

GYMNASE DE BURIER

JOHAN LINK
3M5

Système de stabilisation PID



Résumé du travail de maturité

Ce travail consiste en la réalisation d'un système permettant de faire tenir une balle en équilibre sur un plateau qui a la possibilité de s'incliner sur deux axes pour annuler les mouvements de la balle. En d'autres termes, si on lance une balle sur ce plateau, ce dernier s'inclinera pour l'empêcher de tomber et il positionnera la balle en son centre.

Partie matérielle

Le système est composé de deux parties bien distinctes : une base qui contient trois moteurs avec leurs bras sur lesquels se trouve le plateau et un support qui maintient une caméra au-dessus du plateau. La base qui contient les moteurs est une structure en plastique de forme circulaire et de même dimension que le plateau. Cette structure est composée de plusieurs pièces en plastique qui ont été modélisées dans un logiciel puis fabriquées à l'aide d'une imprimante 3D.

Circuit électronique

La base qui contient les moteurs comprend également un circuit imprimé qui permet de contrôler les moteurs et de communiquer avec un ordinateur. J'ai conçu ce circuit spécialement pour ce projet. Je l'ai dessiné à l'aide d'un logiciel, puis il a été fabriqué en chine. Le circuit contient un microcontrôleur similaire à celui de certaines cartes Arduino, par conséquent j'ai pu programmer mon propre circuit électronique à l'aide de l'environnement Arduino.

Programmation de l'ensemble

Le système fonctionne en trois temps : premièrement, la caméra prend une photo du plateau et de la balle et la transmet à un ordinateur, ensuite un programme Python analyse l'image pour détecter la position et la vitesse de la balle et en déduit que le plateau doit s'incliner d'un certain angle pour annuler les mouvements de la balle. Pour finir, le circuit imprimé fait bouger les moteurs en fonction des informations reçues par l'ordinateur. Le programme Python est la partie la plus importante du système car il traite les images reçues par la caméra, les analyse, puis en déduit à l'aide d'un régulateur PID l'inclinaison qu'il faut donner au plateau.

Utilisation d'un régulateur PID

Le régulateur PID permet de comparer la position réelle de la balle avec une position souhaitée. En prenant en compte plusieurs facteurs comme la position et la vitesse de la balle, on peut connaître avec précision l'inclinaison qu'il faut donner au plateau pour que la balle se stabilise à une position voulue.

Remerciements

Je souhaite remercier M. Salanon de m'avoir suivi tout au long de ce TM et pour la relecture de mon travail.

Table des matières

1	Introduction	4
1.1	Motivations	4
1.2	Présentation du projet	4
1.3	Les cartes Arduino	5
2	Conception de la partie mécanique	6
2.1	Fonctionnement général du système mécanique	6
2.2	Conceptualisation géométrique du problème	6
2.3	Choix des moteurs	7
2.4	Conception du design du système	8
2.5	Fabrication des différents éléments	9
2.6	Assemblages du système	11
2.7	Fabrication du support et du boîtier de la caméra	11
3	Résolution des équations à l'aide de Python	13
3.1	Rôle de ce programme	13
3.2	Fonctionnement du programme	13
4	Conception du circuit imprimé	16
4.1	Pourquoi réaliser un circuit imprimé ?	16
4.2	Cahier des charges	16
4.3	Qu'est ce qu'un microcontrôleur ?	16
4.4	Choix du microcontrôleur	17
4.5	Conception du PCB	17
4.6	Fabrication du PCB	18
4.7	Assemblage des composants	18
4.8	Initialisation du microcontrôleur	19
5	Fonctionnement d'un régulateur PID	21
5.1	Qu'est ce qu'un régulateur PID ?	21
5.2	La régulation Proportionnelle	21
5.3	La régulation Proportionnelle et Intégrale	22
5.4	La régulation Proportionnelle, Intégrale et Dérivée	22
6	Mise en place du régulateur PID dans le programme Python	23
6.1	Transformation des équations	23
6.2	Un exemple concret	23
6.2.1	Détermination des composantes I_x et I_y	23
6.2.2	Détermination de l'angle α	25
6.2.3	Détermination de l'angle β	26
6.2.4	Détermination des angles de chaque moteur	28

7 Utilisation de OpenCV avec Python	29
7.1 Qu'est-ce que OpenCV ?	29
7.2 Programme intermédiaire	29
7.3 Reconnaissance d'une couleur	30
8 L'interface graphique	31
8.1 Pourquoi faire une interface graphique ?	31
8.2 Fonctionnalités de l'interface	31
9 Le code Arduino	33
9.1 Communication avec le programme Python	33
9.2 Contrôle des servomoteurs	33
10 Conclusion	34
Bibliographie	35
Annexes	37
A solveEquation.py	38
B programmeIntermediaire.py	41
C interface.py	43
D arduinoCode.ino	55
E Détermination des angles θ des servomoteurs	57
F Quelques plans du projet	60
G Prototypes	67
H Quelques esquisses du projet	68

1 Introduction

1.1 Motivations

L'électronique, la programmation et l'informatique de façon générale sont des domaines qui m'intéressent depuis longtemps, c'est pourquoi j'ai souhaité réaliser un travail de maturité rassemblant toutes ces disciplines afin d'approfondir mes connaissances. J'ai toujours voulu savoir comment les objets ou les machines qui nous entourent fonctionnent c'est pourquoi j'ai l'habitude d'en démonter pour mieux comprendre leur mécanisme. J'ai commencé à m'intéresser à l'électronique il y a de nombreuses années, j'ai commencé par la fabrication de simples circuits électriques puis, plus tard, je me suis également intéressé à la programmation. Il y a quelques années j'ai découvert l'existence des cartes Arduino qui m'ont permis de réaliser des projets de plus en plus intéressants en me permettant de travailler à la fois du côté logiciel et matériel. Les cartes Arduino s'utilisent de manière très intuitive et sont entièrement open-source, de plus une énorme quantité de documentation est présente sur internet. Ce travail de maturité est donc l'occasion de faire un projet concret qui contient une partie hardware et une partie software.

1.2 Présentation du projet

Mon projet consiste en la réalisation d'un système permettant de faire tenir en équilibre une balle sur un plateau. Des servomoteurs permettront au plateau de s'orienter avec un certain angle d'inclinaison pour contrebalancer les mouvements de la bille. Une caméra placée au-dessus du dispositif transmettra en direct des images du plateau à un ordinateur qui sera chargé de les analyser pour en tirer des conclusions telles que la position de la bille, sa vitesse et son accélération. Un programme informatique sera alors chargé de traiter ces données et de les transmettre par USB à l'Arduino qui contrôlera la rotation de chaque servomoteur indépendamment. Le plateau bougera en conséquence des déplacements de la bille, même si cette dernière est déstabilisée par du vent ou une force quelconque, l'Arduino inclinera le plateau pour qu'elle ne tombe jamais.

Ce projet me permettra de concevoir un système contenant une partie mécanique et électronique, je serai également mené à coder un programme capable de traiter des images et d'en déduire la position d'un objet d'une couleur définie. Dans un premier temps je vais construire et étudier les différents mécanismes qui permettront de faire bouger le plateau. Ensuite je devrai dessiner et fabriquer un circuit électrique contenant un microcontrôleur capable de communiquer avec l'ordinateur et capable de diriger les servomoteurs. Je finirai mon travail par la réalisation du programme qui traitera les images de la caméra et qui en déduira l'inclinaison du plateau nécessaire à l'équilibre de la bille.

1.3 Les cartes Arduino

Les cartes Arduino sont de petites cartes électroniques qui sont principalement composées d'un microcontrôleur autour duquel se trouve de nombreuses entrées et sorties qui permettent à la carte d'interagir avec différents composants électroniques ou quelconques systèmes extérieurs à la carte. Plusieurs modèles de cartes existent, la différence majeure qui les sépare est la puissance du microcontrôleur. Par exemple un microcontrôleur contenu dans un Arduino Mega contiendra une EEPROM d'une capacité de 4 kB alors qu'un Arduino Uno aura une EEPROM limitée à 1 kB. De plus les cartes plus puissantes ont également plus d'entrées et de sorties.

L'Arduino a été fondé en 2005 par un étudiant Italien, par la suite ce simple projet d'étudiant a révolutionné l'univers de l'électronique. Ces cartes permettent à n'importe quelle personne de concevoir un projet à moindre coût. Un Arduino Uno coûte environ 20 euros ce qui rend ces cartes accessibles. Mais l'avantage d'utiliser ces cartes open-source est d'avoir accès à une immense communauté de personnes publant des tutoriels ou différents projets sur internet. De plus, Arduino fournit également un logiciel de programmation gratuit qui permet de programmer très simplement n'importe quelles cartes Arduino ou d'autres cartes avec un microcontrôleur similaire. En effet le logiciel Arduino peut également servir à programmer une carte d'une autre marque qu'Arduino ou encore une carte fabriquée par soi-même. C'est un exemple d'un des nombreux avantages que les logiciels open-source offrent.

2 Conception de la partie mécanique

2.1 Fonctionnement général du système mécanique

Le plateau sur lequel reposera la bille devra être capable de s'orienter sur deux axes et devra pouvoir s'incliner de 35° au maximum. Étant donné que la plate-forme doit se mouvoir sur deux axes j'ai rapidement commencé à réfléchir à un système comportant deux moteurs. Cependant, après réflexion je me suis rendu compte qu'il était plus simple d'utiliser trois moteurs (fig.1). L'ajout d'un moteur permet également de rajouter un degré de liberté au plateau, en effet avec trois moteurs la plate-forme peut s'incliner sur deux axes et peut également bouger de façon verticale, mais ce degré de liberté ne sera pas utilisé dans ce projet. À présent nous savons que le système comprendra trois moteurs placés à 120° l'un de l'autre et que le mouvement de rotation de ces derniers devra entraîner l'inclinaison du plateau. Pour ce faire, chaque moteur mettra en rotation un bras qui sera composé de deux parties reliées entre elles par une articulation (fig.2). Une bille métallique sera fixée à l'extrémité du bras de chaque moteur et cette bille fera office de rotule entre le bout du bras et le plateau situé en dessus.

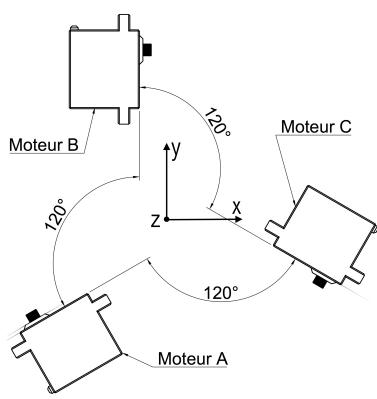


FIGURE 1 – Disposition des moteurs

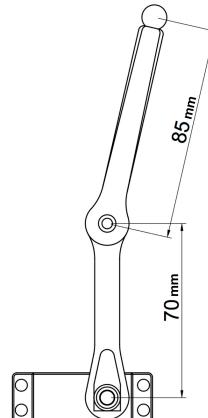


FIGURE 2 – Bras d'un moteur

2.2 Conceptualisation géométrique du problème

Comme dit précédemment, le dispositif comporte trois moteurs, il faut donc trouver un moyen de connaître l'angle du bras de chaque moteur en fonction d'une inclinaison donnée du plateau. Les moteurs ont une position fixe, chaque bras se déplace dans un plan vertical et la distance entre le bout de chaque bras est constante. Dans un premier temps j'ai commencé à réaliser différents calculs permettant de mettre en relation les différentes contraintes du problème. J'ai

également établi un système permettant de définir la position du plateau en fonction de deux angles. Pour ce faire j'ai ajouté dans mes calculs un vecteur \vec{v} (fig.3) de norme 1 qui est toujours perpendiculaire au plateau et dont l'origine correspond au centre du plateau. L'orientation de ce vecteur est défini par les angles α et β .

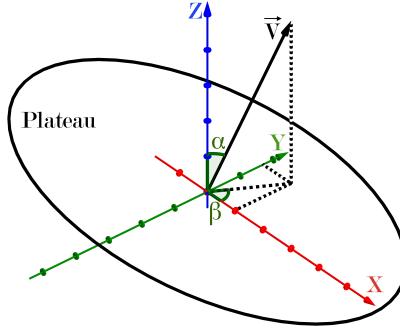


FIGURE 3 – Vecteur \vec{v} perpendiculaire au plateau

Après avoir réalisé plusieurs calculs¹ j'ai pu établir des équations qui permettent de déterminer la position de chaque bille située sur le bout des bras en fonction d'une inclinaison connue du plateau. Après avoir trouvé la position dans l'espace d'une bille on doit encore trouver l'angle du moteur qui permet de positionner le bout du bras avec la bille au bon endroit. L'équation² suivante permet de calculer l'angle θ que le bras doit avoir par rapport à l'horizontale :

$$(L - r \cos \theta - \sqrt{X^2 + Y^2})^2 + (r \sin \theta - Z)^2 = l^2 \quad (1)$$

où L est la distance entre l'arbre d'un moteur et l'origine (fig.1), r est la longueur de la première portion du bras de chaque moteur et l est la longueur du deuxième segment du bras (segment sur lequel est attaché la bille). X, Y et Z sont les coordonnées³ de la bille qui ont été calculées précédemment. L'équation (1) doit donc être utilisée individuellement pour chaque moteur, en d'autres termes on calcule d'abord la position de la bille du bras du moteur A et on utilise l'équation (1) pour déterminer l'angle du moteur A, ensuite on refait les mêmes calculs pour les moteurs B et C.

2.3 Choix des moteurs

Comme vu précédemment, trois bras composés de deux parties chacun devront maintenir et faire bouger le plateau situé en dessus d'eux. Il est donc

1. Ces calculs sont décrits aux pages 57 et 58.

2. L'équation 1 est décrite plus précisément à la page 59.

3. Le repère orthonormé de ces coordonnées se trouve sur le plan qui contient les trois moteurs et se trouve à équidistance de ces derniers. (figure 1, page 6)

évident qu'il est nécessaire d'avoir des moteurs pour mettre en mouvement chaque bras. Cependant mon projet contient plusieurs contraintes qu'il faut prendre en compte pour le choix des moteurs. Premièrement, la masse du plateau repose entièrement sur les moteurs, ils doivent alors être relativement puissants pour le soulever et le maintenir dans une certaine position. Deuxièmement, les moteurs doivent pouvoir incliner le plateau rapidement pour compenser les déplacements parfois rapides de la bille. Troisièmement, il faut que les moteurs soient précis pour que le système fonctionne correctement.

Il existe énormément de types de moteurs différents, cependant je me suis intéressé à deux types de moteurs en particulier : les moteurs pas-à-pas et les servomoteurs. Les servomoteurs sont rapides, puissants et assez précis, ils peuvent cependant effectuer des rotations de seulement 180° ou 360° au maximum mais cela n'est pas un problème pour ce projet. Ils sont très pratiques car ils sont relativement petits et sont très faciles à utiliser avec un Arduino. Je me suis également intéressé aux moteurs pas-à-pas car ils sont plus précis que les servomoteurs, cependant ils comportent quelques désavantages, par exemple, il faut utiliser des drivers⁴ pour les faire fonctionner avec un Arduino.

J'ai finalement décidé d'utiliser des servomoteurs pour ce projet car ils sont plus simples à utiliser et ont l'avantage de pouvoir maintenir leur position même si une force extérieure est exercée sur l'arbre du moteur. En effet les servomoteurs contiennent un circuit électronique qui compare la position réelle du moteur avec la position souhaitée, par conséquent ces moteurs réajustent leur position en continu et de manière automatique.

2.4 Conception du design du système

Cette partie consiste en la conception de l'armature qui devra maintenir les différents éléments tels que les servomoteurs, le circuit imprimé ou encore les divers connecteurs (connecteur pour l'alimentation, port USB et connecteur pour un joystick). J'ai commencé par modéliser les différentes pièces dans le logiciel *Fusion 360*. Le plateau a un diamètre de 27 cm, par conséquent le socle qui contient les moteurs fait également ce diamètre. La majorité des pièces vont être imprimées en 3D par la suite, malheureusement la majorité des imprimantes 3D ne sont pas capables d'imprimer des pièces aussi grandes, c'est pourquoi j'ai dû diviser plusieurs pièces en plusieurs petites pièces qui peuvent s'assembler à l'aide de boulons et d'écrous.

La figure (fig.5, page 9) permet de se rendre compte de toutes les différentes pièces qui composent le système. On aperçoit également trois disques, le disque se trouvant tout en haut correspond au plateau sur lequel roulera la bille, les deux autres disques forment respectivement le haut et le bas du boîtier qui contient les moteurs, le circuit électronique et les autres composants. Les pièces imprimées en 3D forment le pourtour du boîtier et certaines sont également présentes à l'intérieur et servent de support aux moteurs et au circuit.

4. Petits circuits électroniques contrôlant la rotation des moteurs pas-à-pas.

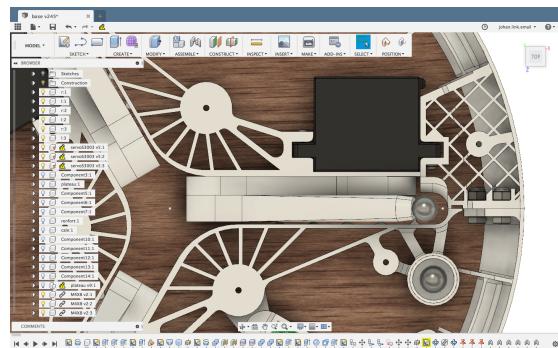


FIGURE 4 – Capture d’écran de mon projet dans *Fusion 360*

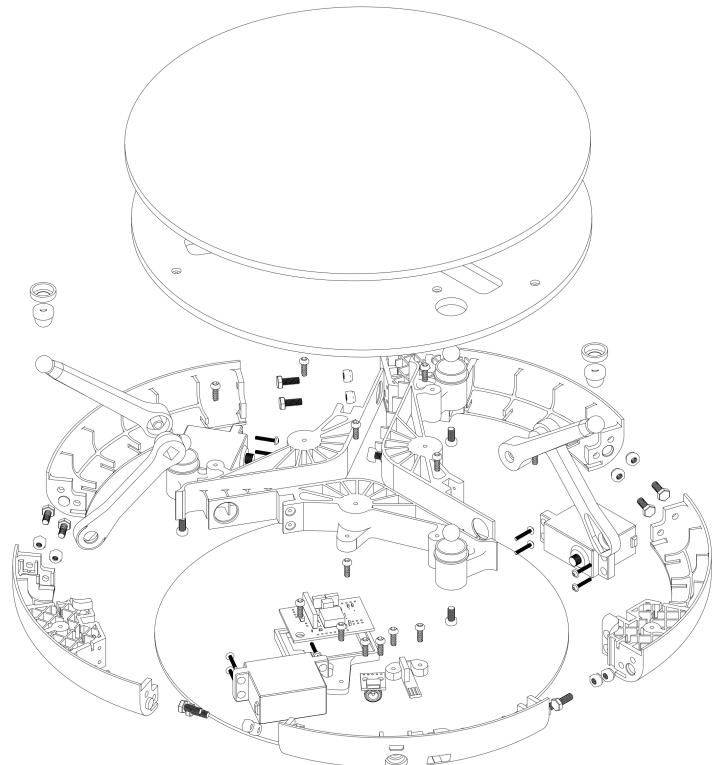


FIGURE 5 – Vue générale de tous les éléments du système

2.5 Fabrication des différents éléments

Étant donné que le système comprend un grand nombre de grands éléments à imprimer en 3D, j'ai décidé de faire appel au site *3D Hubs*. Il m'a suffi de

télécharger tous les fichiers de mes pièces sur leur site, puis ils se sont occupés d'imprimer les différents éléments et de les envoyer. De plus, les pièces que j'ai dessinées sont relativement complexes, il fallait donc une imprimante 3D qui dispose d'une grande précision pour pouvoir les fabriquer. Les pièces ont toutes été imprimées en ABS (acrylonitrile butadiène styrène). L'ABS est un plastique très courant dans l'industrie, il est souvent utilisé dans les appareils électroménagers ou encore dans les jouets comme par exemple les briques *LEGO*. Ce plastique comporte des caractéristiques intéressantes car il est plutôt léger et est très résistant aux chocs. Un de ses plus grands défauts est d'avoir tendance à jaunir lorsqu'il est exposé à la lumière du soleil.

Les trois disques que l'on voit sur la figure 5 (page 9) ont été fabriqué par l'*Atelier 12Mill*⁵. Il s'agit d'un atelier, basé à Lausanne, qui permet de faire fabriquer toutes sortes de pièces avec des techniques différentes. Deux des disques que j'ai fait fabriquer ont été fraisés dans une plaque en acrylique transparent. L'un de ces deux disques a été peint en blanc. Le troisième disque est le plateau sur lequel repose la bille, il a été fabriqué en carton car il fallait un matériau léger pour que le plateau n'ait pas trop d'inertie. Après avoir testé un plateau en bois, je me suis rendu compte qu'un plateau avec trop d'inertie avait tendance, lors de mouvements brusques, à se décrocher des bras magnétiques qui le supportent.

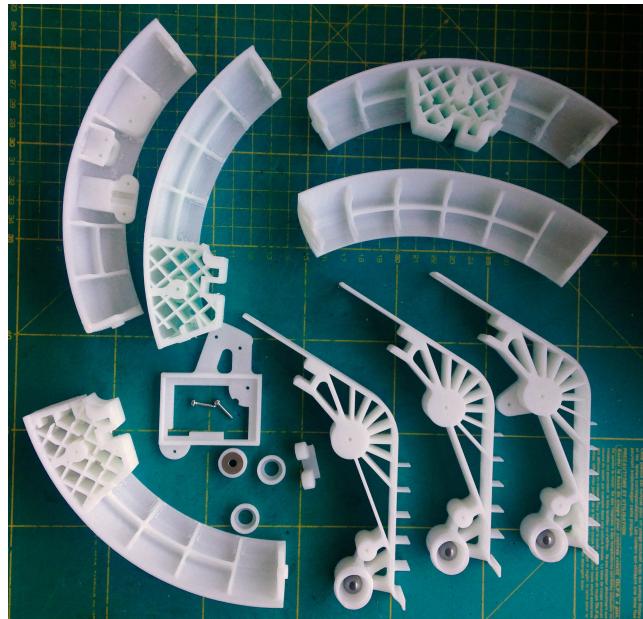


FIGURE 6 – Pièces imprimées en 3D

5. Adresse : Avenue de Sévelin 48 - 1004 Lausanne

2.6 Assemblages du système

La dernière étape de la construction consiste en l'assemblage de toutes les pièces du système. Tous les éléments s'emboitent et sont maintenus entre eux par des boulons et des écrous. Cette étape est relativement simple si les pièces ont été correctement dessinées au préalable. Dans mon cas l'assemblage s'est déroulé convenablement, bien que certaines pièces comportaient des défauts qui étaient dus à l'impression 3D. Ces petits défauts ont causé des désalignements, mais heureusement il est assez facile de modifier les pièces à l'aide d'un simple cutter.

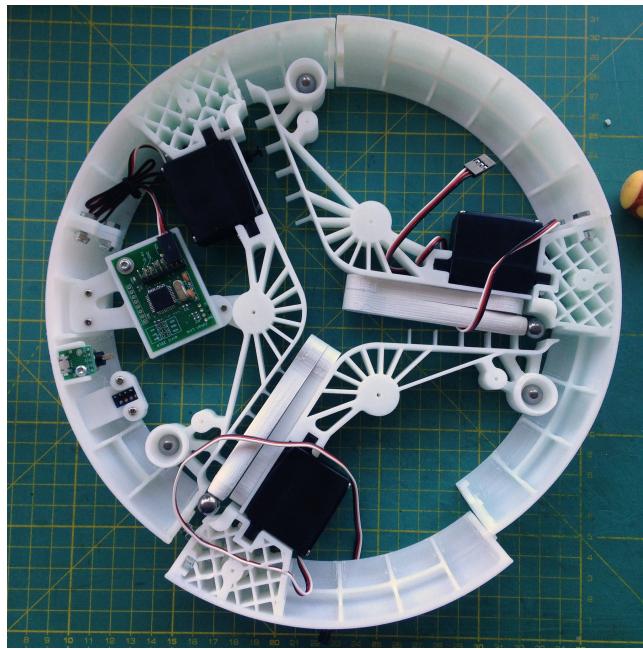


FIGURE 7 – Début de l'assemblage des pièces

2.7 Fabrication du support et du boîtier de la caméra

La caméra qui doit filmer le plateau est placée sur un support qui est complètement séparé de la base qui contient les servomoteurs. Le boîtier de la caméra est monté sur un tube en aluminium qui lui-même est emboité à un support qui contient trois aimants qui permettent à ce support de se fixer sur des surfaces métalliques. Le boîtier de la caméra ainsi que le support ont été imprimés en 3D. Le câble USB de la caméra passe à l'intérieur du tube en aluminium et sort à l'arrière du support aimanté.

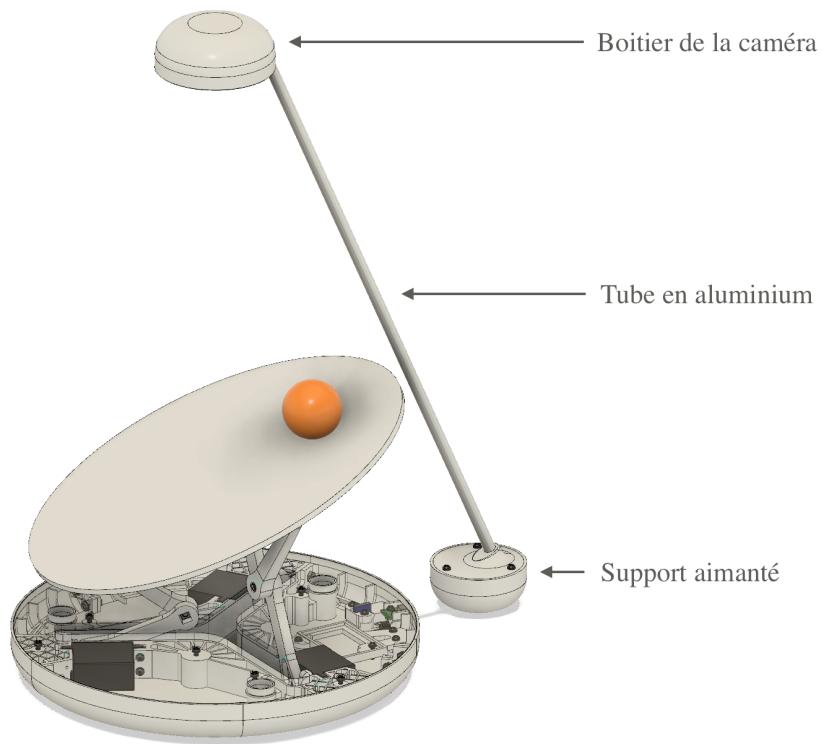


FIGURE 8 – Capture d'écran du support de la caméra dans *Fusion 360*



FIGURE 9 – Caméra utilisée pour ce projet

3 Résolution des équations à l'aide de Python

3.1 Rôle de ce programme

Les équations présentées dans la partie 2.2 de la page 6 sont particulièrement longues et complexes, il est donc très difficile de les manipuler pour en isoler une variable par exemple. Ces équations sont trop compliquées pour être résolues sans l'aide d'un ordinateur, c'est pourquoi j'ai décidé d'écrire un programme python permettant de résoudre les équations. Le programme génère un fichier texte qui contient les résultats de toutes les équations que le programme à résolues pour obtenir l'angle de chaque moteur en fonction des angles α et β qui définissent l'inclinaison du plateau.

data.txt			
0.8		1.4#53.7	52.96 52.26
1.0		1.4#53.89	52.95 52.08
1.2		1.4#54.07	52.95 51.91
1.4		1.4#54.25	52.95 51.73
1.6		1.4#54.43	52.94 51.55
1.8		1.4#54.62	52.94 51.37
2.0		1.4#54.81	52.94 51.19
2.2		1.4#54.98	52.94 51.01
2.4		1.4#55.16	52.94 50.84
2.6		1.4#55.35	52.94 50.66
2.8		1.4#55.53	52.94 50.48
3.0		1.4#55.71	52.94 50.3
3.2		1.4#55.89	52.94 50.12
3.4		1.4#56.08	52.94 49.94
3.6		1.4#56.26	52.94 49.76

FIGURE 10 – Capture d'écran du fichier texte

Chacune des lignes comporte les informations suivantes : α | β # θ_A | θ_B | θ_C

3.2 Fonctionnement du programme

Ce programme génère un fichier texte en résolvant plusieurs systèmes d'équations, pour ce faire il est nécessaire de fixer certaines constantes au début du code. Voici la liste des différentes valeurs qu'il faut rentrer dans le programme pour qu'il puisse fonctionner :

- Distance entre l'arbre d'un moteur et le centre de la base⁶ : L = 8.5cm
- Hauteur du plateau par rapport à la base : d = 11.5cm
- Distance entre l'extrémité de chaque bras : D = 18cm
- Longueur du premier segment de chaque bras : r = 7cm

6. La base correspond au socle sur lequel sont fixés tous les moteurs.

— Longueur du deuxième segment de chaque bras : $l = 8.5\text{cm}$

Le programme doit à présent effectuer plusieurs systèmes d'équations, j'ai donc décidé d'utiliser la fonction `fsolve` issue du module `scipy.optimize`. Cette fonction permet de résoudre des systèmes d'équations en utilisant une méthode de résolution approchée, par conséquent les solutions ne seront pas exactes mais sont suffisamment précises pour ce projet.

```

1 def equationsKMT(p):
2     k, m, t = p
3     equation1 = (-cos(alpha)*cos(pi/6)*k)**2 + (-cos(alpha)*m-cos(
4         alpha)*sin(pi/6)*k)**2 + (sin(beta)*sin(alpha)*m+cos(pi/6)*
5         cos(beta)*sin(alpha)*k+sin(beta)*sin(alpha)*sin(pi/6)*k)**2
6     - 1
7     equation2 = (-cos(alpha)*cos(pi/6)*t)**2 + (cos(alpha)*sin(pi
8         /6)*t+cos(alpha)*m)**2 + (cos(beta)*sin(alpha)*cos(pi/6)*t-
9         sin(beta)*sin(alpha)*sin(pi/6)*t-sin(beta)*sin(alpha)*m)**2
10    - 1
11     equation3 = (cos(alpha)*cos(pi/6)*k+cos(alpha)*cos(pi/6)*t)**2
12     + (cos(alpha)*sin(pi/6)*k-cos(alpha)*sin(pi/6)*t)**2 + (-
13         cos(pi/6)*cos(beta)*sin(alpha)*k-sin(beta)*sin(alpha)*sin(
14             pi/6)*k-cos(beta)*sin(alpha)*cos(pi/6)*t+sin(beta)*sin(
15                 alpha)*sin(pi/6)*t)**2 - 1
16
17     return (equation1, equation2, equation3)

```

Ci-dessus se trouve une fonction qui contient un système d'équations⁷ à trois inconnues. Dans la suite du code la commande `fsolve(equationsKMT, (1, 1, 1))` est utilisée pour déterminer la valeur des variables⁸ k , m et t .

La partie la plus importante de ce code est une boucle dans laquelle le programme va résoudre les systèmes d'équations plusieurs fois en faisant varier les angles α et β . L'angle α varie entre 0° et 35° par pas de 0.2° et β varie entre 0° et 360° par pas de 0.2° également.

```

1 for beta in range(0, 360*5+1):
2     beta = radians(beta)/5
3     for alpha in range(0, 35*5+1):
4         alpha = radians(alpha)/5
5         k,m,t = fsolve(equationsKMT, (1, 1, 1))
6         k,m,t = -D*k, -D*m, -D*t
7         Xa, Ya, Za = k*cos(alpha)*cos(pi/6), k*cos(alpha)*sin(pi/6)
8             , k*(-cos(pi/6)*cos(beta)*sin(alpha)-sin(beta)*sin(
9                 alpha)*sin(pi/6))+d
10        Xb, Yb, Zb = 0, m*(-cos(alpha)), m*sin(beta)*sin(alpha)+d
11        Xc, Yc, Zc = t*(-cos(alpha)*cos(pi/6)), t*cos(alpha)*sin(pi
12            /6), t*(cos(beta)*sin(alpha)*cos(pi/6)-sin(beta)*sin(
13                alpha)*sin(pi/6))+d
14        tetaA = degrees(fsolve(equationTeta, 1, args=(Xa, Ya, Za)))
15        tetaB = degrees(fsolve(equationTeta, 1, args=(Xb, Yb, Zb)))
16        tetaC = degrees(fsolve(equationTeta, 1, args=(Xc, Yc, Zc)))
17        tetaA = round(tetaA, 2)

```

7. Ce système d'équations est présenté à la page 58.

8. Les variables k , m et t permettent de calculer la position du bout de chaque bras.

```
14     tetaB = round(tetaB , 2)
15     tetaC = round(tetaC , 2)
16
17     separateur = "|"
18     data = str(round(degrees(alpha),2)) + separateur + str(
19         round(degrees(beta),2)) + "#" + str(tetaA) + separateur
20         + str(tetaB) + separateur + str(tetaC) + "\n"
21     fichier.write(data)
```

Le fichier texte produit par ce programme sera utilisé plus tard par le programme Python *interface.py* qui s'occupera du traitement des images et de la communication avec le circuit imprimé.

4 Conception du circuit imprimé

4.1 Pourquoi réaliser un circuit imprimé ?

Les cartes Arduino sont très pratiques pour réaliser rapidement des prototypes de circuits électroniques. Cependant lorsqu'on utilise ces cartes on se retrouve souvent avec des dizaines de câbles reliant l'Arduino à une platine d'expérimentation⁹ qui est souvent elle-même reliée à d'autres composants comme par exemple des servomoteurs. Cela n'est pas un problème lorsqu'on travaille sur un prototype ou lorsqu'on veut tester des circuits, mais si on souhaite fabriquer un système plus fiable et moins encombrant il est préférable de réaliser un circuit imprimé.

4.2 Cahier des charges

C'est souvent très utile de fabriquer un circuit imprimé mais il faut garder en tête qu'il est très difficile de modifier le circuit de ce dernier après l'avoir fabriqué. C'est pourquoi il faut bien réfléchir aux différents composants et fonctions du circuit avant de le faire fabriquer. Le but de cette étape est de dresser une liste des différentes actions que le circuit sera capable d'effectuer. Le circuit dont j'ai besoin pour mon projet est très simple et contient relativement peu de composants, voici la liste des différents éléments qui seront présents dans le futur circuit :

- 1 microcontrôleur ATMEGA32U4-AU
- 1 oscillateur quartz d'une fréquence de 16MHz
- 2 condensateurs céramique de 22pf
- 1 Une diode électroluminescente (led)
- 1 Une résistance de 1kΩ
- 1 Une résistance de 10kΩ
- 2 Une résistance de 22Ω
- 1 condensateur de 0.1µf
- 1 condensateur de 1µf
- 1 condensateur de 10µf
- 3 sorties PWM pour le contrôle des servomoteurs
- 1 entrée pour une alimentation 5 volts destinée aux servomoteurs
- 4 pins destiné à être reliés à un port micro USB
- 4 pins destiné à l'utilisation d'un joystick¹⁰ utilisant l'interface I2C

4.3 Qu'est ce qu'un microcontrôleur ?

Un microcontrôleur se présente sous la forme d'une puce électronique qui contient tous les éléments essentiels au bon fonctionnement d'un ordinateur. En

9. Plus connue sous le nom de "breadboard" en anglais.

10. Pour des raisons techniques, le joystick ne sera finalement pas utilisé dans la suite du projet.

effet on y retrouve les composants suivants : un microprocesseur, une mémoire RAM¹¹, une mémoire EEPROM¹², une mémoire Flash¹³, plusieurs entrées / sorties et un oscillateur.

4.4 Choix du microcontrôleur

Comme présenté dans la partie 4.2, le circuit imprimé sera équipé d'un microcontrôleur ATMEGA32U4-AU. Cette puce est exactement la même qui est utilisée dans les Arduino Leonardo. Son avantage premier est d'avoir la capacité de gérer la communication USB sans l'aide d'un autre processeur. Cela permet de simplifier la conception du circuit imprimé tout en réduisant les coûts car moins de composants sont utilisés.

4.5 Conception du PCB

Pour concevoir le circuit imprimé de ce projet j'ai utilisé le logiciel *EAGLE* qui est un logiciel de conception assistée par ordinateur. La création du circuit s'est déroulé en deux étapes, j'ai commencé par la réalisation du schéma électrique(fig.11), puis j'ai réalisé le routage du circuit(fig.12). Le routage consiste à positionner les différents composants du circuit et à définir les passages par lesquels les connexions vont passer.

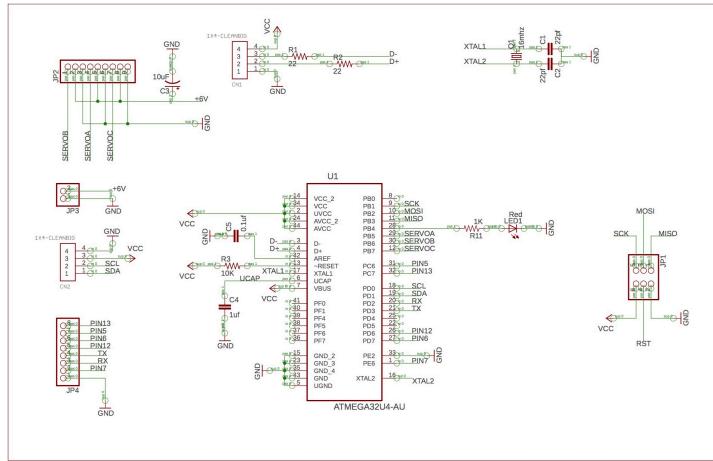


FIGURE 11 – Schéma du circuit imprimé

-
- 11. La RAM est une mémoire rapide mais qui stocke les informations de manière provisoire.
 - 12. L'EEPROM est une mémoire non volatile.
 - 13. La mémoire Flash des microcontrôleurs sert à stocker le programme que l'on a téléchargé.

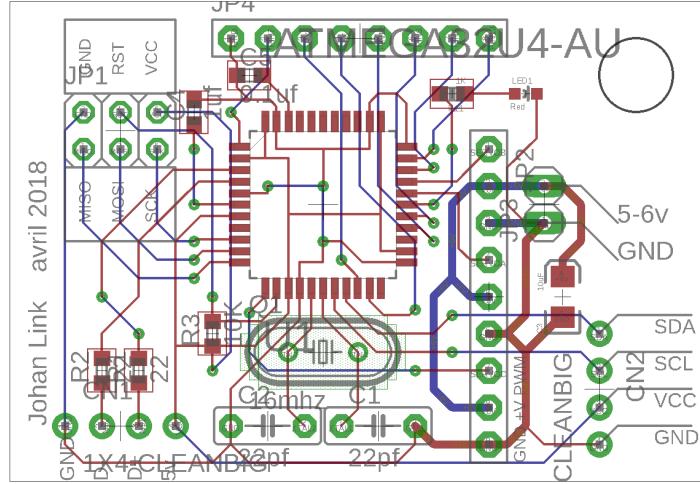


FIGURE 12 – Circuit imprimé

4.6 Fabrication du PCB

Après avoir réalisé le circuit sur le logiciel *EAGLE* il faut le fabriquer. Pour ce faire j'ai utilisé le site <https://www.pcbway.com> qui est un fabricant de circuits imprimés situé à Shenzhen en Chine. Ce site est particulièrement connu pour son rapport qualité-prix. L'utilisation de ce site est très simple, il suffit d'importer les fichiers *Gerber*¹⁴ de son circuit sur le site et de choisir certains paramètres tels que la couleur du circuit ou encore son épaisseur, ensuite la fabrication se déroule sur quelques jours et le circuit est envoyé à sa destination.

4.7 Assemblage des composants

Après avoir reçu le circuit imprimé il faut encore y souder tous les composants. Cette partie est particulièrement compliquée car les pièces utilisées sont très petites¹⁵ et donc difficiles à souder. Malheureusement certains composants tels que le microcontrôleur ne sont disponibles que dans des formats très petits. Je ne suis pas équipé avec les bons outils pour souder ce type de composants, cependant j'ai tout de même réussi à souder le microcontrôleur et les autres composants à l'aide d'un fer à souder standard. J'ai également fabriqué une panne¹⁶ à souder à l'aide d'un fil de cuivre afin d'avoir une panne assez petite pour pouvoir souder les composants. Par conséquent les soudures effectuées ne

14. Le format *Gerber* est un format standard pour stocker les informations d'un circuit imprimé.

15. Les composants utilisés sont soudés à la surface du circuit imprimé. Ce sont des composants SMD (Surface Mounted Device).

16. La panne du fer à souder est le bout du fer. Il s'agit de la partie qui est en contact avec la soudure.

sont pas toujours très propres mais fonctionnent correctement. On peut voir sur la figure 13 que j'ai utilisé la caméra de mon téléphone en guise de loupe pour mieux voir les pattes des composants.



FIGURE 13 – Soudage du microcontrôleur sur le PCB

4.8 Initialisation du microcontrôleur

Après avoir soudé tous les composants on ne peut pas encore utiliser le circuit, en effet le microcontrôleur a besoin de contenir un bootloader. Le bootloader est un microprogramme qui doit être dans le microcontrôleur pour que ce dernier puisse être programmé avec l'environnement Arduino. Il est relativement simple d'installer le bootloader dans le microcontrôleur, pour commencer il va falloir téléverser l'exemple *ArduinoISP* (Fichier > Exemples > ArduinoISP) dans un arduino UNO. Ensuite il faut faire les connexions suivantes entre l'arduino UNO et le circuit imprimé :

Broches de l'Arduino UNO	Broches du circuit imprimé
10	RST
11	MOSI
12	MISO
13	SCK
5v	Vcc
Gnd	Gnd

Les broches citées dans la colonne de droite du tableau sont toutes regroupées en haut à gauche du circuit imprimé (fig.12). Après avoir réalisé ces branchements il suffit de graver la séquence d'initialisation depuis le logiciel Arduino

(Outils > Graver la séquence d'initialisation). Une fois cette étape terminée le microcontrôleur est prêt à être programmé en branchant le circuit imprimé à l'ordinateur avec un câble USB.

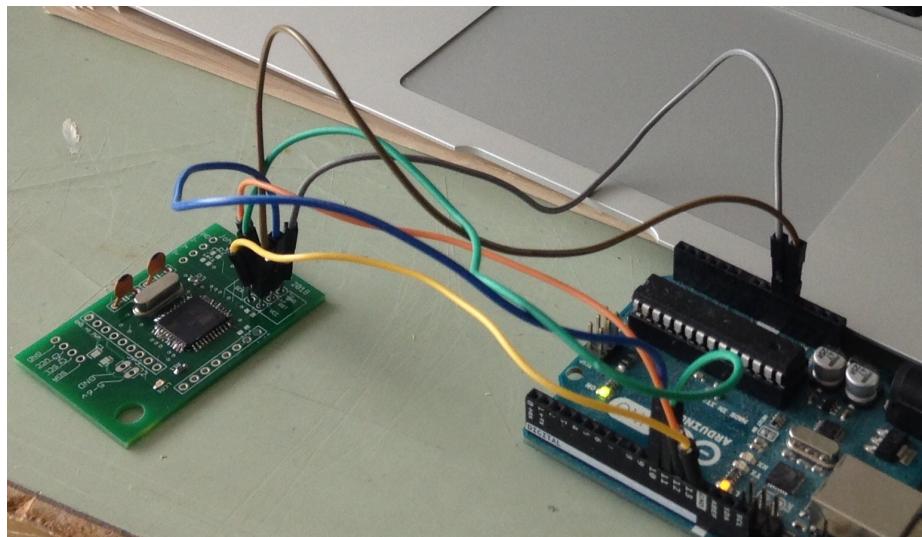


FIGURE 14 – Initialisation du microcontrôleur

5 Fonctionnement d'un régulateur PID

5.1 Qu'est ce qu'un régulateur PID ?

Un régulateur PID est un système dont l'objectif est de comparer l'écart entre une grandeur réelle et une valeur que l'on souhaite atteindre. Le but de ce système est de parvenir à la consigne¹⁷ le plus rapidement possible et cela peu importe les éléments qui peuvent perturber le régulateur. Dans le cadre de ce travail de maturité, le régulateur PID sera un élément-clé car il comparera la position réelle de la bille sur le plateau avec la position souhaitée. À partir de cette comparaison le système pourra déduire l'inclinaison requise du plateau pour déplacer la bille au bon emplacement.

5.2 La régulation Proportionnelle

Un régulateur PID est composé de trois types de régulations, le premier est la régulation proportionnelle. Ce principe de régulation est le plus simple car il met en relation la valeur mesurée avec la consigne de manière linéaire. En d'autres termes, plus la bille sera éloignée de la position souhaitée (consigne), plus le plateau se penchera. Pour un régulateur proportionnel on a la relation suivante :

$$I = K_p(C - M) \quad (2)$$

où I est l'instruction, K_p est un coefficient, C est la consigne et M est la valeur réelle mesurée. L'instruction est la valeur qui permet de déterminer les angles des servomoteurs pour qu'ils placent le plateau dans la bonne position. Le coefficient K_p doit être déterminé expérimentalement. Si K_p est grand alors la consigne sera atteinte plus rapidement et avec précision mais le système risque de devenir instable. Il faut donc trouver un compromis entre la stabilité et la rapidité du système.

La régulation proportionnelle ne prend pas en compte la vitesse à laquelle l'écart entre la mesure et la consigne évolue. La durée de l'écart n'est pas non plus pris en compte. De plus ce type de régulation a un autre défaut car, dans certains cas, la consigne n'est jamais atteinte. En effet il se peut que le système se stabilise légèrement en dessus ou en dessous de la consigne et crée une erreur statique. Par exemple, si le système que j'ai fabriqué se trouve sur une surface qui n'est pas parfaitement à l'horizontale, la bille sur le plateau ne se stabilisera jamais à la position voulue mais s'en approchera. L'équation (2) nous montre que l'instruction est égale à 0 si l'erreur entre la consigne et la mesure est nulle. Selon cette équation, le plateau doit avoir une inclinaison de 0° pour maintenir la bille à la bonne place, or si le système se trouve sur une surface qui est légèrement en pente, le plateau sera également penché bien que l'ordinateur a donné l'ordre au système de placer le plateau à l'horizontale. Par conséquent la bille tentera d'atteindre la consigne mais elle se stabilisera un peu à côté. En d'autres termes,

17. Valeur que l'on souhaite atteindre.

lorsque la bille se trouve loin de la consigne le plateau va s'incliner pour faire rouler la bille jusqu'à la consigne, à ce moment l'erreur entre la mesure et la consigne sera nulle, donc l'instruction le sera aussi et par conséquent le plateau se placera à 0° d'inclinaison. Mais comme le système n'est pas sur un sol parfaitement horizontal, la bille va commencer à s'éloigner de la consigne et donc le système penchera le plateau pour refaire parvenir la bille à la position voulue et là le problème recommence. Le système fini donc par se stabiliser en plaçant la bille légèrement à coté de la consigne.

5.3 La régulation Proportionnelle et Intégrale

L'ajout de la régulation intégrale va permettre d'augmenter les performances du système précédent. Le régulateur intégral a pour but de corriger l'erreur statique qui est un des défauts de la régulation proportionnelle. Pour ce faire, le régulateur va prendre en compte la durée de l'écart entre la consigne et la mesure. Par conséquent le système va, plusieurs dizaines de fois par seconde, additionner à une variable du programme la valeur de l'écart. Cette variable contient la somme de toutes les erreurs qui ont eu lieu depuis l'allumage du système. Finalement le programme additionne le résultat de la régulation proportionnelle avec le résultat de la régulation intégrale. On peut donc résumer le régulateur proportionnel et intégral avec l'équation suivante :

$$I = K_p(C - M) + K_i \int_0^t (C - M) dt \quad (3)$$

où K_i est un coefficient qui doit être déterminé expérimentalement.

5.4 La régulation Proportionnelle, Intégrale et Dérivée

Le dernier système de régulation fonctionne avec la dérivée de la position de la bille. En effet, les deux régulateurs présentés précédemment ne prennent pas en compte la vitesse de la bille. Pourtant la vitesse de la bille est un élément très important pour pouvoir réguler sa position. Le programme va calculer la vitesse de la bille en temps réel en additionnant la position de la bille avec sa position précédente et en divisant le résultat obtenu par le temps qui s'est écoulé entre deux images envoyées par la caméra qui filme la bille. Ce temps se situe aux alentours de 0.03 secondes car la caméra filme à une vitesse de 30 images/secondes. On peut résumer le régulateur proportionnelle, intégrale et dérivée par l'équation suivante :

$$I = K_p(C - M) + K_i \int_0^t (C - M) dt + K_d \frac{dr}{dt} \quad (4)$$

où dr est la distance parcourue par la bille dans un intervalle de temps (dt) qui tend vers 0 et K_d est un coefficient qui doit être déterminé expérimentalement.

6 Mise en place du régulateur PID dans le programme Python

6.1 Transformation des équations

Les équations (2), (3) et (4), sont présentées avec des notions d'intégrales et de dérivées. Les équations ont été formulées ainsi pour les présenter de manière générale dans une notation qui est fréquemment utilisée en physique. Dans ce projet, c'est l'équation (4) qui va être utilisée car elle contient les trois formes de régulation présentées. Cette équation va être traitée par un ordinateur, on peut donc légèrement la modifier pour supprimer l'intégrale et la dérivée :

$$I = K_p(C - M) + K_i S_{erreur} + K_d \frac{M_{precedente} - M}{0.03} \quad (5)$$

où S_{erreur} est une variable du programme qui contient la somme de toutes les erreurs¹⁸ qui ont eu lieu depuis l'allumage du système, M est la position actuelle, mesurée par la caméra, de la bille et $M_{precedente}$ est la position précédente de la bille.

Dans le programme Python, l'équation (5) est présente deux fois car les axes X et Y sont traités séparément, on obtient donc les deux équations suivantes :

$$I_x = K_p(C_x - M_x) + K_i S_{erreurX} + K_d \frac{M_{precedenteX} - M_x}{0.03} \quad (6)$$

$$I_y = K_p(C_y - M_y) + K_i S_{erreurY} + K_d \frac{M_{precedenteY} - M_y}{0.03} \quad (7)$$

Grâce aux équations (6) et (7) nous obtenons les composantes du vecteur I qui permettent de savoir dans quelle direction la bille doit se déplacer. Maintenant, nous devons en déduire la valeur des angles α et β (fig.3, page 7) qui définissent l'inclinaison du plateau.

6.2 Un exemple concret

6.2.1 Détermination des composantes I_x et I_y

Cette section permet d'expliquer le fonctionnement du programme Python à travers un exemple concret. Pour ce faire j'ai lancé une balle de ping-pong sur le plateau du système et j'ai récupéré les différentes données traitées par le programme à un moment donné. En voici une partie :

- Position de la balle en x : $M_x = 176px$
- Position de la balle en y : $M_y = 147px$
- Précédente position de la balle en x : $M_{precedenteX} = 184px$
- Précédente position de la balle en y : $M_{precedenteY} = 160px$

¹⁸. L'erreur est définie par la soustraction de la consigne par la position mesurée de la bille.

- Somme de toutes les erreurs en x : $S_{erreurX} = 6179px$
- Somme de toutes les erreurs en y : $S_{erreurY} = 2333px$

Il est important de préciser que les valeurs ci-dessus sont définies en pixels car elles ont été mesurées à partir des images de la caméra située au-dessus du plateau. Il faut également rappeler que l'origine du système d'axes se situe en haut à gauche du champ de vision de la caméra, de plus l'axe des y est dirigé vers le bas. La figure 15 (page 25) est une vue schématique de ce que la caméra filme. Dans cet exemple, la balle doit se stabiliser au milieu du plateau par conséquent la consigne sera définie ainsi :

- Position de la consigne en x : $C_x = 240px$
- Position de la consigne en y : $C_y = 240px$

La caméra est réglée pour filmer avec une résolution de $480px$ par $640px$, l'image est ensuite recordée pour avoir une résolution de $480px$ par $480px$. Etant donné que la caméra se situe exactement au-dessus du plateau, le centre du plateau correspond au centre de l'image, donc le centre du plateau se situe à la position ($240px ; 240px$) de l'image (voir figure 15, page 25).

Pour cet exemple, les coefficients K_p , K_i et K_d ont les valeurs suivantes :

- Coefficient proportionnel : $K_p = 5$
- Coefficient intégrale : $K_i = 0.1$
- Coefficient dérivée : $K_d = 4.5$

Maintenant le programme peut utiliser les équations (6) et (7) pour calculer I_x et I_y :

$$\begin{aligned} I_x &= K_p(C_x - M_x) + K_i S_{erreurX} + K_d \frac{M_{precedenteX} - M_x}{0.03} \\ &= 5 * (240 - 176) + 0.1 * 6179 + 4.5 * \frac{184 - 176}{0.0333} \\ &\approx 2018.98 \end{aligned} \quad (8)$$

$$\begin{aligned} I_y &= K_p(C_y - M_y) + K_i S_{erreurY} + K_d \frac{M_{precedenteY} - M_y}{0.03} \\ &= 5 * (240 - 147) + 0.1 * 2333 + 4.5 * \frac{160 - 147}{0.0333} \\ &\approx 2455.05 \end{aligned} \quad (9)$$

Les valeurs obtenues sont ensuite divisées par 10000, cette opération a pour but de pouvoir utiliser des coefficients plus petits et donc plus agréables à régler. Il serait donc possible de garder I_x et I_y sans faire la division par 10000, mais les

coefficients K_p , K_i et K_d seraient alors 10000 fois plus grands et donc moins agréables à manipuler.

$$I_x = \frac{2018.98}{10000} \approx 0.2019 \quad (10)$$

$$I_y = \frac{2455.05}{10000} \approx 0.2455 \quad (11)$$

6.2.2 Détermination de l'angle α

Nous avons déterminé les composantes I_x et I_y , maintenant nous pouvons facilement en déduire l'angle α qui correspond à l'angle entre un plan horizontal et le plan qui contient le plateau du système. La section 2.2 (page 6) expliquait qu'un vecteur \vec{v} est créé et que ce vecteur est toujours perpendiculaire au plateau. On peut donc définir l'angle alpha comme étant l'angle entre la verticale et le vecteur \vec{v} . La figure 16 ci-dessous permet de mieux comprendre les liens entre les différents éléments. Il est aussi utile de rappeler que $\|\vec{v}\| = 1$. Le point M présent sur les deux figures représente le centre de la balle qui est sur le plateau.

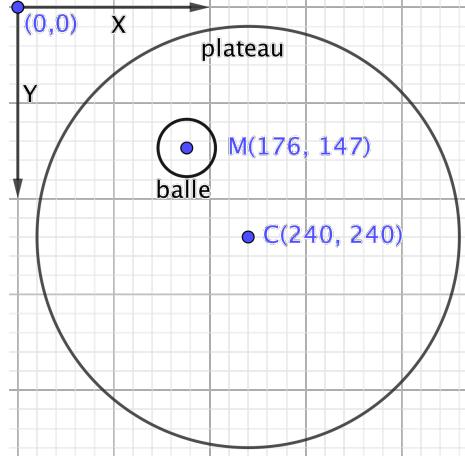


FIGURE 15 – Vu de dessus du problème

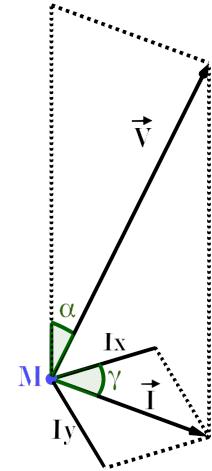


FIGURE 16 – Construction de \vec{v}

À l'aide des formules basiques de trigonométrie on peut facilement trouver la relation suivante :

$$\sin(\alpha) = \frac{\sqrt{I_x^2 + I_y^2}}{1} \quad (12)$$

Ensuite il suffit d'isoler α :

$$\alpha = \arcsin \left(\frac{\sqrt{I_x^2 + I_y^2}}{1} \right) \quad (13)$$

On peut utiliser les valeurs de I_x et I_y trouvées avec les équations (10) et (11) (page 25) :

$$\alpha = \arcsin \left(\frac{\sqrt{0.2019^2 + 0.2455^2}}{1} \right) \approx 18.53^\circ \quad (14)$$

Maintenant nous savons que le plateau devra s'incliner de 18.53° par rapport à l'horizontale. Le système fabriqué est capable d'incliner le plateau de 35° au maximum, par conséquent le programme Python contient une condition qui empêche le système de s'incliner de plus de 35° . Autrement dit, l'angle α est fixé par le programme à 35° lorsque l'équation (13) donne une valeur supérieure à 35° . Cet angle est également fixé à 35° lorsque $\sqrt{I_x^2 + I_y^2} > 1$.

6.2.3 Détermination de l'angle β

L'angle β est un peu plus compliqué à trouver que l'angle α . Dans un premier temps nous allons déterminer l'angle γ présent sur la figure 16 (page 25). Cette première étape est assez simple :

$$\gamma = \arctan \left(\frac{I_y}{I_x} \right) \quad (15)$$

En utilisant les valeurs de I_x et I_y trouvées avec les équations (10) et (11) (page 25) on obtient :

$$\gamma = \arctan \left(\frac{0.2455}{0.2019} \right) \approx 50.56^\circ \quad (16)$$

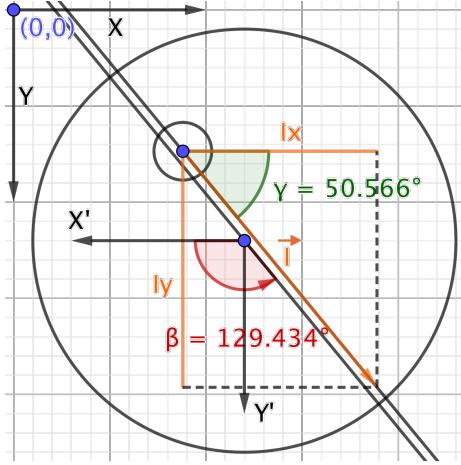


FIGURE 17 – Détermination de l’angle β

Pour des raisons de visibilité, la norme du vecteur \vec{I} a été multiplié par 1000 sur ce schéma.

Les axes X' et Y' du schéma ci-dessus représentent le repère orthonormé présent sur la figure 1 (page 6).

L’équation (15) donnera toujours un angle compris dans l’intervalle $]-90^\circ; 90^\circ[$, pourtant on a défini dans la section 3.2 (page 13) qu’il faut un angle β compris dans l’intervalle $[0^\circ; 360^\circ[$ pour définir la direction de l’inclinaison du plateau. Il faut alors déduire l’angle β à partir de l’angle γ . Le problème comporte quatre cas de figure, les voici :

- $I_x > 0$ et $I_y \geq 0$
- $I_x > 0$ et $I_y \leq 0$
- $I_x < 0$ et $I_y \geq 0$
- $I_x < 0$ et $I_y \leq 0$

Le premier cas de figure correspond à la situation vue dans la figure 17. Les deux composantes sont positives, donc il faut faire $180 - \gamma$ pour obtenir l’angle β . Pour les quatre cas de figure il faut faire une opération différente.

- | | |
|-----------------------------|--------------------------|
| — $I_x > 0$ et $I_y \geq 0$ | $\beta = 180 - \gamma $ |
| — $I_x > 0$ et $I_y \leq 0$ | $\beta = 180 + \gamma $ |
| — $I_x < 0$ et $I_y \geq 0$ | $\beta = \gamma $ |
| — $I_x < 0$ et $I_y \leq 0$ | $\beta = 360 - \gamma $ |

Nous devons ajouter encore deux conditions à la liste ci-dessus car l’équation (15) (page 26) est indéterminée si I_x est nulle :

- | | |
|-----------------------------|---------------|
| — $I_x = 0$ et $I_y \geq 0$ | $\beta = 90$ |
| — $I_x = 0$ et $I_y \leq 0$ | $\beta = 270$ |

Avec l'équation (16) (page 26) nous avons calculé que $\gamma = 50.56$. Nous savons grâce aux équations (10) et (11) (page 25) que les composantes du vecteur \vec{I} sont positives. Nous pouvons en déduire que $\beta = 180 - 50.56 = 129.44^\circ$.

6.2.4 Détermination des angles de chaque moteur

Nous avons trouvé les angles α et β qui définissent l'inclinaison et l'orientation du plateau, maintenant le programme doit faire tourner les moteurs pour placer le plateau dans la bonne position. Dans un premier temps, les angles trouvés doivent être arrondis à 0.2 près. La section 3.2 (page 13) expliquait qu'un fichier texte a été créé pour contenir toutes les relations entre l'angle de chaque moteur et les angles α et β . Par conséquent le programme Python qui a déterminé α et β va aller chercher dans le fichier texte la ligne qui commence par "18.6 | 129.4". La ligne trouvée sera la suivante : "18.6|129.4#51.24|37.09|73.18". À gauche du # on retrouve les angles α et β et à droite du # trois nombres correspondent respectivement à l'angle θ de chaque moteur. Pour finir, le programme envoie ces trois angles par USB à l'Arduino qui contrôle le système.

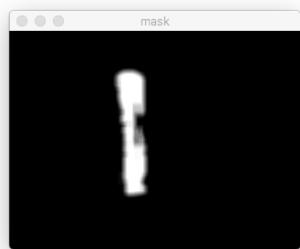
7 Utilisation de OpenCV avec Python

7.1 Qu'est-ce que OpenCV ?

OpenCV (Open Source Computer Vision) est une bibliothèque contenant plus de 2500 d'algorithmes permettant de faire du traitement d'images. OpenCV est la bibliothèque la plus utilisée en ce qui concerne la vision par ordinateur. Cette bibliothèque est disponible pour les langages C, C++ et Python. Pour ce projet nous utiliserons le langage Python car il est particulièrement simple à utiliser.

7.2 Programme intermédiaire

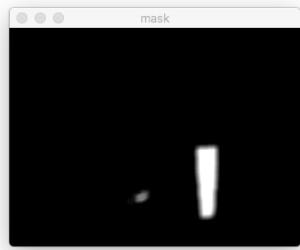
Afin de me familiariser avec OpenCV j'ai décidé d'écrire un exemple de programme servant à détecter la position d'un objet d'une certaine couleur. Cet exemple est très important car il sera très utile pour le programme final du projet. L'utilisation de ce code est relativement simple, en effet lorsqu'on lance le programme une fenêtre s'ouvre et affiche en direct les images prises par la webcam branchée à l'ordinateur. Ensuite il suffit de cliquer à l'aide de la souris sur un objet que l'on souhaite et le programme va détecter sa couleur puis entourer cet objet avec un cercle jaune.



(a) Image traitée



(b) Image final



(c) Image traitée



(d) Image final

FIGURE 18 – Capture d'écran du programme intermédiaire en fonctionnement

7.3 Reconnaissance d'une couleur

Grâce à OpenCV, nous pouvons détecter une couleur dans une image relativement facilement. Dans un premier temps il faut savoir que OpenCV utilise l'espace colorimétrique HSV¹⁹, cela veut dire que dans le programme les couleurs seront définies selon trois nombres. Le premier est un nombre compris dans l'intervalle [0; 179] et permet de définir la teinte d'une couleur. Le deuxième et le dernier nombre sont compris dans l'intervalle [0; 255] et permettent de définir respectivement la saturation et la valeur²⁰ de la couleur. Si par exemple nous voulons que le programme détecte la couleur rouge dans une image, il faut lui donner un intervalle de couleur dans lequel se trouve la couleur que l'on veut isoler. L'intervalle permettra au programme de détecter tous les rouges entre du rouge très clair et du rouge foncé. Pour commencer, nous créons deux variables qui contiennent respectivement le rouge clair et le rouge foncé dans l'espace colorimétrique HSV.

```
1 rougeClair = np.array([0, 100, 100])
2 rougeFonce = np.array([0, 250, 250])
```

Ensuite le programme va créer une image en noir et blanc (figure 18) où les zones blanches montrent l'emplacement de la couleur recherchée.

```
1 imageNoirBlanc = cv2.inRange(imageCouleur, rougeClair, rougeFonce)
```

19. HSV signifie *Hue Saturation Value*.

20. La valeur d'une couleur correspond à sa luminosité.

8 L'interface graphique

8.1 Pourquoi faire une interface graphique ?

Afin d'utiliser le système que j'ai fabriqué avec facilité, j'ai décidé de programmer en Python une interface graphique me permettant de contrôler plusieurs aspects du système.

8.2 Fonctionnalités de l'interface

L'interface est composée de quatre parties bien distinctes :

- La première partie permet de faire des réglages liés à la vidéo prise par la caméra.
- La deuxième partie permet de contrôler les moteurs, et permet également de démarrer le système.
- La troisième partie permet de régler les coefficients K_p , K_i et K_d .
- La dernière partie permet de faire suivre à la balle une forme géométrique.

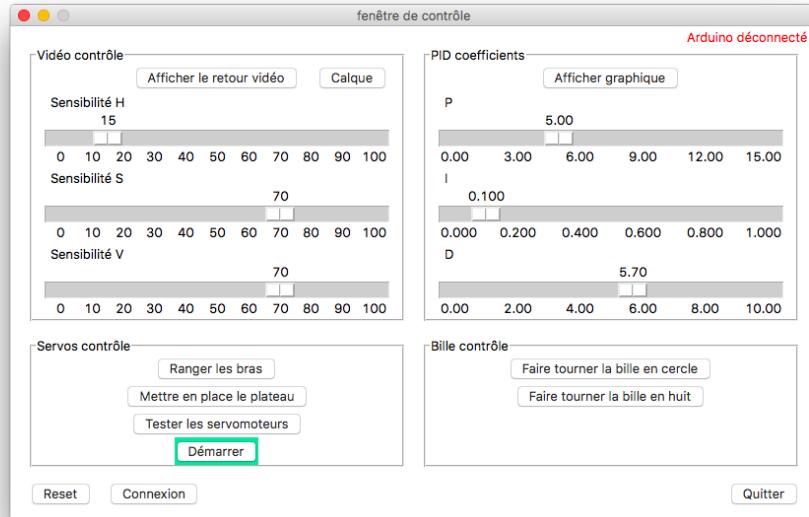


FIGURE 19 – Capture d'écran de l'interface graphique

Les boutons "Afficher le retour vidéo" et "Afficher graphique" permettent d'ouvrir les fenêtres ci-dessous.

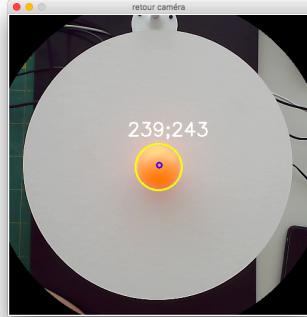


FIGURE 20 – Retour vidéo de la caméra

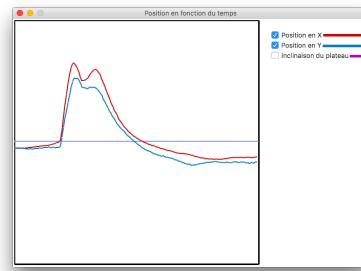


FIGURE 21 – Graphique de la position de la balle en fonction du temps

La fenêtre (figure 20) diffuse les images prises par la caméra. Au centre, nous pouvons y voir la balle de ping-pong orange qui est entouré d'un cercle jaune. Ce cercle permet se rendre compte que le programme détecte le bon objet, de plus le programme rajoute, au-dessus du cercle, les coordonnées de la balle. Au centre du cercle jaune, un point bleu marque l'emplacement de la consigne. La fenêtre (figure 21) affiche un graphique de la position de la balle en fonction du temps. La ligne rouge représente la position de la balle selon l'axe des x et la ligne bleu représente la position de la balle selon l'axe des y en fonction du temps.

9 Le code Arduino

9.1 Communication avec le programme Python

On peut facilement faire communiquer un programme Python avec un Arduino. La bibliothèque *serial* permet au programme Python d'envoyer des informations à un périphérique USB.

```
1 ser = serial.Serial(nom_du_port_USB, 9600)
2 ser.write("information")
```

Dans l'exemple ci-dessus, la commande *write* envoie à l'Arduino le mot "information". De son côté, l'Arduino doit juste lire le port USB de la manière suivante :

```
1 if (Serial.available() > 0){
2     Serial.println(Serial.readString())
3 }
```

9.2 Contrôle des servomoteurs

Normalement on utilise la fonction *write* pour faire tourner un servomoteur à l'aide d'un Arduino, cependant cette fonction permet de faire tourner le moteur avec une précision de seulement 1°. La section 3.2 (page 13) a permis de calculer les angles que les moteurs doivent avoir pour mettre le plateau dans n'importe quelles positions. Ces angles ont des valeurs précises au centième près et les moteurs doivent avoir ces angles avec précision. Il faut utiliser la fonction *writeMicroseconds* qui permet de faire tourner un servomoteur avec beaucoup de précision. Si nous écrivons *writeMicroseconds(1000)* le servomoteur se placera avec un angle de 0° et si nous écrivons *writeMicroseconds(2000)* le servomoteur se placera avec un angle de 180°. La fonction *writeMicroseconds* est donc moins intuitive à utiliser que la fonction *write* car nous pouvons mettre une valeur en degré dans la fonction *write*. Cependant la fonction *writeMicroseconds* permet d'avoir plus de précision.

10 Conclusion

Pour conclure, ce travail de maturité a eu un impact positif et m'a beaucoup appris. Ce projet a été l'occasion de réaliser un projet concret qui contient une partie hardware et une partie software. Au cours de ce travail j'ai été confronté à plusieurs problèmes techniques que j'ai dû surmonter. Certains de ces problèmes ont eu lieu pendant la construction du système et d'autres problèmes sont survenus pendant la programmation du système. J'ai dû trouver des solutions quitte à devoir faire quelques concessions. La partie matérielle de ce projet est tout aussi importante que la partie informatique, en effet la modélisation et la fabrication du système m'a demandé autant de temps que la programmation de l'ensemble.

Malgré les difficultés rencontrées, le système fonctionne bien. Bien entendu ce système a ses limites, par exemple si la balle est lancée trop rapidement le système réagira trop lentement et la bille tombera du plateau. Plusieurs aspects du mécanisme pourraient être améliorés pour perfectionner le système. La plus grande source d'imprécision est l'articulation de chaque bras qui comporte du jeu et cela a pour effet de rendre le plateau instable. Du jeu est également présent à la jonction entre la base de chaque bras et l'arbre du moteur auquel il est emboité et vissé. La caméra utilisée pour ce projet semble avoir un peu de latence et cela est également une source de problèmes puisque le système réagi en retard sur les mouvements de la balle. Heureusement la latence est assez faible et n'empêche pas le système de fonctionner. Le dernier point qui pourrait être amélioré dans ce travail est la détermination des coefficients du régulateur PID, en effet ces coefficients ont été trouvés en faisant un certain nombre d'essais expérimentaux. Si ces coefficients avaient été déterminés de manière plus rigoureuse les résultats seraient sans doute encore meilleurs. Une vidéo de présentation de mon travail est disponible à l'adresse suivante : <https://www.youtube.com/watch?v=57DbEEBF7sE>

Bibliographie

- [1] abcclim. Principe de régulation p-pi-pid, 2018 (consulté en juin 2018). <https://www.abcclim.net/regulation-p-pi-pid.html>.
- [2] Kar Anirban. Opencv object tracking by colour detection in python, 2017 (consulté en juin 2018). <https://thecodacus.com/opencv-object-tracking-colour-detection-python/#.W1cKZC30lp->.
- [3] maike hao. Linux and python : auto-detect arduino serial port [duplicate], juillet 2012 (consulté en avril 2018). <https://stackoverflow.com/questions/11364879/linux-and-python-auto-detect-arduino-serial-port>.
- [4] Mark. A very nonlinear system of three equations, septembre 2013 (consulté en avril 2018). <https://ask.sagemath.org/question/10506/a-very-nonlinear-system-of-three-equations/>.
- [5] Ferdinand Piette. Implémenter un pid sans faire de calculs!, août 2011 (consulté en juillet 2018). <http://www.ferdinandpiette.com/blog/2011/08/implementer-un-pid-sans-faire-de-calculs/>.
- [6] Adrian Rosebrock. Ball tracking with opencv, septembre 2015 (consulté en juin 2018). <https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/>.
- [7] The scientificsentence. Modes de régulation, 2007 (consulté en juillet 2018). <http://scientificsentence.net/Regulation/Regulation5.html>.
- [8] Sparkfun. Sparkfun atmega32u4 breakout, 2015 (consulté en avril 2018). <https://www.sparkfun.com/products/retired/11117>.
- [9] user3704293. How to split a string using a specific delimiter in arduino?, avril 2015 (consulté en juillet 2018). <https://stackoverflow.com/questions/29671455/how-to-split-a-string-using-a-specific-delimiter-in-arduino>.
- [10] Wikipedia. Pid controller, juillet 2018 (consulté en juillet 2018). https://en.wikipedia.org/wiki/PID_controller.

Table des figures

1	Disposition des moteurs	6
2	Bras d'un moteur	6
3	Vecteur \vec{v} perpendiculaire au plateau	7
4	Capture d'écran de mon projet dans <i>Fusion 360</i>	9
5	Vue générale de tous les éléments du système	9
6	Pièces imprimées en 3D	10
7	Début de l'assemblage des pièces	11
8	Capture d'écran du support de la caméra dans <i>Fusion 360</i>	12
9	Caméra utilisée pour ce projet	12
10	Capture d'écran du fichier texte	13
11	Schéma du circuit imprimé	17
12	Circuit imprimé	18
13	Soudage du microcontrôleur sur le PCB	19
14	Initialisation du microcontrôleur	20
15	Vu de dessus du problème	25
16	Construction de \vec{v}	25
17	Détermination de l'angle β	27
18	Capture d'écran du programme intermédiaire en fonctionnement	29
19	Capture d'écran de l'interface graphique	31
20	Retour vidéo de la caméra	32
21	Graphique de la position de la balle en fonction du temps	32

Annexes

A	solveEquation.py	38
B	programmeIntermediaire.py	41
C	interface.py	43
D	arduinoCode.ino	55
E	Détermination des angles θ des servomoteurs	57
F	Quelques plans du projet	60
G	Prototypes	67
H	Quelques esquisses du projet	68

E Détermination des angles θ des servomoteurs

$$\text{Vecteur } \vec{v} : \begin{pmatrix} \cos\beta\sin\alpha \\ \sin\beta\sin\alpha \\ \cos\alpha \end{pmatrix} \perp \pi$$

Plan du plateau (π) : $\cos\beta\sin\alpha x + \sin\beta\sin\alpha y + \cos\alpha z + d = 0$

$$\text{Plan vertical contenant le bras du moteur A (}planA\text{)} : \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} -\cos(30) \\ -\sin(30) \\ 0 \end{pmatrix} = \begin{pmatrix} \sin(30) \\ -\cos(30) \\ 0 \end{pmatrix} \Rightarrow \sin(30)x - \cos(30)y = 0$$

$$\text{Plan vertical contenant le bras du moteur B (}planB\text{)} : \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \Rightarrow -x = 0$$

$$\text{Plan vertical contenant le bras du moteur C (}planC\text{)} : \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \wedge \begin{pmatrix} \cos(30) \\ -\sin(30) \\ 0 \end{pmatrix} = \begin{pmatrix} \sin(30) \\ \cos(30) \\ 0 \end{pmatrix} \Rightarrow \sin(30)x + \cos(30)y = 0$$

$$\text{Vecteur } \vec{a} \text{ se trouvant sur la droite } planA \cap \pi : \begin{pmatrix} \cos\beta\sin\alpha \\ \sin\beta\sin\alpha \\ \cos\alpha \end{pmatrix} \wedge \begin{pmatrix} \sin(30) \\ -\cos(30) \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\alpha\cos(30) \\ \cos\alpha\sin(30) \\ -\cos(30)\cos\beta\sin\alpha - \sin\beta\sin\alpha\sin(30) \end{pmatrix}$$

$$\text{Vecteur } \vec{b} \text{ se trouvant sur la droite } planB \cap \pi : \begin{pmatrix} \cos\beta\sin\alpha \\ \sin\beta\sin\alpha \\ \cos\alpha \end{pmatrix} \wedge \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\cos\alpha \end{pmatrix}$$

$$\text{Vecteur } \vec{c} \text{ se trouvant sur la droite } planC \cap \pi : \begin{pmatrix} \cos\beta\sin\alpha \\ \sin\beta\sin\alpha \\ \cos\alpha \end{pmatrix} \wedge \begin{pmatrix} \sin(30) \\ \cos(30) \\ 0 \end{pmatrix} = \begin{pmatrix} -\cos\alpha\cos(30) \\ -\cos\alpha\sin(30) \\ \cos(30)\cos\beta\sin\alpha - \sin\beta\sin\alpha\sin(30) \end{pmatrix}$$

d est la hauteur du plateau au-dessus des moteurs lorsque le plateau est à plat.

$$\text{Point } A, \text{ position de l'extrémité du bras du moteur A : } A = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} + k \begin{pmatrix} 0 & \cos\alpha\cos(30) \\ -\cos(30)\cos\beta\sin\alpha & \cos\alpha\sin(30) \\ 0 & \sin\beta\sin\alpha\sin(30) \end{pmatrix} = \begin{pmatrix} X_a \\ Y_a \\ Z_a \end{pmatrix}$$

$$\text{Point } B, \text{ position de l'extrémité du bras du moteur B : } B = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} + m \begin{pmatrix} 0 \\ -\cos\alpha \\ \sin\beta\sin\alpha \end{pmatrix} = \begin{pmatrix} X_b \\ Y_b \\ Z_b \end{pmatrix}$$

$$\text{Point } C, \text{ position de l'extrémité du bras du moteur C : } C = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} + t \begin{pmatrix} \cos\alpha\cos(30) \\ -\cos\alpha\sin(30) \\ \cos(30)\cos\beta\sin\alpha - \sin\beta\sin\alpha\sin(30) \end{pmatrix} = \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix}$$

$$58 \quad \begin{aligned} \|\overrightarrow{AB}\| = D \Rightarrow \|\overrightarrow{OB} - \overrightarrow{OA}\| &= D = \left\| \begin{pmatrix} 0 \\ -m\cos\alpha \\ m\sin\beta\sin\alpha + d \end{pmatrix} - \begin{pmatrix} k\cos\alpha\cos(30) \\ k\cos\alpha\sin(30) \\ -k\cos(30)\cos\beta\sin\alpha - k\sin\beta\sin\alpha\sin(30) + d \end{pmatrix} \right\| \\ \|\overrightarrow{BC}\| = D \Rightarrow \|\overrightarrow{OC} - \overrightarrow{OB}\| &= D = \left\| \begin{pmatrix} t\cos\beta\sin\alpha\cos(30) \\ t\cos\beta\sin\alpha\sin(30) + d \\ k\cos\alpha\cos(30) \end{pmatrix} - \begin{pmatrix} 0 \\ -m\cos\alpha \\ m\sin\beta\sin\alpha + d \end{pmatrix} \right\| \\ \|\overrightarrow{CA}\| = D \Rightarrow \|\overrightarrow{OA} - \overrightarrow{OC}\| &= D = \left\| \begin{pmatrix} -k\cos(30)\cos\beta\sin\alpha - k\sin\beta\sin\alpha\sin(30) + d \\ k\cos\alpha\sin(30) \\ -t\cos\alpha\cos(30) \end{pmatrix} - \begin{pmatrix} t\cos\beta\sin\alpha\cos(30) \\ t\cos\beta\sin\alpha\sin(30) + d \\ t\cos\beta\sin\alpha\cos(30) - t\sin\beta\sin\alpha\sin(30) + d \end{pmatrix} \right\| \end{aligned}$$

$$\begin{cases} (-k\cos\alpha\cos(30))^2 + (-m\cos\alpha - k\cos\alpha\sin(30))^2 + (m\sin\beta\sin\alpha + d + k\cos(30)\cos\beta\sin\alpha + k\sin\beta\sin\alpha\sin(30) - d)^2 = D^2 \\ (-t\cos\alpha\cos(30))^2 + (t\cos\beta\sin\alpha\cos(30) + m\cos\alpha)^2 + (t\cos\beta\sin\alpha\sin(30) - t\sin\beta\sin\alpha - d)^2 = D^2 \\ (k\cos\alpha\cos(30) + t\cos\alpha\cos(30))^2 + (k\cos\alpha\sin(30) - t\cos\alpha\sin(30) - t\cos\beta\sin\alpha - k\sin\beta\sin\alpha\sin(30) + d - t\cos\beta\sin\alpha\cos(30) \\ + t\sin\beta\sin\alpha\sin(30) - d)^2 = D^2 \end{cases}$$

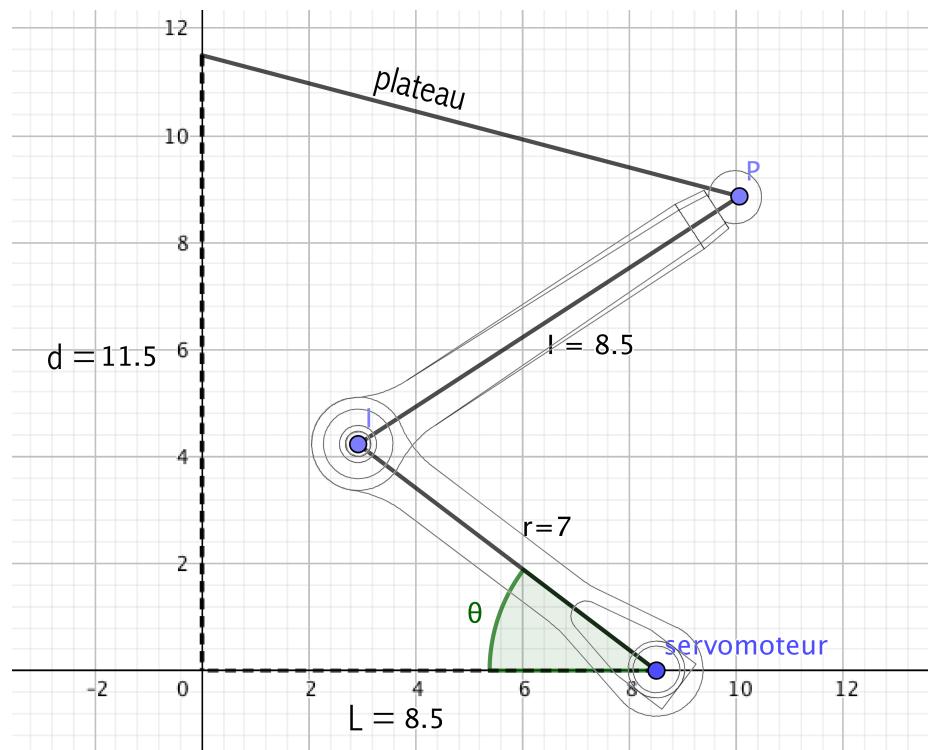
Point I , emplacement de l'articulation du bras d'un moteur : $I(L - r\cos\theta; r\sin\theta)$

Point P , emplacement de l'extrémité du bras d'un moteur : $P(\sqrt{X_p^2 + Y_p^2}; Z_p)$

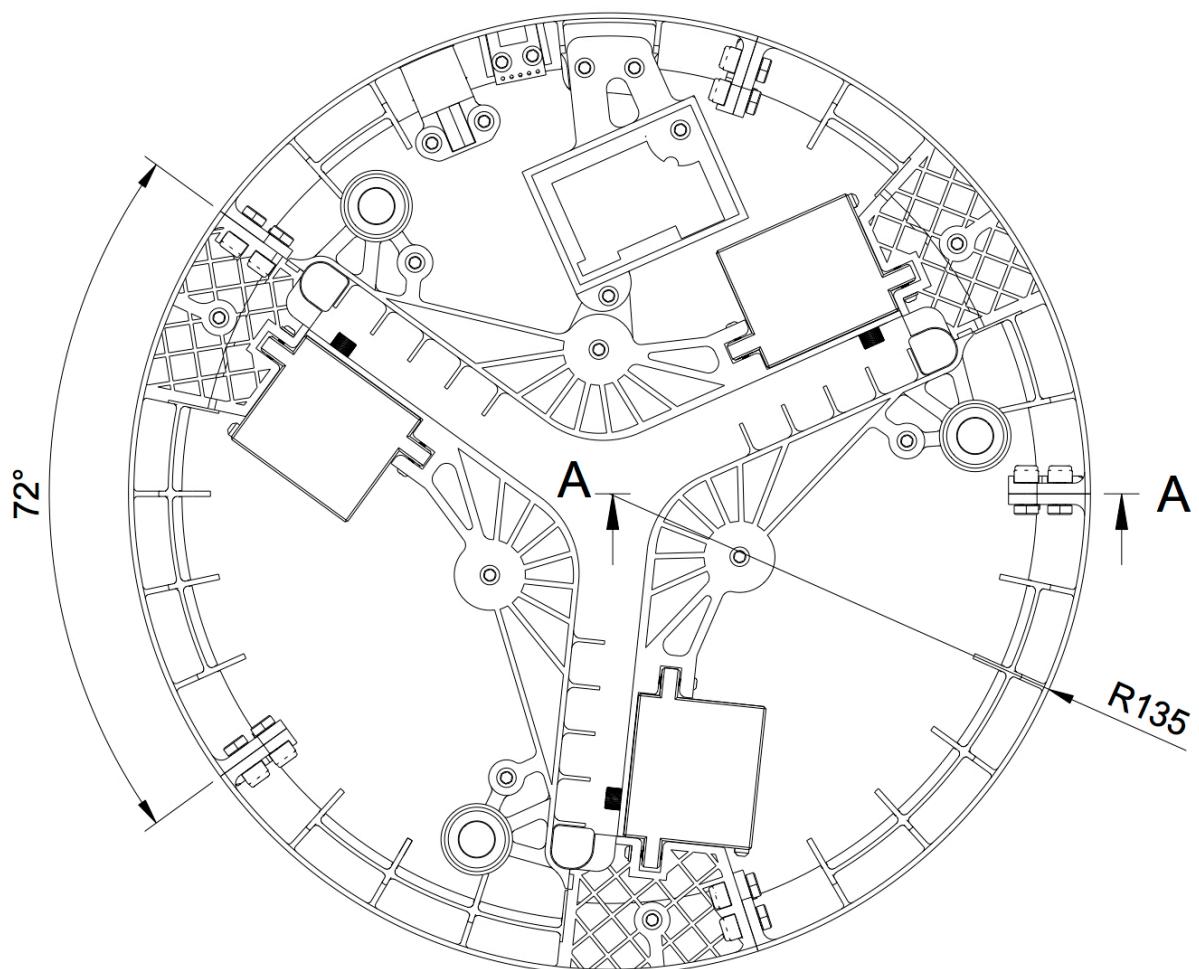
$$\|\overrightarrow{PI}\| = l$$

$$\overrightarrow{PI} = \begin{pmatrix} L - r\cos\theta \\ r\sin\theta \end{pmatrix} - \begin{pmatrix} \sqrt{X_p^2 + Y_p^2} \\ Z_p \end{pmatrix}$$

$$(L - r\cos\theta - \sqrt{X_p^2 + Y_p^2})^2 + (r\sin\theta - Z_p)^2 = l^2$$

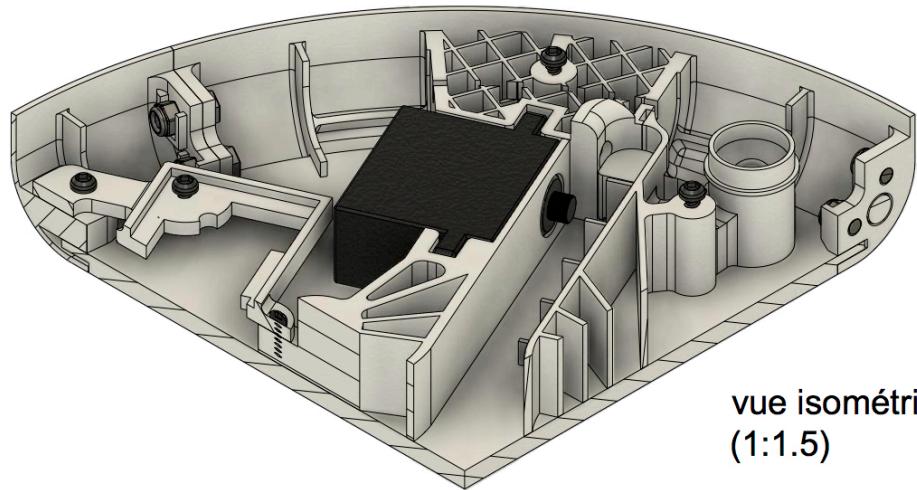
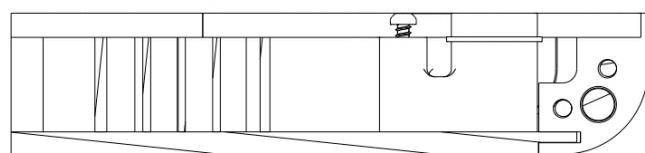


F Quelques plans du projet

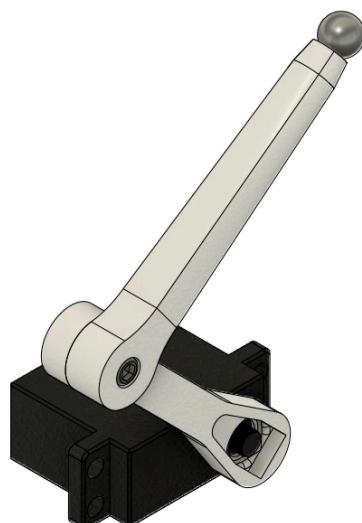
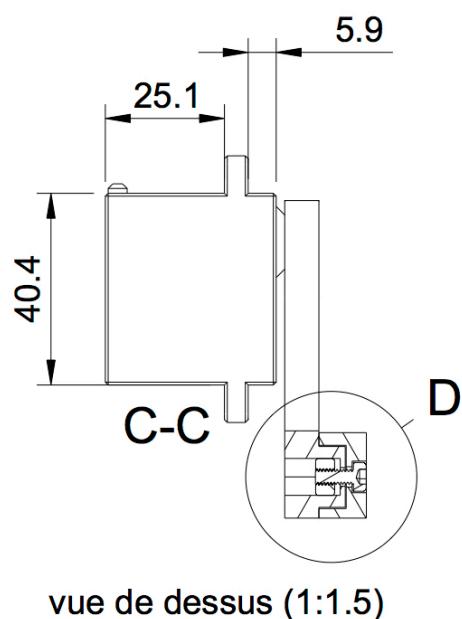
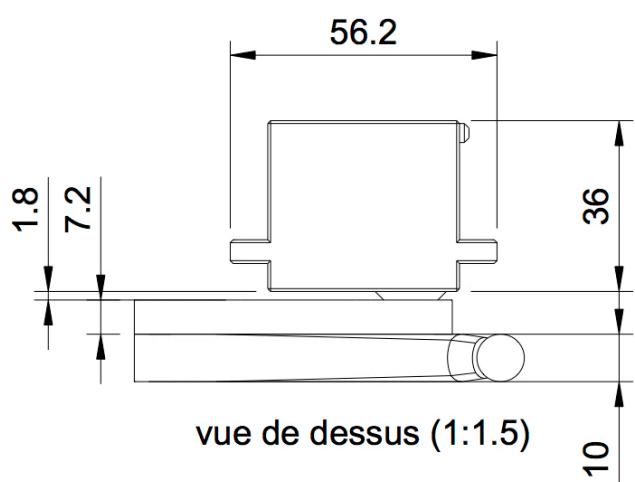
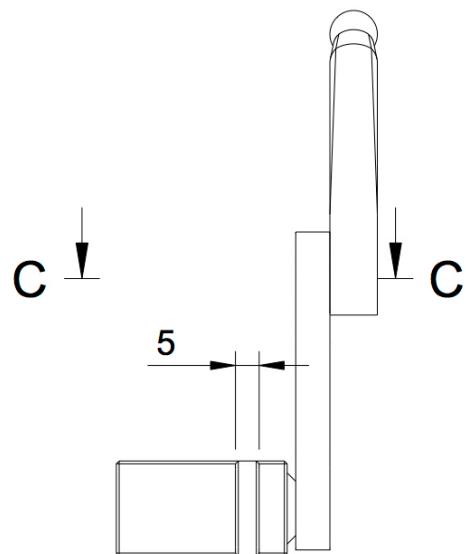
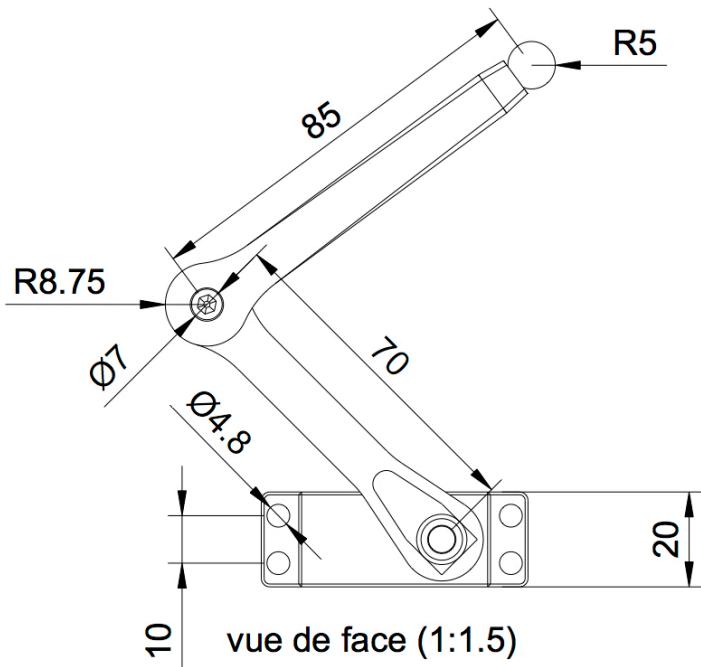


vue de dessus (1:2)

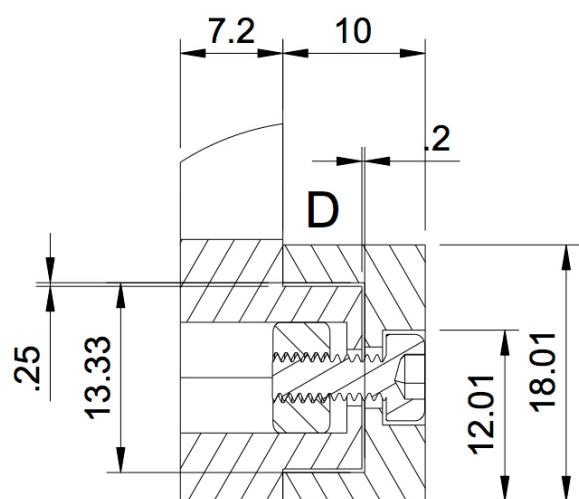
A-A (1:1.5)



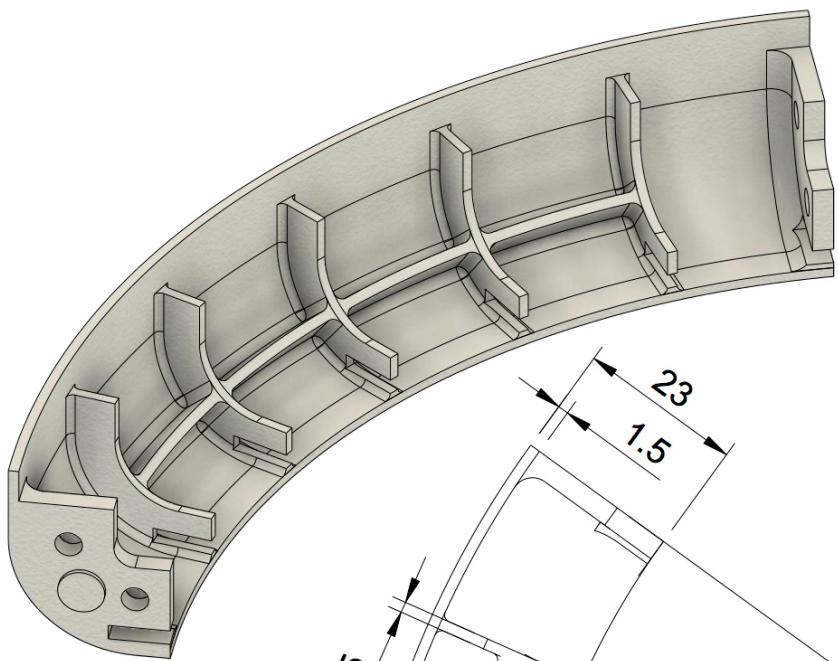
vue isométrique
(1:1.5)



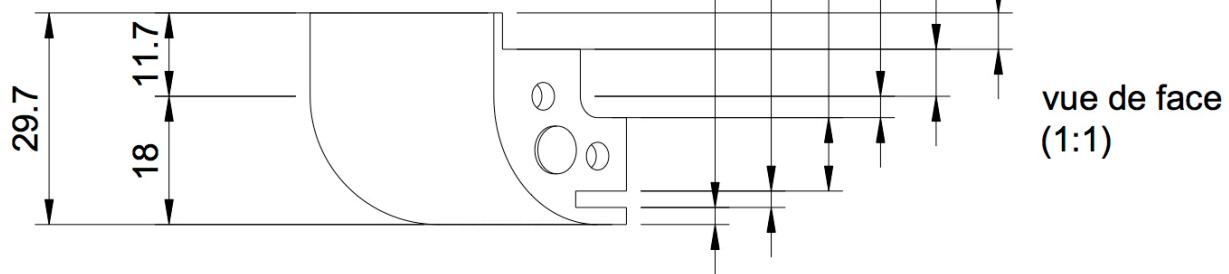
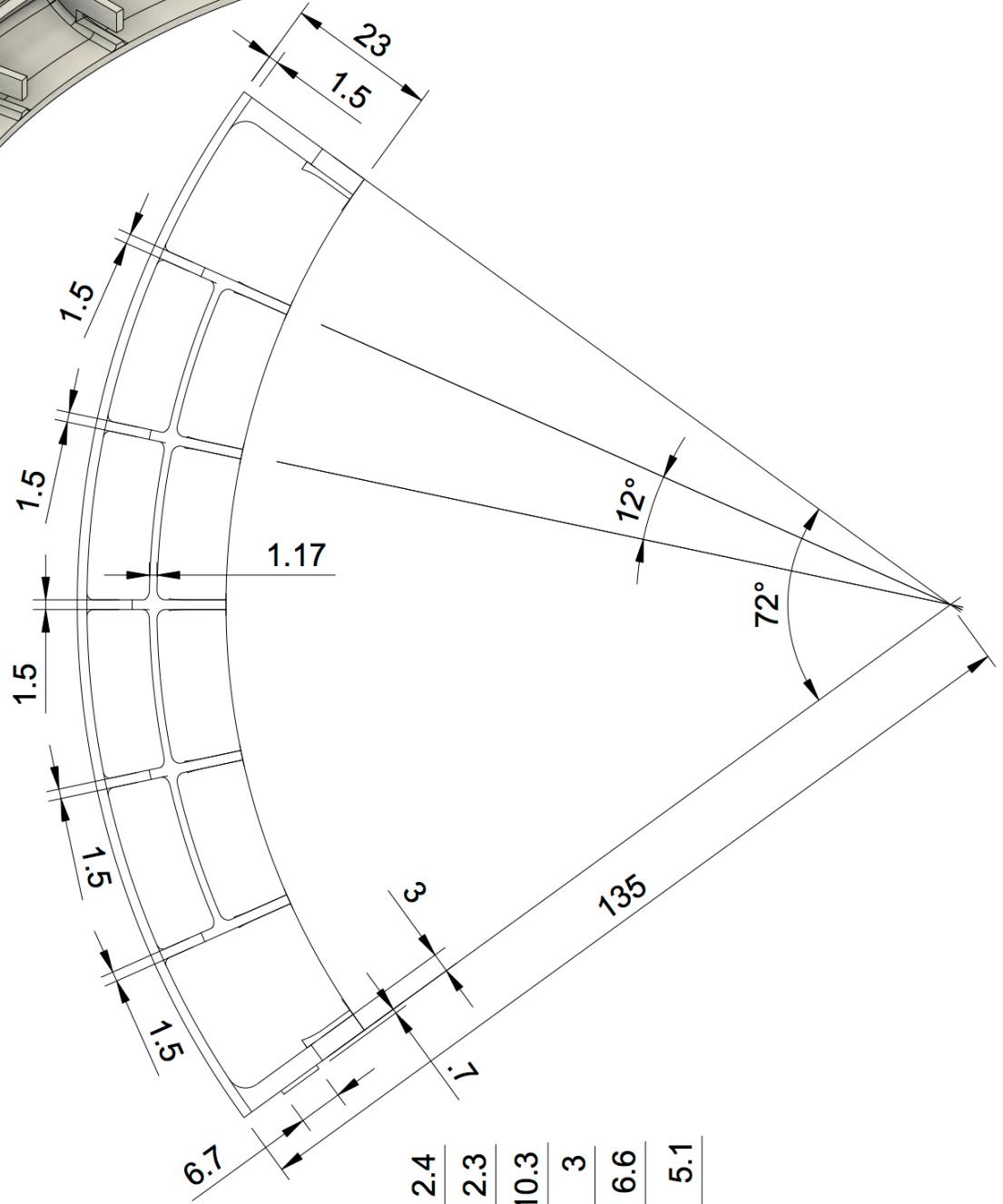
Isometric view of the servomotor and its arm (1:1.5)



vue isométrique
(1:1)



Vue de dessus
(1:1)



G Prototypes

