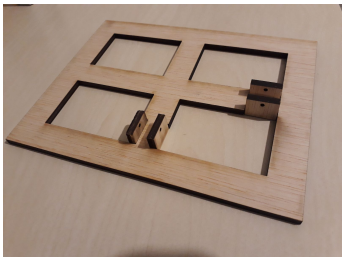
18/12/2018 :

- Compréhension des principes liés au PID.
- Création du premier châssis sur Inventor.



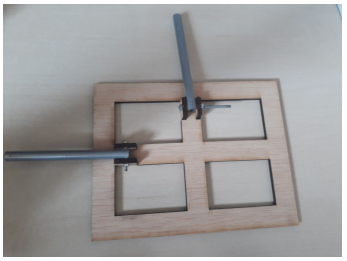
11/01/2019 :

- Programme pour tester l'écran.
- Début de la partie communication.
- Première découpe pour le support de l'écran.



18/01/2019 :

- Mapping de l'écran.
- Premier modèle de bielle (en bois).
- Programme test des servomoteurs.

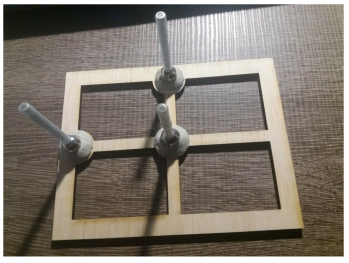


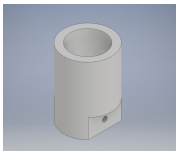
06/02/2019

- A partir de cette séance nous nous sommes mis à travailler à deux sur la maquette afin de la finir le plus tôt possible.
- Trouvé une alimentation pour les servomoteurs.
- Réalisation de nouvelle bielle (geomag)*.
- Mise en place des liaisons rotules *.

13/02/2019 :

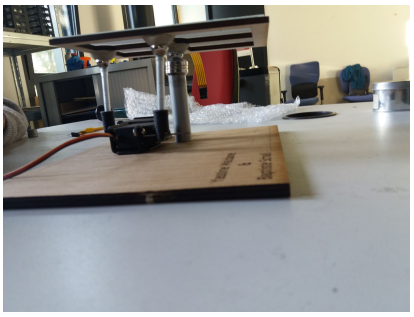
- Réalisation d'un nouveau support pour l'écran.
- Montage : Bielles + Liaisons rotules + support écran.
- Mise en place et fixations des servomoteurs.
- Schématisation des liaisons bielles/servos.





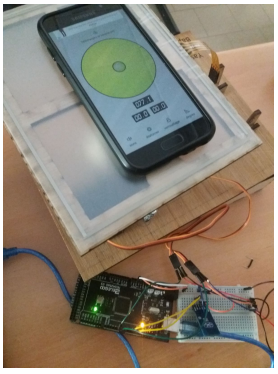
28/03/2019

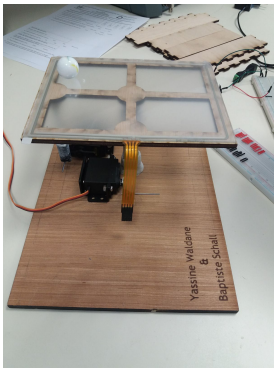
- Impression en 3D des liaisons bielles/servos *.
- Création du pilier central.



06-07/03/2019

- Finitions de la maquette: Fixations et soudures.
- Création d'un compartiment pour y ranger la partie électrique.
- Branchement propre de l'électronique.
- Equilibrage*.
- Premier test*.





La suite :

- Le reste du projet est essentiellement du code, qui sera expliqué dans le III.2 ***

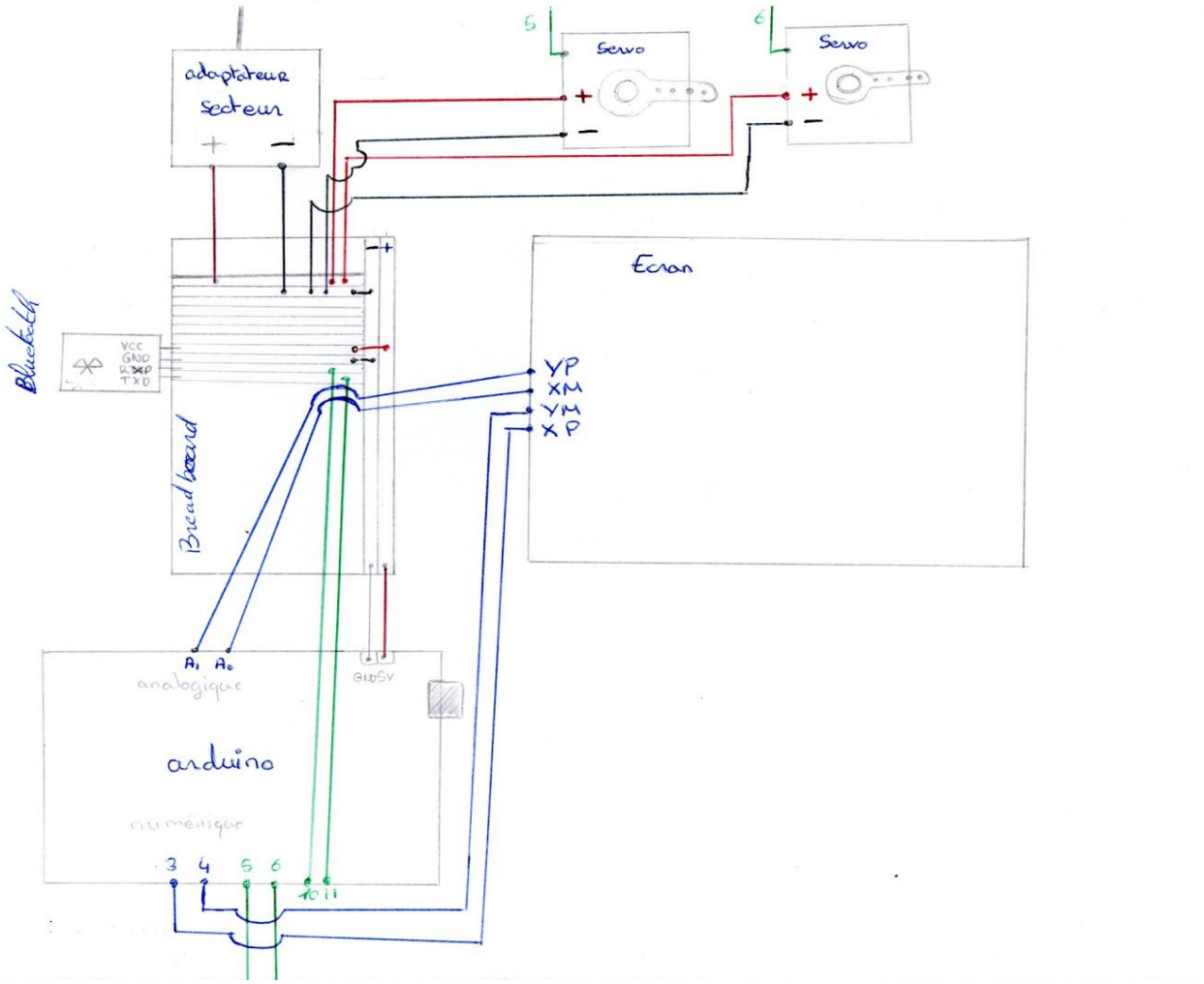
III Résultat:

III.1 Montage et Partie maquette :

Montage

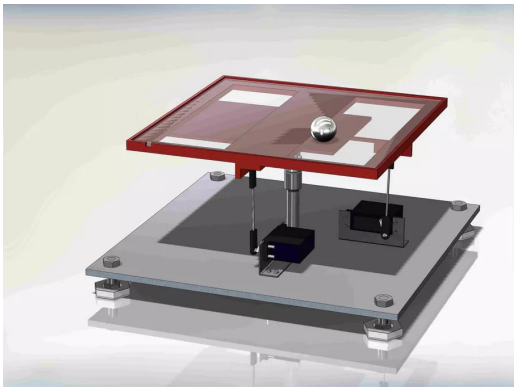
Il nous a fallu lier la partie capteur (écran et bluetooth) et la partie traitement des données (arduino) ainsi que la partie exécutive (moteurs) sans oublier l'alimentation du circuit.

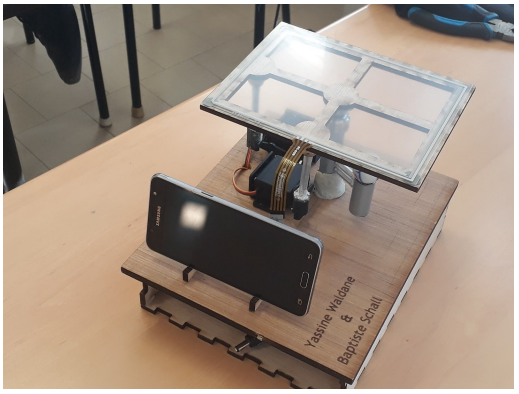
Voici le schéma de notre montage:



L'important est de ne pas oublier le fil joignant les deux masses (celle de l'Arduino et celle de l'adaptateur secteur). Sur la maquette chaque fil est codé afin de faciliter le montage.

Maquette





Au final, la maquette correspond pleinement à nos attentes.

- Aucune restriction au niveau des mobilités, grâce au choix de nos liaisons, aucune contrainte n'a été négligée.

- Très peu de câbles apparents car toute la partie électronique se trouve dans le compartiment dédié. Cela apporte beaucoup au niveau esthétique.

- Un bon équilibrage, la maquette est parfaitement stable, il faut juste modifier les angles des moteurs à l'initialisation car la surface sur laquelle la maquette n'est pas toujours à niveau.

III.2 Partie Code :

Tout d'abords nous allons lister et expliquer tous les codes intermédiaires utilisés durant ce projet:

1) Servo + Potentiomètre :

```
#include <Servo.h>


Servo s1;
Servo s2;

int valeur1 ;
int val1 =0;
int val2=0;
int valeur2 ;

void setup() {
  Serial.begin(9600);
  s1.attach(10);
  s1.write(0);

  s2.attach(11);
  s2.write(0);
}

void loop() {

  val1 = analogRead(A10);
  valeur1= map(val1,0,1023,0,50);
  Serial.println(valeur1);
  s1.write(valeur1);


  val2= analogRead(A11);
  valeur2=map(val2,0,1023,0,50);
  Serial.println(val2);
  s2.write(valeur2);
```

2) Servo + Bluetooth

```
void setup() {
  Serial.begin(9600);
  BlueT.begin(9600);

  s1.attach(5);
  s2.attach(6);
  Initial();

}

void loop() {
  if (BlueT.available()){
       Data=BlueT.read();
  if (Data=='A'){
        valeur1 =BlueT.parseInt();
        if(valeur1<70){
          s1.write(valeur1);}
        }
  if (Data=='B'){
        valeur2 =BlueT.parseInt();
        if(valeur2<70){
          s2.write(valeur2);
          }
        }
  }
```

Le premier programme a servi à tester la réactivité des servomoteurs, en effet pour notre projet, nos servos se doivent d'être très réactifs. On note que dans ce programme les angles des servos sont donnés par des potentiomètre, dans le code final ces angles seront donnés par des calculs effectués par le PID.

Le deuxième programme est une mise en place du mode manuel dans notre projet, les angles des servos sont donnés par des sliders présents sur la manette bluetooth

```
#include <TouchScreen.h>
#include <stdint.h>


#define YP A0
#define XM A1
#define YM 3
#define XP 4
TouchScreen ts = TouchScreen( XP, YP, XM, YM,300);
//float convertX = 0.01515;
//float convertY = 0.01;

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

}


void loop() {
  TSPoint p = ts.getPoint();
  if (p.z > ts.pressureThreshhold ){ // Si la pression est supérieur au seuil minimum.
    double valX= p.x;
    double valY = p.y;
   valX=map(valX,66,965,10,172);
   valY=map(valY,102,925,10,130);


    Serial.print("bois there's ball ");
    Serial.println((valY));
    Serial.println((valX));
```

3) TouchScreen

Ce programme permet de tester l'écran capacitif, si il détecte bien la bille. On note qu'au départ la bille que nous avions choisi était trop légère et n'était pas détectée par l'écran.

L'écran nous renvoie des valeurs variants de 66 à 965 en X et 102 à 925 en Y
d'où l'utilisation de la fonction map afin que les valeurs affichées soient en mm .

4) CheckScreen

```
boolean CheckScreen(){
  TSPoint p = ts.getPoint();
  if (p.z > ts.pressureThreshhold ){
    return true;
  }
  return false;
}
```

5) Timer

```
if (CheckScreen()){
    temps=0;}
else{ temps +=1;
      Serial.println(temps);
      delay(500);
      if(temps==300){

        s1.detach();
        s2.detach();
      }
    }
```

Le programme 4, est une simple fonction qui renvoie un booléen en fonction de la présence ou non de la bille sur l'écran. Cette fonction est utile pour mettre en place un système de timer. Si la bille n'est pas sur l'écran, on commence à compter, au bout d'un certain temps, on éteint les servomoteurs car le projet n'est plus utilisé, si il y a une bille sur l'écran alors le timer est réinitialisé.

6) Le PID controller, le code final

```
PID myPIDY(&InputY, &OutputY, &SetpointY, Kpy, Kiy, Kdy, DIRECT);
```

```
void setup()
{ BlueT.begin(9600);

  Serial.begin(9600);
  servoX.attach(5);
  servoY.attach(6);
  // Make plate flat
  OutputX=56;
  OutputY=28;
  servoX.write(OutputX);
  servoY.write(OutputY);
   p = ts.getPoint();

  SetpointX = 90.0;
  SetpointY = 70.0;

  myPIDX.SetMode(AUTOMATIC);
  myPIDX.SetOutputLimits(10, 80);
  myPIDY.SetMode(AUTOMATIC);
  myPIDY.SetOutputLimits(10, 65);

  myPIDX.SetSampleTime(Ts);
  myPIDY.SetSampleTime(Ts);
  delay(100);
```

Tout d'abords dans le setup, on commence par rendre la surface sur laquelle la bille va être déposée plate grâce à cette partie, puis on définit le point auquel on veut se rendre en mm, ces Setpoints sont des paramètres pour le PID, Input Y sera la position de la bille , OutputY l'angle donné, Setpoint ou la bille doit aller et Kpy,Kiy,Kdy des constantes déterminées par l'expérience.
Dans la partie, on met les calculs de PID en mode Automatic, ils s'effectuent avec comme intervalle de temps Ts (Sampletime), SetOutputLimits permet de limiter les angles donnés au servos.

Pour la détermination des constantes Kp,Ki et Kd, on réalise des expériences sur la maquette.

Méthode de détermination des constantes Ki,Kd,Kp

Pour ce projet, deux Pid différents sont utilisés donc il y a six constantes à régler, on ne peut s'occuper des six simultanément, on fait donc au cas par cas.
D'abord on choisit un axe, Y ou X , admettons qu'on choisisse l'axe X. Le Setpoint en Y est : 70mm on placera donc toujours la bille le long de l'axe Y donc à 70mm du bord afin que le setpoint en Y soit directement atteint et que seul les calculs en X importent.

Pour la constante Proportional on place simplement la bille à la distance maximal du centre (en X) et on observe si l'angle donné au servo X est trop ou pas assez grand, s'il donne ou pas trop de vitesse à la bille.

Nous avons fait plusieurs tests et nous avons décidé de négliger la constante Integral, en effet celle ci agissant en fonction du temps et notre expérience étant très courte, elle n'est pas d'une grande utilité.

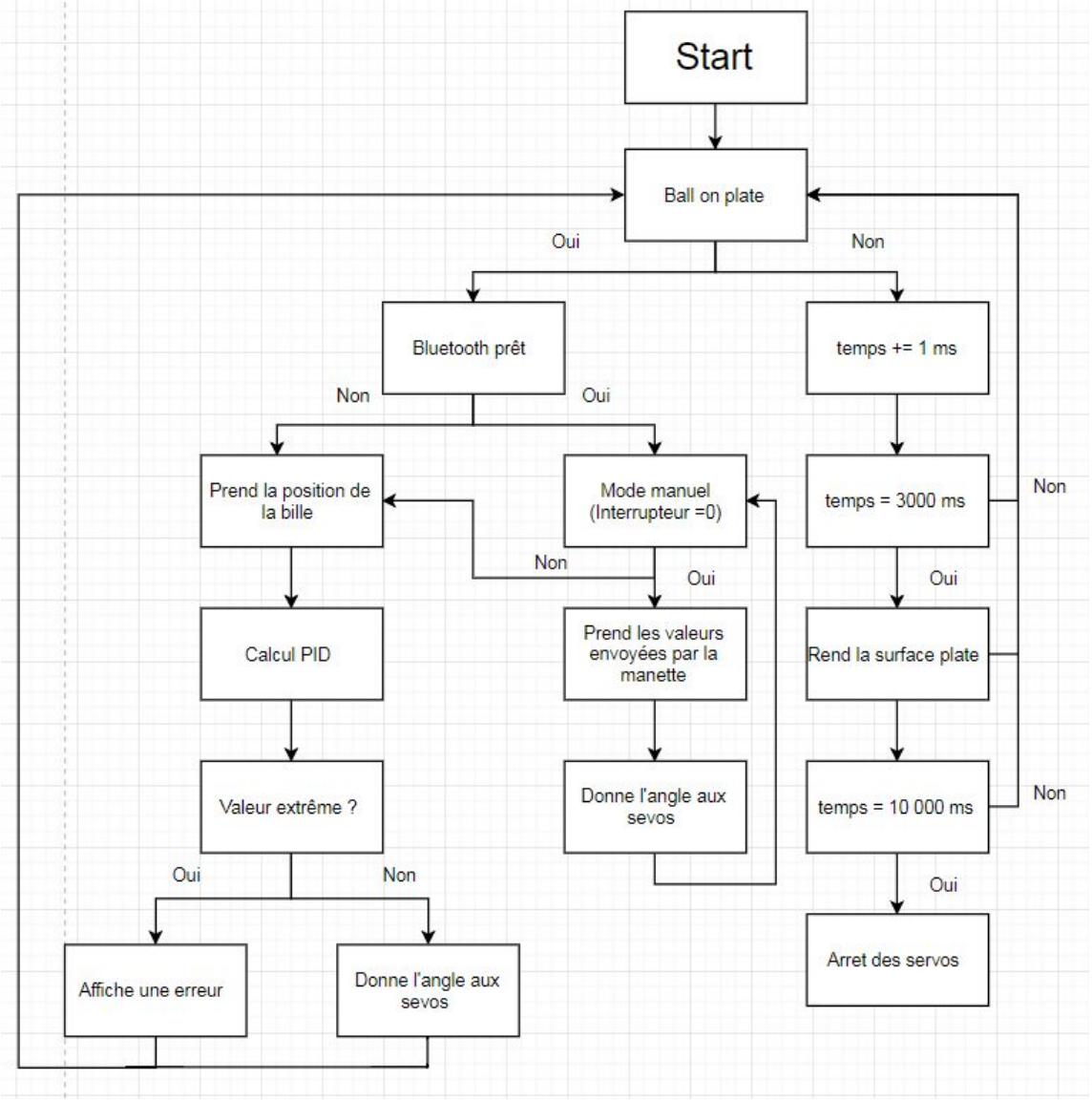
Pour la constante Derivativ, on observe la décélération de la bille quand celle ci s'approche du centre et on règle en fonction des résultats.

Pour le PID en Y, on procède de même, la détermination des constantes des PID est une étape assez longue car elle nécessite beaucoup d'expériences

Le reste du programme respect cette algorithme :

Il comprend des programmes déjà mentionnés et expliqués, comme le mode manuel et le Timer, pour ce qui est des calculs, ceux ci sont résumés par un simple appel de fonction.

```
 myPIDX.Compute();
myPIDY.Compute();
```

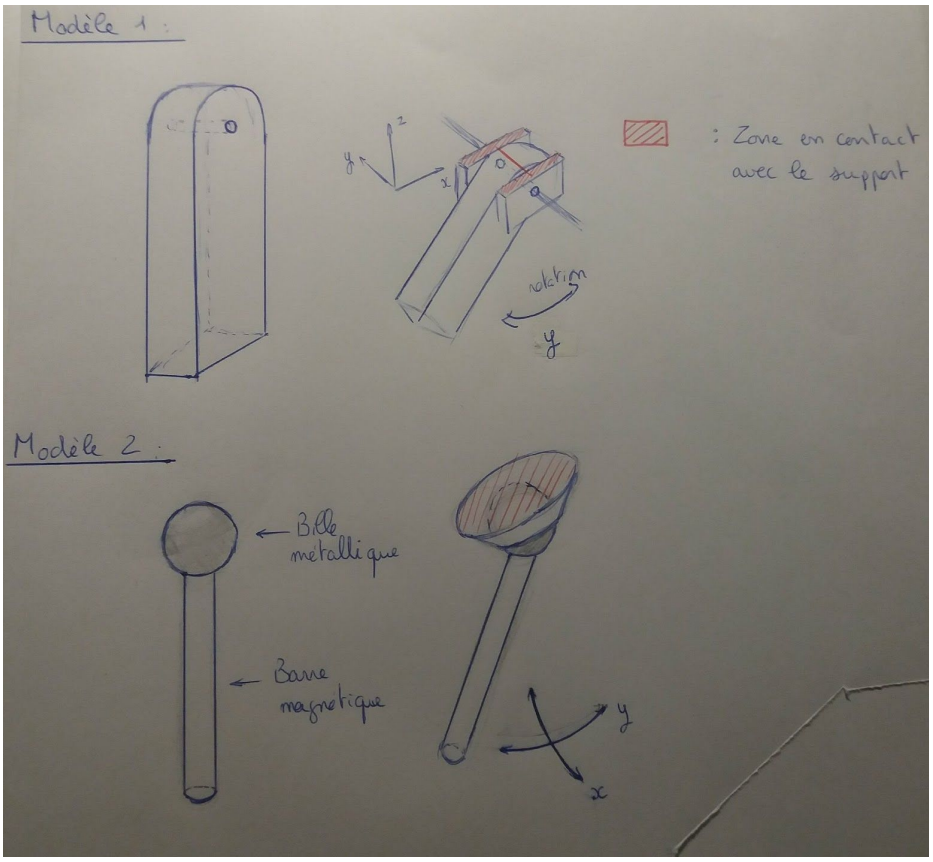


III.3 Difficultés rencontrées et solutions apportées :

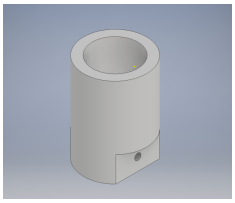
- La création des bielles :

Dans notre projet, le fait que la plaque ait toutes les mobilités est primordial, il nous fallait donc des liaisons rotules reliants les bielles et le support écran. Le problème étant qu'au départ, nous n'avions pas pris en compte cette contrainte.

Schéma:



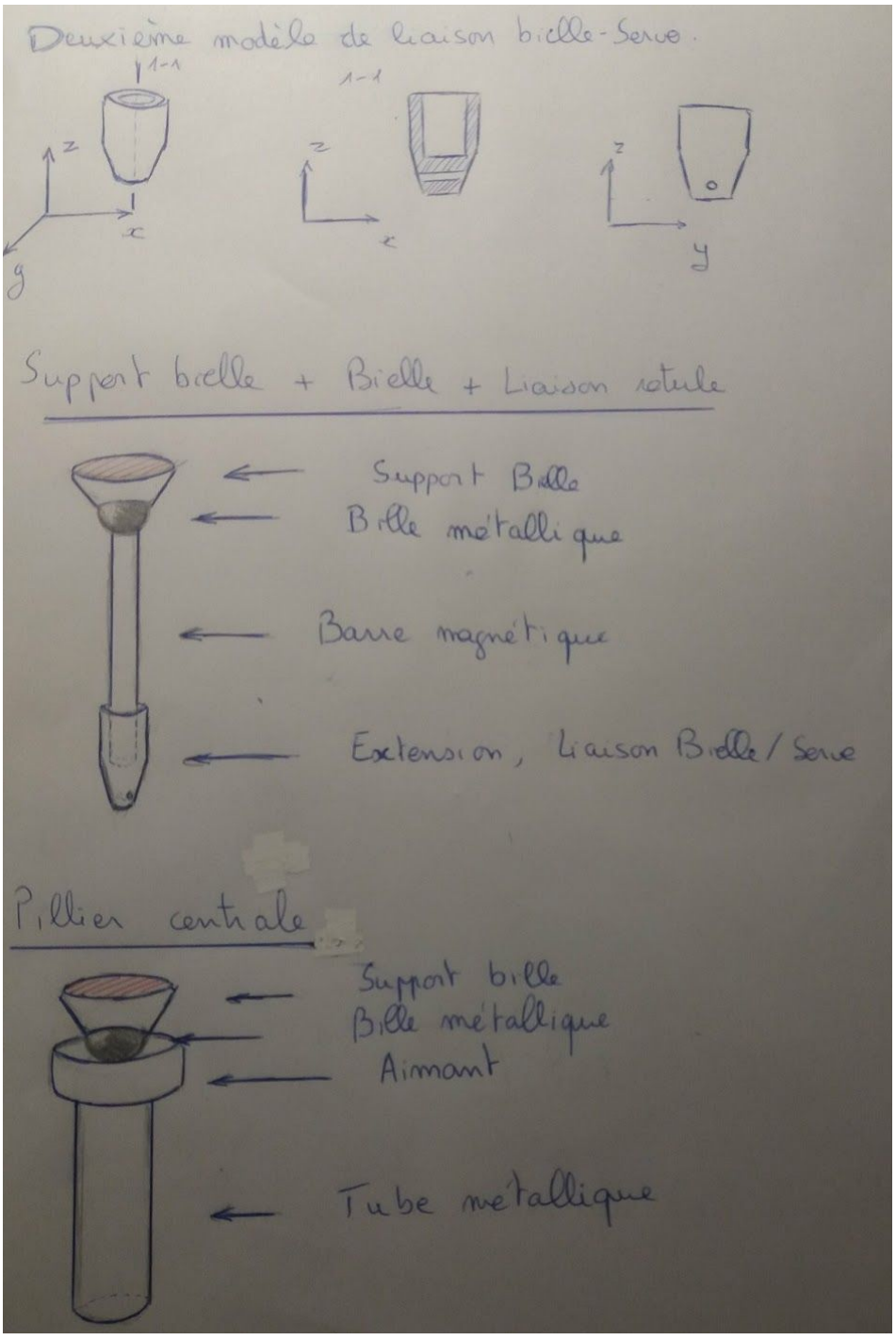
Les nouvelles bielles (modèle 2) permettent une rotation selon x, y et z (z pas nécessaire). Celles-ci cependant, soulèvent un nouveau problème, les barres magnétiques n'étant pas perçables, pour les relier aux servomoteurs, nous nous devons de créer des extensions: Les liaisons Bielle-Servos, qui doivent être des liaisons pivots.



Cette pièce n'était pas réalisable avec la découpe laser, nous avons donc dû l'imprimer en 3D.
L'imprimante 3D mise à notre disposition au Fablab est d'une grande imprécision, nous avons donc dû nous y prendre à plusieurs reprises, ce qui nous a fait prendre du temps.

De plus les liaisons Bielle-Servos ne correspondait pas à nos attentes,en ponçant les anciennes pièces on obtient un modèle plus convenable.

Schéma des différentes pièces:



- Equilibrage et premier test:

L'équilibre est au coeur de notre projet, avoir un support parfaitement droit est obligatoire, de ce fait nous avons accordé beaucoup de temps à la mesure des pièces afin que celles ci soient exactement au même niveau: bielle+extension+support , pilier+support.

Le problème est que le support (table) sur lequel nous poserons notre projet ne sera pas tout le temps droit, 1 degrés d'écart pourrait fausser la démonstration.
Une solution était envisageable, cependant manque de temps nous n'avons pas pu la mettre en place (voir conclusion).

Durant nos premier test, nous avons vite remarqué le principale point fort de notre maquette, les servomoteurs étaient très réactifs, ce ne serait donc pas un frein pour la suite.
Le problème est lui venu durant le test du mode manuel avec la manette bluetooth, à la connexion de notre manette, des valeurs excessivements grandes étaient envoyées à l'arduino, un problème assez simple à régler, néanmoins qui a bien failli endommagé nos servomoteurs, en effet, en leur donnant des valeurs trop grandes ceux ci butés contre le socle en bois.

- <u>L'écran capacitif:</u>

Bien que tous les problèmes furent réglés, un nouveau problème fit son apparition, en effet durant la phase de test de l'écran (touch Screen) celui ci fonctionnait parfaitement, cependant durant la dernière semaine des bugs à son égard sont apparues, peut-être à force de le transporter.

Durant les tests du PID, la stabilisation de la bille ne fonctionnait pas dû à des grands accoups des servomoteurs, au début nous pensions que cela était dû à une mauvaise configuration du PID ou alors à une erreur dans les calculs de celui ci. Mais le problème était en fait liée à l'écran : celui ci se mit à nous renvoyer des valeurs incohérentes, par exemple -1mm, ce qui est impossible, ces valeurs étant des Input pour le contrôleur PID , faussaient alors les calculs, les angles donnés aux servomoteurs.
Dans un premier temps nous pensions que, comme pour les valeurs extrêmes du bluetooth il suffisait de ne prendre que les valeurs positives ou supérieur à un seuil, et de donner un angle aux servomoteurs que si les valeurs étaient cohérentes.
Cette solution n'était pas la bonne, en effet il arrivait assez souvent que, les valeurs renvoyées par l'écran soient incohérentes durant plusieurs loop, ce qui bloquait les calculs de PID et donc les servomoteurs trop longtemps, la bille tombait.

```
if(valY >0 && valX >0){
  myPIDX.Compute();
  myPIDY.Compute();

  testX= OutputX;
  testY= OutputY;

  if (testY ==0 || testX ==0){Serial.println("wtf");}
  if  (testX != 0 && testY !=0){
    Serial.println(testX);
    Serial.println(testY);
    servoX.write(testX);
    servoY.write(testY);}
```

```
55.91
39.40
55.91
39.40
wtf
wtf
wtf
wtf
wtf
wtf
wtf
wtf
wtf
wtf
```

Dans cet exemple le message 'wtf' correspond à une loop sans calcul dû aux valeurs incohérentes

Puis nous avons eu une autre idée, si les valeurs ne correspondaient pas, alors le programme recalcule la position de la bille, cependant étant donné le nombre de calcul demandés à l'arduino, les servos étaient ralentis

.

Au final ce problème n'a pas pu être résolu et compromet la fonctionnalité total de notre projet.

IV. Conclusion

Emploi du temps :

Planning											
Activité	Répartition	Semaines									
		1	2	3	4	5	6	7	8	9	
Commandes (ebay)	à deux										
Création modèle de chassis	Yassine										
Comprendre/coder le PID	Baptiste										
Asservissement des moteurs	Yassine										
Fabrication du chassis	à deux										
Codage de l'écran	Baptiste										
Tests finaux	à deux										

Voici le planning que nous avons finalement suivi, il y a plusieurs choses à remarquer.

Les Livraisons ebay, celles ci ont duré bien plus longtemps que prévu, retardant notre avancée dans le projet.

Le temps allouer à la maquette et au PID ont drastiquements changé, en effet le PID était la partie qui nous apeuré le plus, étant la plus complexe en façade, au final s'approprier le PID n'aura pris que deux séances si ce n'est moins, et au contraire la maquette aura pris bien plus de temps que prévu.

Sous-estimer le temps alloué à la maquette aura été une des plus grandes erreurs commises.

Améliorations possibles:

Notre projet est bien évidemment très largement améliorable, et nous avons plusieurs idées d'amélioration:

- Afin que initialisation des moteurs se fasse de manière automatique, un accéléromètre pourrait être utilisé, avec un programme associé pour que peu importe la surface sur laquelle notre projet est posé, il s'adapte sans préréglages
- Acheter un écran capacitif de meilleur qualité, car c'est l'élément principal de notre projet
- Remplacer l'écran capacitif par un véritable écran tactile, avec affichage, pour pouvoir réaliser des jeux.

Bibliographie:

Liens utilisés dans notre projet :
[1]: Projet nous ayant inspiré
https://www.instructables.com/id/3DOF-Ball-on-Plate-Using-Closed-Loop-Stepper-Motor/
[2]: Projet similaire
https://www.hackster.io/davidhamor/ball-and-plate-c4802
[3]: Projet similaire
http://www.matheraetsel.de/ball_1.html
[4]: Vidéo explicative (PID)
https://www.youtube.com/watch?v=0vqWyramGy8&list=LLTq9V_SHyzc7WeAzT649SWw&index=96&t=10s