UCLouvain

École polytechnique de Louvain

In-flight hand: a preliminary study for puppetry with a drone

Author: **Martin D'URSEL**
Supervisor: **Pierre LATTEUR**
Readers: **Renaud RONSSE, Martin STEINMETZ, Julia ZAMBONI**
Academic year 2021–2022
Master [120] in Electro-mechanical Engineering

Abstract

This master's thesis aims to lay the first steps towards the preparation of a live show where drones would lift the animated limbs of different characters. This project in art and technology is part of Julia Zamboni's ongoing research on the semblance of life in machines. The goal of the project is to develop a flying arm system, with a robotic arm serving as the "puppet". This work presents the construction process, and future automation prospects. First, the robotic arm will be developed using rapid prototyping techniques. Afterwards, a quadcopter will be designed to lift this arm, and the compatibility of it's movements with the arm's will be reviewed. Following that, the ground work for automation of this drone will be laid by developing a low-level controller suitable for the unusual profile of the drone-arm pair. Finally, it's viability will be validated trough a Simulink simulation.

Acknowledgements

I would like to personally thank:

Pierre Latteur for the supervision and the fruitful guidance during the whole duration of this project;

Julia Zamboni for her continuous help, guidance and advice, as well as for the suggestion of this work;

Benoît Herman for the help in the finger design, the access to the 3D printers and the guidance on their use;

Thierry Daras for the electronic material lent;

Gaëtan Permentiers for his collaboration, friendly words and enthusiasm during our work together;

Michaël Daris for the realisation of the drone;

Renaud Ronsse for the helpful guidance on prosthetic hands;

My family and friends for their unwavering support;

And finally, **every reader of this thesis** for their time and attention;

Contents

1 Introduction **5**
- 1.1 Character Creation: El Captain . . . 6
- 1.2 Project's goals . . . 7
  - 1.2.1 Requirements for the arm . . . 8
  - 1.2.2 Requirements for the drone . . . 8
- 1.3 State-of-the-art . . . 8
  - 1.3.1 PuppetCopter . . . 8
  - 1.3.2 Verity studios . . . 10
  - 1.3.3 Painter drones: Tusru robotics' "grafiti drone" and the PaintCopter . . . 11

2 Arm and hand **13**
- 2.1 Hand . . . 13
  - 2.1.1 First model . . . 13
  - 2.1.2 Pneumatic fingers . . . 15
  - 2.1.3 3D Printed fingers . . . 16
- 2.2 Actuators . . . 18
  - 2.2.1 For the fingers . . . 19
  - 2.2.2 For the elbow . . . 21
- 2.3 Electronics and battery . . . 24
  - 2.3.1 Battery dimensioning . . . 25
- 2.4 Assembly . . . 26
  - 2.4.1 First test . . . 26
  - 2.4.2 Improvements . . . 27
- 2.5 Final model . . . 28

3 Drone hardware **30**
- 3.1 Frame & battery . . . 30
- 3.2 Motors and Propellers . . . 31
  - 3.2.1 Noise considerations . . . 32
  - 3.2.2 Models used . . . 35
- 3.3 Electronics . . . 37
- 3.4 Characteristics . . . 38

4 Drone software, automation and simulation **40**
- 4.1 Control loop for a remote-controlled drone . . . 40
- 4.2 Control loop for autonomous flight . . . 42
  - 4.2.1 Qualisys motion capture . . . 43
- 4.3 Flight control system . . . 44

4.3.1 PID controllers . . . 45
4.3.2 Controller architecture . . . 46
4.3.3 Position and altitude controllers . . . 47
4.4 Internal drone control . . . 49
4.5 Simulation . . . 51
4.5.1 Choice of simulation software . . . 51
4.5.2 Modifications to the template . . . 53
4.5.3 Controllers tuning . . . 56

5 Results **58**
5.1 Arm animations . . . 58
5.2 Test flight . . . 61
5.3 Controller and simulation . . . 62

6 Conclusion **64**

Appendix **65**

A ESP32 code **67**

B Finger elastics **71**

C 3D models **73**

D PID Step responses and gains **75**

Chapter 1

Introduction

Art and technology often go hand in hand. From the early to modern days, artists looked for new ways to improve their arts and new uncharted territories to explore. Technology offers them just that. The inventions of color tubes revolutionized painting and led to the new genre of impressionism, electric instruments changed music forever, and computers drastically rewrote what art is today. This project is a collaboration between artists and engineering in exploring drones as a tool for the creation of puppetry in the skies. It involves the articulation of traditional puppetry and new technology to create a lifelike credible character.

The animation shows for which the results of this thesis are used are part of researcher Julia Zamboni's decade-long work of investigating how to create lifelike machines. The main challenge of this performance is to create a believable, engaging character with a drone. Making the device appear to have a will, thoughts, and emotions using techniques to bring them to life. Zamboni [1] proposes that artists and engineers working with lifelike machines should consider five main expressive elements that can be deployed to transform the perception of a device from lifeless to lifelike.

- The Body of the Machine, which can have varying levels of embodiment (from robotic to computational) and representation (from realistic to abstract).
- Its Behavior, which most often is movement and voice interaction. It also includes different levels of autonomy. Machines may be completely reliant on the human controller, or can be autonomous.
- The context, including the scenery, lighting, sound, and music
- And lastly, the device's name, which is a choice that can indicate the gender or the nationality of the character.

The current project involves the development and control of flying characters. It involves embodied systems in which drones lift mechanical systems with humanoid, stylized shapes. While art elements will be added to the scene, like scenery, light, and music, the technical construction of the puppet's body and its movements described in this work will have a crucial effect on making the performance lifelike and credible or if it fails, mechanical and artificial. The machine construction will affect its movement and general capacity to convince as make-believe.

Since there are few examples of research on this topic, the present is a preliminary investigation of drone puppetry departing from a simple experiment to test and understand the technical demands, strong points, and limitations of these art performers. The present case study starts with one drone, remote-controlled, performing in an internal, controlled environment. The drone carries a mechatronic arm with articulated fingers and forearm, which is also remote controlled with predefined sets of actions.

The study comprises the technical development of this arm-drone system. It solves issues related to the system's embodied features, including the propeller's airflow, drag, and lift forces; and how the puppets' shape, mechanical structures of the body, effectors, mass, weight, and material impact the drone's movements, trajectory, stability, and synchronization. It hopes to open the way for future research with more complex and autonomous flight systems including multiple drone and characters.

1.1 Character Creation: El Captain

The objective of this study is to develop and control the arm of El Captain, a pompous Spaniard soldier who pretends to be a hero and a lover but is a coward. His body is comprised of two machines, the flying hand with articulated fingers and forearm that is the subject of this work and a torso that is a human-controlled industrial aspirator (figure 1.1). The first drone puppet was made as a hand because hands are highly used in human communication. Simple actions like closing the fist, pointing a finger, or trembling nervously can convey emotions and be understood as meaningful actions, like playing, greeting, or threatening. A reasonably simple system of arms and fingers can be enough to, with other elements, create a believable, engaging character.



(a) Concept drawing (b) Realisation

Figure 1.1: El Captain: main body

The body of the character is build with a industrial aspirator. It has a long tube that looks like a neck, and light bulbs that look like eyes. It is controlled with a rod connected to the basis of the neck by a human. The rod has a cardboard shield attached to it, and the human's legs are also the character's legs. The name El Captain is taken from a character of the Commedia dell'arte.

As the show is non-verbal, the communication of the character will occur mainly through its movement. Therefore, the drone's trajectory coupled with the movement of the arm needs to achieve some level of fluidity, speed, and precision for the project to succeed. Moreover, the two distinct machines that compose the character need to be perceived as one entity by the spectator.

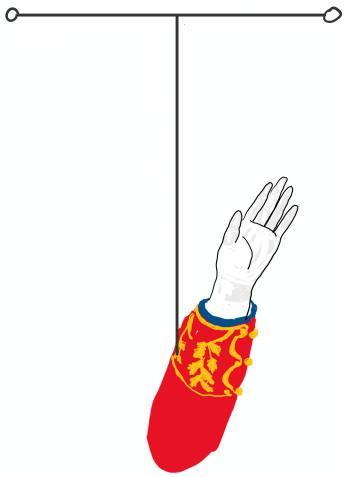


Figure 1.2: El Captain: hand concept drawing

Music will be used as a fundamental medium to make the meaning of the character's actions clearer to the audience. It will underline the character's emotions and bring forward its movements. Bombastic music when he is prideful, the light tone when he is happy, etc. The movements and the music complement each other. A trumpet will play the music because it has loud, well-defined messages, that ideally superpose the drones sounds. It also has a military reference that goes well with the character. The music creates the feeling of heroism, adventure, and sensuality, as the character sees himself. Ultimately, the results of this research will be evaluated based on these demos and the efficiency of the arm-drone performance, as described in the goals section below.

1.2 Project's goals

This goal of the project is to design and manufacture a mechanical arm for the character, with moving fingers and forearm, able to perform pre-recorded sequences of movements on command, attached to a remote-controlled drone. The arm-drone system must synchronize the drone's trajectory, fingers, and arm movements, and the movements and positions of the hand must be easily seen and recognized by the audience.

1.2.1 Requirements for the arm

The arm is mainly independent from the drone: It has his own battery, microprocessor and actuators, meaning he can be controlled separately.

- **Dynamism:** the arm must have smooth and natural-looking movements.
- **Robust:** The device must be stable and reliable during performance.
- **Easy to replace:** Parts must be easy to replace if worn down or break during the show.
- **Duration:** The battery must be able to run for at least thirty minutes of the show.
- **Automation:** Prepare the way for future automation, by realising a preliminary study including the arm, the drone and the captors. The main goal being to test the automation software and evaluate the controller needed.

1.2.2 Requirements for the drone

- **Payload:** The drone should be able to carry the arm as payload, which weights around 1kg. It does not need to be remarkably fast, but it needs to be strong and stable to carry the puppet.
- **Size:** Its diameter should be around the same size as the arm's length, and not much bigger than that, so that it does not take much attention from the arm.
- **Silent:** the drone should be as quiet as possible. For an indoor show of 30 to 40 minutes, the regular drone noise could quickly become overwhelming for the audience.
- **Versatile:** The drone needs to be easily modified to adapt to the project's future needs such as automation.
- **Safe:** The propellers should be protected.

1.3 State-of-the-art

This section will review some of the more proeminent use of drone in the artistic field. Indeed, the flexibility of UAV's allow their use in various artistic applications. They can be used directly as the focus of a performance or indirectly as a actuators for something else.

1.3.1 PuppetCopter

The PuppetCopter was developped as a focus project by a team of ETH Zurich student [2] and for Kevin Eggers' bachelor thesis [3]. The premise of this project is to control a string puppet with a drone. The concept is interesting: it is the first time that UAV's are used for puppetry, and Disney even brought and patented the concept [4] proving there is a potential future for such projects in entertainment parks. The project's puppet is totally 3D printed and has no actuators of its own: it is set in motions by strings pulled by modules attached to the drone, as shown in fig. 1.3

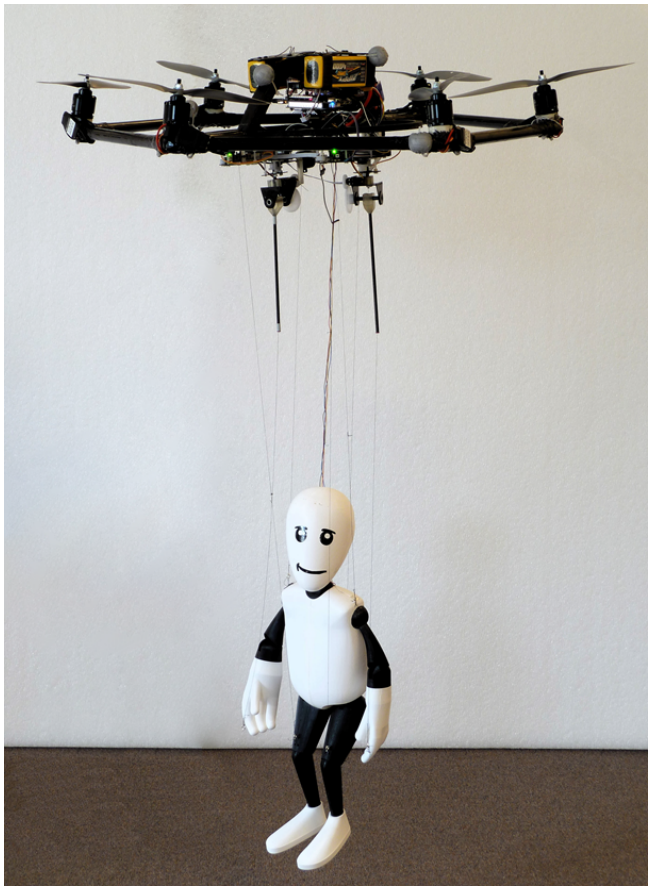


Figure 1.3: drone with puppet

Each string actuation module is composed of three degrees of freedom: Two servomotors control the θ and ϕ spherical coordinates of the end effector, while the string can be reeled up by another actuator to control its length.This method provides a nice way of moving a puppet with a drone. However the result are not lifelike: the actuation modules can not match the speed and dexterity of an actual puppeteer's hand, giving a slow and robotic result.

It is also interesting to note that Disney's patent [4] covers the case of a puppet actuated by a fleet of drones pulling one string each. This would enable puppets that are much bigger than the drones pulling them, or than a human could possibly control. Actuation modules would no longer be needed: by controlling the position of the drone with respect to the puppet one could generate the right directional traction needed in the string. The control would be automatic, thanks to a ground control system with a processor executing a fleet manager module storing a different flight plan for each of the UAV's in memory. For multicopters, each include a local controller operating to move it trough a series of way points defined by the flight plan generated by the ground control system. The idea, although patented, has never been tested so it remains to be seen how practical it really is, and how satisfactory the artistic results are in terms of movements and life likeness.

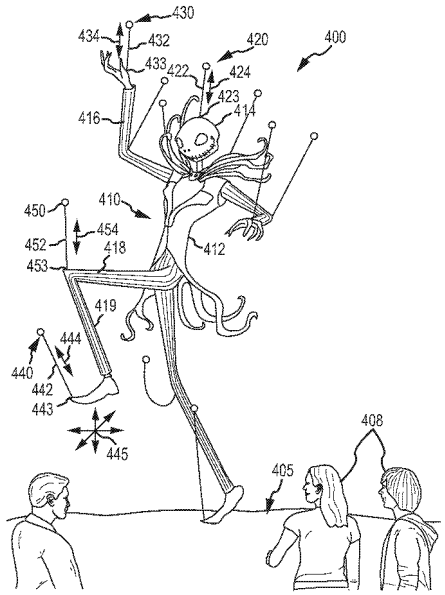


FIG.4

Figure 1.4: Figure from Disney's patent

1.3.2 Verity studios

Verity studios is a company created by Raffaello D'Andrea, professor at ETH Zurich. It specialises in autonomous indoor drone systems for entertainment purposes. Its flagship product is the Lucie micro drone, a small drone weighting about 50g. Fully autonomous and able to fly in swarm of hunderd of drones, it also comes with custom costumes if the high-intensity RGBW light equipped on each one is not enough to produce the desired visual effect [5, 6, 7].

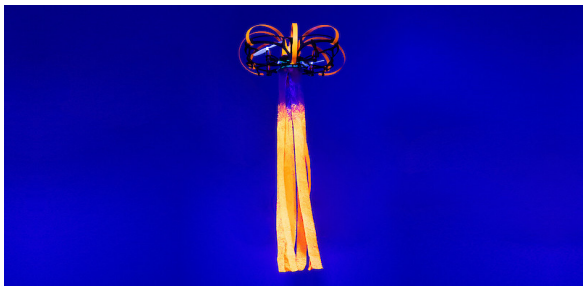


Figure 1.5: Luclille drone dressed as a jellyfish



Figure 1.6: Stage Flyers Drones dressed as lampshades for the Cirque du soleil

The company's main advantages that allow it to commercialise artistic-purpose drones like no other, are the advanced UWB RFID location technology, trough which verity's drones are able to know precisely where they are in the room at any time [8], and the thorough emphasis on security in their system. Indeed the Lucille drone are very light, even with costumes, preventing them from hurting a human even if they were to fail. The heavier drone proposed by Verity studio, the "Stage Flyer" is equipped with fully redundant systems: he is programmed to land safely even if a battery, motor, connector, propeller or sensor fails [5].

The artistic possibilities offered by Verity drones are vast: trough advanced swarm programming, the coordination between the drones is very good, and they can perform a dance choreography [9] like no other before. On top of different costumes and lightning, the heavier stage flyer drone can carry payload such as a mirrors or confetti's that allow for more creative functions. However, contrary to the subject of this thesis or the other projects presented here, the focus of attention is always the drone. No extra actuators are placed on the drone to realise specific actions, nor is the drone used to actuate something external.

1.3.3 Painter drones: Tusru robotics' "grafiti drone" and the PaintCopter

Painting is another potential artistic application of UAV's. Here, the drone is used as a tool to create long-lasting art, as opposed to the performative and ephemeral nature of the other usages described above. The premise is fairly simple: Use a drone equipped with a paint sprayer to draw on hard to reach surfaces. Two entities are currently developing such systems. The first is the Russian robotic firm Tsuru robotics, which plans to commercialise an off-the-shelf "grafiti drone" [10]. The current prototype is able to accurately reproduce a previously computer-based drawing, localising itself in space trough external cameras and on board sensors. This means it can only operate inside, and with cameras previously installed in front of the wall to paint. The paint used comes from standard spray paints can, fitted with custom caps. An umbrella-type protection is used around the mouth of the spraying can to prevent paint to get all over the drone [11, 12].

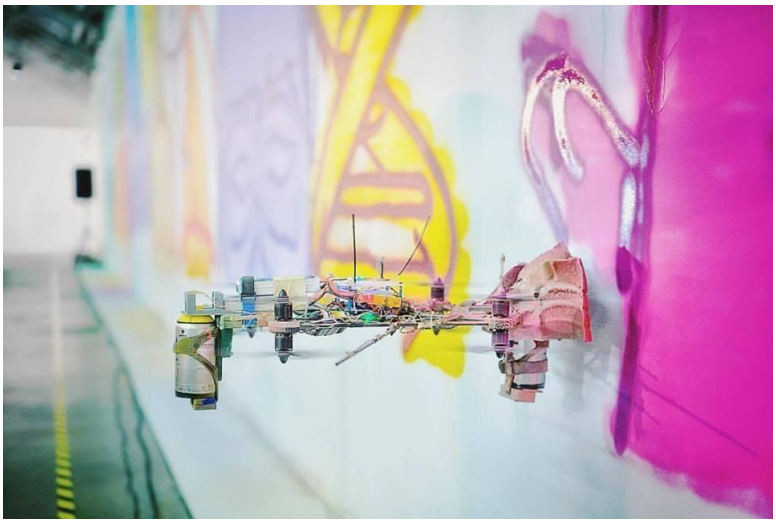


Figure 1.7: Tsuru robotics' prototype

The second developers of such systems are students from ETH Zurich, in collaboration with Disney. The "PaintCopter" they are developing is a drone equipped with a spray gun. The gun is set on a tilt-pan mechanism, itself located at the end of an arm. The drone is tethered so that the paint, power, and some data are fed continuously to the drone trough a line. The system is equipped with a Intel RealSense R200 depth camera so it can see the distance separating it from the wall, allowing it to to adapt its position to be able to paint evenly on an uneven surface. [13]

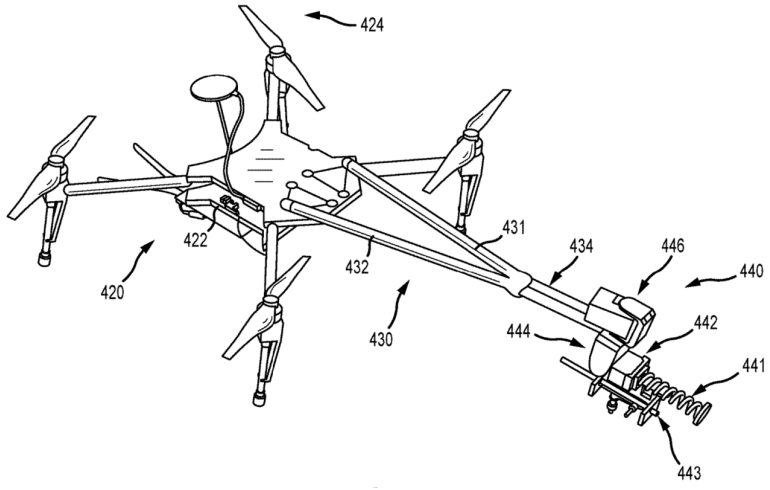


Figure 1.8: PaintCopter, Disney's patent

Chapter 2

Arm and hand

The methodology chosen for this project is to first focus on the hand, then on the arm and finally on the drone destined to carry them. That way, it is ensured that the arm is right for the show. Afterwards the drone is dimmentioned to adjust to that precise payload. This section will discuss the different steps taken in the realisation of the arm and hand combination, what were the difficulties encountered and what went wrong/well.

2.1 Hand

The hand went trough various developments in his conception and trough many modifications became the final model that corresponds to the needs of the project. It is the central part of the system. Indeed, the whole project can be seen as a way to ideally actuate it. The fluidity of its movement, their life-likeness and the ease to control the fingers are crucial.

2.1.1 First model

The first attempt at creating a 3D-printed remote controlled hand, was actuated by two servos in order to be able to do three different hand positions: closed fist, pointed finger and open hand. The goal was to see how effective and expressive the combination of a 3D printed hand and servomotors would be.

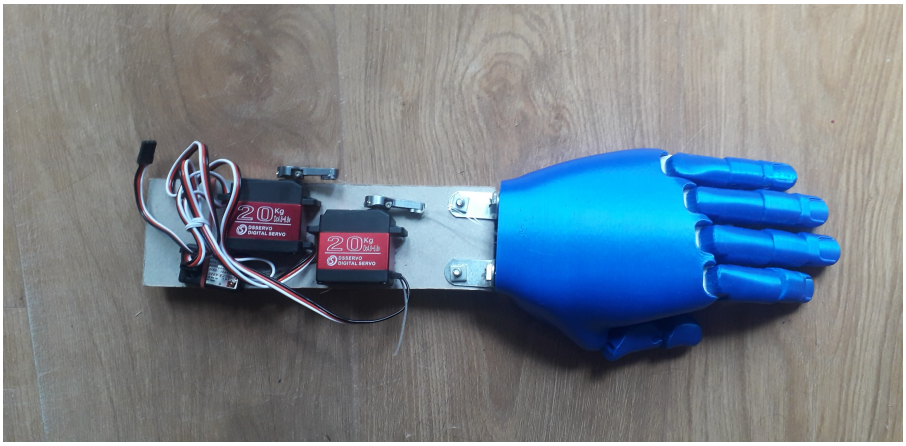


Figure 2.1: first hand and servos

The hand that was printed was directly taken from the Flexy-Hand project [14, 15], which is an open-source project aiming to develop a rudimentary, affordable and easily replicable prosthesis. It consists of one big part for the palm, then small parts for the phalanges and joint, each printed separately. The joints are printed in flexible filament, while the rest is regular PLA. The principle of operation is that wires are attached to the finger tips, and go trough specifically designed holes to the back of the hand. There they are linked to actuators that pull them in order to clench the fingers. The unclenching is assured by the elasticity of the joints, that will put the hand back in the open position once the tension in the strings is gone.



Figure 2.2: Flexy-Hand: palm

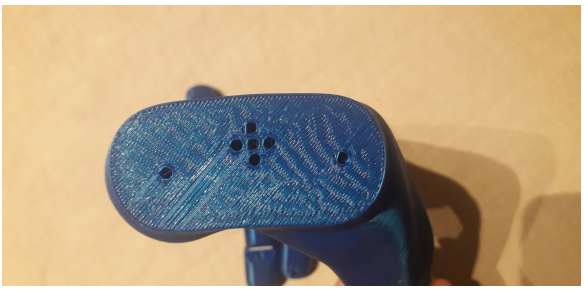


Figure 2.3: Flexy-Hand: back

Since the main goal of this prototype was to test the 3D printed hand, the motors selected were significantly over-dimensioned in order to make sure they could pull the fingers without any trouble. Therefore, ANNIMOS 20KG Digital Servos were used as they offer an easy to control and robust solution to the actuation problem. Their 20 [Kg/cm] torque made sure even one of those motors was sufficient to clench four fingers at once without any trouble. Indeed the force required to pull a finger was estimated to be $\approx 300[g]$ per finger, and the length of the lever is 2.5 [cm]. Even taking into consideration a safety factor of two and pulling four fingers at once, the servo could still take on a draw weight per finger of one kilogram. This did prove amply sufficient.
As for controlling those servos, the FS-GT3B FlySky reciever-transmitter pair was used. This simple but effective radio controller allowed for easy remote control of the servomotors without any extra electronics needed, which simplified the testing of the design a lot.

After having put everything together and testing the movements and responsiveness of the fingers, a few things became apparent. First the hand looked good, and the model was aesthetically sufficient. One thing to notice however, is that a bigger finger to hand ratio would help with the expressiveness of the setup, at the cost of realistic hand proportions. Second the servos worked really well, although they were significantly over-sized, especially taking into consideration that later models would have a motor for almost each individual finger.The big flaw in the 3D model proved to be the joints. Indeed, their elasticity proved insufficient to pull the fingers back at a reasonable speed. Furthermore, if the hand was facing downwards, gravity was enough to bend the fingers in a semi clenched position, giving the hand a sloppy aspect. The hand model had to be changed in favor of a design offering a better pull-back mechanism. Seeing those results, it was decided to build a new hand from scratch, and thereby being able to design every part in accordance to the needs presented by the project. The plan was to first develop a single finger mechanism, then build a whole hand once the process was validated.

2.1.2 Pneumatic fingers

Seeing how the first design of the hand failed due to lack of elasticity of the joints, the first finger design tested was a pneumatic one. Indeed, if the mechanical part was at fault, replacing them by a quick pneumatic actuator would have offered quick movements of the fingers and lightweight design.

In order to have a good pneumatic actuated finger, the first step was to design a pull-back mechanism that could fit around a tube. Originally, a simple elastic was tested, as shown on fig 2.4, but it proved too hard to stretch for a pneumatic system. Magnets were then used as a replacement much mores successfully (fig 2.5). The tilted position of the magnets allowed the pneumatic system to shear them open when it was activated, thus overcoming the non-linearity of the magnet's attraction. After putting on rudimentary pneumatic isolation and testing the system shown on fig 2.6 the finger mechanism works properly.



Figure 2.4: textile elastic



Figure 2.5: magnets

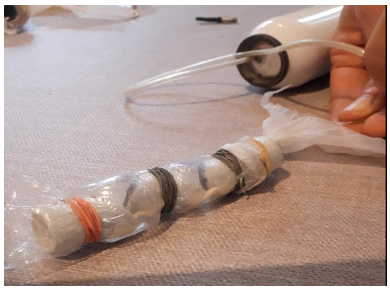


Figure 2.6: final version

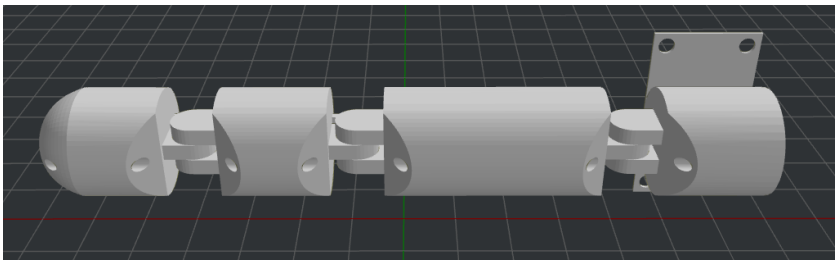
Ultimately it was still decided that the pneumatic system was not fit for the project. The reason for that was triple:

- First, the weight of the valves needed to control the air flow offset the gains in weight of having no motors, thus rendering the weight argument invalid
- Second, pneumatic systems require a lot of compressed air, and a compressor included in the system is out of question in order to retain portability. Therefore compressed air bottles have to be used, whose autonomy is limited.
- Third, the difficulty to manufacture the fingers efficiently using prototyping techniques. Indeed building a finger that was easily inflatable and with a built-in clenching mechanism proved very difficult with the available infrastructures, and one could not be made without significant air loss, reducing autonomy even further.

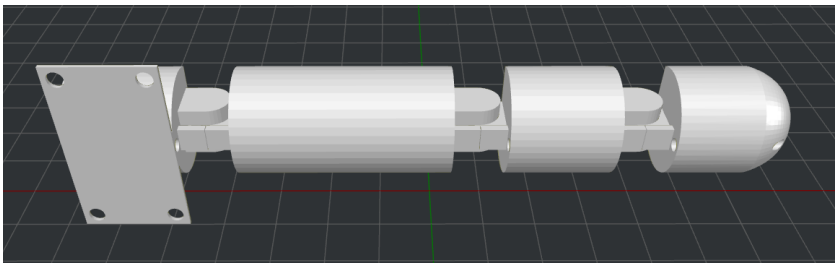
Therefore, it was decided to go back to a design using servomotors and 3D printed fingers. However this time, the model had to be built and designed from scratch, in order to have the proper cinematic.

2.1.3 3D Printed fingers

Unlike the first design that was encompassing the whole hand, the new model is only focusing on one finger to validate the mechanism. This has some implication in term of overall looks, as the palm is no longer printed, but is replaced by a simple board to which the individual finger are fixed. The first 3D model of the single finger looked as shown on figure 2.7. The principle of action is to have one "flexor muscle", being the thread going trough the front set of holes and pulled by the servos, and one "extensor muscle" being an elastic going trough the back set of holes and working passively. The finger is printed in rigid material, and uses hinges as articulations between the phalanges. The front end of the phalanges is also sloped, in order to have just the right clenching angle when the flexor string is pulled by the servo actuating the finger. On the back, stops are put in place underneath the holes to stop the elastic from bending the finger in an unnatural position.



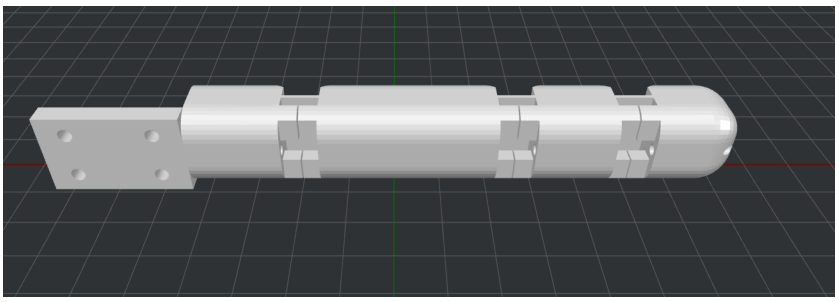
(a) front



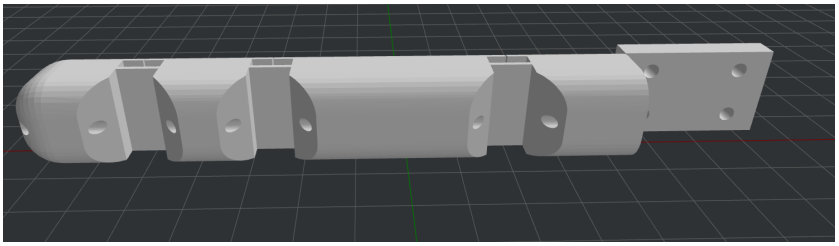
(b) back

Figure 2.7: First idea of finger design

Although the first design's idea was not bad for use, it was poorly optimised for 3D printing. Several changes had to be made in order to make the design easy to print and more robust. First and foremost, the material was changed. Instead of classic PLA, the semi-flexible PolyFlex TPU90 was used. With a shore A hardness of 90 [16] this allowed to have live joints, thin layers of 2mm of material between the phalanxes, instead of hinges. The thin layer is flexible, while the thicker parts of the fingers remain rigid enough, although they are made out of the same material. The shape of the finger had to be changed as well, going from a fully round shape to a square shape with rounded edges, to avoid the use of support while printing. The support plate went from being under the finger from being behind it to allow the finger to be printed sideways, which is necessary to the impression of joints without heavy support. The stops were also moved from under the hole to being around it, to gain space and make sure they do not get in the way of the elastic, causing irregularities in the movement.



(a) front



(b) back

Figure 2.8: Second finger design

Due to the new material not everything was printed right from the first try. Indeed different filaments need different printer settings, and it took a bit of adjustment to find the right one for PolyFlex. First as is often the case with flexible filaments, some oozing happened, sown in figure 2.9 . This is usually due to too high temperatures or too slow travel speed [17]. Secondly, the lower layers of the print tended to be deformed. This phenomenon is called "warping" and occurs when the bottom layers cool down too fast, and thus shrink before the next layers are deposited. When then upper layers cool down as well, they deform the structure lightly. Those deformation accumulate and end up causing the print to fail, as seen in figure 2.10 . This problem can be resolved by using a heated bed (already in use when the faillure occured), by speeding up the impression, or lastly by printing a raft around the piece's base to prevent any deformation [18]. Both problems were resolved by increasing the printing speed and by printing the fingers one by one: printing in batches slows down the impression of an individual finger and thus increases the occurrences of oozing and warping.



Figure 2.9: oozing



Figure 2.10: warping

Finally the printed version, as shown on figure 2.11 worked as expected. The movements were smooth and a simple elastic was enough to put the finger nicely back in its place. The only change that was deemed necessary was to increase the lengths of the two last phalanxes, in order to make the finger movements more expressive and visible from far away, even if it not longer corresponded to realistic finger proportions.

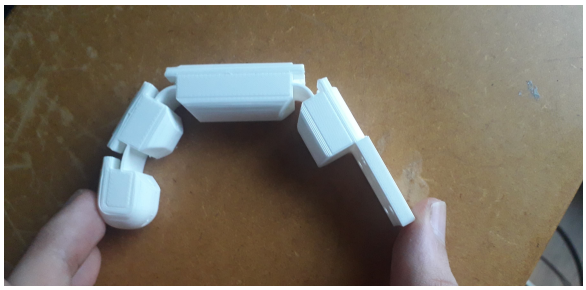


Figure 2.11: printed finger

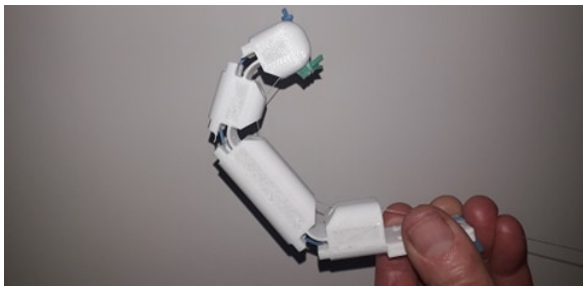


Figure 2.12: with elastic and threads

However, after a prolonged usage of this design in practice, some flaws became apparent. Indeed the kitchen elastics used as extensors have low durability and need to be replaced frequently. Furthermore, the holes designed for those elastics were barely big enough to let them trough, causing quicker wear and difficult replacement. The next step was therefore to use more durable elastics and make a new design with slightly bigger holes, without compromising the integrity of the finger's structure.

Wider holes opened possibilities to use a wider variety of elastics. In order to have optimal finger movements, empirical test were conducted using different finger material and elastics. First, fingers printed in fully flexible filament [19] were tested. Those did not yield significant results as they bent under the tension of the extensors. Slightly harder material was also used, with a shore hardness of A95. The difference in performance was negligible. It was decided that the initial material was the best and therefore had to be kept.

The different elastics tested were: unextruded FilaFlex printing filament, big rubber elastics, small, 1.5[mm] diameter elastic cords used for textile, and a bigger, 2.5[mm] variant. The results were that the FilaFlex filament was too stiff, and only the finger's firs phalanx bent. The rubber elastic had better bending, but its durability was as poor as the smaller kitchen variant previously tested. The two elastic cords had good durability and satisfactory movement. However, the fingers equipped with the smaller variant had a tendency to bend when the arm was moving around, ruining the lifelike impression. The bigger version was therefore picked as the definitive extensor elastic.

2.2 Actuators

After having determined the finger, the general layout of the arm could be decided. The main frame was picked to be made out of wood, on which the fingers and actuators are screwed. Furthermore, the articulations of the arm where picked as a two Degrees of freedom elbow linking the arm with the rod coming from the drone, with no wrist articulation nor shoulder. This keeps the structure simple, while still leaving a wide range of motions possible for animations, especially taking into account the drone's movements.

Servomotor were picked in order to actuate the fingers and the elbow. Indeed that type of motor is cheap, available off-the-shelf and easy to control. The various models available ensure the right motor dimensions for the parts to actuate. The following table 2.1 shows the models available that were pre-selected, as they cost less than 35€, have a minimum of 1.5 [kg/cm] torque and have metallic gears for higher wear durability.

Servo model	Torque @6v	Operating Travel	dimensions	weight	price
SER0046	1.5 [kg*cm]	270°	22.9x12.2x32.5[mm]	12[g]	5.85€
SER0039	1.8 [kg*cm]	180°	22.3x11.8x26.3 [mm]	12[g]	5€
SER0047	1.5 [kg*cm]	180°	22.9x12.2x32.5[mm]	12[g]	5.85€
SER0011	2.5 [kg*cm]	180°	23x12.1x28.8[mm]	14.7[g]	5.85€
MG90D	2.4 [kg*cm]	90°	22.8x12.2x28.5[mm]	13.4[g]	8.44€
528-1496-ND	3.2 [kg*cm]	180°	37x54x20[mm]	40[g]	10.13€
HS311	3.5 [kg*cm]	180°	40x20x36.5[mm]	43[g]	10.39€
DSS-M15	13.5 [kg*cm]	180°	54.5x20x47.5[mm]	52[g]	12.19€
HS422	4.1 [kg*cm]	360°	53x20x36[mm]	45.5[g]	13.78€
DS3218	20 [kg*cm]	270°	40x20x40.5[mm]	60[g]	20.27€
DS3225	24 [kg*cm]	270°	40x20x38.5[mm]	60[g]	30.41€

Table 2.1: Different models of servomotors

2.2.1 For the fingers

After designing and printed the fingers, the next step was to pick a servo to actuate them. The first test, as shown in section 2.4.1 was conducted with HS311 servos that were available in the university's stock and thus easily accessible and free to use. The servos had to be outfitted with custom-printed connectors (see figure C.4 in appendix C) in order to have enough draw length to pull the cables far enough for the full actuation of the fingers. However, those servos are quite heavy, and have an operable travel of 180°, meaning the pulley needed is wider than the one for motors having a longer angle of travel. It was, therefore, opted to pick lighter servos that were better dimensioned for the task. The drawing length for the fingers equipped with kitchen elastics to fully clench was measured to be 56 [mm] and the draw force 3.40 [N], or about 350 [g].

The first step to pick a motor was to compute the pulley radius needed to fully clench the finger: $r = \frac{\Delta L}{\alpha}$ where ΔL is the draw length and α the angle of travel in radiant. This equation is not totally accurate in practice, since the cable does not stick perfectly to the pulley. To compensate this inaccuracy, a 20% increase in length was applied to the ΔL. This gives pulley radiuses of about 22.5 [mm] for 180° servos, 14.5 [mm] for the 270° ones, and 42.8 [mm] for the 90°ones. This last radius was clearly prohibitive due to the limited space available, and thus the MG90D servo was eliminated for this application.

Knowing the pulley radius, the respective torque needed for each model to clench the finger can be computed: $\tau = r * F * sin(\theta)$. This gives a minimum torque of 0.79 [kg*cm] for the 180° servos and 0.51 [kg*cm] for the 270° ones. However, to ensure a seamless working, and guarantee the operating speed (the torque given is the stalling torque), a safety factor of two was picked. This gives minimum torques of 1.58 [kg*cm] and 1.02 [kg*cm] for 180° and 270° servos respectively. The potential candidates were the SER0011, SER0039 and SER0046 servos, as they were the lighter, smaller and cheaper motors that still fitted the criterions. Out of those three, the SER0046 was selected due to its wider operating travel, meaning space was saved on the pulleys.

One optimisation could still be made: the little finger and the ring finger never move independently from each other in the application. It is therefore not necessary to have two different motors to actuate them, they can both be actuated by one servo. That servo would need twice the torque however, so 3.16 [kg*cm] and 2.04 [kg*cm] for 180° and 270° servos respectively. Using the SER0046 is therefore not possible, but the HS311 is perfect for this role. The final actuators for the hand are thus three SER0046 equipped with 14.5 [mm] pulleys and one HS311 with a 22.5 [mm] pulley for the two lower fingers. The final result of the servos equipped with adequate 3D printed pulleys looked as in the CAD shown in figure 2.13 .

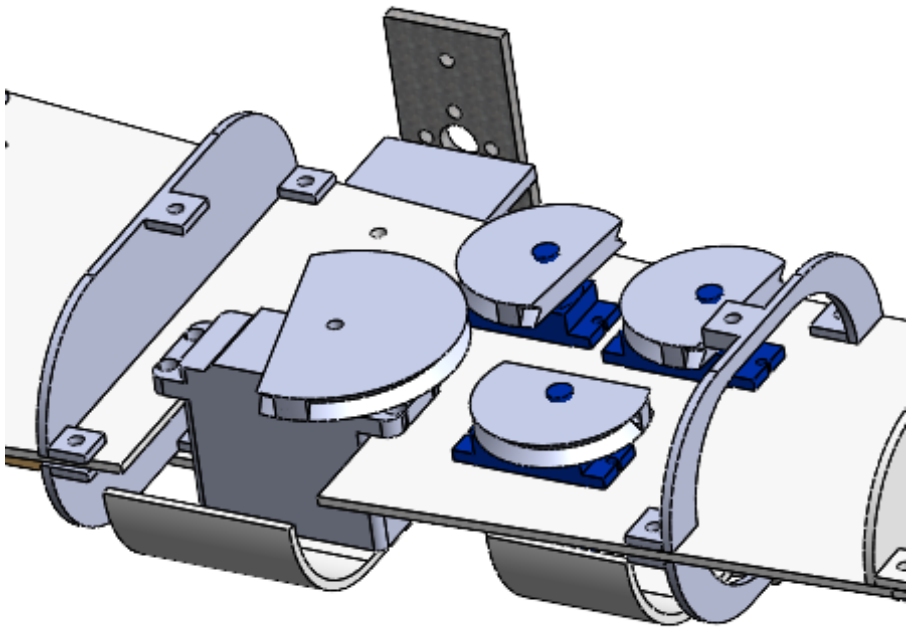


Figure 2.13: Servos with their pulleys, Solidworks CAD

2.2.2 For the elbow

The elbow of the arm is a central piece: it has to be able to point the whole system in the right direction. It is composed of a pan-tilt mechanism described in figure 2.14, where the two servomotors control the θ and ϕ spherical coordinates of the hand's position.



Figure 2.14: pan-tilt mechanism

In order to compute the torque needed for the servos, we needed to know the weights of the different components of the arm, as well as their position respective to the elbow. A schematic of the arm's disposition is shown in figure 2.15

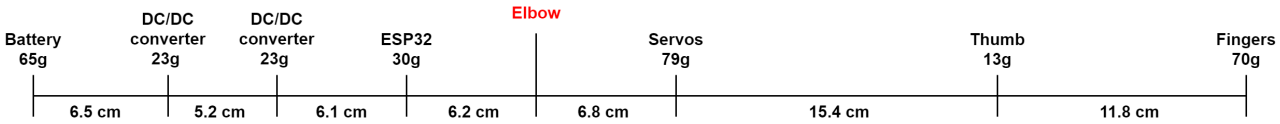


Figure 2.15: Arm weight distribution schematic

Since the arm was well balanced, the remaining torque at rest is small: 0.76 [kg*cm]. But fast movements are wanted, and thus fast accelerations. The real torque needed is therefore much higher than that. Ultimately, the servo picked was the DS3118, as 20 [Kg*cm] was deemed enough to achieve satisfactory movement speed.

Those servos are initially controlled in position, but control over speed is crucial for this project. Some movements need to be fast and aggressive, while other need to be smooth and gentle. Another difficulty is that when using simple position control, when one orders a large elbow movement at max speed, the arm's momentum causes the whole system to wiggle when it reaches the final point. This is less than ideal, and needs to be addressed. Two velocity profiles were studied to be able to control speed and have minimal momentum when reaching the final point.

First, the trapezoidal velocity profile [20]. His input parameters are the final position to be reached by the motor, q_f and the time he has to get there, t_f. Using the pre-determined maximum acceleration a_c, we can compute the position of the motor over time using the following equations:

$$t_c = \frac{t_f}{2} - \frac{1}{2}\sqrt{\frac{t_f^2 a_c - 4(q_f - q_i)}{a_c}} \tag{2.1}$$

Where t_c is the time needed to reach the desired speed.
Using equation 2.1 we can compute $q(t)$, the position over time:

$$q(t) = \begin{cases} q_i + \frac{1}{2}a_c t^2; & 0 \leq t \leq t_c \\ q_i + a_c t_c \left(t - \frac{t_c}{2}\right); & t_c < t \leq (t_f - t_c) \\ q_f - \frac{1}{2}a_c (t - t_f)^2; & (t_f - t_c) < t \leq t_f \end{cases} \tag{2.2}$$

with q_i the initial position. This positon over time is then fed as instruction to the servomotor. The position, speed and acceleration profiles are shown on figure 2.16 below.

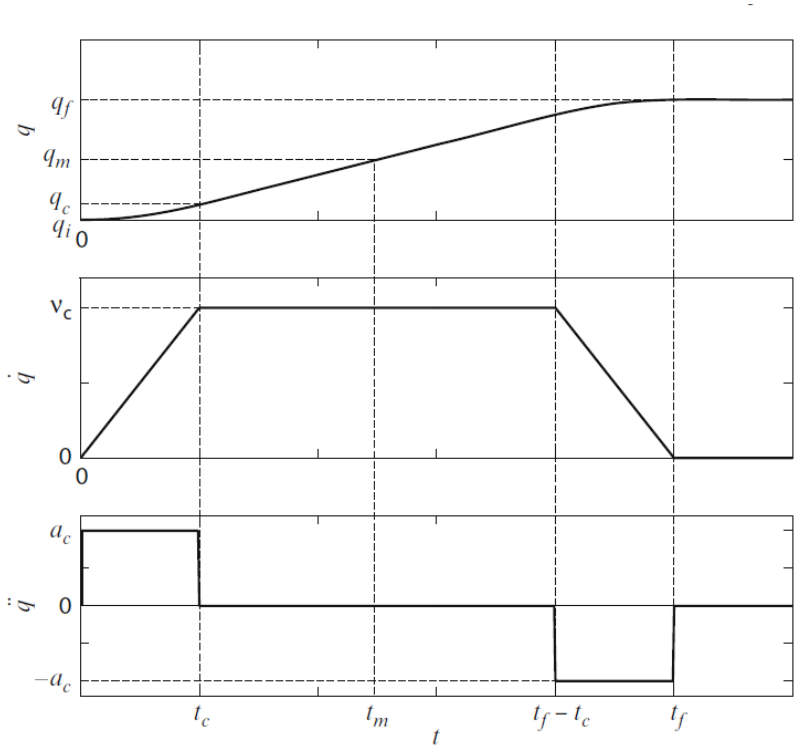


Figure 2.16: Trapezoidal velocity profile [20]

As can be seen, this method guarantees that the motor never accelerate more than the given acceleration. However that maximum acceleration is always used, and there is a minimum time for each movement defined by the parameter:

$$t_{f,min} = 2\sqrt{\frac{q_f - q_i}{a_c}} \tag{2.3}$$

Indeed, for that time $t_f = 2t_c$, and the profile is degenerated to a "bang-bang" profile where the maximum acceleration/deceleration is always used. There is no speed plateau anymore.

The second profile studied is a modified bang-bang control, where the input parameters are the final position q_f and the movement time t_f, instead of just the final position as in classical bang-bang [21]. The goal here is to minimize the acceleration for a given movement time. In order to obtain this result, the acceleration needed is computed first before the bang-bang equation can be applied.

$$a_c = \frac{4(q_f - q_i)}{t_f} \tag{2.4}$$

The equations of position in function of time then becomes:

$$q(t) = \begin{cases} q_i + \frac{1}{2}a_c t^2; & 0 \leq t \leq \frac{t_f}{2} \\ q_f - \frac{1}{2}a_c(t - t_f)^2; & \frac{t_f}{2} < t \leq t_f \end{cases} \tag{2.5}$$

The position, speed and acceleration profiles are shown on figure 2.17 below.

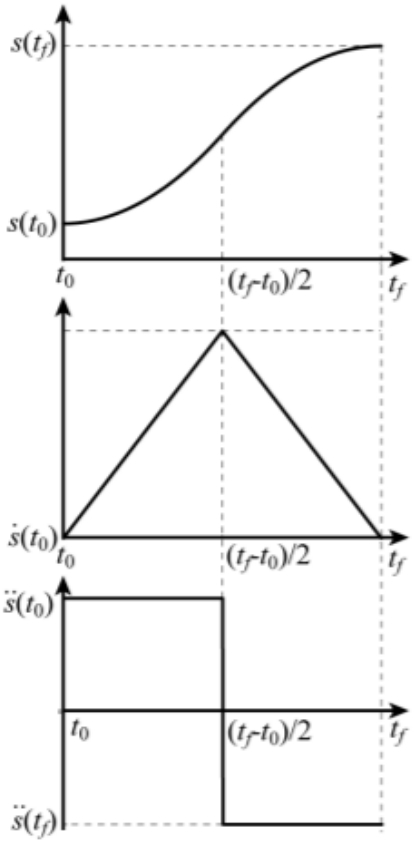


Figure 2.17: Bang-Bang velocity profile [21]

This profile offers the benefit of always providing the minimum acceleration required to reach the desired position in the desired speed. The max speed reached is however higher than for the trapezoidal profile for the same movement. Both velocity profiles offer satisfactory results, and thus both are used. The trapezoidal profile is preferred for slower movement where the speed needs to be constant for aesthetic purposes. The bang-bang profile is used for faster movements, including the longer ones such as in the slap animation.

2.3 Electronics and battery

The electric layout of the arm looks as shown on figure 2.18. The LiPo battery supplies 11.1 [V] DC to two DC/DC converters. The smaller one supplies the microcontroller with a 5 [V] output. The nominal current of the buck is 2 [A], while the controller only consumes up to 400 [mA], so there are no risk of overheating it. The bigger DC/DC converters supplies the different servos with 6 [V]. Its nominal current is 12 [A]. Indeed it is designed for the worst case scenario where all the servos are simultaneously stalled: The stall currents being 1.1 [A], 0.8 [A] and 1.8 [A] for the SER0046, HS311 and DS3218 respectively. This gives a worst case current of 7.7 [A], much more than the small buck could handle. In practice, however, this maximum current will never be reached and the bigger converter is mostly a safety measure.

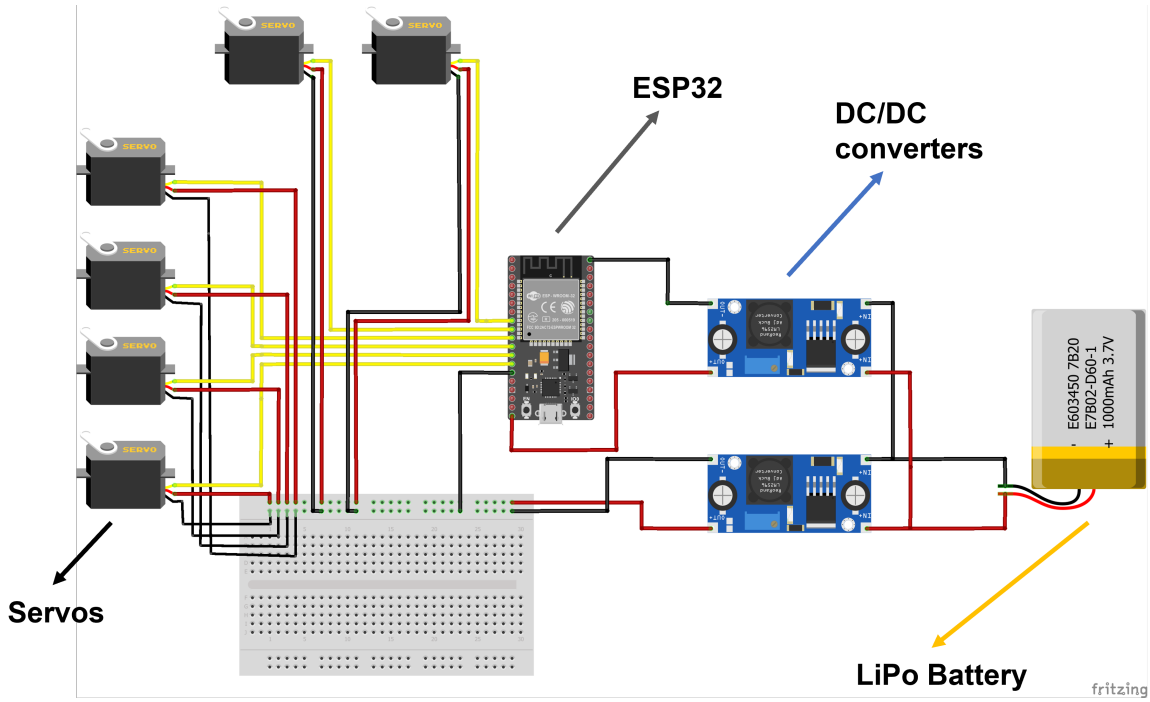


Figure 2.18: Electric layout of the arm

The ESP-WROOM-32 microprocessor is based on the Arduino architecture, but it includes Wi-Fi and Bluetooth capabilities, which makes it easy to use for remote-controlled applications. Its processing power is more limited than controllers such as the Raspberry pi, but the tasks it needed to handle were quite simple: Send PWM signals to drive the different servos in function of the instruction given via Wi-Fi. As the electricity consumption is also quite low with 400mA maximum consumption, it was the ideal solution for a first prototype. However although communications via Wi-Fi are easy to use and implement, and thus great to test the arm's functionalities without having the drone, they are not reliable when used in a room potentially full of external Wi-Fi signals.

The communication between the operator and the board is done through a website hosted by the ESP32. Any device having access to the internet can connect to this site whose URL is the board's IP address, and thus serve as a remote, it just has to be connected to the same network as the micro controller. The operator can activate the toggles corresponding to the desired animations on said website, as shown on figure 2.19. Those animations are short sequences of movements lasting between one and ten seconds that exemplify the arm's expressive possibilities. Examples of such animations include slapping, playing rock-paper-scissor or menacing the crowd with a finger gun.

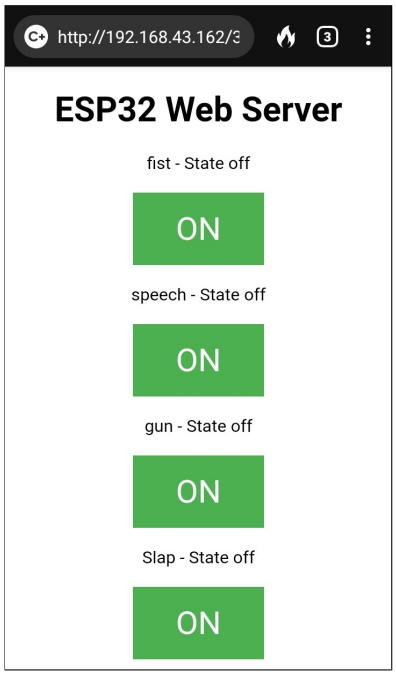


Figure 2.19: Toggles corresponding to the animations, on the ESP32's website

2.3.1 Battery dimensioning

The battery used to supply the arm is independent from the drone's battery: It, therefore, must have about the same autonomy under normal working conditions. To compute the needed battery capacity, the assumption is that one servomotor on average is active at all times. The mean active and idle current consumption for each servo model is given in table below:

Servo model	idle current	running current
SER0046	40 [mA]	440 [mA]
HS311	7.7 [mA]	400 [mA]
DS3218	4 [mA]	800 [mA]

Table 2.2: Servo's current

This gives a weighted average of 22.6 [mA] and 553 [mA] for the idle and running current respectively. Using the assumptions about the servos usage, the computed mean current used by the motors is 666.4 [mA]. Adding the 200 [mA] used by the esp32 in active mode [22], gives a constant consumption of 866.4 [mA]. The target autonomy being 30 minutes, a 800 [mAH] 3S LiPo battery was deemed amply sufficient, as it should be able to supply the arm for about 55 minutes under normal use.

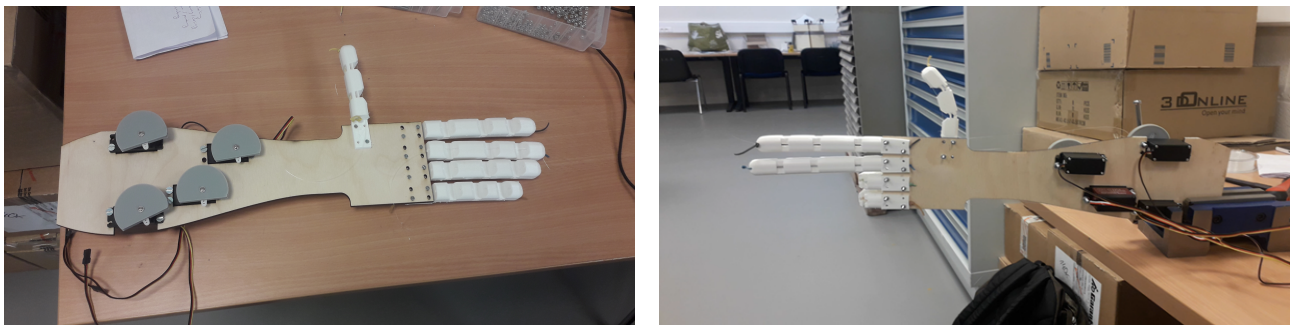
2.4 Assembly

This section will discuss the overall working of the arm when assembled, the ways the different components interact, and how it connects to the drone. The aesthetic aspect is also important for an artistic performance: The arm should be wrapped in a cloth sleeve to hide its mechanisms, rending make-believe easier. This is also discussed in this section.

2.4.1 First test

The fingers and HS311 servos were screwed onto a laser-cut piece of wood, as shown in figure 2.20, in order to test their working together. The remote control worked great and the movements as expected, but it remained an early prototype that could be improved. First, the connections between the motors and the micro controller was still done with a breadboard and not a PCB. Cable were loose and not soldered. Second, the wooden plank used was too thin and could bend instead of the fingers when the servos were pulling. Finally, a more well rounded chassis was needed in order to both be more rigid, but also accommodate space for the battery, micro controller and the elbow joints between the arm and the drone.

Furthermore, the power supply was not yet using a rechargeable battery and thus not optimal. The 5V dc required by the servos and ESP32 was provided by four non-rechargeable standard AA batteries, but those run out quick and a more sustainable solution was needed. The battery dimensioning is done in section 2.3.1. Additionally, the HS311 servos used for the first test because they where readily available in the university's stock, although less powerful than the ones used for the first hand, were still too powerful for the simple actuation of a finger. A more thorough dimensioning in order to select the right size of motors was needed, as done in section 2.2.1.



(a) Front (b) Back

Figure 2.20: First arm assembly

2.4.2 Improvements

After having done the first test with only the motors and fingers on the board, it was time to think of the true final form of the arm, with every component attached. Having designed a new board shape, the first step was to place the motors as closely possible to the fingers. That way, the cables are shorter, which facilitates design. Next came the crucial placement of the pan-tilt elbow mechanism: Since it is trough that mechanism that the arm is connected to the drone, it needs to be balanced around that point to avoid a resulting torque on the drone and the elbow servo at rest. The movements of the arm also needed to be taken into account in that placement however, as placing the pan-tilt mechanism too close to the hand would mean a smaller movement radius, and thus less expressiveness. Taking that into consideration, the anchor point was placed around the middle of the arm, and the heaviest free part, the battery, was placed on the extreme left side to act as a counterweight. The final weight distribution is described in figure 2.15, and the resulting arm is slightly unbalanced towards the front due to the need to preserve the movement's amplitude. However that imbalance is small. The result can be seen on figure 2.21. The microcontroller and the servos' cables were soldered to the back side of the arm.

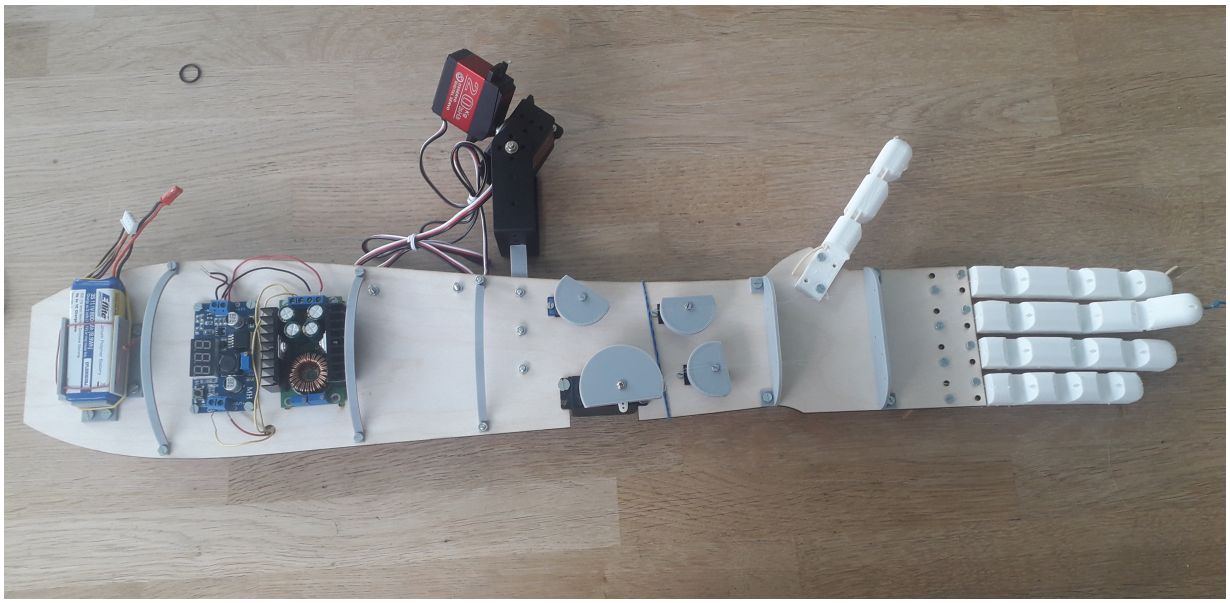


Figure 2.21: Improved arm assembly, front

After placing the functional parts, 3D printed hoops were placed at regular intervals all over the arm. Those had two functions: first, be the support for the decorative sleeve that had to clothe the arm, as shown in figure 1.2. Secondly, on the front part, they also served as a guide for the cables pulling the fingers. Those hoops where however insufficient to prevent the servo pulleys from getting caught in the sleeve, and they where thus supplemented with a protective plate around the relevant areas in the final model.

2.5 Final model

The assembly presented in the previous section 2.4.2 worked as intended, but usage revealed some improvements that could still be made. First and foremost, the goal was to put a decorative sleeve on the arm. To ensure this sleeve did not hamper the pulley's movements, a special protective shell was 3D printed and screwed around the motors, as shown on figure 2.22. This way, the sleeve could wrap around the arm without risking getting caught in the pulley's movements. Furthermore, a L piece was machined out of stainless steel to replace the 3D printed piece that was previously linking the elbow with the arm.

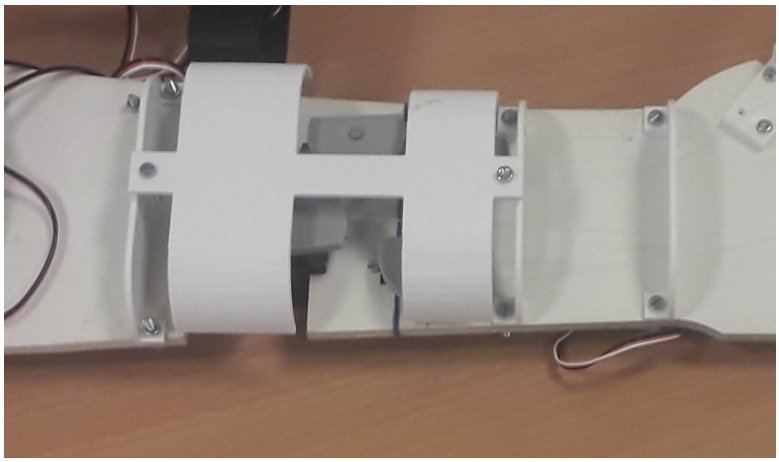


Figure 2.22: Servos protected by the shell

Some aesthetic arrangements where also made to improve the final look of the model. First, all the visible hoops where reprinted in white, and the wooden piece was painted too. Then, it was decided that the wrist would not be covered by the sleeve, and it thus had to be devoid of motors, pulleys, etc... To achieve this, the arrangement of the board was slightly changed, with the servos more central. Finally, the sleeve was designed by artist Julia Zamboni and fitted on the arm. The final result looked as shown on figure 2.23.

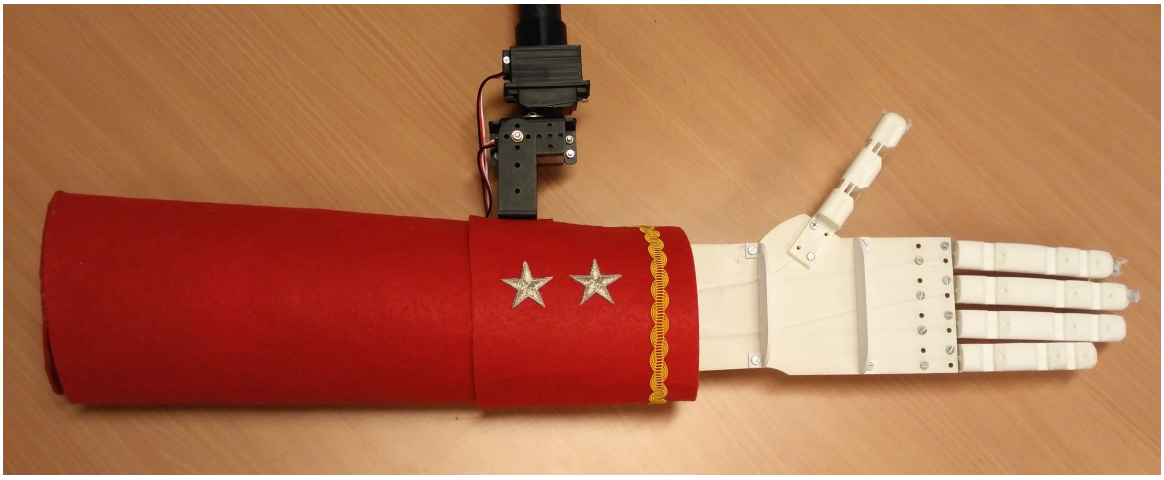


Figure 2.23: Final arm with sleeve

In order to attach the arm to the carbon rod hanging from the drone, a custom piece, shown in figure 2.24 was 3D printed. This piece has radial clamping for the rod, and a special housing for the elbow's upper servo, where it can be screwed with ease. The design avoids cantilevers as much as possible in order to require less printing supports. The printing filament is semi-flexible PolyFlex TPU95, featuring a shore hardness of 95A [23], so that the piece would bend rather than break in case of high stress resulting from the arm's movements.

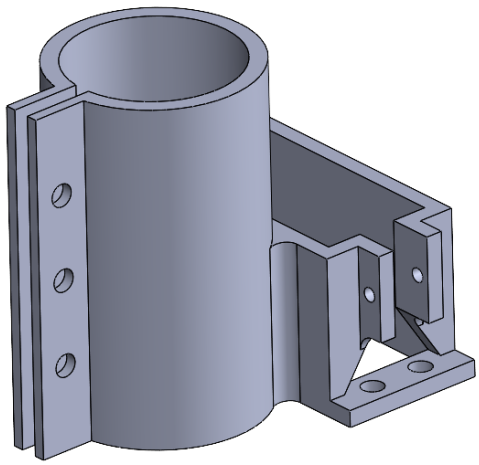


Figure 2.24: Junction piece between the elbow's upper servo and the rod hanging from the drone

Chapter 3

Drone hardware

In order to carry the arm, a special drone was needed. It had to be silent, stable, with a carry capacity sufficient to lift the arm's weight of 1 [kg] with ease and enough autonomy to last for the show's duration. This drone was realised by Michaël Daris, founder of RC Take-off[1], a company focused on the creation and sale of custom-made, high-performance drones. This section will review the specificities of the drone's hardware, and discuss it's performances and different component choices.

3.1 Frame & battery

The drone needs to support the arm described in chapter 2 without being too close from it: The arm needs to be able to point upwards without colliding with the drone and it must not be affected by the air blowing from the propellers. It is also important to have a clear visual distinction between drone and arm: the arm is the show's focus, not the drone. To ensure this is the case, a 1m carbon rod of 30 [mm] of diameter is linking the two parts, as shown on figure 3.1. More than just a link, this rod is a central piece of the assembly: The battery is attached to it in order to keep the frame as small as possible, which reduces the visual impact of the drone. The rod is attached to the drone frame via a rubber junction, that serve as a mean to decouple the drone and the arm's movements. Indeed the drone needs to tilt in roll or pitch to move sideways or forward (see figure 4.3 for definition of those angles). Those movements should not be translated to the rod nor the arm. This junction, shown in figure 3.2 is realised with a hose, whose length will be adjusted to have the optimal rigidity. The goal is to avoid a pendulum effect when the drone is moving, but also have a degree of decoupling as explained above.

[1] https://www.rctopro.be



Figure 3.1: Custom-made drone carrying the arm trough the carbon rod.



Figure 3.2: Flexible junction between the rod and the drone.

The rod's solution was privileged over another possible solution, using ropes to hang the arm from the drone, for two reasons: First the ropes would not have the convenient side effect of holding the battery and reducing the drone size. Secondly, although the ropes give decoupling between the arm and the drone, they can not avoid some degree of oscillation from the arm after it has moved. The platform to attach the fixations being only the upper part of the elbow, it is not wide enough to allow for rope dispositions that would limit those oscillation.

The battery used for the tests is a 6s 6000 [mAh] LiPo battery, providing easy recharge and replacement, as multiple of such batteries are available, meaning multiple can be prepared in advance before a flight. However, this model needs to be fixed around the rod, and its capacity is limited compared to the potential final battery used for the show. Indeed, it would not be able to last for the entirety of the show's estimated thirty minutes. The final battery for the show is custom-made, and consist of fitting multiple Lithium-ion cells inside the rod, offering a more discreet look and more power, at the cost of harder recharge, no possibility of easily swapping batteries, and more expensive materials.

3.2 Motors and Propellers

This section will discuss the motors, propellers and electronic speed controllers combination picked for this project. Those elements will determine how the drone behaves, its power consumption and the noise it generates. Therefore, they need to be compatible not only amongst themselves, but also align with the general uses of the quadcopter and its goals.

3.2.1 Noise considerations

Keeping the noise to a minimum is a crucial part the drone's conception goal, as high volumes would disturb the show's spectators and disrupt any music that accompanies the characters. It is therefore worthwhile to spend a little time studying the noise caused by drone propellers and the ways to reduce it.

The sound generated by a flying quadcoper comes from two sources: The motors that produces a typical high-pitch noise, and the interaction of the propeller blades with the air. This latter is the dominant source in many if not all cases [24]. The following parameters impact this noise: The propeller speed and the propeller length, along with the number of blades and their pitch (illustrated in figure 3.3). This means that the thrust generated by the propeller impacts the noise, since more thrust requires longer or faster spinning propellers. This also means that heavier drones, needing more thrust to be lifted, will also produce more sound.

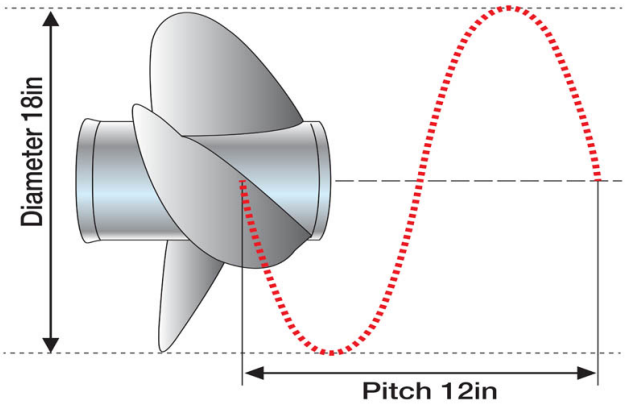


Figure 3.3: Illustration of blade pitch for a marine propeller. [25]

Considering that the thrust and weight of the system are dictated by the needs and displacements from chapter 2's arm, the main parameter impacting the noise is the blade tip speed. Indeed, fan affinity laws by Madison [26] tells us that the radiated sound power of a fan W scales as follows:

$$W \propto D^2 u_{tip}^5 \tag{3.1}$$

With D the diameter (length) of the blade, and u_{tip} the tip speed. Following the calculations suggested by Brungart [27], the equation conservation of angular momentum is used to derive the expression of propeller static trust T at hover, assuming minimal change in pressure (and thus low speed):

$$T = \rho Q U_e \tag{3.2}$$

Where ρ is the air density, Q the the volumetric flow rate trough the propeller disk and U_e the flow velocity when exiting the propeller disk. Assuming that:

$$Q \propto \omega D^4 \tag{3.3}$$

$$U_e \propto \omega D \tag{3.4}$$

Where ω is the angular velocity of the propeller. The propeller trust is proportional to:

$$T \propto \omega^2 D^4 \tag{3.5}$$

Yet, the blade tip speed u_{tip} is proportional to ωD. By substituting this into equation 3.1, the result becomes:

$$W \propto \omega^5 D^7 \tag{3.6}$$

By combining equation 3.5 and 3.6 the impact of a change in blade diameter at fixed thrust can be predicted. Let D_1 and D_2 be two different diameter of propellers:

$$\omega_1^2 D_1^4 = \omega_2^2 D_2^4 \tag{3.7}$$

$$\frac{W_2}{W_1} \propto \frac{\omega_2^5 D_2^7}{\omega_1^5 D_1^7} \tag{3.8}$$

$$\frac{W_2}{W_1} \propto \left(\frac{D_1}{D_2}\right)^3 \tag{3.9}$$

This conservative analysis shows that a blade twice as big as another blade will produce eight time less sound power for the same thrust. Although in practice heavier motors are needed to rotate longer blades, the noise increase due to higher trust only slightly offset the noise reduction described above. Experimental results, shown in figure 3.4 validate this theory [27]. It is important to notice that those quieter properties come at a cost: longer propeller have more inertia, and are therefore harder to accelerate. Drone having comparatively longer propellers are therefore less nimble that their short-propeller counterpart, even if they are more silent. This project thankfully does not require quick accelerations or deceleration of the drone and it is therefore not problematic to use long propellers to obtain a quieter quadcopter.

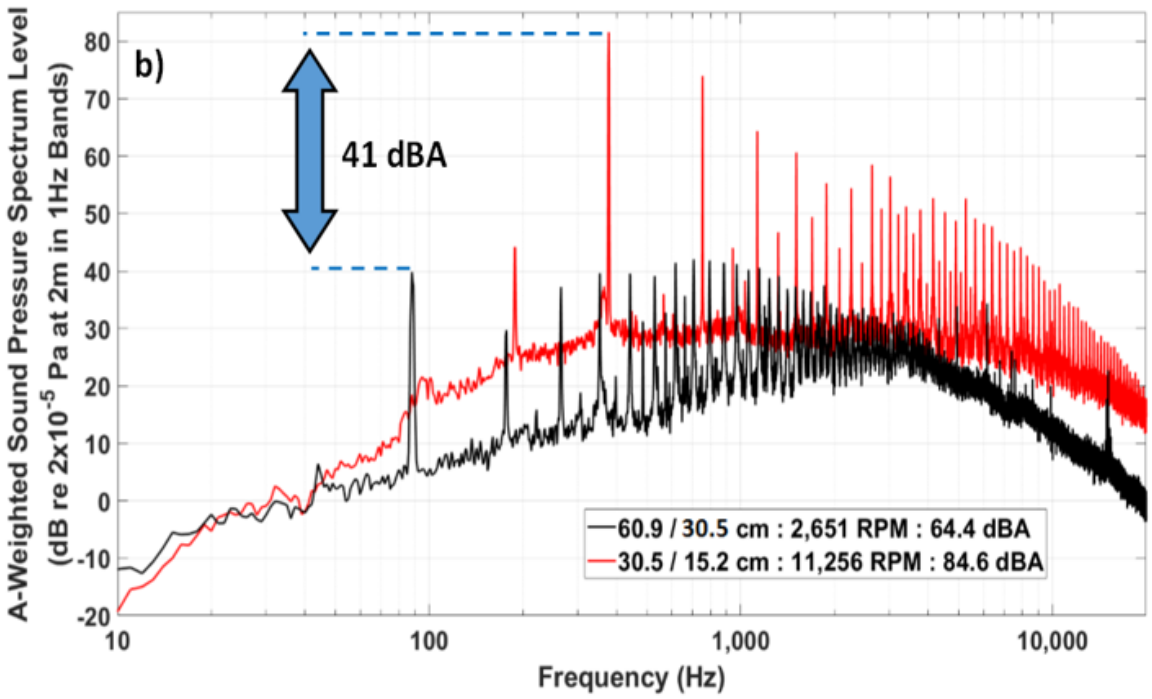


Figure 3.4: Sound pressure measured for 60.9 cm and 30.5 cm diameter propellers providing 27.8N of static thrust.

Equations 3.5 and 3.6 can also be used to study the impact of an increase in thrust at fixed propeller diameter on the radiated sound power:

$$\frac{W_2}{W_1} \propto \left(\frac{T_2}{T_1}\right)^{\frac{5}{2}} \tag{3.10}$$

Since thrust at hover is directly proportional to the drone weight, this equation shows the importance of minimising weight in order to reduce noise, since the sound power is proportional to the drone mass to the power $\frac{5}{2}$.

Other parameters of rotor shape have also been studied to infer their impact on noise. The first one is the blade pitch, measured in theoretical forward displacement of the propeller after one rotation, as illustrated in figure 3.3. However, experimental results by Brungart [27] have shown that the pitch's impact on noise is negligible compared to the propeller's diameter for a constant thrust, as seen in figure 3.5. Those experiments also showed that the number of blades per propeller had no significant impact on noise, as the increased efficiency, and thus lowered tip speed, induced by the extra blades was offset by the increased propeller blade to blade interaction generated noise.

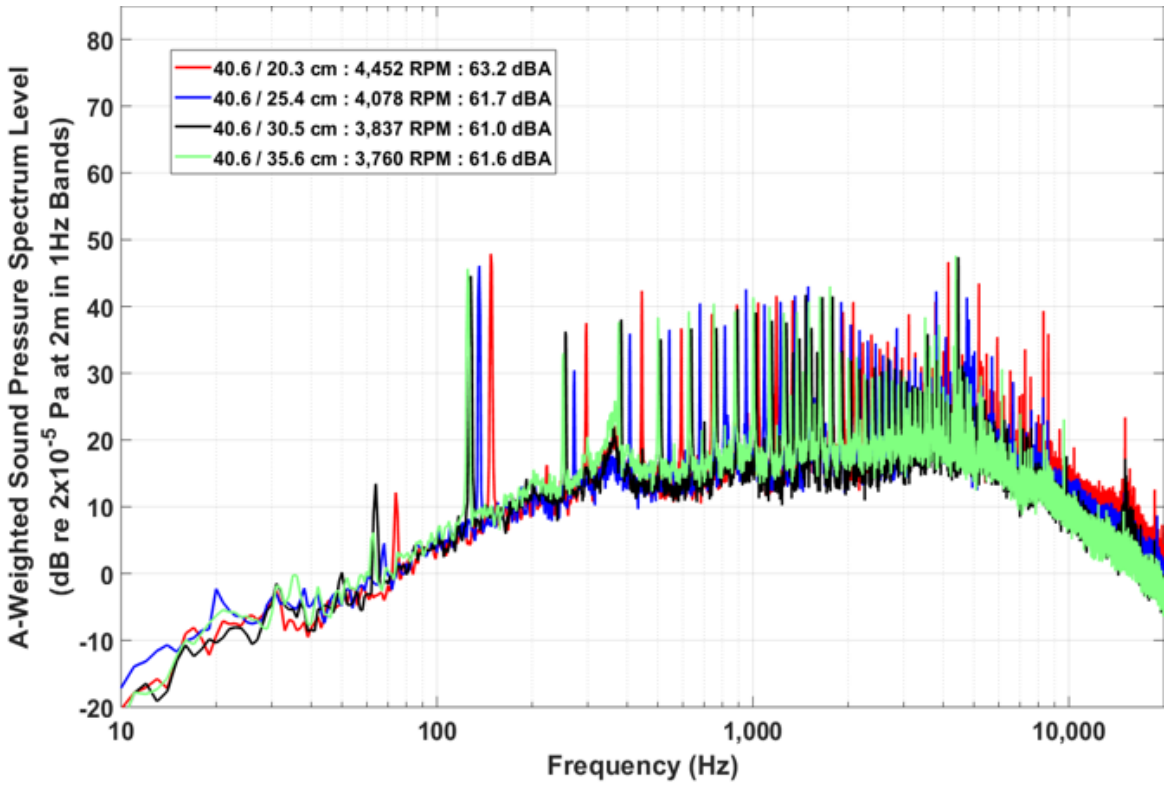


Figure 3.5: Sound pressure measured for 40.6 [cm] diameter blades providing 13.9 [N] of static thrust, with various pitch.

Seeing those results, and taking into account the context of this project, it is clear that the main parameter of focus needs to be the propeller length. The blade numbers and their pitch should not be dictated by noise requirements. Large, slow-moving propellers are ideally suited to ensure low sound emitted during the live show, although at the cost of manoeuvrability. Ideally, the weight of the drone-arm assembly should be kept at a minimum, while still keeping the expressive and functional exigences intact.

3.2.2 Models used

To buy the optimal propellers, a pre-selection was made following Michaël Daris' recommendations. The most relevant ones for the project are shown in the table 3.1 below:

Propeller model	Diameter [cm]	Pitch [cm]	weight [g]	winglet
NS16x5.4	40.7	13.7	13	Yes
NS17x5.8	43.2	14.7	16.5	Yes
NS18x6	45.7	15.2	17	Yes
NS20x6.2	50.8	15.7	23	Yes
G18x5.9	45.7	15	34	No
G20x6.5	50.8	16.5	44	No
FA16.2×5.3	41.2	13.3	30	Yes
FA18.2×5.9	46.2	15	40	Yes
FA20.2×6.6	51.3	16.8	57	Yes
P17x5.8	43.2	14.7	27	No
P18x6.1	45.7	15.5	32	No
P19x5.7	48.3	14.5	41	No

Table 3.1: Available propellers from T-motor

Taking into consideration the results from section 3.2.1 above, the chosen propellers where picked as wide as the artistic constraint permitted. Indeed, for the drone not to dwarf the arm too much, it was decided with the artist that the drone's length should not exceed one meter, considering the arm's length of 70.9 cm. Therefore, the maximum theoretical diameter for the propeller was 50 cm, assuming propellers are perfectly close together. Furthermore, winglets such as in figure 3.6 are preferred since they are designed to improve efficiency, meaning identical thrust can be achieved with less RPM, further reducing the blade tip speed, and thus noise. Therefore, the model picked was the FA18.2×5.9 Propellers, since they come the closest to 50cm diameter while being smaller and have winglets.



Figure 3.6: Illustration of the winglets' effect on the blade generated vortex [28]

The next step was then to pick suitable motors for the FA18.2×5.9 propellers. The compatible ones are listed in the table 3.2 below. The Kv value represents the output RPM of the motor at no load per volt of input. The thrust and RPM values are measured for propellers of comparable size of the FA18.2×5.9 and at 50% thrust, since the characteristics for lower blade speeds are the most important for this project, where the goal is to remain as quiet as possible.

Motor model	Kv [RPM/V]	Thrust at 50% [g]	RPM at 50%	weight [g]
Antigravity MN5006 KV300	300	968	2988	108
Antigravity MN5006 KV450	450	853	2805	106
MN501-S KV240	240	1339	2361	170
MN501-S KV300	300	1382	3186	170
MN501-S KV360	360	1450	3658	175
U7-V2.0 KV280	280	980	3133	255
U7-V2.0 KV420	420	2250	4200	255
U7-V2.0 KV490	490	2720	4900	258

Table 3.2: Motors compatible with the FA18.2×5.9 Propellers

Considering that the drone with payload is estimated to weight about 3.5 [Kg], all motors should suffice to lift it, even if its only at 50% throttle. The next criterion to select the motor is therefore having the lowest possible KV and therefore RPM, to produce as little noise as possible. The best choice is thus the MN501-S KV240. Its interesting to note that at 100% throttle, one of those motors has a thrust of 4187 [g]. The quadcopter having four of those means that it only uses about 20% of its total capacity with its current weight, all in the interest of silent flight. Another benefit of picking those motors is that they are available in the form of an integrated propulsion system, which encompasses both an ALPHA 60A electronic speed controller (ESC) and a MN501-S KV240 motor, as shown in figure 3.7. This solution has the benefit of being robust and easy to install, and lets the ESC's benefit from the ventilation from the propellers to cool down.



Figure 3.7: MN501-S motors with integrated ESC's [2]

[2]https://store.tmotor.com/goods.php?id=797

The role of the ESC is to convert the DC current supplied by the battery and the reference signal from the flight controller into a three phase AC current having appropriate frequency to get the motor to rotate at the desired speed. The ALPHA 60A electronic speed controllers used for this project have a continuous current rating of 60 [A], much more than the peak current of 25 [A] drawn by the MN501-S motors, meaning a sufficient margin of safety is attained.

3.3 Electronics

When it was made, the drone's electronics where comprised of a NAZA V2 flight controller, which is known to be very reliable in operations without GPS, alongside a Spektrum TM1000 telemetry module. Those components are robust and perfect for the first manual flights. However they are bound to change when the project evolves into an automatic piloting, and are therefore disposed right on top of the drone, where they are easily accessible and replaceable by other components if the need arise. This is shown in figure 3.8.

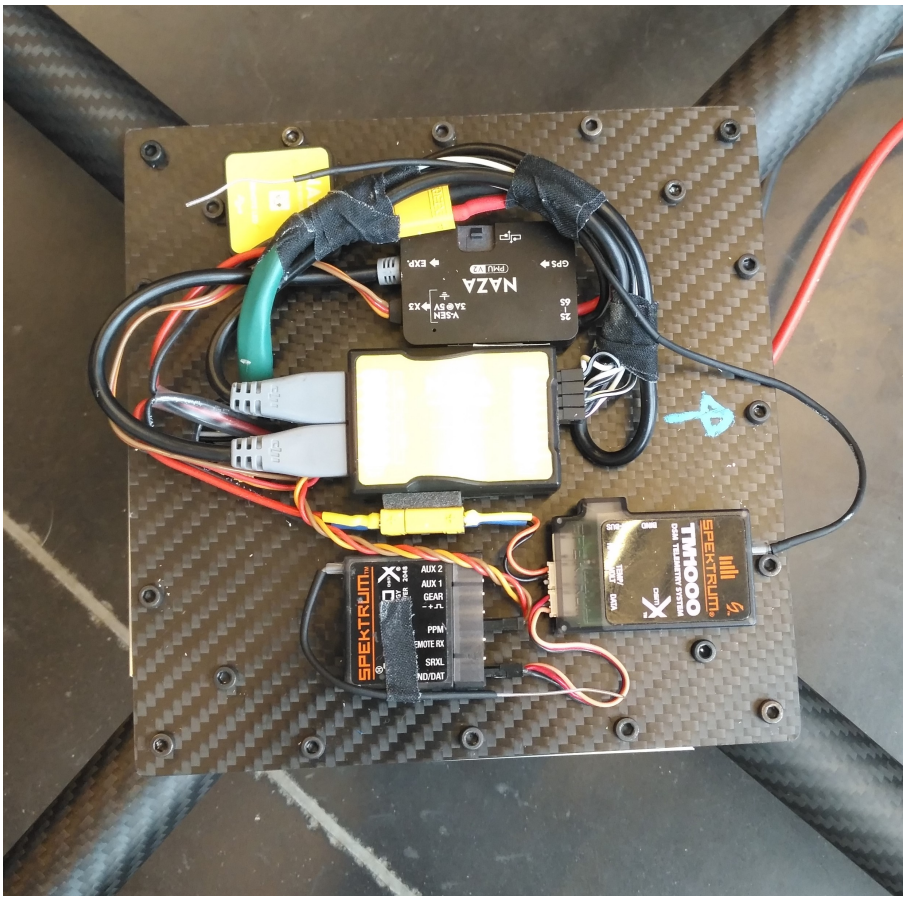


Figure 3.8: Drone electronics, as made

3.4 Characteristics

Having picked the components of the drone, some of its characteristics such as flight time can be computed. This is done using the eCalc software[3]. Input parameters are given in table 3.3, where the battery C-rating represents the maximum current it can output as a proportion of its capacity. Here, a 6000 [mAh] battery with a C-rating of 20 will have a max current output of $20 * 6000$ [mA] thus 120 [A]. Its important to note that those computation are made for the drone without payload, and using the LiPo battery mounted on the rod and not Li-Ion batteries installed in the rod. The addition of payload will of course diminish the drone's flight time and max speed, but on the other hand an increase in battery capacity would mean longer flight times.

Inputs			
Weigth w/o Battery:	2500 [g]	Nr rotors:	4
Frame size:	70 [cm]	Battery type:	LiPo 6000 [mAh]
Battery continuous discharge rate:	20C	Battery max discharge rate:	30C
Battery configuration:	6S	Max discharge:	85%
ESC type:	max 60 [A]	Motor type:	MN501-S KV240
Propeller type:	T-Motor CF	Propeller Diameter:	45.7 [cm]
Propeller pitch:	17.5 [cm]	Nr of blades:	2

Table 3.3: Input parameters for the eCalc software

The results are shown in figure 3.4. Again, the max speed, rate of climb, range and max tilt will be significantly smaller with the payload, but they are not crucial for this project. The flight time will also be smaller but is heavily dependent on the batteries which will change following the needs of the project.

Outputs			
Battery			
Flight time at max speed:	4.7min	Load:	10.91C
Flight time at hover:	17.6 [min]	Energy:	133.2 [Wh]
Motor at hover			
Current:	4.34 [A]	Voltage:	21.78 [V]
Revolutions:	2655 [RPM]	Throttle:	54%
electric power:	94.6 [W]	Specific thrust:	8.86 [G/W]
Quadcopter			
Max. tilt:	60°	Max. speed:	51 [Km/h]
Est. range:	3694 [m]	Rate of climb:	5.9 [m/s]

Table 3.4: Outputs from the eCalc software

The graph in figure 3.9 shows the estimated flight time and range in function of the drone speed. The project does not require to travel long distances, and the speed required is relatively low, therefore a good estimation of the flight time can be made by looking at the low speed values.

[3]https://www.ecalc.ch

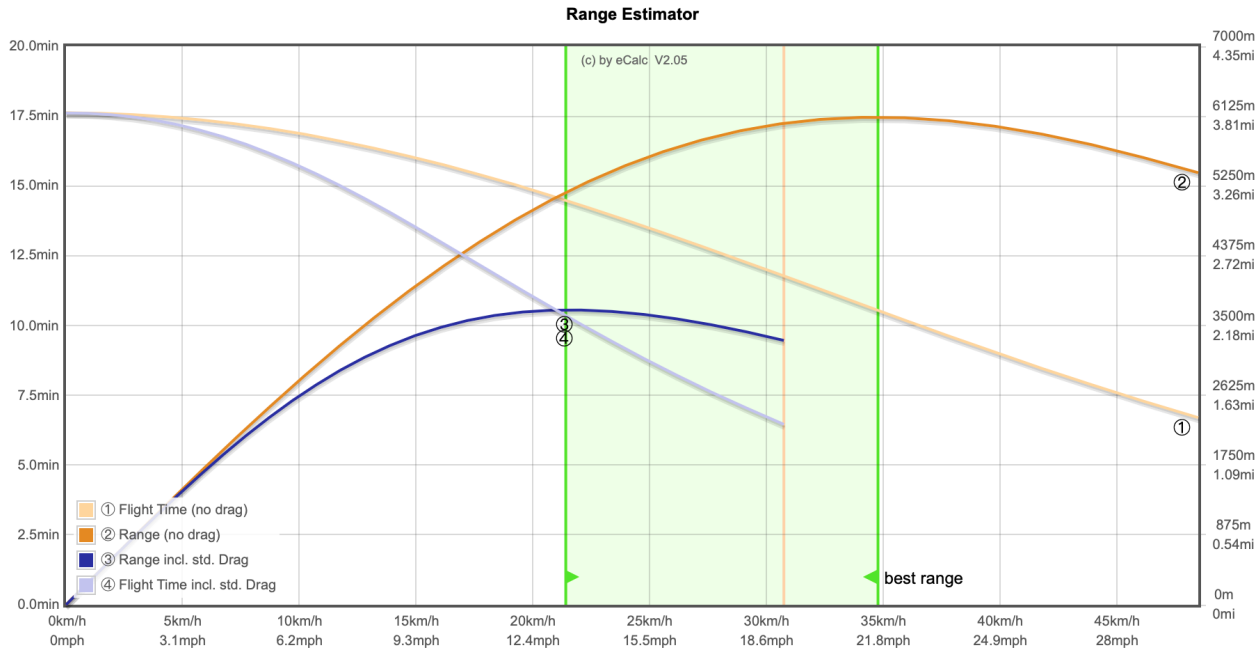


Figure 3.9: Drone flight time and range in function of speed

Chapter 4

Drone software, automation and simulation

This chapter will discuss the software of the project, including the future automation prospects and simulation thereof. Indeed, although this work focuses a lot on the selection and assembly of the hardware, the end goal of the project at large is to have the drone perform autonomously on stage. In order to do so, a robust controller architecture is needed, including captors sensing the quadcopter's position and a ground computer determining the desired trajectory and transmitting it to the drone.

4.1 Control loop for a remote-controlled drone

Since the end goal of the project is to automate the drone, the methodology employed will be as follows: first the control loop of a drone piloted by a human will be studied, then the various function performed by the pilot will be replaced by autonomous parts. Figure 4.1 shows the different pieces playing a part in the controlling of the drone. Those all play a different role and their interaction forms the command loop, described in figure 4.2.

First the sensors estimate where the drone is. Based on that estimated position, the control system compares where the drone is estimated to be with the position where he *wants* the drone to be. Where the pilot wants the drone to be is called the reference trajectory. Once the comparison is done, the pilot corrects the course of the drone: if its altitude is too low compared to the reference, he will increase the thrust, if the drone is leaning left, he will steer right etc... This correction is sent to the the drone via a transmitter, the remote. The remote sends the instructions to the drone via radio waves, and the drone interprets them and convert them in motor thrust. This interaction will be discussed in section 4.4.

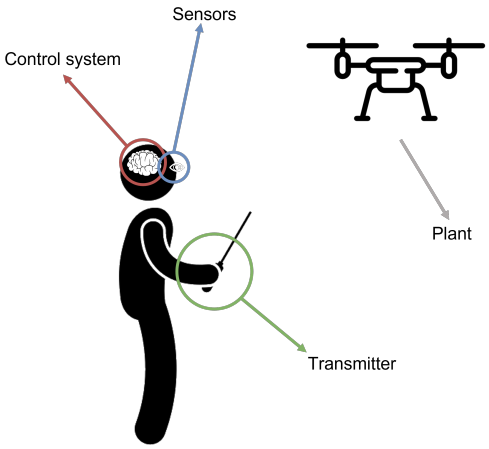


Figure 4.1: Drone piloted by a human

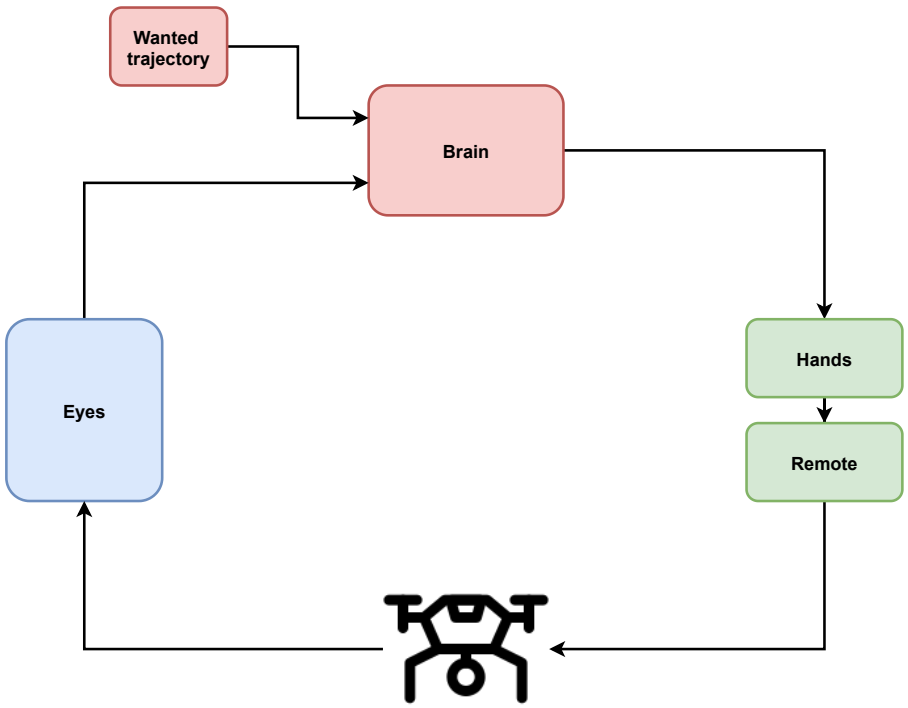


Figure 4.2: Control loop formed by the pilot and the drone.

In order to understand what happens between the drone and the pilot, it is useful to look into the steering variables. A drone remote does not control in position, but in angular acceleration and thrust. The angles whose reference acceleration is determined by the remote are called roll pitch and yaw, are described in figure 4.3. Thrust is the total upwards thrust exerted by all four propellers. A typical drone remote will be assigned as shown in figure 4.4, with the left joystick controlling thrust and yaw, and the right one controlling roll and pitch. However, for both automatic processes and human pilots, the reference trajectory and the estimate from the sensors are given in position. Therefore, when the control system compares the reference trajectory with the estimates from its sensors he needs to translate the error in position into an response in roll pitch yaw and thrust: for example, if he sees the drone is too much on the left, he needs to first roll towards the right, then increase thrust to move there. This translation is also a function of the control system.

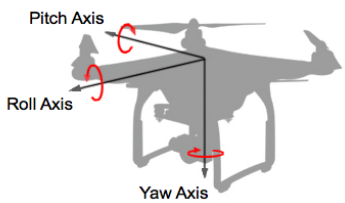


Figure 4.3: Roll pitch and yaw [29].

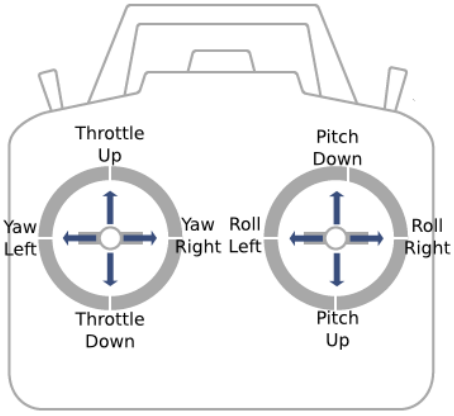


Figure 4.4: Joystick attribution for a typical drone remote [30].

4.2 Control loop for autonomous flight

After having reviewed the control loop of a drone with an human pilot, the automated loop can now be discussed. This loop is shown in figure 4.5, and was designed in collaboration with Gaëtan Permentiers, a student realising a master's thesis on this project as well. Here, the sensors are cameras, the control system a software and the emitter a ground station. The reference trajectory, in position, is generated by the user beforehand using Matlab. During flight, it is compared to the measured position estimated by the Qualisys Track Manager (QTM) software by extracting data from the video stream coming from the cameras. The cameras and the QTM software are discussed further in section 4.2.1. The comparison of the two positions, is then processed by the control system that generates an appropriate impulse and translates it in roll, pitch, yaw and thrust commands. The control system is explained in detail in section 4.3. The generated impulse is then translated in a MAVLink format readable by the radio transmitter, and then sent to the drone trough radio signal. The drone receive the instructions and converts them into thrust for each motor using the internal drone control loop described in figure 4.16. Note that the upper part of the loop shown in figure 4.5 all is located inside one computer, using three different softwares: QTM to process the camera stream and convert it into an estimated position, Matlab to implement the reference trajectory and the control system, and MissionPlanner for the ground station communicating with the external radio emitter.

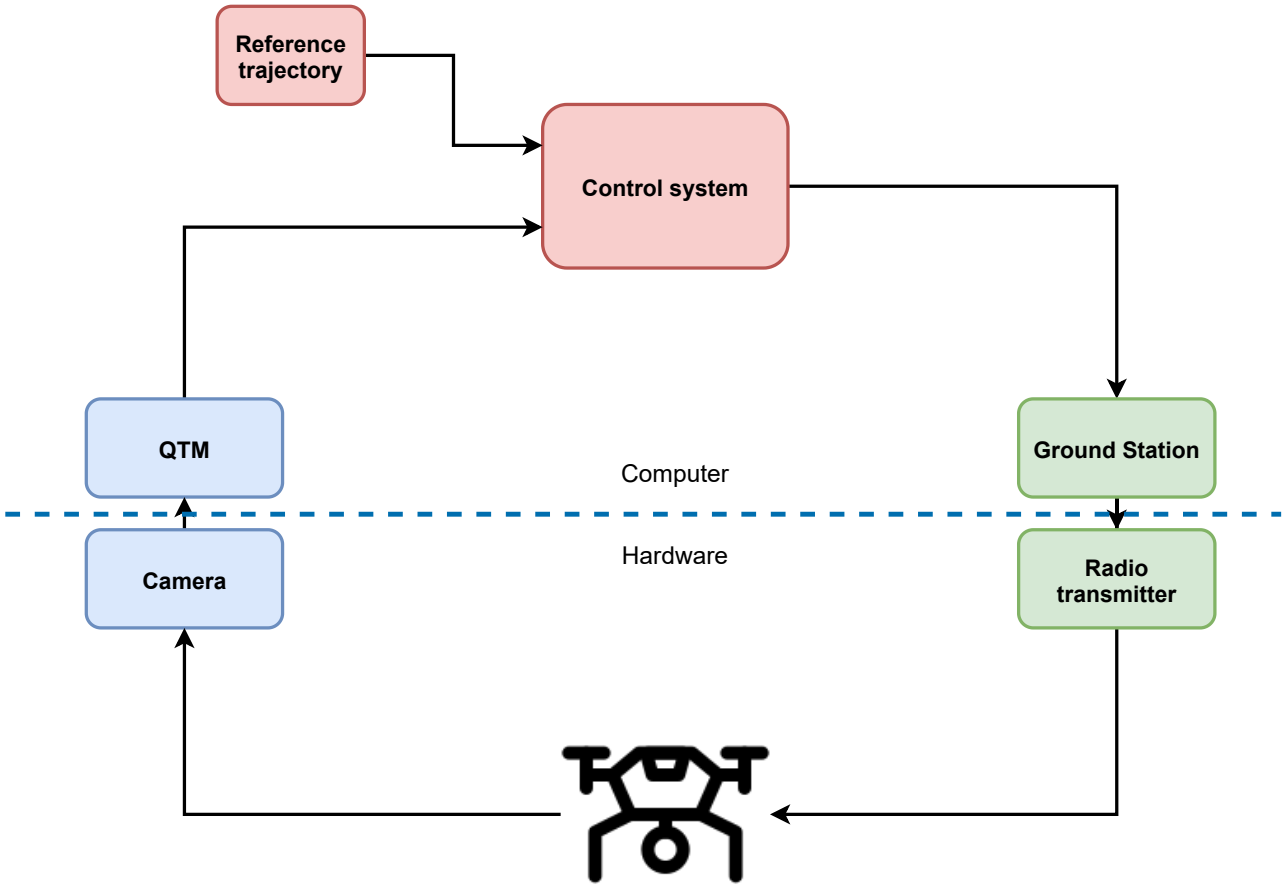


Figure 4.5: Control Loop for the automated drone

4.2.1 Qualisys motion capture

In order to detect the drone's position, it was opted to use the qualisys motion capture cameras. More precisely an array of twelve Arqus A5 cameras located at the ID2Move drone incubator[1]. Those cameras, shown in figure 4.6, are capturing 5 Mega Pixels wich will allow great precision, at a rate of 700Hz, allowing for low latency. Those cameras are paired with special markers shown in figure 4.7: The markers are made to reflect 850nm infrared light, that is emitted by LED's placed around the camera. This reflection is then caught back by the camera wich is outfitted with a special filter that only lets trough infrareds. This system makes it easy for the computer to detect the markers on the images as they are the only object that can clearly be distinguished.



Figure 4.6: Arqus A5 camera



Figure 4.7: Markers reflecting infrared light.

The data collected by the cameras is then treated by the Qualisys track manager (QTM) software, that computes the coordinates of every marker in 3D space. To do so, it uses a technique called stereophotogrammetry, illustrated in figure4.8. This techniques consists of drawing a line from the camera's focal point, trough the point on the image plane where the object was detected, and then computes the intersections between the lines formed coming from different cameras.

[1]https://www.id2move.eu/

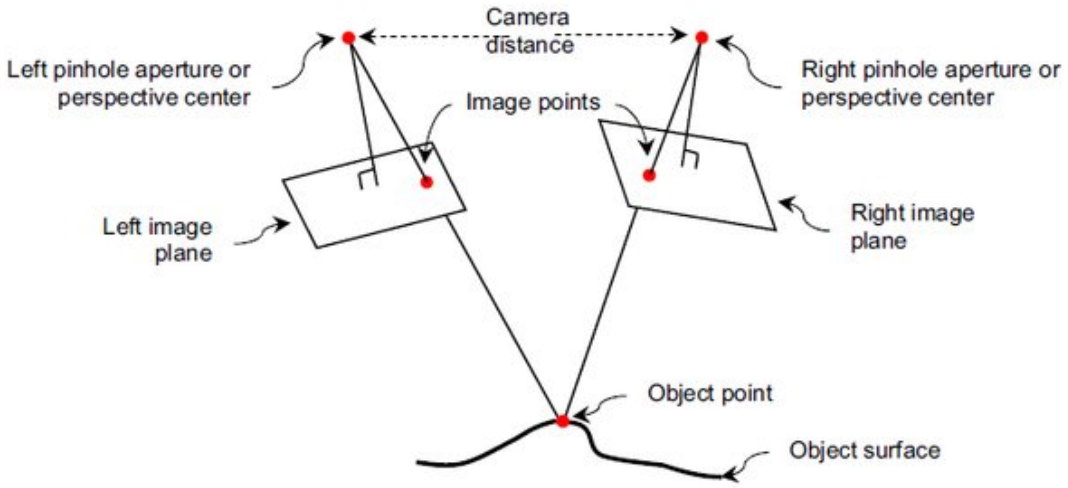


Figure 4.8: Stereophotogrammetry illustrated [31]

Before use, the system has to be calibrated. The calibration process consists of waving around a "T" shaped stick outfitted with markers. This allows the system to precisely know the orientation of each camera in the reference frame. The origin of the coordinate system is defined by a L piece placed on the ground during calibration, as shown in fig 4.9. In post-processing, the markers can be grouped as objects inside QTM, such as in fig 4.10, where they have their own coordinate system, which is ideal for this project as it lets us keep track of the drone's orientation.



Figure 4.9: L defining origin during calibration.

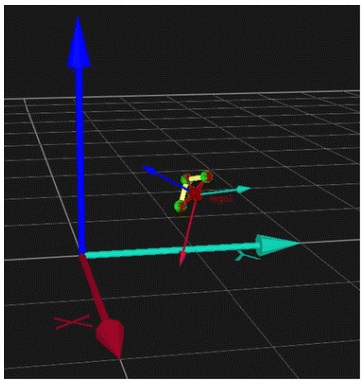


Figure 4.10: QTM object.

4.3 Flight control system

The flight control system is tasked with generating the instruction to send to the drone, taking as input the reference trajectory and the estimated position from the Qualisys software. It is adapted from Matlab's Quadcopter project [32], itself based on Sertac Karaman and Fabian Riether's work [33]. Before starting the explanation of the detailed system, it is useful to remind the working principles of Proportional-Integral-Derivative (PID) controllers, as many will be used for his realisation.

4.3.1 PID controllers

A PID controller is a control loop algorithm that works using feedback. It is one of the most common control method used in the industry, and serves to regulate all kinds of systems whose input needs to be modified continuously to react to changes. It controls the input of a system, such as the amount of hot air blown into the interior of a car for example, and try to attain a desired output, such as the interior's temperature. In order to do so, a PID algorithm evaluate the error, which is defined as the difference between the desired and the measured output, or the difference between the temperature indicated on the car's dashboard and the measured temperature in this example. They then multiply this error, it's integral and it's derivative by a proportional gain k_p, an integral gain k_i and a derivative gain k_d respectively. Finally they sum these three up and feed the result as input for the controlled system. The the effect of the different gains and their purpose is explained later in this section. The new output is then compared to the desired output to determine the new error, and the cycle continues. This procedure is illustrated in figure 4.11, where P is the system to control, y the output of this system, r the desired or reference output, e the error and u the input determined by the controller.

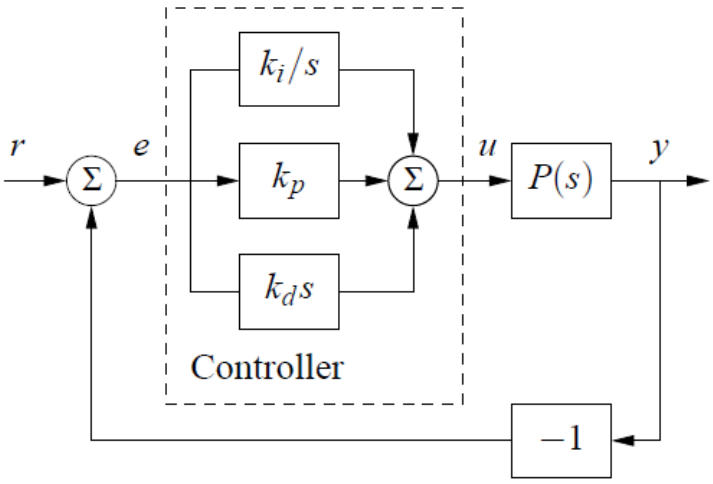
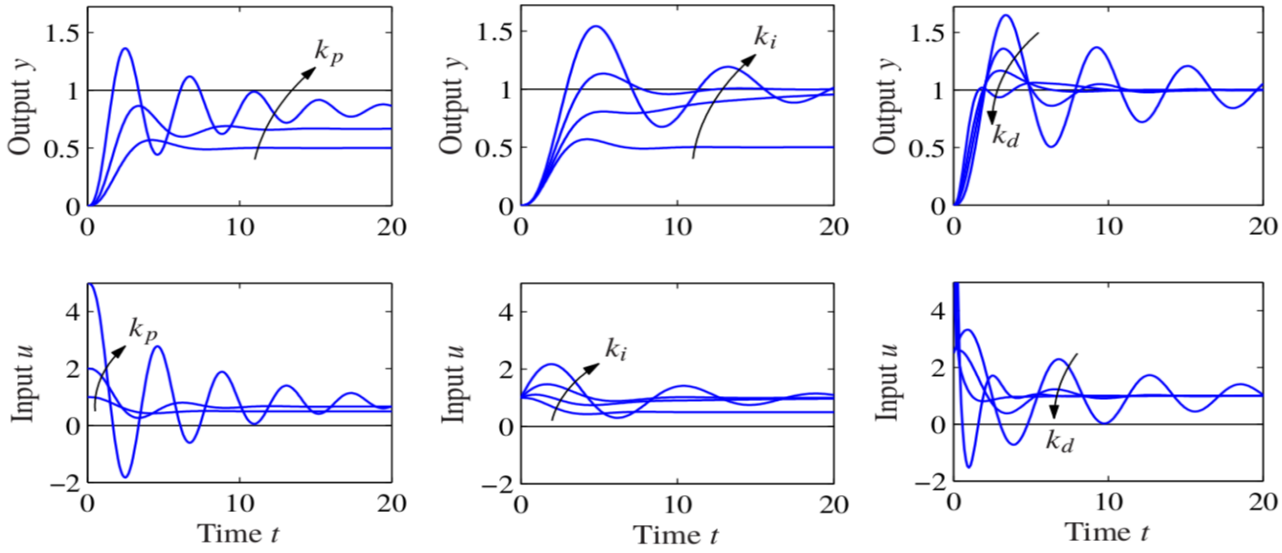


Figure 4.11: Block diagram of a closed loop system with a PID controller [34].

The right k_p, k_i and k_d parameters for a PID vary, and it is crucial to chose them well or the system might malfunction. The impact of those gains for a system whose reference went from 0 to 1 at time 0 is shown in figure 4.12. The response to this change is called the step response. Figure 4.12a shows the output y and the input u for a controller having only a proportional gain, also called P controller. As can be seen, a k_p gain too small means that the controller will never reach the desired output or take very long, while a gain too big means that the output will oscillate. Also, the controller might stabilise around an output that is different than the desired one: this is called steady state error, and is caused by constant disturbances. The example of a quadcopter's altitude controller illustrate this concept: If the altitude controller was realised with only a P controller, the drone's motors would stop when it has reached it's desired position: the error is then zero and therefore the input, which is just the error multiplied by a gain, will be as well. This will mean that the drone's motor stop, causing it's altitude to drop. Because of this, the error will increase and the controller will again activate the motors. But each time it reaches the desired altitude the same process will repeat. Therefore the drone will ultimately stabilise at an altitude inferior to the desired one, where the error with respect to the reference position times the k_p gain is just enough to offset gravity.

Figure 4.12b shows a controller with an integral term, added to avoid the steady-state error. Indeed in the quadcopter altitude controller example, the integral gain would correct the error induced by gravity: By integrating the error over time, the integral term would ultimately request an upward thrust that offsets gravity. And since the error is zero when the drone has reached the desired position, meaning the integral of the error does not change anymore, it would stay at that value.

Finally, figure 4.12b illustrates the step response for a PID controller, and shows the impact of the k_i gain. The role of this gain is to dampen the process, and thereby reduce oscillation in the output. By taking the derivative of the error term, it reduces the command u when the output is reaching the desired state too fast. By doing so, it prevents the system from significantly overshooting, thereby improving the controller's stability.



(a) P controller (b) PI controller (c) PID controller

Figure 4.12: Responses to step changes in the reference value for a system with a proportional controller (a), PI controller (b) and PID controller (c) [34].

4.3.2 Controller architecture

The role of the controller presented in this section is to ensure low-level control of the drone. This means it takes as input a reference position determined by the high-level controller and outputs commands for the motors. The high-level controller on the other hand dynamically adapts the drone's reference trajectory following changes in it's environment. High-level functions that could be implemented are for example following a specified object trough the room or obstacle avoidance. Those functions would determine the position where the quadcopter needs to be or in which direction he should point, and the controller described in this section would then bring it there. At this stage of the project however, since the environment in which the drone evolves is very predictable, the high-level controller simply remains a set of hard coded location. The structure of the control system is presented in figure 4.13.

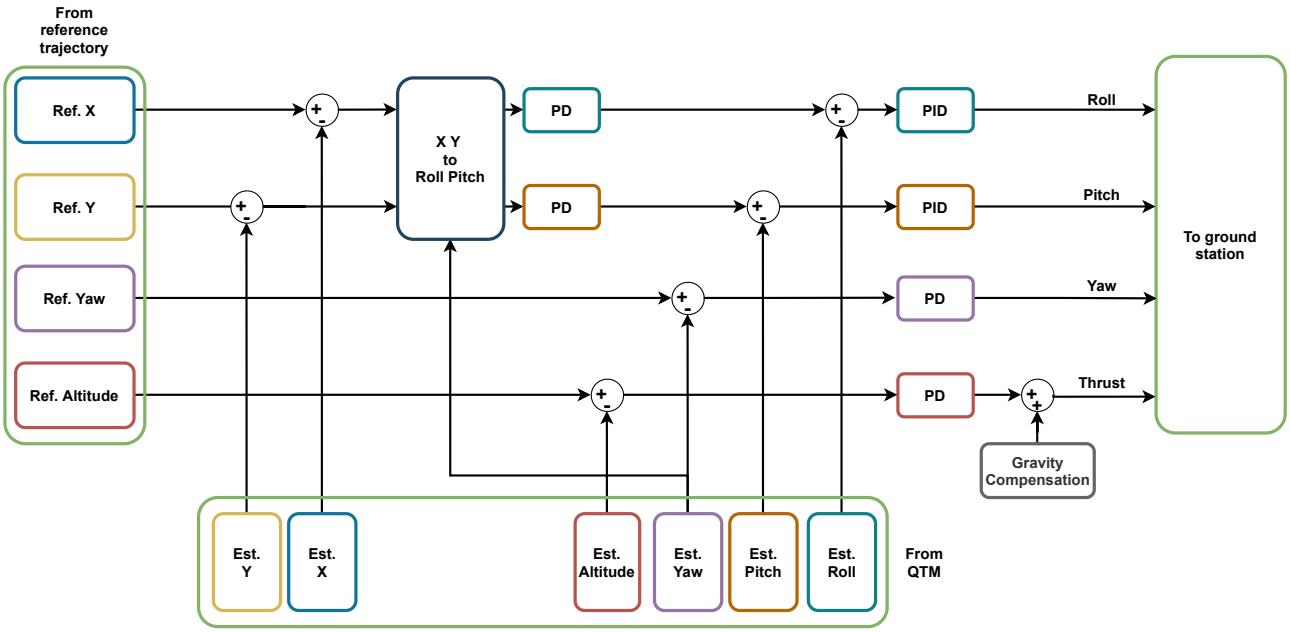


Figure 4.13: Diagram of the flight control system

The controller is composed of six PID's located into two loops. The outer loop compares the reference XY position with the measured one. It then rotates the frame with an angle equal to the yaw to get the errors in a new reference frame, X' Y', aligned with the drone:

$$\begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} X_e \\ Y_e \end{bmatrix} = \begin{bmatrix} X'_e \\ Y'_e \end{bmatrix} \tag{4.1}$$

The result is then fed into two PD controllers whose output is taken as reference roll and pitch for the inner loop. This inner loop compares those reference roll and pitch, along with the reference altitude and yaw, to the measured values. The error is then fed into four PID's that control the Roll, Pitch, Yaw and Thrust value sent to the drone via the ground station.

4.3.3 Position and altitude controllers

The fact that the output of the outer loop, an error in distance, is then fed into a loop controlling the roll and pitch angles, thereby correcting a distance with an angle, might be surprising for the reader. Indeed the loop ultimately only control the drone's angular orientation, but not his thrust. The control in thrust is in fact assured by the altitude controller. By ensuring that the drone stays at the same altitude even whens its orientation change, the altitude controller will provide the sideways thrust needed, as illustrated in figure 4.14. When the drone is not paralel to the ground anymore his thrust vector will no longer be normal with respect to the ground. This in turn means that drone will lose altitude, which will be corrected by the altitude controller by increasing thrust, thus creating a the sideways force needed to move the quadcopter in the desired direction.

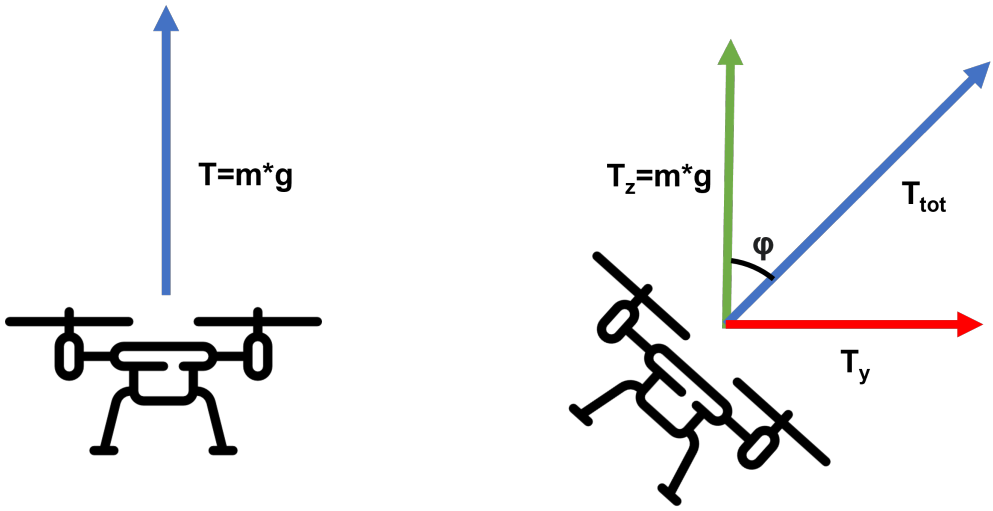
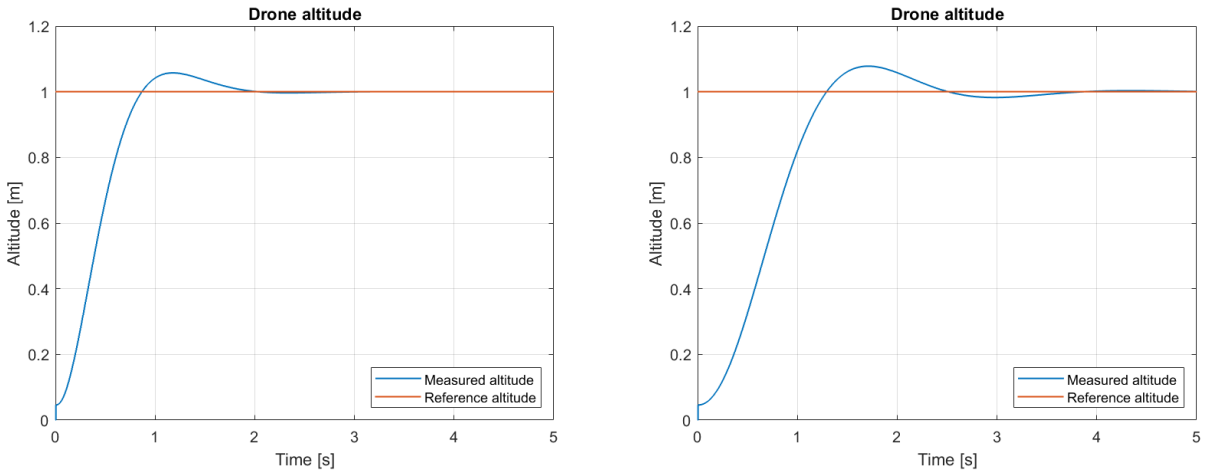


Figure 4.14: Effect of a roll angle ϕ on the quadcopter's horizontal thrust, when the vertical thrust is maintained constant by the altitude controller.

The altitude controller generating the thrust command is fed forward with a compensation of gravity, which permits a simplification of the PID in a simpler PD controller, since the static error is eliminated by this feedforward. figure 4.15 compares two simulated parrot drones, starting from the ground and trying to reach 1m of altitude, one using a gravity feedforwad with a PD as altitude controller, and the other using a PID: the use of the feedforward instead of an integral terms provides faster response time, less overshoot and less oscillation, and is thus preferred. This method requires to know in advance the exact weight of the drone and its payload, while the integral term is more versatile, but the weight is well known in this project and it is therefore not an issue.



(a) With gravity feedforward

(b) With integral gain

Figure 4.15: Reference vs simulated drone altitude

4.4 Internal drone control

The drone itself also has his own control loop, called the internal loop and looking as shown in figure 4.16. It is present in both the autonomous and remote controlled case. This loop take the roll, pitch and yaw angular acceleration and the thrust sent by the remote as reference values, and compares them to the values measured by the drone's internal sensors, a gyroscope in the simplest case. The difference between measured and reference angular acceleration is then fed to PID controllers, that determine the optimal impulse to send to the motors to correct the errors. Those PID's are tuned by the constructor, but they can still be modified aftherwards if needed. The impulses, together with the reference thrust which is directly fed, are sent to the motor mixing algorithm. This algorithm translates the impulses in roll pitch yaw and thrust into thrust to be generated by each specific motor of the drone, and will be discussed below. It is also worthwhile to mention that some drones have much more complicated control loops featuring a host of other sensors making it easier to pilot the drone, but the loop is kept as simple as possible here.

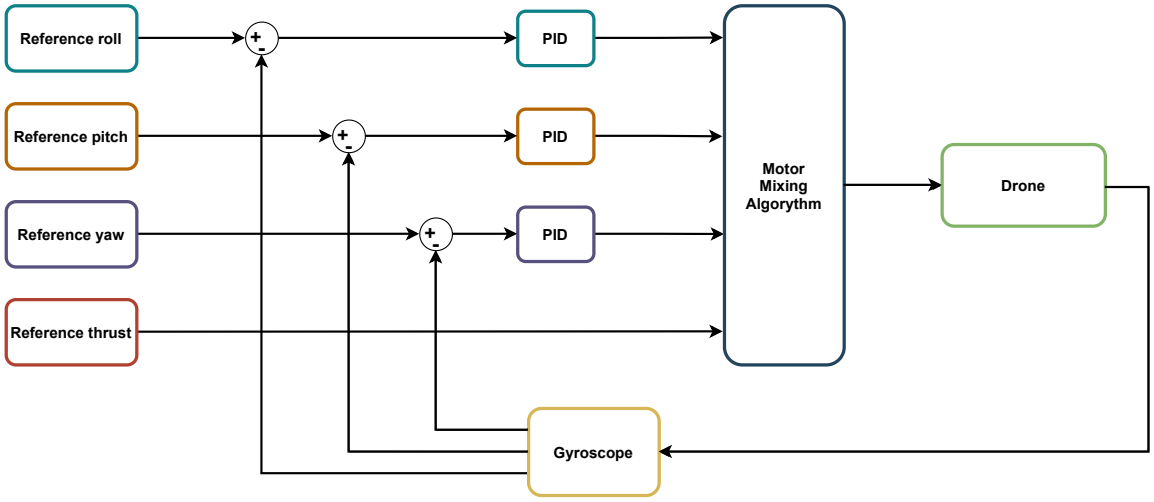


Figure 4.16: Internal drone control loop

To understand the motor mixing algorithm, one must know the configuration of the quadcopter's motor/propeller pairs, shown in figure 4.17. Then, one can infer the relation between the thrust in each individual motor and the resulting roll, pitch, yaw and total thrust: First, the total thrust T is equal to the sum of the thrust generated by each motor individually:

$$T = M_1 + M_2 + M_3 + M_4 \tag{4.2}$$

Then, the roll angular acceleration ϕ'' will be the result of a difference in thrust generated by the motors of one side compared to the other: if the motors placed left of the drone provide more trust than those placed on the right, the drone will roll clockwise, and it will roll counterclockwise otherwise. Therefore equation 4.3 can be written. Note that for this equation to be rigorously exact, torques caused by external forces should be substracted to the motor thrusts, and then they should be multiplied by the moment of inertia around the roll axis and divided by the distance in X between the motor and the center of mass. However since those parameters are mostly constant in time, they are not needed for the algorithm and will therefore not be written in order to simplify the following equations.

$$\phi'' = M_2 + M_3 - M_1 - M_4 \tag{4.3}$$

The same reflection is also valid for the pitch angular acceleration θ'', except now it is the front motor thrust compared to the rear motor thrust: If the rear provides more thrust, the quadcopter will tilt forward, in the negative direction.

$$\theta'' = M_1 + M_2 - M_3 - M_4 \tag{4.4}$$

Finally, for the yaw angular acceleration ψ'', one must take into account the direction of the propeller's rotation: M_2 and M_4 rotate clockwise, while M_1 and M_3 rotate counterclockwise. Therefore, if M_2 and M_4 provide more thrust than M1 and M_3, the quadcopter will rotate counterclockwise, in the negative direction.

$$\psi'' = M_1 + M_3 - M_2 - M_4 \tag{4.5}$$

Once the system formed by equation 4.2, 4.3, 4.4 and 4.5 is known, it can be inverted to form the motor mixing algorithm:

$$\begin{bmatrix} T \\ \phi'' \\ \theta'' \\ \psi'' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} \tag{4.6}$$

And the motor mixing algorithm finally becomes:

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} T \\ \phi'' \\ \theta'' \\ \psi'' \end{bmatrix} \tag{4.7}$$

The obtained matrix makes sense intuitively: The motor 1 contributes positively to thrust, such as all motors. He contributes negatively to roll since he is located on the left side. He contributes positively to pitch since he is in the front. And finally, he contributes positively to yaw because he rotates counterclockwise. A similar reflection can be held for each individual motor to obtain the full matrix.

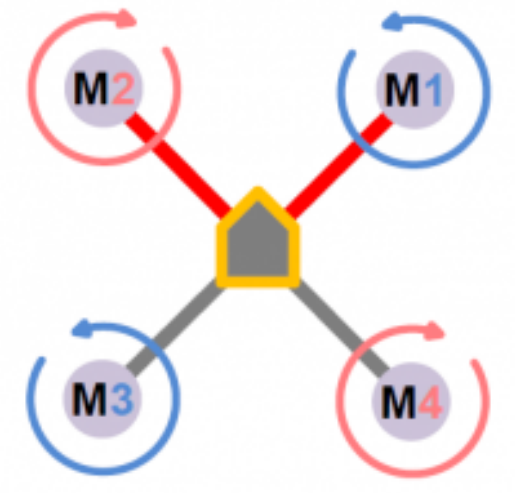


Figure 4.17: Configuration of a quadcopter's motors [35].

4.5 Simulation

To complete the flight controller, the only thing still needed is to tune the gains of our six PID controllers. In order to do so, an idea is needed of how the system will behave. One method would be to try out different values on the real drone and see what the results are. However, this would be way too risky for the drone and it's surroundings, and would most probably mean partial or total destruction of the quadcopter. Another way of estimating the drone's behaviour is thus needed. This is done via a simulation. The goal of this simulation is not to exactly determine the best controller gain, but to give a good enough estimate so that the controller can be put on the real drone without risking to destroy it. From there, the fine tuning can be done on the hardware.

4.5.1 Choice of simulation software

Multiple software are capable of simulating a quadcopter in flight, with each their different specificities, advantages and drawbacks. The challenge for this work is to simulate the drone while it carries the arm described in chapter 2, which will modify its behaviour. The goal is also to simulate short-distance flights instead of long-distance ones, as the application requires the drone to stay on the relatively small scene. Furthermore, since the QTM software used to extract data from the cameras is Windows based, it is preferable to have the simulation running in Windows as well. Now that the selection criterion are clear, the principal simulation softwares can be reviewed:

- **Gazebo:** A simulation environment for mobile robots, Gazebo features a powerful physics engine and great versatility. It allows the creation of custom models, which is perfect for the drone-arm simulation. However, Gazebo is Linux based, wich makes it hard to use for this project

- **JMAVSim:** Is a simple multirotor/Quad simulator that allows to fly copter type vehicle running PX4 around a simulated world. It is easy to set up and can be used to test that vehicles can take off, fly, land, and responds appropriately to various fail conditions [36]. Unfortunately, it is only made to simulate PX4 based vehicles without ground station and it does not support custom geometries as needed for this project.

- **AirSim:** A simulator based on Unreal Engine, it requests a lot of processing power, but is ideal to simulate camera vision thanks to its high visual accuracy. However since there is no need to simulate the cameras directly for this project, it is needlessly computational heavy.

- **MissionPlanner:** A simple and easy to use trajectory simulator aimed at simulating longer distance flight. It offers good compatibility with ground stations, but not a precise enough control for this application.

- **Simulink model:** Not a simulator on its own, it allows to program an "Home-made" simulation. This has the advantage of offering endless customisability, at the cost of high complexity and more work for the user. As a Matlab application, if offers plugins for various applications. The block-based design improves readability.

Seeing those options, it was decided to opt for a custom-made simulation in Simulink. Fortunately, examples of such simulations already exist, and bar some modifications could be adapted to this project. The base model picked for this work is called "Quadcopter Project" [32], and it part of the examples included in Simulink's Aerospace blockset[2]. It is based off the work from Pounds et al[37], and simulates a Parrot Mambo drone. A general view of the main blocks composing this simulation can be seen in figure 4.18.

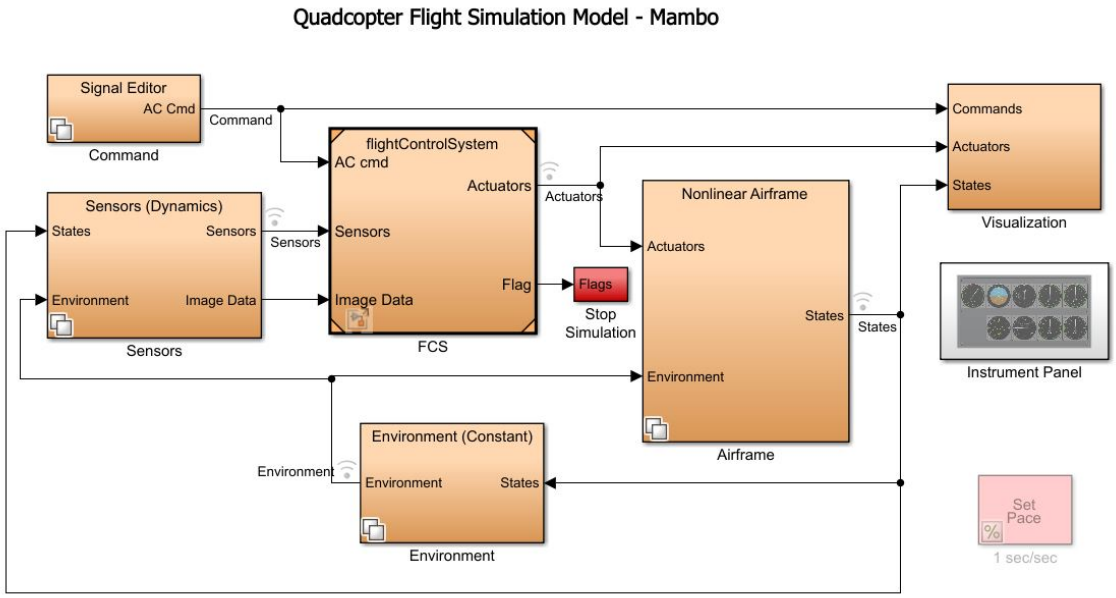


Figure 4.18: Blocks of the original quadcopter flight simulation model.

First, the command bloc issues position commands. Those can be specified beforehand by the user, in the form of spreadsheets or matlab data, or even issued in real time via a joystick connected to the computer. That way, different trajectories can be simulated, from very simple hovering to more complex tasks. Those commands are then fed in the flight control system block. This block contains the controller, similar to the one described in section 4.3, that needs to be validated. It takes as input the data sent by the simulated sensors in the sensors block, and outputs commands for the motors. Once those commands are generated, they are sent to the Airframe block. This block is composed of two subsystems. The first one models the motor-propeller pairs and converts the commands sent by the flight control system in forces and torques acting on the drone. It does so by using a model developed by Pounds et al [37], applying the environment variables stored in the environment block such as air density, temperature, pressure and gravity to the fluid's mechanic equations. The second subsystem then converts those forces and torques into quadcopter displacement, modelling the drone as a rigid body. Two version of this block exist, one non-linear model for maximum accuracy, and a linear one, to tune the PID controllers mathematically using Simulink's PID tuning function. The resulting drone displacement and speed is then fed back to the block modelling the sensors, and the cycle continues. During the whole process, the visualization block is showing the displacements of the drone visually using Simulink 3D, as shown on figure 4.19.

[2]https://nl.mathworks.com/products/aerospace-blockset.html

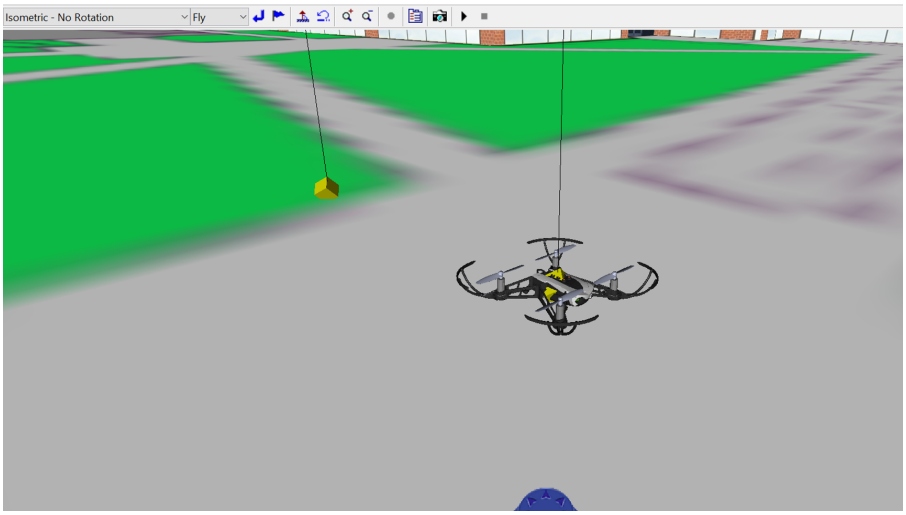


Figure 4.19: Drone as shown by the simulation.

4.5.2 Modifications to the template

In order to use the simulation for this project, some modifications had to be made. Those modifications fall into two categories: First the structural modifications, where the model itself had to be changed. Then the parameter modifications, where the values used for the equations are replaced to correspond to the actual quadcopter and not to the Parrot Drone. This section will discuss those changes.

Amongst the changes that needed to be made in the structure, the biggest one was the simulation and use of sensors. Indeed the original simulation use a variety of on-board sensors, and blend their data together in order to get a position estimate, and this project uses external Qualisys cameras with the QTM software that already performs position estimation. Therefore, the sensors block was replaced by a simple feed-trough of the position calculated by the Airframe block, with noise added if needed to test the robustness of the controller. This is enough to simulate the Qualisys system from the point of view of the controller, since the external QTM software already performs position estimation. The position estimator block, that was blending together the info of the different sensors, also got removed. After this was done, the landing logic had to be removed: The original file had a different control system using different sensors when it was close to the ground, and this does of course not apply to this project.

When those changes where realised, a small test was done in order to test the adapted simulation: The reference trajectory was set at a constant $\begin{bmatrix} X & Y & Z & \phi \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & 0 \end{bmatrix}$, in order to get the drone to hover one meter above the ground. The test showed that the origin of the world reference frame was not positioned at the start point of the simulated quadcopter, causing the drone to shoot towards the reference frame's origin at the start of the simulation. To compensate for this, the X and Y parameters of the drone's position fed trough to the control system to simulate the cameras are derived then integrated. This makes sure that the drone always sees his starting position as the himself as starting at $\begin{bmatrix} X & Y \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$. After this modification was made, the drone was hovering nicely one meter above it's start location, as requested. The origin of the reference frame was then modified and the derivation-integration removed to avoid errors. The next step after having modified the structure of the simulation, was to modify the parameters used to simulate the drone. The default parameters used to simulate the Parrot drone are shown in table 4.1.

Parameter		Value	
Moments of Inertia		$\begin{bmatrix} 5.82857 \times 10^{-5} & 0 & 0 \\ 0 & 7.16914 \times 10^{-5} & 0 \\ 0 & 0 & 1 \times 10^{-4} \end{bmatrix}$	
Mass	0.063[Kg]	d	0.0624[m]
h	-0.015876 [m]	reference drag area	$7.853 \times 10^{-5}[m^2]$
c_d	0	number of blades	2
rotor blade mass	3.75×10^{-4}[Kg]	rotor chord	0.008[m]
rotor radius	0.033[m]	rotor C_t	0.0107
rotor θ_0	$14.6°$	rotor θ_{tip}	$6.8°$
rotor lift slope	5.5	Motor command to ω^2 gain	13840.8

Table 4.1: Simulation parameters for the Parrot drone

In order for the simulation to provide relevant results, those parameters needed to be adapted to this project. However they are not all straightforward, and many needed to be computed. The first parameter, the matrix of moments of inertia, is not easily measurable on the physical system. In order to evaluate it, a CAD model of the drone and the arm was made in Solidworks, and the weight of each specific part was specified. This model is shown in figure 4.20. Having this model meant that the Solidworks Mass Properties tool could be used to compute the matrix of moments of inertia at the center of mass. The result are given in equation 4.8 (moments of inertia in $[Kg * m^2]$). As expected this matrix is several orders of magnitude above the one used to simulate the Parrot drone. Computing this also gives the d and h parameters: h=32.55[cm] is the eight of the propellers above the center of gravity, and d=35[cm] is the drone's arm length, determined during construction.

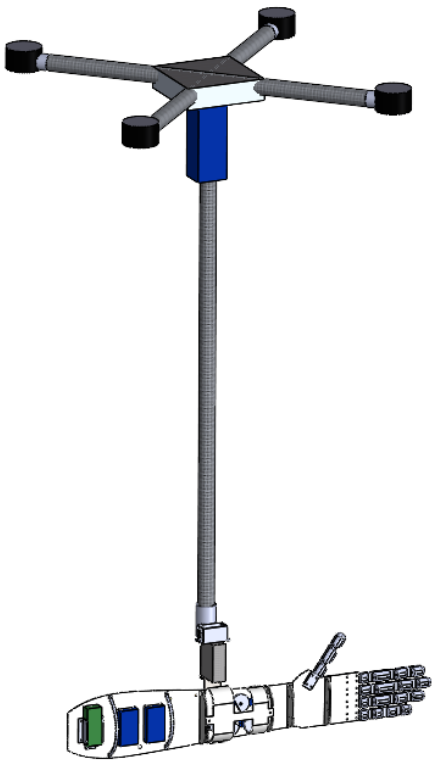


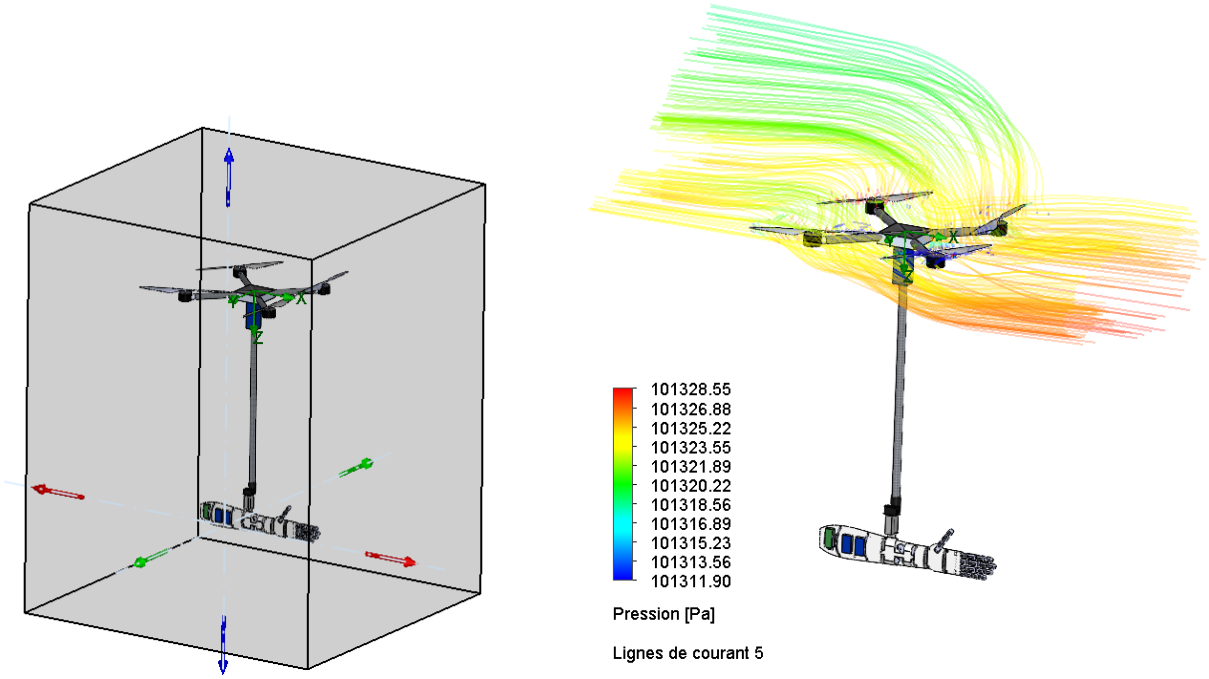
Figure 4.20: Solidworks model used to compute the moments of inertia .

$$I = \begin{bmatrix} 1.097 & 1.711 \times 10^{-4} & 1.135 \times 10^{-2} \\ 1.711 \times 10^{-4} & 1.123 & 2.344 \times 10^{-2} \\ 1.135 \times 10^{-2} & 2.344 \times 10^{-2} & 2.667 \times 10^{-1} \end{bmatrix} \tag{4.8}$$

Measuring the drone mass is straightforward: A simple scale was used to weight the drone, the battery and the arm, that gave a combined weight of 4267[g]. However, knowing the drone's reference area and drag coefficient c_d requires specific computations. In order to compute those, Solidworks Flow Simulation was used on the model, as shown in figure 4.21. The simulation's goal was to compute the drag force F_d for air blowing at 1 [m/s] in the X direction (from the drone's front to the back) while the propellers where spinning at 3000 [RPM]. The computational domain is shown in figure 4.21a and the result was $F_d = 0.013[N]$. This very low result, obtained for a speed that is around the expected speed for the usage made of the drone in this project, means that the drag force can be neglected since it is several orders of magnitude smaller than the force exerted by gravity. The original simulation also neglected the drag, since the used drag coefficient is zero. Nevertheless, the drag coefficient will still be computed for completeness, and for possible higher speed use of the drone. The drag equation is given by:

$$F_d = \frac{1}{2}\rho u^2 c_d A \tag{4.9}$$

Where ρ is the fluid's mass density (in this case air), u is the fluid's speed with respect to the drone, and A the reference area. Taking this area as the orthogonal projection of the system in the X direction gives $A = 940[cm^2]$. Combining the simulations results and equation 4.9 then results in $c_d = 0.226$.



(a) Computational domain

(b) Flow lines around the propellers

Figure 4.21: Solidworks Flow Simulation

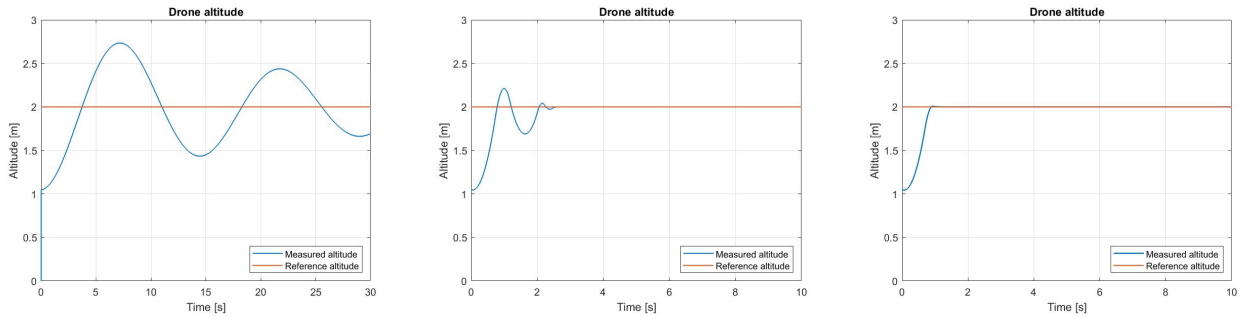
The number of rotor blades, the blade mass, the blade chord, the blade radius, the torque coefficient C_t, the blade angle at the rotor's base θ_0, the blade angle at the rotor's tip θ_{tip} and the rotor lift slope can be directly measured, taken from the propellers' datasheet or are similar to the values used by Pounds et al [37]. the motor command to squared speed gain can be derived from the motor's datasheet. Finally, the values used for the simulation are given in table 4.2.

Parameter		**Value**	
Moments of Inertia		$\begin{bmatrix} 1.097 & 1.711 \times 10^{-4} & 1.135 \times 10^{-2} \\ 1.711 \times 10^{-4} & 1.123 & 2.344 \times 10^{-2} \\ 1.135 \times 10^{-2} & 2.344 \times 10^{-2} & 2.667 \times 10^{-1} \end{bmatrix}$	
Mass	4.267[Kg]	d	0.35[m]
h	-0.3255 [m]	reference drag area	$9.4 \times 10^{-2}[m^2]$
c_d	0.226	number of blades	2
rotor blade mass	4×10^{-2}[Kg]	rotor chord	0.058[m]
rotor radius	0.231[m]	rotor C_t	0.0047
rotor θ_0	$13.7°$	rotor θ_{tip}	$4.4°$
rotor lift slope	5.5	Motor command to ω^2 gain	622.7

Table 4.2: Simulation parameters for the project's drone

4.5.3 Controllers tuning

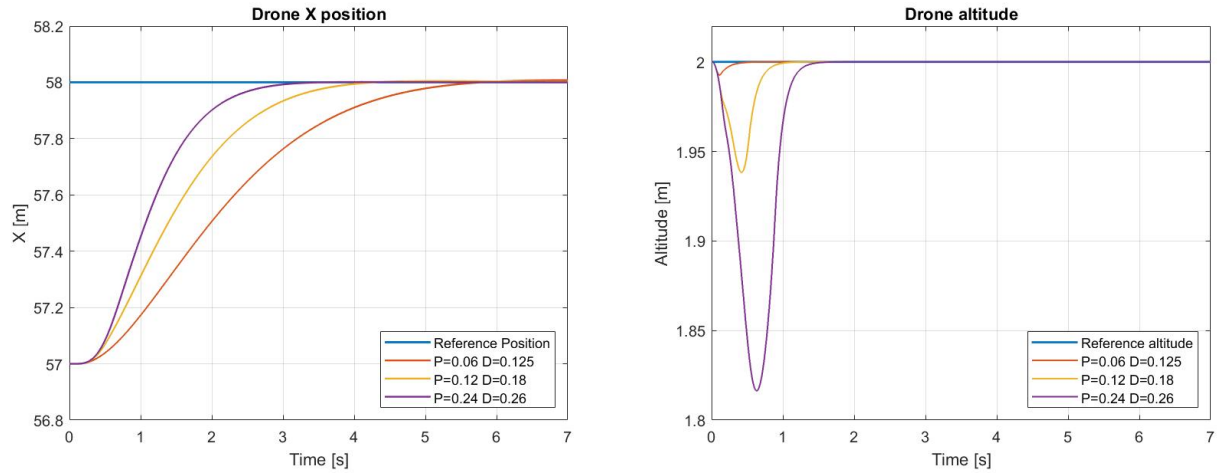
Having adapted the simulations to this project's specifications, it can now be used to adapt the controllers described in section 4.3.2. First, the simulation is used in linear mode, to be able to generate fitting PID gains with the Simulink PID tuner feature. The simulation is then be used in the more realistic non-linear mode, and the generated gains are fine-tuned. The first controller to tune is the Altitude controller, as he is independent from the others but is crucial for the position controllers to work. The gravity feedforward of section 4.3.3 is kept in use with the updated mass. The tuning process is illustrated in figure 4.22. First the base gains used for the Parrot drone are kept. The results are shown in figure 4.22a, and it is clear they are not adequate: the drone keeps oscillating around the requested altitude, and never stabilises. Seeing this, new gains are generated using Simulink PID tuner on the airframe linearized model. The resulting step response, computed using the non-linear model, is shown in figure 4.22b. This result is much better, but still not optimal: the drone overshoots the desired altitude at first, then undershoots to correct. Therefore the gains are further tuned by trial and error, using increments of 20% on the base Simulink-generated gains. The final response, shown in figure 4.22c is adequate: the overshoot is very small and the response time is less than a second.



(a) Parrot base gains (b) Gains generated by Simulink auto tune (c) Fine tuned gains

Figure 4.22: Altitude controller step response

The same process is then repeated for the yaw, pitch and roll controllers. In order to do so, the inner loop of the controller (shown in figure 4.13) is disconnected from the outer loop, and the reference position is given directly in roll and pitch instead of X Y. The resulting step response are shown in appendix D. After having tuned the inner loop, the outer loop X Y controllers can be tuned. Since they depend on the altitude controller to function, as explained in section 4.3.3, their performance is a trade-off between quick lateral displacement at the cost of a momentary drop in altitude, or slower movements with a smaller altitude drop. This phenomenon is shown in figure 4.23, where three different set of P and D gains are tested for a forward displacement of one meter, and the corresponding altitude drop is measured.



(a) Forward displacement time (b) Altitude drop during the displacement

Figure 4.23: Three different sets of P and D gains tested for a forward displacement of 1 [m].

Ultimately, the right gains for this controller depend on the usage. For the time being, the gains used will be $P = 0.12$ and $D = 0.18$ as they offer a good balance between an altitude drop of less than 7 [cm] and a step response time of about four seconds, but this can of course be changed depending on the project's needs. The X and Y position controller are nearly the same, and so are the roll and pitch controllers. They work the same way, on axes that have similar moments of inertia. Therefore the gains developed for one can be used for the other with minimal adaptation, which saves tuning time.

Chapter 5

Results

This chapter will describe and analyze the results of this work. What worked, what did not and improvements that could be made. First, the arm's movements will be compared to the concept animations created by the artist. The timings, robustness and responsiveness will be evaluated. Secondly, the drone and arm combination will be discussed by reviewing a test flight. The perturbations by the arm movements on the drone's flight and the influence of the drone flight's on the arm's movements level will be the main points of focus. Finally, the flight controller and groundwork for future automation prospects laid by the simulation will be addressed.

5.1 Arm animations

To test the arm's capabilities, three short sequences of movements where designed by the artist. Those animations are a marching sequence where the arm moves up and down to mimic a marching soldier, a game of rock paper scissor and a slap followed by a pointed finger moving up and down signifying "no no".

In order to test those animations in conditions that are as close to the actual show as possible, they where tested in a dark room next to the body of El Captain that will be used for the artistic presentation. As shown on figure 5.1, the character is moved by hand, and relies on light effect to fully manifest it's expressive potential. The animations where tested on the hand-held arm and the results filmed for later analysis. The results of this analysis are discussed in this section.



Figure 5.1: El Captain setup



Figure 5.2: Marching sequence.

The first animation, shown in figure 5.2 consists of simple up and down movements of the arm in sync with the up and down leg movement of the character, to mimic a soldier marching. The goal of this is to see the synchronisation of the body and the arm in a basic movement. The result is satisfactory, as the impression of marching is convincing. But the marching speed is limited by the speed of the elbow, and the arm is unable to move up and down at more than 1 [Hz]. This limits the artistic freedom for fast and aggressive movements.

The next trial animation consist of a rock paper scissor game ending in a pointed finger symbolising a gun. The goal of this sequence of movements is to observe the synchronisation of the fingers with the elbow to assess the rhythmic capacities of the arm in relation to the rhythm of the game. The result is shown in figure 5.3. Although the goal of synchronising fingers with the elbow was attained, the system showed it's limitations: Due to the replacement of the elastics explained in section 2.1.3, the servos used to actuate the fingers where not dimentionnned correctly any more, and stuttering occurs when they are activated together in quick succession. Nevertheless, excluding the stuttering, the animation was a success and the meaning behind the gestures could be understood.



Figure 5.3: Rock Paper Scissor.

Finally, the last animation featured a slap followed by a pointed finger waving up and down, expressing disapproval, as shown in figure 5.4 . The goal was to test interactions of the arm with other elements. This time, El Captain's body was not used as the owner of the arm, but as the recipient of the slap. The result was convincing in terms of gestures, but showed how hard it was for a human operator to play along with a robotic actor. Indeed, the timing of the machine can be hard to predict, and it is unable to correct it's trajectory to hit the intended target: The human operator has to place itself right in front of where the arm will hit, which is also a difficulty.



Figure 5.4: Slapping animation.

Overall, the obtained results where promising, and the expressive potential of a robotic arm for puppetry was validated, especially with the support of proper light play and sound. However, the arm's robustness leaves room for improvements, mainly because of the servo's stutter and Wi-Fi reliance of the ESP32. Indeed it was conceived as a stand-alone prototype, in order to estimate the weight and movements that the drone would need to support, as it was made before it. It also served evaluate the expressiveness potential of an elbow and fingers only, with less degrees of freedom than an human forearm. It succeeded in both missions, but with the low robustness expected of a prototype. After having tested the various capabilities of the arm, it was also noted that the dynamism of the animation was limited due to the maximum speed of the motors. Faster motors would open up more artistic possibilities.

The way to improve those flaws would be to keep the successful parts of the design but to replace the lacking ones, while combining the drone and arm in to one system, which is now possible thanks to the realisation of the drone. That way, the 3D printed fingers with their elastics and the Pan-Tilt mechanism used for the elbow would be kept, as they have proven to be robust and meet the needs of this project. The ESP32, the servos actuating the fingers and the arm's battery however should be replaced. Removing the ESP32 and using the drone flight controller to direct the arm instead would allow for a simpler, more robust control and a better synchronisation between the arm and the quadcopter. Likewise, the servos could use the drone's power supply to function, which would remove the bottleneck of any of the two system having less autonomy than the other. Finally, the servos used for the fingers should be replaced by more powerful actuators in order to handle the fingers with their new elastics and provide faster, more fluid movements.

5.2 Test flight

After the completion of the drone, the combination of the arm and the drone could be tested. The test flight was performed at the location of RCTakeoff's store, and the pilot was the drone's builder Michaël Daris. A picture of the setup is shown in figure 5.5. The goal of the flight was to test whether the arm hampered the drone's movements and if the drone's movements would reverberate on the arm. The former proved to not be the case: the arm's movements have minimal impact on the drone thanks to the drone's high inertia, the flexible junction acting as a damper, and the fact that the elbow is located close to the arm's center of gravity, thus limiting the torque caused by the arm's movements. However, the flexible junction that dampens the arm movements can also cause the arm to oscillate if the drone is swung repeatedly left and right as in figure 5.5b, thus creating a pendulum effect. Should the issue persist, the junction would need to be tightened to reduce the flexibility, and by extension the oscillations.



(a) Normal flight

(b) Oscillation

Figure 5.5: Drone and arm in flight.

The test flight was also the opportunity to analyse the noise produced by the drone during flight. Unfortunately, the smartphone microphone used to record the sound was not callibrated, and therefore the absolute value in decibels are not available, but relative values can nevertheless be obtained. The relative power spectrum estimate, visible in figure 5.6 shows that the power emitted by the propellers is stronger at about 100[Hz], with smaller peaks around 200 and 300[Hz]. This means that the drone's sound is more bearable for the human ear, as lower frequencies are perceived as quieter [38].

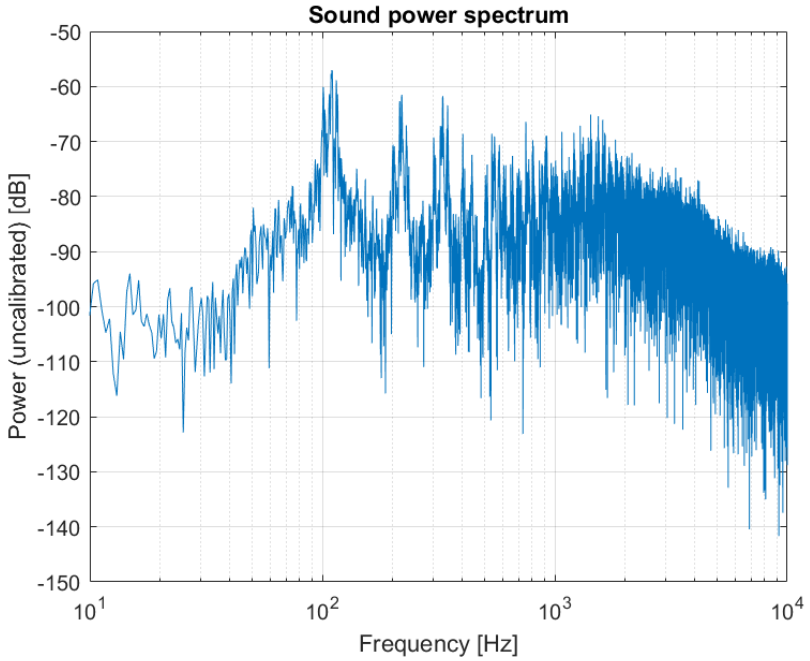


Figure 5.6: Spectogram of the relative sound power emited by the drone in flight

5.3 Controller and simulation

In order to test the controllers developed in section 4.5.3, and the behaviour of the drone, a simulation was generated with a more complex trajectory. The drone is tasked to follow a square trajectory, spending five seconds on each side of the square, at different reference speeds. The result of the first test, with a reference speed of 1[m/s], is shown in figure 5.7. The trajectory is well followed by the drone, with slight overshoots caused by the momentum during the changes of direction. Some drops in altitude also happen around those points, caused by the re-orientation of the drone. Those errors are accentuated by a faster reference speed, as shown in figure 5.8, where the reference speed is 3 [m/s]. The errors caused by the momentum during turns are accentuated, and the altitude is changing more as the controller needs to be more aggressive to counteract the rapid changes in direction. Those errors could be attenuated with a more elaborate reference trajectory where the required speed drops around the corners, so the drone is not at full speed when it needs to turn, but this is not the goal of the present simulation, where the aim is to see the reactiveness of the quadcopter to commands fed in real-time. The results in that regard are satisfactory, since the intended use of the quadcopter does not require speeds of more than 1[m/s], and the controller could be exported to the real drone. The simulation is however not a perfect mirror of reality. The strongest hypothesis made for it's realisation is that the drone and the arm act as a rigid body. This is not the case in reality, as the junction between the drone and the rod holding the arm is flexible. This means that the simulation will overestimate the torque needed for the drone to pitch or roll, while not taking into account the potential effect of the arm oscillating. Nevertheless, the simulation results are useful as a basis point for further tuning of the flight control system on the real drone. They need to be regarded as indicative of the order of magnitude of the needed controller gains, but not as exact values.

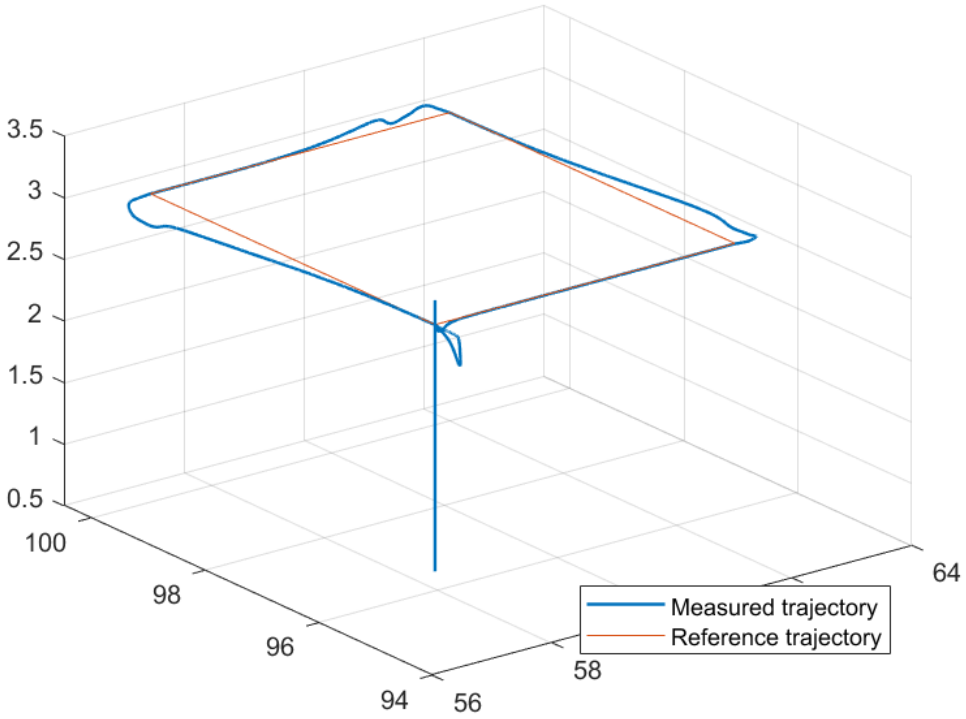


Figure 5.7: Drone trajectory at 1[m/s]

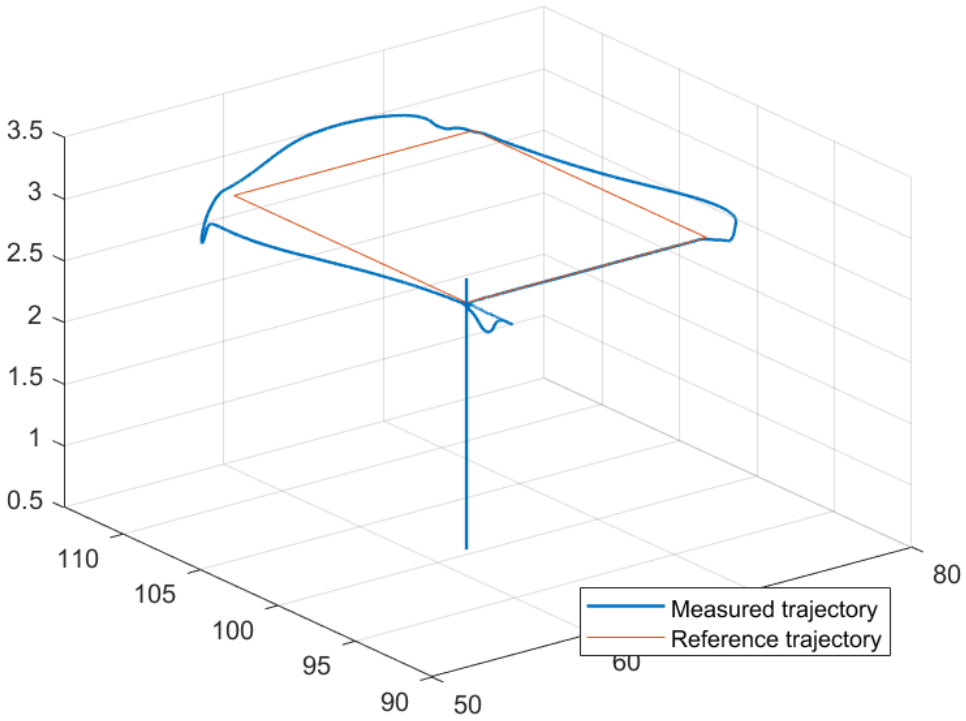


Figure 5.8: Drone trajectory at 3[m/s]

Chapter 6

Conclusion

This master's thesis is the first step in a much broader work, and the end result is still far from reach. However, many milestones have been reached and significant progress has been made in order to achieve autonomous puppetry by drones. First, an expressive robotic arm was made entirely with rapid prototyping and inexpensive, off-the-shelf available components. The 3D printed fingers, made out of semi-flexible material in order to have live junction, were a success in terms of movements and lifelikeness.

Although the arm did not have the robustness and hoped for and could be more dynamic, it validated the potential in term of expressiveness of servo-based robotic arms with 3D printed fingers and served as a prototype on which the drone's specificities where based. Clear improvements paths are available for the future, the main one being to synergize the now existing drone with the arm by combining their batteries and controllers, treating the drone-arm pair not as two different systems but as a whole comprised of two sub-systems.

The custom-made drone will continue to be useful until the end of this project. Indeed, it's high stability even when carrying the arm, it's low noise output and his malleability makes it the perfect tool, even taking into consideration possible changes in the show. Future work will possibly require a change in flight controller in order to be better suited for automation, and the utilisation of a new lithium-ion battery, composed of multiple cells located in the rod between the arm and the drone, in order to reach longer autonomy sufficient for the final show.

Finally, a simulation was made as the first step of the automation process. This simulations requested the computation of many parameters, such as the moments of inertia matrix, with precise CAD models. Once done, this permitted the validation of the low-level controller consisting of six PID controllers layered in two loops, and will serve to validate the high level controller in the future. In the meantime, the general framework for automation, with the usage of a ground station and the Qualisys motion capture cameras was laid in collaboration with Gaëtan Permentiers, a student realising a master's thesis on this project as well.

This work was a long but exciting journey in the world of robotics, automation and quadcopters, with detours in the land of 3D printing and rapid prototyping. Although still in an early development stage, it gives a glimpse of the potential of UAV's in the artistic field, as yet another testimony of the extraordinary versatility of quadcopter drones.

Bibliography

[1] Julia Ghorayeb Zamboni. "Robot Ludens: Inducing the Semblance of Life in Machines". In: (2020).

[2] M. Ailinger & K. Egger & D. Freitag & R. Hofstetter & D. Keidel & P. Lustenberger & F. Martinoni & G. Wiedebach. "Focus Project PuppetCopter technical report". In: *ETH Zurich* (2013).

[3] K. Egger. "Simulation and Animation of a Robotic Puppet". In: *Bachelor thesis ETH Zurich* (2013).

[4] Robert Scott Trowbridge, James Alexander Stark, and Clifford Wong. "Aerial Display System with Marionettes Articulated and Supported by Airborne Devices". U.S. pat. 8876571. Nov. 3, 2014.

[5] *Verity Studio*. URL: https://veritystudios.com/. (accessed: 05.08.2021).

[6] zurichmade. *Drone costume design by Verity Studios*. URL: https://zurichmade.zhdk.ch/refresh-2/exhibition/drone-costume-design-by-verity-studios/. (accessed: 05.08.2021).

[7] Keith Saw. *Verity Studios Aims to Fly Higher With Autonomous Drone Entertainment*. URL: https://www.roboticsbusinessreview.com/unmanned/verity-studios-fly-higher-drone-entertainment/. (accessed: 05.08.2021).

[8] Raffaello D'Andrea Markus Hehn Markus Waibel. "Distributed localization systems and methods and self-localizing apparatus". U.S. pat. 9945929. Apr. 17, 2018.

[9] Federico Augugliaro, Angela P. Schoellig, and Raffaello D'Andrea. "Methods for Designing and Executing an Aerial Dance Choreography". In: *IEEE ROBOTICS AND AUTOMATION MAGAZINE* (2013). DOI: 10.1109/MRA.2013.2275693.

[10] Tsuru Robotics. *Katsuru Beta*. URL: https://tsuru.su/en/project/katsuru-beta/. (accessed: 10.08.2021).

[11] Tsuru robotics. *World's First Graffiti Drone (Part 1)*. URL: https://diydrones.com/profiles/blogs/first-in-the-world-graffiti-drone-part-1. (accessed: 10.08.2021).

[12] Tsuru robotics. *World's First Graffiti Drone (Part 2)*. URL: https://diydrones.com/profiles/blogs/world-s-first-graffiti-drone-part-2. (accessed: 10.08.2021).

[13] Anurag Sai Vempati et al. "PaintCopter: An Autonomous UAV for Spray Painting on Three-Dimensional Surfaces". In: *IEEE ROBOTICS AND AUTOMATION LETTERS* (2018). DOI: 10.1109/LRA.2018.2846278..

[14] Enabling the future. *THE FLEXY HAND AND FLEXY HAND 2*. URL: https://enablingthefuture.org/upper-limb-prosthetics/the-flexy-hand/. (accessed: 40.08.2021).

[15] Gyrobot. *Thingiverse: Flexy-Hand*. URL: https://www.thingiverse.com/thing:242639. (accessed: 14.08.2021).

[16] Polymaker. *PolyFlex TPU90*. URL: https://eu.polymaker.com/product/polyflex-tpu90/. (accessed: 05.09.2021).

[17] ninjatek. *TROUBLESHOOTING GUIDE*. URL: https://ninjatek.com/support/troubleshooting-guide/. (accessed: 05.09.2021).

[18] simplify3D. *Print Quality Troubleshooting Guide*. URL: https://www.simplify3d.com/support/print-quality-troubleshooting/. (accessed: 05.09.2021).
[19] Ninjatek. *Ninjaflex*. URL: https://ninjatek.com/shop/ninjaflex/. (accessed: 01.01.2021).
[20] Tadej Bajd et al. *Robotics*. INTELLIGENT SYSTEMS, CONTROL, AND AUTOMATION: SCIENCE AND ENGINEERING. Springer, 2010. ISBN: 978-90-481-3775-6.
[21] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley sons, inc., 2006.
[22] Last Minute Engineers. *Insight Into ESP32 Sleep Modes Their Power Consumption*. URL: https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/. (accessed: 10.09.2021).
[23] Polymaker. *PolyFlex TPU95*. URL: https://eu.polymaker.com/product/polyflex-tpu95/. (accessed: 28.11.2021).
[24] Nanyaporn Intaratep et al. "Experimental Study of Quadcopter Acoustics and Performance at Static Thrust Conditions". In: *Aeroacoustics Conferences* (2016). DOI: 10.2514/6.2016-2873.
[25] Practical Boat owner. *How to choose the right propeller for your boat*. URL: https://www.pbo.co.uk/gear/choose-right-propeller-boat-62058. (accessed: 02.01.2021).
[26] Richard Madison. *Fan engineering ed. 8*. Buffalo Forge Company, 1983.
[27] Timothy A. Brungart & Stephen T. Olsonb & Brian L. Kline & Zachary W. Yoasd. "The reduction of quadcopter propeller noise". In: *INCE/USA* (2019). DOI: 10.3397/1/376723.
[28] T-motor. *FA18.2×5.9 Push Type Prop*. URL: https://store.tmotor.com/goods.php?id=880. (accessed: 10.10.2021).
[29] DJI. *Flight Controlt*. URL: https://developer.dji.com/mobile-sdk/documentation/introduction/flightController_concepts.html. (accessed: 02.01.2021).
[30] SMACCMPilot. *What is an RC transmitter?* URL: https://smaccmpilot.org/hardware/rc-controller.html. (accessed: 02.01.2021).
[31] Dennis Thevara and Vasanth Ch. "Application of photogrammetry to automated finishing operations". In: *IOP Conference Series: Materials Science and Engineering* 402 (Sept. 2018). DOI: 10.1088/1757-899X/402/1/012025.
[32] MathWorks. *Quadcopter Project*. URL: https://nl.mathworks.com/help/aeroblks/quadcopter-project.html. (accessed: 19.12.2021).
[33] Fabian Riether, Eric Wieser, and Jerome Bouvard. *RollingSpiderEdu*. URL: https://github.com/Parrot-Developers/RollingSpiderEdu. (accessed: 19.12.2021).
[34] Karl Johan Aström and Richard M Murray. *Feedback systems : an introduction for scientists and engineers*. Princeton university press, 2010.
[35] Mark Boover. *Quadcopter Standard configuration*. URL: https://www.quora.com/Can-I-use-a-counterclockwise-motor-clockwise-in-a-quadcopter-Should-I-attach-a-clockwise-or-anti-clockwise-propeller-to-it. (accessed: 02.01.2021).
[36] PX4 user guide. *jMAVSim with SITL*. URL: https://docs.px4.io/master/en/simulation/jmavsim.html. (accessed: 20.12.2021).
[37] Paul E. I. Pounds, Robert E. Mahony, and Peter Corke. "Modelling and control of a quad-rotor robot". In: 2006.
[38] Meyer-Bisch C. "Les chiffres du bruit [Measuring noise]." In: *Med Sci (Paris)* (2015). DOI: 10.1051/medsci/2005215546.

Appendix A

ESP32 code

Function generating the bang-bang velocity profile for the elbow:

```
void bangbang5(int qf, int tf){ //sets motor 5 to position qf in time tf (in
    centiseconds) using bang bang minimum acceleration
  int qi = pos5;
  int mid = tf/2;//middle value of the trajectory
  int tc = mid; //max acceleration time
  float acc = 1000*4*(qf-qi)/(tf*tf);//*1000 to avoid rounding up to 0
  float a = 4*(qf-qi);//debug variables
  float b = (tf*tf);
 for(t=0 ; t<tf; t=t+1){
  if(t>tf){
    t=tf;//make sure we do not go too far with time, needed if time increment >1
  }
  if (t<=tc){
   pos5= qi + acc*t*t/2/1000;
   Serial.println("phase1");
   Serial.println(pos5);
  } else if(t>tc){
   pos5= qf - acc*(t-tf)*(t-tf)/2/1000;
   Serial.println("phase2");
   Serial.println(pos5);
  }
  myservo5.write(pos5);
  delay(10);
 }
}
```

Function generating the trapezoidal velocity profile for the elbow:

```
void move5(int qf, int tf){ //function to slowly move the elbow servo 5 with a
    trapezoidal velocity proffile
  float mid = 0;
  int qi = pos5;
  mid = tf/2;//middle value of the time
  float tc = 0; //max acceleration time
  float ac = 0;
  if (qi<qf){
  ac=0.06; //max acceleration for the elbow servos able to do 60   in 200ms max
  } else if(qi>=qf){
    ac= -0.06; //max acceleration for the elbow servos able to do 60   in 200ms
    max
  }
  tc = tf/2 - sqrt(tf*tf - 4*(qf - qi)/ac)/2;
  Serial.println("tc = ");
   Serial.println(tc);
 for(t=0 ; t<=tf; t=t+1){
  if (t<=tc){
   pos5= qi + ac*t*t/2;
   Serial.println("phase1");
   Serial.println(pos5);
  } else if(t>tc && t<(tf-tc)){

   pos5= qi + ac*tc*(t-tc/2);
   Serial.println("phase2");
    Serial.println(pos5);
  } else if(t>=(tf-tc)){
   pos5= qf - ac*(t-tf)*(t-tf)/2;
   Serial.println("phase3");
    Serial.println(pos5);
  }
  myservo5.write(pos5);
  delay(10);
 }
}
```

Main loop generating the commands for the motors and managig the web server. Showed here with only the march animation for simplicity. Adapted from the "ESP32 Web Server – Arduino IDE" tutorial by The random nerd tutorials[1].

```
void loop(){
  WiFiClient client = server.available();   // Listen for incoming clients

  if (client) {                             // If a new client connects,
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");          // print a message out in the serial
    port
    String currentLine = "";                // make a String to hold incoming
   data from the client
    while (client.connected() && currentTime - previousTime <= timeoutTime) {
   // loop while the client's connected
      currentTime = millis();
      if (client.available()) {             // if there's bytes to read from the
    client,
        char c = client.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline
   character
          // if the current line is blank, you got two newline characters in a
   row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200
    OK)
            // and a content-type so the client knows what's coming, then a
   blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // turns the GPIOs on and off
            if (header.indexOf("GET /27/on") >= 0) {
              Serial.println("march on");
              march = "on";
              for (i=0; i<= 20; i++){
                bangbang5(80, 50);//60
                delay(100);
                bangbang5(170, 50);
                delay(100);
              }

            }
            else if (header.indexOf("GET /27/off") >= 0) {
              Serial.println("march off");
              march = "off";
              bangbang5(90, 100);

            }
```

[1]https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

```
            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to fit your preferences
            client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");
            client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px;");
            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
            client.println(".button2 {background-color: #555555;}</style></head>");

            // Web Page Heading
            client.println("<body><h1>ESP32 Web Server</h1>");

            // Display current state, and ON/OFF buttons for GPIO 27
            client.println("<p>march - State " + march + "</p>");
            // If the output27State is off, it displays the ON button
            if (march=="off") {
              client.println("<p><a href=\"/27/on\"><button class=\"button\">ON</button></a></p>");
            } else {
              client.println("<p><a href=\"/27/off\"><button class=\"button button2\">OFF</button></a></p>");
            }

            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Appendix B

Finger elastics

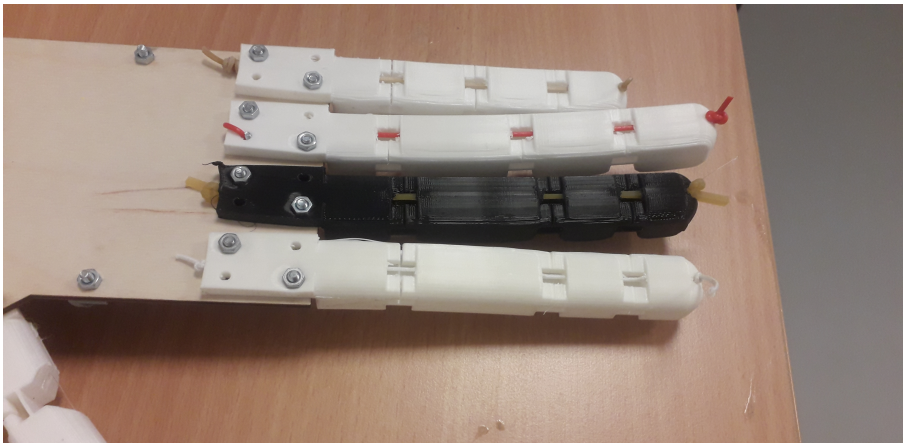


Figure B.1: Test bench for various elastics. From top to bottom: kitchen elastic, unextruded FilaFlex, 95A material with kitchen elastic, small textile elastic.

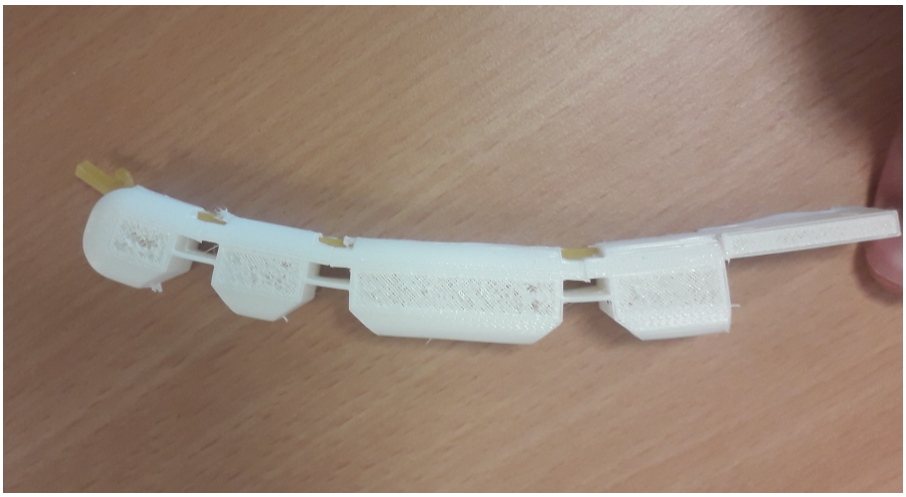


Figure B.2: Result of the finger printed in fully flexible Ninjaflex material [19], with a normal kitchen elastic.



(a) Small textile elastic

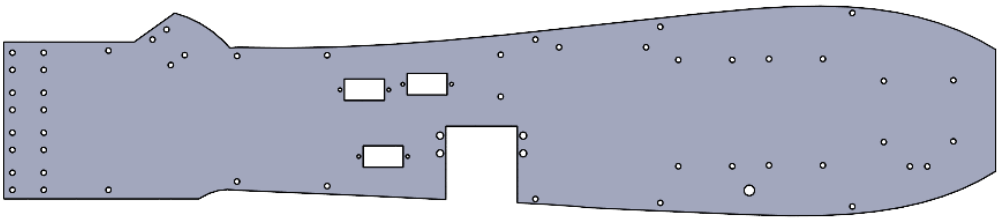


(b) Bigger textile elastic

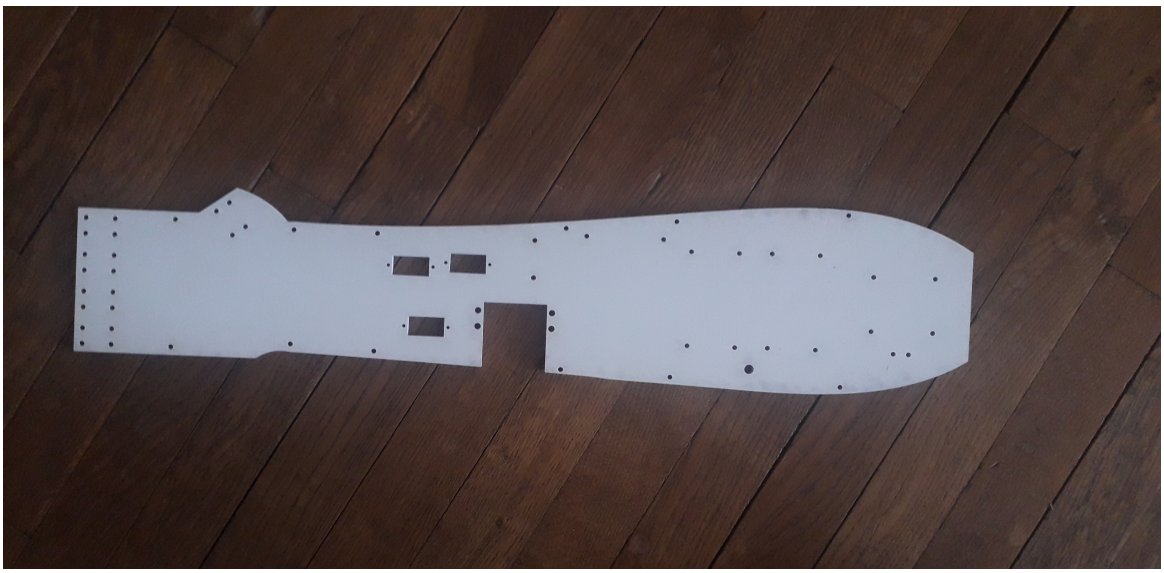
Figure B.3: The two variants of the used textile elastics

Appendix C

3D models



(a) 3D model in Solidworks



(b) Laser cut wood, painted white

Figure C.1: Base frame for the arm.

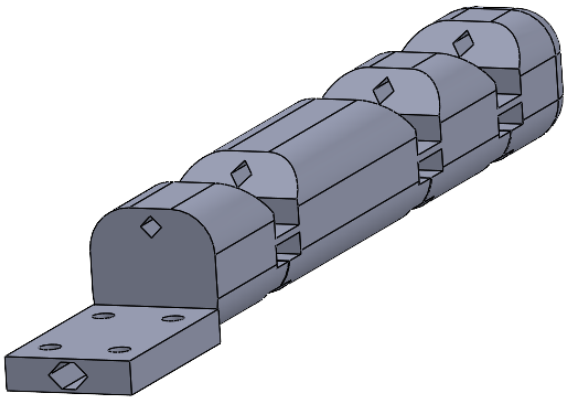


Figure C.2: Final finger model with adapted hole for wider elastics.

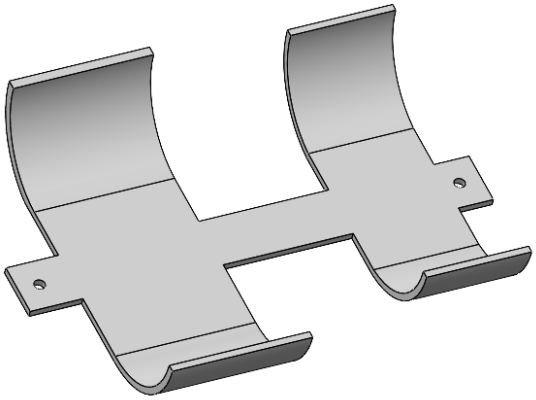


Figure C.3: Protection plate separating the servos and pulleys from the sleeve.

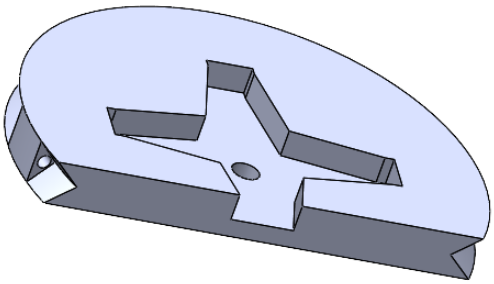
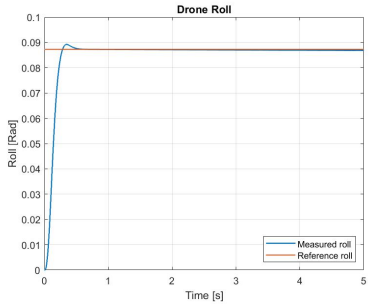


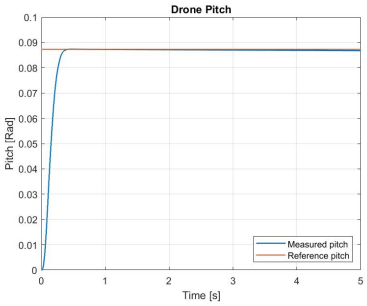
Figure C.4: Pulley design for the HS311 servo.

Appendix D

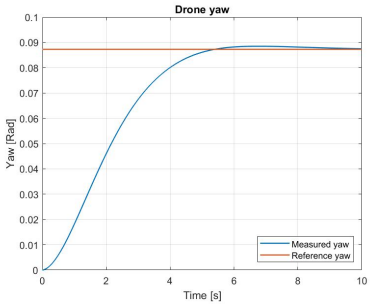
PID Step responses and gains



(a) Roll controller: P=380, I=1, D=40

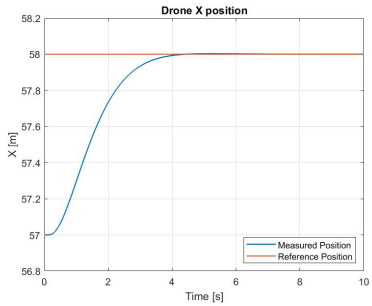


(b) Pitch controller: P=320, I=1, D=40

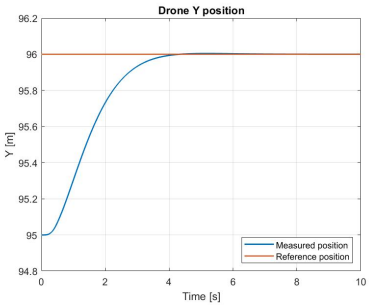


(c) Yaw controller: P=0.2, D=0.4

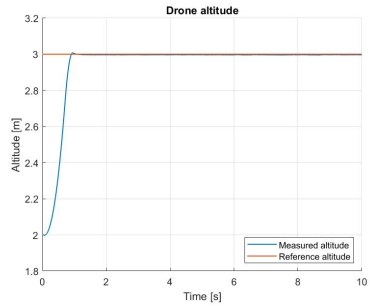
Figure D.1: Step response of the roll, pitch and yaw controllers for a step of $5\,\mathrm{deg}$



(a) X position controller: P=-0.12, D=-0.18



(b) Y position controller: P=0.12, D=0.18



(c) Altitude controller: P=52000, D=6400, gravity feedforward.

Figure D.2: Step response of the X, Y and altitude position controllers for a step of 1[m]

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**