GYMNASE PROVENCE
MATURITY WORK 2017

REMOTE CONTROLLED CYLINDER

Project supervisors
Laurent BESSON
Javier IGLESIAS

Océane PATINY

Lausanne, March 25, 2019

Abstract

This one year project consists of the development of a remote controlled cylinder that uses servo motors to displace its center of mass, which leads the cylinder to accelerate. This implies that very diversified fields such as mathematics, physics, programming, electronics and mechanics had to be explored and partly understood to be combined in the final prototype. This prototype has an acrylic glass base structure with a diameter of 32 cm, it is controlled using a C.H.I.P. Pro (a micro-controller similar to Raspberry Pi) and a customized dedicated extension board. The board includes a driver for four servo motors and a 6 axis accelerometer. Finally, the power supply consists of a pack of three AA batteries. All the programming was made using JavaScript scripts that are run using Node.js and a Shell. In addition, the cylinder is remotely controlled through a web page, which make it easy and intuitive for the user. The final features of this robotic cylinder are the following: it can roll forwards and backwards with a maximal speed of 1.04 [m/s], stop and keep itself balanced on a gentle slope of maximally 3 degrees. Also, it is only working well on flat, hard surfaces, though it could easily be adapted to rougher surfaces. The possibility has also been considered of creating a sphere that uses the same technologies as the cylinder that was constructed in this project. Finally, this work is meant to be as open-source as possible. Therefore, the software that was developed is under MIT licence, whereas the hardware is under CERN OHL. The results of the project can be seen easily on: https://youtu.be/QggOjoGjJhA

Acknowledgements

For having kept me on Earth in my most delirious states, I would first like to express all my gratitude to my project supervisors, Mr. Laurent Besson and Mr. Javier Iglésias, who have also valiantly waited for me to find the rails that led straight to... somewhere, a place that I believe, is not too lost. I do also owe a special thank to my father, who maybe finally sucked me into the world of programming and with whom I had sometimes animated, but productive discussions. In addition, my appreciation goes to the Octanis association, whose members are indirectly at the essence of this work. To my outstanding companion Héloïse Piguet, I need to formulate my gratitude for all the moments of pure despair and unsatisfaction we spent together. I do also want to thank my mother and my brother, for having, respectively, helped me to change my point of view when I was trying to solve various problems and for having dirtied his hands while making brass weights used for experiments. To Mr. Marc Wettstein, I dedicate my most sincere acknowledgments for the knowledge he accepted to share with me during these long hours spend at the Makerspace of Renens. Regarding Michaël Zasso and Miguel Asencio, who both helped me to get better in JavaScript programming, I know they were desperate when facing my recurrent errors, but for their unlimited patience, I thank them. Lastly, all my consideration goes to Frank Bonnet for his precious advice.

Contents

Acknowledgements

1 What is this project about? 7
1.1 Introduction . . . 7
1.2 Defining the project . . . 8

2 Theoretical physics 9
2.1 Definitions . . . 9
2.1.1 Frame of reference . . . 9
2.1.2 Kinetics . . . 10
2.1.3 Rigid body rotation . . . 11
2.1.4 Playing with vectors . . . 11
2.2 The Wheel experiment . . . 13
2.3 Physical resolution . . . 16
2.3.1 Cylinder movement when the mass position is constant . . . 22

3 Already existing projects 23
3.1 The Sphero . . . 23
3.2 The Cubli . . . 24
3.3 Review of the literature . . . 26

4 Approach 29
4.1 How to move the mass? . . . 29
4.1.1 Stepper motor in the center . . . 29
4.1.2 Pulleys . . . 30
4.1.3 Rack and pinion drives . . . 32
4.1.4 Long pistons . . . 33
4.1.5 Short pistons . . . 34
4.1.6 Electromagnets . . . 36
4.1.7 Servos on the outside . . . 37
4.1.8 Servos in the center . . . 38
4.2 Chosen approach: servo motors . . . 39

5 Theoretical movement of the mass 43
5.1 Modeling the problem using geoGebra . . . 43
5.2 Analysis of the data exported from GeoGebra . . . 48
5.3 Mathematical expression of mass position . . . 51
5.3.1 From a to α . . . 56
5.3.2 Servos angles when the mass is in the center . . . 56

6 Mechanics **59**
6.1 Prototype 1 . . . 59
6.2 Prototype 2 . . . 61
6.3 Prototype 3 . . . 63

7 Electronics **67**
7.1 Choice of the micro-controller . . . 67
7.2 Peripherals: accelerometer and servo driver . . . 70
7.3 The C.H.I.P. Pro extension PCB . . . 73

8 Software **77**
8.1 Bash, JavaScript and node . . . 77
8.1.1 Peripherals control implementation . . . 79
8.2 From mathematical formula to JavaScript . . . 80
8.3 Prototype parameters and calibration . . . 82
8.3.1 Parameters acquisition procedure . . . 82
8.4 Web page for the control of the cylinder . . . 85
8.5 Final code structure . . . 88

9 Results **91**
9.1 Mass' movement . . . 91
9.2 Cylinder's movement . . . 93
9.3 Physical properties . . . 96
9.4 Power consumption and autonomy . . . 97
9.5 Control . . . 97
9.6 Price . . . 98

10 Discussion and conclusion **99**

Appendix **i**

Bibliography **vii**

List of Figures **xi**

List of Tables **xiii**

Acronyms **xvi**

Chapter 1

What is this project about?

Lamentor, ergo sum.

1.1 Introduction

Initially, this work began with a desperate search of an idea, that could have been appropriate, but overall not too simple to realize: I needed something challenging, maybe out of my reach, but that would be motivating enough to keep me focused during a whole year.
And that idea, I got it on a glacier, in June 2016, looking at a rover developed by an association called Octanis, that is mainly composed of EPFL students. The rover was being tested before its trip to Antarctica. That day, there was a lot of wind and I was discussing with my father about the best wind-proof design that should be used for an autonomous rover.
Immediately, I thought of a sphere, because if it rolls, it can't ever fall on its side! And just after I thought: but wait, how would I roll a sphere at distance? With my father, we discussed a long time over that question and found it really interesting. And indeed, I got so excited about it that I decided to do that as my maturity work. It had all the advantages: it was for sure not too easy, it would allow me to learn technologies I didn't knew at all and I was interested in, and it was fun!

But when I got home, I saw that something similar had already been made on Kickstarter and that it had had a lot of success. I felt so miserable: why would I try to redo something others had already done and that worked perfectly fine?
Finally, I realized that this project wasn't using at all the same technology I wanted to use. So why wouldn't I do my own version? And I tried...
Yet, I quickly realized that building a remotely controlled sphere was beyond my abilities, considering the time and the resources I had. I therefore decided to bring the problem to its 2D version and to build a cylinder.

1.2 Defining the project

As the title of this work might tell, the objective of this project is to create a remote controlled cylinder that rolls when the position of its center of mass is moved. This idea, that can be summarized in one sentence, can seem very simple, but in fact, it requires a wide range of different skills. Indeed, this work is related to five main poles: mechanics, electronics, programming, physics and mathematics.
The mechanical part is obviously fairly consistent since the project aims to realize a physical object, that has not been done before. This implies that this work is mainly research, which leads to a lot of prototyping. All that begins with conception: pencil sketching, but also 2D design, using software like FreeCad. Afterwards, came the part of building itself which allowed to learn how to use as well more traditional technologies, like a metal lathe, a drill press or a milling machine, as modern tools like a laser cutter. Finally, the choice of materials for their different properties was also needed, for example, brass for its high density (8.5 [g/cm^3]) and acrylic glass for its resistance.
Secondly, electronics concern all the technology used to simply induce the movement, retrieve the position of the cylinder and communicate. The choice of the most adapted micro-controller was not easy, since it had to be as light as possible and powerful enough to realize important calculation. So, basic programming of MCUs such as Arduino (ATmega32U4) or C.H.I.P. (an equivalent of the Raspberry Pi) was an important part of this project. WiFi devices, an accelerometer and other peripherals have also been used. Finally, the development of a PCB using the Eagle software has been done, which allowed getting used to electronics concepts.
In the case of programming, JavaScript has mainly been used to allow all the control of the cylinder itself, which includes the mass displacement, the acquisition of the accelerometer's data and all tests needed for debug. On the client side, basic CSS and HTML were used. Regarding physics, the main field that was needed is the kinematics of rigid body. Indeed, physics are the tool that allows to conciliate a physical object and purely mathematical theory, the step between these being fairly difficult to cross. The notions of inertia and center of mass were also really important.
Mathematics resulted of physics, in some way, but one field of them in particular, geometry, kept recurring. More precisely, it was symmetries that were literally everywhere and that helped a lot to solve some problems. This more mathematical part led to learn the GeoGebra software, practical for geometry, algebra, and animation of mathematical figures.

The last point is the open-source aspect of this project. Indeed, all the development process was published on GitHub since the very beginning. To see all the work that has been done and to find even more information, please check the following repositories:

- https://github.com/opatiny/tm : for all the general content.
- https://github.com/opatiny/chip : for the JavaScript code in particular.
- https://github.com/opatiny/TM.ltx : to have the latex file of this report.

Finally, the results of this work have been put under two different licenses: the software part is under MIT licence, whereas the hardware is under CERN OHL.

Chapter 2

Theoretical physics

2.1 Definitions

In this section are going to be explained all the physics principles and constants definitions that will be needed to understand this work.

2.1.1 Frame of reference

A frame of reference is a unit axis system defined by the person that solves the problem. Every value is then based on this reference, that is generally considered as position and time zero. In a Cartesian frame of reference, the mostly used one, the origin point O is still and the axes do always point in the same direction. The three axes are named $\vec{e_1}$, $\vec{e_2}$ and $\vec{e_3}$, each of these vectors corresponds to one space dimension.

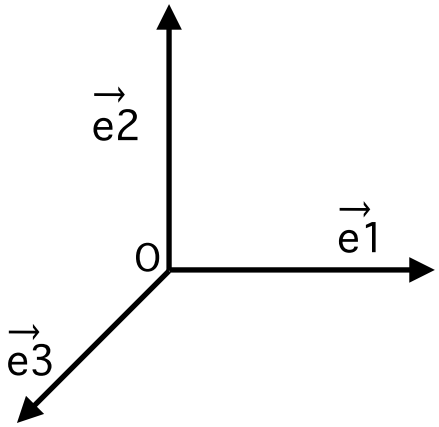


Figure 2.1: Cartesian frame of reference

However, considering our case, we might need to use another kind of frame: a cylindrical coordinates system. This type of frame is also formed with three orthogonal vectors, but two of them rotate around the origin. The unit axes are named $\vec{e_\theta}$, $\vec{e_R}$ and $\vec{e_z}$. This system of coordinates is usually a lot more practical for problems that include rotations, because, for instance, the position is a lot simpler, since it does only correspond to an angle and a radius.

2.1.2 Kinetics

Every object evolving in space has three main properties: position ($\vec{x}$), speed ($\vec{v}$) and acceleration ($\vec{a}$). They all vary in what can be considered as a fourth dimension: time (t). Indeed, the position is a function of time that can be written $\vec{x}(t)$, $\vec{v}(t)$ is the derivative of this function, and $\vec{a}(t)$ is its second derivative. In other words:

$$\vec{a}(t) = \big(\vec{v}(t)\big)' = \big(\vec{x}(t)\big)'' \tag{2.1}$$

In physics, the derivatives would rather be written like in equation 2.2.

$$\vec{a} = \dot{\vec{v}} = \ddot{\vec{x}} \tag{2.2}$$

In calculation, mean speed ($\vec{v}_m$) and acceleration ($\vec{a}_m$) are often used. This is their definitions:

$$\vec{v}_m = \frac{\Delta\vec{x}}{\Delta t} \quad \text{and} \quad \vec{a}_m = \frac{\Delta\vec{v}}{\Delta t} \tag{2.3}$$

Three other essential tools are the laws of Newton.

$$\begin{aligned} N1 &: \sum_i \vec{F}_i^{ext} = \vec{0} \iff \vec{a} = \vec{0} \\ N2 &: \sum_i \vec{F}_i^{ext} = m \cdot \vec{a} \\ N3 &: \vec{F}_{A/B} = -\vec{F}_{B/A} \end{aligned} \tag{2.4}$$

Then comes the first law of conservation: energy.

$$\Delta E = 0 \tag{2.5}$$

The difference of kinetic energy (K) is equal to the opposite of the sum of the works done by the forces applied on the system.

$$\Delta K = \sum_i W_i(\vec{F}^{ext}) \tag{2.6}$$

Where the value of K is:

$$K = \frac{1}{2}mv^2 \tag{2.7}$$

And the work done by a force is:

$$W(\vec{F}) = \vec{F} \cdot \Delta\vec{x} \tag{2.8}$$

We will finally use the momentum conservation formula:

$$\Delta\vec{p} = \vec{0} \tag{2.9}$$

Which can be used if $\sum_i \vec{F}_i^{ext} = \vec{0}$ and if m is constant. In this formula, $\vec{p}$ is the momentum, defined as

$$\vec{p} = m \cdot \vec{v} \tag{2.10}$$

2.1.3 Rigid body rotation

The basic unit of rotations is the angle (θ), which is often used to define a position. Then, angular speed and acceleration are derivatives of the angular position. The angular speed, ω, is the first derivative of θ, whereas the angular acceleration, α, is its second derivative. Consequently, an equation similar to Equation 2.2 can be written.

$$\vec{\alpha} = \dot{\vec{\omega}} = \ddot{\vec{\theta}} \tag{2.11}$$

When talking about rotations, the physician leaves the material point to use the rigid body. A rigid body is a system in which distances do not vary. Every rigid body has a mass, but also another property: the way its mass is distributed in space. This property is called the moment of inertia, its symbol is I. I is defined in Equation 2.12. Note that the inertia of an object is different for every rotation axis, except when the body has symmetries. In Equation 2.12, the moment of inertia considers a rotation axis passing through a point O of an object of mass m.

$$I_O = \sum_{\alpha} m_\alpha \cdot OP_\alpha^2 \tag{2.12}$$

However, there is a trivial formula that allows, when one knows the inertia of an object (of total mass M) with respect to an axis, to calculate it for any other axis that is parallel to the first one (Δ).

$$I_\Delta = I_O + d(\Delta, O)^2 \cdot M \quad \text{(Steiner's Theorem)} \tag{2.13}$$

Also, the analogy between translations (physics of the material point) and rotations can be made, which is the equation that is used the most to solve our rotations problems. It consists of an equivalent of the second law of Newton, but for rotations:

$$N2, R : \sum_{\alpha} \vec{M}_{O,\alpha} = I_O \cdot \vec{\alpha} \tag{2.14}$$

In this equation, $\vec{M}$ are the torques produced by a force applied at a certain distance to a reference point. The physical definition of the torque is given in Equation 2.15.

$$\vec{M}_O = \vec{OP} \wedge \vec{F} \tag{2.15}$$

A link between the translation speed and the rotation speed exists, that is described in Equation 2.16. In that equation, the point O is a reference point, whose speed is known.

$$\vec{v}_P = \vec{v}_O + \vec{\omega} \wedge \vec{OP} \tag{2.16}$$

For the angular and translation accelerations, the link is the following:

$$a = \alpha r \tag{2.17}$$

Where r is the the radius of the object that is rolling.

2.1.4 Playing with vectors

These properties of vectors are based on the definitions given in the book *Calculus - The Classical Edition*, written by Swokowski [6].

Dot product

Definition 2.1.1 *Dot product The dot product is the product of the parallel components of two vectors, the result of a dot product is a scalar.*

Considering two vectors $\vec{a} = a_1 \cdot (\vec{e_1}) + a_2 \cdot (\vec{e_2}) + a_3 \cdot (\vec{e_3})$ and $\vec{b} = b_1 \cdot (\vec{e_1}) + b_2 \cdot (\vec{e_2}) + b_3 \cdot (\vec{e_3})$, their dot product is the following:

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{2.18}$$

Another mathematical definition is:

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos(\gamma) \tag{2.19}$$

Where γ is the angle between $\vec{a}$ amd $\vec{b}$. The dot product is null when the vectors are perpendicular.

Vector product

Definition 2.1.2 *Vector product The vector product is the product of the perpendicular components of two vectors, the result of this operation is a vector perpendicular to the two initial vectors. Note that the vector product is not permutational.*

Considering two vectors $\vec{a} = a_1 \cdot (\vec{e_1}) + a_2 \cdot (\vec{e_2}) + a_3 \cdot (\vec{e_3})$ and $\vec{b} = b_1 \cdot (\vec{e_1}) + b_2 \cdot (\vec{e_2}) + b_3 \cdot (\vec{e_3})$, their vector product is the following:

$$\vec{a} \wedge \vec{b} = \vec{e_1}(a_2 b_3 - a_3 b_2) + \vec{e_2}(a_3 b_1 - a_1 b_3) + \vec{e_3}(a_1 b_2 - a_2 b_1) \tag{2.20}$$

The vector product is also defined as:

$$\vec{a} \wedge \vec{b} = |\vec{a}||\vec{b}| \sin(\alpha) \cdot \vec{u} \tag{2.21}$$

Where α is the angle between $\vec{a}$ and $\vec{b}$ and $\vec{u}$ is a unitary vector perpendicular to the plane defined by $\vec{a}$ and $\vec{b}$. The vector product is null when the vectors are parallel or anti-parallel.
Here are some useful vector product properties:

$$(\vec{a} + \vec{b}) \wedge \vec{c} = \vec{a} \wedge \vec{c} + \vec{b} \wedge \vec{c} \tag{2.22}$$

$$(r\vec{a}) \wedge \vec{b} = \vec{a} \wedge (r\vec{b}) = r(\vec{a} \wedge \vec{b}) \tag{2.23}$$

In equation 2.23, r is a scalar.

$$\vec{a} \wedge (\vec{b} \wedge \vec{c}) = \vec{b}(\vec{a} \cdot \vec{c}) - \vec{c}(\vec{a} \cdot \vec{b}) \tag{2.24}$$

$$(\vec{a} \wedge \vec{b})' = (\vec{a})' \wedge \vec{b} + \vec{a} \wedge (\vec{b})' \tag{2.25}$$

General properties

The equation underneath describes the derivative of a vector. Note that a vector does only vary when it rotates.

$$\dot{A\vec{B}} = \vec{\omega} \wedge A\vec{B} \tag{2.26}$$

2.2 The Wheel experiment

For someone who has only studied physics of the material point, problems with rotating objects can be really unpredictable. It is for that reason that what we call the "Wheel experiment" has been made. The hardware used for this experiment consists of a homogeneous wooden disk drilled with symmetrical arrays of small holes that allow to fix various masses to it. In addition, the disk has a center piece that allows it to rotate with low friction around a steel axis that is supported by a wooden base[1]. All these elements are visible on Figure 2.4. The final system is useful do make experiments involving torques, rotations, inertia and symmetries.

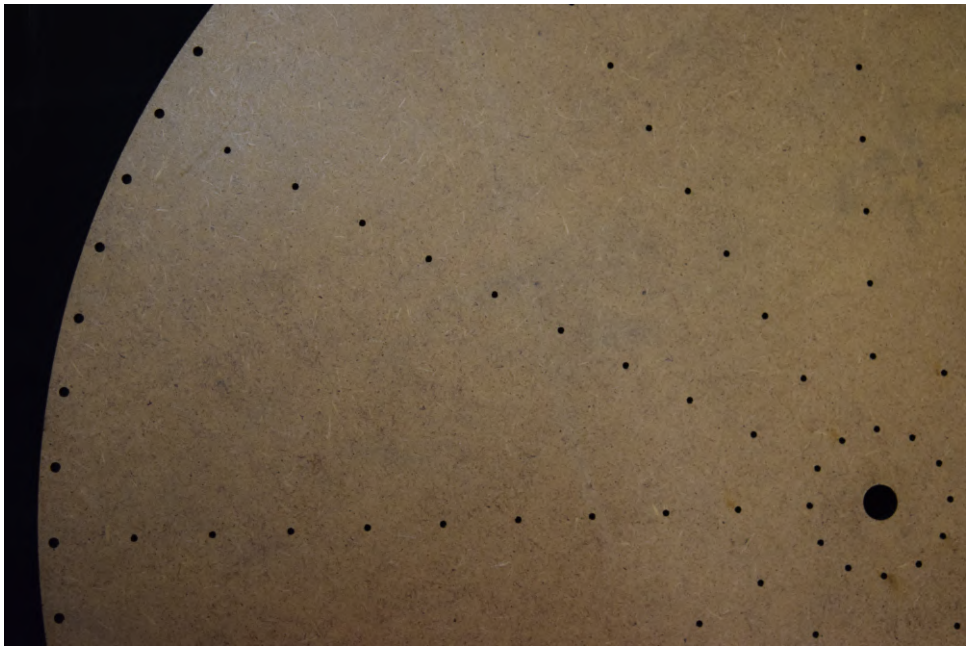


Figure 2.2: Close view of the disk: the hole in the center and the arrays of holes are visible. The disk has a radius of 28 cm and the smallest holes have a diameter of 3 mm.

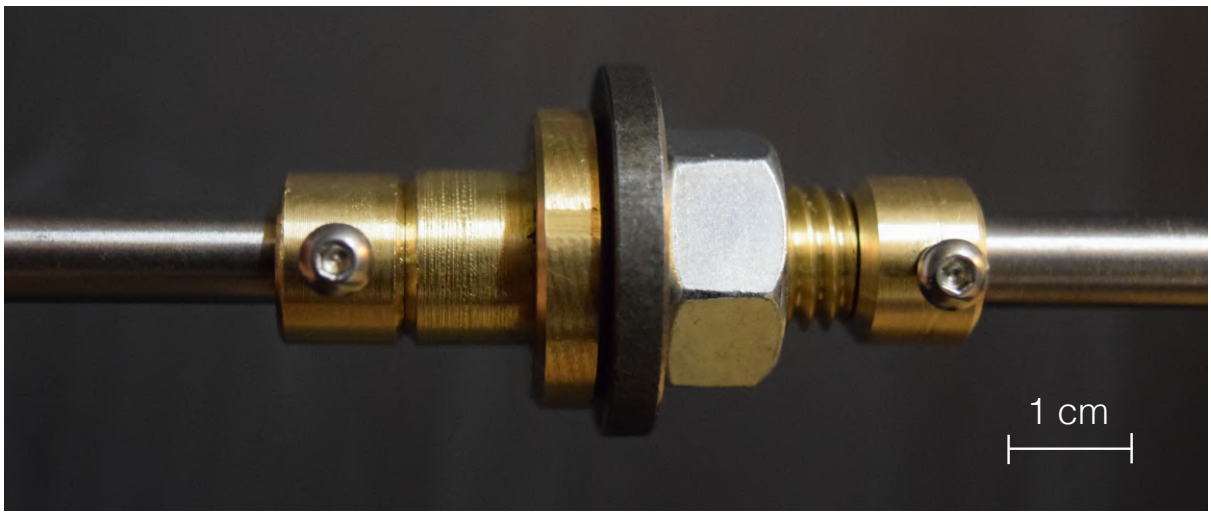


Figure 2.3: The part allowing the disk to rotate on the steel axis.

[1]To see the FreeCad files of the disk and the base of the Wheel, please refer to the following page: https://github.com/opatiny/tm/tree/master/freecad/wheel

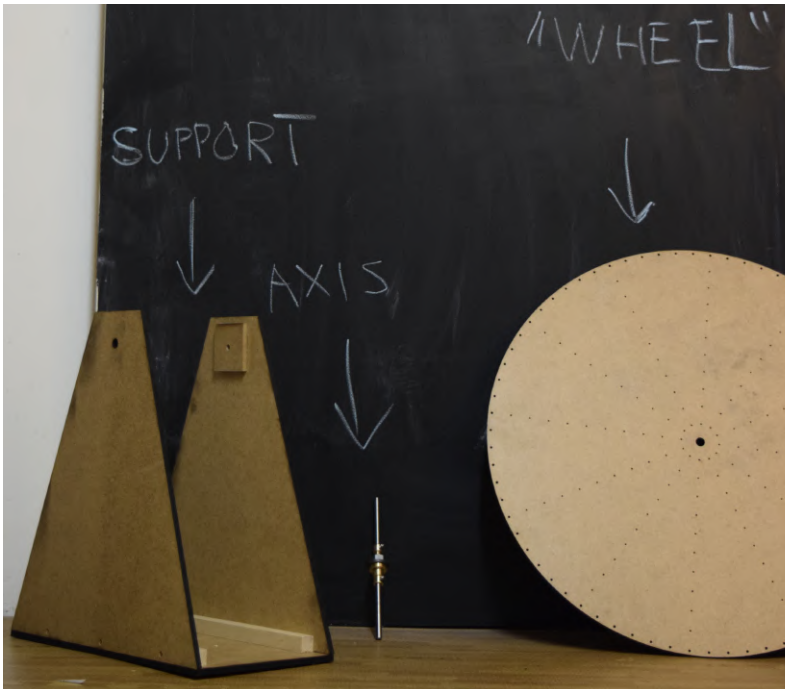


Figure 2.4: The base or support of the Wheel, the rotation axis and the wooden disk.

Our goal using this disk is to convince ourselves that, in the case of a rotating object that presents symmetries of the mass distribution around the rotation axis, it will rotate at a constant speed if no force is applied to it. To verify this assertion, we simply placed equal brass masses at different positions on the disk. Four cases of symmetry were tested: with no additional mass than the disk, with two masses placed on a line through the center of the disk, with three masses on an equilateral triangle and finally, with four masses places on the vertices of a square. The three last cases we enumerated can be seen on Figure 2.5.

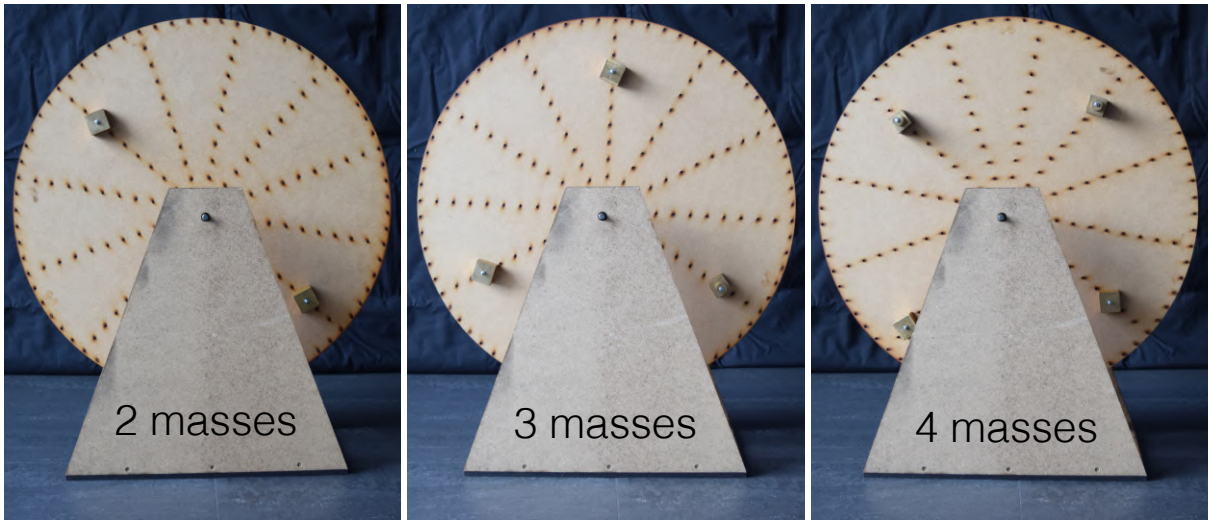


Figure 2.5: Three symmetry cases that were tested: segment, equilateral triangle and square.

All the measures we did are purely qualitative, since they had to help to forge a better instinct towards rotations problems. Yet, to confirm our conclusions, we tested a case with only one mass, i.e. an asymmetrical case. Then, it was noticed that they were only two balanced positions for the disk: when the mass is directly over the rotation axis, that

position is unstable, and when the mass is just under the rotation axis, this one being stable. Therefore, if we make the disk rotate, it will first accomplish whole revolutions, then oscillate for a while and finally stop in the configuration with the mass at the lower possible position (damped oscillatory motion). (We can guess that, in similar problems, the stablest position is always the one with the lower potential energy.)
On the contrary, for all the arrangements illustrated in Figure 2.5, it was seen that for any position of the disk, that one would remain balanced. Moreover, when the disk spun, it did not seem to accelerate and decelerate as it did when only one mass was fixed on it[1]. That point is important, because it allows the following conclusion: the angular speed of any rotating object with inhomogeneous mass, but a symmetrical distribution of it around the rotation axis, will remain constant if the sum of the external forces applied to it is null (Newton, *Lex* 2). At least in 2D (for a cylinder), we have shown through this experience that having masses at specific points around our cylinder won't be problematic. For example, if we need to place motors or the batteries on the border of the cylinder, this could be done without any consequences on the constancy of the cylinder's speeds as long as it is built with symmetries. If it is not the case, calibration at the scale of the software might be required to compensate the movements produced by the masses, which is really complex.

[1] All these observations can be seen in the movie: https://youtu.be/ovayvtKzIYs

2.3 Physical resolution

Goal: Express the formula that gives the speed of the cylinder depending on the speed of the mass in its center.

A cylinder that rolls thanks to a mass that is moved in it is not a rigid body. Therefore, this system must be divided into two bodies, so that the laws of physics can be applied. We consider a first body that is composed of the cylinder, the servos, the electronics and so on. The second one is the mass that is moved, as well as an imaginary axis, connected to the center of the cylinder. It is possible to make this assumption because, in the final function, one term is the torque that is produced by the motors, in our case three servos on the outside. This means that the function is generally the same, but the value of that torque is different if the motors are on the periphery of the cylinder or in its center. In addition, the distribution of the motors mass is included in the moment of inertia of the cylinder.

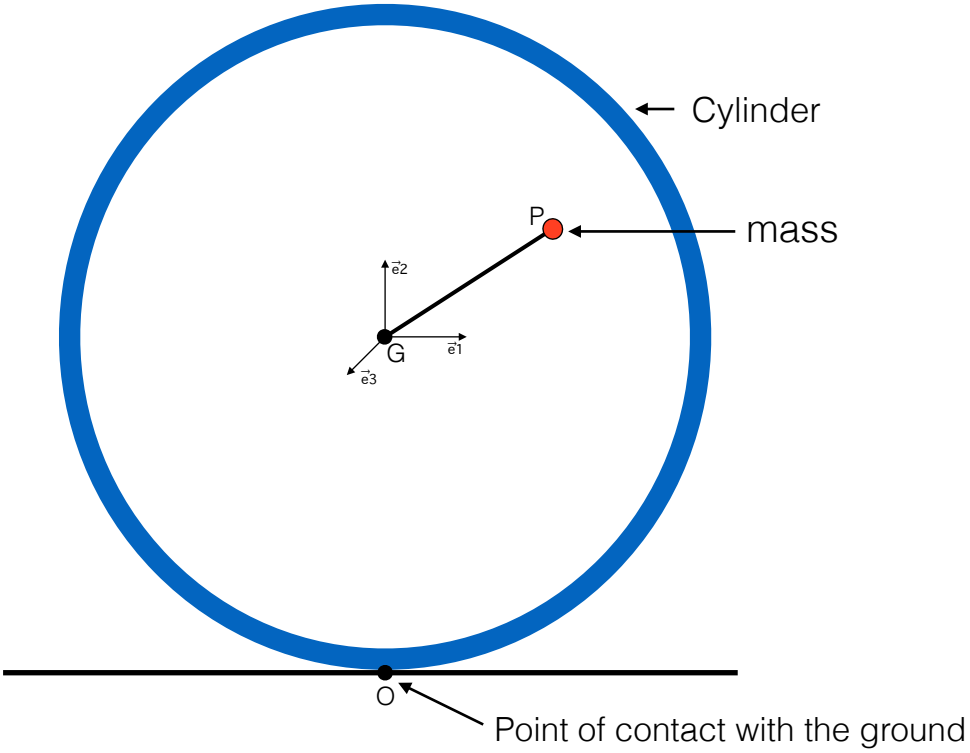


Figure 2.6: The system and the frame of reference considered in the problem.

The properties of the two bodies are described in Table 2.1.

	M	m
Mass	M	m
Point of reference	G	P
Moment of inertia	$I_{M,G}$	$I_{m,P}$
Translation speed	$\vec{v}_G$	$\vec{v}_P$
Rotation speed	$\vec{\Omega}$	$\vec{\omega}$

Table 2.1: Properties of M and m.

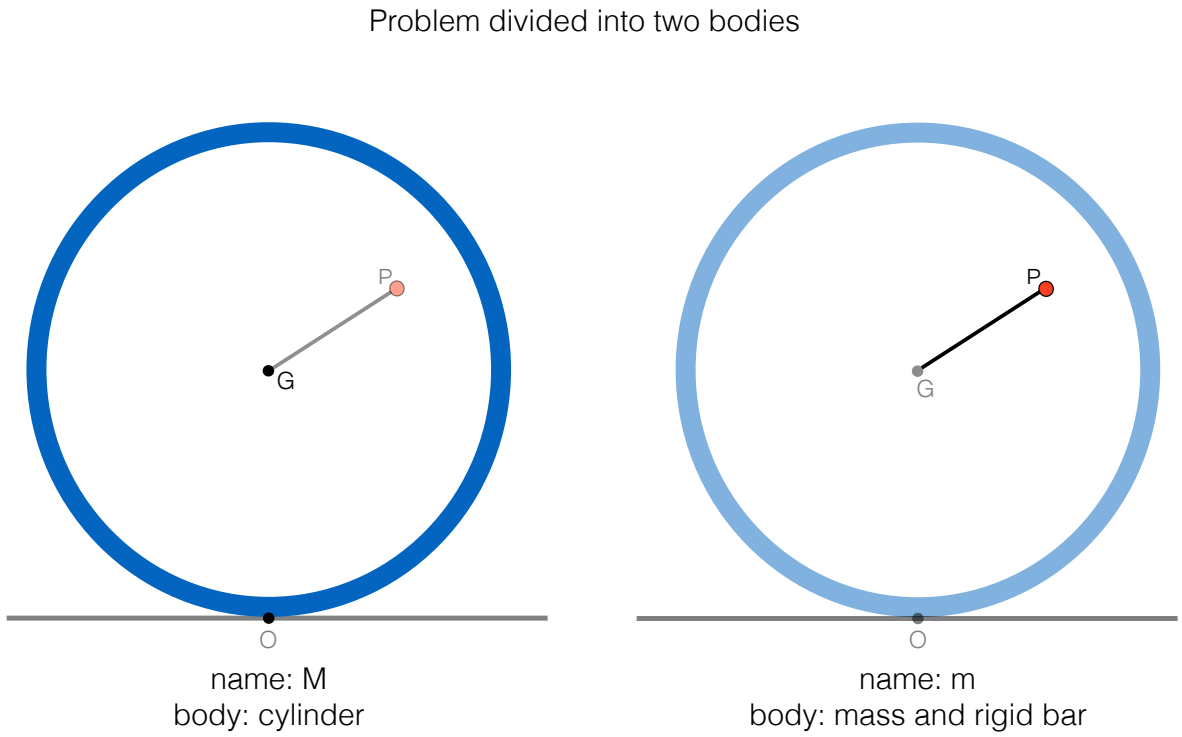


Figure 2.7: Scheme of the two bodies m and M

Laws and definitions

For more information about the properties and definitions of dot and vector product, please refer to Section 2.1.4.

Newton's second law for translations

$$N2, T : \sum_i \vec{F}_i^{ext} = m \cdot \vec{a} \tag{2.27}$$

Newton's second law for rotations

$$N2, R : \sum_\alpha \vec{M}_{O,\alpha} = I_O \cdot \vec{\alpha} \tag{2.28}$$

Newton's third law

$$N3 : \vec{F}_{A/B} = -\vec{F}_{B/A} \tag{2.29}$$

From rotation to translation speed of a point A

$$\vec{v}_A = \vec{v}_O + \vec{\omega} \wedge \vec{OA} \tag{2.30}$$

Vector product with a constant

$$(r\vec{a}) \wedge \vec{b} = \vec{a} \wedge r\vec{b} = r(\vec{a} \wedge \vec{b}) \tag{2.31}$$

Vector product with a sum of vectors

$$(\vec{a} + \vec{b}) \wedge \vec{c} = \vec{a} \wedge \vec{c} + \vec{b} \wedge \vec{c} \tag{2.32}$$

Vector product with a vector product

$$\vec{a} \wedge (\vec{b} \wedge \vec{c}) = \vec{b}(\vec{a} \cdot \vec{c}) - \vec{c}(\vec{a} \cdot \vec{b}) \tag{2.33}$$

Vector product derivative

$$(\vec{a} \wedge \vec{b})' = (\vec{a})' \wedge \vec{b} + \vec{a} \wedge (\vec{b})' \tag{2.34}$$

Vector derivative

$$\dot{\vec{AB}} = \vec{\omega} \wedge \vec{AB} \tag{2.35}$$

Forces

The forces applying on M are:

- Earth's gravity: $\vec{F}_{T,M} = Mg \cdot (-\vec{e}_2)$
- Ground's force: $\vec{F}_{ground} = F_{ground,1} \cdot \vec{e_1} + F_{ground,2} \cdot \vec{e_2} + F_{ground,3} \cdot \vec{e_3}$
- The force of m on M: $\vec{F}_{m/M} = F_{m/M,1} \cdot \vec{e_1} + F_{m/M,2} \cdot \vec{e_2}$

On the other hand, the forces applying on m are:

- Earth's gravity: $\vec{F}_{T,m} = mg \cdot -\vec{e}_2$
- The force of M on m: $\vec{F}_{M/m} = F_{M/m,1} \cdot \vec{e_1} + F_{M/m,2} \cdot \vec{e_2}$

Literal application

First of all, using Newton's third law, it can be seen that $\vec{F}_{M/m} = -\vec{F}_{m/M}$, consequently, $-\vec{F}_{M/m}$ will be called $\vec{F}$. Newton's second law for translations is then applied for both M and m.

$$M: \quad \vec{F}_{T,M} + \vec{F}_{ground} + \vec{F} = M\dot{\vec{v}}_G \tag{2.36}$$

$$m: \quad \vec{F}_{T,m} - \vec{F} = m\dot{\vec{v}}_P \tag{2.37}$$

The same is done with Newton's second law for rotations. G is considered as a reference point for the torques applied to M, and P is the reference for the torques applied to m. It is done so, because these are the easiest points for which the moment of inertia of the two bodies can be calculated.

$$M: \quad \vec{GG} \wedge \vec{F}_{T,M} + \vec{GO} \wedge \vec{F}_{ground} + \vec{C} = I_{M,G}\dot{\vec{\Omega}} \tag{2.38}$$

$$m: \quad \vec{PP} \wedge \vec{F}_{T,m} + \vec{PG} \wedge (-\vec{F}) - \vec{C} = I_{m,P}\dot{\vec{\omega}} \tag{2.39}$$

In Equations 2.38 and 2.39, $\vec{C}$ is the torque produced by the servos / the motor. Indeed, this electronic part brings additional energy to the system, since it uses the energy of a battery either to maintain the mass at a certain position, or to make it move. In addition, the first terms of these two equations are null, because so are the lengths of the vectors $\vec{GG}$ and $\vec{PP}$.

To solve the equations, it is first needed to isolate $\vec{F}$ in Equation 2.37.

$$\vec{F} = m(\vec{g} - \dot{\vec{v}}_P) \tag{2.40}$$

By replacing this value of $\vec{F}$ in Equation 2.36 and ordering the terms, $\vec{F}_{ground}$ can be isolated.

$$\vec{F}_{ground} = M(\dot{\vec{v}}_G - \vec{g}) + m(\dot{\vec{v}}_P - \vec{g}) \tag{2.41}$$

Next, the values of the two forces $\vec{F}$ and $\vec{F}_{ground}$ should be replaced in Equations 2.38 and 2.39, but it would result in having both the angular and translation accelerations of M and m in the same equations. To avoid that, it is needed to find the link between the two accelerations. Definition 2.30 is applied to M and m's speeds. The reference points for the speeds of the two bodies are O for M and G for m. Here are the resulting equations.

$$M: \quad \vec{v}_G = \vec{v}_O + \vec{\Omega} \wedge \vec{OG} \tag{2.42}$$
$$m: \quad \vec{v}_P = \vec{v}_G + \vec{\omega} \wedge \vec{GP} \tag{2.43}$$

Since the cylinder is rolling and not slipping on the ground, the speed of point O is null. Consequently, the two equations become:

$$M: \quad \vec{v}_G = \vec{\Omega} \wedge \vec{OG} \tag{2.44}$$
$$m: \quad \vec{v}_P = \vec{\Omega} \wedge \vec{OG} + \vec{\omega} \wedge \vec{GP} \tag{2.45}$$

Nevertheless, what is needed in Equations 2.40 and 2.41 is not angular speed, but angular acceleration, therefore, the two equations for speed must be derived using definition 2.34.

$$M: \quad \dot{\vec{v}}_G = \dot{\vec{\Omega}} \wedge \vec{OG} + \vec{\Omega} \wedge \dot{\vec{OG}} \tag{2.46}$$
$$m: \quad \dot{\vec{v}}_P = \dot{\vec{\Omega}} \wedge \vec{OG} + \vec{\Omega} \wedge \dot{\vec{OG}} + \dot{\vec{\omega}} \wedge \vec{GP} + \vec{\omega} \wedge \dot{\vec{GP}} \tag{2.47}$$

However, the term $\vec{\Omega} \wedge \dot{\vec{OG}}$ is null, because the vector $\vec{OG}$ does not vary. In addition, using Equation 2.35, we can express $\dot{\vec{GP}}$ as $\vec{\omega} \wedge \vec{GP}$. These modifications lead to the following final values of the angular accelerations:

$$M: \quad \dot{\vec{v}}_G = \dot{\vec{\Omega}} \wedge \vec{OG} \tag{2.48}$$
$$m: \quad \dot{\vec{v}}_P = \dot{\vec{\Omega}} \wedge \vec{OG} + \dot{\vec{\omega}} \wedge \vec{GP} + \vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP}) \tag{2.49}$$

These values can now be incorporated into Equations 2.40 and 2.41.

$$\vec{F} = m\vec{g} - m(\dot{\vec{\Omega}} \wedge \vec{OG} + \dot{\vec{\omega}} \wedge \vec{GP} + \vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP})) \tag{2.50}$$
$$\vec{F}_{ground} = M(\dot{\vec{\Omega}} \wedge \vec{OG}) - M\vec{g} + m(\dot{\vec{\Omega}} \wedge \vec{OG} + \dot{\vec{\omega}} \wedge \vec{GP} + \vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP})) - m\vec{g} \tag{2.51}$$

Now, $\vec{F}$ and $\vec{F}_{ground}$ can finally be replaced in the equations that use Newton's second law for rotation (Equations 2.53 and 2.52)

$$\begin{aligned} M: \quad & \vec{GO} \wedge \left[M(\dot{\vec{\Omega}} \wedge \vec{OG}) - M\vec{g} + m(\dot{\vec{\Omega}} \wedge \vec{OG} + \dot{\vec{\omega}} \wedge \vec{GP} + \vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP})) - m\vec{g}\right] \\ & + \vec{C} = I_{M,G}\dot{\vec{\Omega}} \end{aligned} \tag{2.52}$$

$$m: \quad \vec{PG} \wedge \left[m\vec{g} - m(\dot{\vec{\Omega}} \wedge \vec{OG} + \dot{\vec{\omega}} \wedge \vec{GP} + \vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP}))\right] - \vec{C} = I_{m,P}\dot{\vec{\omega}} \tag{2.53}$$

To develop these equations, definition 2.32 is first used to split the vector products.

$$\begin{aligned} M: \quad & \vec{GO} \wedge M(\dot{\vec{\Omega}} \wedge \vec{OG}) + \vec{GO} \wedge (-M\vec{g}) + \vec{GO} \wedge m(\dot{\vec{\Omega}} \wedge \vec{OG}) + \vec{GO} \wedge m(\dot{\vec{\omega}} \wedge \vec{GP}) \\ & + \vec{GO} \wedge m(\vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP})) + \vec{GO} \wedge (-m\vec{g}) + \vec{C} = I_{M,G}\dot{\vec{\Omega}} \end{aligned} \tag{2.54}$$

$$\begin{aligned} m: \quad & \vec{PG} \wedge m\vec{g} + \vec{PG} \wedge [-m(\dot{\vec{\Omega}} \wedge \vec{OG})] + \vec{PG} \wedge [-m(\dot{\vec{\omega}} \wedge \vec{GP})] \\ & + \vec{PG} \wedge [-m(\vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP}))] - \vec{C} = I_{m,P}\dot{\vec{\omega}} \end{aligned} \tag{2.55}$$

Then, by using the vector property of Equations 2.31 and 2.33, the vector products of a vector with a vector product can be developed. All these parts have been separated, to make it easier to simplify the terms.

$$\begin{aligned} M: \quad & \vec{GO} \wedge M(\dot{\vec{\Omega}} \wedge \vec{OG}) = M\dot{\vec{\Omega}}(\vec{GO} \cdot \vec{OG}) - \vec{OG}(\vec{GO} \cdot M\dot{\vec{\Omega}}) \\ & \vec{GO} \wedge (-M\vec{g}) = \vec{0} \\ & \vec{GO} \wedge m(\dot{\vec{\Omega}} \wedge \vec{OG}) = m\dot{\vec{\Omega}}(\vec{GO} \cdot \vec{OG}) - \vec{OG}(\vec{GO} \cdot m\dot{\vec{\Omega}}) \\ & \vec{GO} \wedge m(\dot{\vec{\omega}} \wedge \vec{GP}) = m\dot{\vec{\omega}}(\vec{GO} \cdot \vec{GP}) - \vec{GP}(\vec{GO} \cdot m\dot{\vec{\omega}}) \\ & \vec{GO} \wedge m(\vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP})) = \vec{GO} \wedge \left[\vec{\omega}(m\vec{\omega} \cdot \vec{GP}) - \vec{GP}(m\vec{\omega} \cdot \vec{\omega})\right] \\ & \vec{GO} \wedge (-m\vec{g}) = \vec{0} \end{aligned} \tag{2.56}$$

$$\begin{aligned} m: \quad & \vec{PG} \wedge m\vec{g} = -PG_1 mg \cdot \vec{e_3} \\ & \vec{PG} \wedge [-m(\dot{\vec{\Omega}} \wedge \vec{OG})] = -m\dot{\vec{\Omega}}(\vec{PG} \cdot \vec{OG}) + \vec{OG}(\vec{PG} \cdot m\dot{\vec{\Omega}}) \\ & \vec{PG} \wedge [-m(\dot{\vec{\omega}} \wedge \vec{GP})] = -m\dot{\vec{\omega}}(\vec{PG} \cdot \vec{GP}) + \vec{GP}(\vec{PG} \cdot m\dot{\vec{\omega}}) \\ & \vec{PG} \wedge [-m(\vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP}))] = \vec{PG} \wedge \left[-\vec{\omega}(m\vec{\omega} \cdot \vec{GP}) + \vec{GP}(m\vec{\omega} \cdot \vec{\omega})\right] \end{aligned} \tag{2.57}$$

To simplify Equations 2.57 and 2.56, a hypothesis must be set down.

Hypothesis 1 *The magnitude of the angular speed of m and M is only non zero under $\vec{e_3}$. In other words:*

$$\vec{\omega} = \omega \cdot \vec{e_3} \quad \textit{and} \quad \vec{\Omega} = \Omega \cdot \vec{e_3} \tag{2.58}$$

In addition, the vector $\vec{GP}$ is described with its component $GP_1 \cdot \vec{e_1}$ and $GP_2 \cdot \vec{e_2}$. To obtain the result of Equations 2.59 and 2.60, the dot product of all pairs of vectors perpendicular to each other have been suppressed. Also, it should not be forgotten that the vector $\vec{GO}$ points along $\vec{e_2}$ negative.

$$\begin{aligned} M: \quad & \vec{GO} \wedge M(\dot{\vec{\Omega}} \wedge \vec{OG}) = -M\dot{\vec{\Omega}}GO^2 \\ & \vec{GO} \wedge m(\dot{\vec{\Omega}} \wedge \vec{OG}) = -m\dot{\vec{\Omega}}GO^2 \\ & \vec{GO} \wedge m(\dot{\vec{\omega}} \wedge \vec{GP}) = -m\dot{\vec{\omega}}GOGP_2 \\ & \vec{GO} \wedge m(\vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP})) = (-\vec{GO}) \wedge \vec{GP}m\omega^2 = -GOGP_1 m\omega^2 \cdot \vec{e_3} \end{aligned} \tag{2.59}$$

$$
\begin{aligned}
m: \quad & \vec{PG} \wedge m\vec{g} = -PG_1 mg \cdot \vec{e_3} \\
& \vec{PG} \wedge [-m(\dot{\vec{\Omega}} \wedge \vec{OG})] = m\dot{\vec{\Omega}} PG_2 OG \\
& \vec{PG} \wedge [-m(\dot{\vec{\omega}} \wedge \vec{GP})] = m\dot{\vec{\omega}} GP^2 \\
& \vec{PG} \wedge [-m(\vec{\omega} \wedge (\vec{\omega} \wedge \vec{GP}))] = -\vec{GP} \wedge \vec{GP} m\omega^2 = \vec{0}
\end{aligned} \tag{2.60}
$$

Substituting these final terms into Equations 2.54 and 2.55, the following result is obtained.

$$M: \quad -M\dot{\vec{\Omega}} GO^2 - m\dot{\vec{\Omega}} GO^2 - m\dot{\vec{\omega}} GOGP - GOGP_1 m\omega^2 \cdot \vec{e_3} + \vec{C} = I_{M,G}\dot{\vec{\Omega}} \tag{2.61}$$

$$m: \quad -\vec{C} - PG_1 mg \cdot \vec{e_3} - m\dot{\vec{\Omega}} PG_2 OG - m\dot{\vec{\omega}} GP^2 = I_{m,P}\dot{\vec{\omega}} \tag{2.62}$$

The final step to arrange these two equations is to move all the terms for one angular speed on the left side of the equation, leaving the remaining ones on the other side.

$$M: \quad \dot{\vec{\Omega}}[I_{M,G} + GO^2(M+m))] = -m\dot{\vec{\omega}} GOGP_2 - m\omega^2 GOGP_1 \cdot \vec{e_3} + \vec{C} \tag{2.63}$$

$$m: \quad \dot{\vec{\omega}}(I_{m,P} + mGP^2) = -m\dot{\vec{\Omega}} PG_2 OG - PG_1 mg \cdot \vec{e_3} - \vec{C} \tag{2.64}$$

Finally, these two equations can be written under a scalar form, for everything is along $\vec{e_3}$ (positive or negative). Since $\vec{\Omega}$ points towards $-\vec{e_3}$, it has been the direction chosen to express the equations. Yet, it will be noticed that some signs have changed.

$$M: \quad \dot{\Omega}[I_{M,G} + GO^2(M+m)] = -m\dot{\omega} GOGP_2 - m\omega^2 GOGP_1 + C \tag{2.65}$$

$$m: \quad \dot{\omega}(I_{m,P} + mGP^2) = -m\dot{\Omega} PG_2 OG - PG_1 mg - C \tag{2.66}$$

It can now be seen that Equations 2.65 and 2.66 are coupled differential equations, which signifies they do both include the normal form of a variable and its derivative. Since the resolution of this kind of system requires a high level of mathematics, it won't be done in this work. This could have been useful to understand some of the experimental results that we have obtained, but we did not have the necessary tools to really fully understand these last equations. Yet, it was really interesting to have another point of view on this project, a strictly theoretical one.

Remark 1 *Note that these equations can as well express the linear movement of a cylinder as the linear movement of a sphere, the difference would only stay in the value of the moment of inertia of the body M and on the torque $\vec{C}$ of the motors.*

2.3.1 Cylinder movement when the mass position is constant

What was eventually developed in the code was a feature that allows to keep the mass still with respect to the contact point of the cylinder with the floor, at a position where the torque produced by the mass weight is optimal. This case is when $\vec{GP}$ is parallel to the ground (horizontal). The advantage of keeping the mass angular velocity null is that then, the cylinder movement is supposed to be a lot easier. To solve this problem, we can work with scalar equations. This is allowed, because of hypothesis 1, which implies that all the torques and the angular acceleration are in the same direction. Also, the only force that produces a non-zero torque is $\vec{F}_{T,M}$, that is perpendicular to $\vec{GP}$. So, the torque of this force, $M_{T,m}$ is only:

$$M_{T,m} = GPmg \tag{2.67}$$

In addition, we know the scalar forms of N2,R and Equation 2.17, which is the link between translation and angular acceleration:

$$\sum M = I\alpha \tag{2.68}$$

$$a = \alpha r \tag{2.69}$$

In this last equation, r corresponds to OG in this case. If these three equations are put together, we obtain:

$$GPmg = I_{M,O}\frac{a}{r} \tag{2.70}$$

Yet, we can use definition 2.71 to calculate the inertia of the mass for an axis that passes through O, because we consider that we now its inertia with respect to G.

$$I_O = I_{M,G} + OG^2 \cdot (M + m) \tag{2.71}$$

Which finally allows to isolate a:

$$a = \frac{OG \cdot GPmg}{I_{M,G} + OG^2 \cdot (M + m)} \tag{2.72}$$

This last equation allows an easy calculation of the theoretical cylinder translation acceleration. Indeed, the equations being solved, it is showed that the movement of the cylinder is a uniform acceleration in this specific case. When experiencing with the prototypes that we developed, this assertion was confirmed.

Remark 2 *Note that the result of Equation 2.72 is exactly the same as the one that is obtained with Equations 2.65 and 2.66 if ω and $\dot{\omega}$ are null.*

Chapter 3

Already existing projects

3.1 The Sphero

The Sphero is a start-up product now widely sold all over the world. It consists of a little sphere that can be held in a hand. What is so special about it is that it can be controlled with a smartphone. It can roll over various surfaces, climb slopes, it is even amphibian. The Sphero has a very resistant polycarbonate shell, which makes it shockproof too. To allow the outer sphere to be perfectly round and sealed, the ball is charged inductively.

Figure 3.1: Second version of the Sphero, the wheels that make it move can be seen in white. [12]

The important point is to explain the technology that this project uses to produce the movement of the ball. Indeed, it is completely different than displacing the gravity center of the object and using the torque it produces to induce movement. Actually, the Sphero

can be seen like a hamster ball: there is a little car inside it, with its wheels laid on the inner surface of the shell. So, when the wheels rotate, they make the whole sphere move in the same direction. To turn, the Sphero just uses a little wheel perpendicular to the ones that accelerate it straight.

What is interesting in this project is the smartphone control of the Sphero. The developers have developed a specific mobile application intended for this purpose, which allows the control to be very intuitive since it can be done only with joysticks. However, the most astute aspect of this product is the induction charging system, which is the only charging solution that allows to have a totally hermetic sphere.
To explain briefly, induction is a technology that allows to transfer energy from one point to another without contact or wire[1].

3.2 The Cubli

The Cubli is the name of a swiss project developed by students at ETH Zürich which began in February 2011 and is lead by the researchers Michael Muehlebach and Prof. Raffaello D'Andrea. The final prototype basically consists of a 15 cm sided cube that can move by flipping from one of its sides to another. It can also stand on one of its edges or corners and maintain that position for a longer time. This signifies that even if a light force is applied to it, the cube will slightly move and then quickly rebalance itself to come back to its original position.

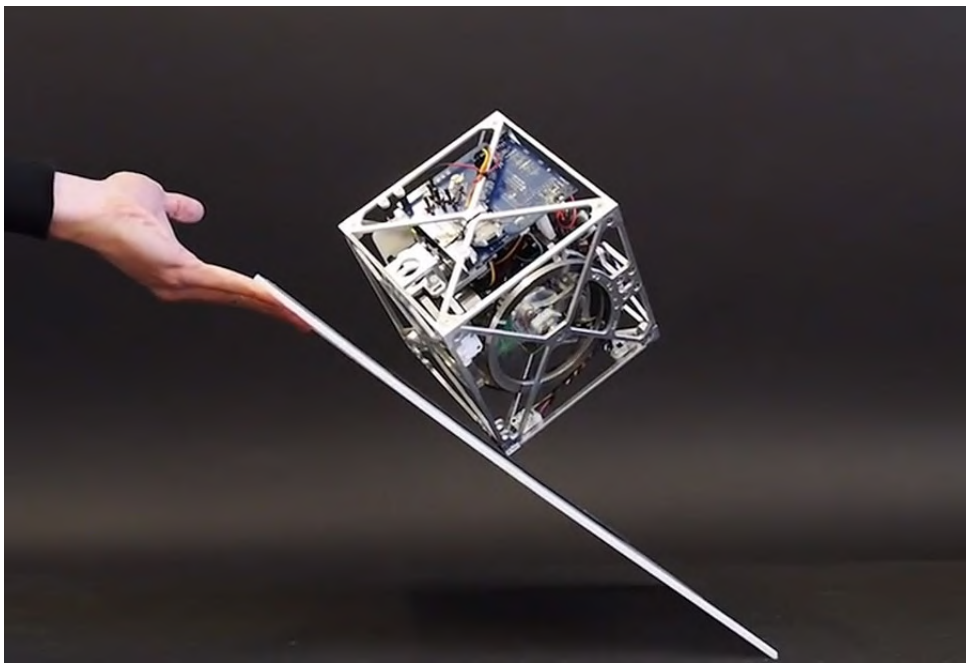


Figure 3.2: The Cubli balancing on it's side, on a slope. [16]

[1]In case of charging, it works thanks to an oscillating magnetic field emitted by one coil, that matches the resonant frequency of another coil. In these conditions, The magnetic field "induces" an electrical current in the second coil. This energy can then be used to recharge a battery, for example [18, 59].

Figure 3.3: The plates balanced by gymnasts are inverted pendulums. [13]

To describe this little cube with a physical eye, it can be seen as an inverted pendulum, which is similar to a regular pendulum, but its mass is situated above and not under the point where it is fixed. That signifies that the mass - support connection has to be rigid. In addition, inverted pendulums are unstable, since they do only rest on the ground on a very little surface, that is not fixed to be unable to rotate. Therefore forces need to be applied constantly to keep them balanced (see example at Figure 3.3).
In the Cubli project, the technology used to keep the cube at the desired position is reaction wheels, which are fundamentally a heavy disk fixed orthogonally to a motor's axis. The disk's weight is as far as possible from its center to enhance its inertia. Increasing the disk's angular acceleration leads to a proportional increase of its momentum, which must be compensated by an equivalent momentum in the other direction (because momentum is conserved). It is this procedure that allows the Cubli to move.

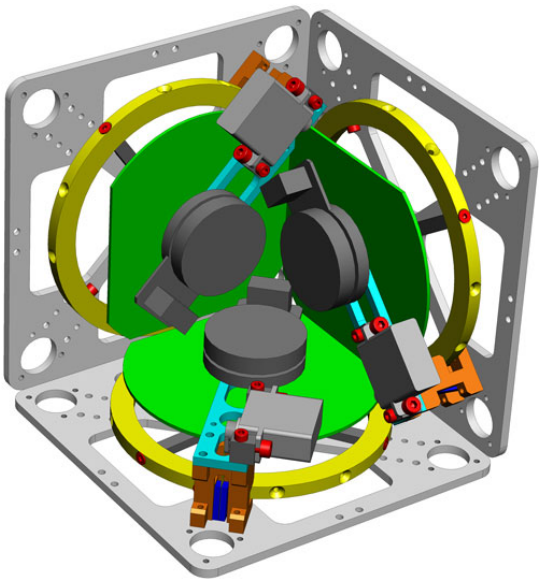


Figure 3.4: Inner components of the Cubli, the three reaction wheels are in yellow. [16]

The principle of this device is described as follows in *The Cloud, Paper Planes, and the Cube*, the thesis article of the Cubli's inventor.

> "Reaction wheels mounted on three faces of the cube rotate at high angular velocities and then brake suddenly, causing the Cubli to jump up. Once the Cubli has almost reached the corner stand up position, controlled motor torques are applied to make it balance on its edge or corner [3]."

Note that the three wheels are orthogonal.

Of course, this project is not so close to ours, but the analogy between a cube that moves and a sphere can still be done easily. We have been particularly interested in the ability of the Cubli to remain balanced on a slope (Figure 3.2), because it might be a feature we could give to our prototypes.

3.3 Review of the literature

In this section, the scientific literature concerning spherical robots is reviewed. This aims to realize to what extend our project is innovative and what gap it could fill in this field.

The study of spherical robots has been developing intensely since more than twenty years. Already in 1996, an article promoted the advantages of this type of remote controlled robots. It said, for instance:

> "The spherical construction offers extraordinary motion properties in cases where turning over or falling down are risks for the robot to continue motion. Also it has amazing capability to recover from collisions with obstacles or another robots traveling in the environment [60]."

This article does also propose application fields for this kind of robots:

> "In its simplest version the rolling robot can be used for example surveillance or monitoring applications with sensors mounted either on the inner mechanics and sensing remotely through the cover (like a millimeter wave radar or alpha spectrometer), or mounted on the cover and communicating wirelessly [60]."

These are obviously all things that we could imagine to do with a spherical robot using our technology to move. Yet, in this article, the method they use is similar to the one of the Sphero: "This construction is moved by a motor driven wheel that can be turned to cause the heading of the motion to change [60]."
Later, in 1999, was imagined one of the few approaches to move a sphere that uses a principle similar to ours (moving the center of mass of the sphere). The idea is described as follows in the article *Simple Motion Planning Strategies for Spherobot: A Spherical Mobile Robot*:

> "We are engaged in research and development of Spherobot, a mobile robot with a spherical exoskeleton and a novel internal propulsion mechanism. The propulsion mechanism will distribute weights radially along spokes fixed inside the sphere and enable the robot to accelerate, move with constant velocity, or servo at a point [61]."

In 2012, a paper entitled *A Review of Active Mechanical Driving Principles of Spherical Robots* was written about the different concepts of spherical robots developed until then. An interesting point for us is that it exposes the advantages of these robots and how they were considered among scientists at the time:

> "The quintessential spherical robot would have true holonomy[1] and hence would be able to move in any direction without having to change its orientation. The current research direction of spherical robots is heavily focused on the internal mechanics and the corresponding control systems. One design has not yet emerged dominantly among the others and diverse methodologies of internal driving mechanics have resulted in a wide range of robotic characteristics and capabilities [1]."

This challenge is described more precisely a few lines further:

> "For true holonomy, the research challenge becomes developing an internal drive mechanism that can provide omnidirectional output torque to a sphere that can arbitrarily rotate around it, regardless of the orientation of either the sphere or the drive mechanism. In essence, the inner mechanics must be able to rotate three-dimensionally independently from the outer shell. Since the outer shell must be connected to the inner mechanics in some manner, this poses a difficult design challenge [1]."

Before the importance of this quotation is explained, it must just be added that in 2015, further studies about spherical robots were made, which still used a little car inside a rigid hermetic shell to move the sphere [2].
So, the reason why these quotations are important is that they show that our approach to induce the movement by displacing the sphere's center of mass is original and was not really considered before in scientific papers.

[1] For the author, holonomy is the ability for a vehicle to be able to move in any direction independently of its orientation.

Chapter 4

Approach

In this chapter, we expose the different approaches that were invented to displace the mass inside the cylinder. We try to find the ones that could be used both in a cylinder and a sphere. The goal is to discuss about the different possibilities that we have invented to fulfill the movement and to find the most appropriate solution. The final choices are explained and discussed.

4.1 How to move the mass?

In this first section, the technology that can use to move the mass in the center are discussed. Remember that to make the cylinder roll, its gravity center must be moved and generally, the best way to do that is to position a mass away from the center of the object. Many possibilities exist to achieve that, some of them have been listed underneath. In every part of this section, the general principle of the technology is described, as well as its advantages and disadvantages and if it might, or not, be used in a sphere (or whether if it does only work in a cylinder). Another information that is given is the zone that the mass can reach (the reachable zone), because the wider it is, the bigger the acceleration of the cylinder might be.

4.1.1 Stepper motor in the center

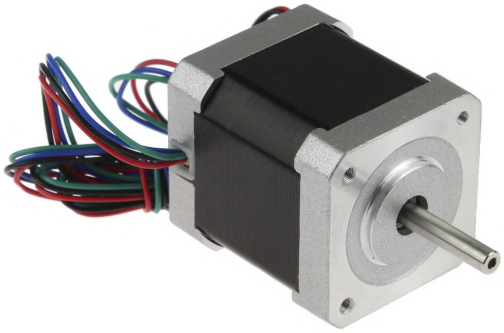


Figure 4.1: A stepper motor. [9]

A stepper is a motor that moves from a given angle at every electric impulsion it receives, using electromagnets. Thanks to the technology it uses, it can make an infinite number

of rotations. This kind of motor can rotate in both directions. This technology might easily be used in a cylinder, by placing one stepper in the center and by attaching a mass on an arm to it (Figure 4.1).

Advantages

- The control of the position is very easy because it is directly linked to the movement of the stepper.
- The mass of the stepper is in the center of the cylinder, so it is symmetrical and it is not adding a big moment of inertia.
- Easy to obtain.

Disadvantages

- If the stepper misses a step (for example if there is too much strength on it), the position of the mass is lost.
- Fairly heavy.
- Drains a lot of current.
- The technology would be difficult to adapt to a sphere.

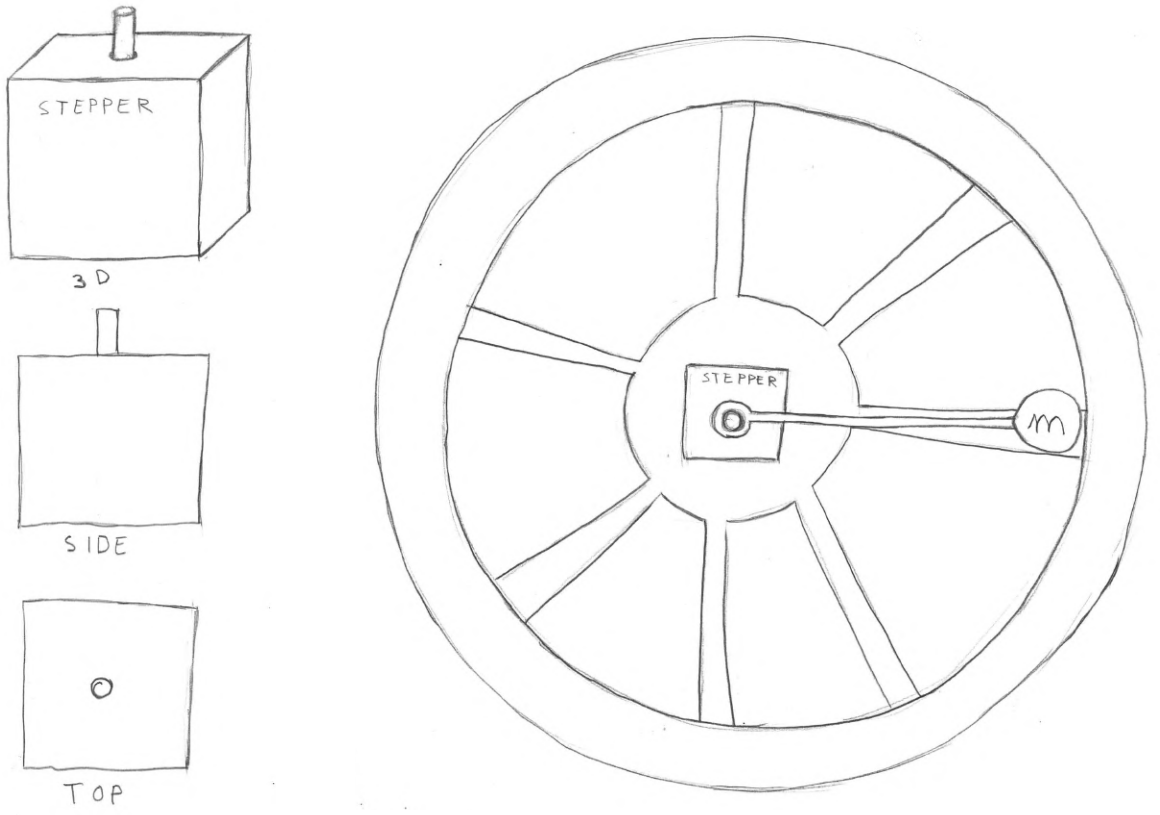


Figure 4.2: Approach with one stepper in the center.

4.1.2 Pulleys

What is called a "pulley" in this approach is a stepper on which a horizontal wheel with a guide slot is placed. In the slot, a rope could wind, with one extremity fixed to the wheel, whereas the other one would be attached on the mass in the center. Finally, a guiding piece would allow to keep the rope aligned with the guide slot. Three of these pulleys could be fixed on the outside of a cylinder (or in the center, so that their mass

contributes to the mass displaced), and the three steppers turning in one direction or the other, in a synchronized way, would make the mass move (Figure 4.4).

Advantages

- The displacement of the mass is significant, this will allow to accelerate the cylinder a lot more than in other approaches.
- The mass moves in a lot more instinctive way than with servo motors, because the movements are linear.

Disadvantages

- Ropes or chains need to be used, which it is not easy to control in a mechanical system (it can be worn, get tangled,...).
- Using three steppers away from the center of the system will create a big moment of inertia since these motors are heavy. This problem is not existing if the steppers are placed in the middle of the cylinder.

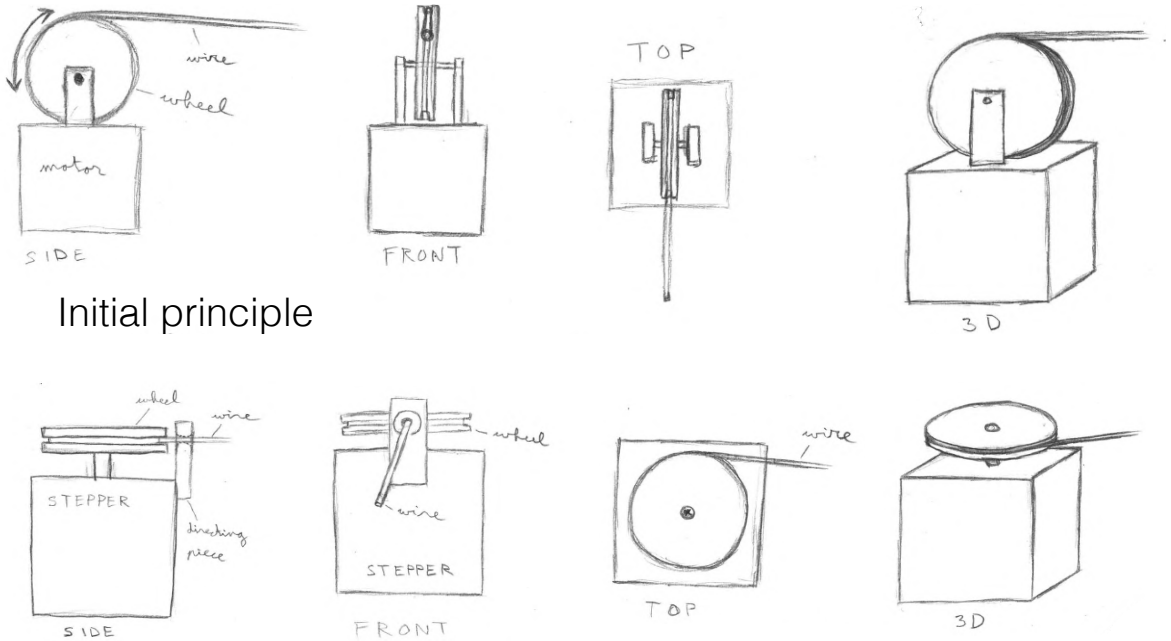


Figure 4.3: Explanation of the "pulley" system.

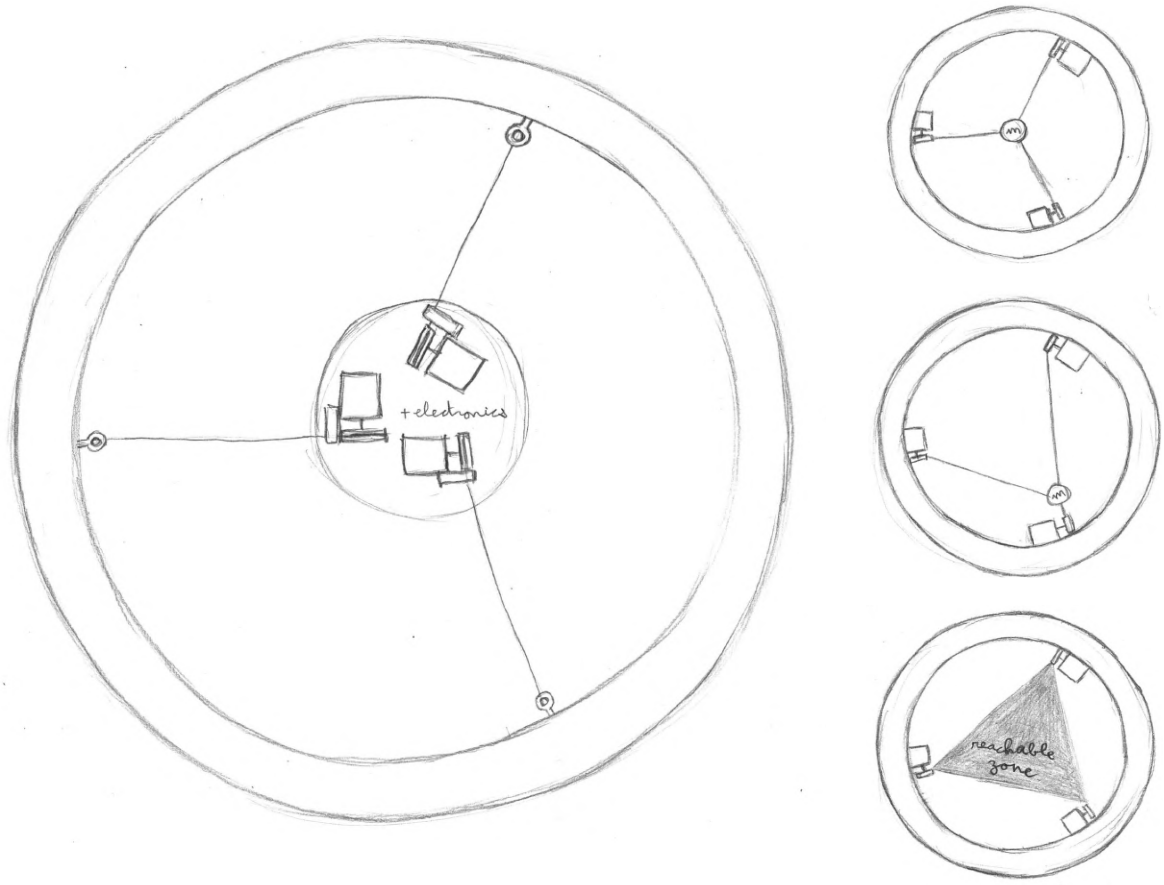


Figure 4.4: Approach with three "pulleys" in the center.

4.1.3 Rack and pinion drives

Again, this section is about an invented technology similar to two rack and pinions drives glued together and motorized, so that a little truck composed of the geared wheels and of a motor can go up or down the geared bar. To move a mass, three of these should be used, the three trucks connected together to form a platform (the mass in the center), whereas one extremity of the geared bars would be fixed to the outer cylinder so that they are free to rotate. To understand better this approach, refer to Figure 4.6.

Figure 4.5: A standard rack and pinion driver. [7]

Advantages

- The movement of the mass is defined by the position of the trucks on the rigid bars (linear), this is easy to determine.
- The mass of the trucks additions to the mass displaced.
- Everything is mechanical and reliable.

Disadvantages

- They might be some tensions applied on the racks and pinions drives, because of the angles they form one with another.
- It can be used in a sphere, but it might be difficult to make the geared bars cross so that they can move.
- We do not know if it exists.

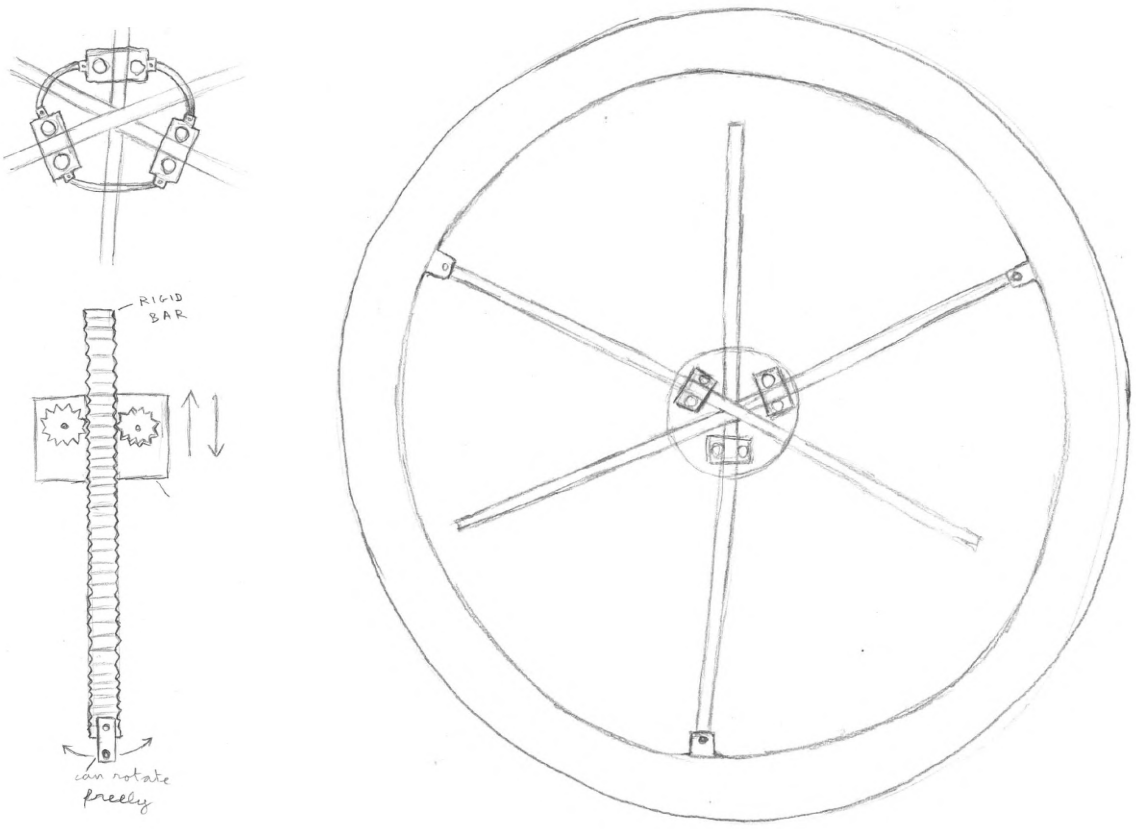


Figure 4.6: Approach with an invented technology similar to rack and pinion drives.

4.1.4 Long pistons

The principle is fairly simple: the extremity a "piston", something similar to an antenna that might expand or shorten (see Figure 4.7), would be fixed to the mass in the center with a rigid bar. Using three of them, connected to the mass with metallic rods would allow moving the mass in the center of the cylinder.

Advantages

- The movement is linear, on the contrary of servo motors, which might facilitate calculation.
- Easy to determine the position of the mass.
- The technology might be used in a sphere.
- The reachable zone is wide.

Disadvantages

- We don't know if it exists.
- There is a big torque at the end of the piston, it should be very resistant.
- The movement of the mass would be limited by the possible expansion of the piston.

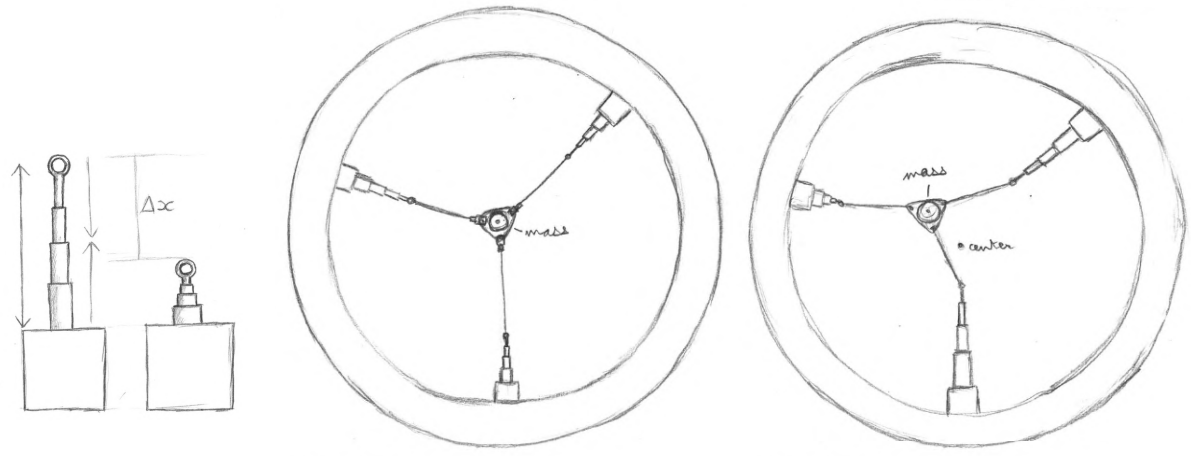


Figure 4.7: Approach with three "long pistons" on the outside of the cylinder.

4.1.5 Short pistons

This technique uses many little objects (that we call "short pistons") that have the property to be pushed out or pushed in so that their length varies. Like in Section 4.1.6, many of these pistons would be placed around the cylinder, and their synchronized movement could make a second cylinder move. This inner cylinder would only be slightly smaller than the external one, and it might contain all the electronics, or it might just be heavy.

Advantages

- Even if each piston is not very strong, all of them together must be able to displace a big mass.
- It could be used in a sphere.

Disadvantages

- Many of them on the outside will make an important mass to move.
- We do not know if it exists.

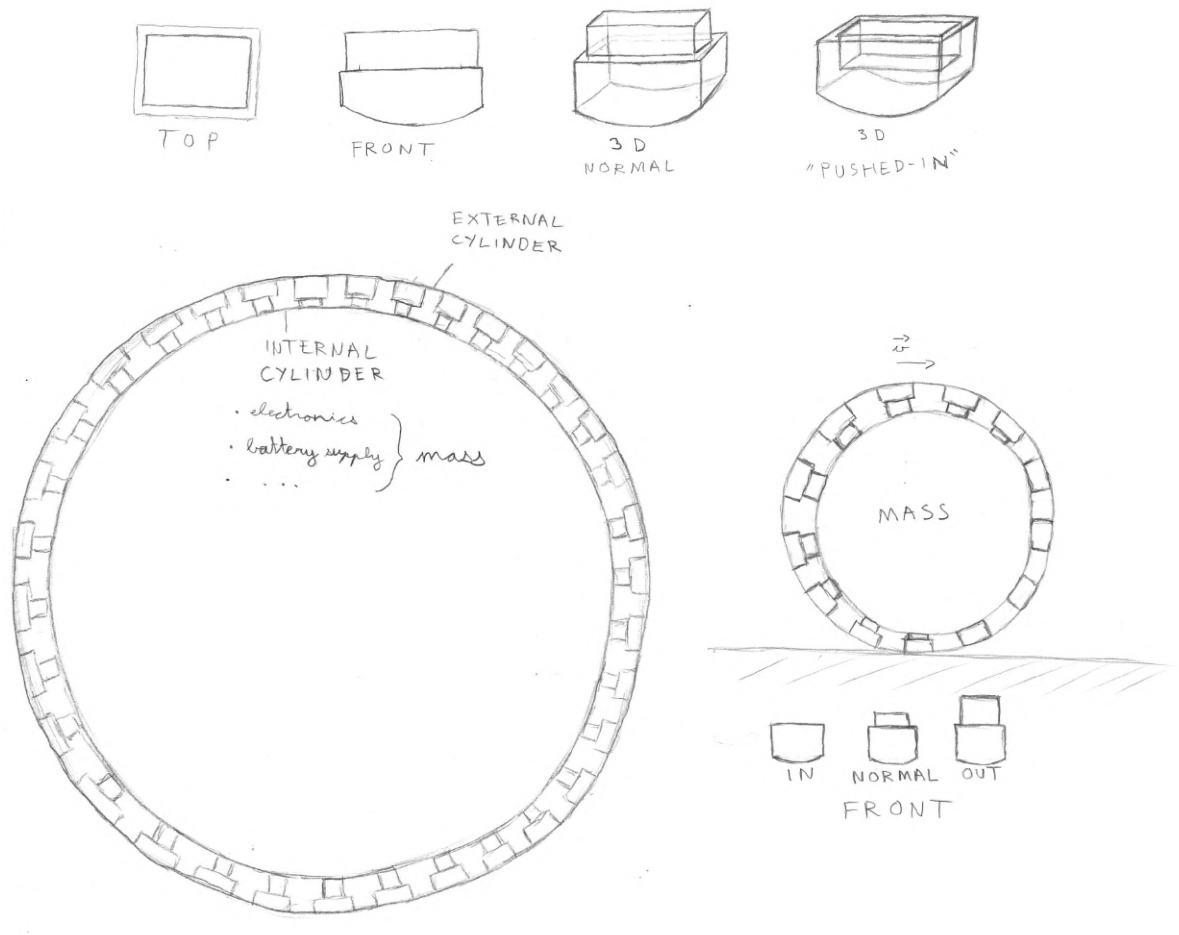


Figure 4.8: Approach with all the "short pistons" around of the external cylinder.

4.1.6 Electromagnets

One approach that seemed very interesting was to use electromagnets, to move a metallic sphere in the center of the cylinder (scheme on Figure 4.10). To explain quickly what an electromagnet is, it is an object that produces a magnetic field when as electric current runs through it. In addition, electromagnets that suit our this approach (appropriate size and power) can nowadays be bought easily. In a sphere, it is likely that a few dozen magnets would be needed to cover all its internal surface. Yet, since the magnetic field is proportional to $\frac{1}{R^2}$, the mass would have a very non-intuitive acceleration, difficult to forecast and to use in the calculation.

Advantages

- This technique allows to move the mass at a maximal distance from the center.
- Acceleration of masses thanks to electromagnets has already been realized. It has generally been applied to heavy and fast transport installations like trains, an example is the Maglev trains used in China.[52]
- Could be used in a sphere.

Disadvantages

- Expensive.
- Heavy.
- Difficult to control the movement of the mass, and its acceleration. Everything would lay in an exact synchronization.

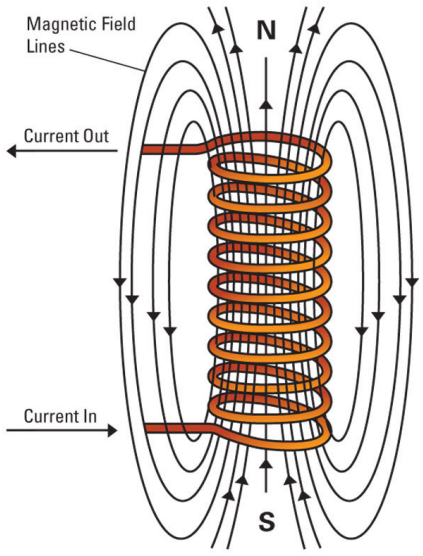


Figure 4.9: Scheme of an electromagnet. [10]

Figure 4.10: An electromagnet attracting metallic beads.

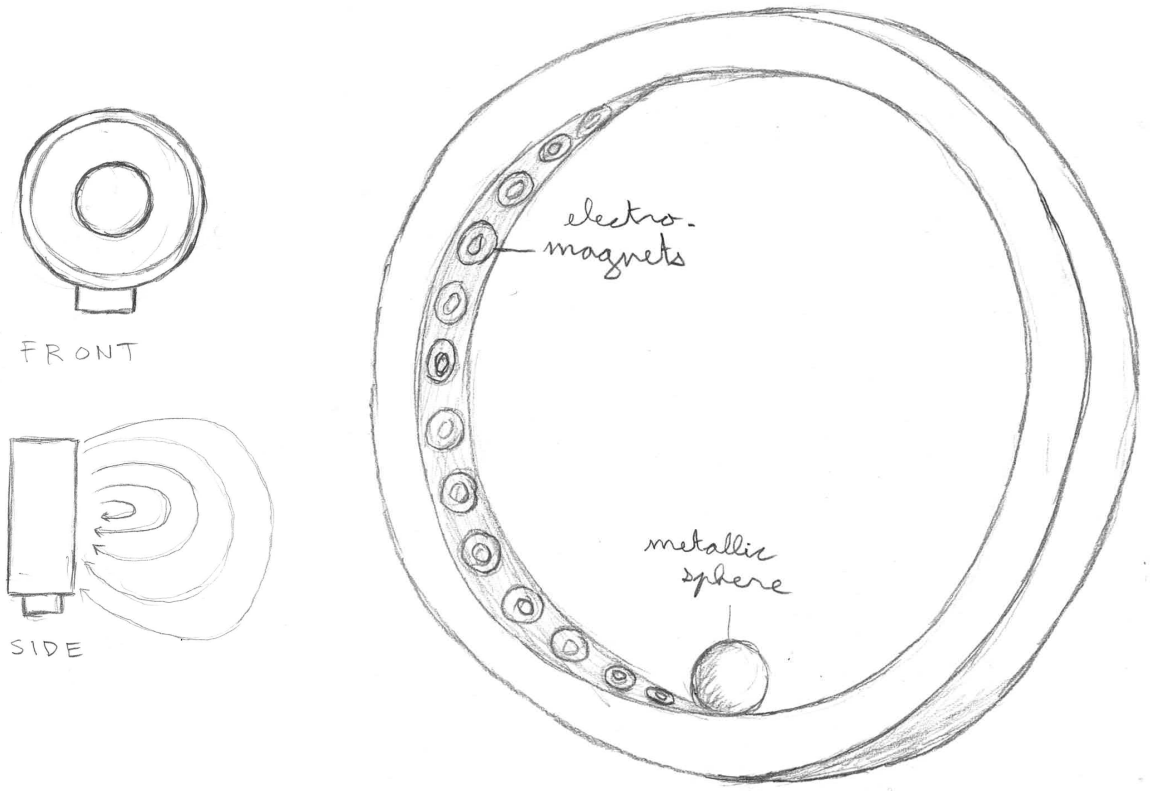


Figure 4.11: Approach with electromagnets on the outside of the cylinder.

4.1.7 Servos on the outside

A servo is a kind of small and light motor that can only move from 0 to 180 degrees. It is controlled through a PWM signal, using a I^2C communication [30]. To use this technology in our project, three servos might be placed around the cylinder at regular intervals, in a symmetrical way. By fixing their arms to the mass in the center using rigid bars, the mass might be moved.

Advantages

- Affordable price.
- Easy control.
- Fairly light.
- Not too voluminous.
- Could be used in a sphere (then 4 servos required).

Disadvantages

- The maximal area that the mass could reach would be equivalent to a circle of the same radius as the length of the servo's arms, which is fairly limiting.
- The servos angles depending on the mass position are difficult to determine mathematically.

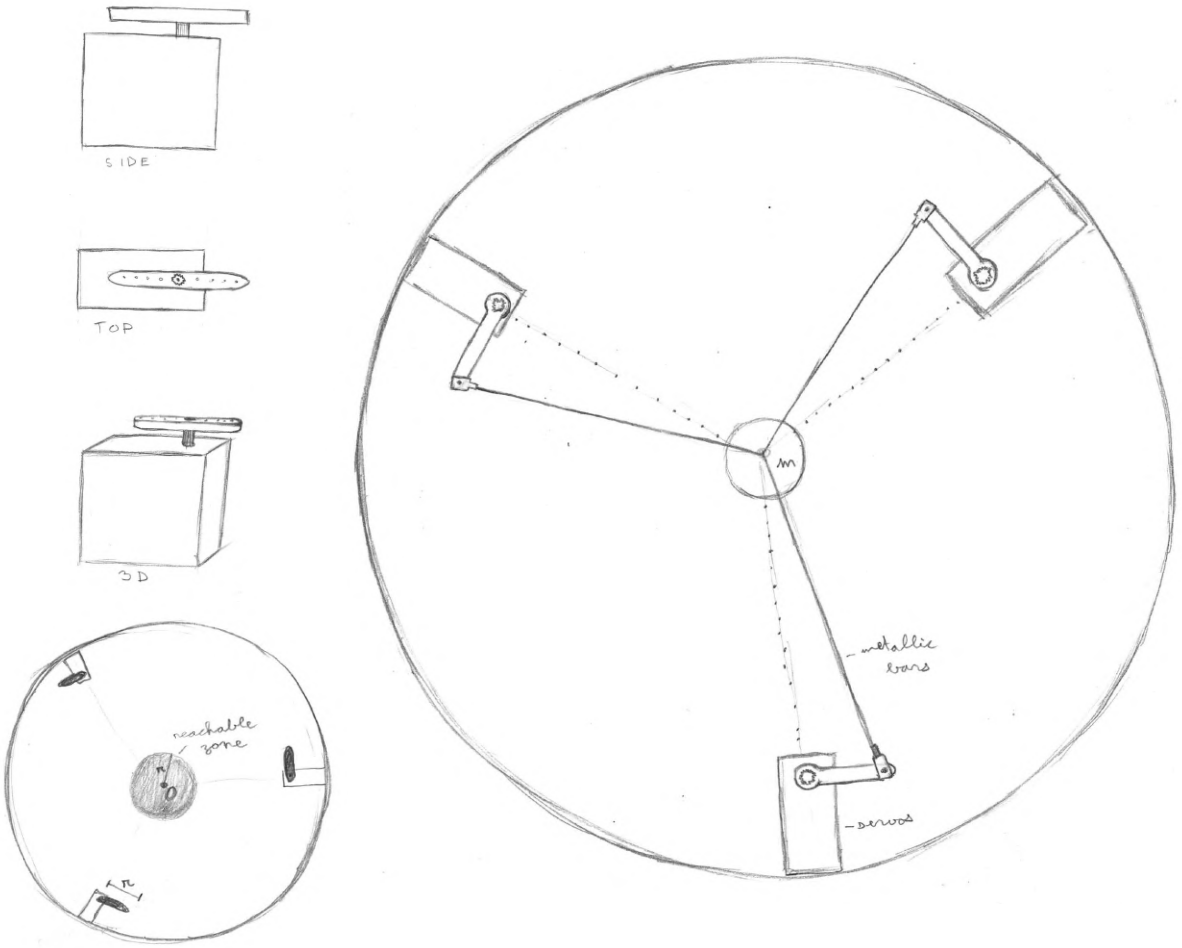


Figure 4.12: Approach with three servos on the outside of the cylinder.

4.1.8 Servos in the center

In this option, three servos would be placed on a support in the middle of the cylinder, for everything else, this approach is similar to Section 4.1.7, but the metallic bars are fixed to the cylinder with a piece that lets them rotate freely.

Advantages

- Using this method would make a smart use of the weight of the servos, which would add to the mass that is moved.
- Could be used in a sphere

Disadvantages

- The bars of the servos do not meet in a point in the center, which creates a new freedom of movement, complicating calculation.

4.2 Chosen approach: servo motors

Figure 4.13: A standard servo motor. [8]

The approach that has finally been chosen is the one using three servos that are equally distributed on the periphery of the cylinder. It has already been explained that servos are a kind of motors that can only complete 180°, back and forth and that they are controlled by a PWM signal. More information can be seen in Figure 4.14, that exposes specifications of servos that are adapted to our project because their power, weight and size are reasonable. In addition, metal geared servos will be used to allow better resistance to wear (instead of plastic gears).

Modulation:	Digital
Torque:	**4.8V:** 130.54 oz-in (9.40 kg-cm) **6.0V:** 152.76 oz-in (11.00 kg-cm)
Speed:	**4.8V:** 0.20 sec/60° **6.0V:** 0.16 sec/60°
Weight:	1.94 oz (55.0 g)
Dimensions:	**Length:** 1.60 in (40.7 mm) **Width:** 0.78 in (19.7 mm) **Height:** 1.69 in (42.9 mm)
Gear Type:	Metal
Pulse Cycle:	1 ms
Typical Price:	8.50 USD

Figure 4.14: Specifications of MG995 servo motor (used in the second cylinder prototype). [17]

In all, this approach was the possibility that seemed the best compromise, including a certain lightness, an affordable price, an easy control and a lot of documentation about the programming easily accessible, since servos are nowadays a wide-spread technology. Yet, the most important point is that with this technique, it is possible to place the mass in the center and so, not to have any influence the movement of the cylinder. In other words, the control of the cylinder can be stopped, which allows it to roll freely. This advantage can lead to spare a lot of power, since the servos have the particularity to consume power constantly, but they use a lot more energy when they are in motion. Also, a disadvantage of these motors is that they consume drastically more when strength is applied to them (see Figure 4.15).

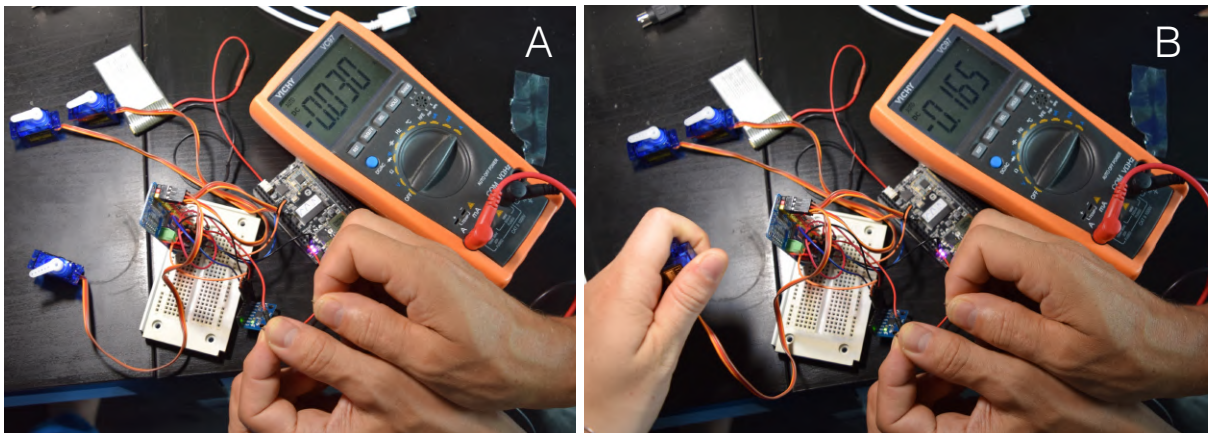


Figure 4.15: Current in [A] that a mini servo consumes if A, its arm is free and B, a force is applied to it. The consumption can be up to five and a half times more in case B.

However, one main point had to be solved: is it really possible, or not, to displace the mass moving simultaneously three servos, without tensions? It can be easily understood that only two of them could be used, since the intersection of two segments is a point, but a third servo allows more stability, relieves the other ones from part of the power they would need to produce and overall, the use of three motors allows to arrange them into a symmetrical plane figure: an equilateral triangle. Indeed, a balanced distribution of mass is critical in our problem to avoid involuntary accelerations of the cylinder. The parallel with the 3D problem, the sphere, can also be made: in that case, four servos could form a tetrahedron.

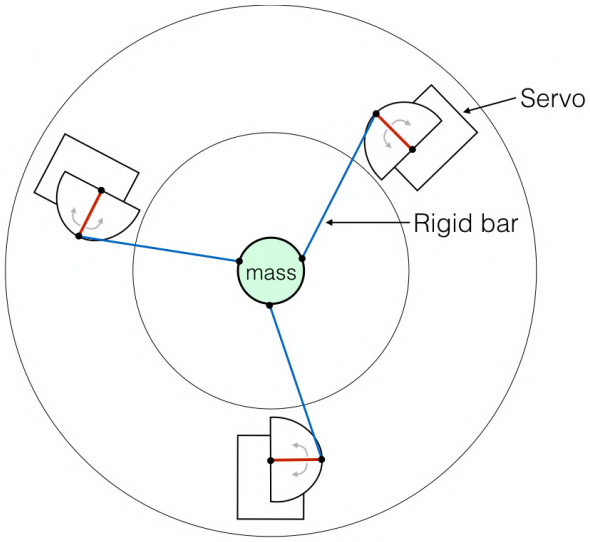


Figure 4.16: Scheme of the chosen approach: using three servo motors.

So, a very easy way has been used to verify if this approach was physically possible. A cardboard, wood and tape model has been approximately realized, which has shown that this idea could be used without any major problem[1].

[1]To see the model working, please watch the following YouTube video: https://youtu.be/4SuPhpwOK5g

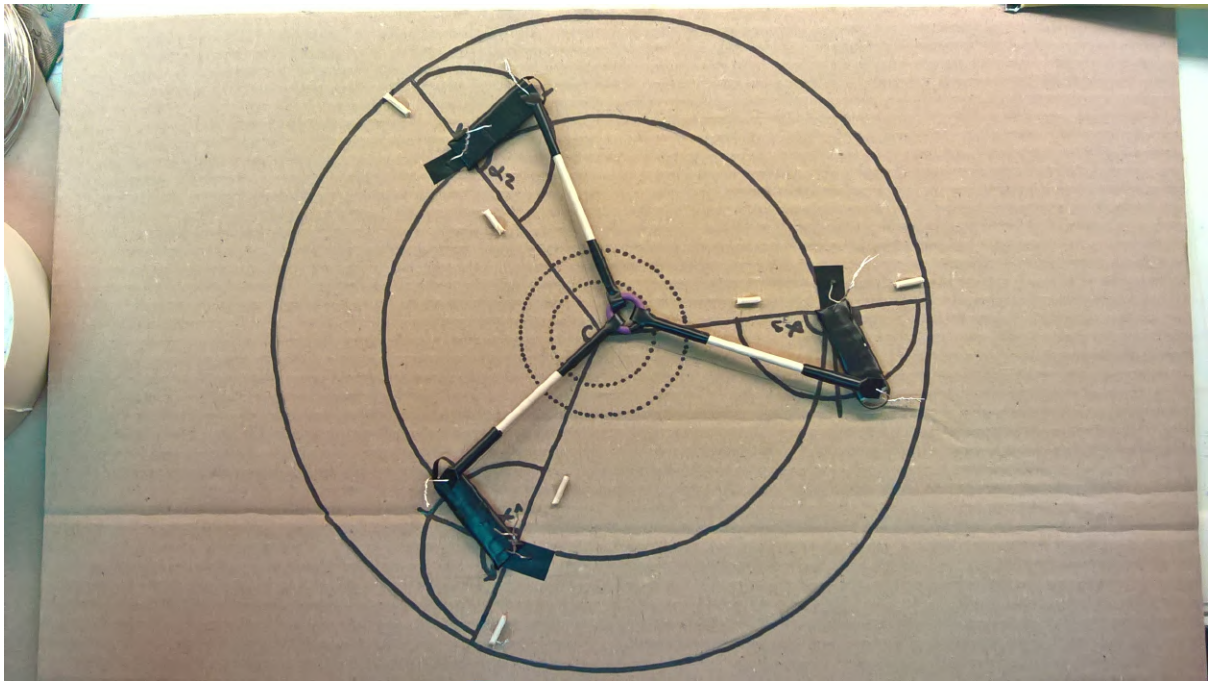


Figure 4.17: The first model, made of cardboard.

In the model, which can be seen on Figure 4.17, the servos are represented by the three small semicircles with their axes the thick black segments. These axes can rotate around the middle of the base of the semicircles, but their movement has been limited to 180 degrees using little pieces of wood glued to the cardboard piece since it is the maximal range we might get using servos. To the end of the servos axes, rigid wooden bars have been fixed, so that they could rotate freely. All these bars have the same length, which is equivalent to the distance between the rotation point of the servos axes and the center C of the cylinder. The bars meet on a little ring (in purple) and are fixed to it so that they may rotate. By placing a pen, or anything else, in the purple ring and by moving it around, the corresponding movement of the servos can be observed. Using this model, it has been demonstrated that it is possible to use three servos to move the mass of our cylinder on a circle.

Chapter 5

Theoretical movement of the mass

Since it has been verified that the principle of using three servos to move the mass is possible (as we have seen in Section 4.2), it is now needed to understand better how the movements of the servos are and how they could be mathematically expressed.
Therefore, the goals of this section are to get used to the function describing the angle of a servo depending on the position of the mass that moves on a circle in the cylinder. For that, the GeoGebra software was used. Then, the mathematical function has been expressed and demonstrated, so that it was possible to use it directly in a JavaScript code.

5.1 Modeling the problem using geoGebra

Since the function of the angles of the servos depending on the position of the mass seemed fairly difficult at a first glance, it has been decided to model the 2D problem and to animate it using the GeoGebra program, which is convenient for the animation of 2D figures it allows[2]. This experimentation allowed to determine many properties of the function.

To begin, the general working of GeoGebra should be explained. This software allows general and easy manipulation of geometrical shapes, such as lines, circles, polygons, and so on. If the user has already created shapes, he can then add points at their intersections, give specific names to the objects he has drawn, add angles and various lines like bisectors or medians. After having drawn a scheme, some parts of it can be animated and everything, that is connected to these, will move as well. The animation of the shapes can be made indirect using *sliders*, then, when the value of the slider is modified, the corresponding magnitude will change as well. In our case, it is an angle, the angle theta formed by the mass moving on a circle, which is animated using the slider of the same name (referring to Figure 5.3, this angle and slider would be in purple).

[2]For more information about this software, you can visit the following webpage: https://www.geogebra.org/. It might be interesting for the reader to download this software and to get the models that are presented in this section (to be downloaded on the GitHub repository in tm/movingMassOnCircle/geogebra/), since experimenting by oneself does always help understanding.

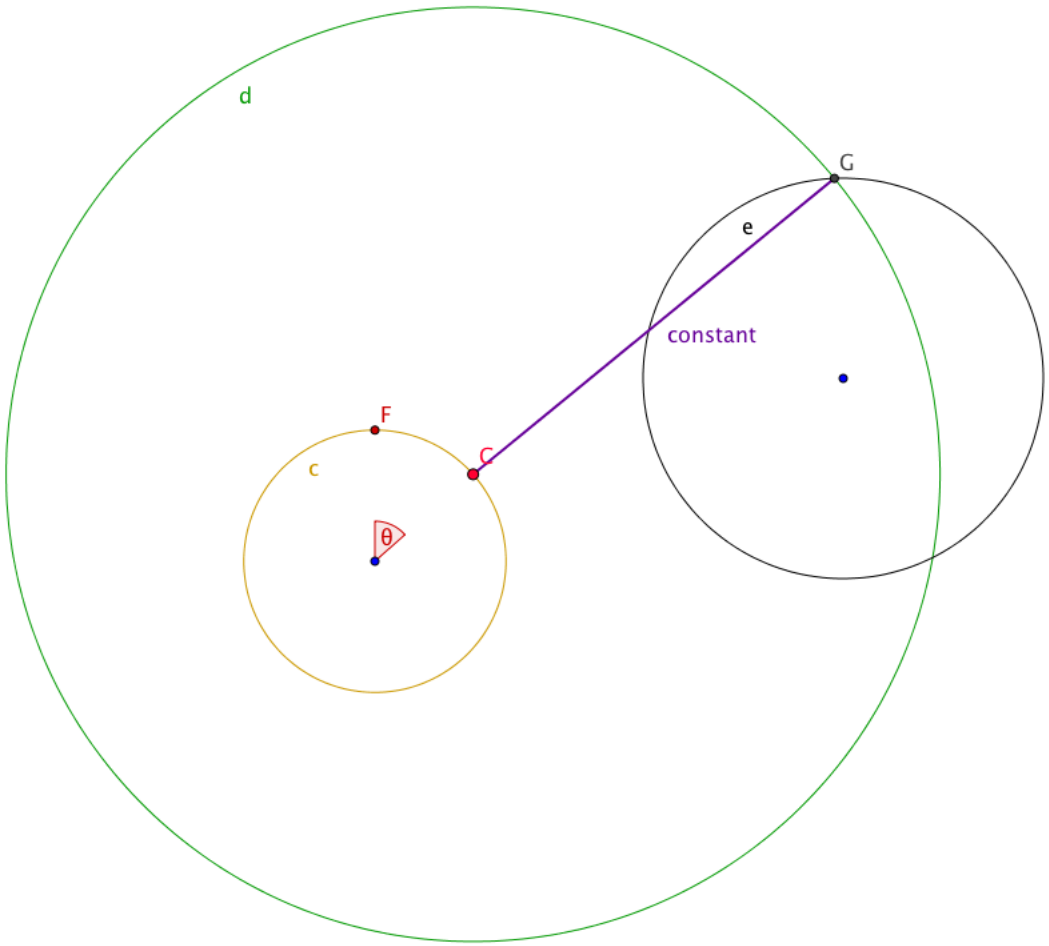


Figure 5.1: A method to built an animated segment of constant length on GeoGebra.

In addition, we needed to define segments with their lengths constant while the figure is animated. In GeoGebra, there is no trivial way to do that, nevertheless, the following step by step solution can be used, the example is illustrated by Figure 5.1.

1. Create a mobile point that can move either on a segment or a circle when its animation is started. Here, the point is the red point C, which can be moved on the circle c. This point is the first end of your segment.

2. Create a circle (here d) of center C and radius equivalent to the length of the constant segment you want to obtain.

3. Draw another shape on which the other end of your segment will move, in our case, the circle e. Circles d and e must intersect, whatever the position of C is.

4. Define a new point (G), as one of the two intersections of d and e.

5. Draw your segment by linking points C and G. It is here the purple segment.

6. Hide c to leave only the segment visible. Then, when C will be moved on c, the segment CG will keep its length constant.

The same method was used on our model to make the three green segments of Figure 5.3. In this model, the red points (the ends of the servos arms) and the purple point (the

mass) can move freely on the black curves (the semicircles or the circles with smaller radii). The element that is animated is the angle θ formed between the vertical, the point O and the mass. Everything else move depending on the mass movement.

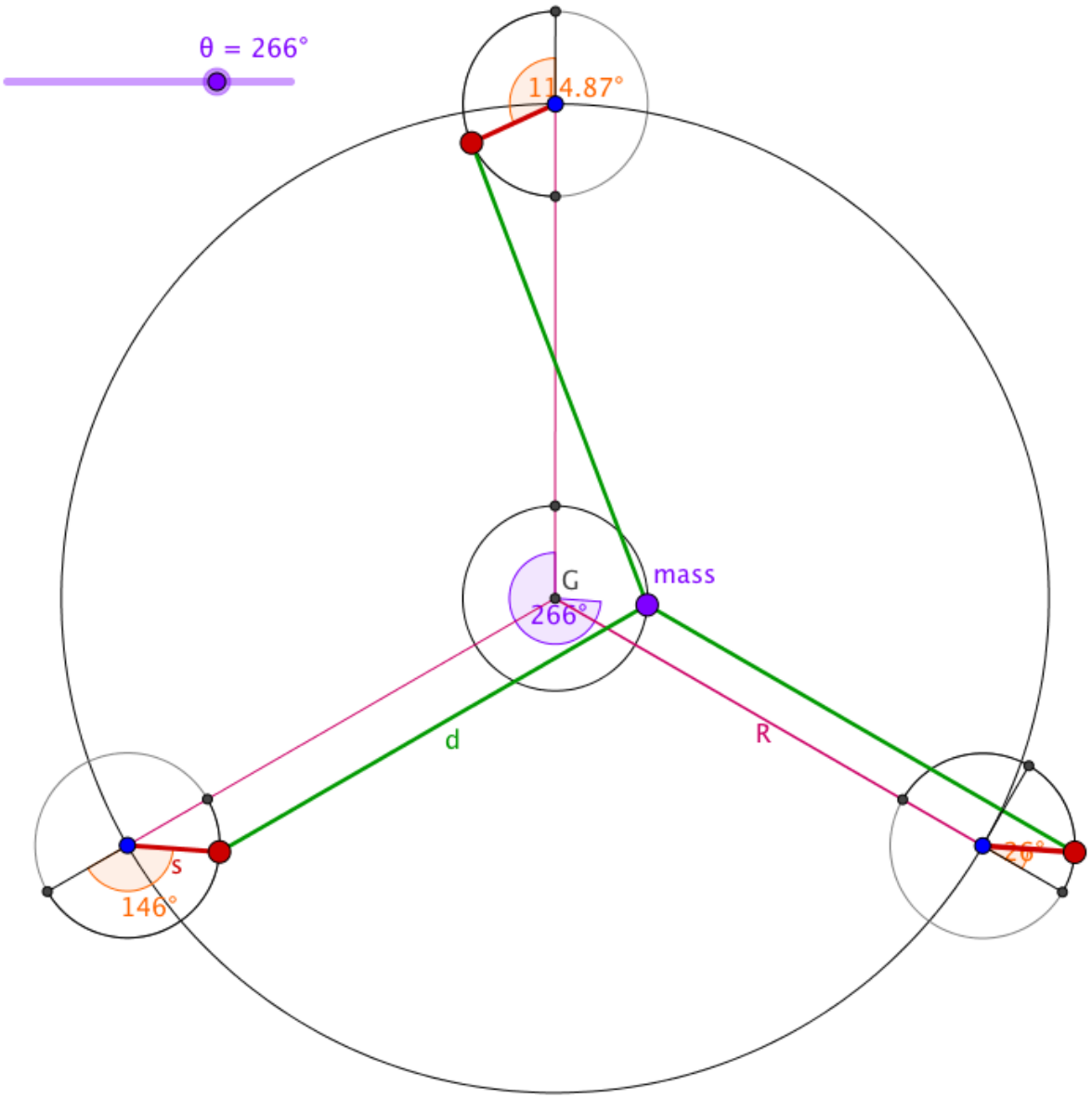


Figure 5.2: GeoGebra model of the problem using the parameters of cylinderPrototype2, representing the case where $s = r$ and $d = R$.

Figure 5.2represents a case with two particularities: first, the radius (r) of the circle on which the mass moves is equivalent to the one of the circle defined by the end of the servo arms (s). The second specificity is that the radius between the servos axes (blue points) and the center of the cylinder (G), called R, is equivalent to the length defined by the mass and the servo arms end (red points), which is d. These characteristics make this case very interesting since it is extremely symmetrical: for two of the servos, d is parallel to R. In addition, this case is also really important because it is the one with the maximal value of r for that radius s. Yet, r is the value that we want to optimize, because the acceleration of the whole cylinder is influenced by the distance between the mass and the center of the cylinder. Thus, the bigger r is, the better will our control of the cylinder

be. We pretend that Figure 5.2 shows the maximal value of $\frac{r}{s}$, as the animation of the model would not work with any bigger value. Indeed, this limitation is very logical: it is just needed to imagine the case when the angle of one servo is equal to 180 degrees, it is then impossible to move the mass any further in the direction of that servo. If this last point has been understood, we can continue to consider the situation in which the angle of one servo is maximal and we can easily deduce that if d was either bigger or smaller than R, the mass could never reach the center of the cylinder. Therefore, we can formulate Assertions 1 and 2, which are based on experiments done with the animated GeoGebra models of the cylinder.

Assertion 1 *It is only possible for the mass to reach all the points on a circle if $d = R$.*

Assertion 2 *For every cylinder with $d = R$ and given servos arms length, there is a maximal radius of the circle on which the mass can move, which radius is equivalent to the servos arm length.*

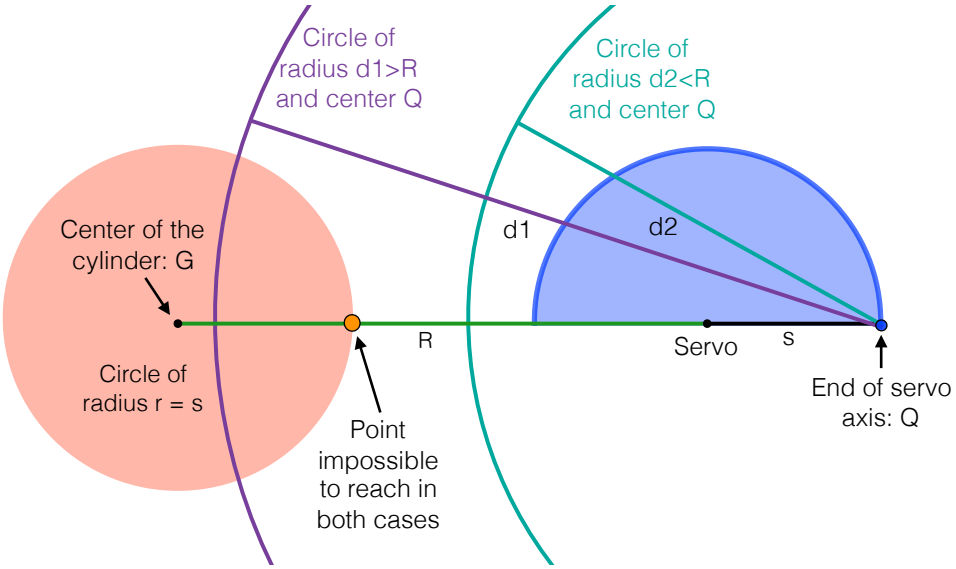


Figure 5.3: One servo and the maximal circle that can be reached by m. Illustrating why Assertion 1 is true: if d is either bigger or smaller than R, some points of the red circle can not be reached.

Assertions 1 and 2 are highly useful, since they give indications on the optimal way to realize the cylinder hardware. Still, one last ratio needs to be studied, which is $\frac{s}{R}$ in the case when $r = s$. That value is not at all as trivial to imagine as the others, since the hole mechanics, which means the three servos, need to be imagined in motion. In this situation, the GeoGebra model has been really useful, since it allowed to find that the maximal limit of this ratio is when $R = s$, this case is represented in Figure 5.4. However, the green segments do then need to have an infinite speed at some point of their movement (see Figure 5.8), yet, this is impossible with a physical hardware. So, it is needed to find the limit of $\frac{s}{R}$ that will be possible to reach with the hardware and especially the maximal servo speed. This is partly what was studied in Section 5.2. This observation allows to formulate another assertion.

Assertion 3 *The bigger the $\frac{s}{R}$ ratio, the faster the servos arms must move at some point of their movement.*

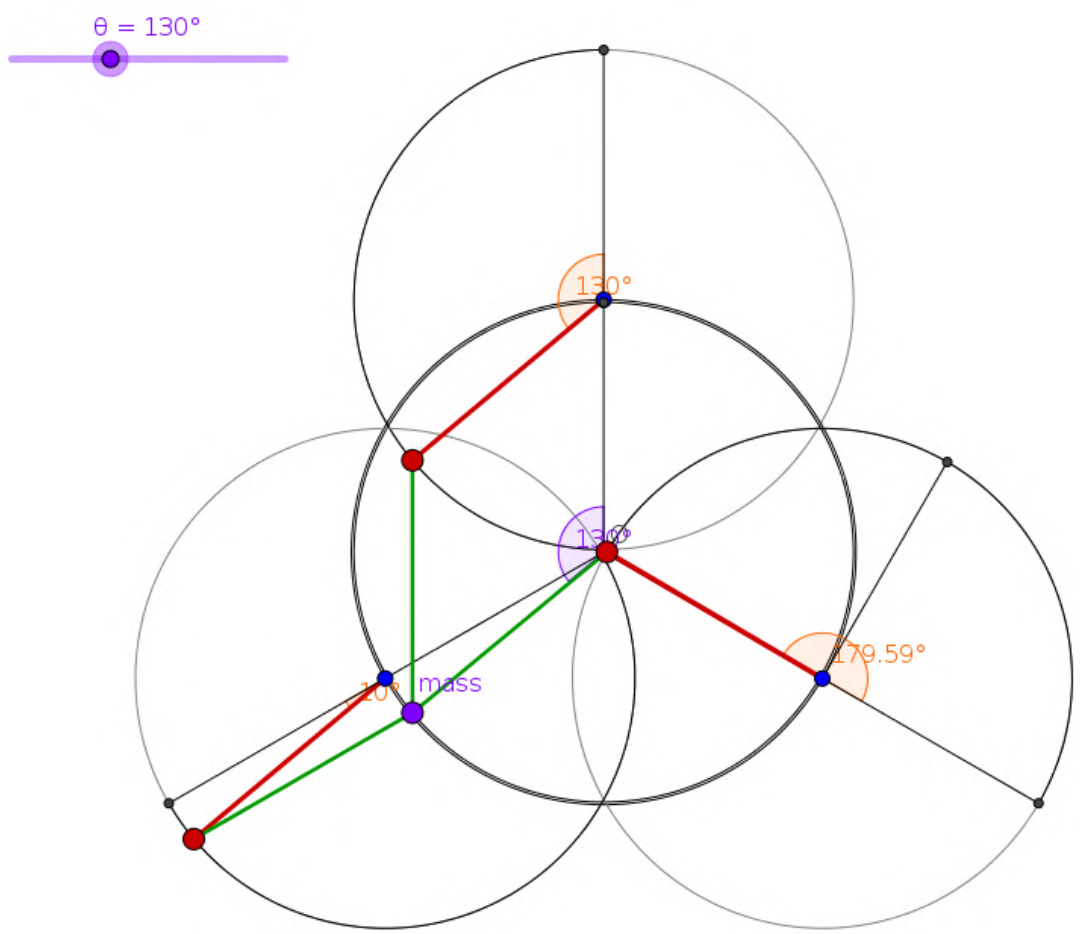


Figure 5.4: Maximal $\frac{s}{R}$: when $R = r$. Also, r is slightly smaller than s, so that all the values of theta could be represented in the animated model. Indeed, otherwise, GeoGebra would not be able to draw some of the red and green segments.

5.2 Analysis of the data exported from GeoGebra

First of all, we wanted to visualize what the function looks like when the dimensions correspond to the ones of one of the prototypes we developed. On this first chart, Figure 5.5, it can be seen that the function is oscillating, and that it repeats every 2π [rad]. Also, it is asymmetrical: the increasing phase is not similar to the decreasing one. Indeed, the increasing part of the graph is linear, which signifies that the servo has then a constant speed[1]. In contrast, when the angle is decreasing, we observe that the speed of the servo is first low and it then then increases abruptly (these observations are based on the variation of the slope). One final conclusion based on this first graph is that the functions corresponding to the angles of the three different servos angle $alpha1$, $alpha2$ and $alpha3$ are exactly the same function, but translated of 120°. So, in the code that was developed, only one function could be used every time, but it was evaluated in $theta + 120$ for the second servo angle and in $theta + 240$ for the third angle. So, if we call $formula$ the function that we want to find, we can get, in degrees:

$$alpha1 = formula(theta)$$
$$alpha2 = formula(theta + 120)$$
$$alpha3 = formula(theta + 240)$$

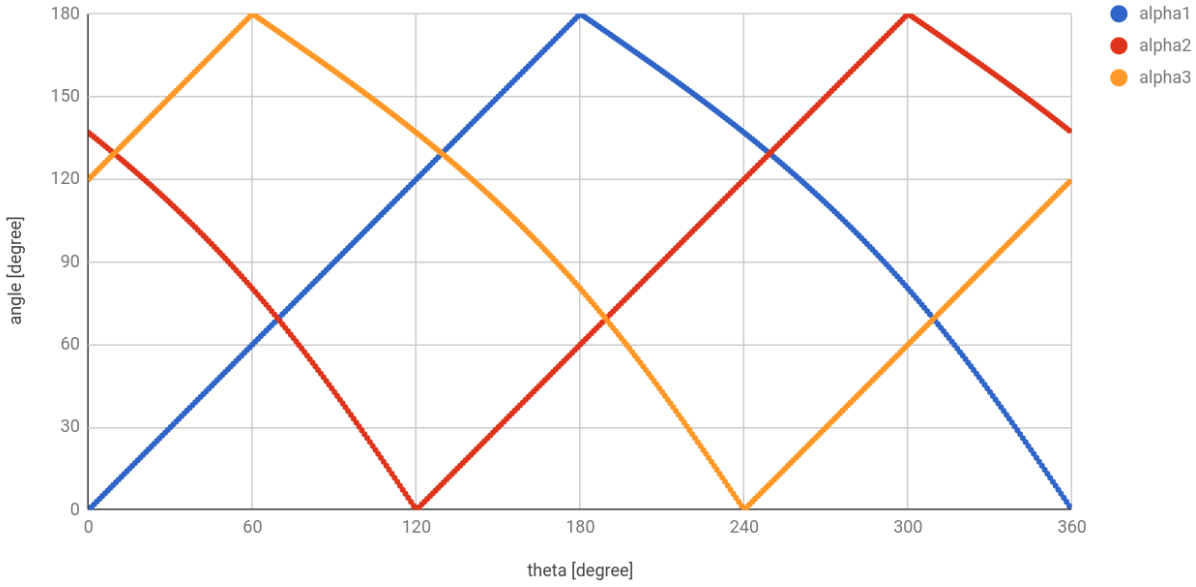


Figure 5.5: Values corresponding to the second cylinder prototype we developed. $R = 150$ [mm], $d = 109.2$ [mm], $s = 16.2$ [mm]. In addition, $r = s$.

Next, we wanted to vary the three parameters we already talked about in Section 5.1: R and s and r. Indeed, it was noticed that the ratio between these parameters had some limitations. Therefore, we wanted to understand how the function evolves depending on the dimensions of the cylinder. Figure 5.6 represents a first variation of these parameters. On it, we can first see the consequences of a smaller r: the function is flatter, as r is smaller. So, the servos do not make as important movements as with a bigger radius, which is logical. This leads us to formulate a following assertion:

[1]The servo angle is equal to zero degrees when it is pointing towards the outside of the cylinder. It is equal to 180 degrees when it points towards the center of the cylinder. c.f. Figure 5.2

Assertion 4 *The smaller the mass circle radius (r), the smaller the variations of the servos angles ($\Delta\alpha$).*

Otherwise, the influence of the variation of d on the function is subtler, indeed, the increasing phase is the same independently on this parameter. However, its variation does affect the decreasing interval, that is more curved when d is small. Yet, the curve that is formed by the function is directly linked to the angular speed of the servos. This speed is simply the slope of the function's plot. Consequently, the maximal angular speed the servo increases with the ratio $\frac{s}{R}$. Which can be reformulated as a last assertion:

Assertion 5 *The maximal angular speed of the servos converges to infinity when $r = R$. This assertion is clearly visible on Figure 5.8.*

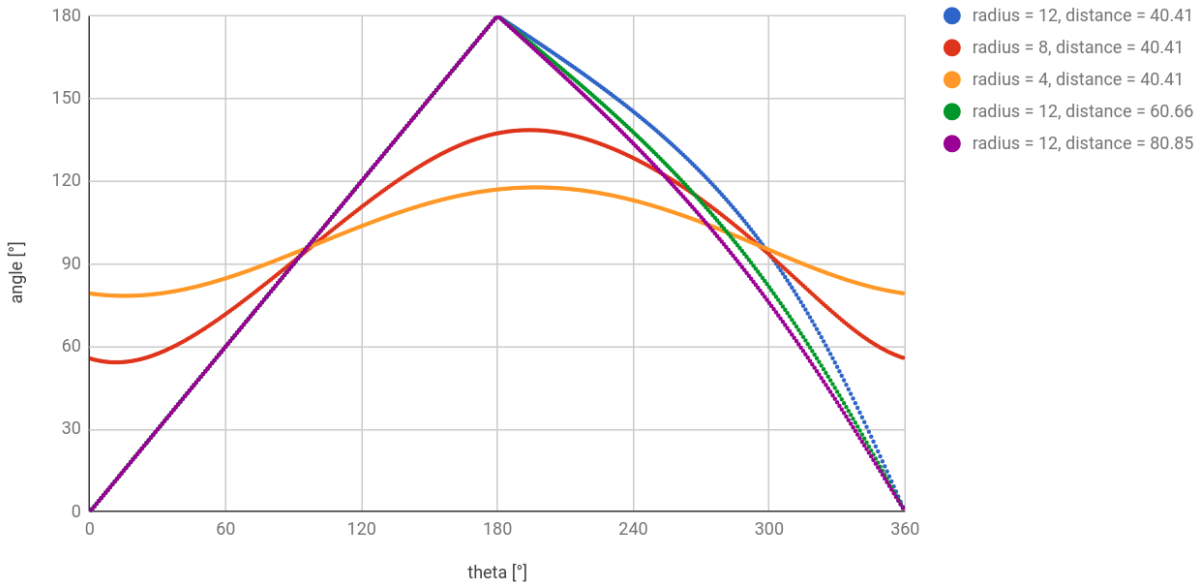


Figure 5.6: Values of alpha obtained when R and r are varied.

In Figure 5.7, a more detailed plot of the r variation is showed.

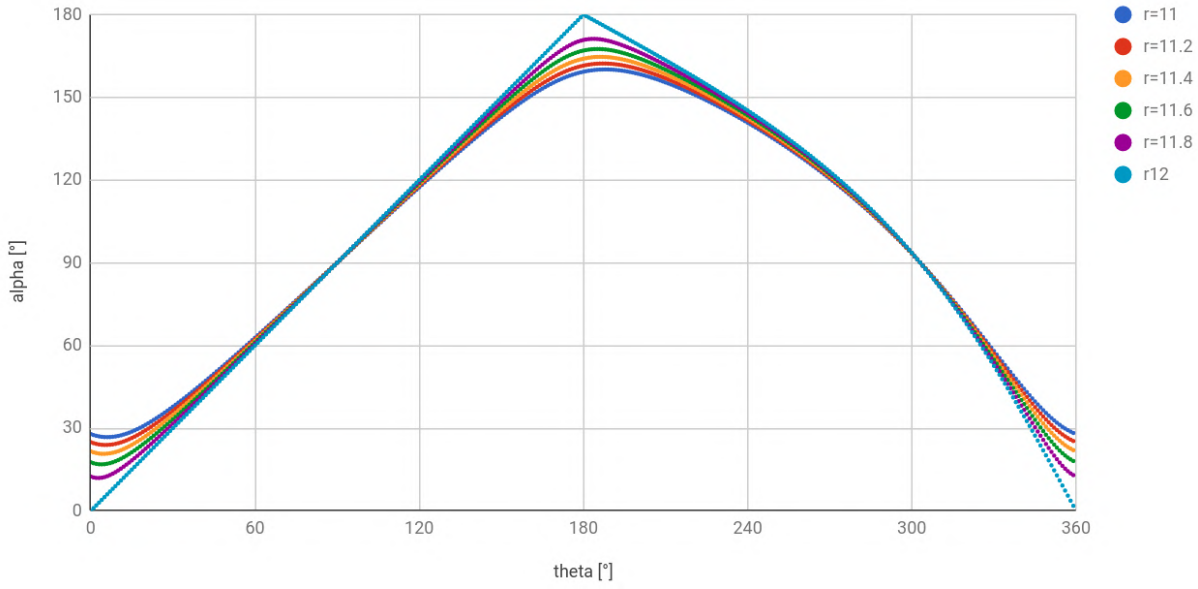


Figure 5.7: Values obtained when R is constant and r is varied.

The plot represented in Figure 5.8 shows that there is a limit of the ratio $\frac{s}{R}$ that we should not cross when building the prototypes. Indeed, if this ratio is too big, it would not be possible to move the mass on a circle, because the speed of the servos should be superior to their maximal possible speed.

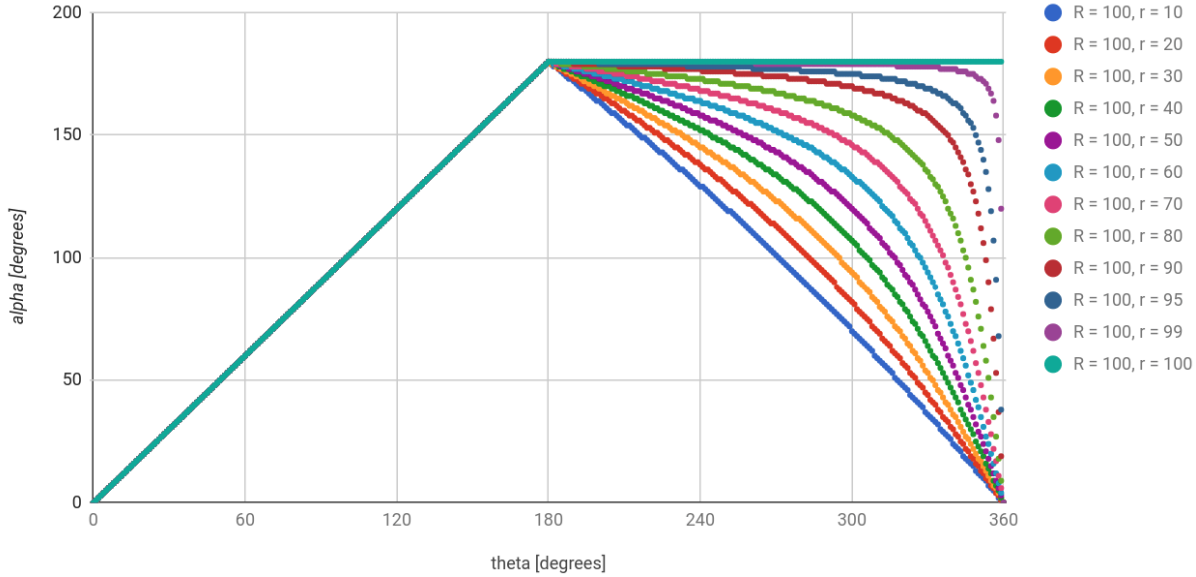


Figure 5.8: Values obtained when s is constant and R is varied. Also, $r = s$. We can see that when $r = R$, the servos maximal speed is infinite.

5.3 Mathematical expression of mass position

Goal: Express the angles of the three servos depending on constants and the position of the mass as a mathematical function.

Based on Figure 5.9 we shall first explain the labels given to the different important points, segments, angles and radii frequently used in this chapter.
Three servos are used on the outside of our main cylinder, they all point towards the center and the distance between the center of the cylinder and the the servo's axis is the same for the three of them. We call this distance R. The servos are positioned to form an equilateral triangle (because it is a symmetrical shape).
$\alpha 1$, $\alpha 2$ and $\alpha 3$ are the angles of the three servos. Because of the mechanics of the servos, these angles are limited between 0 and 180°.
The distance between the ends of the arms of the servos ($Q1$, $Q2$ and $Q3$) and the center of the cylinder is the same for the three servos. This is the distance d.
Finally, the length of the arms of the servos are called s (they all have the same length) and the radius of the circle in the center is r.
The position of m, the mass in the center, is described by the angle that is formed between R, the center G of the cylinder and the position of m. This is the angle θ.
When a problem is solved for only one of the servos to be then applied to all of them, a general name is given for the angle of the servo (α) and for the end of its arm ($Q(a, b)$).

Name	Definition
G	Center of the cylinder
c	Circle of given radius and with center G.
m	Mass that needs to be moved on the circle c.
R	Distance between the servo's axis and G.
d	Distance between the end of the servos arm and m.
s	Length of the servos arms.
r	Radius of the circle c.
P	Position of the mass.
x, y	Coordinates of the position of the mass.
θ	Angle between R, G and P.
$Q1$, $Q2$, $Q3$	Points corresponding to the end of the arms of the three servos.
a, b	Coordinates of Q.
$\alpha 1$, $\alpha 2$, $\alpha 3$	Angles of the three servos, they can be between 0 and 180 °.
α and $Q(a, b)$	General appellations of $\alpha 1$, $\alpha 2$, $\alpha 3$ and $Q1$, $Q2$, $Q3$.

Table 5.1: Summary of the objects used in further calculation

To solve this problem, we first try to find the expression of a depending on constants x, y, d, R and s, because it is then easy to determine α using a cosine function. To begin, it is needed to have a scheme of the problem. The Figure 5.9 illustrates the situation.

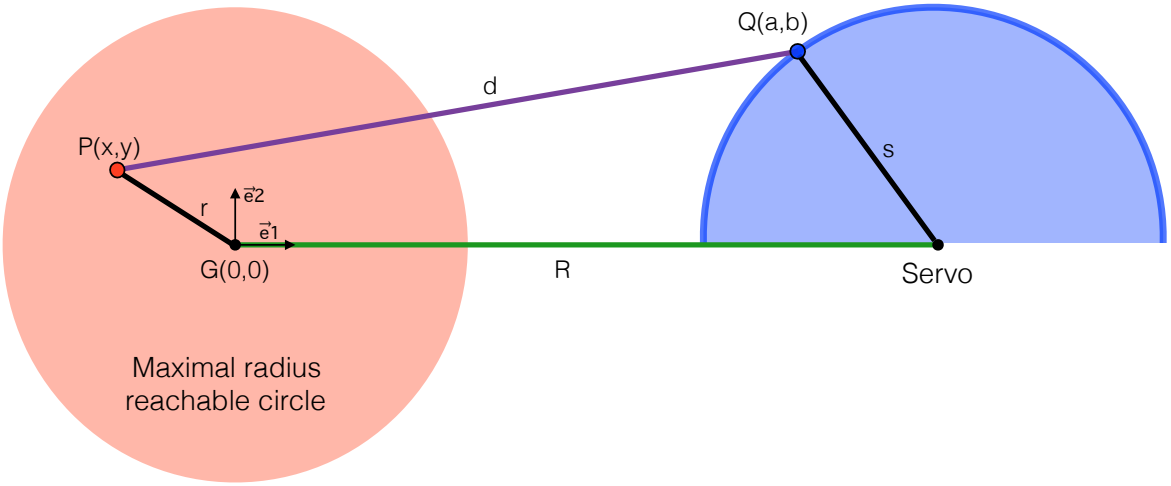


Figure 5.9: The mass that is moved and one servo: The red point corresponds to the mass in the cylinder (m), the light red area represents all the possible positions of m, the blue point is the end of the servo's arm.

Then, according to the vectors properties, if $\vec{d} = (a-x)\vec{e1} + (b-y)\vec{e2}$, its magnitude can be defined as:

$$\vec{d} = \begin{pmatrix} a-x \\ b-y \end{pmatrix} \rightarrow ||\vec{d}|| = \sqrt{(a-x)^2 + (b-y)^2} \tag{5.1}$$

On the other hand, using the Pythagorean theorem, we can express b as the following.

$$b = \sqrt{s^2 - (R-a)^2} \tag{5.2}$$

Thus,

$$d = \sqrt{(a-x)^2 + (\sqrt{s^2 - (R-a)^2} - y)^2} \tag{5.3}$$

This is the expression that is needed to be solved for a. To begin, both sides should be squared to obtain:

$$d^2 = (a-x)^2 + (\sqrt{s^2 - (R-a)^2} - y)^2 \tag{5.4}$$

We next develop and simplify the previous equation.

$$d^2 = a^2 - 2ax + x^2 + s^2 - (R-a)^2 - 2y\sqrt{s^2 - (R-a)^2} + y^2 \tag{5.5}$$

Next, the square root is isolated on one side and all the other terms are brought on the other.

$$2y\sqrt{s^2 - (R-a)^2} = x^2 + y^2 + s^2 - d^2 - R^2 + 2a(R-x) \tag{5.6}$$

At this point, all the terms that do not contain a are gathered into a constant called Δ.

$$\Delta = x^2 + y^2 + s^2 - d^2 - R^2 \tag{5.7}$$

Therefore, Equation 5.6 becomes:

$$2y\sqrt{s^2-(R-a)^2} = \Delta + 2a(R-x) \tag{5.8}$$

We continue by squaring both sides again:

$$4y^2(s^2-(R-a)^2) = (\Delta + 2a(R-x))^2 \tag{5.9}$$

Developing the left part of Equation 5.9, we have the following:

$$\begin{aligned} &4y^2(s^2-(R-a)^2) \\ &= 4y^2(s^2-R^2+2aR-a^2) \\ &= 4y^2(s^2-R^2)+8y^2aR-4y^2a^2 \end{aligned} \tag{5.10}$$

Proceeding similarly for the right part of Equation 5.9, we get:

$$(\Delta + 2a(R-x))^2 = \Delta^2 + 4\Delta a(R-x) + 4a^2(R-x)^2 \tag{5.11}$$

Inserting these simplifications back into Equation 5.9 and ordering the terms, it can be seen that the equation is a second degree polynomial.

$$-4y^2(s^2-R^2) - 8y^2aR + 4y^2a^2 + \Delta^2 + 4\Delta a(R-x) + 4a^2(R-x)^2 = 0 \tag{5.12}$$

Finally, this is the polynomial obtained after factorization.

$$4a^2[y^2+(R-x)^2] + 4a[-2y^2R + (R-x)\Delta] + \Delta^2 - 4y^2(s^2-R^2) = 0 \tag{5.13}$$

The following formula is considered for the resolution of a second degree polynomial.

$$na^2 + oa + p = 0 \rightarrow a = \frac{-o \pm \sqrt{o^2-4np}}{2n} \tag{5.14}$$

In the case of Equation 5.13, these are the values of n,o and p.

$$\begin{aligned} n &= 4[y^2+(R-x)^2] \\ o &= 4[-2y^2R + (R-x)\Delta] \\ p &= \Delta^2 - 4y^2(s^2-R^2) \end{aligned} \tag{5.15}$$

Therefore, the equation can be solved for a.

$$\begin{aligned} a &= \frac{-o \pm \sqrt{o^2-4np}}{2n} \\ &= \frac{-4[-2y^2R + (R-x)\Delta]}{2\big(4[y^2+(R-x)^2]\big)} \\ &\pm \frac{\Big[\big(4[-2y^2R + (R-x)\Delta]\big)^2 - 4\big(4[y^2+(R-x)^2]\big)\big(\Delta^2 - 4y^2(s^2-R^2)\big)\Big]^{\frac{1}{2}}}{2\big(4[y^2+(R-x)^2]\big)} \end{aligned} \tag{5.16}$$

To develop Equation 5.16, we will first work on the terms of the square root.

$$\begin{aligned}
&\big(4[-2y^2R + (R-x)\Delta]\big)^2 - 4\big(4[y^2 + (R-x)^2]\big)\big(\Delta^2 - 4y^2(s^2 - R^2)\big) \\
&= 16[4y^4R^2 - 4y^2R(R-x)\Delta + (R-x)^2\Delta^2] \\
&- 16[y^2\Delta^2 - 4y^4(s^2 - R^2) + (R-x)^2\Delta^2 - 4y^2(s^2 - R^2)(R-x)^2] \\
&= 16[4y^4R^2 - 4y^2R(R-x)\Delta - y^2\Delta^2 \\
&\quad + 4y^4(s^2 - R^2) + 4y^2(s^2 - R^2)(R-x)^2] \\
&= 16y^2[4\big(y^2R^2 - R(R-x)\Delta + y^2(s^2 - R^2) + (s^2 - R^2)(R-x)^2\big) - \Delta^2]
\end{aligned} \tag{5.17}$$

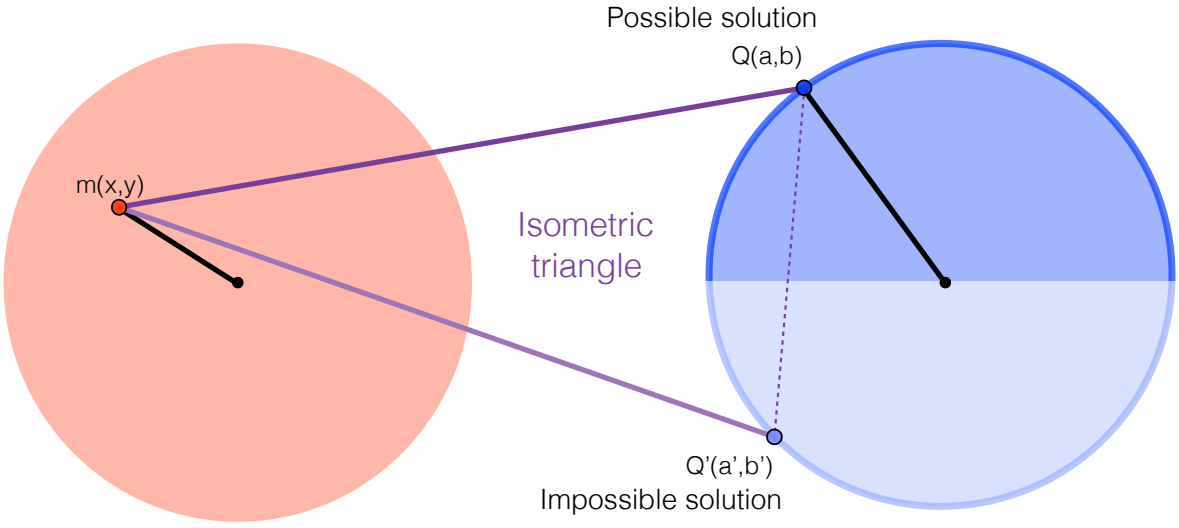
If we put Equation 5.17 back under the square root, we get:

$$\begin{aligned}
&\Big[16y^2[4\big(y^2R^2 - R(R-x)\Delta + y^2(s^2 - R^2) + (s^2 - R^2)(R-x)^2\big) - \Delta^2]\Big]^{\frac{1}{2}} \\
&= 4y\Big[4\big(y^2R^2 - R(R-x)\Delta + y^2(s^2 - R^2) + (s^2 - R^2)(R-x)^2\big) - \Delta^2\Big]^{\frac{1}{2}}
\end{aligned} \tag{5.18}$$

Finally, the square root can be placed in the equation for a again and the coefficient 4 can be simplified, which leads to the following result for a.

$$\begin{aligned}
a = f(x,y,R,s,d) &= \frac{2y^2R - (R-x)\Delta}{2\big(y^2 + (R-x)^2\big)} \\
&\pm \frac{y\Big[4\big(y^2R^2 - R(R-x)\Delta + y^2(s^2 - R^2) + (s^2 - R^2)(R-x)^2\big) - \Delta^2\Big]^{\frac{1}{2}}}{2\big(y^2 + (R-x)^2\big)}
\end{aligned} \tag{5.19}$$

Where $\Delta = x^2 + y^2 + s^2 - d^2 - R^2$.



The equation has at last been solved, nevertheless, it returns two possible values of a for one position of the mass, however in our case, only one of the two is correct, because the servos can only sweep from 0 to 180 degrees. To know which value is correct, we made a deduction based on observations, which is that when $y > 0$, then the correct value of a is the bigger result that Equation 5.19 returns. On the contrary, when $y < 0$, the value of a that must be considered is the smaller one. Examples of the tests we made are visible in Figure 5.10.

Hypothesis 2

$$\begin{aligned} y > 0 &\rightarrow max(f(x, y, R, s, d)) \\ y < 0 &\rightarrow min(f(x, y, R, s, d)) \end{aligned} \tag{5.20}$$

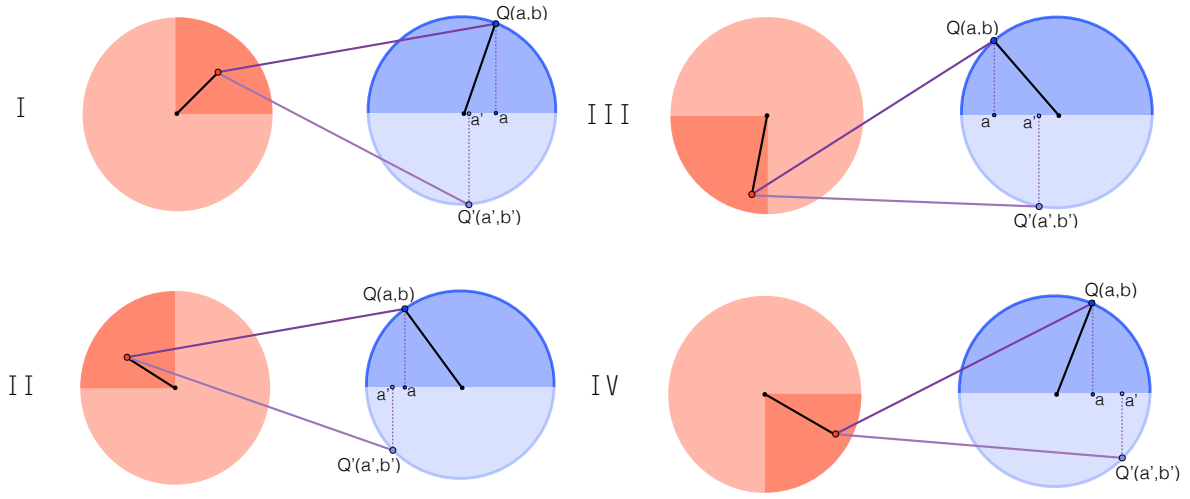


Figure 5.10: Illustration of Hypothesis 2. One position of m has been represented for every quadrant, as well as the corresponding Q and Q'.

5.3.1 From a to α

Finally, we shouldn't forget that the value we need is α. To find it using a, one should base himself on Figure 5.11 to find the following equation:

$$\alpha = \cos^{-1}\left(\frac{a-R}{s}\right) \tag{5.21}$$

This formula can now be used to express any angle α corresponding to an x, y position of the mass.

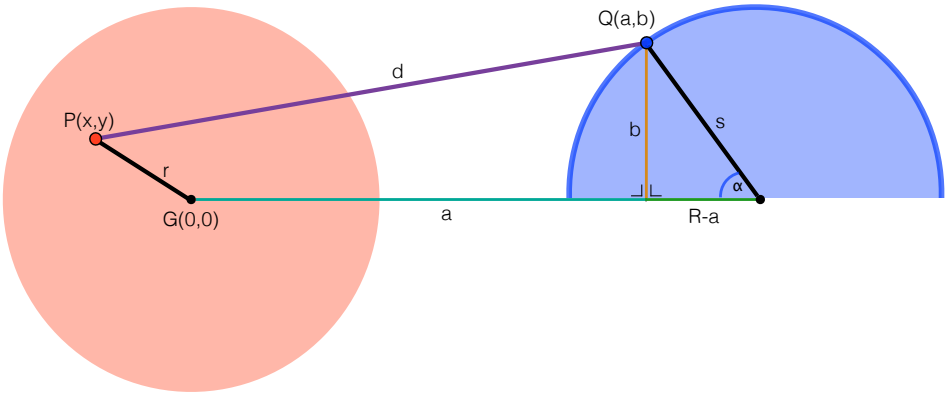


Figure 5.11: To find α, we use the rectangular triangle with sides b, $R-a$ and s.

5.3.2 Servos angles when the mass is in the center

Since one of the most interesting positions for the mass to be is the center of the cylinder (because the torque is then equal to zero and we do not influence the cylinder's movement), we developed a simpler general expression of this case. Considering Figure 5.12 allows to visualize the problem. The triangle with sides dsR is unspecified, as we want to solve the case for the most general situation. The sides of the triangle are known constants and we want to find the value of alpha, which is the same for the three servos, since the case is symmetrical.

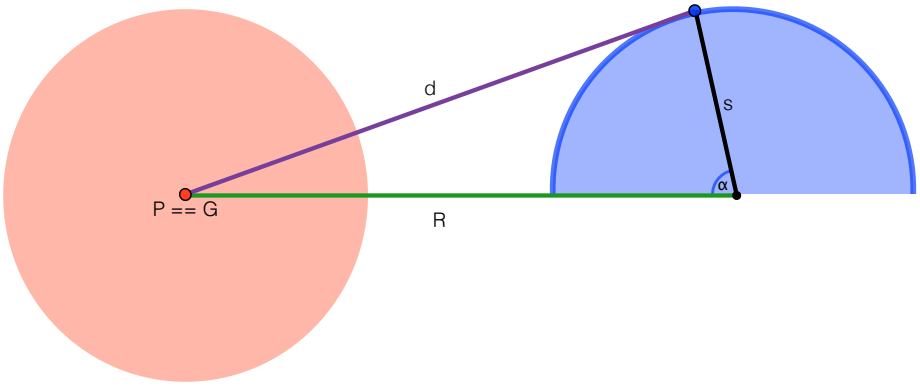


Figure 5.12: Scheme of one servo when the mass is in the center (on point G).

To find the value of α depending on d, s and R, we divide the dsR triangle into two rectangular triangles formed by the height of $\triangle dsR$ that passes through Q. We call that height h, and the two segments cut by h are x and $R - x$. This division is showed at Figure 5.13.

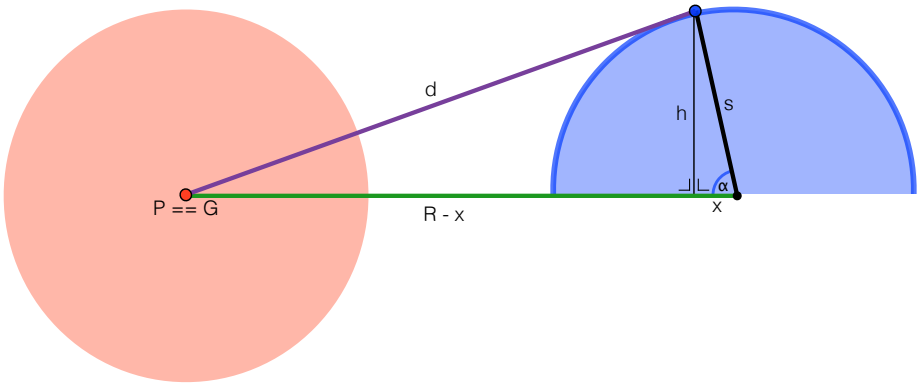


Figure 5.13: The dsR triangle can be divided into two rectangular triangles.

To begin, we can use the cosine function to express α.

$$\cos\alpha = \frac{x}{s} \rightarrow \alpha = \cos^{-1}\left(\frac{x}{s}\right) \tag{5.22}$$

Now, we can apply the Pythagorean theorem to our two rectangular triangles, which will allow us to find x.

$$\begin{aligned} d^2 &= h^2 + (R-x)^2 \\ s^2 &= h^2 + x^2 \end{aligned} \tag{5.23}$$

We then continue by isolating and replacing the unknowns (h and x).

$$\begin{aligned} h^2 &= s^2 - x^2 \\ d^2 &= s^2 - x^2 + R^2 - 2Rx + x^2 \\ &= s^2 + R^2 - 2Rx \end{aligned} \tag{5.24}$$

We finish isolating x.

$$x = \frac{s^2 + R^2 - d^2}{2R} \tag{5.25}$$

To obtain α, we replace x from Equation 5.22 by its expression in Equation 5.25.

$$\alpha = \cos^{-1}\left(\frac{s^2 + R^2 - d^2}{2Rs}\right) \tag{5.26}$$

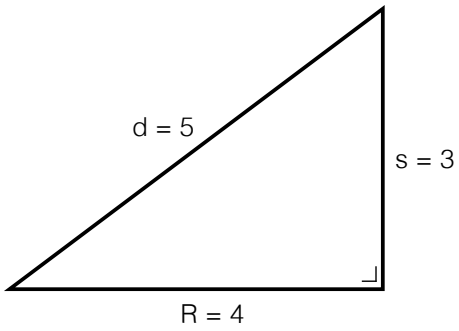


Figure 5.14: Testing Equation 5.26 with a rectangular triangle.

To test this equation, we can use our favorite rectangular triangle: 3-4-5 (refer to Figure 5.14). Using the values of this triangle sets α to 90 degrees. In addition, we can bring the inverse cosine to the other side of the equation, so that we get a cosine again. Replacing d, R and s by the values of the sides of the triangle gives the following result:

$$\cos(90°) \stackrel{?}{=} \frac{9 + 16 - 25}{2 \cdot 3 \cdot 4} \tag{5.27}$$

This statement is true and it allows to convince of the exactitude of Equation 5.26.

Chapter 6

Mechanics

During this work, three different prototypes of cylinders have been realized, yet, what we call cylinderPrototype1 (first prototype, cp1 for short) and cylinderPrototype2 (second prototype, cp2 for short) are very similar. In this section, the differences and properties of these three cylinders will be presented. The accent will be put on the purely hardware aspect of those, this implies explanation of the materials used to build the structure of the cylinder and the methods used to work them. At last, the defects of our prototypes will be presented and solutions suggested to avoid these in hypothetical further versions of the cylinder.
All the machines used to realize the prototypes belong to the Made@UC makerspace[2].

6.1 Prototype 1

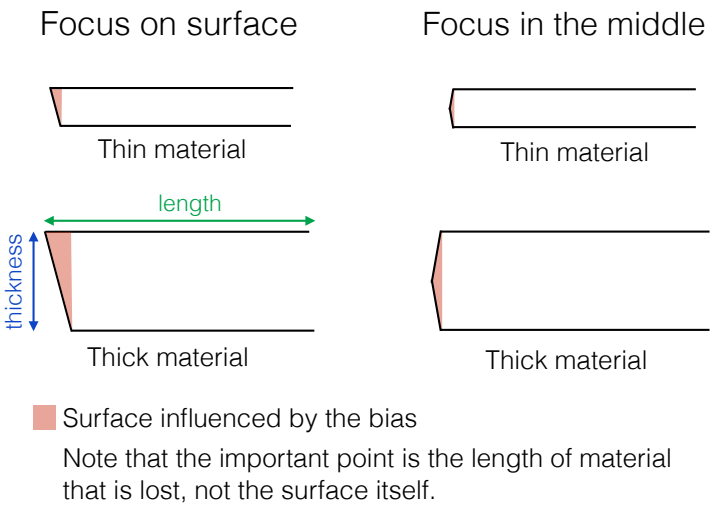


Figure 6.1: Cutting materials with a laser cutter: bias problems.

First of all, the decision to use materials that can be cut using a laser cutter was made, because of the impressive precision and swiftness this technique allows. Indeed, it does only take a few minutes to cut out the cylinder base, which made by hand would take

[2]This place is a workshop situated in Renens which offers the possibility to use a lot of quality tools to anyone. No specific background is needed. For more, visit: https://made.univercite.ch/#espace

hours and a lot of specific skills to be done. The uncertainty of the machine is around a tenth of a millimeter, however, the cut is not perfectly aligned, because the cutter uses a converging lens to focus the laser (to get maximal power), so the cut is always a little biased, yet, this uncertainty can be made reproducible by calibrating the cutter before every use. In our case, the focus has always been made at the surface of our materials, which shouldn't be problematic since they are fairly thin. In the case we had used very thick materials, it would have been better to make the focus in the center of the material, to avoid having a too big difference in the length of both sides of the cut part (refer to Figure 6.1).

Nevertheless, the consequence of the use of the laser cutter is that vectorial files of the object to cut are needed, to realize these, we used the open-source software FreeCad. The exported file corresponding to cp1 can be seen on Figure 6.2.

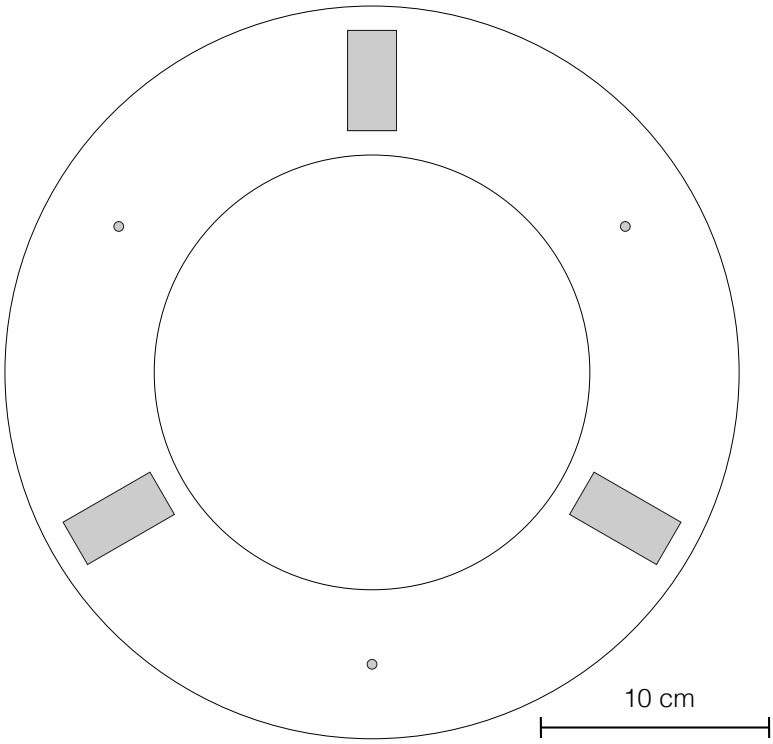


Figure 6.2: FreeCad drawing of cp1 base.

The concept we thought of was to do the cylinder as open as possible to simplify debugging. It consists of two rings connected with three rods of equal length, to make the rings parallel. This approach was the easiest to get a cylinder, but also to fix the servos, which are just screwed on one of the rings. In this prototype, both rings are the same and have a hole in the middle for two reasons: to minimize the mass and again, to make it easier to see what is happening inside. Yet, one question remained crucial, which was the size the cylinder should have, since it had to be big enough to include the servos and make their use effective, but it had to be as small as possible to reduce production costs. The compromise that was found was to make the cylinder's diameter 30 cm long. In this case, the cylinder base was cut out of 6 mm MDF[1], which has the benefit of a low price (14 Fr/m^2), but is fairly heavy. It still suited very well for a first prototype. To connect

[1] "Medium-density fibreboard: a wood-substitute material used in interior decoration [26]."

the two pieces together, we used three sections of about 10 cm of an aluminium tube through which we placed a piece of 4 mm threaded rod approximately 3 cm longer. The extremities of the threaded rod where then passed in the holes of the MDF base and were fixed with bolts. This method allowed us to get a cylinder very easily.

Figure 6.3: Information about cp1.

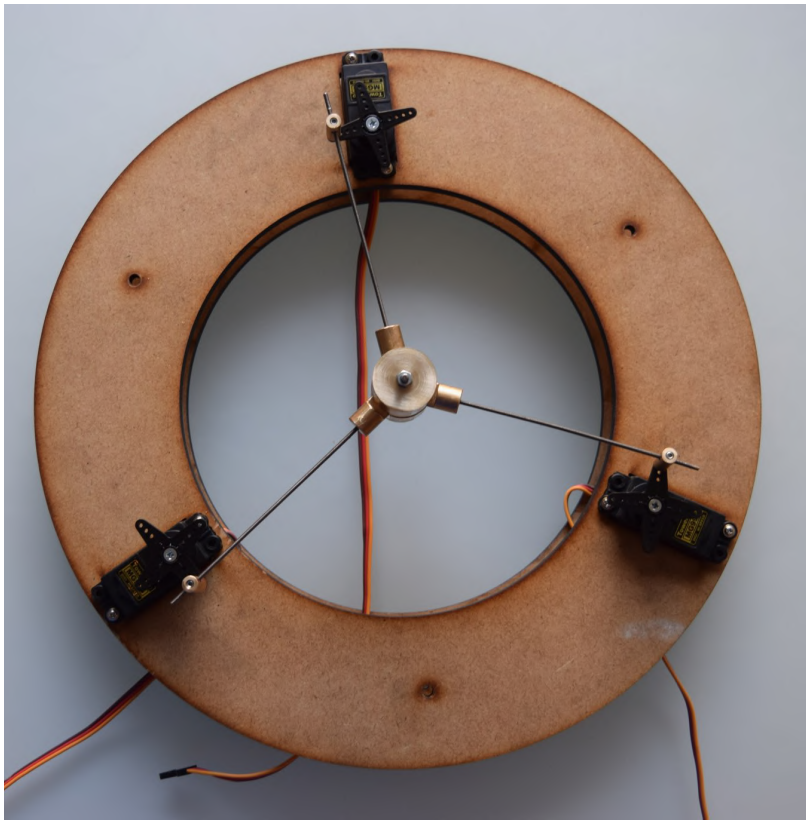
Finally, this prototype was used as a final proof of concept, as it allowed to show that it is possible to move the whole cylinder with the displacement of a mass supported by three servos, in other words, that the ratio between the cylinder's mass and the mass in its center would be sufficient to induce the movement. It is for that reason that elastic band was used (see Figure 6.3): at the time, the mathematical formula of Section 5.3 had not yet been demonstrated and the mass movement could only be really approximative and stochastic.

6.2 Prototype 2

The second prototype of the cylinder that has been developed is very similar to the first one regarding the cylinder itself. Indeed, it uses the same hardware as cp1. Yet another part has been actively enhanced, which is the servo arm / mass connection. In this prototype, what was used is 2 mm thick steel threaded rod that is screwed into metallic brass pieces that allow the connection with the mass (all the pieces are visible on Figure 6.4)[1]. These parts have been manufactured after that an important conclusion

[1]Since the three brass center pieces were also used in the final prototype, more information about them is given in Section 6.3.

was made: the bars fixed to the end of the servo axes can not meet on the boarder of the mass, they need to meet as precisely as possible in its center, so that their intersection is actually one point. Otherwise, unnecessary freedoms of movement could be given to the mass, which might make the control of its position even more difficult. The mass itself is made out of brass as well and is composed of two equal cylindrical sections with a threaded hole in their center, so that a threaded rod of the same diameter could be used to fix them together. To do that, two bolts were added to ensure a better fixation. The choice of using brass to do all these pieces was fairly easy, since it is a very dense material[1] and it is easy to manufacture, especially when using a metal lathe or a milling machine, because it is a very soft metal, compared to steel, for instance.



Second cylinder prototype

Material: 6mm MDF

Cylinder radius: 150 mm

Distance from center to servo rotation axis: 109.2 mm

Servo axis length: 16.2 mm

Cylinder mass: 630 g

Mass mass: 150 g

Mass - servo connection: 2 mm threaded rod

Maths properties

R = 150 mm

d = 109.2 mm

s = 16.2 mm

M = 630 g

m = 150 g

Figure 6.4: Information about cp2.

In this prototype, the standard cross servo arms that are included when the servos are bought have been used, but this is bothering because these axes are really short and do consequently only allow a minor displacement. Another problematic aspect, is that it was wanted to allow this prototype to roll in an autonomous way and therefore, the electronics had to be fixed directly onto it. Yet, it could not be fixed on the MDF part of the prototype, because it was too far from the center of the cylinder and it would totally unbalance it. To still be able to run tests, a piece of cardboard has been taped to one of the MDF pieces and cut to allow the cables go through. The electronics could be fixed on the cardboard and so is did not influence the movement of the cylinder that much. However, small rocks were also used to make the cylinder even more balanced. Despite all, a visible acceleration and deceleration were observed when this prototype rolled freely, which was really problematic.

[1]Brass density: between 8400 and 8730 kg/m^3 [51].

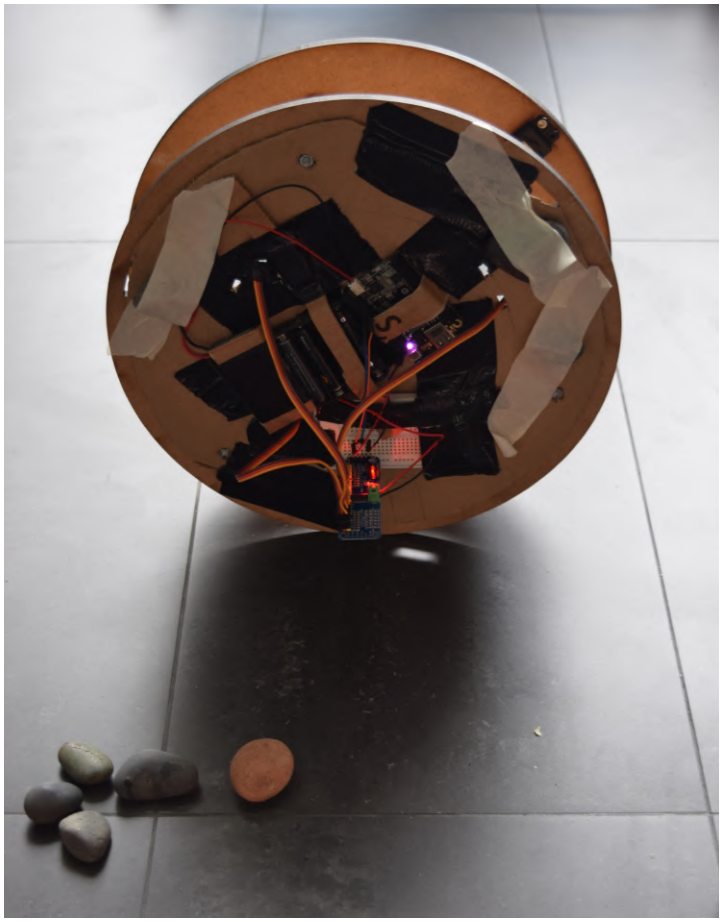


Figure 6.5: cp2 autonomous with the electronics attached to the cardboard piece.

6.3 Prototype 3

The last prototype that has been realized, cylinderPrototype3, shows a really big difference with the other ones, even though the principle remains the same: two rings of stiff material connected with aluminium tubes and threaded rod. Firstly, the material of the cylinder has been changed, PMMA[1] has been used instead of MDF. The main advantage of it is that it is not as sensitive to humidity as MDF, which, as a wood composite, tends to swell in humid environment. Also, this third prototype has been made thicker than cp2, indeed, it is 10.2 cm thick, which allow it to have a better stability. In addition, the amount of material that is required to do the PMMA rings has been optimized, to avoid mass far away from the center of the cylinder (see Figure 6.6), which would increase its inertia. Another important modification is that a PMMA support has been left in the middle of the back ring piece to allow the electronics to be fixed easily. One more modification that has been added is three slits that point to the center of the cylinder and that allow to fix brass masses similar to the one in the center. This system is used to rebalance the cylinder after the addition of the electronics, so that its speed remains constant while it is rolling freely. In the end, the slits have also been used to wind the servos cables, which were too long. The six small rectangles that are visible on the right

[1] "PMMA (polymethyl methacrylate) is a transparent thermoplastic. Developed initially in 1928, it is a synthetic polymer of methyl methacrylate and is often used as a light-resistant or shatter-resistant alternative to glass[38]." PMMA and acrylic glass are two names for the same material.

of Figure 6.6 have a similar goal: they are used to maintain the cables that come from the servos in place, to avoid them to be on the trajectory of the servo arms. Finally, the arrays of holes in the center of the back piece of this prototype were meant to facilitate the fixation of the electronics. During all the testing time, tightening plastic straps were used to fix it, but in the end, the battery pack and the rest of the electronics have been glued on the PMMA using a hot glue gun. To avoid all the bolts to unscrew, but also to spread tension more evenly around the PMMA holes, little plastic rings were placed under every bolt and the mass piece.

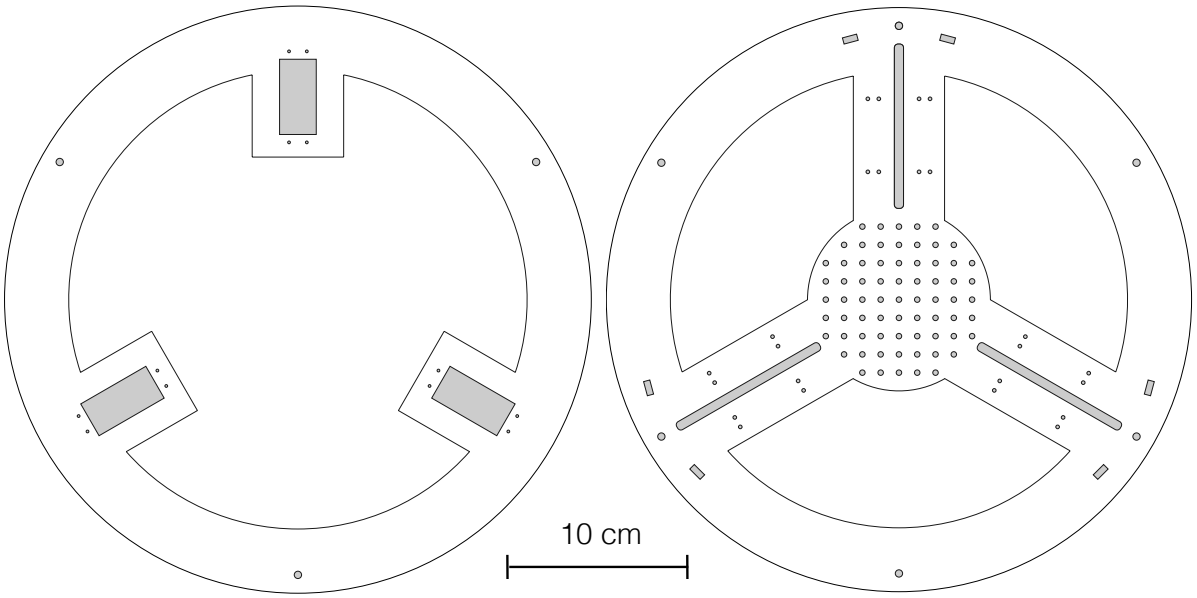


Figure 6.6: FreeCad drawing of cp3 base: front (left) and back (right) pieces are different.

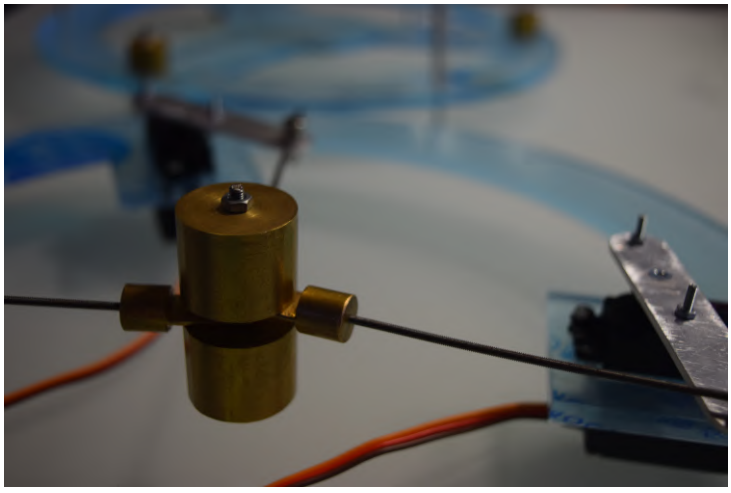


Figure 6.7: Close view of the center mass pieces.

For this prototype, the mass and the parts that connect it to the threaded rods are the same that for cp2 (see Figure 6.7), because that system did work very well. Yet, the servos arms themselves have been built to be a lot longer: they are now 50 mm long, so

more than 3 times longer than the servos arms of the previous prototype[1]. They have been made out of flat aluminium profile and are designed to be fixed on a standard servo axis using two M2 screws. The reason why we did not make the arms to fit directly on the servo is that they should be fixed on the servo gear, which dimensions are not trivial to find and even more difficult to draw in a modelling software. The whole servo axis and the connection pieces to fix it to the mass are visible on Figures 6.8 and 6.9.

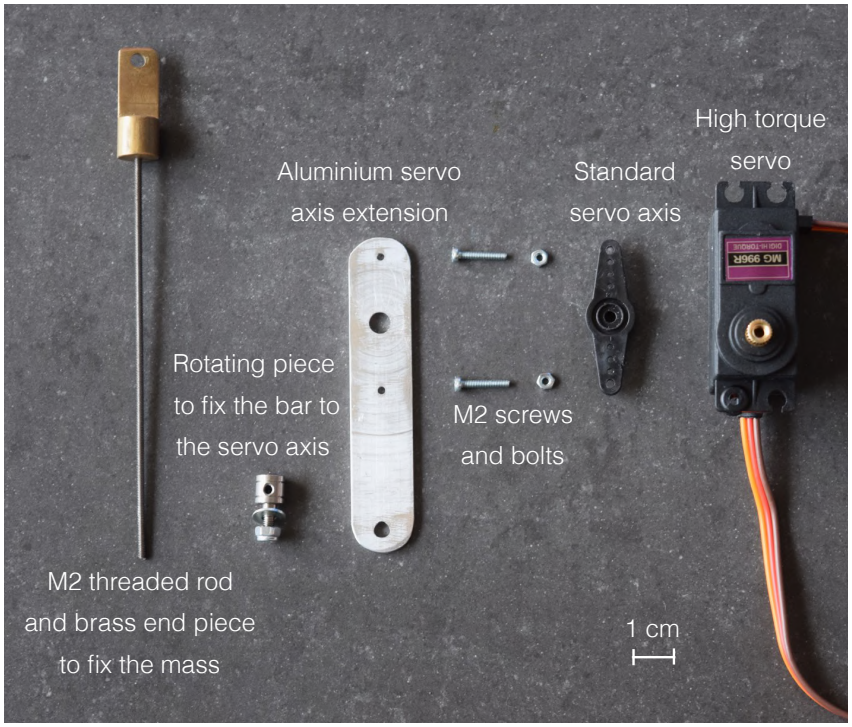


Figure 6.8: The different parts that connect the servo to the mass.

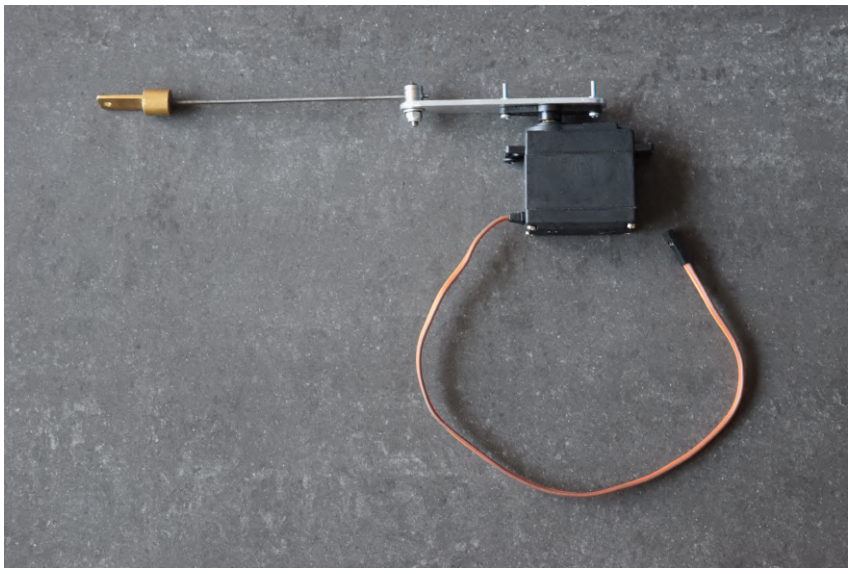


Figure 6.9: The servo / mass connection all built.

[1]The arms have not been done longer because of the speed limitation explained in Section 5.2.

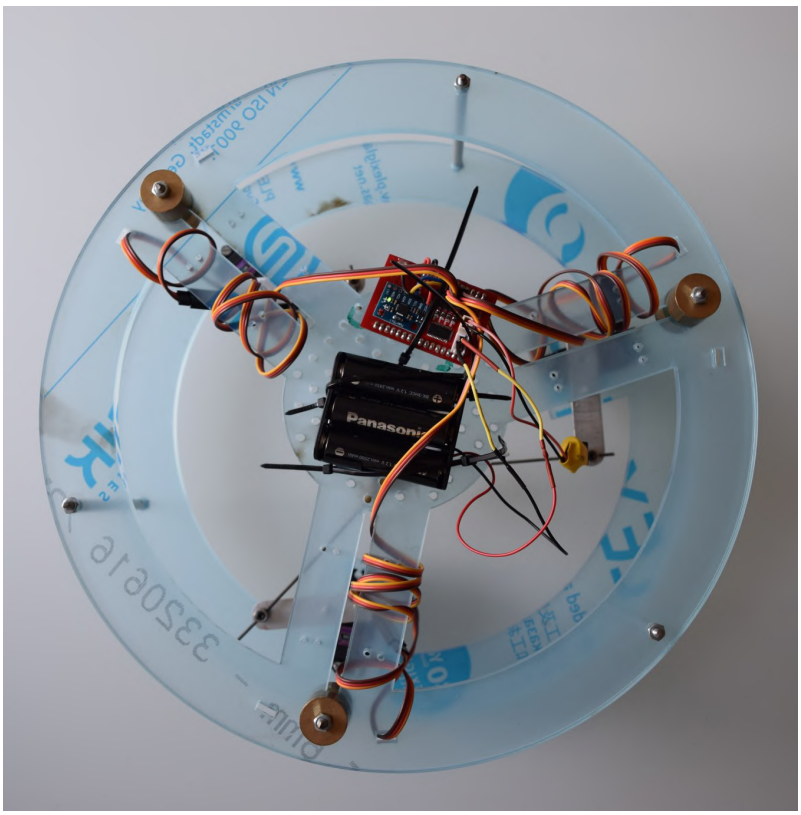


Figure 6.10: Information about cp3.

In all, this prototype has very few defects and suited very well for all experiments and development protocols. Still, if it was wanted to enhance it, polycabonate[1] could be used instead of Polymethyl methacrylate (PMMA), for it is an extremely resistant plastic (it is used for police shields) and our cylinder might need to resist to chocks and uneven surfaces. Another point would be to cover the contact surface with the ground with a non-slip, slightly elastic material, to allow a better grip on various surfaces. However, a technique should be found to allow this non-slip substance to be spread really evenly on the rings, otherwise, the rolling of the cylinder would be hindered. One idea would be to 3D-print a kind of tear using a flexible rubber plastic. Lastly, it might be interesting to build an hermetic version of the cylinder, to make it suitable for outdoor conditions.

[1] "Polycarbonate is a tough, transparent plastic material with outstanding strength, stiffness, and impact resistance [27]."

Chapter 7

Electronics

This chapter explains the process of choosing the micro-controller that suits the best for this project, as well as the I^2C peripherals that have been used to allow the control of the servos and to know the position of the cylinder. This is followed by the explanation of the choices made to realize an extension board for the micro-controller that includes the peripherals in an optimized way. At last, the enhancements of the board in question that could still be done are discussed.

7.1 Choice of the micro-controller

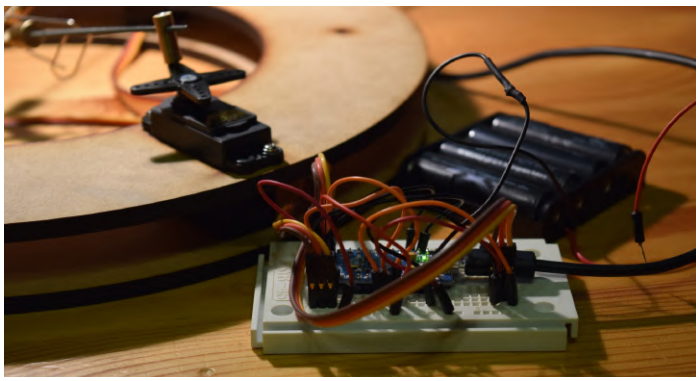


Figure 7.1: Arduino Pro Micro was used on a breadboard for initial tests of the mass movement.

The choice of the micro-controller used in this project was a delicate question due to the important amount of possibilities that exist nowadays. In addition, each ofthem is generally going to be specifically efficient for one type of task.
Also, the range of power of these MCUs is also very wide. In our case, what was needed was something fairly powerful, that had the feature to handle a web page and that worked with the languages that we wanted to use: JavaScript, bash and HTML.
It is for that reason that we definitely could not use an Arduino[2], board even though it is very practical for PWM control since a typical Arduino has over 6 pins that allow it. For that reason, we still used it for the first tests of the movement of the mass around the cylinder's center (see Figure 7.1).

[2] "Open-source electronic prototyping platform enabling users to create interactive electronic objects [22]."

In contrast, the Raspberry Pi Zero, which is an integrated operating system on a very little board (65x40x5), is powerful enough [55]. It also suits for our project because its memory and speed are adapted and it has I/O pins. The interesting fact is that another very similar MCU running Linux by default has come up more or less at the same time as the Pi zero, which is the C.H.I.P. Pro. Table 7.1[1] shows at what point their are similar.

Property	RPi Zero W	C.H.I.P. Pro
Possible Linux interface	yes	yes
BlueTooth included	yes	yes
WiFi included	yes	yes
Aditionnal servo control module required	yes	yes
Flash memory	optional	SLC 512Mb
RAM	512Mb	256Mb
Size	30x65x5 mm	30x45x4.5 mm
Mass	9 g	less than 7.5 g
Price	$ 9	$ 16

Table 7.1: Comparison of Raspberry Pi Zero W and C.H.I.P. Pro features.

It must be added that the actual price of a functioning Raspberry Pi would not be as low as in the table, since an additional SD memory card has to be bought and added to it, whereas the C.H.I.P.'s SLC memory is already on the board, which is also an advantage because Single Level Cell (SLC) has a better durability than SD Card MLC [39].
Knowing these facts, it can be understood that the choice between both technology was not simple but finally it is the C.H.I.P. that has been selected, mainly because it has reliable on board flash memory. It seemed therefore to be an interesting technology to learn. Yet, on the board, they are only two pins available for PWM (pins 9 and 10). To solve this problem, an I^2C module was needed for the control of the servos, it is that module that produces the PWM signal.
During the project, we used the two existing kinds of C.H.I.P.: Pro and non-pro. The non-pro version was used in the second cylinder prototype, which allowed to make tests and get used to the technology. The Pro version, on the other hand, was used in the final prototype and has been very handy because of its ridiculous size and weight, even though problems with the size of the flash memory (512Mb) were encountered. For instance, Vim[2] could first not be installed after everything crucial was in the flash. To solve this problem, a very simple shell command that deals with packages has allowed to recover up to 90Mb, which allowed us to install `vim` without further problems. Here is the command we used.

```
1 | apt-get clean
```

[1] This table is based on the information found in [55, 4] (for the Raspberry Pi) and [43] (for the C.H.I.P. Pro).

[2] "Vim is a highly configurable text editor built to make creating and changing any kind of text very efficient. It is included as "vi" with most UNIX systems and with Apple OS X [29]."

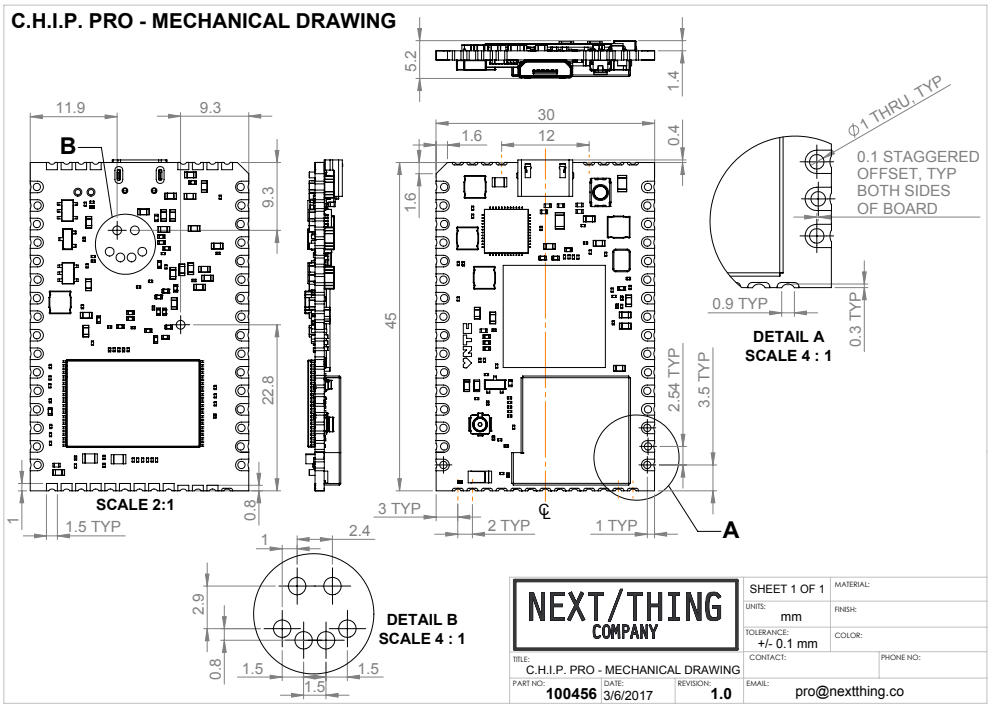


Figure 7.2: Dimensions of the C.H.I.P. Pro [11].

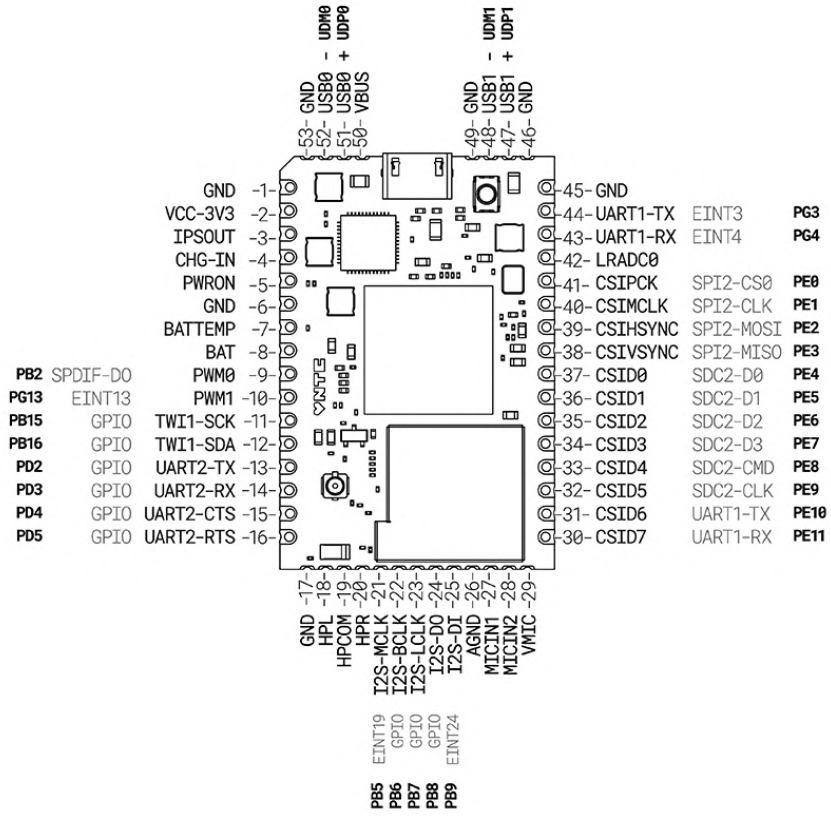


Figure 7.3: Pin mapping of the C.H.I.P. Pro [11].

7.2 Peripherals: accelerometer and servo driver

Overall we needed two main functionalities: firstly, a device that would allow to know the position of the cylinder in space and secondly, a peripheral to control servos using a C.H.I.P., which is not, as it was mentioned before, a standard option. To reach our objectives we had to find what boards were already existing on the market.

Regarding the first functionality, an additional distinction must be made. Indeed, they are two different aspects that are interesting: knowing the position of the whole cylinder relatively to its environment or just knowing at what speed and in what direction the cylinder is moving, which is the information necessary for the control itself.

To determine the speed and direction of the cylinder, the easiest solution was to use a gyroscope coupled with an accelerometer. That solution is a common and inexpensive technology adapted very well to our case since it gives the orientation and the acceleration of a device along the cartesian space dimensions. Many circuits of this kind are easy to get online, but we chose to use the MPU5060[1], which is described as follow in its product specification datasheet.

> "The MPU-6050 is the world's first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output [42]."

This shows that the circuit will be easy to include to a C.H.I.P. system, since it communicates through I^2C. In addition, this chip can be found on a very small board (20x15 mm), practical to include to the project. The device costs approximately 2 USD [23], which is not restrictive. A lot of different data coming from the gyroscope might be used, typically the acceleration in g under the three dimensions of space and the orientation of the chip in degrees. That last information is the one that has been the most useful because we were able to process the angles variation to determine the speed of the cylinder. The gyroscope soldered on its board is showed on Figure 7.5.

At first, it also seemed useful to know the relative position of the cylinder. That thought came mainly from the fact that if the project was extended to a sphere, it could be useful to use it out of sight: we could imagine that it might be used for exploration, for long distance travels or for transport. With that in mind, GPS would have been an interesting approach to know the approximate position of the cylinder. After a global overlook of the eBay sale propositions of devices small enough to be used in this project, the best accuracy is from five to ten meters and the position refresh is maximally every second. Also, the price of a such device is about 7 to 25 US dollars [24, 49, 41]. Another GPS solution, which might be described as more serious, is the XGPS150A Universal Bluetooth GPS Receiver, which has an accuracy of 2.5 meters [54]. To sum up, GPS seems to be a good thing to add to the cylinder if the project was to be further continued.

[1]The datasheet of the gyroscope and accelerometer can be found on the following webpage: https://store.invensense.com/ (last visited on 09.10.2017)

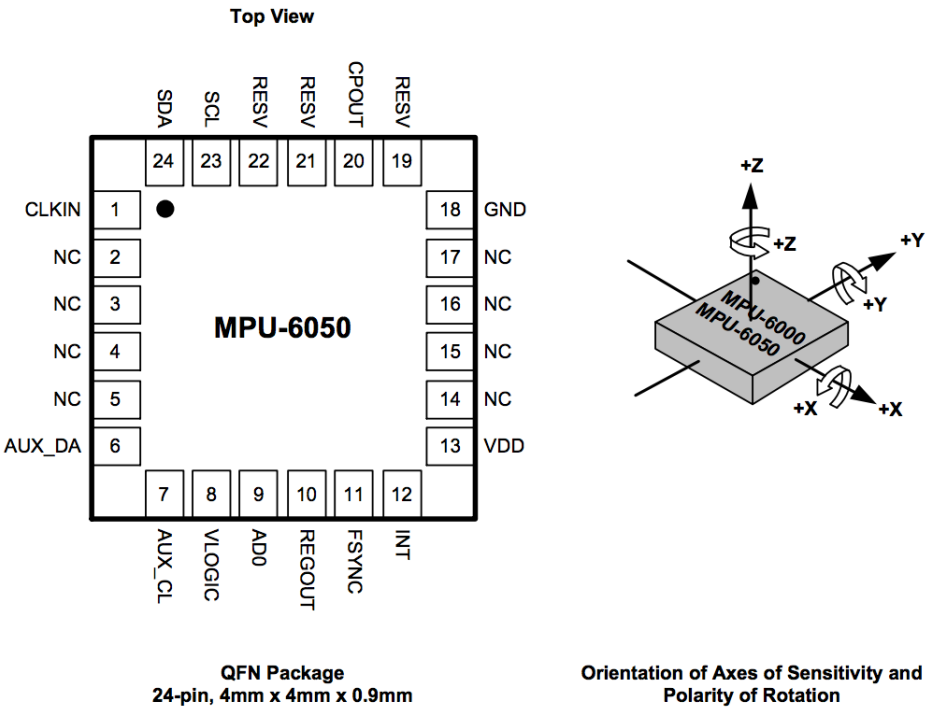


Figure 7.4: Pin mapping of MPU6050 [15]

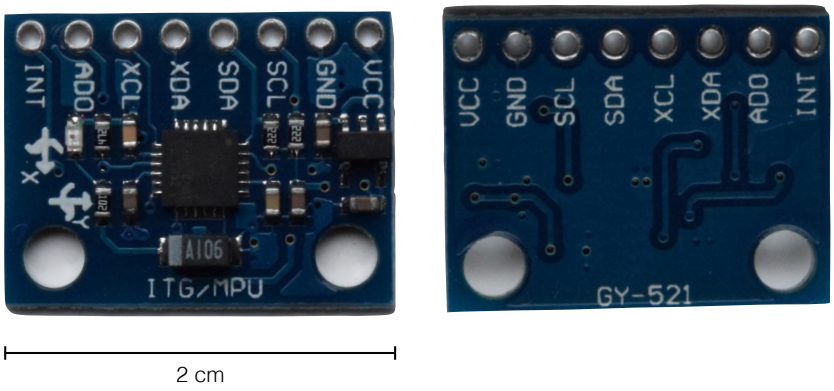


Figure 7.5: MPU6050 gyroscope breakout board

Regarding the second peripheral, what was needed was a chip that could allow to output the PWM signal to control four servos, while having an I^2C input from the C.H.I.P.. The easiest solution was to use a chip initially designed for LED control, the PCA9685[1], which is very common. It is also inexpensive (about 2 USD [25]) and its use is well documented. All these advantages do encourage to adopt this option, however, the only ready board including this driver and adapted to the control of servos is produced by *Adafruit*[2] and is designed to use 16 of servos at the same time. As a result, the board is unwieldy and not at all optimized for our project. Yet, it is a good starting point, because there is not

[1]The datasheet of the LED controller (which we use as a servo driver) can be found on https://cdn-shop.adafruit.com/ (last visited on 09.10.2017)

[2]"Adafruit was founded in 2005 by MIT hacker & engineer, Limor "Ladyada" Fried. Her goal was to create the best place online for learning electronics and making the best designed products for makers of all ages and skill levels [20]."

another obvious option. The quote underneath presents interesting technical points of the PCA9685 that show it does suit very well for what we want to do.

> "It's an i2c-controlled PWM driver with a built-in clock. [...] It is 5V compliant, which means you can control it from a 3.3V microcontroller and still safely drive up to 6V outputs. [The driver has] 6 address select pins so you can wire up to 62 of these on a single i2c bus, a total of 992 outputs - that's a lot of servos or LEDs. [It has an] adjustable frequency PWM up to about 1.6 KHz. [...] [21]"

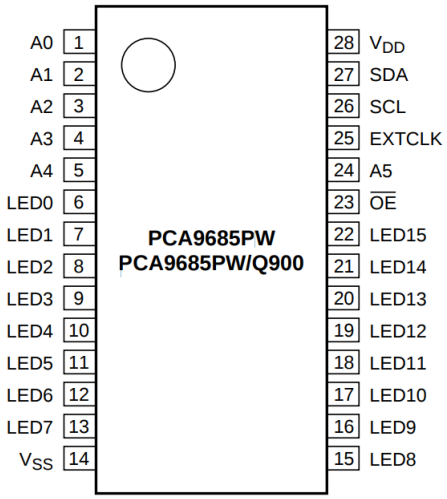


Figure 7.6: Pin mapping of PCA9685 [14]

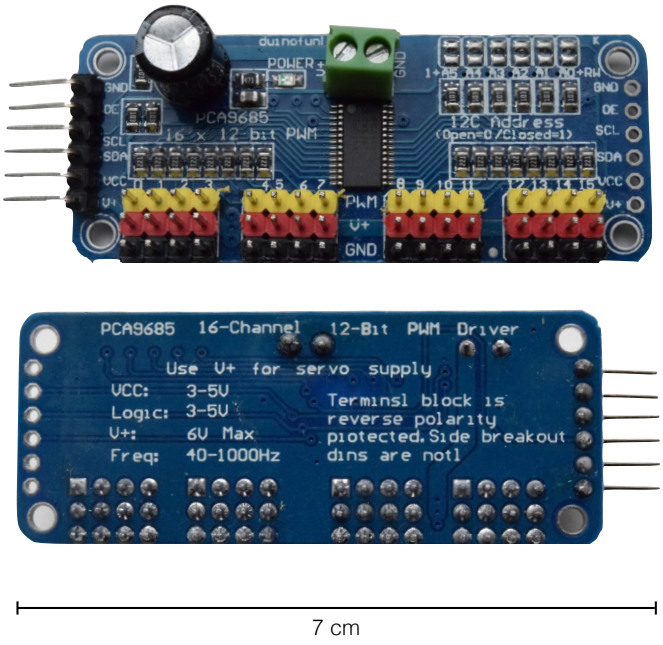


Figure 7.7: 16 channels servo driver based on PCA9685

7.3 The C.H.I.P. Pro extension PCB

After the micro-controller was chosen and the two peripherals we needed were tested in the second cylinder prototype, it has been realized that all these three different boards were very unwieldy (see Figure 7.8). In addition, two different energy supplies were needed, one for the servos and one for the Micro Controller Unit (MCU). It has been done so, because it would have been dangerous to lead all the power used by the servos through the C.H.I.P. board, since it is not adapted to these amounts of current. As a result, not only were they many untrustworthy connections, since cables were just plugged into a breadboard without soldering, but also the whole electronic system was heavy and difficult to fix on the second cylinder prototype. Also, when mounted on the cylinder base, it has caused the whole movement of the cylinder to be influenced, because it was not balanced any more. That influence had to be compensated by adding small masses on the cylinder MDF rings. Yet, this was not really efficient.

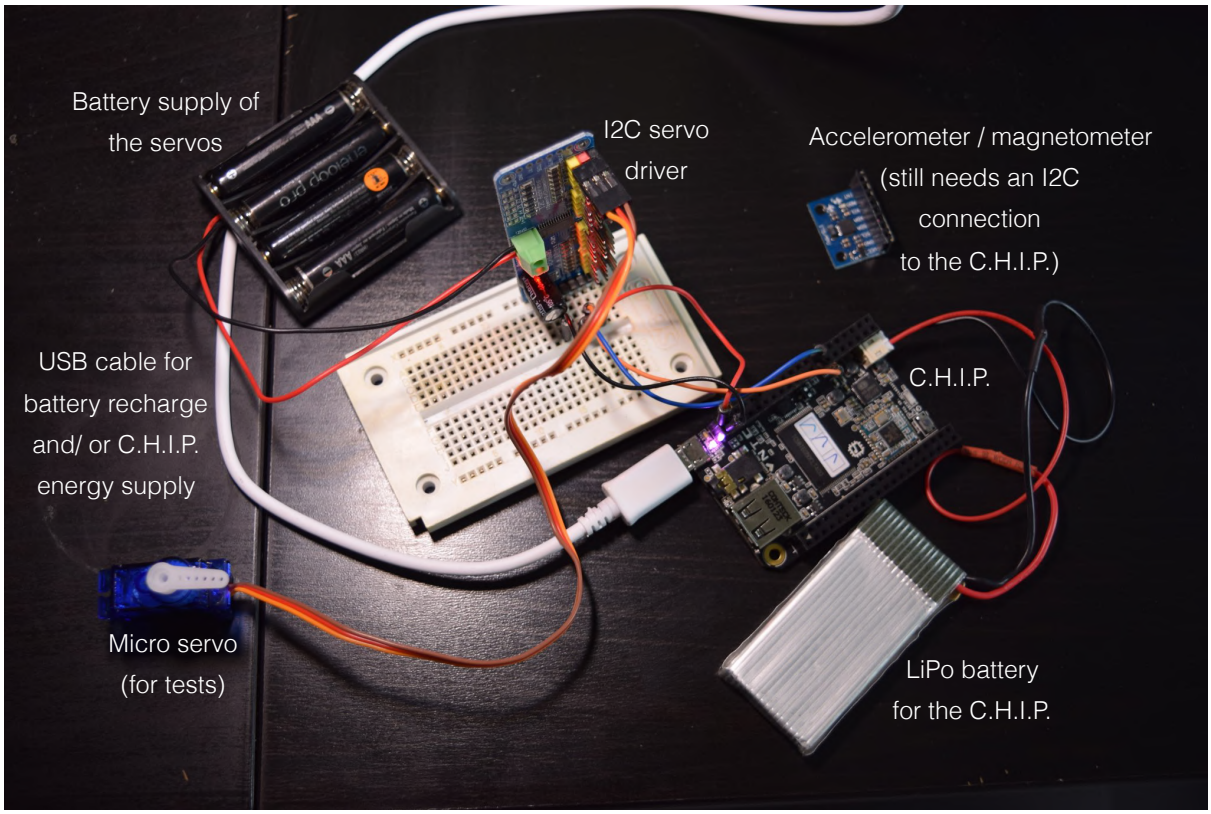


Figure 7.8: Electronics of cylinderPrototype2 during test phase.

For all these reasons, it has been decided to use the Eagle[1] software to create a dedicated extension board that could be fixed on a C.H.I.P. Pro and would allow to group both peripherals on a lot smaller support. What has been done precisely, has been to first find the footprints of the three main components: the C.H.I.P. Pro, the PCA9685 and the MPU6050. Afterwards, they have all been grouped on the C.H.I.P. Pro footprint and jumpers were added on its borders to allow to solder the C.H.I.P. Pro pins directly on it (the two large rows of green pads on Figure 7.9). Then, a two pins jumper has been added on the VCC lane that led to the servos and the C.H.I.P., to potentially measure the

[1] "PCB design software for students, makers, and professionals. Includes 2 schematic sheets, 2 signal layers, and 80 cm2 board area. Available for Windows, Mac, and Linux [28]."

hole energy consumption. Also, four rows of three pins jumpers were added to ensure an easy way of connecting the servos directly to the extension board. We have planned four of these instead of the three necessary, to have directly an possibility to use of this PCB in a sphere, which would require the use of four servos. Two connectors were added for power supply: one connected to the BAT pin of the C.H.I.P. and the other on the IN5V pin. In addition, the PCB has two layers of which one is a ground plane. This allows to avoid the large number of connections to ground and so, make the manual rooting a lot easier.

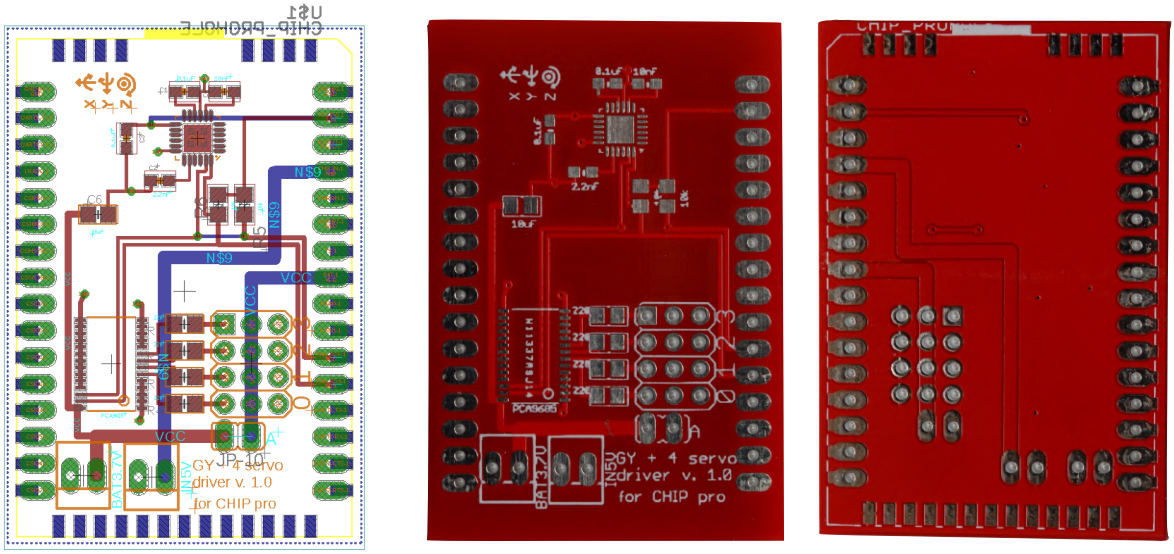


Figure 7.9: Final eagle file and actual board (34x50 mm).

Many aspects have been forecast while the board was created, however, the soldering revealed a few problems that needed to be solved. First of all, the MPU6050 was too small to be soldered by hand. We therefore decided to solder directly the whole board that included the MPU6050. Secondly, a reset button allowing to turn the C.H.I.P. on when a battery is used to power it on had to be added (see Figure 7.10). These two modifications led to have an entirely functional board with a possible I^2C[1] communication with the gyroscope and the servo driver. Yet, many modifications could be done to enhance the board. As a first step, version 1.0 does not allow to power on the C.H.I.P., the servo driver and the accelerometer without turning the servos on as well. This is a problem, because it is often necessary to have the servos not powered for debugging and testing. For instance, a battery pack placed on the BAT input and a 5V power supply had to be used, without the two pins jumper, to be able to run tests. To make those easier, control LEDs could be added to check the presence of current just after the IN5V and the BAT pins. Furthermore, it would be better to add a place to solder the accelerometer board directly on the extension PCB, since it is a lot easier than to develop a technique to solder that component alone using reflow and an oven (since it is too small to solder by hand). Another enhancement would be to add a connector on the IN5V so that it

[1] An I2C communication uses two wires, one of them is a clock and the other one is used to transfer the data. Generally, they are I2C buses, on which many slaves of a micro-controller can be put at the same time [34].

directly corresponds to the standard power supply connector, because in the first version, an adapter has been needed. Finally, it might be useful to include a battery recharge circuit to the board.

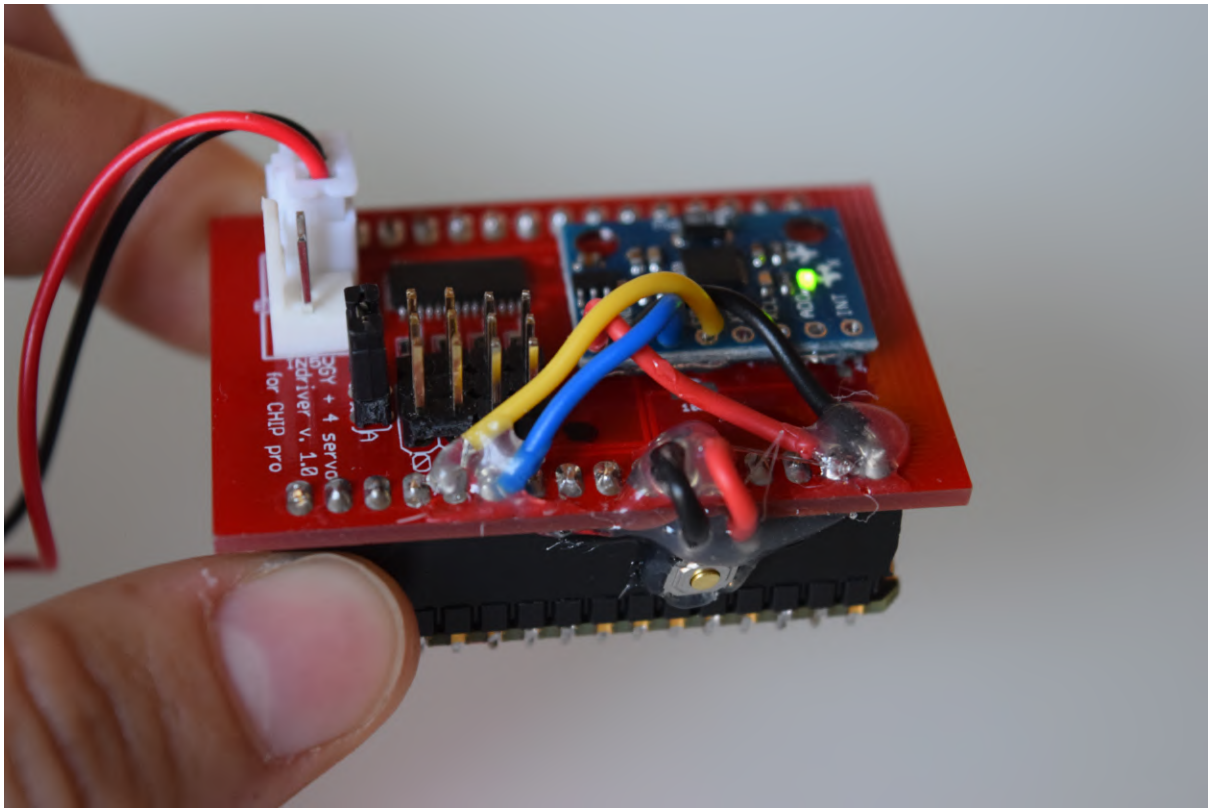


Figure 7.10: Completely soldered PCB.

Chapter 8

Software

In this chapter, the programming languages that were used and the different functions that were provided to the cylinder are explained. This includes mainly JavaScript and bash programming. For instance, we developed a script that includes the mathematical function that we demonstrated in Section 5.3 and two other ones that allow the use of the peripherals we chose in Section 7.2. Also, explanations about the creation of the web page dedicated to the cylinder control are provided.

8.1 Bash, JavaScript and node

After the choice of the micro-controller had been made, the way of programming was fairly defined, according to the capacity of the C.H.I.P. It has already been explained that this MCU is basically like a computer, since it allows to install a complete operating system and a lot of various software. However, we have already mentioned that the memory space has been problematic for the installation of the last packages we needed. Practically, we installed Debian on all the different C.H.I.P.s we used (see Section 7.1), because it is very handy to control from bash[2]. Having no graphical interface, it is the only way to interact with the MCU. To avoid the use of a USB cable, we pre-assigned an IP address to the MAC address of the C.H.I.P.s we used. The two IP addresses that we assigned are 192.168.1.10 (cylinderPrototype2) and 192.168.1.11 (cylinderPrototype3). Also, the C.H.I.P. connects automatically to the WiFi when it is turned on. This configuration allows us to connect to the MCU through ssh.
To be able to program easily, all the code we developed is stored in its own GitHub repository[3]. This is convenient, because to obtain the last version of the code on a new C.H.I.P., it is just needed to clone the repository on it. Since these MCUs do not have a graphical interface, and in order not to have to `push` and `pull` all the time to make tests, we used a mounting protocol called `sshfs` . After the C.H.I.P. is mounted using this method, it is possible to access it as if it were on the computer we worked on. So, we could use WebStorm[4] to edit all the code easily and test it instantly on the cylinder. The command to mount the MCU in a pre-existing folder called `mountedChip` is the following:

[2]Bash is a kind of shell
[3]https://github.com/opatiny/chip
[4]"Powerful IDE for modern JavaScript development [57]."

```
1 | sshfs root@192.168.1.11:/root/git/opatiny/chip/ mountedChip
```

Listing 8.1: Bash command to mount the MCU in an empty local folder.

In addition, we chose to use the last specifications of JavaScript (ECMAScript 6) as the main programming language. So a specific JavaScript engine was required to be able to run JS code from the terminal. We used Node.js[1] that embeds V8 JavaScript engine and allows to execute JavaScript code from the terminal. Moreover it allows to use easily hundreds of thousands packages available on npm[2].
Our project can be considered like an unpublished npm package, since it contains a `package.json` [3] file that describes the project itself, its dependencies and the developer dependencies, which are packages that are only needed by the programmer. We have listed all the main packages we used underneath.

Dependencies:

- `chip-io` : control of the C.H.I.P. IO from JavaScript.
- `debug` : used for debug. Practical because it allows to put comments in the code that are similar to `console.log()` but only visible in a debug mode.
- `delay` : delays in [ms] inside JS scripts using `promises` [4].
- `johny-five` : to allow the basic JS scripts (packages) we recovered from Johny-five to function (c.f. Section 8.1.1).
- `express` : to serve static HTML pages.
- `express-ws` : to use WebSocket[5].

Developer dependencies:

- `eslint` : to enforce syntax of all our JS scripts (spaces, carriage returns, ...), check unreachable code and unused variables. It helps to improve the quality of the code, which is especially required for team work.
- `jest` : to execute test cases, that mainly concerns the JS function dealing with the mathematical formula. Indeed, a lot of the code is difficult to test because it requires I^2C modules.

Also, *travis* was used to run the test cases and check ESLint automatically after every commit. Finally, we used a globally installed package, `pm2` , which allows to run scripts automatically every time the MCU boots or after it has crashed. The process to add a file to `pm2` and general `pm2` commands are described in Listing 8.2.

[1] "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine [44]."
[2] "The npm registry hosts almost half a million packages of free, reusable code — the largest software registry in the world [45]."
[3] This file is on https://github.com/opatiny/chip/blob/master/package.json
[4] "The Promise object represents the eventual completion (or failure) of an asynchronous operation, and its resulting value [48]." The `promise` objects allow to facilitate asynchronous code in JavaScript. They have only appeared in the last version of JavaScript. For us, these objects have been important to be able to make delays, that were used to regulate the speed of the mass in the center, for example. In all, our code is also asynchronous because it uses `async` functions, that include the expression `await`. Yet, we used it to simplify the code to obtain a functioning delay.
[5] Refer to Section 8.4.

```
npm install -g pm2 #to install the package globally
pm2 startup #add file to /etc/init.d to run scripts at boot
pm2 start script.js --name ="scriptName" #to run a script with pm2
pm2 save #to save all scripts running with pm2
pm2 status #to see all scripts running with pm2
pm2 stop all #to stop all scripts actually running with pm2
pm2 restart all #to run all scripts that are saved in pm2
```

Listing 8.2: List of bash commands to add a JS script to pm2.

8.1.1 Peripherals control implementation

The control of the peripherals is based on the Johny-Five[1] library. This library wraps the I²C communication with the gyroscope / accelerometer (MPU-6050) and the servos controller (PCA9685) to provide simple event based functions. Thanks to existing examples from the library we succeeded to implement a reactive system based on the orientation of the cylinder. In general, only two main functions had to be written to control our peripherals. The first one manages the gyroscope and the data it produces (Listing 8.3) whereas the other one allows the I²C control of a servo with a C.H.I.P. through a PCA9685 (Listing 8.4).
All our code is using these two elementary programs.

```
//this program returns values of the accelerometer on change

var Five = require('johnny-five');

var {
    ChipIO
} = require('../preferences');

var board = new Five.Board({
    io: new ChipIO()
});

board.on('ready', function () {
    var accelerometer = new Five.Accelerometer({
        controller: 'MPU6050',
        sensitivity: 16384 //optional
    });

    accelerometer.on('change', function () {
        var result = {
            x: this.x,
            y: this.y,
            z: this.z,
            acceleration: this.acceleration,
            inclination: this.inclination,
            orientation: this.orientation
        };
        console.log(result);
    });
});
```

Listing 8.3: JS code allowing to extract the accelerometer's values. [35]

[1] "Johnny-Five is a framework that gives NodeBots a consistent API [(application programming interface)] and platform across several hardware systems [37]."

```
1  //control of one servo through, the code brings it to a given angle (
       degrees), the angle you give is directly transmitted to the servo.
2
3  var five = require('johnny-five');
4  var chipio = require('chip-io');
5
6  var board = new five.Board({
7      io: new chipio()
8  });
9
10 board.on('ready', function () {
11     console.log('Connected');
12
13   //Initialize the servo instance
14   var a = new five.Servo({
15     address: 0x40,
16     controller: "PCA9685",
17     pin: 0,
18   });
19
20   var degrees = 0;
21
22   a.to(degrees);
23
24 });
```

Listing 8.4: JS code allowing to control one servo using our MCU and a PCA9685. [36]

8.2 From mathematical formula to JavaScript

In Sections 5.1 and 5.3, we developed two approaches that allow to obtain the values of a servo angle (α) depending on the position of the mass. The mathematical formula has been imported into a JavaScript function, that we have called `fomula` [1]. This function has exactly the same parameters as the mathematical formula, which are x (`xmassPosition`), y (`yMassPosition`), d (`distance`), R (`bigRadius`) and s (`radiusServo`). These two first parameters are the ones that will vary depending on the position of the mass that is wanted, whereas the three other ones are constants that depend on the cylinder prototype that is used. In addition, the angle of the mass, *theta*, is called `angleCenter` in all of our code.

At the beginning, we used the values exported from GeoGebra to test the mass movement, but this approach requires to use a 2D model of the cylinder, which is not as general as the mathematical approach. This method was also very unpractical to get final values that could be used and was definitely not the approach we intended to use in the end. After the formula has finally been adapted to JavaScript, it was still required to verify if it was correct. So, this section aims to compare the results of the two approaches: GeoGebra and mathematical formula implemented in JavaScript.

[1] This script is the `/src/returnAngleFormula` file.

	JavaScript	**GeoGebra**	**Difference**
theta [°]	alpha [°]	alpha [°]	[°]
0	0.00	0.00	0.0000
30	30.00	30.00	0.0000
60	60.00	60.00	0.0000
90	90.00	90.00	0.0000
120	120.00	120.00	0.0000
150	150.00	150.00	0.0000
180	180.00	180.00	0.0000
210	159.74	159.74	0.0038
240	137.90	137.90	0.0034
270	112.62	112.62	0.0001
300	81.79	81.79	0.0032
330	43.79	43.79	0.0027

Table 8.1: Comparing values of α from the JavaScript formula and from the GeoGebra model.

In Table 8.1, we have exported the values from GeoGebra and from our JavaScript formula for all the angles in the center that are a multiple of 30° between 0° and 360°. We can see that the difference between the two is very low, since it is of the order of 10^{-3°, which is definitely not a range of precision that would influence the servos' movement. However, we shall question ourselves were this uncertainty comes from. Considering the way JavaScript manipulates decimal numbers, it might be the long calculation that has lead to an uncertainty of that order. Indeed, JavaScript approximates decimal numbers to a precision encoded on 64 bits in the following way, according to w3schools.

> "Unlike many other programming languages, JavaScript does not define different types of numbers, like integers, short, long, floating-point etc. JavaScript numbers are always stored as double precision floating point numbers, following the international IEEE 754 standard. This format stores numbers in 64 bits, where the number (the fraction) is stored in bits 0 to 51, the exponent in bits 52 to 62, and the sign in bit 63 [56]."

Due to that, the kind of uncertainty that can be seen in Figure 8.1 is understandable. However, the magnitude of the differences of Table 8.1 can not be explained. Anyway, it can be considered that the mathematical formula of Section 5.3 and its implementation into JavaScript are correct despite the minor uncertainties.

```
> 0.1 + 0.2
0.30000000000000004
```

Figure 8.1: Node.js computing decimal numbers.

8.3 Prototype parameters and calibration

In this section, the files containing the parameters of the different prototypes are explained. Also, we give a general procedure that allows to make a similar file adapted to another prototype. The advantage of using such scripts, that do only contain parameters, is that the code is then really easy to adapt to any predefined prototype[1]. Basically, what is defined in these scripts is firstly the cylinder dimensions, which were either measured with a caliper, found on product specifications datasheets or taken from the FreeCad models that were used to cut the cylinder base.

The second thing that we defined in the parameters files are values that allow to individually calibrate the three servos. Indeed, when the programming of the servos angles became more precise, we had problems with tensions in the system. Also, the real movement range of the servos is inferior to 180° (under 160°). It was therefore really important to adapt the code to this constrain, because if it was not, the hole cylinder electronics could have been damaged. The cause of the tensions was not easy to find and was actually quite complex: there is a difference between the angle in degrees that is send to the servo motor and the real angle to which it moves.
So, we used a crossed multiplication to transpose the angles returned by the `formula` to the corresponding angles that must be send to the servos.

This whole procedure is precisely described in Section 8.3.1.

8.3.1 Parameters acquisition procedure

When using a new prototype, create a new JS document containing the parameters of the system:

- The file should be named `cylinderPrototypeX.js` , where X is the number of the prototype
- `radiusServo` (s)
- `bigRadius` (R)
- `distance` (d)
- `cylinderRadius` (the outer radius of the cylinder prototype)
- The parameters of the servos (in the three JS objects `infoServo1` , `infoServo2` and `infoServo3`)

Then, to know the relationship between the real angle to which the servos move and the angle that was sent to them, it is needed to do as follows:

- Assemble the servos on one face of the cylinder.
- Print a protractor, laminate it and center it on the servo axis, with the 0 degrees toward the center of the cylinder (see Figure 8.2).

[1] All these scripts are in `chip/src/prefs`.

- Put the arms on the servos so that they can be easily moved of position.
- Move the servos axes to 0 code degrees .
- Move the arms so that they are the closer to point the center of the cylinder (they will not point it exactly, probably, because they are not enough gears on the servo axis). The arms must be slightly in a negative angle.
- Move the axes to 90 code degrees and go back to a small code angle, do that as many times you need to find the code angle that correspond to 0 real degrees on the protractor.
- Write down the minimal real angle (`minRealAngle` , that we set as 0°) you just found and the code angle that corresponds to `minRealAngle` (`codeMinAngle`). Add this information in the object `infoServo` .
- Fix the servos axes with screws.
- Place the servos axes to zero code degrees and then to 180 code degrees. Look how many real degrees it is on the protractor (this is `codeMaxAngle`). Sweep from small code angle to big one, until you find the smaller code angle that has the same real angle as for 180 code degrees (`maxRealAngle`).
- Write `maxRealAngle` and `codeMaxAngle` in `infoServo` .

This procedure was uncertain for two reasons: first, the position of the protractor was not really precise, because it was aligned using the servo orientation. Also, the measures of the angles themselves were not exact, since the servo arms (especially for cp3) were very large (15 [mm]). To improve the values, a solution would have been to fix something similar to needle on the servo axis, so that the angles were lot more precise.

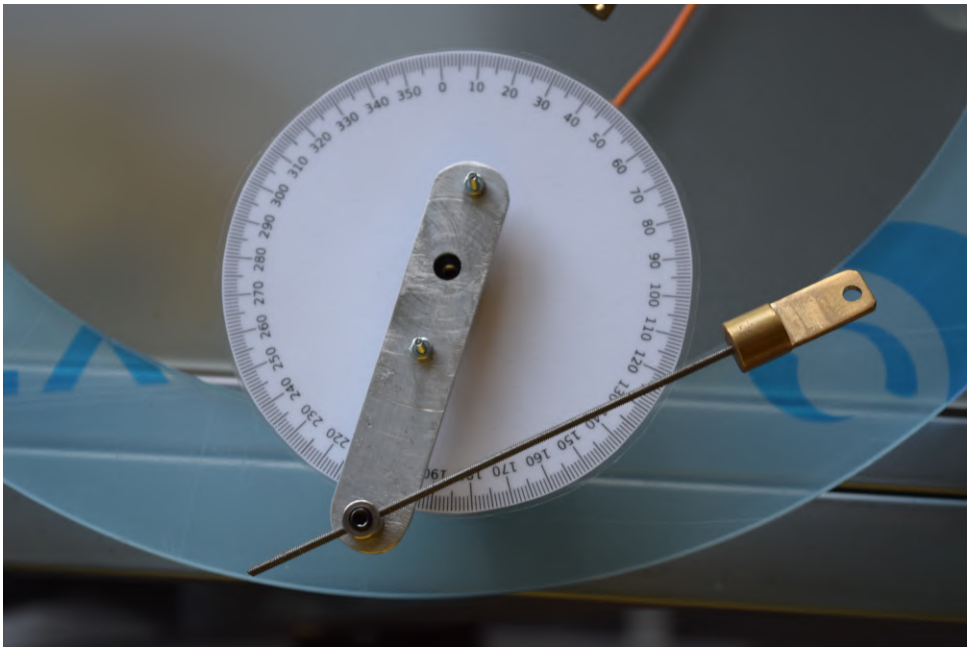


Figure 8.2: Servos calibration setup, on cylinderPrototype3.

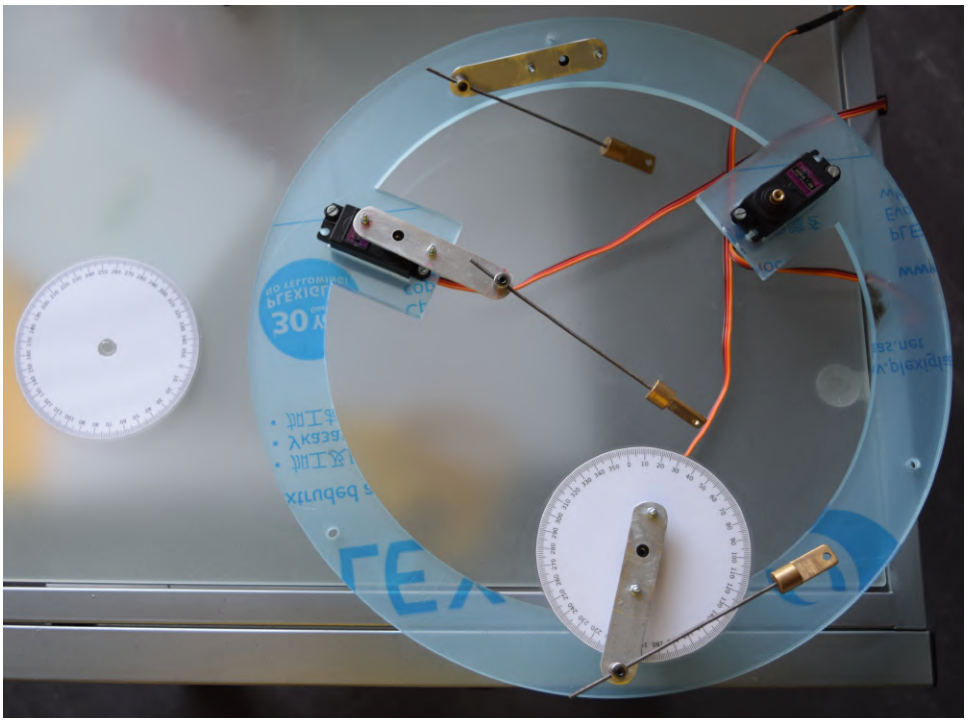


Figure 8.3: Servos calibration setup, on cylinderPrototype3.

The dimensions parameters of cp3 are shown in Listing 8.5.

```
const radiusServo = 50; //s in [mm]
const maxRadiusCenter = 40; //maximal radius r in [mm]

const bigRadius = 100; //R in [mm]
const distance = 100; //d in [mm]
const cylinderRadius = 160; //radius of the cylinder in [mm]
```

Listing 8.5: Dimensions of cp3.

The final `infoServo` object for one servo is shown in Listing 8.6, as well as the function that allows the conversion from code angle to servo angle.

```
var infoServo1 = {
    minRealAngle: 0, //minimal real angle that the servo can reach
    maxRealAngle: 156, //maximal real angle that the servo can reach
    codeMinAngle: 10,  //code angle that corresponds to trueMinAngle
    codeMaxAngle: 180 //code angle that corresponds to trueMaxAngle
};

// function that allows to transpose the formula angle to servo angle
function setServoAngle(angle, info) {
        var trueAngle = Math.round(angle * (info.codeMaxAngle
        - info.codeMinAngle) / (info.maxRealAngle - info.minRealAngle)
        + info.codeMinAngle);

    return (trueAngle);
}
```

Listing 8.6: Values that allow to calibrate one of the servos of cp3 and corresponding function.

8.4 Web page for the control of the cylinder

In this section, we explain aspects of the final code that is used for the control of the cylinder. It is the code grouped in `chip/src/webControl` , which contains all the files dedicated to the web interface, but also the final JS functions with specific control features.
To simplify the control of the cylinder, a web page has been developed, which includes sliders and buttons. We used this system to make the control easy and understandable for everyone. This is an important goal, since it is one of the aspects we identified in the Sphero project, about which we talked in Section 3.1. The resulting web page is shown in Figure 8.4. It includes three buttons that respectively, from left to right, allow to modify the radius on which the mass moves with a slider, place the mass in the center and run the script that maintains the cylinder stable on a gentle slope. Also, epoch[1] in [ms] is shown.

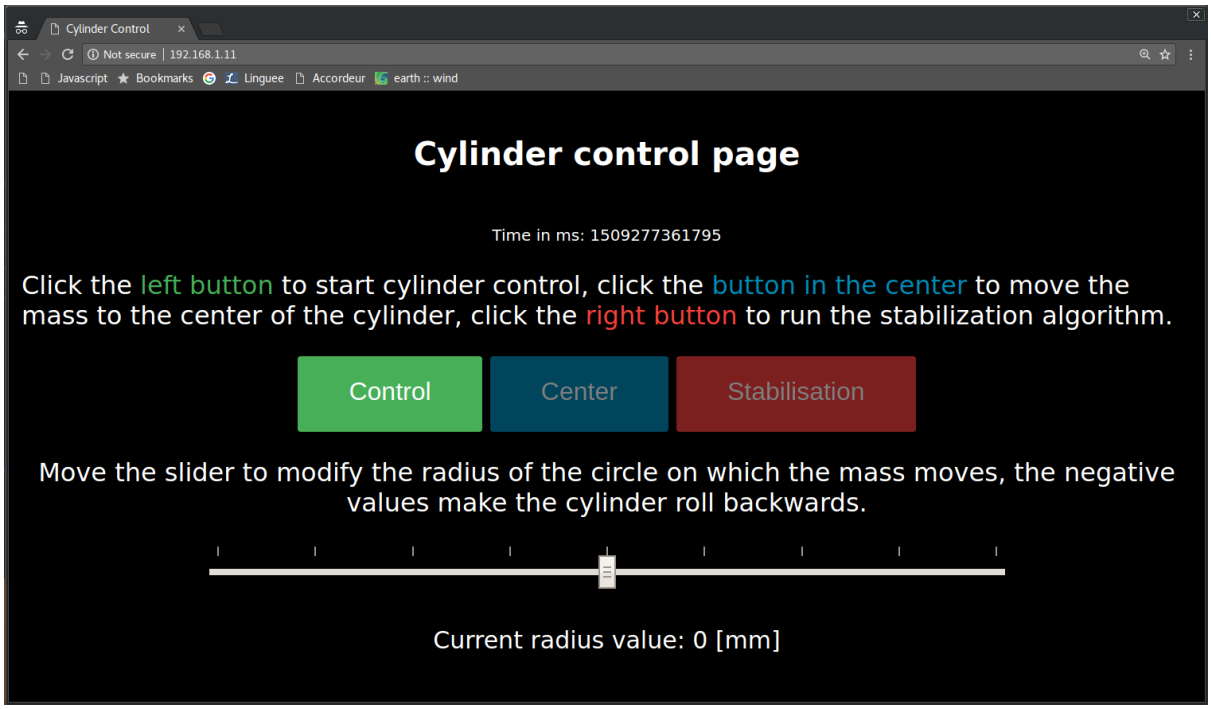


Figure 8.4: The final web page, when the 'Control' button is clicked.

To obtain this result, it was needed to learn basic skills in HTML and scripting. HTML, or HyperText Markup Language, is a language that allows to format text, images, figures, and so on. It is really popular since nowadays, all webpages scripts are in this language. It is the web browser that transforms the script into a graphical output that is shown on the screen. So, while being on a webpage, the URL[2] that is on the top of the page is a path that leads to the location of the HTML file corresponding to the page on a public server. The server is addressed with a domain name that is attributed to its IP address. For

[1] "The Unix epoch (or Unix time or POSIX time or Unix timestamp) is the number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT), not counting leap seconds (in ISO 8601: 1970-01-01T00:00:00Z). Literally speaking the epoch is Unix time 0 (midnight 1/1/1970), but 'epoch' is often used as a synonym for 'Unix time' [31]."

[2] URL is an acronym for Uniform Resource Locator and is a reference (an address) to a resource on the Internet [46].

instance, consider the web page www.google.com, if we use the bash command `nslookup` , we can see that on the 09.18.2017 at 13:34, this page corresponded to the IP address 172.217.16.196.

Until now, we explained that any content on the web corresponds to an HTML file that exists on a public server. However, one information lacks: the method one's computer uses to access these files. This method is HTTP, for HyperText Transfer Protocol. It is a communication system that has become standard for all web researches [58]. The default port for HTTP communications is port 80, which we use in our project for the HTTP communication between the web page and the C.H.I.P. This has the advantage that, to obtain the web page on any computer, it is not necessary to specify the port used for the communication (since it is the standard one). In addition, our HTML homepage is named `index.html` , which is also the file that is served by default. So, when connected to the same WiFi rooter as the cylinder, it is just needed to type `192.168.1.11` in the address bar of a web browser to get the web page. The first phase of the web process is just loading this static web page. We use an npm package called Express, which is a "fast, unopinionated, minimalist web framework for node [32]" to serve the page, like any normal web server.

Yet, there is one problem with standard HTTP connections: it is that, when someone charges a static web page, the connection stays only opened long enough for the client to get the HTML file. In our case, it would not be possible to use standard HTTP connections, because every time the person that controls the cylinder modifies a parameter of the web page, it must be "instantly" sent to the C.H.I.P., which must react as quickly as possible to these changes. To provide a bi-directional, event-based message handler that allows to interact constantly from client to server and inversely, we decided to use WebSockets (WS for short). This technology is described as follows on MDN:

> "Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.
> Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a send() method, and receive data from server to browser by an onmessage event handler [53]."

So, what we did is to include WS in `index.html` , inside a JS script that it contains. Therefore, when this file is required by a client, it will open a WS connection, that will remain opened as long as the client stays on the web page. Through this connection, we send strings, which are the only type of message allowed with WS. These messages are event-based, and are sent through WiFi every time the client changes a parameter of the web page. We send two types of messages: when one of the three buttons is clicked and when the value of the slider is modified. The information that we send, which is normally an object, is stringified using the `JSON.stringify()` JS native function. When the message arrives on the server side, it is converted to an object again using the `JSON.parse()` function. Figure 8.5 illustrates the process that has just been explained.

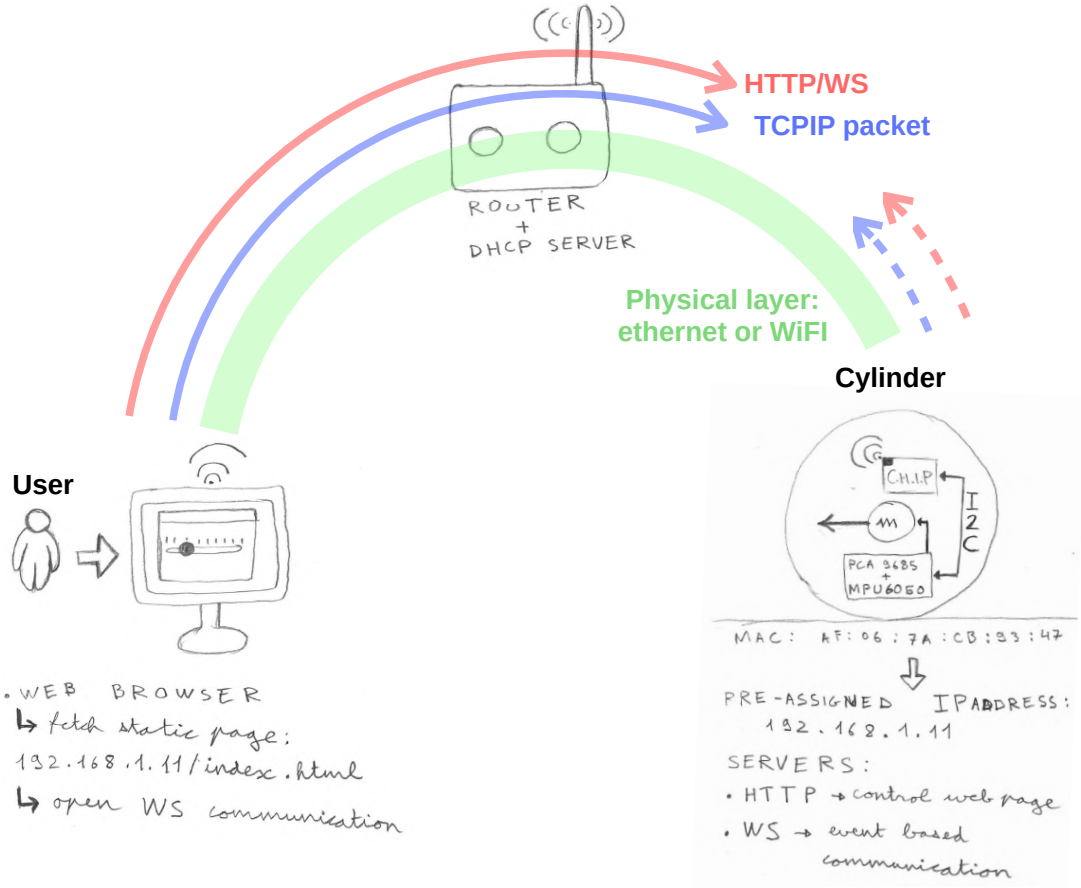


Figure 8.5: Communication protocol overview: from user to cylinder prototype.

To make it easier to understand how the WS messages are used in the code that is in the C.H.I.P., an annotated screenshot of the code structure is showed on Figure 8.6.

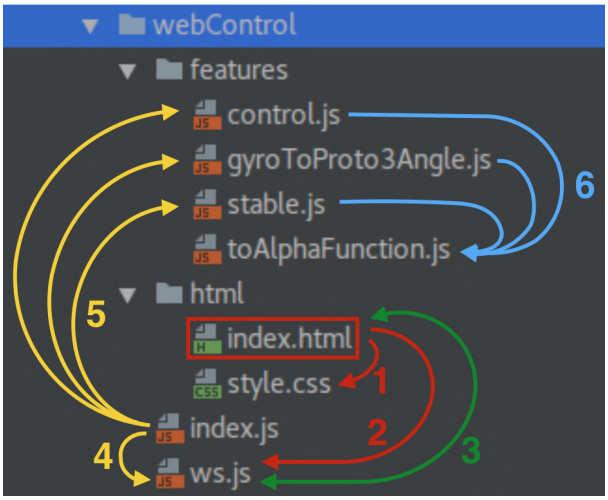


Figure 8.6: Structure of code in the `webControl` folder and order in which the files are run.

Underneath, we explain the different steps that happen in the code after a client has required the `index.html` file with Express (c.f. Figure 8.6).

1. The file `index.html` is read on the client side. It notices that the `style.css` file contains its layout settings. The `style.css` file is loaded.

2. `index.html` contains a JavaScript code that instantiate a WebSockets connection with the server.

3. On the server, the script `ws.js` will ensure the communication with the client through WS. Since there, any modification of the web page, on the client side, produces an object that is sent to `ws.js` , where the object is called `prefs` . The `prefs` object is exported.

4. The main JavaScript script, `index.js` requires `prefs` from `ws.js` . It does also manage the I2C communication with the gyroscope and the servo driver.

5. `index.js` requires all the different functions that correspond to a specific cylinder feature (`control.js` , `gyroToProto3Angle.js` and `stable.js`).

6. These three functions require a last JS script that contains the mathematical function that was demonstrated in Section 5.3.

8.5 Final code structure

It can be considered that different levels of programming were entcountered over time, from very basic programs to more and more complex algorithms. In this section, all the code that we developed will be briefly presented. In all, we wanted to get a code that is divided in chunks as small as possible, because it is a lot easier to debug, understand, maintain and reuse.
First of all, all the code we developed is placed in the `opatiny/chip/src` folder of the repository containing the code that corresponds to this project[1]. We consider `opatiny/chip` as the first level of the code structure. In this folder, we have ordered the code so that the general programs used many times are directly accessible. This concerns the JS function containing the mathematical formula, a script that is preferences about the prototype that is used and a file containing the servos pins definition. All the folders that are described underneath are on the same level.

- `__tests__` : test cases of the `formula` and others.

- `commandLineFunctions` : various functions that allow to control the cylinder with parameters that are directly written in the command line. It contains features that allow to move the cylinder of a certain distance, make the mass move on a circle and make the mass move on a spiral. All files are required by `main.js` .

- `firstTests` : chunks of code that have been initially developed. The two basic Johny-Five programs adapted to our project can be found here, as well as a program to make the servos sweep and a code that controls the movement of three servos depending on the position returned by an accelerometer.

[1] https://github.com/opatiny/chip/tree/master

- `prefs` : definition of constant parameters of cylinderPrototype2 and cylinderPrototype3 like dimensions and servo calibration. They are required by the other scripts. These scripts are practical to adapt the code to any new prototype[1].
- `realGyroUse` : first code that uses the accelerometer values to define the mass position. Still uses command line functions. Two features are functionning in this folder. First, the ability to keep the mass at a constant position while the cylinder moves. That position is the one that produces the biggest possible torque. This allows to accelerate the cylinder. Secondly, the code that allows to rebalance the cylinder constantly, to keep it stable on a gentle slope.
- `servoControl` : very first code files that allow to move the mass to any Cartesian position (`goToPoint.js`) or to make it move on a circle around the center of the cylinder (`movingOnCircle.js`).
- `webControl` : final code that allows the control of the cylinder through a web page. Three main control modes are available: modify the horizontal distance from the center of the cylinder to the mass, place the mass in the center of the cylinder and let it stabilize on a gentle slope. All code files are required by `index.js` .
- `ws` : webSocket example that helped to understand how it works[2].

An example of command to run any JS script from bash using node is:

```
1 | node main.js
```

Listing 8.7: Bash command to run the `main.js` script.

In addition, after we developed the command line functions, a typical command would be:

```
1 | node goToPoint.js -x 20 -y 10
```

Listing 8.8: Bash command to run the `goToPoint.js` script with two command line arguments: `-x` and `-y` which are the coordinates of the mass position in mm.

[1] The way in which these parameters are defined is explained in section 8.3

[2] This code is taken from https://github.com/cheminfo-js/chip

Chapter 9

Results

This chapter aims to present all the results obtained during this project. We mainly focus on the final cylinder prototype (cp3) and the features it was provided with. To see a video that summarizes cp3 functions, follow this link: https://youtu.be/QggOjoGjJhA.

9.1 Mass' movement

The last cylinder's prototype can have a maximal radial displacement of the mass of 40 mm. To verify the precision of the mass' movement, we had the idea to take long aperture pictures of the cylinder, when a small LED module is fixed to the mass (refer to Figure 9.1). That way, we could really observe the trajectory that the mass follows. Three of these tests are shown on Figures 9.2 and 9.3. On this first figure, it can be seen that the movement is definitely not perfectly circular when it is programmed to move on a circle. However, it is really reproducible, since the mass has completed several revolutions while these pictures were taken. Figure 9.3 shows how precise the movement is, since the spiral maximal diameter is only 8 cm long. We can also observe that with the parameters that were used, the smaller the radius the mass moves on, the less circular it is.

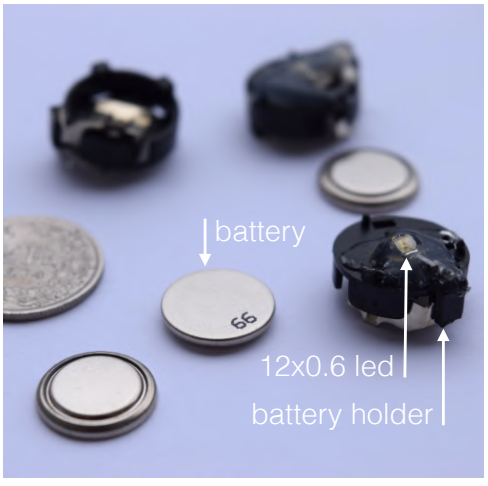


Figure 9.1: Module used for long aperture pictures: button battery holder, SMD resistor and LED.

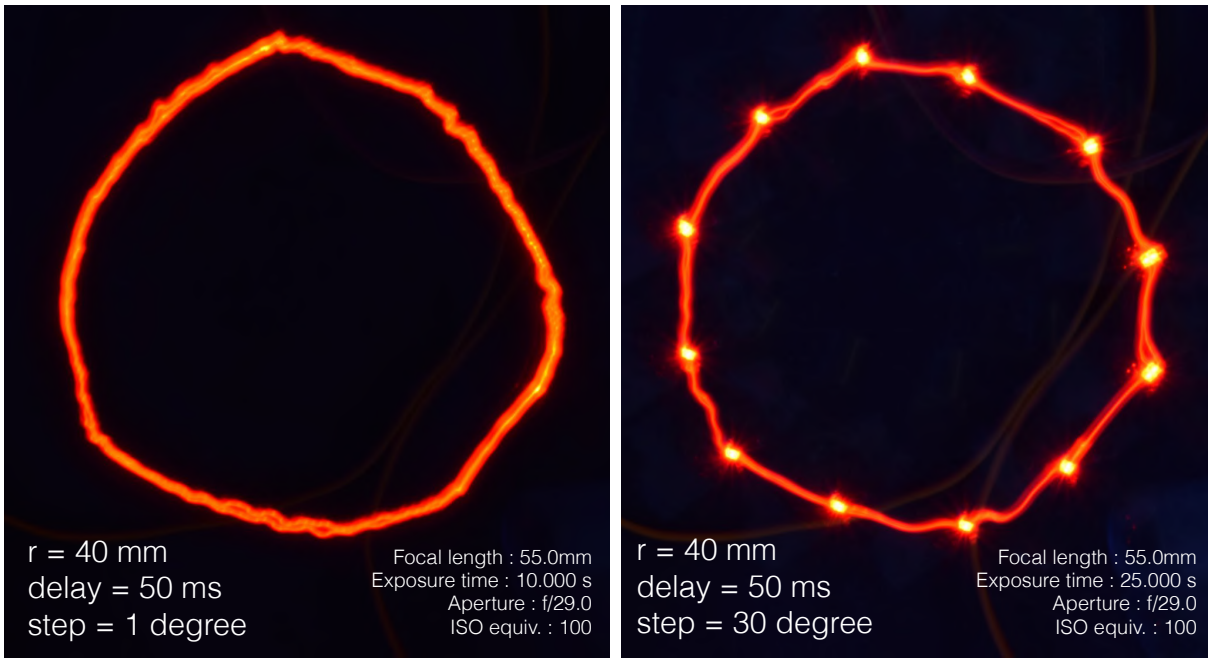


Figure 9.2: Long aperture picture of the mass moving on a circle when: left, increasing *theta* of 1 degree every 50 ms and right, increasing *theta* of 30 degrees every 50 ms.

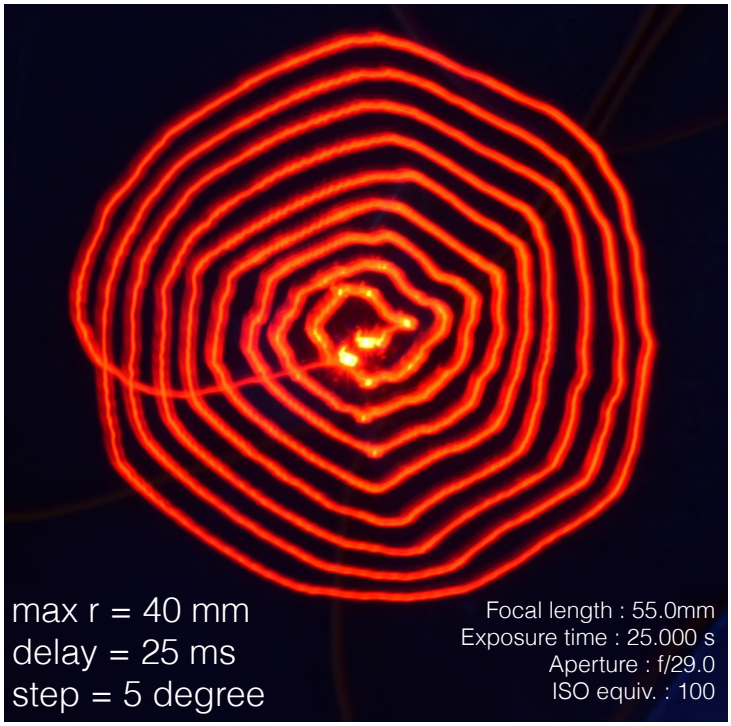


Figure 9.3: Long aperture picture of the mass moving on a spiral from maximal to minimal possible radii (40 mm to 0 mm).

9.2 Cylinder's movement

To consult all the data used to make the charts in this section, please click here[1].

Once cp3 was sufficiently reliable not to suddenly turn off while we were testing it, experiments have been carried out to quantify the movement of the cylinder depending on the radius the mass in the center moves on.
To realize this experiment, we filmed the cylinder with the mass in the center moving on various radii. To start the cylinder's movement once that the mass was on a precise radius, the cylinder was held by one person while the other set the radius. The experiment was held on a tiled floor, which was irregular at the joint of two tiles. Radii from 10 mm to 40 mm were tested, with an increase of 5 mm every time. A test was also made with a 5 mm radius, but the cylinder did not move at all, due to the irregularities of the rings' contact surfaces and to the ones of the floor.

Then, we used the LoggerPro[2] software to analyze the data. Indeed, LoggerPro allows to import movies and to place points on the movies at different times. Therefore, the position of an object depending on time can be inferred. The software also allows to set the scale of the image by measuring the size of an object of the movie. In our case, a double meter stick was used. In addition, the time and space origin can be set. We chose to define time zero the moment when the cylinder is released, also, position zero is the position of one of the servos at time zero. The further measure points are placed every time one of the servos reaches a certain height (see Figure 9.4).

Figure 9.4: Screenshot of the movie with annotations. Blue points: measures, Green line: allows to set the scale, Yellow axes: sets origin.

[1] Equivalent url:
docs.google.com/spreadsheets/d/1cB8XICBxSQ7DYsR84JScTMrtCtLb7akIpOW2s1j5ruM/edit?usp=sharing

[2] "Logger Pro is award-winning, data-collection and analysis software for Windows and Mac computers [40]."

The data exported from LoggerPro consists of the position of the cylinder at different times and of its speed, which is the derivative of the position. Regarding the uncertainties, we consider the one on time as negligible compared to the one on position. Indeed, time is measured by counting the number of frames of the movie, which ensures a precision of 1/24th of a second. The precision of the position, on another hand, could be approximated to 5 cm. This uncertainty is a consequence of the method used to mark the measure points (in blue) on the movie. In addition, the fastest the cylinder was rolling, the less precise the measures are, because the position variation between two consecutive frames became too big, and the frames themselves were not sharp. Lastly, there could be an influence of the deformation of the image caused by the lens of the camera.

The results that were obtained are displayed on the underneath charts. They are interesting, because Figure 9.5 shows that the position of the cylinder evolves quadratically. Therefore, we can deduce that the speed should be growing linearly and that the acceleration is constant. Logically, that makes sense, since the torque produced by the mass is supposedly constant, and that it is the only one affecting the speed significantly (unlike friction). Also, Figure 9.6 confirms that there is a linear phase in the speed of the cylinder. Yet, Figure 9.7 shows that the results also fit very well a logarithmic curve, especially on the final values. This could be explained by the fact that the cylinder then reaches a limit speed that is caused by the servos motors maximal speed. However, the maximal speed does not really seem to have been reached, since the curves are not actually flat at the end. To obtain better results, the movement should have been observed on a longer distance, which is nevertheless complex, since it requires a wide space.

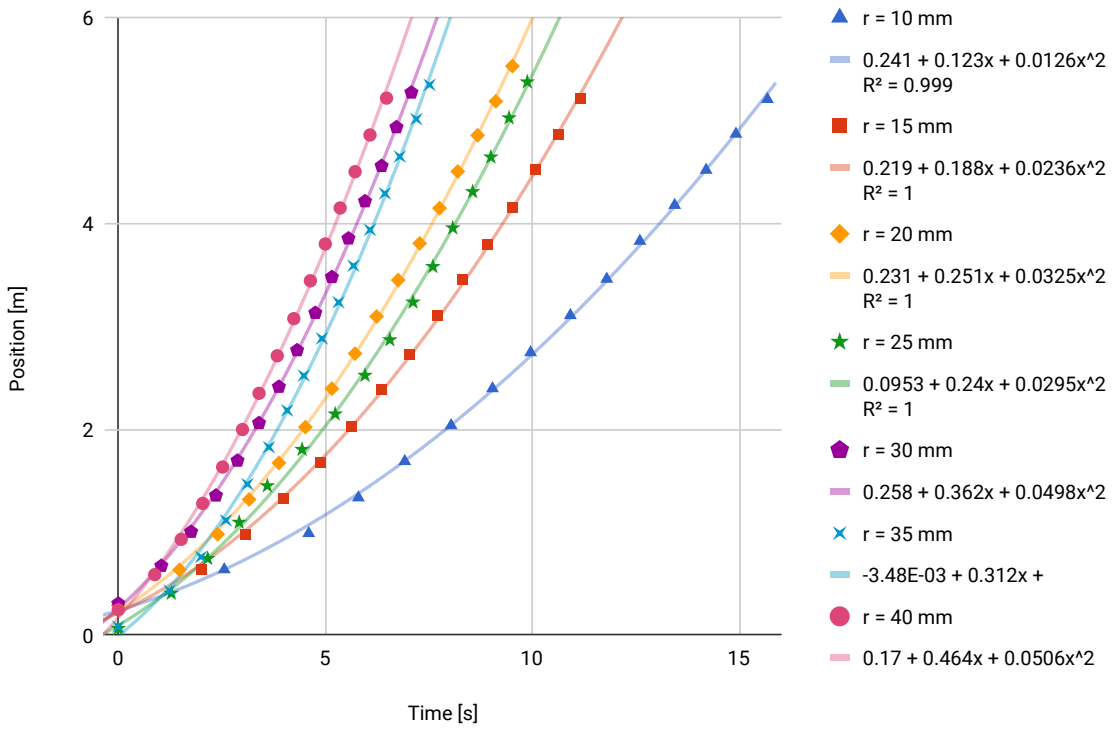


Figure 9.5: Position of the cylinder versus time for different mass' radii with quadratic trendlines.

In addition, the same type of logarithmic behaviour can be seen on Figure 9.8, which displays the maximal speed the cylinder can reach depending on the radius the mass moves on. Again, that could be explained by the servos reaching their maximal possible speed. Finally, Figure 9.9 shows measures done after the cylinder had rolled over a long distance with the mass moving on the maximal radius. On it, we can clearly see what its maximal speed is: about 1.03 m/s. Also, the speed is extremely constant.

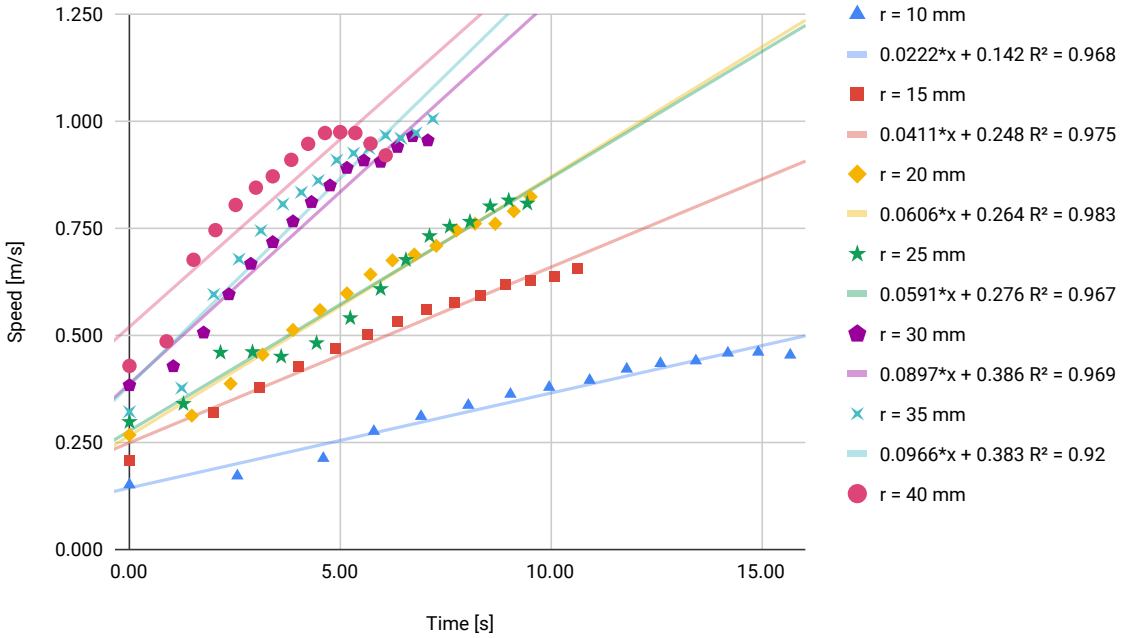


Figure 9.6: Speed of the cylinder versus time for different mass' radii with linear trendlines.

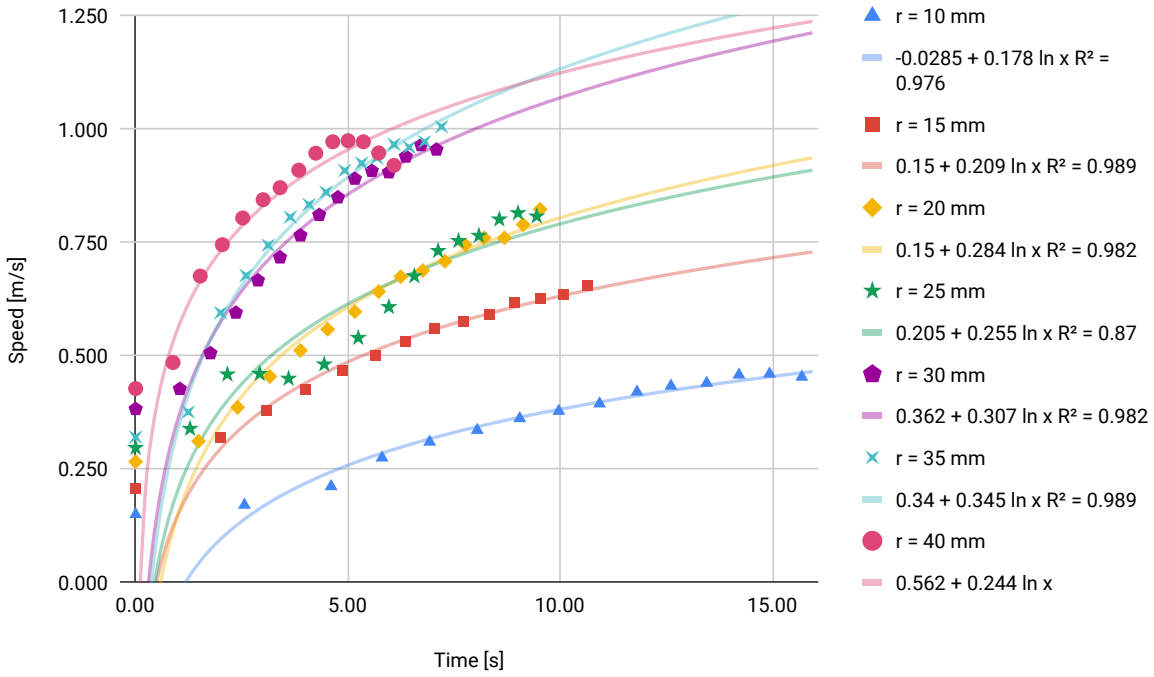


Figure 9.7: Speed of the cylinder versus time for different mass' radii with logarithmic trendlines.

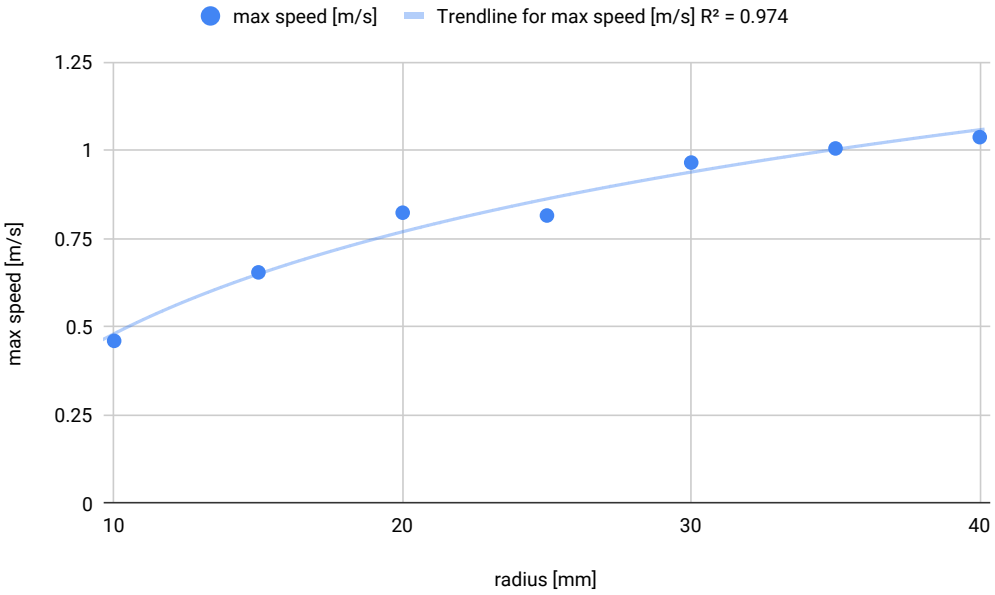


Figure 9.8: Maximal speed of the cylinder versus mass' radii with a logarithmic trendline.

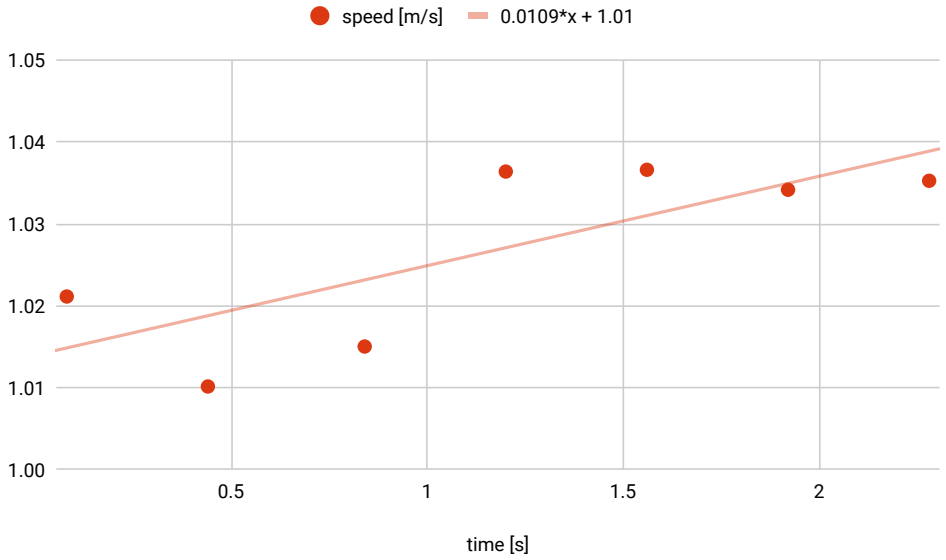


Figure 9.9: Position and speed of the cylinder versus time once the speed has stabilized, the mass moves on the maximal radius (40 mm).

9.3 Physical properties

CylinderPrototype3 has got a diameter of 32 cm for a width of 15 cm with electronics. Yet, the two PMMA rings that form the base of the cylinder are only 12 cm apart. This distance is still sufficient to achieve a good stability. It uses three high torque, metal-geared servos motors on which custom 5 cm long arms are fixed. This allows a maximal displacement of the mass from the center of the cylinder of approximately 40 mm with a precision of 2 mm. On the contrary of previous prototypes, the electronic connections are reliable since instead of using a breadboard to connect components, a dedicated PCB is used. This allows to use the cylinder for longer periods of time. The total mass of the cylinder with electronics and batteries included is around 1380 g whereas the mass in the center itself weights 150 g.

In general, the cylinder is really designed to roll on hard and really flat surfaces, but it can also roll on softer and more irregular ones, like grass. The problem, however is that it could not be used on concrete, for example, because the hard and small irregularities could cause cracks in the acrylic glass rings. To avoid that, protective rubber circles could be added around the rings. Also, we tried to make it roll up slopes, but it does only work when the slope is less than 3 degrees. When the slope is exactly 3 degrees, the cylinder remains balanced.

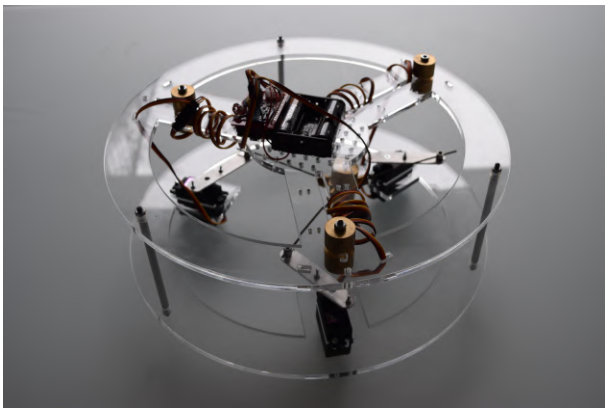


Figure 9.10: Last cylinder prototype.

9.4 Power consumption and autonomy

The final prototype is powered by Three AA Eneloop batteries, that have a capacity of at least 2450 mAh. This allows to have an output tension of 3.6 V when the batteries are full. Yet, the normal tension required to power up servo motors it 5 V. Therefore, the mass in the center could still be a lot heavier if the power supply was changed. However, the advantages of AA batteries are that they are easy to find, affordable and rechargeable. The cylinder's current consumption varies hugely if the servos move or not. Indeed, at rest, the consumption is of about 0.25 A whereas it rises up to 0.67 A when the three servos move. From these values, we can calculate that the time the cylinder can remain in standby is of 9.8 h (ready to move) and be moving for 3.7 h. Practically, the cylinder was kept on standby for one and a half hours.

9.5 Control

They are two different ways to control the cylinder. The first one is from the command line while having established an `ssh` connection. That method has mainly been used to implement new software and to debug, but it could still be used for regular control. The advantage of this method is that they are more programs available than with the control through the webpage, even though it is more complex to use. The second way to control the cylinder is through the dedicated webpage. When using that method, both the cylinder and the device from which the control is done must be connected to a WiFi rooter called "chemexper". In normal conditions, the cylinder connects automatically to the right WiFi rooter. Once that is done, the user must type "192.168.1.11" in a web browser and the webpage appears. From there, they are three buttons that allows different functions.

The first one allows to make the cylinder roll at different speeds, backwards and forwards, using a slider. That feature works really well since there is nearly no perceptible delay when changing the speed or the direction (except the one due to the inertia of the cylinder). The second feature is really simple, since it just places the mass in the center of the cylinder. The third one, on another hand, is supposed to allow the cylinder to remain stable on a gentle slope. Yet, this does not work very well. First of all, the cylinder has to be initially placed on the slope at a specific position. Then, it does indeed not roll down the slope, but it quickly starts to oscillate. Once it does, the oscillation increases until the cylinder finally rolls down. Therefore, that functionality would still need to be enhanced, for instance by implementing a PID controller [1].
In general, the web control is still very satisfying, because it is really easy. Indeed, various people have tested it and all managed to use it with hardly no explanations.

9.6 Price

Even though price was not a constrain in the choice of materials, it has still been taken into account. Indeed, to pursue in our open-source approach, it seemed important to have an accessible price. Therefore, the final prototype has been only built with materials that can be found in a hardware store and online, which should allow anyone to make his own cylinder fairly easily. Eventually, the cylinder's costs are around 69 CHF (see Table 9.1 for details), which is really low. The highest expense is due to the servo motors and is therefore not easy to reduce. Yet, only using MDF instead of PMMA for the rings would already allow saving around 8 CHF.

Part	Price per unit	Quantity needed	Actual price
Brass rod	15 CHF/kg	372 g	5.6 CHF
PMMA	60 CHF/m^2	0.18 m^2	10.8 CHF
Threaded rod	3.1 CHF/m	75 cm	2.3 CHF
Aluminium profile	3.3 CHF/m	16 cm	0.5 CHF
Aluminium tube	3.2 CHF/2m	33 cm	0.5 CHF
Servo motor	8.5 CHF/each	3	25.5 CHF
C.H.I.P. Pro	9 CHF/each	1 g	9 CHF
Extension board	25 CHF/10	1	2.5 CHF
Accelerometer	1.4 CHF/each	1	1.4 CHF
Servo driver	2.1 CHF/each	1	2.1 CHF
Battery holder	0.5 CHF/each	1	0.5 CHF
AA batteries	11 CHF/4	3	8.3 CHF
Total price			**69.0 CHF**

Table 9.1: Price of all the different parts of cp3 ant total price.

[1] "PID (proportional integral derivative) controllers use a control loop feedback mechanism to control process variables and are the most accurate and stable controller.[47]"

Chapter 10

Discussion and conclusion

Since this project has been realized on a long period of time and is very diversified, they are many aspects that need to be at least briefly discussed. However, the more specific critics have been done independently in every chapter, to make it more understandable. So, in this section, we will try to be critical towards the final result of our work.
In the end, we obtained a prototype that includes all of the initially desired properties. First of all, cylinderPrototype3 is well designed and well balanced and has therefore a movement that appears to be uniformly accelerated. The materials that were used to build it are optimized on many aspects: their light weight, which reduces the cylinder's inertia, they are waterproof, resistant and aesthetic. The brass masses that we added on it to allow to equilibrate the cylinder after the electronics were glued on it were efficient, but could have been a lot lighter. Indeed, the compensation was a lot smaller than expected, because the batteries were placed nearly exactly on the center of the cylinder, so the torque their mass produced was small. Also, the technology that we used to move the mass, with three servo motors, was a good approach because it worked well and provided a sufficient acceleration to the cylinder to be interesting. The magnitude of the acceleration allows an intuitive and responsive control. In addition, the development of a dedicated PCB that combined an accelerometer and a driver for four servos allowed to reduce considerably the mass and size of the electronics. The MCU that we used, a C.H.I.P. Pro, was also practical for its power, memory and speed.

Furthermore, the power supply, three AA batteries, appeared to be satisfactory, since it allowed to have an autonomy of more than one hour with the cylinder constantly turned on and ready to be controlled. That aspect also shows that it is really reliable and that the energy consumption is reasonable. We did also manage to provide various possibilities for the cylinder control, all of them using WiFi. First, the scripts can simply be run from a bash using the files names and command line parameters. Finally, the cylinder can also be controlled remotely from a web page that was developed for this usage. In that case, they are three possible options: the radius of the circle the mass moves on can be varied to modify its acceleration, the mass can be placed in the center and finally, a script can maintain the cylinder balanced on a gentle slope. The graphical interface for the control is very satisfying, because it makes the control easy and accessible to anyone.

Perspectives

Obviously, the initial goal we wanted to achieve during this whole year, was to make this project as general as possible. This includes, for instance, splitting the code into little reusable chunks and placing four footprints for the servos on the PCB that we developed. This generalization aims to forecast the possibility to apply the technologies that we invented to a sphere, which would be a lot more useful product than a cylinder. Indeed, the ability to control the movement of the cylinder on a straight line is already interesting, but making a sphere would allow to turn left and right. Therefore, it would be possible to move the sphere to any desired position that is covered by a single WiFi hotspot. This feature would offer a huge amount of applications for this product. Indeed, regarding this work, the question shall be raised: apart from a purely educative purpose, what could be done with the concept that has been developed?

To examine one example, it has already been considered to use spherical wheels in the car industry, as explained in the Touring magazine of April 2017. In a short article, it is explained that the tire company Goodyear has envisaged the production of spherical car wheels with incorporated artificial intelligence, that would allow to adapt its outer structure to the state of the road (whether it is wet, dry, uneven, etc.). In the concept of this company, the wheels would be maintained with magnetic suspension and they would move using individual electric motors [5]. So, friction with the rest of the car will be considerably diminished, but in all, the main advantage of these wheels is that they would be extremely maneuverable since they are in principle free to rotate in any direction. Another advantage is that the parking could be made sideways, which might optimize parking surfaces [19]. However, this idea is very theoretical and it is unlikely that it will ever be concretised.

Therefore, if it was demonstrated that our technology can really be used in a sphere, it would be innovative and have advantages that other methods do not have. For instance, our concept is easy, low-cost, efficient, reliable and overall, it offers the possibility to place the mass that is displaced in the center of the sphere, which allows to let it roll freely with minimal energy costs. Also, the whole sphere would only be made of one part, since all the components are connected together. So, the relative position between the outer shell is always known, because it is not possible for the shell to slip under the wheels of the drive mechanism, like in a standard approach.

Looking further, other applications could be considered. For instance, the cylinder's concept could be extended to transportation. Indeed, a cylinder with a diameter of approximately three meters could be used to transport one person linearly. The idea being to take advantage of the person's weight and to use it as the mass in the center of the cylinder. In that case, six motors could be used, placed on both rings of the cylinder to allow more power and stability (see Figure 10.1).

Another application of the cylinder could be to connect two of them with a rigid bar and ball bearings, which would allow to move three dimensionally depending on the individual movements of the cylinders (refer to Figure 10.2). Indeed, rotating one cylinder clockwise and the other the other way around would make the whole system turn left or right. If

that principle was extended, it could even be used instead of the regular Segway[1] wheels. Since the Segway is basically an inverted pendulum [33], using that technology would allow to easily compensate the torque produced by the driver to keep the vehicle stable (see Figure 10.3).

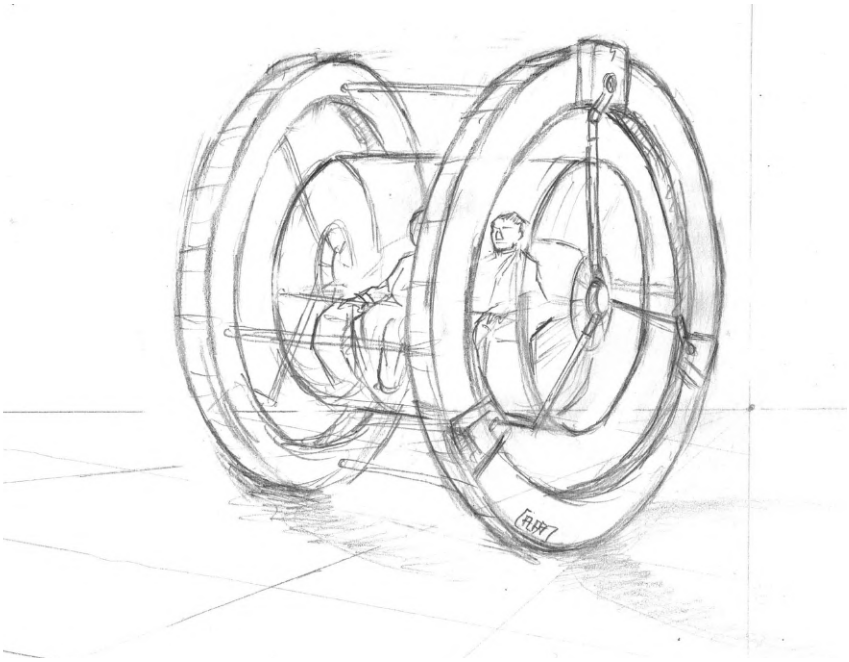


Figure 10.1: Extending the cylinder's concept to a personal transporter (drawing by Heloïse Piguet).

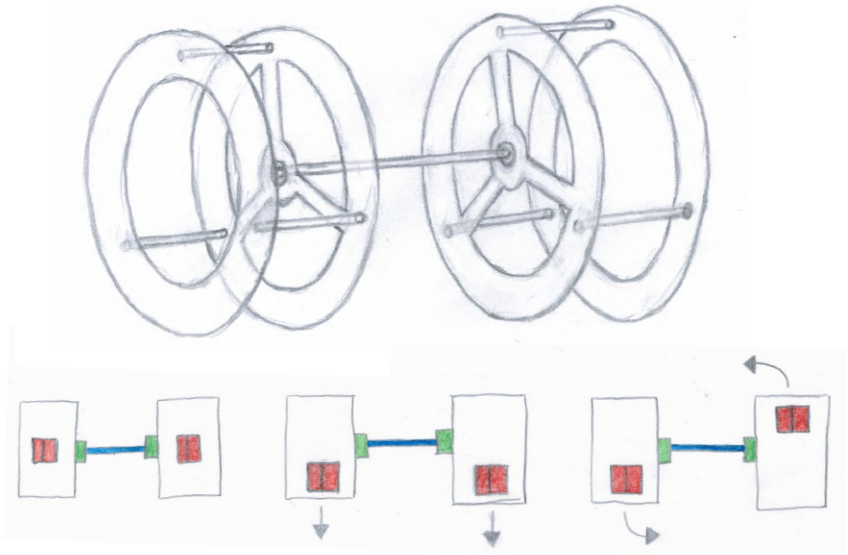


Figure 10.2: Connecting two cylinders together would allow to move on a plane. In blue: rigid bar, in green: ball bearings, in red: the masses in the center of the cylinders.

[1] "The Segway PT (originally Segway HT) is a two-wheeled, self-balancing personal transporter by Segway Inc. Invented by Dean Kamen and brought to market in 2001.[50]"

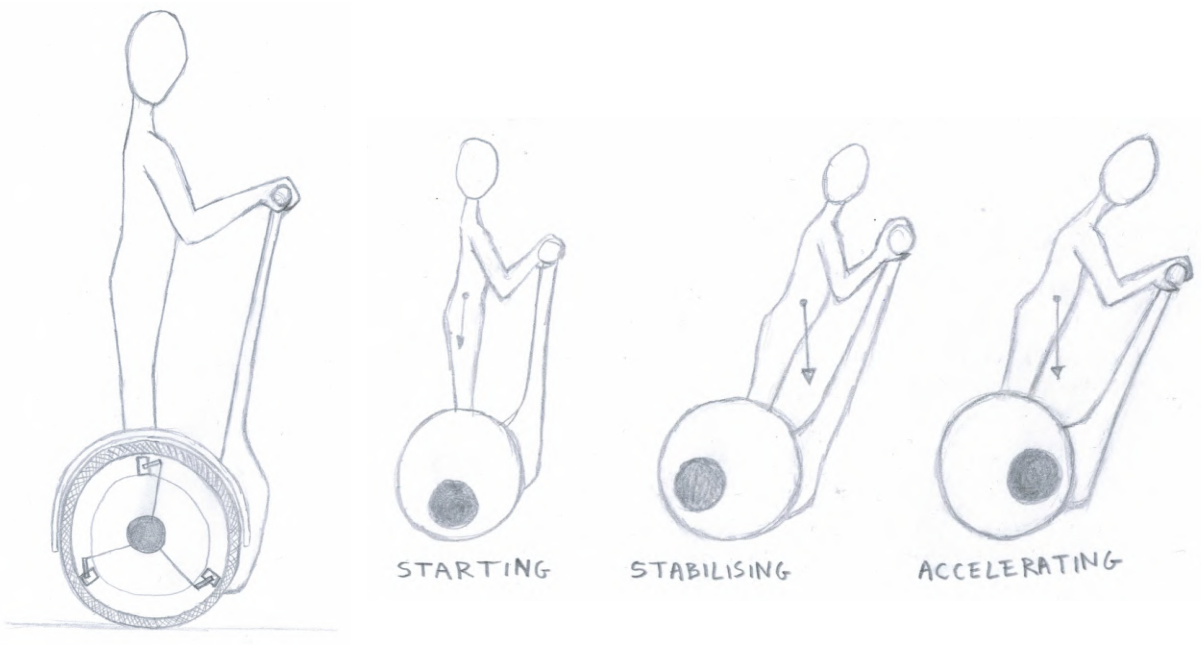


Figure 10.3: Replacing the regular Segway wheels by two cylinders. The masses can be displaced either to stabilize the person by compensating the torque they produce or to accelerate.

Appendix

Links to videos related to the project

The Wheel experiment results:
https://youtu.be/ovayvtKzIYs

The cardboard model in movement:
https://youtu.be/4SuPhpwOK5g

First tests aiming to move the mass of cylinderPrototype2:
https://youtu.be/1zOCNZ_r5pY

To see one of the first tests we did using the C.H.I.P. to make servos sweep, a system with the C.H.I.P., the accelerometer and the servo driver and measures of the servos energy consumption:
https://youtu.be/h4J0GEFvUUs

First test with cylinderPrototype2 moving:
https://youtu.be/H_8SvWPwM_c

Physical experimentation of the cylinder principle:
https://youtu.be/Q-3C4qCMcLs

Problems with cylinderPrototype3 and when it works:
https://youtu.be/Wetc8O-wjYI

Slow motion of cylinderPrototype3:
https://youtu.be/c1Su1ZOGJ0o

Time lapse showing how cylinderPrototype3 is built:
https://youtu.be/vpDdWu5c0N0

Final recapitulating movie of the last prototype:
https://youtu.be/QggOjoGjJhA

Bibliography

Articles

[1] Richard Chase and Abhilash Pandya. "A Review of Active Mechanical Driving Principles of Spherical Robots". In: *Robotics* 1.1 (Nov. 2012), pp. 3–23. ISSN: 2218-6581. DOI: 10.3390/robotics1010003. URL: http://www.mdpi.com/2218-6581/1/1/3/.

[2] Yury L. Karavaev and Alexander A. Kilin. "The dynamics and control of a spherical robot with an internal omniwheel platform". In: *Regular and Chaotic Dynamics* 20.2 (Mar. 2015), pp. 134–152. ISSN: 1560-3547. DOI: 10.1134/S1560354715020033. URL: http://link.springer.com/10.1134/S1560354715020033.

[3] Gajamohan Mohanarajah et al. "The Cloud, Paper Planes, and the Cube". In: (1980). URL: http://e-collection.library.ethz.ch/eserv/eth:47041/eth-47041-02.pdf.

[4] Adafruit learning system. "Introducing the Raspberry Pi Zero". In: (2017). URL: https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-zero.pdf.

[5] Touring Club Switzerland. "La roue réinventée, p.48 left column". In: *Magazine de la mobilité - touring* (Apr. 2017), p. 92. URL: https://issuu.com/touring-online/docs/inet%7B%5C_%7Dtouring%7B%5C_%7D2017%7B%5C_%7Df%7B%5C_%7D04/49.

Books

[6] Earl William Swokowski. *Calculus (The Classic Edition).*

Images

[7] Atelier du Grand Morin. *Crémaillères et engrenanges, des produits pour l'industrie.* 2017. URL: http://www.grandmorin.fr/produits/ (visited on 06/28/2017).

[8] Hobby and You. *TowerPro MG996R Metal Gear Servo MotorA Complete Robotic Hobby Store.* URL: http://www.hobbyandyou.com/towerpro-mg996r-metal-gear-servo-motor (visited on 10/13/2017).

[9] *Inventables Blog: Stepper Motors.* 2017. URL: http://blog.inventables.com/p/stepper-motors.html (visited on 06/26/2017).

[10] Museum of Science in Boston and Northeastern University. *Electromagnets | Magic of magnetism*. URL: http://www.ece.neu.edu/fac-ece/nian/mom/electromagnets.html (visited on 02/22/2017).

[11] *Next Thing Co. C.H.I.P. Pro Documentation*. URL: https://docs.getchip.com/chip%7B%5C_%7Dpro.html%7B%5C#%7Dget-started-with-c-h-i-p-pro (visited on 07/10/2017).

[12] Sphero. *(14) What is Sphero SPRK Edition? - YouTube*. 2015. URL: https://www.youtube.com/watch?v=Yg8LmEkI%7B%5C_%7D0c (visited on 06/23/2017).

[13] *Szanghajscy Akrobaci w Szanghaj, zdjęcia, godziny otwarcia, cena biletu wstępu, dojazd i położenie*. URL: http://www.transazja.pl/atlas/1796236/rozrywka/21,64/Szanghajscy%7B%5C_%7DAkrobaci (visited on 05/20/2017).

Screenshots

[14] Adafruit. *PCA9685 - product data sheet*. 2015. URL: https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf.

[15] InvenSense Inc. *MPU-6000 and MPU-6050 Product Specification Revision 3.4 MPU-6000/MPU-6050 Product Specification*. 2013. URL: www.invensense.com%20https://store.invensense.com/datasheets/invensense/MPU-6050%7B%5C_%7DDataSheet%7B%5C_%7DV3%204.pdf.

[16] Waibel Markus Mohanarajah Gajan. *Cubli – A cube that can jump up, balance, and walk across your desk | Robohub*. 2013. URL: http://robohub.org/swiss-robots-cubli-a-cube-that-can-jump-up-balance-and-walk-across-your-desk/ (visited on 05/03/2017).

[17] *TowerPro MG995R Servo Specifications and Reviews*, p. 1. URL: https://servodatabase.com/servo/towerpro/mg995r (visited on 10/01/2017).

Videos

[18] *(14) How Wireless Energy Transfer Works - YouTube*. 2012. URL: https://www.youtube.com/watch?v=-Wf7aadxBkE (visited on 06/26/2017).

[19] *The Future Tire by Goodyear - It's a Sphere! - YouTube*. 2016. URL: https://www.youtube.com/watch?v=RHpxuwcNJfo (visited on 10/09/2017).

Web references

[20] Adafruit. *Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface [PCA9685] ID: 815 - $14.95 : Adafruit Industries, Unique & fun DIY electronics and kits*. URL: https://www.adafruit.com/product/815 (visited on 09/13/2017).

[21] Adafruit. *Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface [PCA9685] ID: 815 - $14.95 : Adafruit Industries, Unique & fun DIY electronics and kits*. URL: https://www.adafruit.com/product/815 (visited on 09/13/2017).

[22] *Arduino - Home*. URL: https://www.arduino.cc/ (visited on 10/13/2017).

[23] Caelectronics8. *MPU-6050 6DOF 3 Axis Gyroscope+Accelerometer Module for Arduino DIY | eBay*. 2017. URL: http://www.ebay.com/itm/MPU-6050-6DOF-3-Axis-Gyroscope-Accelerometer-Module-for-Arduino-DIY-/141976854044?hash=item210e7ade1c:g:OSwAAOSwDwtUm4WP (visited on 10/13/2017).

[24] *Car Vehicle Motorcycle GSM GPS Tracker Locator Global Real Time Tracking Device | eBay*. URL: http://www.ebay.com/itm/Car-Vehicle-Motorcycle-GSM-GPS-Tracker-Locator-Global-Real-Time-Tracking-Device-/232330069435?epid=853903004%7B%5C&%7Dhash=item3617f385bb:g:E5kAAOSwtGlZEst%7B~%7D%7B%5C&%7Dvxp=mtr (visited on 09/03/2017).

[25] Chivazhu. *PCA9685 16 Channel 12-Bit PWM Servo Motor Driver IIC Module For Arduino Robot | eBay*. URL: http://www.ebay.com/itm/PCA9685-16-Channel-12-Bit-PWM-Servo-Motor-Driver-IIC-Module-For-Arduino-Robot-/201536578341?epid=526951286%7B%5C&%7Dhash=item2eec842325:g:csQAAOSwu1VW2WOp (visited on 10/13/2017).

[26] Collins English Dictionnary. *MDF definition and meaning | Collins English Dictionary*. URL: https://www.collinsdictionary.com/dictionary/english/mdf (visited on 10/09/2017).

[27] Curbell Plastics. *Polycarbonate Plastic | Clear, Impact Resistant Material | Curbell Plastics*. URL: https://www.curbellplastics.com/Research-Solutions/Materials/Polycarbonate (visited on 10/09/2017).

[28] Eagle. *Download EAGLE | Free Download | Autodesk*. URL: https://www.autodesk.com/products/eagle/free-download (visited on 10/19/2017).

[29] Vim the editor. *welcome home : vim online*. URL: http://www.vim.org/ (visited on 10/19/2017).

[30] Eglowstein Howard. *Introduction to Servo Motors*. URL: https://www.sciencebuddies.org/science-fair-projects/references/introduction-to-servo-motors (visited on 10/17/2017).

[31] *Epoch Converter - Unix Timestamp Converter*. URL: https://www.epochconverter.com/ (visited on 11/05/2017).

[32] *Express - Node.js web application framework*. URL: http://expressjs.com/ (visited on 09/22/2017).

[33] Hachman Mark. *The Technology Behind The Segway - ExtremeTech*. 2001. URL: https://www.extremetech.com/extreme/72040-the-technology-behind-the-segway (visited on 04/06/2017).

[34] *I2C - What's That? - I2C Bus*. URL: https://www.i2c-bus.org/ (visited on 10/28/2017).

[35] *JavaScript Robotics: IMU - MPU6050 with Johnny-Five*. URL: http://johnny-five.io/examples/imu-mpu6050/ (visited on 10/28/2017).

[36] *JavaScript Robotics: Servo - PCA9685 with Johnny-Five*. URL: http://johnny-five.io/examples/servo-PCA9685/ (visited on 10/28/2017).

[37] *Johnny-Five: The JavaScript Robotics & IoT Platform*. URL: http://johnny-five.io/ (visited on 10/28/2017).

[38] Laser Spine Insitute. *PMMA (polymethyl Methacrylate) Definition | Laser Spine Institute*. URL: https://www.laserspineinstitute.com/learn%7B%5C_%7Dmore/glossary/definition/pmma%7B%5C_%7Dpolymethyl%7B%5C_%7Dmethacrylate/97/ (visited on 10/20/2017).

[39] Lawton Stephen. *Flash Memory Explained: MLC vs. eMLC vs. SLC vs. TLC*. URL: http://www.tomsitpro.com/articles/flash-data-center-advantages,2-744-2.html (visited on 10/17/2017).

[40] *Logger Pro® | Vernier*. URL: https://www.vernier.com/products/software/lp/ (visited on 03/20/2019).

[41] *Mini GPS Tracker Pet Collar Real Time Locator Kid Cat Dog 2G SIM Tracking Device | eBay*. URL: http://www.ebay.com/itm/Mini-GPS-Tracker-Pet-Collar-Real-Time-Locator-Kid-Cat-Dog-2G-SIM-Tracking-Device-/292219987962?hash=item4409ab27fa:g:sxIAAOSwcUFZl97d (visited on 09/03/2017).

[42] *MPU-6000 and MPU-6050 Product Specification Revision 3.4 MPU-6000/MPU-6050 Product Specification*. 2013. URL: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/.

[43] Next Thing Co. *Next Thing Co. C.H.I.P. Pro Documentation*. URL: https://docs.getchip.com/chip%7B%5C_%7Dpro.html%7B%5C#%7Dget-started-with-c-h-i-p-pro (visited on 07/10/2017).

[44] *Node.js main page*. URL: https://nodejs.org/en/ (visited on 10/28/2017).

[45] *npm main page*. URL: https://www.npmjs.com/ (visited on 10/28/2017).

[46] ORACLE. *What Is a URL? (The Java^{TM} Tutorials > Custom Networking > Working with URLs)*. URL: https://docs.oracle.com/javase/tutorial/networking/urls/definition.html (visited on 09/18/2017).

[47] *PID Controller: Types, What It Is & How It Works | Omega*. URL: https://www.omega.com/prodinfo/pid-controllers.html (visited on 03/25/2019).

[48] *Promise - JavaScript | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global%7B%5C_%7DObjects/Promise (visited on 10/28/2017).

[49] *Realtime GPS GPRS GSM Tracker For Car/Vehicle/Motorcycle Tracking Device Spy NEW*. URL: http://www.ebay.com/itm/Realtime-GPS-GPRS-GSM-Tracker-For-Car-Vehicle-Motorcycle-Tracking-Device-Spy-NEW-/112095734729?hash=item1a196d2bc9:g:hJ4AAOSwMtxXsRza (visited on 09/03/2017).

[50] *Segway*. URL: https://en.wikipedia.org/wiki/Segway (visited on 03/16/2019).

[51] The Engineering ToolBox. *Metals and Alloys - Densities*. URL: https://www.engineeringtoolbox.com/metal-alloys-densities-d%7B%5C_%7D50.html (visited on 10/19/2017).

[52] The International Maglevboard. *Maglev - What is it? And is it relevant?* URL: http://www.maglevboard.net/en/facts (visited on 06/28/2017).

[53] Tutorialspoint. *HTML5 - WebSockets*. URL: https://www.tutorialspoint.com/html5/html5%7B%5C_%7Dwebsocket.htm (visited on 09/18/2017).

[54] *Universal Bluetooth GPS Receiver for Today's Most Popular Handheld Devices | Dual Electronics*. URL: http://gps.dualav.com/explore-by-product/xgps150a/ (visited on 09/03/2017).

[55] Upton Eben. *Raspberry Pi Zero: the $5 computer - Raspberry Pi*. 2015. URL: https://www.raspberrypi.org/blog/raspberry-pi-zero/ (visited on 10/17/2017).

[56] *View Exif online, remove Exif online*. URL: http://www.verexif.com/en/index.php?error=1 (visited on 09/30/2017).

[57] *WebStorm: The Smartest JavaScript IDE by JetBrains*. URL: https://www.jetbrains.com/webstorm/ (visited on 10/29/2017).

[58] *What is HTTP (HyperText Transfer Protocol)?* 2017. URL: https://www.computerhope.com/jargon/h/http.htm (visited on 10/29/2017).

[59] *Wireless Charging & How Inductive Chargers Work • PowerbyProxi*. URL: https://powerbyproxi.com/wireless-charging/ (visited on 06/26/2017).

Other references

[60] A. Halme, T. Schonberg, and Yan Wang. "Motion control of a spherical mobile robot". In: *Proceedings of 4th IEEE International Workshop on Advanced Motion Control - AMC '96 - MIE*. Vol. 1. IEEE, 1996, pp. 259–264. ISBN: 0-7803-3219-9. DOI: 10.1109/AMC.1996.509415. URL: http://ieeexplore.ieee.org/document/509415/.

[61] R. Mukherjee, M.A. Minor, and J.T. Pukrushpan. "Simple motion planning strategies for spherobot: a spherical mobile robot". In: *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*. Vol. 3. IEEE, 1999, pp. 2132–2137. ISBN: 0-7803-5250-5. DOI: 10.1109/CDC.1999.831235. URL: http://ieeexplore.ieee.org/document/831235/.

List of Figures

2.1 Cartesian frame of reference 9
2.2 Close view of the disk: the hole in the center and the arrays of holes are visible. The disk has a radius of 28 cm and the smallest holes have a diameter of 3 mm. 13
2.3 The part allowing the disk to rotate on the steel axis. 13
2.4 The base or support of the Wheel, the rotation axis and the wooden disk. 14
2.5 Three symmetry cases that were tested: segment, equilateral triangle and square. 14
2.6 The system and the frame of reference considered in the problem. 16
2.7 Scheme of the two bodies m and M 17

3.1 Second version of the Sphero, the wheels that make it move can be seen in white. [12] 23
3.2 The Cubli balancing on it's side, on a slope. [16] 24
3.3 The plates balanced by gymnasts are inverted pendulums. [13] 25
3.4 Inner components of the Cubli, the three reaction wheels are in yellow. [16] 25

4.1 A stepper motor. [9] 29
4.2 Approach with one stepper in the center. 30
4.3 Explanation of the "pulley" system. 31
4.4 Approach with three "pulleys" in the center. 32
4.5 A standard rack and pinion driver. [7] 32
4.6 Approach with an invented technology similar to rack and pinion drives. 33
4.7 Approach with three "long pistons" on the outside of the cylinder. 34
4.8 Approach with all the "short pistons" around of the external cylinder. . . 35
4.9 Scheme of an electromagnet. [10] 36
4.10 An electromagnet attracting metallic beads. 36
4.11 Approach with electromagnets on the outside of the cylinder. 37
4.12 Approach with three servos on the outside of the cylinder. 38
4.13 A standard servo motor. [8] 39
4.14 Specifications of MG995 servo motor (used in the second cylinder prototype). [17] 39
4.15 Current in [A] that a mini servo consumes if A, its arm is free and B, a force is applied to it. The consumption can be up to five and a half times more in case B. 40
4.16 Scheme of the chosen approach: using three servo motors. 40
4.17 The first model, made of cardboard. 41

5.1 A method to built an animated segment of constant length on GeoGebra. 44

5.2 GeoGebra model of the problem using the parameters of cylinderPrototype2, representing the case where $s = r$ and $d = R$. 45
5.3 One servo and the maximal circle that can be reached by m. Illustrating why Assertion 1 is true: if d is either bigger or smaller than R, some points of the red circle can not be reached. 46
5.4 Maximal $\frac{s}{R}$: when $R = r$. Also, r is slightly smaller than s, so that all the values of theta could be represented in the animated model. Indeed, otherwise, GeoGebra would not be able to draw some of the red and green segments. 47
5.5 Values corresponding to the second cylinder prototype we developed. $R =$ 150 [mm], $d = 109.2$ [mm], $s = 16.2$ [mm]. In addition, $r = s$. 48
5.6 Values of alpha obtained when R and r are varied. 49
5.7 Values obtained when R is constant and r is varied. 49
5.8 Values obtained when s is constant and R is varied. Also, $r = s$. We can see that when $r = R$, the servos maximal speed is infinite. 50
5.9 The mass that is moved and one servo: The red point corresponds to the mass in the cylinder (m), the light red area represents all the possible positions of m, the blue point is the end of the servo's arm. 52
5.10 Illustration of Hypothesis 2. One position of m has been represented for every quadrant, as well as the corresponding Q and Q'. 55
5.11 To find α, we use the rectangular triangle with sides b, $R - a$ and s. . . . 56
5.12 Scheme of one servo when the mass is in the center (on point G). 56
5.13 The dsR triangle can be divided into two rectangular triangles. 57
5.14 Testing Equation 5.26 with a rectangular triangle. 58

6.1 Cutting materials with a laser cutter: bias problems. 59
6.2 FreeCad drawing of cp1 base. 60
6.3 Information about cp1. 61
6.4 Information about cp2. 62
6.5 cp2 autonomous with the electronics attached to the cardboard piece. . . 63
6.6 FreeCad drawing of cp3 base: front (left) and back (right) pieces are different. 64
6.7 Close view of the center mass pieces. 64
6.8 The different parts that connect the servo to the mass. 65
6.9 The servo / mass connection all built. 65
6.10 Information about cp3. 66

7.1 Arduino Pro Micro was used on a breadboard for initial tests of the mass movement. 67
7.2 Dimensions of the C.H.I.P. Pro [11]. 69
7.3 Pin mapping of the C.H.I.P. Pro [11]. 69
7.4 Pin mapping of MPU6050 [15] 71
7.5 MPU6050 gyroscope breakout board 71
7.6 Pin mapping of PCA9685 [14] 72
7.7 16 channels servo driver based on PCA9685 72
7.8 Electronics of cylinderPrototype2 during test phase. 73
7.9 Final eagle file and actual board (34x50 mm). 74
7.10 Completely soldered PCB. 75

8.1 Node.js computing decimal numbers. 81

8.2 Servos calibration setup, on cylinderPrototype3. 83
8.3 Servos calibration setup, on cylinderPrototype3. 84
8.4 The final web page, when the 'Control' button is clicked. 85
8.5 Communication protocol overview: from user to cylinder prototype. . . . 87
8.6 Structure of code in the `webControl` folder and order in which the files are run. 87

9.1 Module used for long aperture pictures: button battery holder, SMD resistor and LED. 91
9.2 Long aperture picture of the mass moving on a circle when: left, increasing *theta* of 1 degree every 50 ms and right, increasing *theta* of 30 degrees every 50 ms. 92
9.3 Long aperture picture of the mass moving on a spiral from maximal to minimal possible radii (40 mm to 0 mm). 92
9.4 Screenshot of the movie with annotations. Blue points: measures, Green line: allows to set the scale, Yellow axes: sets origin. 93
9.5 Position of the cylinder versus time for different mass' radii with quadratic trendlines. 94
9.6 Speed of the cylinder versus time for different mass' radii with linear trendlines. 95
9.7 Speed of the cylinder versus time for different mass' radii with logarithmic trendlines. 95
9.8 Maximal speed of the cylinder versus mass' radii with a logarithmic trendline. 96
9.9 Position and speed of the cylinder versus time once the speed has stabilized, the mass moves on the maximal radius (40 mm). 96
9.10 Last cylinder prototype. 97

10.1 Extending the cylinder's concept to a personal transporter (drawing by Heloïse Piguet). 101
10.2 Connecting two cylinders together would allow to move on a plane. In blue: rigid bar, in green: ball bearings, in red: the masses in the center of the cylinders. 101
10.3 Replacing the regular Segway wheels by two cylinders. The masses can be displaced either to stabilize the person by compensating the torque they produce or to accelerate. 102

List of Tables

2.1 Properties of M and m. 16

5.1 Summary of the objects used in further calculation 51

7.1 Comparison of Raspberry Pi Zero W and C.H.I.P. Pro features. 68

8.1 Comparing values of α from the JavaScript formula and from the GeoGebra model. 81

9.1 Price of all the different parts of cp3 ant total price. 98

Index

2D model, 43

Accelerometer, 70
Angular acceleration, 11
Angular speed, 11
Arduino, 67

bash, 67, 77
Breadboard, 73

debug, 78
delays, 78

Eagle, 73
Electromagnets, 36
epoch, 85

GeoGebra, 43
GitHub, 43
GPS, 70
Ground plane, 74
Gyroscope, 70

HTML, 67

I2C, 70, 74
Induction charging, 24
Inverted pendulum, 25
IP address, 77, 86

JavaScript, 67, 78

LED, 74
Linux, 73

Mathematics, 51
MCU, 77
MDF, 73

Node.js, 78
npm, 78, 86

PCB, 74
Peripheral, 70
Physical resolution, 16
Plexiglass, 63
plexiglass, 66
PMMA, 63
Polycarbonate, 66
polycarbonate, 23
Power supply, 75
PWM, 68, 71

RAM, 68
Reaction wheels, 25
Rigid body, 11

SD, 68
servomotor, 39
servomotors, 37
Servos maximal speed, 50
Software, 77
Soldering, 74
stepper motor, 29

Torque, 16, 26
Translation acceleration, 10
Translation speed, 10

Unit axis system, 9
URL, 85

vim, 68

WebSockets, 86

Acronyms

HTTP Hyper text transfer protocol. 86

I²C Inter-Integrated Circuit. 37

IDE Integrated Development Environment. 77

IP address Internet Protocol address. 86

JS JavaScript. 78

MCU Micro Controller Unit. 73

MDF Medium-Density Fibreboard. 60

MDN Mozilla Developer Network. 86

MLC Multiple Level Cell. 68

npm node package manager. 78

PCB Printed Control Board. 75

PID Proportional Integral Derivative. 98

PMMA Polymethyl methacrylate. 63, 66

PWM Pulse Width Modulation. 37

RAM Real Access Memory. 68

SD Card Secure Digital Card. 68

SLC Single Level Cell. 68

SMD Surface Mounted Design. 91

stepper Stepper motor, motor that moves from a given angle at every electric impulse it receives. 29

WS WebSockets. 87