DIEGO FERRAZ NAZARÉ

DEVELOPMENT AND MANUFACTURE OF A DELIVERING UAV

Turin
2021

DIEGO FERRAZ NAZARÉ

DEVELOPMENT AND MANUFACTURE OF A DELIVERING UAV

Thesis presented to the Graduate Program in Mechanical Engineering at the Polytechnic School of the University of São Paulo, Brazil and the Polytechnic University of Turin, Italy to obtain the dual degree in Master of Science.

Concentration Area:

Mechanical Engineering

Supervisors:

Prof. Carlo Novara

Prof. Dr. Roberto Spinola Barbosa

Turin

2021

ACKNOWLEDGMENTS

This thesis is ultimately result of the belief of multiple people I have had the pleasure to make connections with.

I would like to thank Aloisio Neto, Ana Paula Manfrinatti Neuenschwander, André Marra, André Pimenta Mathias, Célia Moreira de Almeida, Eckart Spielkamp, Eduardo Naoki Akiyoshi Ichige, Fábio Alves Tomaz, Fernanda Cirillo, Fernando Paiva, Flávia Cristina Escobar Santana, Gabriela Soares, Gustavo Rosolen, Jéssica Gomes, Kristiane Silveira Fernandes, Lucas de Oliveira Fracarolli, Luiz Fernando Portella Staub, Marcelo Seraphim, Miyuki Ichige, Nanci Saraiva Moreira, Paulo Macedo, Rafael Herrera, Rafael Nass de Andrade, Rubens Mario Marques de Freitas, Thiago Falcone e Tunico Barros that gave me not just the financial support, but the motivation to cross the ocean pursuing a diploma on Europe.

Thanks to Ana Virginia Kesselring and the Virginia Center that not just did help improving my English skills, but built the confidence on which later made me learn also Italian.

Thanks to Beatriz Lucca and Felipe Sereno that provided me friendship, support, but more than that, made sure everything would run smoothly while I was abroad.

Thanks to Elvis and Felipe for providing me so many laughs and good times and for helping me out in times of crisis.

Thanks to professor Dr. Edilson Hiroshi Tamai that besides teaching me the technicalities of Mechanical Engineering, help me turning ideas into reality.

Thanks to Felipe Labate, that put me on contact with so many people that would help me in so many ways. Thank you for the long phone calls, for passing your experience to me and for motivating me when I was about to departure for the exchange program.

Thanks also to my mentor, friend, sponsor and second mother, Solange Cunha, for the numerous calls, meetings, planning and the patience to endure me. This would not be possible without you! It was not a second easy, but this is not longer a dream!

Thanks to all the friends that I had the pleasure of living with during my masters: Mário, Pedro (PG), Arthur, Thiago, Gilberto, Daniel, Fabrizio, Guilherme, Marcelo, Felipe and Tiago, I am glad to call you brothers.

I am thankfully also to have had the opportunity to be a member of the Association des États Généraux des Étudiants de l'Europe (AEGEE), my big family in Europe, where I've learnt so much and had the luck of meeting my partner Dorottya, who I am also thankful for supporting me all the way through this work.

Also, special thanks for my advisors Prof. Carlo Novara and Prof. Dr. Roberto Spinola Barbosa for the numerous reviews and feedback throughout the work.

Last but not least, thank to my family for being my rock and to give me everything any person could ask for. Mom and Dad, you are true heroes!

"Que Deus me guarde, pois eu sei que ele não é neutro
Vigia os rico, mas ama os que vem do gueto"
Racionais MC's

ABSTRACT

This thesis presents the steps for the development of a multicopter of the type quadrotor for delivering usage by means of semi-autonomous techniques. After a brief introduction of the state of the art for the UAV (Unmanned Aerial Vehicles) technology, the equations of motion using Euler's angles approach are presented. In sequence, after a review of the propulsion components available on the market, a method for choice of the best fit of commercial components, and therefore dimensioning, is presented. Following, the control approach is introduced as the MATLAB simulation and results. The last part of the work focuses, on the manufacture of the airframe using additive manufacturing processes and on the final assembly of the UAV. Lastly, suggestions for next works are summed up and presented.

Keywords – UAV, Quadrotor, Control, Propulsion Optmization, Authonomous Robot.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The use of Unnamed Aerial Vehicles (UAV), popularly known as drones, have been increasing over the years. Their application covers a wide spectrum of possibilities and is helpful where its use is an advantage in terms of safety and productivity. The Amazon Prime Air, which aims to deliver products by using drones [1], is one of many examples of its potential.

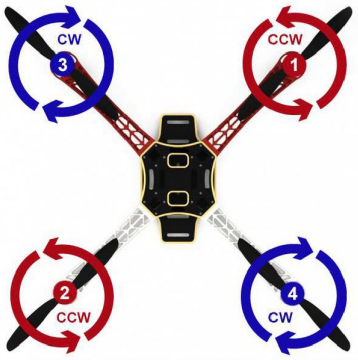
Following this line, the purpose of this thesis is the development and manufacture of an UAV for the transport of goods in low scale. On the next pages, the steps for the construction of such a robot are presented from the introduction of the equations of motion to the manufacture of the airframe, passing through the control and choice of best fit of commercial components.

Before stepping into the content of the work itself, let‘s take a quick look in the general concept of flight of a multicopter. For example, for the quadrotor, a multicopter with 4 rotors, the movement is achieved through the rotation of propellers connected to rotors. Each rotor generates moment and thrust. Considering figure 1, in order to have hover, the moment in relation to the geometric center generated by the thrust on the rotor #1 should cancel the moment generated by the thrust on the rotor #2. Similarly, the moment caused by the thrust on #4 should cancel the moment caused by the thrust on #3. Also, the sum of all reaction torques pointing outwards the paper on the figure 1 should be null.

Any unbalance of thrust or reaction moment causes the quadrotor to move. For example, if an yaw movement is desired, the sum of the reaction torques pointing outwards the paper need to be different than zero, making the quadrotor yaw. If, for example, the quadrotor is desired to move to the right on the figure 1, the thrust on #3 overcomes #4 and similarly, the thrust on #2 overcomes #1 by the same amount. This unbalance, makes the quadrotor to tilt and move to the right. Any other movement, can be similarly achieved by the actuation on the rotors.

As it will be present later, the thrust and the reaction moments change with the

Figure 1: Quadrotor



Source:[2]

angular speed of the rotors.

1.1 Objectives

The main goal of this work is the construction of a functional prototype that simulates the transportation of goods. In order to achieve such goal, the work will be divided into 5 main parts all interconnected. Those parts are:

- **Development of Equations of Motion**: In this step the mathematical model for the UAV will be defined. This first part will be fundamental when implementing the control method.
- **Components Optimization**: In this part several commercial components for the propulsion system will be further analyzed and a method for the definition of the best set of commercial components will be presented. The main dimensions of the UAV are therefore determined in function of the best fit of components chosen.
- **Control and Simulation**: with the mathematical model, the size of the quadcopter and its inertial properties, the control model can be defined and simulations can be performed before the real flight take place. The first tunning of PID paramenters realized on this stage might be modified according to the flight performance obtained during eventual tests.
- **Airframe Manufacture**: With the dimensions of the UAV, the airframe can be designed and simulated through FEM analyses . Then, the manufacture process itself can be started. Considerations about the manufacturing method in order to increase mechanical properties and keep vibrations responses below safe flight conditions will be presented as well.

2 DYNAMIC EQUATIONS OF MOTION

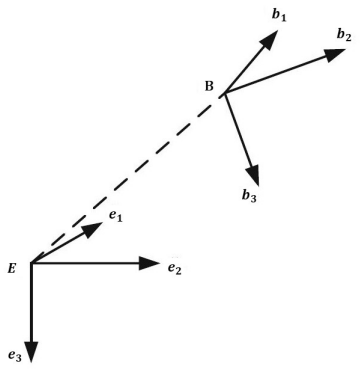
2.1 Attitude Representation

2.1.1 Coordinate frames

Firstly, the two coordinate frames to be used along the development of the dynamic equations of motion are defined. As show on figure 2, the coordinate frame $E\, e_1 e_2 e_3$ is called Earth-fixed Coordinate frame (Earth) and is the inertial frame of reference. The frame described by $B\, b_1 b_2 b_3$ is called the Aircraft Body Coordinate frame (Body) and is fixed to the multicopter [3]. The following versors are used for those frames:

- **Earth**: { $\mathbf{e_1}$ $\mathbf{e_2}$ $\mathbf{e_3}$ }
- **Body**: { $\mathbf{b_1}$ $\mathbf{b_2}$ $\mathbf{b_3}$ }

Figure 2: Coordinate frames



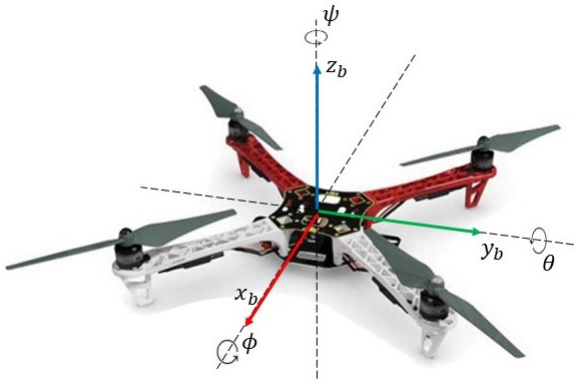
Adapted from:[3]

As one can easily notice from figure 2, both coordinate frames are hand-side oriented.

2.2 Euler's Angles

In order to obtain the equations of motion, the multicopter will be considered as a quadrotor (4 rotors) moving in X-configuration as in figure 3. The origin of the right-hand oriented Body is coincident with the the center of mass of the quadrotor (figure 3).

Figure 3: Aircraft Body Coordinate frame (Body)



Source [2]

The angles θ, ϕ and ψ are called **pitch** angle, **roll** angle and **yaw** angle, respectively. The positive direction of such angles are shown on the figure 3 and coincide with the positive direction of the Body axis. The Euler's Angles representation used is the ZXY, obtained by a sequence of rotations on the following order: rotation ψ around z, rotation ϕ around the new x axis after the previous rotation; and rotation θ around the new y axis after the previous rotation. The rotation matrices are described as follow:

$$R(y,\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$R(x,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi & cos\phi \end{bmatrix}$$

$$R(z,\psi) = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\phi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The resultant matrix R is defined as

$$R = R(z, \psi)R(x, \phi)R(y, \theta) \tag{2.1}$$

In order to simplify the notation, the following convention will be adopted: *sinx* := *sx* and *cosx* := *cx*. Therefore, the matrix R results:

$$R = \begin{bmatrix} c\psi c\theta - s\psi s\phi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ s\psi c\theta + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\theta s\phi c\psi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{2.2}$$

By pre-multiplying a vector described in the Body for the rotation matrix R (2.2), the vector described in the fixed frame (Earth) is obtained.

Such results show that from the Euler Angles θ, ϕ and ψ the rotation matrix can easily be obtained. Now, the inverse problem would be further studied: how to determine the Euler's Angles from the rotation matrix R. In order to solve this, let's assume that the rotation matrix R is known and is represent by the following notation:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.3}$$

As mentioned before, the positive direction of the Euler's angles are shown in figure 3. :

1. **Pitch angle:** the value range for θ is [-π,π];
2. **Roll angle:** the value range for ϕ is [-$\pi/2$,$\pi/2$];
3. **Yaw angle:** the value range for ψ is [-π,π].

from (2.2) and (2.3) the following system of equations can be easily obtained:

$$\begin{aligned} \phi &= arcsin(r_{32}) \\ \theta &= atg2(\tfrac{-r_{31}}{c\phi}, \tfrac{r_{32}}{c\phi}) \\ \psi &= atg2(\tfrac{-r_{12}}{c\phi}, \tfrac{r_{22}}{c\phi}) \end{aligned} \tag{2.4}$$

Here, the *atan*2 function is used over the *atan* since the sign of the angle matters.

from the system of equations (2.4) ϕ is well defined on the interval [-π/2,π/2]. However, when ϕ is equal to -π/2 or π/2 the cosine is null and both θ and ψ are undefined. This is know as the **singularity problem** and is the main drawback of the Euler representation.

2.3 Newton's Second Law

The Newton's Second Law can be written as

$$\mathbf{f} = \sum_{i=1}^{N} \mathbf{f_i} = m\frac{d^E \mathbf{v^c}}{dt} \tag{2.5}$$

where $\mathbf{v^c}$ denotes the linear velocity of the referential of the Body with respect to the Earth expressed in the Earth referential, $\mathbf{f_i}$ is the force applied on the $i-th$ rotor, and m is the total mass of the quadrotor. Furthermore, the overscript E indicates that the derivative is taken in relation to the Earth. The N on the sum symbol indicates the number of rotors and, as discussed before, in the quadrotor case is equal to 4.

Alternatively, the Second Law can be expressed as

$$\mathbf{f} = \frac{d^E \mathbf{L}}{dt} \tag{2.6}$$

where L is the Linear Momentum. The expression (2.6) states that the sum of the forces acting on the body equals the rate of change of the linear momentum.

The figure 4 shows the free body diagram of the quadrotor. The gravity field (not shown on the figure) points on the negative direction of $\mathbf{e_3}$. The position vector of the Body in relation to the Earth described in the Earth is the following

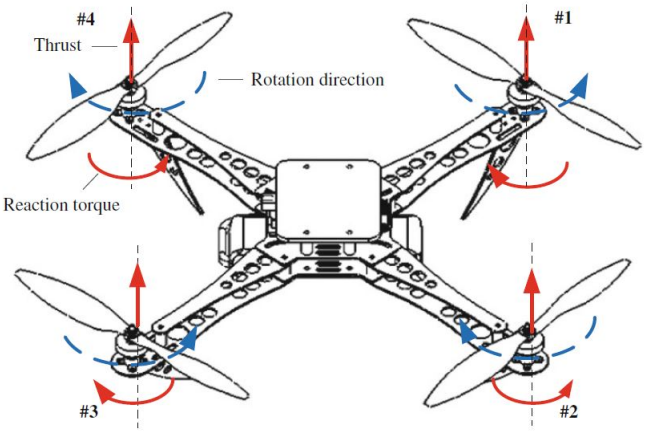
$$\mathbf{r(t)} = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}^T \tag{2.7}$$

Then, by considering the equation (2.5) and the free body diagram on figure 4, one have

$$m\ddot{\mathbf{r}}(t) = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ -(f_1 + f_2 + f_3 + f_4) \end{bmatrix} + \mathbf{f}_{aero} \tag{2.8}$$

where $\mathbf{f}_{aero}$ represents further aerodynamic forces that will be introduce on section 2.5.

Figure 4: free body diagram



Source:[3]

2.4 Euler's Equations

The Euler's Equations describe the rotational dynamics and can be written as

$$\tau_{\mathbf{C}} = \frac{d^E \mathbf{H_C}}{dt} \tag{2.9}$$

where $\tau_{\mathbf{C}}$ is the vector of moments about C, $\mathbf{H_C}$ is the Angular momentum of the center of mass. The expression of the angular momentum is

$$\mathbf{H_C} = J_C \cdot \omega^B \tag{2.10}$$

where ω^B is the angular rotation of the Body with respect to the Earth expressed in the Body and J_C is the inertia matrix of the quadrotor expressed on the Body and described as

$$J_C = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \tag{2.11}$$

At this point, the following assumptions are adopted: (1) e_1, e_2 and e_3 are axis of symmetry of the quadrotor (figure 5); and (2) the origin of the Body coincides with the Center of Gravity (CoG) of the quadrotor. from those assumptions equations (2.9) and (2.11) can be further simplified into:

$$\tau_{\mathbf{C}} = \frac{d^B \mathbf{H_C}}{dt} + \omega^B \times \mathbf{H_C} \tag{2.12}$$

and

$$J_C = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{2.13}$$

The angular velocity can be written in a more convenient notation as

$$\omega^B = \begin{bmatrix} p & q & r \end{bmatrix}^T \tag{2.14}$$

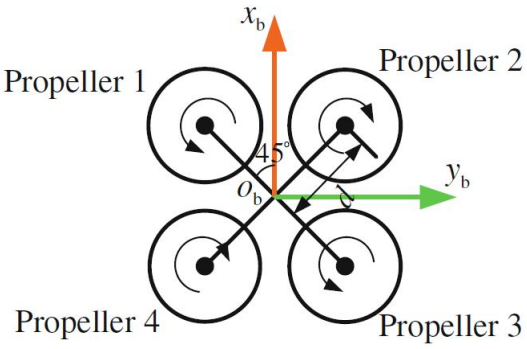
from figure 4, one can easily obtain

$$\tau_C = \begin{bmatrix} \frac{d\sqrt{2}}{2}(f_1 - f_2 - f_3 + f_4) \\ \frac{d\sqrt{2}}{2}(f_1 + f_2 - f_3 - f_4) \\ -\tau_1 + \tau_2 - \tau_3 + \tau_4 \end{bmatrix} \tag{2.15}$$

Then the equation (2.12) can be rewrite as

$$J_C \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{d\sqrt{2}}{2}(f_1 - f_2 - f_3 + f_4) \\ \frac{d\sqrt{2}}{2}(f_1 + f_2 - f_3 - f_4) \\ -\tau_1 + \tau_2 - \tau_3 + \tau_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times J_C \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.16}$$

Figure 5: Quadrotor X-configuration



Source:[3]

2.5 Quadrotor Aerodynamics: Thrust and Moments Model

The rigid body equations (2.8) and (4.31) of the quadrotor are here summarized:

$$\begin{aligned} m\ddot{\mathbf{r}} &= mg\mathbf{e_3} + R\mathbf{f} + \mathbf{f}_{aero} \\ J \cdot \dot{\omega} &= \tau - \omega \times J\omega \end{aligned} \tag{2.17}$$

where $\mathbf{f}$ and τ combine the main nonconservative forces and moments applied to the quadrotor due to the aerodynamics of the rotors [4], while $\mathbf{f}_{aero}$ is the contribution due to the blade flapping phenomenon, that can be neglected if comparing with the other forces.

In the hover case, the steady-state thrust generated by the rotor [3] can be modeled as

$$f_i = C_T \rho A_{r_i} r_i^2 \omega_i^2 \tag{2.18}$$

where C_T is the thrust coefficient dependent on the geometry of the propeller, ρ is the air density, r_i is the radius of the $i-th$ propeller, A_{r_i} is the circumferential area defined by the radius r_i, and ω_i is the angular speed of the $i-th$ propeller (rotor) measured in rad/s. Similarly, the reaction torque due to the rotor drag can be modeled as

$$\tau_i = C_\tau \rho A_{r_i} r_i^3 \omega_i^2 \tag{2.19}$$

where C_τ is the thrust coefficient dependent on the geometry of the propeller. Considering the variation of the air density on the altitude range of the quadrotor negligible for the evaluation of the forces and moments, the following constants can be defined

$$\begin{aligned} c_T &= C_T \rho A_{r_i} r_i^2 \\ c_\tau &= C_\tau \rho A_{r_i} r_i^3 \end{aligned} \tag{2.20}$$

Combining (2.18), (2.19) and (2.20), and from figure 4 and 5 the following matrix system is obtained

$$\begin{bmatrix} \mathbf{f} \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ dc_T\frac{\sqrt{2}}{2} & dc_T\frac{\sqrt{2}}{2} & -dc_T\frac{\sqrt{2}}{2} & -dc_t\frac{\sqrt{2}}{2} \\ -dc_T\frac{\sqrt{2}}{2} & dc_T\frac{\sqrt{2}}{2} & dc_T\frac{\sqrt{2}}{2} & -dc_T\frac{\sqrt{2}}{2} \\ c_M & -c_M & c_M & -c_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2.21}$$

The matrix system (2.21) obtained by a quadrotor can be easily extended for a general case of more rotors. for further details consult [3].

Given the desired thrust and moments, the matrix form allows to solve for the required rotor speeds.

2.6 Summary

This chapter aimed to present the dynamic equations of motion for a quadrotor. By introducing the Euler's angles representation, the rotation matrix to transform the vector on the Body space into the Earth (inertial) one is obtained. Moreover, the aerodynamic characteristics of the non-conservative forces and moments are presented and the concept of blade-flapping is briefly introduced.

The Dynamic Equations of Motion obtained are here summarized

$$\begin{aligned} m\ddot{\mathbf{r}} &= mg\mathbf{e_3} + R\mathbf{f} \\ J \cdot \dot{\omega} &= \tau - \omega \times J\omega \end{aligned}$$

Such equations are a necessary step toward the control of the UAV and will be further discussed throughout the project.

3 OPTIMIZATION OF PARAMETERS FOR COMPONENTS CHOICE

3.1 Propulsion System Modeling

The modelling of the propulsion system is an important initial step towards the design of the UAV and will follow the modelling methodology presented in [3].

The propulsion system is compounded by a number of propulsors, motors and electronic speed controllers that are equal in number, connected to a centralized source of energy. Each of those components will be modelled in order to evaluate the performance of the propulsion system in some particulars flight conditions.

Additionally, the airframe model will be discussed as well, and first expressions for its mass and its surface area will be derived.

3.1.1 Environmental Parameters

Before proceeding to the model of the propulsion system itself, the environmental parameters are first analyzed. The main atmospheric parameter to be considered is the density of the air ρ [kg/m^3], which is function of the local altitude h [m] and the temperature T_t [°C] according to the following expression [5]:

$$\rho = \frac{273 P_a}{101325(273 + T_t)} \rho_0 \tag{3.1}$$

where ρ_0 is the standard air density (0 °C) and P_a is the atmospheric pressure given by

$$P_a = 101325 \left(1 - 0.0065 \frac{h}{273 + T_t}\right)^{5.2561} \tag{3.2}$$

On the height range where the multicopter usually operates, the temperature can be considered constant, and so does the atmospheric pressure P_a and the air density ρ.

Another important parameter is the gravitational acceleration g, also considered constant on the height range operation of the drone.

The environmental parameters are summed up on the table 1.

Table 1: Environmental Conditions

Environmental Parameter	**Value**
Altitude h	200 m
Temperature T_t	40 °C
Air Density ρ	1.103 kg/m^3
Atmospheric Pressure P_a	99132.49 Pa
Gravity Acceleration g	9.81 m/s^2

Source: Author

3.1.2 Propeller Modeling

When searching for a propeller, 4 main parameters may be used to evaluate which might better match a determined application. Those are: the Propeller Diameter D_p, the Propeller Pitch H_p, the Number of Blades B_p and the Propeller Mass m_p.

Usually propellers are described by a four number digit, as in the figure 6, where the 2 first numbers represents the diameter of the propeller, and the last two represent the pitch of the propeller both in inches. The pitch is defined as the distance a propeller would move into a soft solid after one revolution. The figure 6 shows a 9545 propeller with 2 blades.

Figure 6: T-Motor 9545 Propeller

Source: [6]

The performance of the propeller is based on its thrust T and torque M given by the following expressions

$$\begin{aligned} T &= C_T \rho (\tfrac{N}{60})^2 D_p^4 \\ M &= C_M \rho (\tfrac{N}{60})^2 D_p^5 \end{aligned} \tag{3.3}$$

The dimensionless parameters C_T and C_M are the thrust coefficient and the torque coefficient respectively. The can be expressed as [3]

$$\begin{aligned} C_T &= 0.25\pi^3 \lambda \xi^2 B_p K_0 \frac{\epsilon arctan \frac{H_p}{\pi D_p} - \alpha_0}{\pi A + K_0} \\ C_M &= \tfrac{1}{8A} \pi^2 C_d \xi^2 \lambda B_p^2 \end{aligned} \tag{3.4}$$

where

$$C_d = C_{fd} + \frac{\pi A K_0^2}{e} \frac{(\epsilon arctan \frac{H_p}{\pi D_p} - \alpha_0)^2}{(\pi A + K_0)^2} \tag{3.5}$$

The values for A, ϵ, λ, ξ, e, C_{fd}, α_0 and K_0 are dependent of the type, material and technology of the propeller. However, [3] indicates that a set of a mean of those parameters match with experimental data. The values of those parameters are summed on the table 25. All the propellers discussed on this text will be considered as with the same values present in table 25.

Table 2: Propeller Parameters

A	5	λ	0.75	e	0.83	α_0	0
ϵ	0.85	ξ	0.55	C_{fd}	0.015	K_0	6.11

Source: [3]

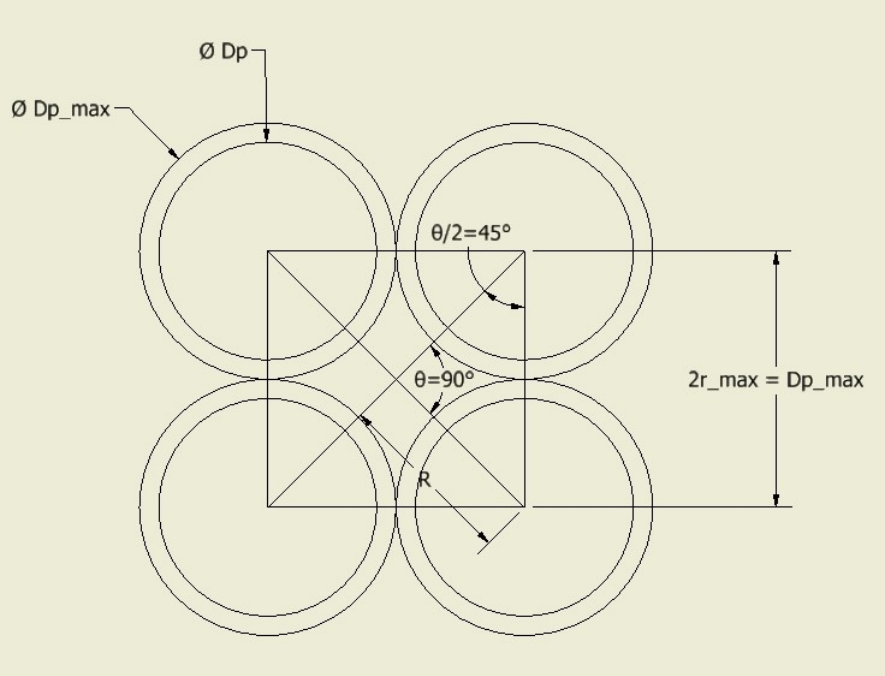
3.1.3 Airframe Modeling

For this initial analysis, the 2 main parameters to be taken into consideration for the modeling of the airframe will be the *diameter D_p of the propeller* and the *material which the airframe would be manufactured*.

The size of the multicopter must match the size of the propellers in order to have a compact configuration without the propellers interfering with each other. In the figure 7 a model of a quadrotor is shown.

In the figure 7, R denotes the measure of the arm of the quadrotor. The size of the arm R is geometrical limited by the size of the propeller. In order to avoid aerodynamic interference caused by the rotation of the propeller, a concentric circumference of diameter $D_{pmax} = 1.2D_p$ is used to set the size of the airframe. For the quadrotor case, the expression for the size of the airframe arm R as function of the propeller dimension is given by:

Figure 7: Quadrotor Configuration



Source: Author

$$R = \frac{r_{max}}{sin(45°)} = \frac{0.6D_p}{sin(45°)} \tag{3.6}$$

Similarly, for a general multicopter with n rotors, one have the following expression

$$R = \frac{r_{max}}{sin(180°/n)} = \frac{0.6D_p}{sin(180°/n)} \tag{3.7}$$

3.1.3.1 Mass of Airframe

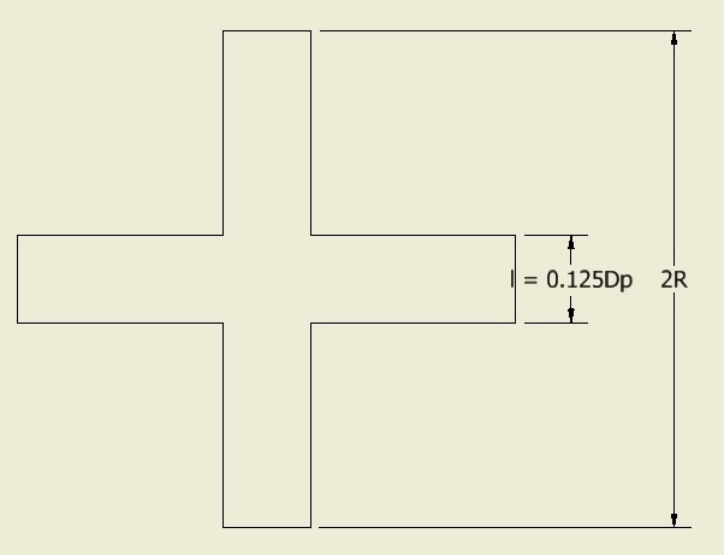
In order to have a first estimation of the mass of the airframe for a quadrotor, a structure such the one of the figure 8 will be used.

All the dimensions of the structure are function of the propeller diameter. Additionally, the thickness d of the structure will be set equal to 10 mm. For the quadrotor case, the expression for the volume of the structure becomes then

$$\begin{aligned} V &= [2Rl + (2R - l)l]d = (4Rl - l^2)d \\ R &= \frac{0.6D_p}{sin(45°)} \end{aligned} \tag{3.8}$$

For a n-rotor, with $n < 2R/l$, a similar structure can be used and one can easily obtained the following expression for the volume

Figure 8: Airframe



Source: Author

$$\begin{aligned} V &= (nRl - l^2)d \\ R &= \frac{0.6D_p}{sin(180°/n)} \end{aligned} \tag{3.9}$$

The mass is then given by

$$m_{af} = 1.25\rho_{af}V \tag{3.10}$$

where a 0.25 factor is added to take into consideration additional weight, as the protection case for the electronic components, bolts, etc. The ρ_{af} is the density of the material to be used. In a vibration point of view, the material has to be rigid enough to prevent the bending and torsion of the arms, being at the same time light in order to improve the endurance time.

3.1.3.2 Surface Area of the Airframe

The surface area of the airframe is an important parameter for the aerodynamic of the flight. As bigger the surface of the drone, as higher drag forces will be present which may lead to poor flight performance.

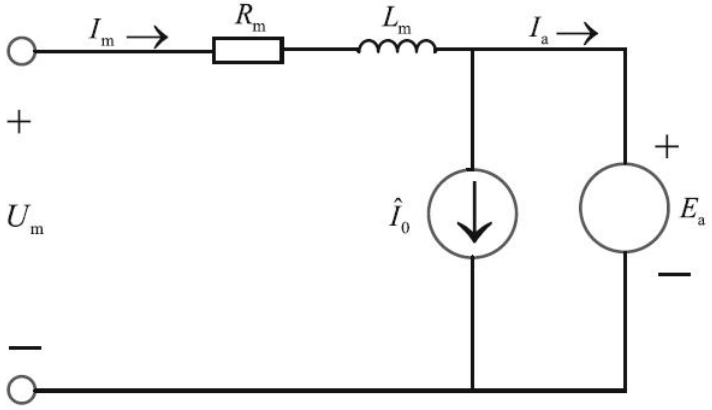
From the figure 7, the maximum surface area of the drone can be considered as, in a general mode, as the total area of the discs delimited by the propellers, plus the area of the internal polygon, minus the area of the intersection between the discs and the internal polygon. For a n-rotor, the following expression can be obtained for the surface area S

$$S = \frac{nD_{pmax}^2}{4}\left[\pi + \frac{1}{tg\frac{\pi}{n}} - \frac{\pi}{2}\left(1 - \frac{2}{n}\right)\right] \tag{3.11}$$

3.1.4 Motor Modeling

The equivalent motor model is shown in figure 9, where U_m is the equivalent motor voltage [V], I_m is the equivalent motor current [A], R_m is the armature resistance [Ω], L_m is the armature inductance [H], $\hat{I}_0$ is the no-load current required to overcome losses due to friction and electrical effects [A], and E_a is the back-electromotive force [V].

Figure 9: Motor Model



Source: [3]

Commercial motors can be evaluate by the following 6 parameters: the KV Value K_{V0}; the Maximum Continuous Current I_{mMax}; the Nominal No-Load Current I_{m0}; the Nominal No-Load Voltage U_{m0}; the Motor Resistance R_m; and the Mass of the Motor m_m. Those parameters are defined as:

- **the KV Value** K_{V0} measures the rotational speed of the no-load motor in RPM's by the voltage applied. A KV920 motor, for example, will have a angular velocity of 920 RPM when 1 V is applied on it.
- **the Maximum Continuous Current** I_{mMax} is a threshold of continuous current which the motor can operate safely with. A maximum continuous current of 15 A, indicates that the motor can safely operates with a continuous current up to 15 A.
- **the Nominal No-Load Current** I_{m0} **and the Nominal No-Load Voltage** U_{m0}: The nominal no-load current, is the current passing through the stator in the idle operation with nominal voltage applied. This voltage is the Nominal No-load Voltage.

- **the Motor Resistance** R_m is the internal resistance of the motor, which can overheat it and reduce its efficiency.

Those parameters are found in a typical test report as in figure 10. The motor of the report is the 2212 KV920 of the T-Motor (figure 11). The 2212 indicates that the motor has a stator diameter of 22 mm (two first digits) and a stator height of 12 mm (two last digits).

Figure 10: Motor Test

Test Report									
Test Item	MN2212 V2.0 KV920			Report NO.			MN.00007		
Specifications									
Internal Resistance	142mΩ			Configuration			9N12P		
Shaft Diameter	4mm			Motor Dimensions			Φ27.5×26.5mm		
Stator Diameter	22mm			Stator Height			13mm		
AWG	20#			Cable Length			400mm		
Weight Including Cables	65g			Weight Excluding Cables			54g		
No.of Cells(Lipo)	2-4S			Idle Current@10v			0.5A		
Max Continuous Power 180S	220W			Max Continuous Current 180S			15A		
Load Testing Data									
Ambient Temperature	/			Voltage			DC Power Supplier		
Item No.	Voltage (V)	Prop	Throttle	Current (A)	Power (W)	Thrust (G)	RPM	Efficiency (G/W)	Operating Temperature (℃)
MN2212 KV920 V2.0	14.8	T-MOTOR 9545	50%	2.1	31.08	293	4260	9.43	/
			65%	4	59.2	476	5300	8.04	
			75%	5.6	82.88	605	5960	7.30	
			85%	7.4	109.52	742	6000	6.78	
			100%	10.3	152.44	918	7350	6.02	
Notes:The test condition of temperature is motor surface temperature in 100% throttle while the motor run 10min.									

Source: [7]

Here, some reasonable simplifications are adopted:

- The armature inductance L_m is neglected;
- The transient process caused by the switching elements are ignored;
- The no-load current $\hat{I}_0$ is considered constant for a given angular speed;

Figure 11: T-Motor 2212 KV920

Source: [7]

- The Nominal No-Load Current I_{m0} can be used as an estimation for the no-load current $\hat{I}_0$.

The aim of the motor modeling is obtain the equivalent current I_m and the equivalent voltage of the motor U_m using the angular speed N and the torque M obtained from the propeller model. As shown in [3] I_m and U_m can be obtained from the following expressions:

$$\begin{aligned} U_m &= \left(\frac{MK_{V0}U_{m0}}{9.55(U_{m0}-I_{m0}R_m)} + I_{m0}\right)R_m + \frac{U_{m0}-I_{m0}R_m}{K_{V0}U_{m0}}N \\ I_m &= \frac{MK_{V0}U_{m0}}{9.55(U_{m0}-I_{m0}R_m)} + I_{m0} \end{aligned} \tag{3.12}$$

3.2 Electronic Speed Controller Modeling

The Electronic Speed Controller, figure 12, is the device in between the battery and the motor. The ESC converts the DC voltage of the battery into the three-phase alternating signal which is synchronized with the rotation of the motor (figure 13).

A model of the ESC is shown in figure 14, where U_e is the ESC input voltage [V], I_e is the ESC input current [A], U_{e0} is the equivalent DC voltage, and R_e is the ESC internal resistance[Ω]. Other 2 important parameters are the Maximum ESC Current I_{eMAX} [A], and the ESC mass m_e [kg], where the first is the parameter that usually names the ESC. As in figure 12, an ESC 15 has a maximum current of 15 A.

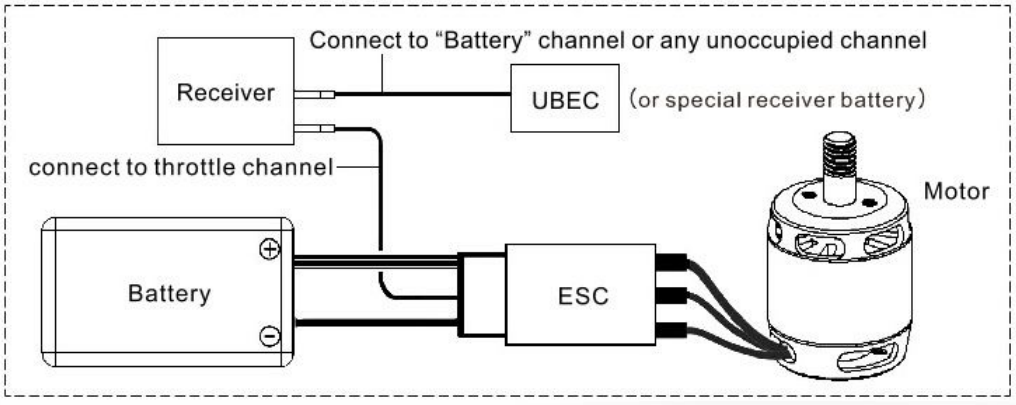
The aim of the ESC model is obtain the ESC input voltage U_e, the ESC input voltage I_e and the throttle command σ given the parameters resulting from the motor model.

From figure 14

Figure 12: Electronic Speed Controller

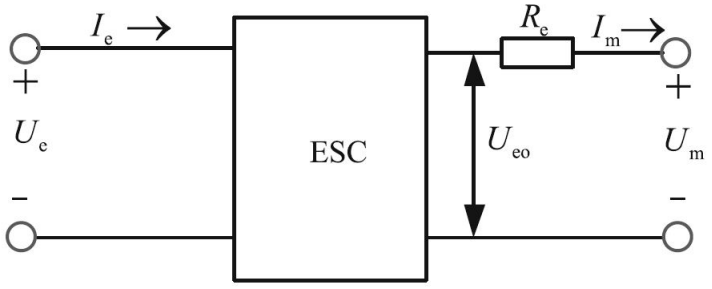
Source: [8]

Figure 13: ESC Wiring



Source: [8]

Figure 14: ESC Model



Source: [3]

$$U_{e0} = U_m + I_m R_m \tag{3.13}$$

The throttle command is then obtained by

$$\sigma = \frac{U_{e0}}{Ue} \approx \frac{U_{e0}}{Ub} \tag{3.14}$$

where U_b is the battery voltage.

The ESC input current I_e can be expressed by the following relation:

$$I_e = \sigma I_m \tag{3.15}$$

The equation 3.15 is based on the fact that the input power of the ESC is equal to its output power.

The ESC voltage U_e is supplied by the battery and can be expressed by

$$U_e = U_b - I_b R_b \tag{3.16}$$

where R_b is the battery internal resistance [Ω] and I_b is the battery current [A]. For a multicopter with n propulsors, the battery current I_b can be expressed as

$$I_b = nI_e + I_{other} \tag{3.17}$$

where I_{other} includes the losses and current from other devices, usually accounts around 1 A.

3.2.1 Battery Modeling

The parameters taken into consideration for the battery modeling are the capacity C_b, the resistance R_b, the total voltage U_b, the maximum discharge rate K_b and the battery weight m_b. The capacity C_b indicates how much electrical charge the battery has and is measured in mAh. The discharge rate is the reason between the current of discharge and the capacity. The maximum discharge rate K_b establishes a threshold for the current of the battery as it will be seen on the section 3.4.3.

As it can be seen on figure 15 the capacity C_b and the maximum discharge rate K_b are the parameters that usually name the battery.

Finally, the battery model mainly aims to obtain the time of endurance t_b from the previous models.

Figure 15: Tattu 3700 Ah Battery

Source: [9]

It is well known that the discharge of the battery is non-linear with its voltage. Here, a simplified method is used where the voltage is considered to remain constant with the discharge process and the capacity of the battery decreases linearly with time [3]. Then, the following expression can be used to express the time of endurance t_b:

$$t_b = 0.06\frac{C_b - C_{min}}{I_b} \tag{3.18}$$

where C_b is expressed in mAh, C_{min} [mAh] is in the range $0.15C_b : 0.20C_b$, and t_b is given in minutes.

3.3 Evaluation of Optimal Parameters

In order to evaluate the set of components which best match the multicopter application, a MATLAB simulation was run by combining dozens of propellers, motors, ESC's and batteries, and three different configurations of multicopter generating more than 10,000,000 combinations. By a sequence of compatibility analyses, incompatible components are then taken out of the analyses. The sets were evaluated by its performance on the following parameters:

- **Maximum time of endurance in hover** t_{eMAX}
- **Maximum system efficiency** η_{MAX}
- **Maximum produced propeller thrust** T_{MAX}
- **Maximum payload mass** $m_{MAXLOAD}$

- **Maximum pitch angle** θ_{MAX}
- **Maximum forward speed** V_{MAX}
- **Maximum flight distance** Z_{MAX}
- **Minimum mass of propulsors** m_{MIN}
- **Parameters of minimum cost**

For each of these items the set which maximize the particular parameter may be determined, and then the set is evaluated on all the parameters.

A second approach, that is the final goal of this report, involves an optimization process which determines a single set that maximizes a series of key parameters with the maximums values obtained on the first approach.

On the table 3 the parameters of the components were summed up. For reference, the

Table 3: Parameters

Component	**Parameter**
Propeller r	$\Theta_{p_r} = (D_{p_r}, H_{p_r}, B_{p_r}, m_{p_r})$
Motor k	$\Theta_{m_k} = (K_{V0_k}, I_{mMAX_k}, I_{m0_k}, U_{m0_k}, R_{m_k}, m_{m_k})$
ESC j	$\Theta_{e_j} = (I_{eMAX_j}, R_{e_j}, m_{e_j})$
Battery i	$\Theta_{b_i} = (C_{b_i}, R_{b_i}, U_{b_i}, K_{b_i}, m_{b_i})$

Source: Author

important expressions for the modelling of the propulsion system were summed up on the table 4. The complete list of components and its parameters are present on appendixes A to D and the MATLAB codes on appendixes E and F.

3.3.1 Maximum Endurance in Hover

The particular condition of flight when the thrust generated by the propulsor group equals the weight $G = gm$ of the multicopter is called *hover*. In this condition the resultant force on the gravitational field direction is null and the multicopter stays fixed in the air at a height h.

In this section, the expression of the maximum endurance in hover will be determined. Given an $ijkr$ set of components, with $0 < i \leq n_b$, $0 < j \leq n_e$, $0 < k \leq n_m$, and $0 < r \leq n_p$, being n_b, n_e, n_m and n_p the number of batteries, ESC's, motors, and propellers being analyzed; the Endurance in Hover t_e is evaluated through the following system of equations

Table 4: Propulsion System Modelling

Propeller model	Thrust Coefficient	$C_T = 0.25\pi^3\lambda\xi^2 B_p K_0 \frac{\epsilon arctan\frac{H_p}{\pi D_p} - \alpha_0}{\pi A + K_0}$	eq 3.4
	Torque Coefficient	$C_M = \frac{1}{8A}\pi^2 C_d \xi^2 \lambda B_p^2$	eq 3.4
	C_d Coefficient	$C_d = C_{fd} + \frac{\pi A K_0^2}{e} \frac{(\epsilon arctan\frac{H_p}{\pi D_p} - \alpha_0)^2}{(\pi A + K_0)^2}$	eq 3.5
	Thrust [N]	$T = C_T \rho (\frac{N}{60})^2 D_p^4$	eq 3.3
	Torque [N/m]	$M = C_M \rho (\frac{N}{60})^2 D_p^5$	eq 3.3
Motor model	Back-electromotive Force Constant	$K_E = \frac{U_{m0} - I_{m0} R_m}{K_{V0} U_{m0}}$	
	Torque Constant	$K_T = 9.55 K_E$	
	Equivalent Current [A]	$I_m = \frac{M}{K_T} + I_{m0}$	eq 3.12
	Equivalent Voltage [V]	$U_m = I_m R_m + K_E N$	eq 3.12
ESC model	Circuit Principle	$U_{e0} = U_m + I_m R_e$	eq 3.13
	Input Throttle	$\sigma = \frac{U_{e0}}{Ue} \approx \frac{U_{e0}}{Ub}$	eq 3.14
	Input Current [A]	$I_e = \sigma I_m$	eq 3.15
	Input Voltage [V]	$U_e = U_b - (nI_e + I_{other})R_b$	eq 3.16
Battery model	Battery Current [A]	$I_b = nI_e + I_{other}$	eq 3.17
	Endurance [min]	$t_b = 0.06(C_b - C_{min})/I_b$	eq 3.18

Source: Adapted from [3]

$$
\begin{aligned}
m_{ijkr} &= m_{af_r} + m_{b_i} + 1.25n(m_{p_r} + m_{m_k} + m_{e_j}) \\
T_{ijkr} &= \frac{gm_{ijkr}}{n} \\
N_{ijkr} &= 60\sqrt{\frac{T_{ijkr}}{\rho D_{p_r}^4 C_{T_r}}} \\
M_{ijkr} &= C_{M_r}\rho(\frac{N_{ijkr}}{60})^2 D_{p_r}^5 \\
U_{m_{ijkr}} &= U_m(\Theta_{m_k}, M_{ijkr}, N_{ijkr}) \\
I_{m_{ijkr}} &= I_m(\Theta_{m_k}, M_{ijkr}, N_{ijkr}) \\
\sigma_{ijkr} &= \sigma(\Theta_{e_j}, \Theta_{b_i}, U_{m_{ijkr}}, I_{m_{ijkr}}) \\
I_{e_{ijkr}} &= \sigma_{ijkr} I_{m_{ijkr}} \\
I_{b_{ijkr}} &= nI_{e_{ijkr}} + I_{other} \\
t_{hover_{ijkr}} &= t_{hover}(\Theta_{b_i}, I_{b_{ijkr}})
\end{aligned}
\tag{3.19}
$$

where the 0.25 factor on the first equation of the system 3.19 is due to the extra weight of wires and other electrical components, the second equation of 3.19 is due the fact the thrust generated by the propulsor group has to equals the weight, and the third equation of 3.19 is obtained from the expression of the thrust on equation 3.3.

Furthermore, the ESC input voltage is obtained as

$$U_{e_{ijkr}} = U_{b_{ijkr}} - I_{b_{ijkr}} R_{b_{ijkr}} \tag{3.20}$$

The maximum endurance on hover is obtained from

$$t_{eMAX} = \max t_{hover_{ijkr}} \tag{3.21}$$

The ESC input voltage for the maximum endurance on hover is the $U_{e_{ijkr}}$ for the set $ijkr$ that satisfy equation 3.21.

3.3.2 Maximum System Efficiency

The system efficiency is the indicator of how much of the electrical energy generated by the battery is converted into mechanical energy, and therefore thrust. As higher is the efficiency as lower is the dissipative losses.

The efficiency of the propulsor set is defined on the maximum throttle mode ($\sigma = 1$), meaning that the battery voltage U_b reaches its peak. Using a numerical solver for non-linear equations, M, N and I_m are determined through the following system of non-linear

equations

$$\begin{aligned} \frac{U_{m_{ijkr}} + I_{m_{ijkr}} R_{m_k}}{U_{b_i}} &= 1 \\ M_{ijkr} - C_{M_r} \rho (\frac{N_{ijkr}}{60})^2 D_{p_r}^5 &= 0 \\ U_{m_{ijkr}} - U_m(\Theta_{m_k}, M_{ijkr}, N_{ijkr}) &= 0 \\ I_{m_{ijkr}} - I_m(\Theta_{m_k}, M_{ijkr}, N_{ijkr}) &= 0 \end{aligned} \tag{3.22}$$

where the first equation of the system 3.22 is the expression of the maximum throttle mode. The system of equations 3.22 is derived from the equations 3.14, 3.3, and 3.12.

From the results of 3.22 one can proceed as

$$\begin{aligned} I_{e_{ijkr}} &= I_{m_{ijkr}} \\ I_{b_{ijkr}} &= n I_{e_{ijkr}} + I_{other} \\ U_{e_{ijkr}} &= U_{b_{ijkr}} - I_{b_{ijkr}} R_{b_{ijkr}} \end{aligned} \tag{3.23}$$

The system efficiency of the set $ijkr$ is then given by the ratio of the propeller output mechanical power and the power generated by the battery as

$$\eta_{ijkr} = \frac{\frac{2\pi}{60} n M_{ijkr} N_{ijkr}}{U_{b_i} I_{b_{ijkr}}} \tag{3.24}$$

The maximum system efficiency is represent by the set $ijkr$ that satisfy the following expression

$$\eta_{MAX} = \max \eta_{ijkr} \tag{3.25}$$

The ESC input voltage for the maximum system efficiency is the $U_{e_{ijkr}}$ for the set $ijkr$ that satisfy the equation 3.25.

3.3.3 Maximums Payload Mass, Produced Propeller Thrust and Pitch Angle

From the results of the system of equations 3.22, the maximum payload mass possible for the $ijkr$ set is given by

$$m_{LOAD_{ijkr}} = n T_{ijkr} - m_{ijkr} g \tag{3.26}$$

The maximum pitch angle possible for the set $ijkr$ is given by

$$\theta_{MAXijkr} = arcos\frac{m_{ijkr}g}{nT_{ijkr}} \tag{3.27}$$

From the results above is possible to determine the maximum payload mass it possible be achieved from all sets as

$$m_{MAXLOAD} = \max m_{LOAD_{ijkr}} \tag{3.28}$$

Similarly, the maximum pitch angle among the sets is given by

$$\theta_{MAX} = \max \theta_{MAXijkr} \tag{3.29}$$

Although it may be confusing, each set $ijkr$ presents a maximum pitch angle denoted as $\theta_{MAXijkr}$. The maximum value of $\theta_{MAXijkr}$ from all the components is the Maximum Pitch Angle θ_{MAX}.

3.3.4 Maximum Forward Flight Speed

As show in figure 16 on a forward movement (neglecting blade flapping effects) the thrust generated by the action of the propellers is split in a component parallel to the direction of the forward flight and a component parallel to the gravitational field. In order to have the movement and the multicopter in a fixed altitude h, the component parallel to the movement has to at least equals the *drag force* F_{drag}, the dissipative force contrary the movement of the UAV; and the component parallel to the gravitational field has to equals the weight G of the multicopter. Then

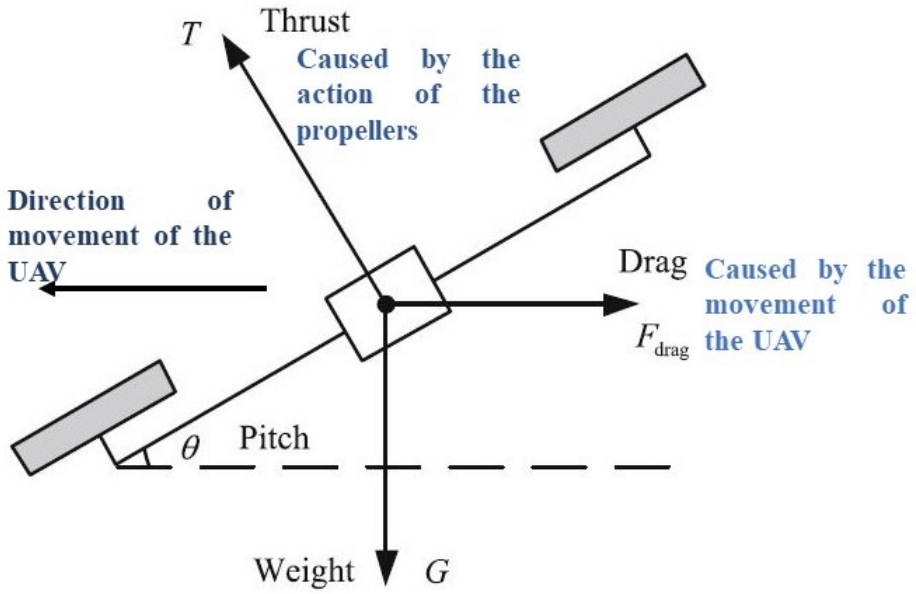
$$F_{drag} = mgtan\theta \tag{3.30}$$

The expression of the drag force is given by [10]

$$F_{drag} = \frac{1}{2}C_D\rho V^2 S \tag{3.31}$$

where V [m/s] is the forward flight speed, S [m^2] is the maximum surface area of the multicopter given by equation 3.11 and C_D is the drag coefficient of the whole multicopter. The drag coefficient C_D is function of the pitch angle θ and can be evaluated experimentally through a CFD analyses. The fitting curve for the CFD results obtained

Figure 16: Forces in the Multicopter



Source: Adapted from [3]

in [11] gives the following relation to C_D

$$C_D = C_{D1}(1 - sin^3\theta) + C_{D2}(1 - cos^3\theta) \tag{3.32}$$

where C_{D1} is equal to 3 and C_{D2} is equal to 1.5. Although the expression 3.32 may not be exactly the same for the quadrotor in study, the expression will be used as a rough approximation in a first effort towards the design.

Isolating the forward speed V in the equation 3.31 and replacing the equation 3.32 on it, the following expression comes

$$V(\theta) = \sqrt{\frac{2mgtan\theta}{\rho S[C_{D1}(1 - sin^3\theta) + C_{D2}(1 - cos^3\theta)]}} \tag{3.33}$$

As it is highlighted in equation 3.33, the forward speed of the multicopter is function of its pitch angle.

For the pitch angle θ interval $0 \leq \theta \leq \frac{\pi}{2}$ the equation 3.33 represents a rising function in θ. Therefore, the maximum forward flight speed for a generic multicopter can be defined by

$$V_{MAX} = V(\theta_{MAX}) \tag{3.34}$$

For a multicopter with the $ijkr$ set of components, the equation 3.33 is written as

$$V_{ijkr}(\theta) = \sqrt{\frac{2m_{ijkr}gtan\theta}{\rho S_r[C_{D1}(1 - sin^3\theta) + C_{D2}(1 - cos^3\theta)]}} \tag{3.35}$$

moreover, the maximum forward flight speed for the multicopter with the $ijkr$ set of components is then

$$V_{MAX} = \max V_{ijkr}(\theta_{MAX}) \tag{3.36}$$

Note that the equations 3.34 and 3.36 are different in essence: the equation 3.34 represents the maximum forward flight speed of a given multicopter with its maximum possible pitch angle; while equation 3.36 represents the set with the maximum forward flight speed among all the maximums forward flight speed of all sets. Although the same notation is used here for simplification, the expression "Maximum Forward Flight Speed" from now on on the text will represent the function 3.36 save if told otherwise.

3.3.5 Maximum Flight Distance

In order to calculate the maximum flight distance, the total flight time has to be evaluated. The set of equations to calculate the total flight time is similar to that of the endurance in hover.

From the figure 16 a $ijkr$ set has the following expression for the thrust T_{ijkr}

$$T_{ijkr} = \frac{m_{ijkr}g}{ncos\theta} \tag{3.37}$$

Replacing equation 3.37 in the expression for thrust 3.3, the angular speed N can be isolated and written as

$$N_{ijkr} = 60\sqrt{\frac{m_{ijkr}g}{\rho C_{T_r} D_{p_r}^4 ncos\theta}} \tag{3.38}$$

And the torque M_{ijkr} is expressed by

$$M_{ijkr} = \frac{m_{ijkr} g C_{M_r} D_{p_r}}{C_{T_r} n cos\theta} \tag{3.39}$$

Hence, the total time of flight of the multicopter compounded by the $ijkr$ set, namely $t_{fly_{ijkr}}$ is calculated through

$$\begin{aligned} U_{m_{ijkr}} &= U_m(\Theta_{m_k}, M_{ijkr}, N_{ijkr}) \\ I_{m_{ijkr}} &= I_m(\Theta_{m_k}, M_{ijkr}, N_{ijkr}) \\ \sigma_{ijkr} &= \sigma(\Theta_{e_j}, \Theta_{b_i}, U_{m_{ijkr}}, I_{m_{ijkr}}) \\ I_{e_{ijkr}} &= \sigma_{ijkr} I_{m_{ijkr}} \\ I_{b_{ijkr}} &= n I_{e_{ijkr}} + I_{other} \\ t_{fly_{ijkr}} &= t_{fly}(\Theta_{b_i}, I_{b_{ijkr}}) \end{aligned} \tag{3.40}$$

Note that the set of equations 3.40 follows the similar procedure it has been seen in the section 3.3.1 for the hover endurance and the flight time is given in *minutes*.

The flight distance for the set $ijkr$, namely Z_{ijkr} [m], can be then written as a function of θ as

$$Z_{ijkr}(\theta) = 60 V_{ijkr}(\theta) t_{fly_{ijkr}}(\theta) \tag{3.41}$$

The maximum flight distance Z_{MAX} is then calculated as

$$Z_{MAX} = \max Z_{ijkr} \tag{3.42}$$

3.3.6 Financial Aspects

Apart from the technical parameters, two financial related parameters will be discussed briefly: the cost and the price.

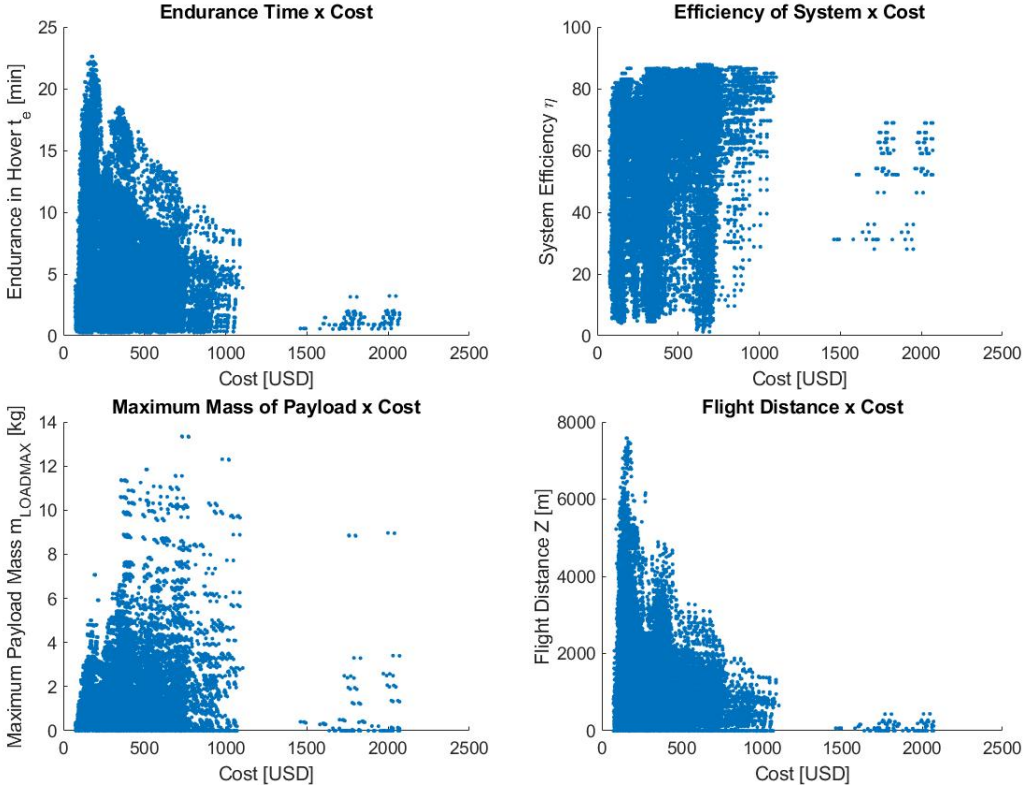
For now, the costs taken into consideration are exclusively that of the propulsor group. Meanwhile, the pricing analysis was made through a survey with around a hundred people.

3.3.6.1 Costs

The cost of the propulsion system is one of the key parameters for the project, and as it will be presented later on, it has a weight on the decision criterion.

The figure 17 shows how *endurance time*, *efficiency of the system*, *payload mass* and *flight distance* changes in function of cost for the propulsor sets of a quadrotor case.

Figure 17: Theoretical Results for a Quadrotor



Source: Author

3.3.6.2 Pricing

For a product be likely profitable its price has to be bigger than its costs. In other words, the market has to be able to pay for a product, more than it costs to develop and manufacture it.

In a survey with around 100 people from 20 countries, an indirect question that could be translated simply into "how much would you be willing to invest in a drone?" was asked. A scale called willingness from 0 to 5 was then defined and the results obtained were

- the willingness for paying until 50 USD for the drone was of 3.4 out of 5;
- the willingness for paying until 100 USD for the drone was of 3.1 out of 5;
- the willingness for paying until 500 USD for the drone was of 1.8 out of 5;
- the willingness for paying until 1000 USD for the drone was of 0.8 out of 5;

From the results, ideally, the costs of the drone should be at least lower than 100 USD in order to have a product that might be profitable.

Its important to remind this is a really simplistic analysis that is not adequate in this specific case, since a highly potential market strategy to be used would be the one where the client is not buying the drone itself, but the service offered for it. In fact, as it will be seen it on the next chapters, the margin for revenue by selling the drone itself for a final client would nonexistent, since the costs of the propulsion components themselves are around twice the price determined by the survey.

Although simplistic however the study is important to define a market strategy by cutting off a first approach of selling drones to the final consumer. In order to have a competitive product in this way, the costs would have to be reduced by cutting intermediaries and therefore manufacturing most of components. Moreover, for this, a further study of market would be needed, what will not be covered here, since it is out of the scope of the project.

3.4 Compatibility Aspects

All the combinations of the components have the compatibility evaluated in 3 aspects: the *maximum angular Speed*, *Motor/ESC Compatibility*, and *Battery Current Compatibility*. Naturally, if the set does not match any of those criteria, its assembly is physically impossible and the parameters assume the values on table 5.

Table 5: Compatibility Parameters

Parameter	**Value**
Endurance in Hover t_e	0
System Efficiency η	0
Propeller Thrust T	0
Payload Mass m_{LOAD}	0
Pitch Angle θ	0
Forward Speed V	0
Flight Distance Z	0
Mass of the set m	∞
Cost of the set	∞

Source: Author

3.4.1 Angular Speed Compatibility

The study of the parameters requires the calculation of the angular speed N from the propeller parameters (as in the equations 3.3). However, the angular speed calculated N_{calc} may be so high that a determined motor couldn't spin and make the flight possible. In this way, a threshold is established for each motor taking into considerations the maximum angular speeds given by its manufacturer. This threshold is the *maximum angular speed* N_{MAX}.

In this way, the angular speed compatibility establishes a condition for the pair propeller-motor where the relation

$$N_{calc_r} \leq N_{MAX_k} \tag{3.43}$$

has to be respected.

For case where N_{calc} is actually bigger than N_{MAX} the parameters are set as in table 5, indicating that the pair motor-k and propeller-r is not feasible and so is not the multicopter flight.

3.4.2 Motor/ESC Compatibility

In order to have the best efficiency of the motor, the current limit of the ESC has to be slightly higher than the maximum current for the motor. Hence, the compatibility of the pair motor k - ESC j is checked through

$$I_{eMAX_j} \geq 1.2 I_{mMAX_k} \tag{3.44}$$

In case the relation is not satisfied, the parameters are set as on the table 5.

3.4.3 Battery Current Compatibility

The battery current for the set $ijkr$ $I_{b_{ijkr}}$ cannot be greater than the maximum current that the battery can withstand. Therefore, the following relation must be respected

$$I_{b_{ijkr}} \leq \frac{C_{b_i} K_{b_i}}{1000} \tag{3.45}$$

with C_{b_i} in mAh and K_{b_i} in C.

For a $I_{b_{ijkr}}$ bigger than the threshold, the parameters assume the values of the table 5.

3.5 Results

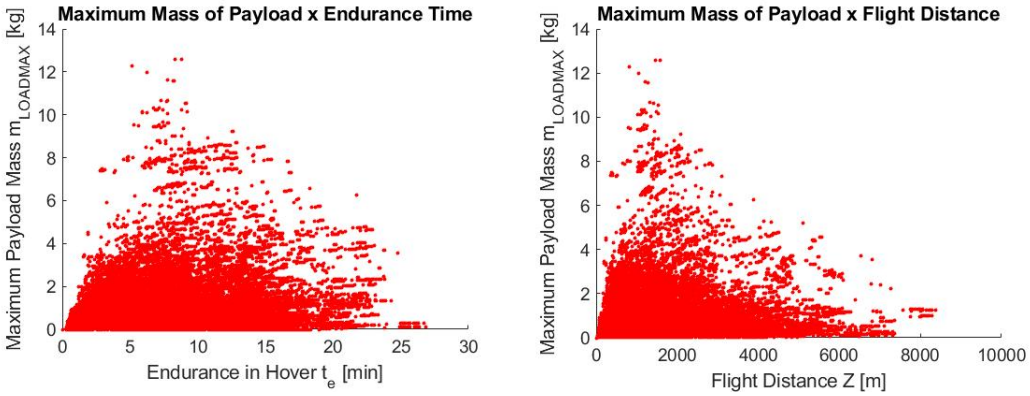
The models approached were implemented in a MATLAB simulation with

- 43 different types of propellers;
- 50 different types of motors;
- 38 different types of ESC's;
- 41 different types of batteries;
- 3 different multicopter configurations.

A total of 10,049,100 different combinations of sets were evaluated where the incompatible ones were excluded. The sets evaluated where those proposed on chapter 3.3, for what the set of components that optimize each of these parameters were determined.

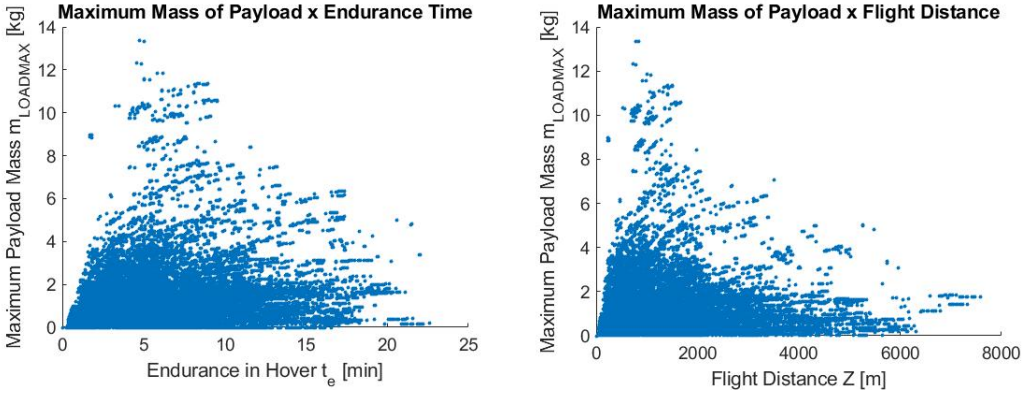
Figures 18, 19 and 20 present the distribution of the theoretical results for tricopters, quadrotors and hexacopter, respectively.

Figure 18: Theoretical Results for a Tricopter



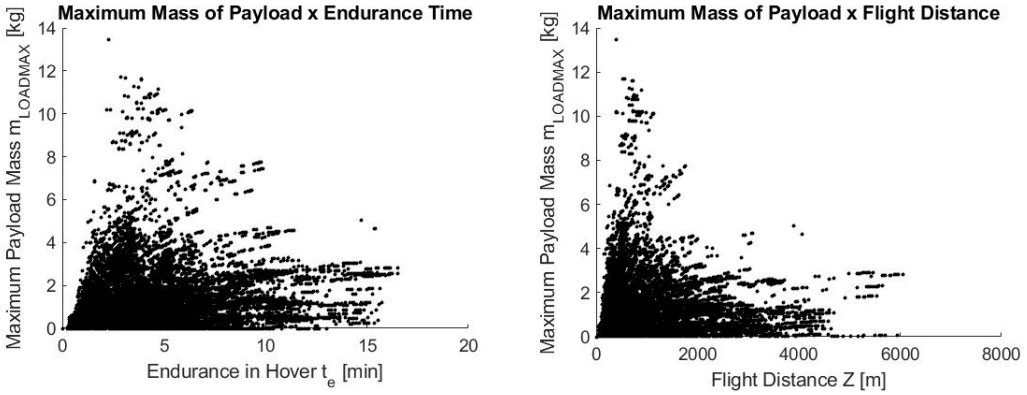
Source: Author

Figure 19: Theoretical Results for a Quadrotor



Source: Author

Figure 20: Theoretical Results for a Hexacopter



Source: Author

3.5.1 Best Set of Components by Optimal Parameters

This section presents the set of components that optimize some of the parameters introduced in chapter 3.3. Those results will be of great importance when defining the decision criterion, which will lead to the set of components to be used as base of the project.

The data of the components are presented in a first table and the results obtained in a second one. All the data and results present on the table are self-explanatory and extra comments will be used only where is strictly necessary.

3.5.1.1 Set with the Maximum Time of Endurance in Hover t_{eMAX}

For the maximum time of endurance in hover, the propeller, the motor and the battery models are the same for both the tricopter and quadcopter cases. On the meantime, the ESC model is the same for all cases.

Table 6: Maximum Time of Endurance in Hover Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp	0.1270 m		0.0650 m
	Hp	0.1143 m		0.1118 m
	Bp	2		2
	m_p	0.0037 kg		0.0004 kg
	m_{af}	0.0886 kg	0.1483 kg	0.0841 kg
	S	0.0557 m^2	0.0779 m^2	0.0345 m^2
Motor	K_{V0}	1400 RPM/V		4500 RPM/V
	I_{mMAX}	12 A		14. 4 A
	I_{m0}	0.2 A		0.76 A
	U_{m0}	10 V		7 V
	R_m	0.325 Ω		0.145 Ω
	m_m	0.018 kg		0.0085 kg
ESC	I_{eMAX}	40 A		
	R_e	0.01 Ω		
	m_e	0.0026 kg		
Battery	C_b	3800 mAh		3700 mAh
	R_b	0.016 Ω		0.016 Ω
	U_b	7.4 V		14.8V
	K_b	65 C		45 C
	m_b	0.14 kg		0.285 kg

Source: Author

Table 7: Maximum Time of Endurance in Hover Results

	3Rotor	**4Rotor**	**6Rotor**
Endurance in Hover t_{eMAX}	26.8 min	22.6 min	16.54 min
System Efficiency η	72.55%	74.47 %	69.68 %
Produced Propeller Thrust T	1.53 N	1.53 N	5.47 N
Payload mass m_{LOAD}	0.113 kg	0.178 kg	2.819 kg
Maximum Pitch Angle θ_{MAX}	40.68 °	44.69 °	80.79 °
Maximum Flight Speed V_{MAX}	5.68 m/s	5.84 m/s	32.38 m/s
Maximum Flight Distance Z_{MAX}	6395.07 m	5071.74 m	6050.28 m
Mass of Multicopter m	0.3546 kg	0.4447 kg	0.5264 kg
Cost of Propulsors	138.82 USD	181.08 USD	205.93 USD

Source: Author

3.5.1.2 Set with the Maximum System Efficiency η_{MAX}

Table 8: Maximum System Efficiency Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp		0.2540 m	
	Hp		0.2286 m	
	Bp		2	
	m_p		0.0041 kg	
	m_{af}	0.3545 kg	0.5932 kg	1.2838 kg
	S	0.223 m^2	0.312 m^2	0.533 m^2
Motor	K_{V0}		450 RPM/V	
	I_{mMAX}		40 A	
	I_{m0}		0.6 A	
	U_{m0}		22 V	
	R_m		0.06 Ω	
	m_m		0.085 kg	
ESC	I_{eMAX}		60 A	
	R_e		0.01 Ω	
	m_e		0.06 kg	
Battery	C_b		3700 mAh	
	R_b		0.016 Ω	
	U_b		14.8V	
	K_b		45 C	
	m_b		0.285 kg	

Source: Author

Table 9: Maximum System Efficiency Results

	3Rotor	**4Rotor**	**6Rotor**
Endurance in Hover t_{eMAX}	17.27 min	13.25 min	8.12 min
System Efficiency η	87.31 %	87.83 %	88.35 %
Produced Propeller Thrust T	11.60 N	11.60 N	11.60 N
Payload mass m_{LOAD}	2.277 kg	3.03 kg	4.336 kg
Maximum Pitch Angle θ_{MAX}	68.75 °	68.75 °	67.04 °
Maximum Flight Speed V_{MAX}	11.53 m/s	11.25 m/s	10.27 m/s
Maximum Flight Distance Z_{MAX}	3889.50 m	2891.01 m	1707.58 m
Mass of Multicopter m	1.269 kg	1.694 kg	2.757 kg
Cost of Propulsors	487.69 USD	634.45 USD	927.97 USD

Source: Author

3.5.1.3 Set with the Maximum Produced Propeller Thrust T_{MAX} and the Maximum Payload Mass $m_{MAXLOAD}$

In this case, the payload mass $m_{MAXLOAD}$ is around 13 kg for all the multicopter configurations, what is a really significant result. However, the endurance in hover is around 5 min for the tricopter and quadrotor and less than 3 minutes for the hexacopter. Furthermore, as it will be present on the next pages, the time of flight decreases exponentially with the increase of the payload mass being carried by the UAV. Therefore, those sets would be unviable for the delivering application.

Table 10: Maximum Produced Propeller Thrust and Maximum Payload Mass Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp	0.5080 m	0.457 m	0.457 m
	Hp	0.1524 m	0.254 m	0.1829 m
	Bp	2	2	2
	m_p	0.0470 kg	0.0041 kg	0.0320 kg
	m_{af}	1.418 kg	1.922 kg	4.1594 kg
	S	0.891 m^2	1.01 m^2	1.727 m^2
Motor	K_{V0}	300 RPM/V	240 RPM/V	
	I_{mMAX}	40 A	40 A	
	I_{m0}	1.1 A	0.9 A	
	U_{m0}	15 V	18 V	
	R_m	0.063 Ω	0.085 Ω	
	m_m	0.136 kg	0.136 kg	
ESC	I_{eMAX}	60 A		
	R_e	0.01 Ω		
	m_e	0.06 kg		
Battery	C_b	1400 mAh	1300 mAh	1400 mAh
	R_b	0.016 Ω	0.016 Ω	0.016 Ω
	U_b	22.2 V	22.2 V	22.2 V
	K_b	120 C	120 C	120 C
	m_b	0.230 kg	0.212 kg	0.230 kg

Source: Author

3.5.1.4 Set with the Maximum Payload Mass $m_{MAXLOAD}$ - Tricopter

Particularlly for the tricopter case, the set with the maximum payload mass differs from the one with maximum produced thrust and its data and results are presented in this dedicated section.

Table 11: Maximum Produced Propeller Thrust and Maximum Payload Mass Results

	3Rotor	**4Rotor**	**6Rotor**
Endurance in Hover t_{eMAX}	5.15 min	4.75 min	2.30 min
System Efficiency η	79.70 %	82.21 %	85.24 %
Produced Propeller Thrust T	48.73 N	40.56 N	32.09 N
Payload mass m_{LOAD}	12.287 kg	13.35 kg	13.47 kg
Maximum Pitch Angle θ_{MAX}	79.64 °	79.07 °	71.62 °
Maximum Flight Speed V_{MAX}	13.41 m/s	13.14 m/s	10.09 m/s
Maximum Flight Distance Z_{MAX}	820.38 m	787.62 m	397.80 m
Mass of Multicopter m	2.617 kg	3.187 kg	6.157 kg
Cost of Propulsors	683.50 USD	730.20 USD	1539.00 USD

Source: Author

Table 12: Maximum Payload Mass Data - 3Rotor

Propeller	Dp	0.3810 m
	Hp	0.254 m
	Bp	2
	m_p	0.0041 kg
	m_{af}	0.7975 kg
	S	0.501 m^2
Motor	K_{V0}	340 RPM/V
	I_{mMAX}	40 A
	I_{m0}	0.9 A
	U_{m0}	22 V
	R_m	0.055 Ω
	m_m	0.1080 kg
ESC	I_{eMAX}	60 A
	R_e	0.01 Ω
	m_e	0.06 kg
Battery	C_b	1300 mAh
	R_b	0.016 Ω
	U_b	22.2 V
	K_b	120 C
	m_b	0.2120 kg

Source: Author

3.5.1.5 Set with the Maximum Forward Speed V_{MAX}

As it can be seen on the table 15 the relatively high flight speeds make those sets strong candidates for racing drones.

Table 13: Maximum Payload Mass Results - 3Rotor

Endurance in Hover t_{eMAX}	8.31 min
System Efficiency η	82.98 %
Produced Propeller Thrust T	46.76 N
Payload mass m_{LOAD}	12.590 kg
Maximum Pitch Angle θ_{MAX}	83.08 °
Maximum Flight Speed V_{MAX}	17.96 m/s
Maximum Flight Distance Z_{MAX}	1470.39 m
Mass of Multicopter m	1.7079 kg
Cost of Propulsors	518.57 USD

Source: Author

Table 14: Maximum Forward Speed Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp	0.0650 m	0.0406 m	0.0650 m
	Hp	0.1118 m	0.0914 m	0.1118 m
	Bp	2	4	2
	m_p	0.0004 kg	0.0012 kg	0.0004 kg
	m_{af}	0.0232 kg	0.0152 kg	0.0841 m
	S	0.0145 m^2	0.0080 m^2	0.0345 m^2
Motor	K_{V0}	4500 RPM/V	6500 RPM/V	4100 RPM/V
	I_{mMAX}	14.4 A	15.1 A	20.2 A
	I_{m0}	0.76 A	0.59 A	1.2 A
	U_{m0}	7 V	7 V	10 V
	R_m	0.145 Ω	0.12 Ω	0.097 Ω
	m_m	0.0085 kg	0.0085 kg	0.0165 kg
ESC	I_{eMAX}	60 A	40 A	
	R_e	0.01 Ω	0.01 Ω	
	m_e	0.0600 kg	0.0026 kg	
Battery	C_b	1300 mAh	3700 mAh	
	R_b	0.016 Ω	0.016 Ω	
	U_b	22.2 V	14.8 V	
	K_b	120 C	45 C	
	m_b	0.2120 kg	0.285 kg	

Source: Author

Table 15: Maximum Forward Speed Results

	3Rotor	**4Rotor**	**6Rotor**
Endurance in Hover t_{eMAX}	6.03 min	7.97 min	13.85 min
System Efficiency η	64.33 %	65.07 %	77.11 %
Produced Propeller Thrust T	10.23 N	3.64 N	5.73 N
Payload mass m_{LOAD}	2.5805 kg	1.05 kg	2.9175 kg
Maximum Pitch Angle θ_{MAX}	80.21 °	72.77 °	80.21 °
Maximum Flight Speed V_{MAX}	48.02 m/s	41.46 m/s	32.97 m/s
Maximum Flight Distance Z_{MAX}	3392.98 m	5245.33 m	5461.31 m
Mass of Multicopter m	0.5466 kg	0.4327 kg	0.5864 kg
Cost of Propulsors	267.72	156.15 USD	202.75 USD

Source: Author

3.5.1.6 Set with the Maximum Flight Distance Z_{MAX}

As it can be seen on table 17, the high endurance time and payload mass and the relatively low costs, make those sets strong candidates for the delivering application.

Table 16: Maximum Flight Distance Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp		0.0650 m	
	Hp		0.1118 m	
	Bp		2	
	m_p		0.0004 kg	
	m_{af}	0.0232 kg	0.0388 kg	0.0841 kg
	S	0.0145 m^2	0.0204 m^2	0.0349 m^2
Motor	K_{V0}		4500 RPM/V	
	I_{mMAX}		14.4 A	
	I_{m0}		0.76 A	
	U_{m0}		7 V	
	R_m		0.145 Ω	
	m_m		0.0085 kg	
ESC	I_{eMAX}		40 A	
	R_e		0.01 Ω	
	m_e		0.0026 kg	
Battery	C_b		3700 mAh	
	R_b		0.016 Ω	
	U_b		14.8 V	
	K_b		45 C	
	m_b		0.285 kg	

Source: Author

Table 17: Maximum Flight Distance Results

	3Rotor	**4Rotor**	**6Rotor**
Endurance in Hover t_{eMAX}	16.78 min	17.24 min	16.54 min
System Efficiency η	69.24 %	69.46 %	69.68 %
Produced Propeller Thrust T	5.47 N	5.47 N	5.47 N
Payload mass m_{LOAD}	1.250 kg	1.778 kg	2.819 kg
Maximum Pitch Angle θ_{MAX}	75.63 °	78.49 °	80.79 °
Maximum Flight Speed V_{MAX}	33.51 m/s	33.74 m/s	32.38 m/s
Maximum Flight Distance Z_{MAX}	8364.99 m	7579.94 m	6050.28 m
Mass of Multicopter m	0.4225 kg	0.4525 kg	0.5264 kg
Cost of Propulsors	126.67 USD	153.09 USD	205.93 USD

Source: Author

3.5.1.7 Set with the Minimum Cost

Table 18: Minimum Cost Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp		0.0310 m	
	Hp		0.0483 m	
	Bp		3	
	m_p		0.0003 kg	
	m_{af}	0.0053 kg	0.0088 kg	0.0191 kg
	S	0.0033 m^2	0.0046 m^2	0.0079 m^2
Motor	K_{V0}		8000 RPM/V	
	I_{mMAX}		11.5 A	
	I_{m0}		0.58 A	
	U_{m0}		3 V	
	R_m		0.163 Ω	
	m_m		0.0042 kg	
ESC	I_{eMAX}		15 A	
	R_e		0.01 Ω	
	m_e		0.0044 kg	
Battery	C_b		250 mAh	
	R_b		0.016 Ω	
	U_b		4.35 V	
	K_b		60 C	
	m_b		0.0065 kg	

Source: Author

Table 19: Minimum Cost Results

	3Rotor	**4Rotor**	**6Rotor**
Endurance in Hover t_{eMAX}	2.43 min	1.95 min	1.35 min
System Efficiency η	49.92 %	52.14 %	54.56 %
Produced Propeller Thrust T	0.18 N	0.18 N	0.18 N
Payload mass m_{LOAD}	0.010 kg	0.014 kg	0.019 kg
Maximum Pitch Angle θ_{MAX}	34.38 °	35.52 °	33.80 °
Maximum Flight Speed V_{MAX}	7.38 m/s	7.33 m/s	6.71 m/s
Maximum Flight Distance Z_{MAX}	901.85 m	701.38 m	447.24 m
Mass of Multicopter m	0.0468 kg	0.0614 kg	0.0940 kg
Cost of Propulsors	55.33 USD	72.34 USD	106.43 USD

Source: Author

3.5.1.8 Best Overall Set of Components

On the section 3.5.1 the sets that optimize the parameters introduced in 3.3 were obtained. However, for a determined application, the ideal would be select a set of components that optimizes a series of key parameters simultaneously. Therefore, a decision criterion should be used in order to determine a single set that could match with the application proposed. Hence, the **criterion P** expression is defined as

$$P_{ijkr} = \frac{\sum_{n}^{N} w_n \frac{p_n}{p_{nMAX}}}{\sum_{n}^{N} w_n} \tag{3.46}$$

where N is the number of parameters in study, w_n is the weight of the parameter p_n, and p_{nMAX} is the maximum value the parameter p_n in the set of values in study. Ideally, the set $ijkr$ holds all parameters as them maximum values and P_{ijkr} is equal to 1.

For the delivering application, the key parameters are the time of endurance, the payload mass and the flight distance, being the two first parameters the most important ones. Then, the equation 3.46 assumes the values from table 20.

Table 20: Criterion P Data

	Parameter	**Weight**
p_1	Time of Endurance t_e	$w_1 = 4$
p_2	Inverse of Propulsor Cost $Cost^{-1}$	$w_2 = 1$
p_3	System Efficiency η	$w_3 = 1$
p_4	Payload Mass m_{LOAD}	$w_4 = 4$
p_5	Flight Speed V	$w_5 = 1$
p_6	Flight Distance Z	$w_6 = 2$

Source: Author

By evaluating the criterion P for each of the set of parameters for the 3Rotor, 4Rotor and 6Rotor, the set of components which best optimize the parameters according to 3.46 was determined. The data for the sets chosen are presented in table 21 and the results summed up on table 22.

Naturally, increasing the the mass carried by the UAV, the time of endurance will decrease. The figure 21 shows how the time of endurance decreases exponentially as the payload mass is increased up to the maximum admissible value of payload mass for the three multicopter configurations. It can be seen that even for the best set of components, when the multicopter is carrying the maximum payload, the endurance time is reduced to a few minutes for all curves. Nevertheless, the blue curve (tricopter) has an endurance of around 25 min, and performs better than the hexacopter in terms of endurance and is able to carry more weight on. However, the quadrotor presents a slight better performance for payloads heavier than 1 kg and a higher payload capacity, and despite having a higher propulsor cost and a lower endurance-cost ratio than the tricopter, presents less complications for the controlling. In this way, for a flight where the quadrotor carries the maximum payload, the flight time is around 2 minutes, which makes the UAV suitable for a delivery distance radius of 1.4 km at maximum speed considering the return to the

base.

Therefore, the configuration to be used on the project will be the **4Rotor**. The complete data for the components and the results obtained are present in the tables 21 and 22.

Table 21: Best Overall Set of Components Data

		3Rotor	**4Rotor**	**6Rotor**
Propeller	Dp	0.1524 m		0.0650 m
	Hp	0.1143 m		0.1118 m
	Bp	2		2
	m_p	0.0049 kg		0.0004 kg
	m_{af}	0.1276 kg	0.2135 kg	0.0232 kg
	S	0.080 m^2	0.112 m^2	0.014 m^2
Motor	K_{V0}	1700 RPM/V		4500 RPM/V
	I_{mMAX}	37.2 A		14.4 A
	I_{m0}	1 A		0.76 A
	U_{m0}	10 V		7 V
	R_m	0.076 Ω		0.145 Ω
	m_m	0.0331 kg		0.0085 kg
ESC	I_{eMAX}	45 A		40 A
	R_e	0.01 Ω		0.01 Ω
	m_e	0.0042 kg		0.0026 kg
Battery	C_b	3700 mAh		
	R_b	0.016 Ω		
	U_b	14.8 V		
	K_b	45 C		
	m_b	0.285 kg		

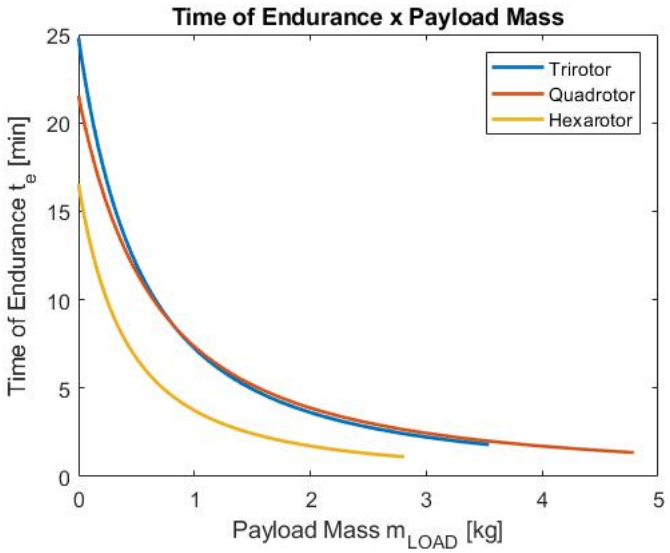
Source: Author

Table 22: Best Overall Set of Components Results

	3Rotor	**4Rotor**	**6Rotor**
Criterion P_{MAX}	0.60	0.63	0.66
Endurance in Hover t_{eMAX}	24.8 min	21.5 min	16.5 min
System Efficiency η	77.72 %	77.92 %	69.68 %
Produced Propeller Thrust T	13.68 N	13.68 N	5.47 N
Payload mass m_{LOAD}	3.542 kg	4.798 kg	2.819 kg
Maximum Pitch Angle θ_{MAX}	81.36 °	81.93 °	80.79 °
Maximum Flight Speed V_{MAX}	23.94 m/s	23.51 m/s	32.38 m/s
Maximum Flight Distance Z_{MAX}	6802.59 m	5484.84 m	6050.28 m
Mass of Multicopter m	0.6423 kg	0.781 kg	0.526 kg
Cost of Propulsors	161.46 USD	199.48 USD	205.93 USD

Source: Author

Figure 21: Time of Endurance x Payload Mass



Source: Author

3.6 Summary

The aim of this chapter was to define and study proper models for the components of the propulsion system as a first step towards the design process.

After defining the environmental parameters, the models for propellers, motors, ESC's and battery are presented, and additionally, the mass of the airframe and its surface are defined as function of the propeller dimensions.

After the modelling process, the optimal parameters to be looking for are defined as:

- **Maximum time of endurance in hover** t_{eMAX}
- **Maximum system efficiency** η_{MAX}
- **Maximum produced propeller thrust** T_{MAX}
- **Maximum payload mass** $m_{MAXLOAD}$
- **Maximum pitch ange** θ_{MAX}
- **Maximum forward speed** V_{MAX}
- **Maximum flight distance** Z_{MAX}
- **Minimum mass of propulsors** m_{MIN}

- **Parameters of minimum cost**

For each of these items a set of components which maximize the particular parameter is determined and the results for each of those sets are presented for each configuration of multicopter. Such results show that each set might be used for a particular application, but none of them are completely adequate for the application the report focus on: delivering. Nonetheless, those results are important to define the maximum value of each of the parameter analyzed.

The last section defines the criterion decision method, and with the aim of the maximum values defined previously, the set that best fit the delivering application is determined in three configurations: tricopter, quadrotor and hexacopter.

The set chosen was then determined and the data of the components are once more presented on the following table.

Table 23: Best Overall Set of Components Data

Propeller	Dp	0.1524 m
	Hp	0.1143 m
	Bp	2
	m_p	0.0049 kg
	m_{af}	0.1276 kg
	S	0.080 m^2
Motor	K_{V0}	1700 RPM/V
	I_{mMAX}	37.2 A
	I_{m0}	1 A
	U_{m0}	10 V
	R_m	0.076 Ω
	m_m	0.0331 kg
ESC	I_{eMAX}	45 A
	R_e	0.01 Ω
	m_e	0.0042 kg
Battery	C_b	3700 mAh
	R_b	0.016 Ω
	U_b	14.8 V
	K_b	45 C
	m_b	0.285 kg

Source: Author

The results are summarized once more as well.

Table 24: Best Overall Set of Components Results

Criterion P_{MAX}	0.63
Time of Endurance in Hover t_{eMAX}	21.5 min
System Efficiency η	77.92 %
Produced Propeller Thrust T	13.68 N
Payload mass m_{LOAD}	4.798 kg
Maximum Pitch Angle θ_{MAX}	81.93 °
Maximum Forward Flight Speed V_{MAX}	23.51 m/s
Maximum Flight Distance Z_{MAX}	5484.84 m
Mass of Multicopter m	0.781 kg
Cost of Propulsors	199.48 USD

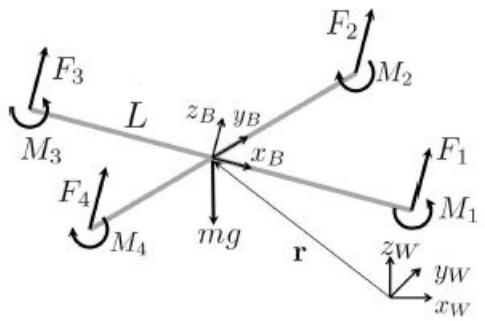
Source: Author

4 CONTROL SIMULATION

4.1 Overview of Modeling

As it has been presented on a previously, the movement of the quadrotor can be described by two set of equations: the Newton's Second Law which describes the acceleration of the center of mass of the quadrotor and therefore its linear movement; and the Euler's Equations which describes the angular movement. On the figure 22 the forces and moments acting on the airframe of the qudrotor are presented, where the forces F_i are the forces generated by the propeller and are responsible for the vertical movement, and the roll and pitch of the quadrotor; while the moments M_i are the drag moments also generated by the action of the propellers and responsible for the yaw movement. The directions x_B, y_B and z_B ($B\, b_1 b_2 b_3$) define the systems of coordinates of body frame, meanwhile x_W, y_W and z_W ($E\, e_1 e_2 e_3$, on figure 2) define the inertial frame. As the equations for the movement will be described in the body frame, for simplicity of notation the position vector will be described simply as $\mathbf{r} = [x, y, z]$. Lastly, the aerodynamical model will be presented due to its importance on the evaluation of the motor dynamics.

Figure 22: Forces and moments acting on the quadrotor frame



Source [12]

4.2 Newton's Second Law

As previously presented, the Newton's second Law for the quadrotor can be described as

$$m\ddot{\mathbf{r}} = mg\mathbf{e_3} + \mathbf{R}f \tag{4.1}$$

where $\mathbf{e_3} = [0\ 0\ 1]^T$ is the versor of the z direction, f is the total thrust generated by the rotors altogether and $\mathbf{R}$ is the rotation matrix defined as

$$R = \begin{bmatrix} c\psi c\theta - s\psi s\phi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ s\psi c\theta + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\theta s\phi c\psi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{4.2}$$

Notice that the blade flapping are not be taken into consideration, since its effects can be neglected in comparison with the other forces.

4.3 Euler's Equations

As presented previously, the Euler's Equations describe the rotational dynamics and can be written as

$$J \cdot \dot{\mathbf{\Theta}} = \tau - \mathbf{\Theta} \times J\mathbf{\Theta} \tag{4.3}$$

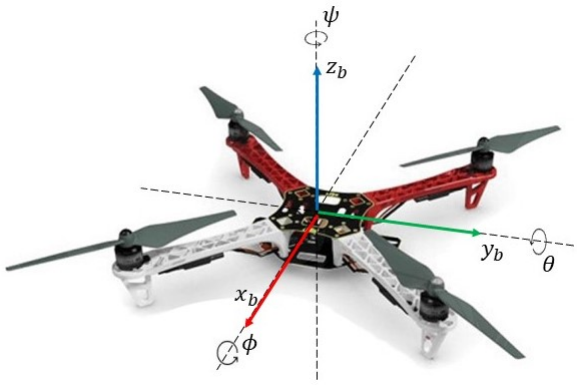
where $\mathbf{\Theta} = [\phi\ \theta\ \psi]^T$.

4.4 Aerodynamics Model

The set of equations for the thrust and moments acting on the airframe in a quadrotor moving in X configuration (figure 23) can be expressed as

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ \frac{l}{2}c_T & \frac{l}{2}c_T & -\frac{l}{2}c_T & -\frac{l}{2}c_T \\ -\frac{l}{2}c_T & \frac{l}{2}c_T & \frac{l}{2}c_T & -\frac{l}{2}c_T \\ c_M & -c_M & c_M & -c_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{4.4}$$

Figure 23: Body Coordinates System



Source: Adapted from [2]

4.5 Linearization of the Model

The set of equations 4.1 and 4.3 despite describing properly the dynamics of the quadrotor are non-linear and highly coupled which makes the analysis and control much more complex. Hence, to simplify the control some reasonable assumptions will be taken in order to linearize the equations. Firstly, the equations for the acceleration of the center of mass will be rewritten as

$$\begin{aligned} \ddot{x} &= \frac{f}{m}(c\psi s\theta + c\theta s\phi s\psi) \\ \ddot{y} &= \frac{f}{m}(s\psi s\theta - c\psi c\theta s\phi) \\ \ddot{z} &= \frac{f}{m}c\phi c\theta - g \end{aligned} \tag{4.5}$$

here two main assumptions are adopted:

- The angles are considered small (which according to [12] the model will be quite robust for angles up to 60°);

- The thrust generated by the motors approximate the weight of the quadrotor (hover condition).

According to the small angles approximation, one has the following relations: $sin\phi \approx \phi$, $cos\phi \approx 1$, $sin\theta \approx \theta$, $cos\theta \approx 1$ and moreover $f \approx mg$. Therefore the set of equations 4.5 can be rewritten as

$$\begin{aligned} \ddot{x} &= g(\phi s\psi + \theta c\psi) \\ \ddot{y} &= g(-\phi c\psi + \theta s\psi) \\ \ddot{z} &= \frac{f}{m} - g \end{aligned} \tag{4.6}$$

where the two first equations are considered to be approximately mg, while on the third equation this assumption is not valid, otherwise there wouldn't be any input on the altitude channel. Notice that 4.6 is now a linear model.

Similarly, the system of equations 4.3 can be rewritten as

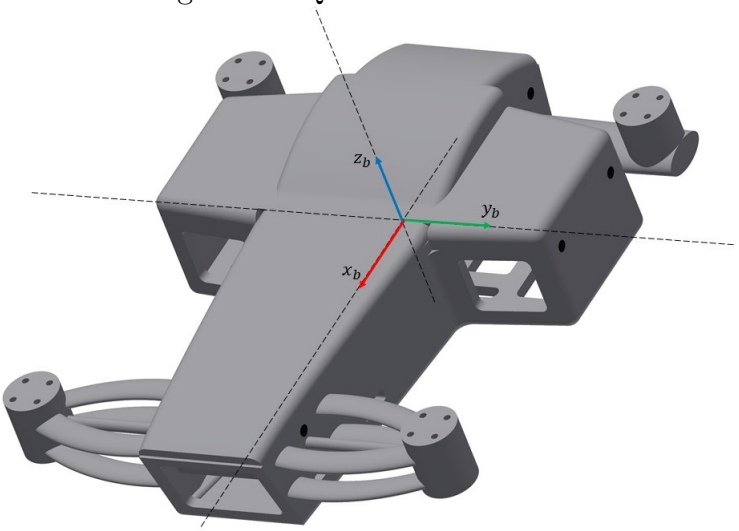
$$\begin{aligned} J_{xx}\dot{p} &= \tau_x - qr(J_{zz} - J_{yy}) \\ J_{yy}\dot{q} &= \tau_y - pr(J_{xx} - J_{zz}) \\ J_{zz}\dot{r} &= \tau_z - pq(J_{yy} - J_{xx}) \end{aligned} \tag{4.7}$$

It can be assumed the yaw angle remains constant and therefore the component r is small. Moreover, around the hover conditions the simplifications $p \approx \dot{\phi}$, $q \approx \dot{\theta}$ and $r \approx \dot{\psi}$ can be considered, which is translate into

$$\begin{aligned} J_{xx}\ddot{\phi} &= \tau_x \\ J_{yy}\ddot{\theta} &= \tau_y \\ J_{zz}\ddot{\psi} &= \tau_z - \dot{\phi}\dot{\theta}(J_{yy} - J_{xx}) \end{aligned} \tag{4.8}$$

Usually due to the simmetry $J_{yy} \approx J_{xx}$ and the subtractive term on the third equation can be neglected. However, this is not true for the quadrotor in question since there are no simmetry around the axis z_b as it can be seen by its airframe in figure 24.

Figure 24: Quadrotor airframe



Source: Author

4.6 The Nested Control Loop

After the overview of the model, the Nested Loop can finally be presented. The Nested Loop consists of the loops that form the controlling logic. As in figure 25 the 4 fundamentals blocks for the controlling of the quadrotor are:

- **Position Controller:** or *outer loop* is the block responsible for the translational dynamics. It has as input the desired trajectory and is feedback by the current position and velocity of the quadrotor in order to adjust the error. This block aims to deliver the desired roll and pitch angles and thrust.

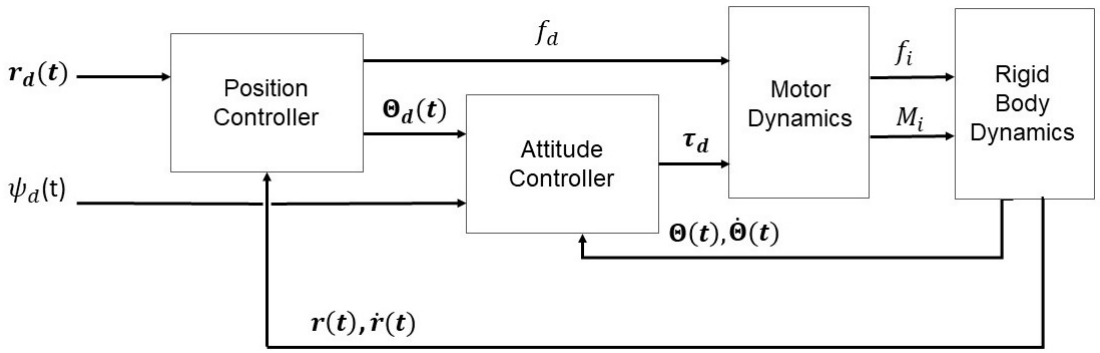
- **Attitude Controller:** or *inner loop* receives the desired angles command from the position controller and the desired yaw angle from the trajectory planner and is feedback by the current angular position and angular velocity from the quadrotor and with those data calculates the desired moments.

- **Motor Dynamics:** The motor dynamics block has as inputs the desired thrust from the position controller and the desired moments from the attitude controller. However, the motor dynamics has a delay before achieving the desired thrust and moments. Roughly speaking, this delay is the core of the motor dynamics block.

- **Rigid Body Dynamics:** This is the block which represents the quadrotor itself by means of its mathematical model and has as input the actual thrust and moments generated by the motors. In practice, the actual angular position and velocities as

the linear position and velocities of the quadrotor are estimated by sensors installed on the airframe and those values differ from the actual ones according to the dynamic of the sensor. However, in order to simplify the approach, the dynamic of the sensors will be neglected and the feedback values will be obtained through the mathematical model of the quadrotor.

Figure 25: The Nested Loop



Source: Author

4.6.1 Position Controller

The aim of the position controller (outer loop) is to generated as output the desired thrust for the motor, and the desired angular position for the attitude controller. In order to achieve such outputs a command acceleration $\mathbf{r_c}$ need to be generated by the position controller. Thus, the position error $\mathbf{e_p} \in^3$ is defined as the difference between the desired trajectory $\mathbf{r_d}$ and the actual trajectory $\mathbf{r}$, with $\mathbf{r_c}$,$\mathbf{r}$ and $\mathbf{r} \in^3$; and then $\mathbf{e_p} = (\mathbf{r_d} - \mathbf{r})$. Ideally the position error $\mathbf{e_p}$ should become null exponentially in time. From the Control Theory (as it described in details in [13]), this can be satisfied by means of a PID controller designed as

$$(\ddot{\mathbf{r}}_\mathbf{d} - \ddot{\mathbf{r}}_\mathbf{c}) + \mathbf{K_d}(\dot{\mathbf{r}}_\mathbf{d} - \dot{\mathbf{r}}) + \mathbf{K_p}(\mathbf{r_d} - \mathbf{r}) + \mathbf{K_i} \int (\mathbf{r_d} - \mathbf{r}) = 0 \tag{4.9}$$

where $\mathbf{K_d}$,$\mathbf{K_p}$ and $\mathbf{K_i} \in^{3x3}$ are the diagonal matrices of derivative, proportional and integrative gains, respectively. Isolating $\ddot{\mathbf{r}}_\mathbf{c}$ in the equation 4.10 one can have

$$\ddot{\mathbf{r}}_\mathbf{c} = \ddot{\mathbf{r}}_\mathbf{d} + \mathbf{K_d}(\dot{\mathbf{r}}_\mathbf{d} - \dot{\mathbf{r}}) + \mathbf{K_p}(\mathbf{r_d} - \mathbf{r}) + \mathbf{K_i} \int (\mathbf{r_d} - \mathbf{r}) \tag{4.10}$$

By multiplying the first of the equations 4.6 by $s\psi$ and the second by $-c\psi$ and summing both equations the following expression is obtained

$$\phi = g^{-1}(\ddot{x}s\psi - \ddot{y}c\psi) \tag{4.11}$$

Similarly, an expression for the pitch angle θ can be obtained as

$$\theta = g^{-1}(\ddot{x}c\psi + \ddot{y}s\psi) \tag{4.12}$$

Those equations can be used together with the command acceleration $\mathbf{r_c}$ to obtain the expression for the desired attitude as

$$\begin{aligned} \phi_d &= g^{-1}(\ddot{x}_c s\psi_d - \ddot{y}_c c\psi_d) \\ \theta_d &= g^{-1}(\ddot{x}_c c\psi_d + \ddot{y}_c s\psi_d) \end{aligned} \tag{4.13}$$

or in matrix form

$$\mathbf{\Theta_d} = g^{-1}\mathbf{A}_\psi \mathbf{r_{c,h}} \tag{4.14}$$

where $\mathbf{r_{c,h}} \in^2$ is the horizontal commanded acceleration.

To complete the position controller as in figure 25, the only step left is calculating the desired thrust, the input of the motor dynamics block, which is obtained by rewriting the third equation of 4.6 and making $f = f_d$ and $\ddot{z} = \ddot{z}_d$. Then, one has

$$f_d = m(g + \ddot{z}_d) \tag{4.15}$$

which completes the position controller.

4.6.1.1 Direction Guaranteed Saturation Function

As discussed previously, the goal of the controller is to make the error between the desired quantity to the actual one goes exponentially to zero. However, it can happen of this error being too large on practice, which can cause problems. Therefore a threshold should be applied in order to avoid such issues. Before introducing the Direction Guaranteed Saturation Function as in [3], the simple saturation function will be analyzed. The saturation function $sat(x, a)$ with $x \in$ and $a \in_+$ is defined as

$$sat(x, a) = \begin{cases} x_k, & |x_k| \leq a \\ a \cdot sign(x_k), & |x_k| > a \end{cases} \tag{4.16}$$

In case $\mathbf{x} \in^n$, the saturation function becomes

$$sat(\mathbf{x}, a) = \begin{bmatrix} sat(x_1, a) \\ . \\ . \\ . \\ sat(x_n, a) \end{bmatrix} \tag{4.17}$$

For the first case where x is a scalar value, the simple saturation function works fine, controlling the error and making it assume a pre-determined value in case it becomes larger than the threshold. The limitations arise when the $x \in^n$ ($n \in_+$) where despite the error been keep limited, the direction of it can stroll from the desired one as shown in figure 26.

To avoid such issue, a *Direction-Guaranteed Saturation Function* is designed as

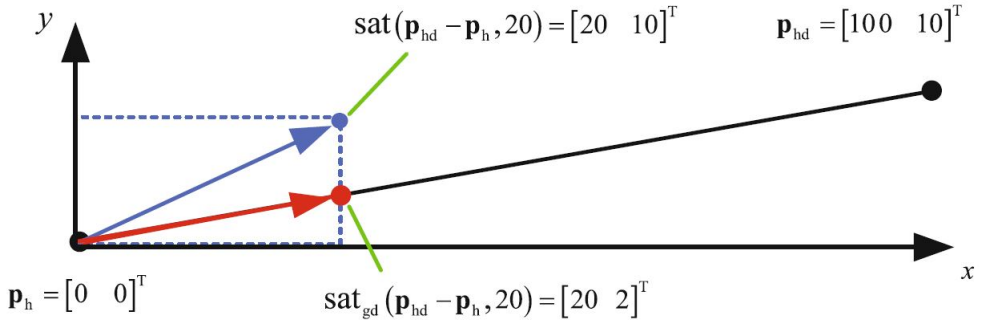
$$sat_{gd}(\mathbf{x}, a) = \kappa_a(\mathbf{x}) \cdot \mathbf{x} \tag{4.18}$$

where

$$\kappa_a(\mathbf{x}) = \begin{cases} 1, & ||\mathbf{x}||_\infty \leq a \\ \frac{a}{||\mathbf{x}||_\infty}, & ||\mathbf{x}||_\infty > a \end{cases} \tag{4.19}$$

and $\mathbf{x} \in^n$, $a \in_+$ and $||\mathbf{x}||_\infty = max(|x_1|, ..., |x_n|)$. As it can be seen in figure 26 besides keeping the error under a pre-determined value, the Direction Guaranteed Saturation Function has the advantage of maintaining the direction of the vector $\mathbf{x}$.

Figure 26: Simple saturation function and direction guaranteed saturation function



Source: [3]

Therefore, using the Direction Guaranteed Saturation Function the equations 4.13

and 4.15 are rewritten as

$$\begin{aligned} \mathbf{\Theta_d} &= sat_{gd}\left(g^{-1}\mathbf{A}_{\psi}\mathbf{r_{c,h}}, \theta_{max}\right) \\ f_d &= sat_{gd}\left(m(g+\ddot{z}_d) - \tfrac{f_{min}+f_{max}}{2}, \tfrac{f_{max}-f_{min}}{2}\right) + \tfrac{f_{min}+f_{max}}{2} \end{aligned} \tag{4.20}$$

where θ_{max} has been evaluated from the previous report, f_{max} is established by the motor manufacturer and f_{min} is considered as 10% of the weight of the quadrotor.

4.6.2 Attitude Controller

The aim of the attitude controller is evaluate the desired moments required for keeping with the trajectory. As in [3] the converge ratio of the inner loop should be 4-10 higher than the position controller in order to guarantee that the outputs of the position controller have been already achieved and by consequence the attitude controller will comply with its tasks.

According to reference [12] the desired moment can be evaluated by using PD control laws as

$$\tau_{\mathbf{d}} = \begin{bmatrix} k_{p,\phi}(\phi_d - \phi) + k_{d,\phi}(p_d - p) \\ k_{p,\theta}(\theta_d - \theta) + k_{d,\theta}(q_d - q) \\ k_{p,\psi}(\psi_d - \psi) + k_{d,\psi}(r_d - r) \end{bmatrix} \tag{4.21}$$

On practice saturation values need to be taken into consideration and the vector of desired moments is rewritten as

$$\tau_{\mathbf{d}} = sat_{gd}\left(\begin{bmatrix} k_{p,\phi}(\phi_d - \phi) + k_{d,\phi}(p_d - p) \\ k_{p,\theta}(\theta_d - \theta) + k_{d,\theta}(q_d - q) \\ k_{p,\psi}(\psi_d - \psi) + k_{d,\psi}(r_d - r) \end{bmatrix}, \tau_{\mathbf{max}}\right) \tag{4.22}$$

with $\tau_{\mathbf{max}} \in^{3x3}$. Recollecting the set of equations 4.4 the maximum moments for the axes x, y and z are given by

$$\begin{aligned} \tau_{max,x} = \tau_{max,y} &= l \cdot f_{max} \\ \tau_{max,z} &= \tfrac{2}{n}\tfrac{c_M}{c_T} f_{max} \end{aligned} \tag{4.23}$$

where $c_M = 2.925 \cdot 10^{-9}$ [Nm/rpm^2] and $c_T = 1.4865 \cdot 10^{-7}$ [N/rpm^2] as from [3].

4.6.3 Motor Dynamics

The motor dynamic is a really complicated function that depends on many factors as range of speed, acceleration or deceleration condition, etc. Moreover, the effective thrust and moments combine the dynamic of the propeller, which is compounded by really complicated relations as well. In order to simplify the analysis without losing generality, the dynamics of the motor is simplified by a first order function described by

$$\omega_i = \omega_{d,i}\left(1 - e^{\frac{t}{T_m}}\right) \tag{4.24}$$

where $\omega_{d,i}$ is the desired speed of the i-th motor and ω_i is its actual speed. The $\omega_{\mathbf{d}}$ is calculated by the inversion of the matrix

$$\begin{bmatrix} f_d \\ \tau_{\mathbf{d}} \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ \frac{l}{2}c_T & \frac{l}{2}c_T & -\frac{l}{2}c_T & -\frac{l}{2}c_T \\ -\frac{l}{2}c_T & \frac{l}{2}c_T & \frac{l}{2}c_T & -\frac{l}{2}c_T \\ c_M & -c_M & c_M & -c_M \end{bmatrix} \begin{bmatrix} \omega_{d,1}^2 \\ \omega_{d,2}^2 \\ \omega_{d,3}^2 \\ \omega_{d,4}^2 \end{bmatrix} \tag{4.25}$$

The actual thrust and moments generated by the motors are obtained by evaluating the equation 4.4 using the values of ω_i values obtained thorugh equation 4.24.

4.6.3.1 Adjust to PID Parameters

By inverting 4.25, one has

$$\begin{bmatrix} \omega_{d,1}^2 \\ \omega_{d,2}^2 \\ \omega_{d,3}^2 \\ \omega_{d,4}^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4c_T} & \frac{1}{2lc_T} & -\frac{1}{2lc_T} & \frac{1}{4c_M} \\ \frac{1}{4c_T} & \frac{1}{2lc_T} & \frac{1}{2lc_T} & -\frac{1}{4c_M} \\ \frac{1}{4c_T} & -\frac{1}{2lc_T} & \frac{1}{2lc_T} & \frac{1}{4c_M} \\ \frac{1}{4c_T} & -\frac{1}{2lc_T} & -\frac{1}{2lc_T} & -\frac{1}{4c_M} \end{bmatrix} \begin{bmatrix} f_d \\ \tau_{\mathbf{d}} \end{bmatrix} \tag{4.26}$$

that can be rewritten into

$$\begin{bmatrix} \omega_{d,1}^2 \\ \omega_{d,2}^2 \\ \omega_{d,3}^2 \\ \omega_{d,4}^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} \frac{f_d}{c_T} \\ \frac{\tau_x}{lc_T} \\ \frac{\tau_y}{lc_T} \\ \frac{\tau_z}{c_M} \end{bmatrix} \tag{4.27}$$

which finally leads to

$$\begin{bmatrix} \omega_{d,1}^2 \\ \omega_{d,2}^2 \\ \omega_{d,3}^2 \\ \omega_{d,4}^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} u_{d,1} \\ u_{d,2} \\ u_{d,3} \\ u_{d,4} \end{bmatrix} \tag{4.28}$$

or in the short form

$$\begin{bmatrix} \omega_{d,i}^2 \end{bmatrix} = \mathbf{M_{4x4}} \cdot \mathbf{u_d}. \tag{4.29}$$

This way the coefficients c_T and c_M that usually are unknown can be compensated by the PID parameters.

4.6.4 Rigid Body Dynamics

The actual values for position $\mathbf{r}$ and velocity $\dot{\mathbf{r}}$, and the angular position $\mathbf{\Theta}$ and angular velocity $\dot{\mathbf{\Theta}}$ can be obtained by the combination of sensors (as accelerometers, GPS's, ultrasounds, cameras, etc) installed on board of the quadrotor. For the sake of the simulation, such values will be obtained by means of the rigid body movement equations. Those equations are

$$\begin{aligned} \ddot{x} &= g(\phi s\psi + \theta c\psi) \\ \ddot{y} &= g(-\phi c\psi + \theta s\psi) \\ \ddot{z} &= \tfrac{f}{m} - g \end{aligned} \tag{4.30}$$

and

$$\begin{aligned} J_{xx}\ddot{\phi} &= \tau_x \\ J_{yy}\ddot{\theta} &= \tau_y \\ J_{zz}\ddot{\psi} &= \tau_z - \dot{\phi}\dot{\theta}(J_{yy} - J_{xx}) \end{aligned} \tag{4.31}$$

whose the values can be integrated in order to obtain the linear and angular velocity. Those can be integrated once more into position and attitude.

4.7 Non Dimensional Tunning

After obtaining the controller model, the only task left is to tunning the PID parameters, which can be a really complicated task. In order to simplify this approach, the non dimensional tunning presented on reference [12] will be used as a first step of the tunning. Starting by the attitude controller, the equation of motion for the rotation around the

axis x is used as an example. This equation is expressed by

$$J_{xx}\ddot{\phi} + k_{d,\phi}\dot{\phi} + k_{p,\phi}\phi = 0 \tag{4.32}$$

Since this a second order system, it will be compared to the general equation of motion given by

$$\ddot{\phi} + 2\xi\omega_n\dot{\phi} + \omega_n^2\phi = 0 \tag{4.33}$$

where ξ is the damping ratio and ω_n is the natural frequency. By comparing equations 4.32 and 4.33, the PD parameters can be obtained as

$$k_{p,phi} = J_{xx}\omega_n^2 \tag{4.34}$$

and

$$k_{d,phi} = 2J_{xx}\xi\omega_n \tag{4.35}$$

According to tests, $\xi \approx 1$ and the natural frequency $\omega_n \approx 9$ rad/s. A similar method for position controller with the damping ratio around 1 and a natural frequency equal to 4 rad/s will be adopted as well.

Although this is not a perfect method for tunning, it is useful as a first attempt and with the results obtained by this first simulation, the PID parameters can be optimized until reaching satisfactory values.

4.8 Simulations

In order to evaluate the effectiveness of the control model, a simulation on MATLAB (which the code is available on the appendix G) has been carried out. All the physical parameters used on the simulation are present on the table 25, while the PID parameters are shown on the table 26.

The quadrotor is expected to fly at a constant altitude describing a square shape trajectory. The desired trajectory $\mathbf{r_d}$ and $\psi_\mathbf{d}$ are presented on the table 27 and the initial conditions are

Table 25: Quadrotor parameters

m [kg]	1.3420	f_{max} [N]	68.9839
g [m/s^2]	9.81	f_{min} [N]	1.4554
J_{xx} [kg·m^2]	0.0034	$\tau_{max,x,y}$ [Nm]	12.6172
J_{yy} [kg·m^2]	0.0054	$\tau_{max,z}$ [Nm]	0.6787
J_{zz} [kg·m^2]	0.0050	θ_{max} [rad]	1.4299
l [mm]	182.9	T_m [s]	0.09

Source: Author

Table 26: PID parameters

$k_{p,x}$	16	$k_{d,x}$	8	$k_{i,x}$	0.01
$k_{p,y}$	16	$k_{d,y}$	8	$k_{i,y}$	0.01
$k_{p,z}$	64	$k_{d,z}$	160	$k_{i,z}$	0.4
$k_{p,\phi}$	0.273	$k_{d,\phi}$	0.0607		
$k_{p,\theta}$	0.435	$k_{d,\theta}$	0.0967		
$k_{p,\psi}$	0.405	$k_{d,\psi}$	0.0900		

Source: Author

$$\begin{aligned} \mathbf{r}_0 &= [0\ 0\ -10]^T && [m] \\ \dot{\mathbf{r}}_0 &= [0\ 0\ 0]^T && [m/s] \\ \mathbf{\Theta}_0 &= [0\ 0\ 0]^T && [rad] \\ \dot{\mathbf{\Theta}}_0 &= [0\ 0\ 0]^T && [rad/s] \\ \omega_0 &= [4000\ 4000\ 4000\ 4000]^T && [rpm] \end{aligned} \tag{4.36}$$

Table 27: Desired trajectory

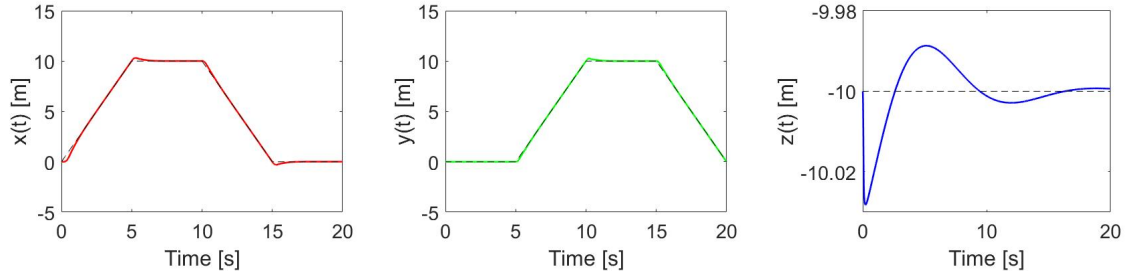
Δt [s]	x_d [m]	y_d [m]	z_d [m]	ψ_d [rad]
$[0, 5)$	2t	0	-10	0
$[5, 10)$	10	2(t-5)	-10	0
$[10, 15)$	10 - 2(t-10)	10	-10	0
$[15, 20)$	0	10 - 2(t-15)	-10	0

Source: Author

The results are presented on the figures 27, 28, 29 and 30 where the dashed lines represent the desired values and the continuous line, the actual ones. As it can be seen on the figures, the control method and the PID parameters adopted return satisfactory values. For the z position, for example, even though the values overshoot, the error is limited to less than 3 cm.

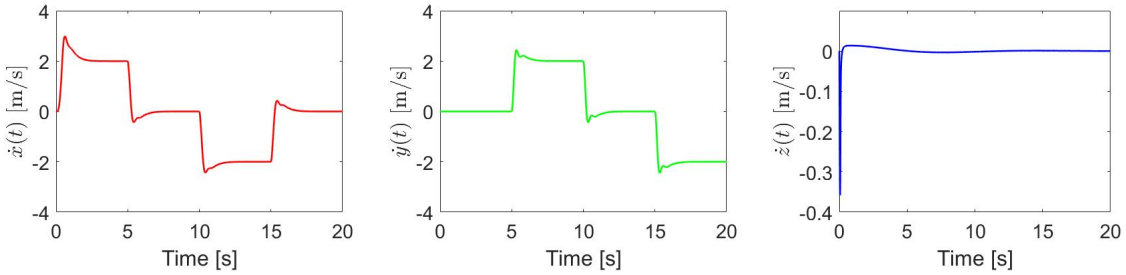
Lastly, the figure 31 shows the trajectory for the quadcopter. It can be visualized that the most problematic zones are the corners of the square, where the quadcopter needs to change direction abruptly. Those regions corresponds to the peaks on the graphics of the

Figure 27: Position



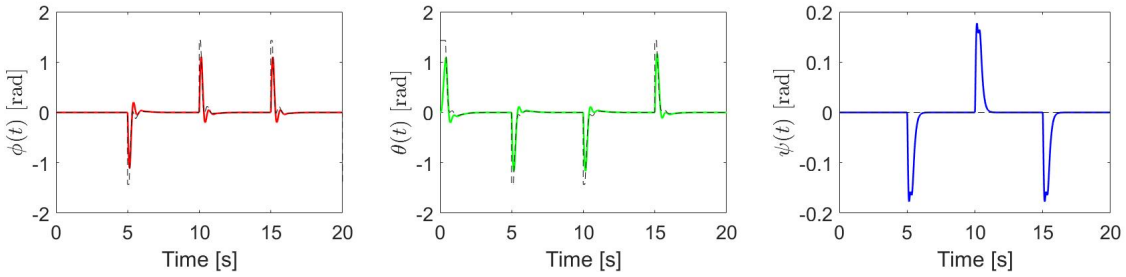
Source: Author

Figure 28: Velocity



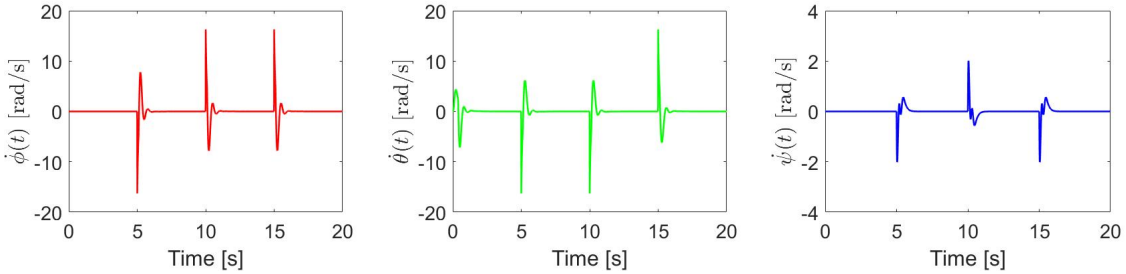
Source: Author

Figure 29: Angular position



Source: Author

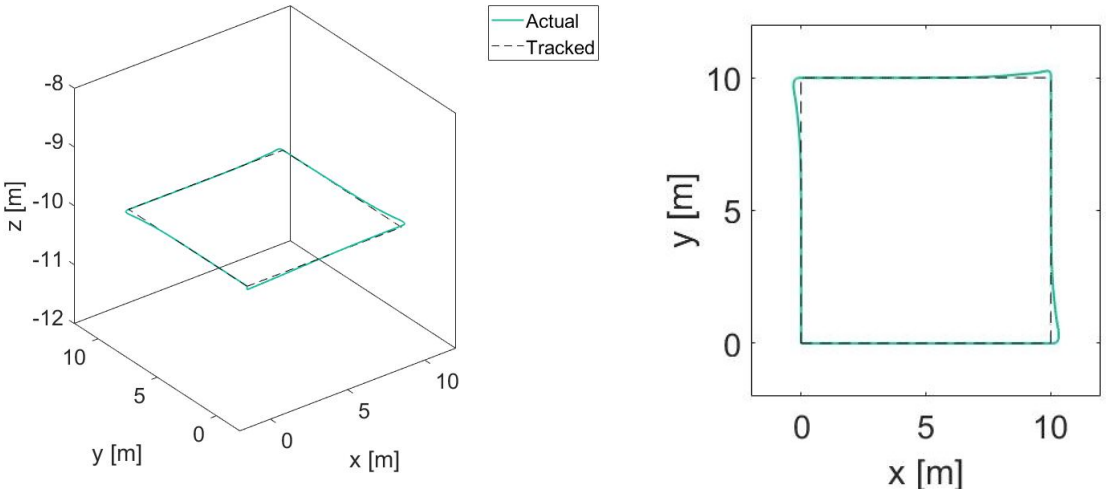
Figure 30: Angular velocity



Source: Author

figures 29 and 30, which is coherent with the practice, since the quadrotor changes its attitude in order to manouver.

Figure 31: Tracking



Source: Author

4.9 Summary

This chapter presented the control method for a quadrotor tracking a trajectory. The control was split into 4 blocks: *position controller*, *attitude controller*, *motor dynamics* and *rigid body dynamics*. Considerations about the saturation has been presented and the direction-guaranteed functions has been introduced. Lastly, the simulation results were presented.

The main equations for the control are here sumarized:

- **Position Controller**

$$\begin{aligned} \ddot{\mathbf{r}}_\mathbf{c} &= \ddot{\mathbf{r}}_\mathbf{d} + \mathbf{K_d}(\dot{\mathbf{r}}_\mathbf{d} - \dot{\mathbf{r}}) + \mathbf{K_p}(\mathbf{r_d} - \mathbf{r}) + \mathbf{K_i}\int(\mathbf{r_d} - \mathbf{r}) \\ \mathbf{\Theta_d} &= sat_{gd}\left(g^{-1}\mathbf{A}_\psi \mathbf{r_{c,h}}, \theta_{max}\right) \\ f_d &= sat_{gd}\left(m(g + \ddot{z}_d) - \tfrac{f_{min}+f_{max}}{2}, \tfrac{f_{max}-f_{min}}{2}\right) + \tfrac{f_{min}+f_{max}}{2} \end{aligned}$$

- **Attitude Controller**

$$\tau_\mathbf{d} = sat_{gd}\left(\begin{bmatrix} k_{p,\phi}(\phi_d - \phi) + k_{d,\phi}(p_d - p) \\ k_{p,\theta}(\theta_d - \theta) + k_{d,\theta}(q_d - q) \\ k_{p,\psi}(\psi_d - \psi) + k_{d,\psi}(r_d - r) \end{bmatrix}, \tau_\mathbf{max}\right)$$

- **Motor Dynamics**

$$\begin{aligned} \left[\omega_{d,i}^2\right] &= \mathbf{M_{4x4}} \cdot \mathbf{u_d} \\ \omega_i &= \omega_{d,i}\left(1 - e^{\frac{t}{T_m}}\right) \\ \mathbf{u} &= \mathbf{M_{4x4}^{-1}} \cdot [\omega_i^2] \end{aligned}$$

- **Rigid Body Dynamics**

$$\begin{aligned}
\ddot{x} &= g(\phi s\psi + \theta c\psi) \\
\ddot{y} &= g(-\phi c\psi + \theta s\psi) \\
\ddot{z} &= \tfrac{f}{m} - g \\
J_{xx}\ddot{\phi} &= \tau_x \\
J_{yy}\ddot{\theta} &= \tau_y \\
J_{zz}\ddot{\psi} &= \tau_z - \dot{\phi}\dot{\theta}(J_{yy} - J_{xx})
\end{aligned}$$

5 UAV DESIGNING AND ASSEMBLING

In this part, the design, the placement of the electronics and the assembly of the UAV will be presented. The chapter is divided in 4 parts: Design of the Aiframe, Manufacuring the Airframe, Onboard Eletronics and Assembling the UAV. The first one presents the design of the airframe from its first sketches on paper until the design on the AutoDesk Inventor software. The second one explains the manufacturing process of the airframe through the relatively new process of additive manufactuing by the FDM molding technology. After that, the onboard electronics, its main components and the placement of the electronics are shown. Lastly, the final assembly of the UAV is presented.

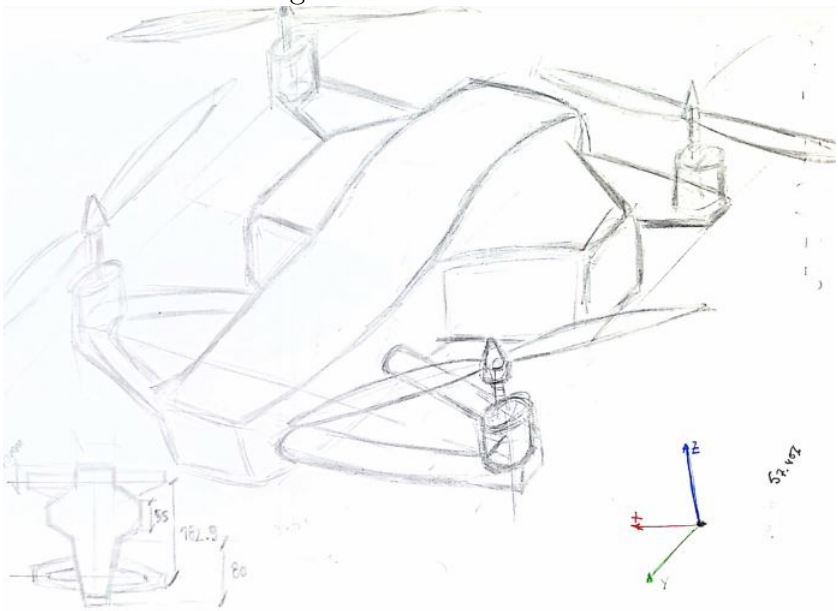
5.1 Design of the Aiframe

The initial design of the airframe is presented on the figure 35. As it can be seen on the figure, the design differs from the conventional commercial "drones" nowadays and was at the same time inspired in a F1 race vehicle (observe the supports of the frontal motors/propellers) and a fighter aircraft. It's important to mention that the airframe has been sized using as a main dimension the size of the propellers as presented on the section 3.1.3.

On figures 33 and 34 its presented the internal part of the CAD model of the airframe. As it can see by the figures the airframe has a shallow profile with several cavities on it, which contributes to decrease the total weight of the structure and provide a better cooling surface to avoid the overheating of the electronics and the battery. Also, it can be noticed some differences between the initial draft and the CAD model; those were implemented after the FEM analysis of the previous models.

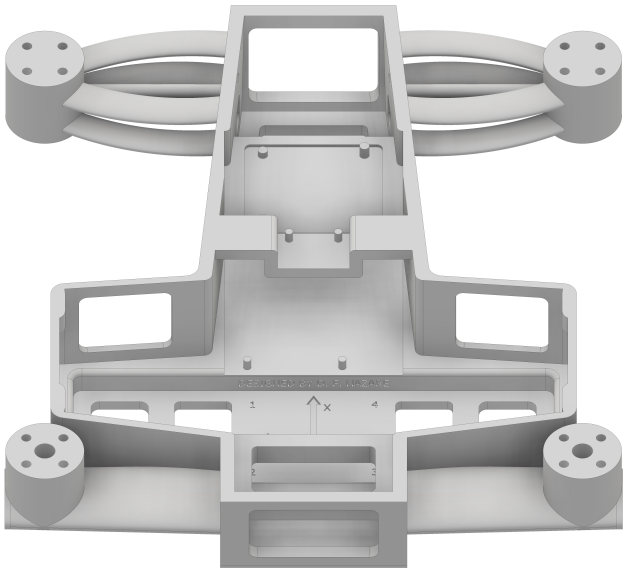
On 36 the details inscripted on the internal part of the airframe (base of the battery). Those inscriptions name the 4 motors of the UAV and shows the orientation of the mobile axis. The "RHO" indication stands for "Right-Hand Orientation" and places the z-axle (pointing outside of the paper).

Figure 32: Initial Draft



Source: Author

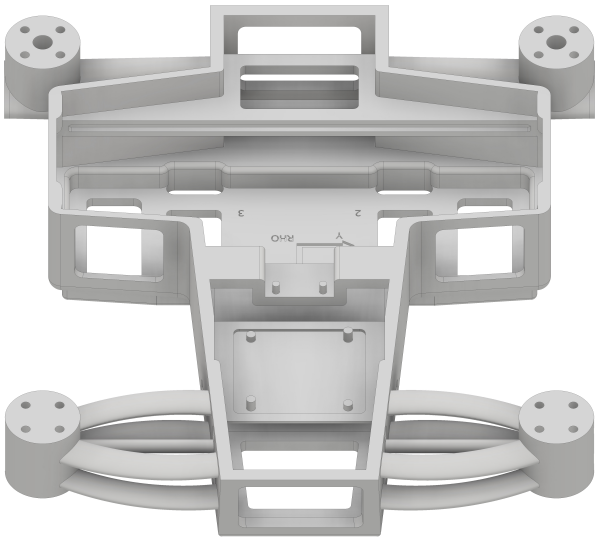
Figure 33: Autodesk Inventor Airframe Design Without Cover



Source: Author

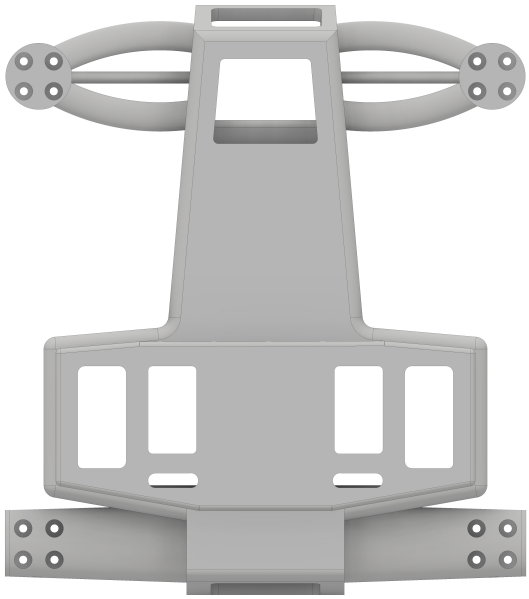
Lastly, figures 37, 38, 39, 40 and 41 shows the complete CAD model of the airframe with the cover included. The cover and base part of the airframe are assembled together with the aim of 8 M3 screws.

Figure 34: Autodesk Inventor Airframe Design Without Cover



Source: Author

Figure 35: Autodesk Inventor Airframe Design



Source: Author

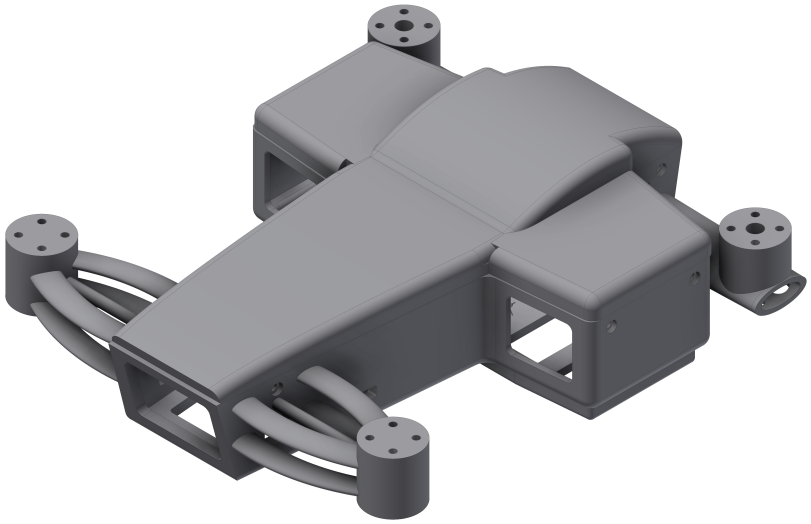
5.2 Manufacuring the Airframe

The prototype of the project has been manufactured making use of the additive manufacture method FDM (Fused Deposition Modelling). The 3D printer used was the Ender-3 from the Creality, which is one of the most suitable machines for beginners since is easy to learn compared to other 3D printers and is relatively cheap (it costs around USD 300). The characteristics of the printing are presented on the table 28.

Figure 36: Autodesk Inventor Airframe Design - Detail

Source: Author

Figure 37: Autodesk Inventor Airframe Design Complete



Source: Author

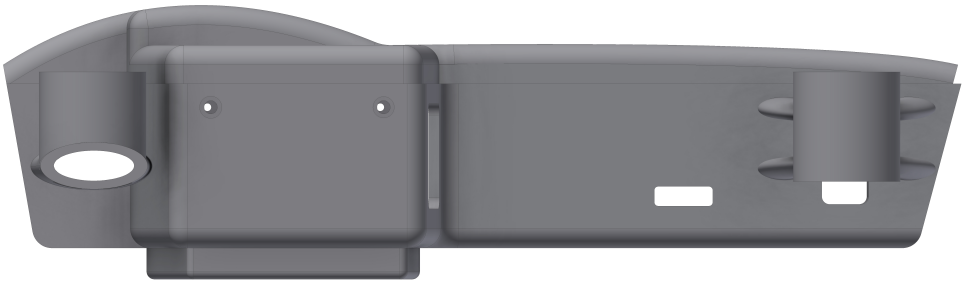
Table 28: 3D Printer Characteristics

Creality Ender-3	
Build Size	220 × 220 × 250 mm
Machine Size	440 × 440 × 465 mm
Molding Technology	FDM
Rated Power	270 W
Rated Voltage	AC 115/230 V 50/60 Hz
Rated Current	4/2.1 A
Net Weight	6.7 kg

Source: [14]

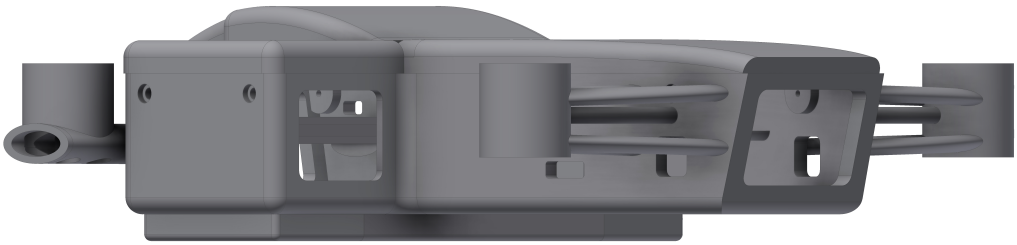
The Creality Under-3 comes with an SD-card compartment, where the printable file can be transferred from a computer. The process of creating the printable file from the

Figure 38: Autodesk Inventor Airframe Design Complete



Source: Author

Figure 39: Autodesk Inventor Airframe Design Complete

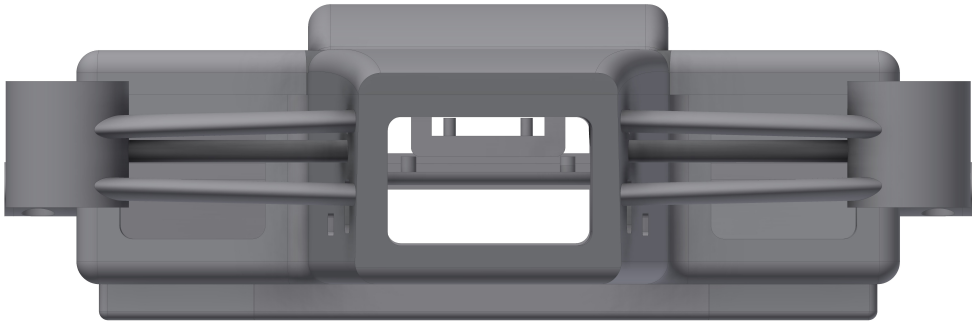


Source: Author

Autodesk Inventor is quite simple: the first thing is saving the CAD into a .stl file which can be read by a kinf of software called "slicer". This last converts the .stl file into a series of coordinates that the printer should move in order to build up the prototype. This series of coordinates is the well known file ".g" used on CNC machines for example. Besides the coordinates, those file carries the main characteristics of the printing. Those characteristics are presented on the table 29 below.

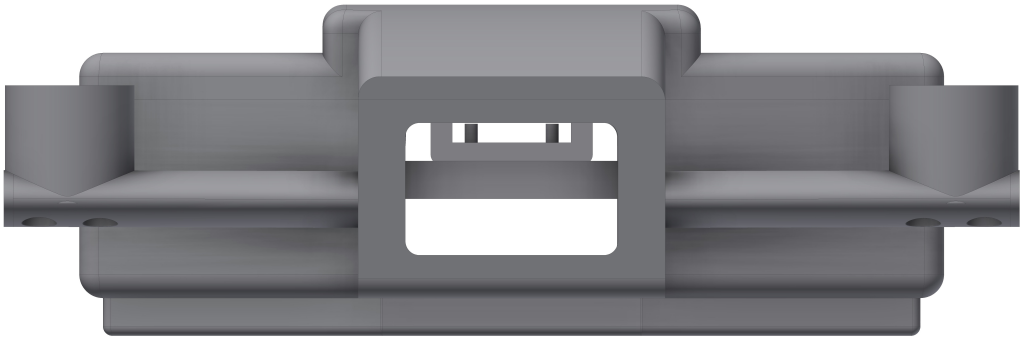
Mechanical Strength is not the strong suit of the FDM process. Said that, the PETG

Figure 40: Autodesk Inventor Airframe Design Complete



Source: Author

Figure 41: Autodesk Inventor Airframe Design Complete



Source: Author

material is the best choice when comes to prototyping due its lower prices if compared to other materials, its relatively high strenght, and its one of the materials that are the most easy to work with. It is worth mentioning that an important characteristic when working with FDM process, its the ability of the material to stick to the bed of the printer: if a material does not stick properly, there is a risk of detachment from the bed during the print process ruinnng the next layers of the printing; a material that usually has this problem is the PLA. From other side, there are some materials that stick so good to the

Table 29: Slicing Characteristics

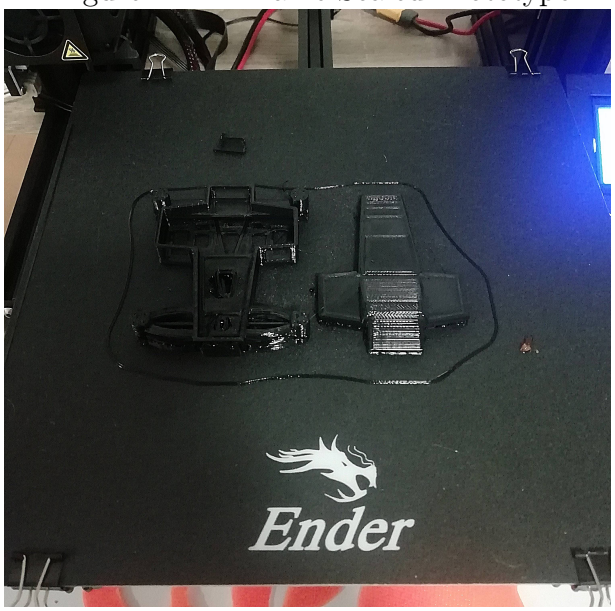
Ultimaker Cura	
Filament Material	PETG
Layer Height	0.12 mm
Infill Density	20 %
Infill Patern	Cubic
Printing Temperature	210 °C
Build Plate Temperature	45 °C
Print Speed	50.0 mm/s
Fan Speed	100 %
Support Placement	Everywhere
Support Overhang Angle	59 °C
Building Plate Adhesion Type	Brim

Source: Author

bed, that sometimes the bed is ruined after the printing, when taking out the printed part; and this might be the case of the PETG. Of course there are methods to circumvent those negative characteristics of those materials, however it is out of the scope of this work. On this thesis, the material used was the "BQ Easy Go PET-G" which its tecnhical sheet is present at the end of this text on the annex A.

Before starting the printing of the actual airframe, a smaller prototype in scale 1:4 has been printed in order to identify potential printing issues and parts where the design should be improved. This first printing took **2h 40 min** (room temperature: 23 ° C), and the result is showed on the picture 42 (without a scale).

Figure 42: Airframe Scaled Prototype



Source: Author

After the prototype, some configurations of the printing process was changed, as the

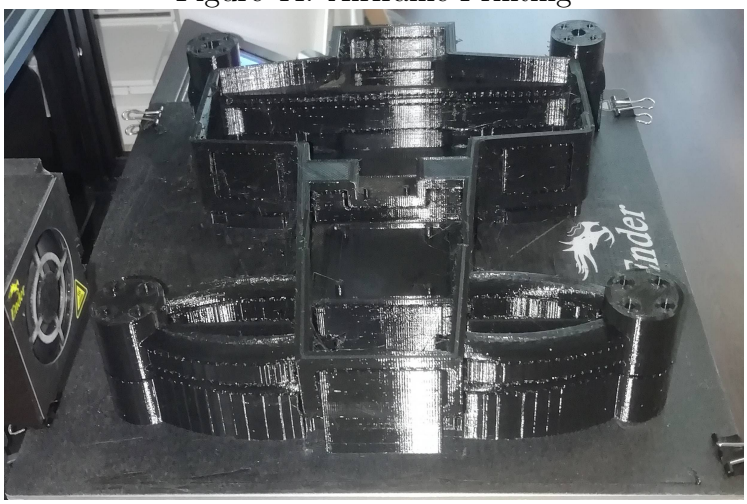
placement of the supports, since the central part of the print had no support at all and the filament was dropped ”in the air”.

The actual airframe has been printed in two parts: first, the cover has been printed in a proccess that took **1 day 6 hours 46 minutes and 7 seconds** (figure 43) while, the main part of the airframe took **3 days 5 hours 52 minutes and 55 seconds** (figure 44)

Figure 43: Airframe Cover Printing

Source: Author

Figure 44: Airframe Printing



Source: Author

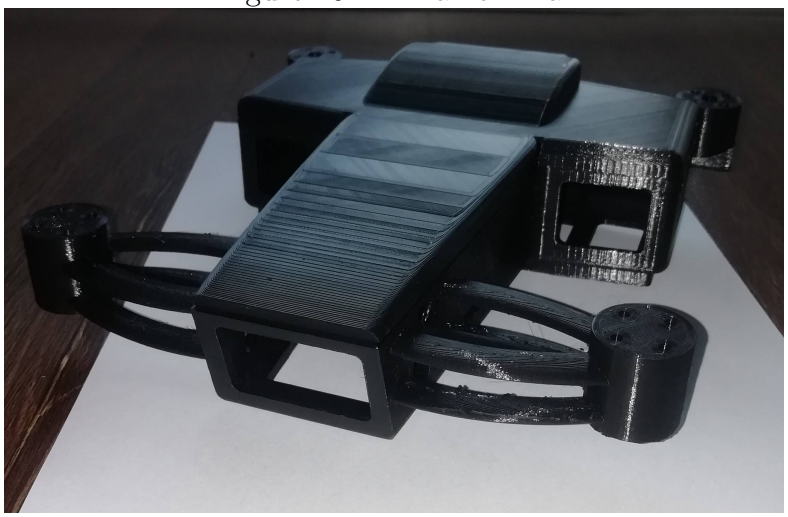
On the figures 45, 46, 47 and 48 the final printing after the removal of the supports are presented.

On the figure 47, the details previoulsy presented on the figure 37 is highlighted.

Figure 45: Airframe Final

Source: Author

Figure 46: Airframe Final

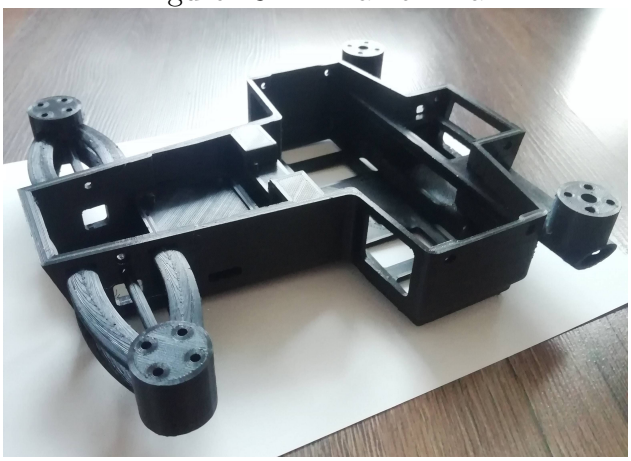


Source: Author

Figure 47: Airframe Final

Source: Author

Figure 48: Airframe Final



Source: Author

5.3 Onboard Electronicss

In this section the onboard electronics of the project will be briefly discussed. The elelectronics are compounded by: a **battery**, an **ESC**, a **microcontroller**, 4 **motors** and a **MEMS** consisting of a 3-axis Acceleromenter and a 3-axis Gyroscope. The main characteristics of the electronics elements are presented on the tables 30, 31, 32, 33 and 34. The motors, ESC and battery have been chosen through the method present on the section 3.3.

Table 30: Motor parameters

Motor	**Name**	XING CAMO X2306 2-6S FPV NextGen Motor
	Brand	iFlight
	K_{V0}	1700 RPM/V
	I_{mMAX}	37.2 A
	I_{m0}	1 A
	U_{m0}	10 V
	R_m	0.076 Ω
	m_m	0.0331 kg

Source: [15]

Table 31: ESC parameters

ESC	**Name**	F45A 6S 4IN1 V2
	Brand	T-Motor
	I_{eMAX}	45 A
	R_e	0.01 Ω
	m_e	0.0042 kg

Source:[16]

Table 32: Battery parameters

Battery	**Name**	nano-tech 4S 65-130C Lipo Pack w/XT-90
	Brand	Turnigy
	C_b	3850 mAh
	R_b	0.016 Ω
	U_b	14.8 V
	K_b	65 C
	m_b	0.454 kg

Source: [17]

The microcontroller used was an Arduino based one, due to its relatively easy to use softwares and hardwares, besides the possibility of consult plenty of online material. Moreover, the arduino counts with libraries whose make the interface with sensors and actuators easier. For this reason, the MPU6050 was choose as the main "sensor" of the project, since it is arduino compatible and the data provided by it is quite easy to gather.

Table 33: Microcontroller parameters

Microcontroller	**Name**	Arduino UNO R3
	Input Voltage	6 - 20 V
	Digital I/O Pins	14 (6 PWM)
	Analog Input Pins	6
	DC Current per I/O Pin	40 mA
	DC Current for 3.3V Pin	50 mA
	Flash Memory	32 KB (0.5 used by bootloader)
	SRAM	2 KB
	EEPROM	1 KB
	Clock Speed	16 MHz

Source: [18]

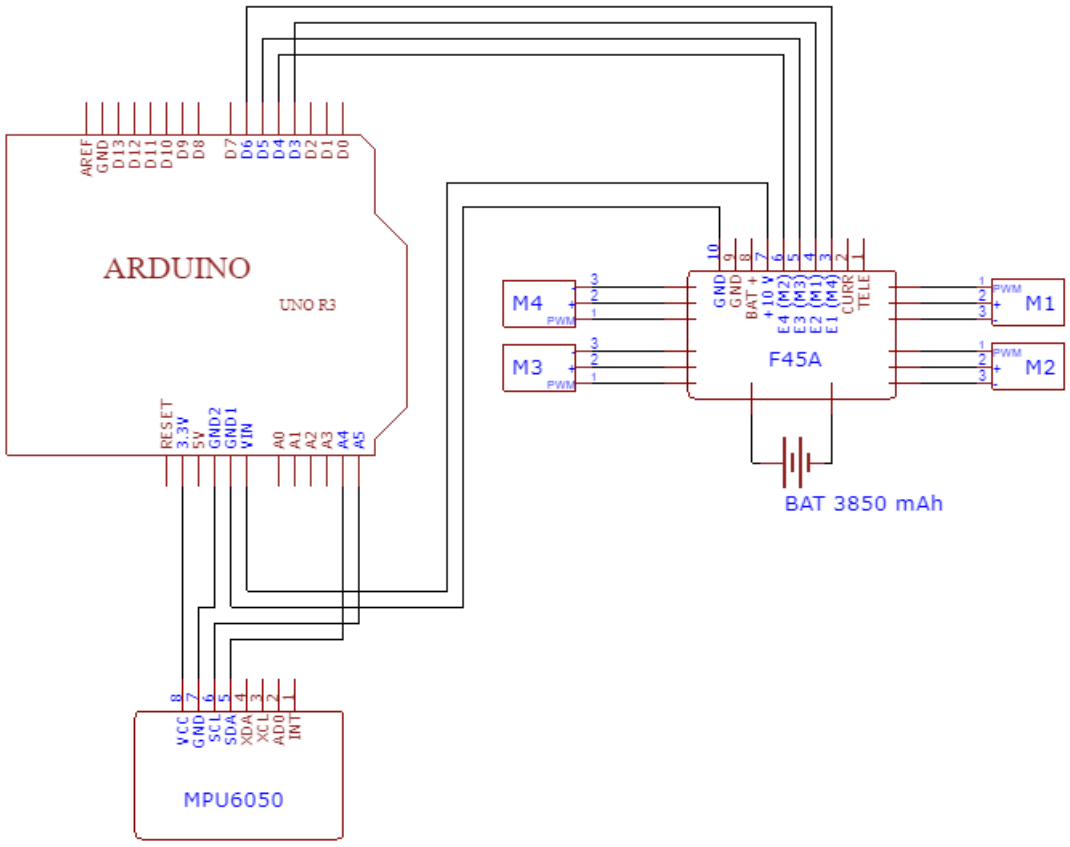
Table 34: Sensor parameters

Sensor	**Name**	SEN-MPU6050
	Brand	Joy-It
	Axes	3 acceleration and 3 gyroscope axes
	Degrees of Freedom	6 DOF
	Voltage Supply	3.3 - 5 V

Source: [19]

The figure 49 shows how the electronics are connected. All the pins used are highlited in blue. Also, the ESC has a Battery Elliminato Circuit (BEC), which means that the battery is connected to the ESC and this one powers up the microcontroller, without the need of an extra battery.

Figure 49: Arduino Schematics



Source: Created by the author on EasyEDA Online Tool

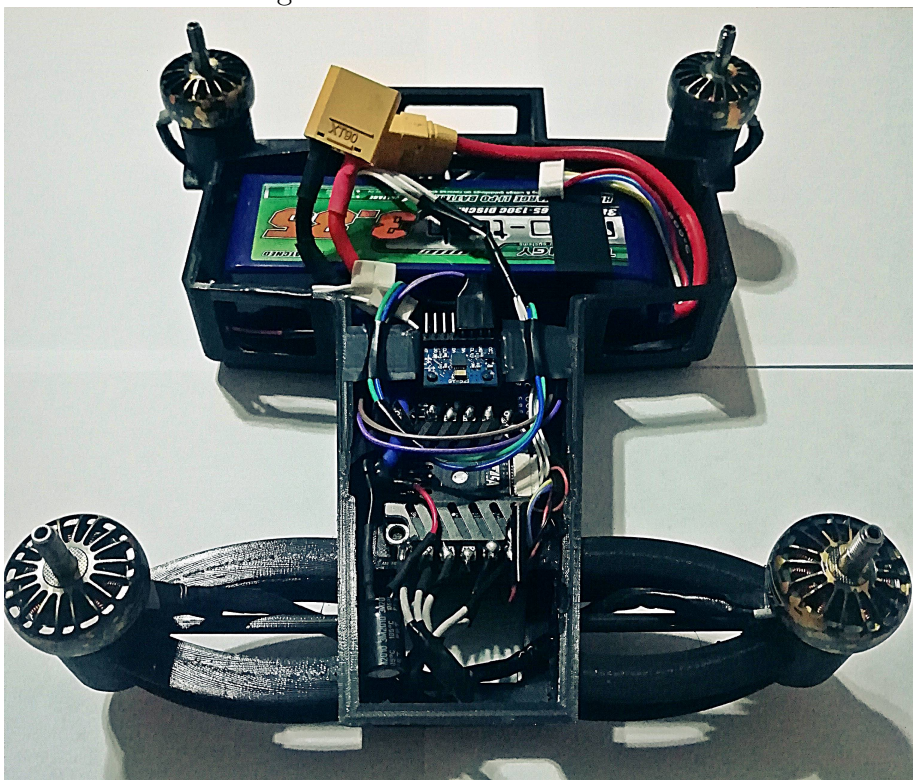
The base arduino code used in combination with the schematic circuit of the figure 49 is presented on the appendix H. As it will be seen on the next section, the codes lacks an effective filter for better results provided by the MPU6050, and the PID parameters must be improved in order to achieve a smooth stabilization and actuation of the motors.

The figure 50 shows how the electronics components have been assembled inside the UAV. As it can be seen on figure 33 the components are placed in two levels and in the figure 50 just the superior level is visible showing the ESC (in black) and the MPU6050 (blue, on the middle). The microcontroller is on the ground level and it cannot be seen on the figure.

5.4 Assembling the UAV

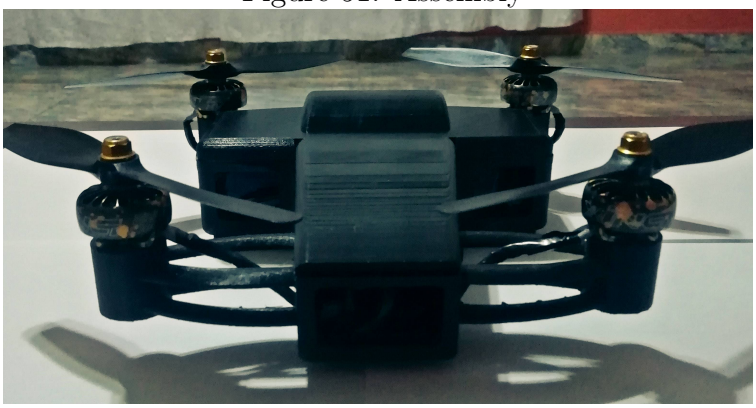
Finally, the UAV completed assembly is shown on figures 51, 52 and 53 below.

Figure 50: Onboard Electronics



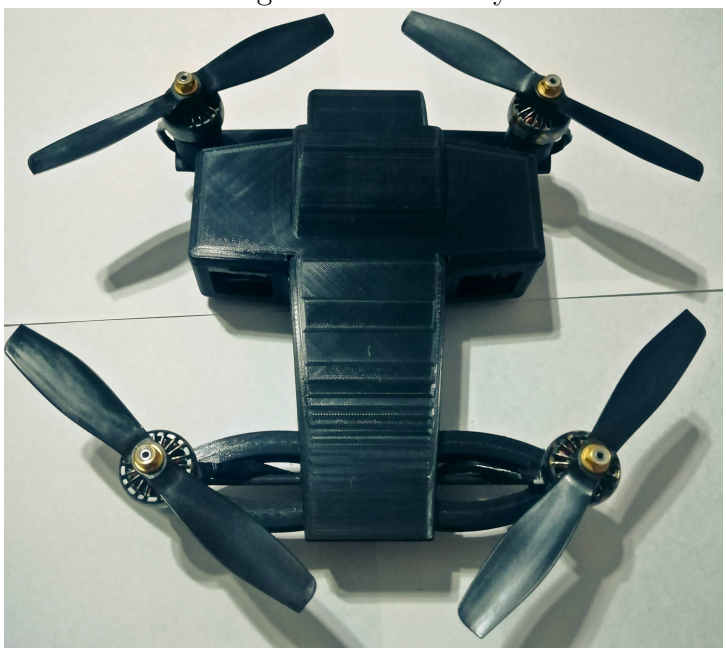
Source: Author

Figure 51: Assembly



Source: Author

Figure 52: Assembly



Source: Author

Figure 53: Assembly

Source: Author

6 CONCLUSION

This work has presented the development of a Unhamed-Aerial-Vehicle (UAV) to perform the task of delivering.

The first chapter has introduced the topic with a brief explanation of the working principles of UAVs of the type multicopter, following the presentation of the objectives of this thesis.

The second chapter aimed to present the dynamic equations of motion for a quadrotor. By introducing the Euler's angles representation, the rotation matrix to transform the vector on the Body space into the Earth (inertial) one is obtained. Moreover, the aerodynamic characteristics of the non-conservative forces and moments are presented and the concept of blade-flapping is briefly introduced. The equations obtained on that section were a necessary step toward the control of the UAV.

The chapter 3 aimed to define and study proper models for the components of the propulsion system as a first step towards the design process. After defining the environmental parameters, the models for propellers, motors, ESC's and battery were presented, and additionally, the mass of the airframe and its surface were defined as function of the propeller dimensions. After the modelling process, was defined the optimal parameters to be analyzed, where for each of the parameters, a set of components which maximize the particular parameter was determined and the results for each of those sets were presented for each configuration of multicopter (3-rotor, 4-rotor and 6-rotor). Such results showed that each set might be used for a particular application, but none of them are completely adequate for the application of delivering. Nonetheless, those results were important to define the maximum value of each of the parameter analyzed. The chapter also presented the criterion decision method, and with the aim of the maximum values defined previously, the set that best fit the delivering application has been determined in three configurations: tricopter, quadrotor and hexacopter. At the end of the chapter the set of components has been chosen and the parameters evaluated for this choice of component presented.

The fourth chapter presented the control method for a quadrotor tracking a trajectory. The control was split into 4 blocks: *position controller*, *attitude controller*, *motor dynamics* and *rigid body dynamics*. Considerations about the saturation has been presented and the direction-guaranteed functions has been introduced. Lastly, the simulation results were presented.

The chapter 5, the manufacture process of the airframe has been presented, from its first sketches on paper to the images of its CAD images. Following this first section the 3D printer and its characteristics were presented as the parameters set on the "slicer". Lastly the onboards electronics and its components are presented finishing the thesis with the assembling of the prototype.

7 SUGGESTIONS FOR FUTURE WORKS

As this work focuses from the choice of commercial components to the assembly of the UAV, in this section it will be provided some points to be improved and implemented on next works:

- **Batteries:** the flight endurance of the nowadays multicopters is intrinsically related to the batteries technology that is on use. A bottleneck to the use of drones with electric propulsion as an alternative to the delivery of goods is the few time they can fly without a charging landing. Although there have been a great development of the batteries on the past few years, it will still be a challenge to the industry to provide new battery models at a reasonable price, that are secure to use and with a green discard after its life. The reference [3] gives some examples of technological advances of the batteries and also some alternatives to the electric propulsion as combustion drones and drones with a power supply fixed on the ground and connected to the drone through an electric cable.

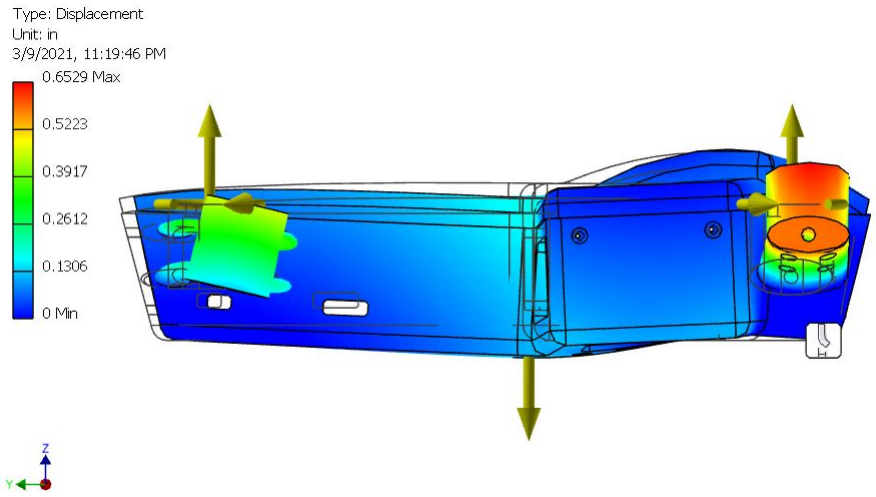
- **Control Model:** the *PID Control Method* presented on the section 4 although is a good start point to controlling the UAV under the conditions stated is not the most robust method. In fact, such a method has been used exclusively due to its didatical approach and wouldn't be suitable to perform the quick acceleration and roll/pitch of competition drones, for example. For proper material regarding the robust control methods, you could start by consulting the references [12], [20], [21] and [22]. For a general reference about control theory, please refer to [13].

- **Manufacturing and Materials:** please notice that the airframe manufactured by the 3d printing method presented is a **functional prototype only** and does not have the mechanical properties required by a true and proper flight, and therefore, the structure might not resist a fall, for example. The limitations of strength are due to the mechanical properties of the material itself and the characteristics of the manufacturing process. The additive manufacturing process has been chosen due

to its flexibility to make changes on the project due to the analysis provided by the printed part. However, as presented on the section 5.2 the complete printing takes around 4.5 days and therefore it would be impossible to manufacture in scale using such method. Optionally, instead of printing your own design you might want to see commercial airframes in carbon fiber, which have a really good mechanical strength. A good website to buy such parts is the *HobbyKing.com* [23] which besides airframes has pretty much everything else you would need to build an UAV.

- **Prototype Improvements:** although many modifications have been made on the prototype in comparison with the initial project, there are still some improvements that could be made. *The first modification* would be related with the base of the ESC. In the maximum throttle configuration the ESC heats up and such heat is not dissipated properly by the base. A possible solution for such problem would make holes on the base for increase the contact area with cold air which would therefore better dissipated the heat. *A second modification* is related to the battery compartment of the UAV. Such compartment can easily deformates as showed by the FEM figure 54 below which could compress the battery and cause a possible damage. Also, FEM studies could be further exploited for improvements of the prototype.

Figure 54: Battery compartment



Source: Author

- **Data filtering:** the measurements given by the MPU6050 (the 6 axis sensor utilized) cannot be used directly since it presents a lot of noise and innacuracies. Therefore, a method should be used in order to provide a better estimation of the measured variables. A useful method that can be used is the Kalman filter that

combines a series of measurements to produce a more precise result. Information regarding the Kalman filter can be consulted on the reference [3]. Another important point to be mentioned is that to have coherent results trough the MPU6050, this one should be properly calibrated. For a practical calibration method for such sensor please check [24].

- **Telemetry Data:** as it can be seen on the figure 49 on the F45A ESC there are two pins not in usage: the pin "CURR" and the pin "TELE". The "CURR" pin provides the current required by the motors, while the pin "TELE" provides informatios as the temperature of the ESC and the rotational speed of the motors. Those data could be stored and displayed in real time if the drone is wireless connected to a computer and it could be useful for improving the control method and predicting the battery status of charge.

- **Autonomous Flight:** to have the UAV fly autonomously, sensors that provide data from which it is possible to place the UAV in space (position sensors) and to perform obstacle avoidance are fundamental. One can think that the position can be integrated from the MPU6050 acceleration data and although this assumption makes sense in theory, it is showed impossible on the practice, since the noise present on the acceleration and therefore accumulated in two integration steps (from acceleration to velocity and to velocity to position) makes the data completely incorrect even if a Kalmann filter is used. Sensors that can be used for the determination of the position more accurately are: GPS, IMU, Cameras and RGB-D (those last two are used to obstacle avoidance as well); which one of those sensors presents its positive points and drawbacks. A reference for autonomous flights in indoor and outdoor environment is [20].

- **Flight Safety:** it cannot be highlighted enough how important is the safety during the flight. Therefore methods and procedures to a safer fly are always welcome. To increase the safety, several measures can be implemented as: redundant turn off bottoms in case of emergency; protection case for the propellers; real time temperature measurement of battery and critical components; and robust obstacle avoidance methods, to name a few. The reference [3] provides a introduction to the safety topic.

  Please always remove the propellers when testing the UAV on ground.
  Please read through the safety tips on the manual that comes with the battery.

- **Flight Tests:** flight tests should be performed and from the evaluation of those the control method on the section 4 could be validated and improved. As stated previoulsy such method is not adviced for competition UAVs and for those more robust methods should be implemented. Also, tests using payload should be performed in order to evaluate the maximum payload capacity of the UAV as it its endurance and those results should be confronted with the theoretical values presented on the table 3.6 for validation.

REFERENCES

1 Amazon.com. *Amazon Prime Air*. URL https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8node=8037720011, accessed: 13.06.2020. Disponível em: ⟨https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011⟩.

2 MANUEL, V.; RES, A.; ARAIZA, E. R. System Identification of a Quad-rotor in X Configuration from Experimental Data. *Reserch in Computing Science*, v. 118, p. 77–86, 2016. Disponível em: ⟨http://www.rcs.cic.ipn.mx/2016{_}118/SystemIdentificationofaQuad-rotorinXConfigurationfromExperimentalData.⟩

3 QUAN, Q. *Introduction to Multicopter Design and Control*. [S.l.]: Springer Nature, 2017. 1–384 p. ISBN 9789811033827.

4 Robert Mahony; Vijay Kumar; Peter Corke. Modeling, Estimation, and Control of Quadrotor. *IEEE*, n. August, p. 32, 2012.

5 Mustafa Cavcar. The International Standard Atmosphere (ISA). *Anadolu University*.

6 FoxTech. *T-MOTOR T9545*. URL https://www.foxtechfpv.com/t-motor-t9545-a-2pcs-pair.html, accessed: 11.03.2021. Disponível em: ⟨https://www.foxtechfpv.com/t-motor-t9545-a-2pcs-pair.html⟩.

7 T-Motor. *MN2212 KV920-V2.0*. URL https://store-en.tmotor.com/goods.php?id=389, accessed: 09.03.2021. Disponível em: ⟨https://store-en.tmotor.com/goods.php?id=389⟩.

8 T-Motor. *AIR 15A 3S*. URL https://store-en.tmotor.com/goods.php?id=366, accessed: 09.03.2021. Disponível em: ⟨https://store-en.tmotor.com/goods.php?id=366⟩.

9 Tattu. *Tattu 3700 Ah Battery*. URL https://www.genstattu.com/tattu-3700mah-45c-4s1p-lipo-battery-pack-with-xt60-plug.html, accessed: 09.03.2021. Disponível em: ⟨https://www.genstattu.com/tattu-3700mah-45c-4s1p-lipo-battery-pack-with-xt60-plug.html⟩.

10 JOHNSON, W. *Rotorcraft aeromechanics*. [S.l.: s.n.], 2006. v. 9781107028. 1–927 p. ISBN 9781139235655.

11 SHI, D. et al. A Practical Performance Evaluation Method for Electric Multicopters. v. 22, n. 3, p. 1337–1348, 2017.

12 MELLINGER, D.; KUMAR, V. Minimum snap trajectory generation and control for quadrotors. *Proceedings - IEEE International Conference on Robotics and Automation*, p. 2520–2525, 2011. ISSN 10504729.

13 OGATA, K. *Modern control engineering*. [S.l.: s.n.], 2017. 1–736 p. ISBN 9781482277449.

14 Creality. *Creality Ender-3*. URL https://www.creality.com/goods-detail/ender-3-3d-printer, accessed: 09.03.2021. Disponível em: ⟨https://www.creality.com/goods-detail/ender-3-3d-printer⟩.

15 iFlight. *XING CAMO X2306 2-6S FPV NextGen Mot*. URL https://shop.iflight-rc.com, accessed: 09.03.2021. Disponível em: ⟨https://shop.iflight-rc.com⟩.

16 T-Motor. *F45A 6S 4IN1 V2*. URL https://store-en.tmotor.com/goods.php?id=899, accessed: 09.03.2021. Disponível em: ⟨https://store-en.tmotor.com/goods.php?id=899⟩.

17 HobbyKing. *Turnigy nano-tech 3850mah 4S 65 130C Lipo Pack w/XT-90*. URL https://hobbyking.com, accessed: 09.03.2021. Disponível em: ⟨https://hobbyking.com⟩.

18 Arduino. *ARDUINO UNO REV3*. URL https://store.arduino.cc/arduino-uno-rev3, accessed: 09.03.2021. Disponível em: ⟨https://store.arduino.cc/arduino-uno-rev3⟩.

19 Joy-IT. *MOTION SENSOR MPU6050*. URL https://joy-it.net/en/products/SEN-MPU6050, accessed: 09.03.2021. Disponível em: ⟨https://joy-it.net/en/products/SEN-MPU6050⟩.

20 SHEN, S. *AUTONOMOUS NAVIGATION IN COMPLEX INDOOR AND OUTDOOR ENVIRONMENTS WITH MICRO AERIAL VEHICLES*. 187 p. Tese (Doutorado) — University of Pennsylvania, 2014.

21 SEBBANE, Y. B. *Smart autonomous aircraft: Flight control and planning for UAV*. [S.l.: s.n.], 2015. 1–414 p. ISBN 9781482299168.

22 GARCÍA, P. C.; Munoz Hernandez, L. E.; GIL, P. G. *Indoor Navigation Strategies for Aerial Autonomous Systems*. [S.l.: s.n.], 2016. 1–286 p. ISBN 9780128053393.

23 HobbyKing.com. *HobbyKing.com*. URL https://hobbyking.com, accessed: 09.03.2021. Disponível em: ⟨https://hobbyking.com/⟩.

24 TEDALDI, D.; PRETTO, A.; MENEGATTI, E. A Robust and Easy IMU Calibration. *Icra*, p. 3042–3049, 2014. Disponível em: ⟨http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6907⟩.

25 GetFPV. *https://www.getfpv.com*. URLhttps://www.getfpv.com/, accessed: 09.03.2021. Disponível em: ⟨https://www.getfpv.com/⟩.

26 EMax. *emaxmodel.com*. URL https://emaxmodel.com, accessed: 09.03.2021. Disponível em: ⟨https://emaxmodel.com⟩.

27 quadcopters.co.uk. *Drone Racing Specialists*. URL https://www.quadcopters.co.uk, accessed: 09.03.2021. Disponível em: ⟨https://www.quadcopters.co.uk⟩.

28 3DEE. *BQ PET-G filament*. URL https://3dee.hu/wp-content/uploads/bq-PETG-Datasheet-EN.pdf, accessed: 09.03.2021. Disponível em: ⟨https://3dee.hu/wp-content/uploads/bq-PETG-Datasheet-EN.pdf⟩.

APPENDIX A – LIST OF PROPELLERS

Source: Adapted from [23], [25], [26] and [27].

	Diameter D_P	Pitch H_P	Blade number B_P	Prop Mass m_P	Cost per unit $	C_D	C_T	C_M	Airframe Mass m_{af}	Surface Area S
Model	**m**	**m**	**blades**	**kg**	**USD**	**ND**	**ND**	**ND**	**kg**	**m²**
1	0.1295	0.1118	3	0.0012	0.8525	0.0920	0.3366	0.0464	0.1543	0.0810994
2	0.1295	0.1321	3	0.0012	0.8400	0.1206	0.3941	0.0608	0.1543	0.0810994
3	0.0650	0.1118	2	0.0004	0.5950	0.2839	0.4193	0.0636	0.0388	0.0204191
4	0.1270	0.1092	3	0.0041	1.0400	0.0916	0.3356	0.0461	0.1483	0.0779502
5	0.0406	0.0914	4	0.0012	1.3600	0.4292	1.0407	0.3844	0.0152	0.0079821
6	0.0310	0.0483	3	0.0003	0.9025	0.2420	0.5778	0.1219	0.0088	0.0046444
7	0.0135	0.1267	3	0.0052	0.8250	1.6849	1.5672	0.8489	0.0017	0.0008808
8	0.2540	0.1194	2	0.0041	2.0750	0.0386	0.1243	0.0087	0.5932	0.3118007
9	0.1778	0.0610	2	0.0041	2.2400	0.0277	0.0910	0.0062	0.2907	0.1527823
10	0.2388	0.1092	2	0.0041	1.9950	0.0374	0.1211	0.0084	0.5241	0.2755071
11	0.2388	0.1092	2	0.0041	2.4950	0.0374	0.1211	0.0084	0.5241	0.2755071
12	0.4064	0.1397	2	0.0041	3.1900	0.0277	0.0912	0.0062	1.5186	0.7982097
13	0.2286	0.1270	2	0.0041	1.5900	0.0479	0.1465	0.0107	0.4805	0.2525585
14	0.2794	0.1270	2	0.0041	2.0350	0.0371	0.1203	0.0083	0.7178	0.3772788
15	0.2540	0.1194	2	0.0041	3.6600	0.0386	0.1243	0.0087	0.5932	0.3118007
16	0.2540	0.2286	2	0.0041	1.7800	0.0985	0.2336	0.0221	0.5932	0.3118007
17	0.3048	0.1143	2	0.0041	1.1900	0.0301	0.0995	0.0067	0.8542	0.448993
18	0.1270	0.1016	3	0.0049	0.3725	0.0817	0.3131	0.0411	0.1483	0.0779502
19	0.1270	0.1143	2	0.0037	0.2550	0.0985	0.2336	0.0221	0.1483	0.0779502
20	0.1524	0.1143	2	0.0049	0.4975	0.0739	0.1962	0.0165	0.2135	0.1122482
21	0.1524	0.1016	3	0.0052	1.4975	0.0619	0.2626	0.0312	0.2135	0.1122482
22	0.1270	0.1143	3	0.0041	0.4475	0.0985	0.3504	0.0496	0.1483	0.0779502
23	0.2794	0.1524	2	0.0041	5.9900	0.0467	0.1439	0.0105	0.7178	0.3772788
24	0.3048	0.2032	2	0.0041	6.6900	0.0619	0.1751	0.0139	0.8542	0.448993
25	0.3302	0.1778	2	0.0041	7.4900	0.0459	0.1421	0.0103	1.0025	0.5269431

26	0.3810	0.2540	2	0.0041	9.9900	0.0619	0.1751	0.0139	1.3347	0.7015515
27	0.4572	0.2540	2	0.0041	12.9900	0.0479	0.1465	0.0107	1.9219	1.0102342
28	0.2794	0.1194	2	0.0041	4.2000	0.0346	0.1132	0.0077	0.7178	0.3772788
29	0.3810	0.1905	2	0.0041	3.0150	0.0417	0.1321	0.0093	1.3347	0.7015515
30	0.7625	0.2540	2	0.0720	339.9900	0.0270	0.0884	0.0060	5.3457	2.8098899
31	0.3558	0.1219	2	0.0105	49.9900	0.0277	0.0910	0.0062	1.1639	0.6118169
32	0.4066	0.1372	2	0.0130	62.9900	0.0273	0.0896	0.0061	1.5200	0.7989956
33	0.4572	0.1829	2	0.0320	89.9000	0.0322	0.1060	0.0072	1.9219	1.0102342
34	0.7620	0.2540	2	0.0930	279.9000	0.0270	0.0885	0.0060	5.3387	2.806206
35	0.5080	0.1524	2	0.0470	54.4500	0.0247	0.0797	0.0055	2.3727	1.2472027
36	0.4330	0.1651	2	0.0340	10.9950	0.0306	0.1011	0.0069	1.7238	0.9061194
37	0.2803	0.1067	2	0.0100	6.9950	0.0306	0.1009	0.0068	0.7224	0.3797133
38	0.0787	0.1016	3	0.0020	0.7475	0.1779	0.4894	0.0896	0.0570	0.029964
39	0.1295	0.1092	3	0.0038	0.7475	0.0887	0.3293	0.0447	0.1543	0.0810994
40	0.1295	0.1270	3	0.0050	0.7190	0.1131	0.3799	0.0570	0.1543	0.0810994
41	0.1295	0.1194	3	0.0044	0.7990	0.1023	0.3583	0.0515	0.1543	0.0810994
42	0.1295	0.1092	3	0.0041	0.7975	0.0887	0.3293	0.0447	0.1543	0.0810994
43	0.1295	0.1168	3	0.0050	0.7975	0.0988	0.3511	0.0498	0.1543	0.0810994

APPENDIX B – LIST OF MOTORS

Source: Adapted from [23], [25], [26] and [27].

Model	KV Value K_{V0} RPM/V	Max Cont Curr I_{mMAX} A	Nom Mo-Load Curr I_{m0} A	Nom No-Load Volt U_{m0} V	Resistance R_m OHM	Motor Mass m_m kg	Cost $ USD	Force cte K_E V/RPM	Torque cte K_T V/RPM	Max Test Angular Speed N_{MAX} RPM
1	4500	22.1	0.6	10	0.3	0.0067	9.99	0.000218	0.002084	45000
2	6000	22.9	0.6	7	0.19	0.0067	9.99	0.000164	0.0015657	42000
3	3700	25.3	0.5	10	0.265	0.0086	11	0.000267	0.0025469	37000
4	6000	31.6	1	7	0.12	0.0086	11	0.000164	0.0015644	42000
5	2800	35.0	0.7	10	0.18	0.0137	11	0.000353	0.0033677	28000
6	3300	43.5	0.9	10	0.148	0.0137	11	0.000299	0.0028554	33000
7	4100	57.7	1.1	10	0.102	0.0137	11	0.000241	0.0023031	41000
8	2400	121.5	1.5	10	0.046	0.0316	17	0.000414	0.0039517	24000
9	1200	16.0	0.9	10	0.15	0.231	13.7	0.000822	0.0078509	12000
10	2700	22.0	0.6	5	0.131	0.1216	18.2	0.000365	0.0034814	52194
11	3800	23.0	0.9	5	0.081	0.12	18.2	0.000259	0.0024765	40588
12	1750	33.8	1.15	10	0.0732	0.2952	18.5	0.000567	0.0054112	28590
13	1950	43.6	1.36	10	0.055	0.02936	18.5	0.000509	0.0048608	30450
14	240	40.0	0.9	18	0.085	0.136	99.9	0.004149	0.0396226	9532
15	300	40.0	1.1	15	0.063	0.136	99.9	0.003318	0.0316863	5603
16	360	40.0	1.4	15	0.045	0.14	99.9	0.002766	0.0264164	6732
17	1400	12.0	0.2	10	0.325	0.018	25.9	0.00071	0.0067771	14000
18	780	13.0	0.4	10	0.173	0.055	46.9	0.001273	0.0121589	7800
19	920	15.0	0.5	10	0.142	0.054	46.9	0.001079	0.0103067	7350
20	470	15.0	0.3	10	0.135	0.08	61.9	0.002119	0.0202369	8200
21	700	21.0	0.3	10	0.092	0.08	61.9	0.001425	0.0136052	8700
22	780	26.0	0.4	10	0.071	0.08	61.9	0.001278	0.0122088	7800
23	380	14.0	0.4	10	0.205	0.082	69.9	0.00261	0.0249255	7300
24	580	18.0	0.4	10	0.11	0.082	69.9	0.001717	0.0163931	6100
25	700	23.0	0.5	10	0.072	0.082	69.9	0.001423	0.0135937	7600
26	1750	39.0	1.2	10	0.065	0.026	28	0.000567	0.0054146	29822

27	1950	50.0	1.2	10	0.051	0.02584	28	0.00051	0.0048675	33180
28	2550	46.0	2	10	0.037	0.02592	28	0.000389	0.0037174	27408
29	340	35.0	0.9	10	0.093	0.145	99.9	0.002917	0.0278531	7497
30	8000	11.5	0.58	3	0.163	0.00424	12.1	0.000121	0.0011561	61503
31	11000	13.6	0.65	3	0.101	0.00408	12.1	8.89E-05	0.0008492	56815
32	300	40.0	0.6	22	0.125	0.0864	85	0.003322	0.0317248	5491
33	450	40.0	0.6	22	0.06	0.0848	85	0.002219	0.0211875	8168
34	170	40.0	0.4	22	0.27	0.1024	90	0.005853	0.0559007	6294
35	340	40.0	0.9	22	0.055	0.108	90	0.002935	0.028025	6482
36	1800	35.2	1.2	10	0.0815	0.0338	14.5	0.00055	0.0052537	43200
37	1700	37.2	1	10	0.0764	0.0331	22.6	0.000584	0.0055747	40800
38	2500	48.0	1.7	10	0.037	0.02824	23.1	0.000397	0.003796	28565
39	2700	53.0	2	10	0.0352	0.028	23.1	0.000368	0.0035121	29167
40	2450	44.6	2	10	0.0473	0.0331	13.9	0.000404	0.0038611	41160
41	4500	14.4	0.76	7	0.145	0.0085	15.8	0.000219	0.0020888	72000
42	6500	15.1	0.59	7	0.12	0.0085	15.8	0.000152	0.0014544	78000
43	4100	20.2	1.2	10	0.097	0.0165	15.2	0.000241	0.0023022	65600
44	3600	15.4	0.9	5.5	0.12	0.0163	15.2	0.000272	0.0026007	57600
45	2400	43.0	1.7	10	0.042	0.0264	26.7	0.000414	0.0039508	27861
46	2600	49.0	2	10	0.036	0.0268	26.7	0.000382	0.0036466	28578
47	3750	19.8	0.6	7.4	0.098	0.0152	15.3	0.000265	0.0025264	37500
48	3750	9.4	0.9	11.1	0.093	0.0158	15.8	0.000265	0.0025275	37500
49	2000	49.0	1.3	10	0.062	0.026	24.3	0.000496	0.0047365	32645
50	2700	56.0	2.3	10	0.035	0.02792	21.9	0.000367	0.0035086	29957

APPENDIX C – LIST OF ESC'S

Source: Adapted from [23], [25], [26] and [27].

Model	Max Curr I_{eMAX} A	Resistance R_e OHM	ESC Mass m_e kg	Cost $ USD
1	45	0.01	0.00425	14.96
2	30	0.01	0.0079	10.94
3	45	0.01	0.0045	16.335
4	25	0.01	0.0059	8.5
5	35	0.01	0.0083	14.36
6	40	0.01	0.002575	10.055
7	20	0.01	0.0039	7.95
8	40	0.01	0.012	18.28
9	30	0.01	0.006	12.41
10	40	0.01	0.0075	16.99
11	35	0.01	0.0075	17.05
12	20	0.01	0.007	4.99
13	12	0.01	0.007	4.99
14	6	0.01	0.007	3.99
15	18	0.01	0.024	4.99
16	30	0.01	0.006	9.24
17	20	0.01	0.006	9.2
18	20	0.01	0.006	9.12
19	12	0.01	0.011	8.52
20	30	0.01	0.028	9.24
21	25	0.01	0.008	9.2
22	20	0.01	0.028	9.12
23	12	0.01	0.008	8.52
24	12	0.01	0.003	9.99
25	35	0.01	0.0063	11.99
26	30	0.01	0.0039	12.99
27	20	0.01	0.004375	9.99
28	15	0.01	0.004375	3.99
29	12	0.01	0.0024	9.99
30	35	0.01	0.0063	11.99
31	30	0.01	0.004875	12.99
32	20	0.01	0.004375	9.99
33	15	0.01	0.004375	3.99
34	30	0.01	0.002575	11.123
35	40	0.01	0.026	39.99
36	20	0.01	0.026	19.99
37	60	0.01	0.06	59.99
38	80	0.01	0.062	69.99

APPENDIX D – LIST OF BATTERIES

Source: Adapted from [23], [25], [26] and [27].

Model	Capacity C_b mAh	Resistance R_b OHM	Total Voltage U_b V	Max Disch Rate K_b 1/h	Battery Mass m_b kg	Cost $ USD
1	450	0.016	4.35	160	0.0153	7.99
2	300	0.016	7.4	70	0.0153	10.99
3	260	0.016	4.35	30	0.007	5.445
4	300	0.016	4.35	60	0.0075	4.4165
5	300	0.016	3.8	60	0.0065	4.7795
6	250	0.016	4.35	60	0.0065	4.235
7	220	0.016	3.7	90	0.0055	5.3845
8	450	0.016	3.8	95	0.013	7.1995
9	500	0.016	3.7	190	0.014	7.1995
10	220	0.016	3.7	90	0.0055	4.3439
11	350	0.016	7.4	70	0.02	9.6195
12	3800	0.016	7.4	65	0.14	30.19
13	2500	0.016	7.4	65	0.106	20.57
14	300	0.016	7.6	75	0.017	10.648
15	450	0.016	7.4	75	0.028	9.7405
16	450	0.016	7.4	75	0.037	8.2159
17	550	0.016	7.4	95	0.0315	10.225
18	850	0.016	11.1	75	0.085	12.681
19	1300	0.016	11.1	75	0.129	13.25
20	300	0.016	11.4	60	0.024	8.8935
21	450	0.016	11.1	75	0.45	13.25
22	450	0.016	11.1	75	0.042	10.817
23	650	0.016	11.1	75	0.085	12.027
24	1550	0.016	11.1	100	0.14	20.449
25	550	0.016	11.1	95	0.046	14.46
26	650	0.016	11.1	95	0.06	15.67
27	850	0.016	11.1	95	0.103	18.695
28	1300	0.016	22.2	120	0.212	38.66
29	650	0.016	22.2	95	0.118	24.14
30	1400	0.016	22.2	120	0.23	40.475
31	1800	0.016	22.2	75	0.285	48.492
32	3700	0.016	14.8	45	0.285	47.412
33	2300	0.016	14.8	45	0.23	34.549
34	1800	0.016	14.8	45	0.23	65.308
35	1300	0.016	22.2	95	0.236	32.292
36	3000	0.016	11.1	30	0.253	24.84

APPENDIX E – SET OF PARAMETERS MATLAB CODE

Source: Author

```
% SET OF PARAMETERS
clear
clc
%INITIALIZATION----------------------------------------------------------
tic
fprintf('Initializing...\n');
general_data = readtable('set_of_parameters.xlsx','Sheet', ...
    'PARAMETERS','Range','E:E');
env_data = readtable('set_of_parameters.xlsx','Sheet', ...
    'PARAMETERS','Range','I:I');
rho = env_data{3,1}; %AIR DENSITY
g = env_data{5,1}; %GRAVITY ACCELERATION
num_of_prop = general_data{1,1};
BMCF = general_data{4,1};
Iother = general_data{5,1};
Cd1 = general_data{6,1};
Cd2 = general_data{7,1};

num_prp_data = 43; %max number of propellers being analyzed
num_mot_data = 50; %max number of motors being analyzed
num_esc_data = 38; %max number of esc's being analyzed
num_bat_data = 36; %max number of batteries being analyzed

%------------------------------------------------------------------------

%PROPELLER---------------------------------------------------------------
prp_data = readtable('set_of_parameters.xlsx', 'Sheet', 'PROPELLER', ...
    'Range','C2:L53', 'ReadVariableNames', false);

```

```
Dp_table = prp_data(3:end,1);
Hp_table = prp_data(3:end,2);
Bp_table = prp_data(3:end,3);
mp_table = prp_data(3:end,4);
cost_prp_table = prp_data(3:end,5);
CD_table = prp_data(3:end,6);
CT_table = prp_data(3:end,7);
CM_table = prp_data(3:end,8);
maf_table = prp_data(3:end,9);
S_table = prp_data(3:end,10);

Dp = zeros(num_prp_data,1);
Hp = zeros(num_prp_data,1);
Bp = zeros(num_prp_data,1);
mp = zeros(num_prp_data,1);
cost_prp = zeros(num_prp_data,1);
CD = zeros(num_prp_data,1);
CT = zeros(num_prp_data,1);
CM = zeros(num_prp_data,1);
maf = zeros(num_prp_data,1);
S = zeros(num_prp_data,1);

for r=1:num_prp_data
    Dp(r) = str2double(Dp_table{r,1});
    Hp(r) = str2double(Hp_table{r,1});
    Bp(r) = str2double(Bp_table{r,1});
    mp(r) = str2double(mp_table{r,1});
    cost_prp(r) = str2double(cost_prp_table{r,1});
    CD(r) = str2double(CD_table{r,1});
    CT(r) = str2double(CT_table{r,1});
    CM(r) = str2double(CM_table{r,1});
    maf(r) = str2double(maf_table{r,1});
    S(r) = str2double(S_table{r,1});
end
%-------------------------------------------------------------------------

%MOTOR-------------------------------------------------------------------
mot_data = readtable('set_of_parameters.xlsx', 'Sheet', 'MOTOR', ...
    'Range', 'C2:L53', 'ReadVariableNames', false);

KV0_table = mot_data(3:end,1);
ImMAX_table = mot_data(3:end,2);
Im0_table = mot_data(3:end,3);
```

```
69  Um0_table = mot_data(3:end,4);
70  Rm_table = mot_data(3:end,5);
71  mm_table = mot_data(3:end,6);
72  cost_mot_table = mot_data(3:end,7);
73  KE_table = mot_data(3:end,8);
74  KT_table = mot_data(3:end,9);
75  NMAX_table = mot_data(3:end,10);
76
77  KV0 = zeros(num_mot_data,1);
78  ImMAX = zeros(num_mot_data,1);
79  Im0 = zeros(num_mot_data,1);
80  Um0 = zeros(num_mot_data,1);
81  Rm = zeros(num_mot_data,1);
82  mm = zeros(num_mot_data,1);
83  cost_mot = zeros(num_mot_data,1);
84  KE = zeros(num_mot_data,1);
85  KT = zeros(num_mot_data,1);
86  NMAX = zeros(num_mot_data,1);
87
88  for k=1:num_mot_data
89      KV0(k) = str2double(KV0_table{k,1});
90      ImMAX(k) = str2double(ImMAX_table{k,1});
91      Im0(k) = str2double(Im0_table{k,1});
92      Um0(k) = str2double(Um0_table{k,1});
93      Rm(k) = str2double(Rm_table{k,1});
94      mm(k) = str2double(mm_table{k,1});
95      cost_mot(k) = str2double(cost_mot_table{k,1});
96      KE(k) = str2double(KE_table{k,1});
97      KT(k) = str2double(KT_table{k,1});
98      NMAX(k) = str2double(NMAX_table{k,1});
99  end
100 %-------------------------------------------------------------------------
101
102 %ESC-------------------------------------------------------------------------
103 esc_data = readtable('set_of_parameters.xlsx', 'Sheet', 'ESC', ...
        'Range', 'C2:F53', 'ReadVariableNames', false);
104
105 IeMAX_table = esc_data(3:end,1);
106 Re_table = esc_data(3:end,2);
107 me_table = esc_data(3:end,3);
108 cost_esc_table = esc_data(3:end,4);
109
110 IeMAX = zeros(num_esc_data,1);
```

```
111 Re = zeros(num_esc_data,1);
112 me = zeros(num_esc_data,1);
113 cost_esc = zeros(num_esc_data,1);
114
115 for j=1:num_esc_data
116     IeMAX(j) = str2double(IeMAX_table{j,1});
117     Re(j) = str2double(Re_table{j,1});
118     me(j) = str2double(me_table{j,1});
119     cost_esc(j) = str2double(cost_esc_table{j,1});
120 end
121 %-------------------------------------------------------------------------
122
123 %BATTERY----------------------------------------------------------------
124 bat_data = readtable('set_of_parameters.xlsx', 'Sheet', 'BATTERY', ...
       'Range', 'C2:H53', 'ReadVariableNames', false);
125
126 Cb_table = bat_data(3:end,1);
127 Rb_table = bat_data(3:end,2);
128 Ub_table = bat_data(3:end,3);
129 Kb_table = bat_data(3:end,4);
130 mb_table = bat_data(3:end,5);
131 cost_bat_table = bat_data(3:end,6);
132
133 Cb = zeros(num_bat_data,1);
134 Rb = zeros(num_bat_data,1);
135 Ub = zeros(num_bat_data,1);
136 Kb = zeros(num_bat_data,1);
137 mb = zeros(num_bat_data,1);
138 cost_bat = zeros(num_bat_data,1);
139
140 for i=1:num_bat_data
141     Cb(i) = str2double(Cb_table{i,1});
142     Rb(i) = str2double(Rb_table{i,1});
143     Ub(i) = str2double(Ub_table{i,1});
144     Kb(i) = str2double(Kb_table{i,1});
145     mb(i) = str2double(mb_table{i,1});
146     cost_bat(i) = str2double(cost_bat_table{i,1});
147 end
148 %-------------------------------------------------------------------------
149
150 %PRP x MOT x ESC x BAT HOVER--------------------------------------------
151 n=1;
152 ntotal = num_prp_data*num_mot_data*num_esc_data*num_bat_data;
```

```
153
154 Ielogical = 0;
155 Iblogical = 0;
156 Imlogical = 0;
157 Nlogical = 0;
158
159 mass_array = zeros(ntotal,1);
160 cost_array = zeros(ntotal,1);
161
162 prpxmotxescxbat = zeros(ntotal,4);
163 N_hov = zeros(ntotal,1);
164 M_hov = zeros(ntotal,1);
165 Im_hov = zeros(ntotal,1);
166 Um_hov = zeros(ntotal,1);
167 Ue0_hov = zeros(ntotal,1);
168 sigma_hov = zeros(ntotal,1);
169 Ie_hov = zeros(ntotal,1);
170 Ib_hov = zeros(ntotal,1);
171 t_hov = zeros(ntotal,1);
172 t_hov_over_cost = zeros(ntotal,1);
173 Ue_hov = zeros(ntotal,1);
174
175 eff_nl = zeros(ntotal,1);
176 Um_nl = zeros(ntotal,1);
177 Im_nl = zeros(ntotal,1);
178 M_nl = zeros(ntotal,1);
179 N_nl = zeros(ntotal,1);
180 Ib_nl = zeros(ntotal,1);
181 Ue_nl = zeros(ntotal,1);
182 eff_nl_over_cost = zeros(ntotal,1);
183 T_nl = zeros(ntotal,1);
184 mload_nl = zeros(ntotal,1);
185 mload_nl_over_cost = zeros(ntotal,1);
186 pitch_nl = zeros(ntotal,1);
187
188 V_max = zeros(ntotal,1);
189 Ie_z = zeros(ntotal,1);
190 Ue_z = zeros(ntotal,1);
191 Ib_z = zeros(ntotal,1);
192 N_z = zeros(ntotal,1);
193 Z_max = zeros(ntotal,1);
194 Z_max_over_cost = zeros(ntotal,1);
195
```

```
clc
fprintf('Evaluating...\n');
for i = 1:num_bat_data
   for j = 1:num_esc_data
        for k = 1:num_mot_data
            for r = 1:num_prp_data
            mass_array(n) = maf(r) + 1.25*(mb(i) + num_of_prop*me(j) ...
                + num_of_prop*mm(k) + num_of_prop*mp(r)); %TOTAL MASS ...
                m(it contains a 25% factor on the mass of the ...
                propulsor to account for the wires, sensors, ...
                microcontroller, etc)
            cost_array(n) = num_of_prop*cost_prp(r)+ ...
                num_of_prop*cost_mot(k)+num_of_prop*cost_esc(j) + ...
                cost_bat(i);%cost

            prpxmotxescxbat(n,1) = i; %NUMBER OF THE BATTERY
            prpxmotxescxbat(n,2) = j; %NUMBER OF THE ESC
            prpxmotxescxbat(n,3) = k; %NUMBER OF THE MOTOR
            prpxmotxescxbat(n,4) = r; %NUMBER OF THE PROPELLER

            N_comp = ...
                60*sqrt(g*mass_array(n)/(rho*num_of_prop*Dp(r)^4*CT(r)));
            if  N_comp > NMAX(k) %ANGULAR SPEED CHECK
                Nlogical = 1;
            else
                N_hov(n) = N_comp; %ANGULAR SPEED N (1)
                M_hov(n) = rho*Dp(r)^5*CM(r)*(N_hov(n)/60)^2; %TORQUE ...
                   M (2)
                Im_hov(n) = M_hov(n)/KT(k) + Im0(k); %Im (3)
                Um_hov(n) = KE(k)*N_hov(n) + Rm(k)*Im_hov(n); %Um (4)
                Ue0_hov(n) = Um_hov(n) + Im_hov(n)*Re(j); %Ue0 (5)
                sigma_hov(n) = Ue0_hov(n)/Ub(i); %sigma (6)
            end

            if IeMAX(j) < 1.2*ImMAX(k) %MOTOR CURRENT CHECK
                Imlogical = 1;
            end

            if sigma_hov(n)*Im_hov(n) > IeMAX(j) %ESC CURRENT CHECK
                Ielogical = 1;
            else
                Ie_hov(n) = sigma_hov(n)*Im_hov(n); %Ie (7)
            end
```

```
231
232             if num_of_prop*Ie_hov(n) + Iother > Cb(i)*Kb(i)/1000 %THE ...
                    CURRENT REQUIRED BY THE ESC'S OUTSTAND THE MAX ...
                    AVAILABLE FROM THE BATTERY
233                 Iblogical = 1;
234             else
235                 Ib_hov(n) = num_of_prop*Ie_hov(n)+Iother; %Ib (8)
236             end
237
238             Ue_hov(n) = Ub(i) - Ib_hov(n)*Rb(i); %Ue (9)
239             t_hov(n) = 0.06*(1-BMCF)*Cb(i)/Ib_hov(n); %Thover (10)
240             t_hov_over_cost(n) = t_hov(n)/cost_array(n); %Thover over ...
                    cost (11)
241
242             %NON LINEAR SYSTEM SOLVER 1---------------------------------
243             fun = @(x) syseff(x, Ub(i), Rb(i), Re(j), Dp(r), CM(r), ...
                    KV0(k), Um0(k), Rm(k), Im0(k), rho, num_of_prop, Iother);
244             x0 = [10, 5, 20, 5000, 10, 10]; % 6 OUTPUTS
245             options = optimset('Display','off');
246             sol = fsolve(fun, x0, options);
247             Um_nl(n) = sol(1); %(12)
248
249             if sol(2) > IeMAX(j) %ESC CURRENT CHECK
250                 Ielogical = 1;
251             else
252                 Im_nl(n) = sol(2); % (13)
253             end
254
255             if sol(4) > NMAX(k) %ANGULAR SPEED CHECK
256                 Nlogical = 1;
257             else
258                 M_nl(n) = sol(3); %(14)
259                 N_nl(n) = sol(4); %(15)
260             end
261
262             if sol(5) > Cb(i)*Kb(i)/1000 %BATTERY CURRENT CHECK
263                 Iblogical = 1;
264             else
265                 Ib_nl(n) = sol(5); %Ib (16)
266             end
267
268             if Ielogical || Iblogical || Imlogical || Nlogical
269                 %SKIP
```

```
270             else
271                 Ue_nl(n) = sol(6); %(17)
272                 over_eff_nl = 2*pi()/60*num_of_prop*M_nl(n)*N_nl(n);
273                 under_eff_nl = Ub(i)*Ib_nl(n);
274                 eff_nl(n) = over_eff_nl/under_eff_nl*100; %SYSTEM ...
                        EFFICIENCY IN PERCENTAGE (18)
275                 eff_nl_over_cost(n) = eff_nl(n)/cost_array(n); ...
                        %SYSTEM EFFICIENCY IN PERCENTAGE/TOTAL COST (19)
276                 T_nl(n) = rho*Dp(r)^4*CT(r)*(N_nl(n)/60)^2;%THRUST (20)
277                 if num_of_prop*T_nl(n)/g < mass_array(n)
278                     %SKYP
279                 else
280                     mload_nl(n) = num_of_prop*T_nl(n)/g - ...
                            mass_array(n);% (21) PAYLOAD MASS(it contains ...
                            a 25% factor on the mass of the propulsor to ...
                            account for the wires, sensors, ...
                            microcontroller, etc)
281                     mload_nl_over_cost(n) = ...
                            mload_nl(n)/cost_array(n); %PAYLOAD OVER COST ...
                            (22)
282                     pitch_nl(n) = ...
                            acos(g*mass_array(n)/num_of_prop/T_nl(n));%MAX ...
                            PITCH ANGLE (23)
283                 end
284             end
285
286             %FORWARD FLIGHT SPEED-------------------------------------
287             over_V_max = sqrt(2*g*mass_array(n)*tan(pitch_nl(n));
288         CDRAG_V_max = ...
                Cd1*(1-(sin(pitch_nl(n)))^3)+Cd2*(1-(cos(pitch_nl(n)))^3));
289             under_V_max = rho*S(r)*CDRAG_V_max;
290             V_max(n) = over_V_max/under_V_max; % (24) FORWARD FLIGHT ...
                    SPEED
291
292
293             %FLIGHT DISTANCE------------------------------------------
294             [Ie_z(n),Ue_z(n), Ib_z(n), N_z(n), Z_max(n)] = dist(g, ...
                    mass_array(n), pitch_nl(n), rho, S(r), Cd1, Cd2, ...
                    CT(r), Dp(r), num_of_prop, CM(r), KT(k), Im0(k), ...
                    KE(k), Rm(k), Ub(i), Iother, BMCF, Cb(i), Re(j)); % ...
                    FLIGHT DISTANCE
295
296             if N_z(n) > NMAX(k) %ANGULAR SPEED CHECK
```

```
297                 Nlogical = 1;
298             end
299
300             if Ie_z(n) > IeMAX(j) %ESC CURRENT CHECK
301                 Ielogical = 1;
302             end
303
304             if Ib_z(n) > Cb(i)*Kb(i)/1000 %THE CURRENT REQUIRED BY ...
                    THE ESC'S OUTSTAND THE MAX AVAILABLE FROM THE BATTERY
305                 Iblogical = 1;
306             end
307
308             if Ielogical || Iblogical || Imlogical || Nlogical
309                 cost_array(n) = Inf;
310                 mass_array(n) = Inf;
311
312                 N_hov(n) = 0;
313                 M_hov(n) = 0;
314                 Im_hov(n) = 0;
315                 Um_hov(n) = 0;
316                 Ue0_hov(n) = 0;
317                 sigma_hov(n) = 0;
318                 Ie_hov(n) = 0;
319                 Ib_hov(n) = 0;
320                 t_hov(n) = 0;
321                 t_hov_over_cost(n) = 0;
322                 Ue_hov(n) = 0;
323
324                 eff_nl(n) = 0;
325                 Um_nl(n) = 0;
326                 Im_nl(n) = 0;
327                 M_nl(n) = 0;
328                 N_nl(n) = 0;
329                 Ib_nl(n) = 0;
330                 Ue_nl(n) = 0;
331                 eff_nl_over_cost(n) = 0;
332                 T_nl(n) = 0;
333                 mload_nl(n) = 0;
334                 mload_nl_over_cost(n) = 0;
335                 pitch_nl(n) = 0;
336
337                 V_max(n) = 0;
338                 Ie_z(n) = 0;
```

```
339                 Ue_z(n) = 0;
340                 Ib_z(n) = 0;
341                 N_z(n) = 0;
342                 Z_max(n) = 0;
343 
344                 Ielogical = 0;
345                 Iblogical = 0;
346                 Imlogical = 0;
347                 Nlogical = 0;
348             else
349                 Z_max_over_cost(n) = Z_max(n)/cost_array(n);
350             end
351 
352             n=n+1;
353             end
354         end
355     end
356 clc
357 fprintf('Step %d/%d: %3.1f %% completed\n',i, num_bat_data, ...
        n/ntotal*100);
358 end
359 %-------------------------------------------------------------------------
360 
361 %RESULTS 1--------------------------------------------------------------
362 clc
363 w_counter = 1;
364 %MAXIMUM HOVER ENDURANCE-----------------------------------------------
365 w_counter = writing_function(t_hov, prpxmotxescxbat, 'D2', 'F2', ...
        'G2', 'H2', 'I2', 'T2', 'W2', 'Z2', 'AC2', Ie_hov, Ue_hov, ...
        Ib_hov, N_hov, w_counter);
366 
367 %MINIMUM COST----------------------------------------------------------
368 w_counter = writing_function(-cost_array, prpxmotxescxbat, 'D4', ...
        'F4', 'G4', 'H4', 'I4', 'T4', 'W4', 'Z4', 'AC4', Ie_hov, Ue_hov, ...
        Ib_hov, N_hov, w_counter);
369 
370 %MAXIMUM TIME OF ENDURANCE OVER COST-----------------------------------
371 w_counter = writing_function(t_hov_over_cost, prpxmotxescxbat, 'D6', ...
        'F6', 'G6', 'H6', 'I6', 'T6', 'W6', 'Z6', 'AC6', Ie_hov, Ue_hov, ...
        Ib_hov, N_hov, w_counter);
372 
373 %SYSTEM EFFICIENCY-----------------------------------------------------
374 w_counter = writing_function(eff_nl, prpxmotxescxbat, 'D8', 'F8', ...
```

```
    'G8', 'H8', 'I8', 'T8', 'W8', 'Z8', 'AC8', Im_nl, Ue_nl, Ib_nl, ...
    N_nl, w_counter);

%MAXIMUM SYSTEM EFFICIENCY OVER COST-----------------------------------
w_counter = writing_function(eff_nl_over_cost, prpxmotxescxbat, ...
    'D10', 'F10', 'G10', 'H10', 'I10', 'T10', 'W10', 'Z10', 'AC10', ...
    Im_nl, Ue_nl, Ib_nl, N_nl, w_counter);

%MAXIMUM PRODUCED PROPELLER THRUST-------------------------------------
w_counter = writing_function(T_nl, prpxmotxescxbat, 'D12', 'F12', ...
    'G12', 'H12', 'I12', 'T12', 'W12', 'Z12', 'AC12', Im_nl, Ue_nl, ...
    Ib_nl, N_nl, w_counter);

%MAXIMUM PAYLOAD MASS--------------------------------------------------
w_counter = writing_function(mload_nl, prpxmotxescxbat, 'D14', 'F14', ...
    'G14', 'H14', 'I14', 'T14', 'W14', 'Z14', 'AC14', Im_nl, Ue_nl, ...
    Ib_nl, N_nl, w_counter);

%MAXIMUM PAYLOAD OVER COST---------------------------------------------
w_counter = writing_function(mload_nl_over_cost, prpxmotxescxbat, ...
    'D16', 'F16', 'G16', 'H16', 'I16', 'T16', 'W16', 'Z16', 'AC16', ...
    Im_nl, Ue_nl, Ib_nl, N_nl, w_counter);

%MAXIMUM PITCH ANGLE---------------------------------------------------
w_counter = writing_function(pitch_nl, prpxmotxescxbat, 'D18', 'F18', ...
    'G18', 'H18', 'I18', 'T18', 'W18', 'Z18', 'AC18', Im_nl, Ue_nl, ...
    Ib_nl, N_nl, w_counter);

%MAXIMUM FORWARD FLIGHT SPEED------------------------------------------
w_counter = writing_function(V_max, prpxmotxescxbat, 'D20', 'F20', ...
    'G20', 'H20', 'I20', 'T20', 'W20', 'Z20', 'AC20', Ie_z, Ue_z, ...
    Ib_z, N_z, w_counter);

%MAXIMUM FLIGHT DISTANCE-----------------------------------------------
w_counter = writing_function(Z_max, prpxmotxescxbat, 'D22', 'F22', ...
    'G22', 'H22', 'I22', 'T22', 'W22', 'Z22', 'AC22', Ie_z, Ue_z, ...
    Ib_z, N_z, w_counter);

%MAXIMUM FLIGHT DISTANCE OVER COST-------------------------------------
w_counter = writing_function(Z_max_over_cost, prpxmotxescxbat, 'D24', ...
    'F24', 'G24', 'H24', 'I24', 'T24', 'W24', 'Z24', 'AC24', Ie_z, ...
    Ue_z, Ib_z, N_z, w_counter);

```

```
400 %MINIMUM MASS-------------------------------------------------------------
401 w_counter = writing_function(-mass_array, prpxmotxescxbat, 'D26', ...
        'F26', 'G26', 'H26', 'I26', 'T26', 'W26', 'Z26', 'AC26', Ie_hov, ...
        Ue_hov, Ib_hov, N_hov, w_counter);
402
403 %-----------------------------------------------------------------------
404 fprintf('Finishing...\n');
405 clc
406 toc
407
408 %FUNCTIONS--------------------------------------------------------------
409 %NON LINEAR SYSTEM SOLVER ----------------------------------------------
410 function F = syseff(x, Ub, Rb, Re, Dp, CM, KV0, Um0, Rm, Im0, rho, ...
        num_of_prop, Iother)
411     F(1) = -Ub + x(1) + x(2)*Re;
412     F(2) = x(3) - rho*Dp^5*CM/60/60*x(4)^2;
413     F(3) = x(1) - (x(3)*KV0*Um0/9.55/(Um0-Im0*Rm) + Im0)*Rm - ...
            (Um0-Im0*Rm)/KV0/Um0*x(4);
414     F(4) = x(2) - (x(3)*KV0*Um0/9.55/(Um0-Im0*Rm) + Im0);
415     F(5) = x(5) - num_of_prop*x(2) - Iother;
416     F(6) = x(6) - Ub + x(5)*Rb;
417 end
418
419 %MAXIMUM FLIGHT DISTANCE------------------------------------------------
420 function [Ie_z, Ue_z, Ib_z, N_z, Z_max] = dist(g, mass, teta_max, ...
        rho, S, Cd1, Cd2, CT, Dp, num_of_prop, CM, KT, Im0, KE, Rm, Ub, ...
        Iother, BMCF, Cb, Re)
421     teta = 0:0.01:real(teta_max);
422     over_V_max = sqrt(2*g*mass*tan(teta);
423     CDRAG = (Cd1*(1-sin(teta).^3)+Cd2*(1-cos(teta).^3)));
424     under_V_max = rho*S*CDRAG;
425     V_max = over_V_max./under_V_max; %FORWARD FLIGHT SPEED
426     N = 60*sqrt(g*mass/rho/CT/Dp^4/num_of_prop./cos(teta));%ANGULAR ...
            SPEED N
427     M = mass*CM*Dp/CT/num_of_prop./cos(teta);%TORQUE M
428     Im = M/KT + Im0; %Im
429     Um = KE*N + Rm*Im; %Um
430     sigma = (Um + Im*Re)/Ub; %sigma
431     Ie = sigma.*Im; %Ie
432     Ib = num_of_prop*Ie + Iother; %Ib
433     T_flight = 0.06*(1-BMCF)*Cb./Ib; %TIME OF FLIGHT
434     Z = 60*V_max.*T_flight; %FLIGHT DISTANCE
435     [Z_max, Z_max_index] = max(Z);
```

```
436      Ie_z = Ie(Z_max_index);
437      Ib_z = Ib(Z_max_index);
438      Ue_z = Ub - Ib_z*Re;
439      N_z = N(Z_max_index);
440  end
441
442  %RESULTS WRITING-----------------------------------------------------------
443  function w_counter = writing_function(array, prpxmotxescxbat, ...
         RangeString1, RangeString2, RangeString3, RangeString4, ...
         RangeString5, RangeString6, RangeString7, RangeString8, ...
         RangeString9, Ie_array, Ue_array, Ib_array, N_array, w_counter)
444      [opt, opt_index] = max(array(1:end,1));
445
446      PRP_model = prpxmotxescxbat(opt_index,4); %PROPELLER MODEL NUMBER
447      MOT_model = prpxmotxescxbat(opt_index,3); %MOTOR MODEL NUMBER
448      ESC_model = prpxmotxescxbat(opt_index,2); %ESC MODEL NUMBER
449      BAT_model = prpxmotxescxbat(opt_index,1); %BATTERY MODEL NUMBER
450
451      file = 'set_of_parameters.xlsx';
452      R = 'RESULTS';
453
454  writetable(array2table(abs(opt)),file,'Sheet',R,'Range',RangeString1);
455  writetable(array2table(PRP_model),file,'Sheet',R,'Range',RangeString2);
456  writetable(array2table(MOT_model),file,'Sheet',R,'Range',RangeString3);
457  writetable(array2table(ESC_model),file,'Sheet',R,'Range',RangeString4);
458  writetable(array2table(BAT_model),file,'Sheet',R,'Range',RangeString5);
459
460      Ie = Ie_array(opt_index);%ESC INPUT CURRENT
461      Ue = Ue_array(opt_index);%ESC INPUT VOLTAGE
462      Ib = Ib_array(opt_index);%BATTERY CURRENT
463      N = N_array(opt_index);%MOTOR SPEED N
464
465      writetable(array2table(Ie),file,'Sheet',R'Range',RangeString6);
466      writetable(array2table(Ue),file,'Sheet',R'Range',RangeString7);
467      writetable(array2table(Ib),file,'Sheet',R,'Range',RangeString8);
468      writetable(array2table(N),file,'Sheet',R,'Range',RangeString9);
469
470      fprintf('Writing... %d/13\n', w_counter);
471      w_counter = w_counter + 1;
472  end
473  %--------------------------------------------------------------------------
```

APPENDIX F – SET OF PARAMETERS MATLAB COMPLEMENTAR CODE

Source: Author

```
%GRAPHICS FOR THE SET OF PARAMETERS
%This code is to be used on the command window by using the results ...
    of the
%simulation "set_of_parameters_opt_5th_attempt"

figure(1)
%TIME OF ENDURANCE IN HOVER x COST
scatter(cost_array, t_hov,10,'filled');
set(gca, 'FontSize', 14)
title('Endurance Time x Cost');
xlabel('Cost [USD]');
ylabel('Endurance in Hover t_{e} [min]');

figure(2)
%SYSTEM EFFICIENCY x COST
scatter(cost_array, eff_nl,10,'filled');
set(gca, 'FontSize', 14)
title('Efficiency of System x Cost');
xlabel('Cost [USD]');
ylabel('System Efficiency \eta');

figure(3)
%PAYLOAD MASS x COST
scatter(cost_array, mload_nl,10,'filled');
set(gca, 'FontSize', 14)
```

```
25 title('Maximum Mass of Payload x Cost');
26 xlabel('Cost [USD]');
27 ylabel('Maximum Payload Mass m_{LOADMAX} [kg]');
28
29 figure(4)
30 %FLIGHT DISTANCE x COST
31 scatter(cost_array, Z_max,10,'filled');
32 set(gca, 'FontSize', 14)
33 title('Flight Distance x Cost');
34 xlabel('Cost [USD]');
35 ylabel('Flight Distance Z [m]');
36
37 figure(5)
38 %TIME OF ENDURANCE IN HOVER x PAYLOAD MASS
39 scatter(t_hov, mload_nl,10,'filled', 'k');
40 set(gca, 'FontSize', 14)
41 title('Maximum Mass of Payload x Endurance Time');
42 xlabel('Endurance in Hover t_{e} [min]');
43 ylabel('Maximum Payload Mass m_{LOADMAX} [kg]');
44
45 figure(6)
46 %FLIGHT DISTANCE x PAYLOAD MASS
47 scatter(Z_max, mload_nl,10,'filled' ,'k');
48 set(gca, 'FontSize', 14)
49 title('Maximum Mass of Payload x Flight Distance');
50 xlabel('Flight Distance Z [m]');
51 ylabel('Maximum Payload Mass m_{LOADMAX} [kg]');
52
53 %DECISION CRITERION
54 w_t = 4;
55 w_cost = 1;
56 w_eta = 1;
57 w_m = 4;
58 w_V = 1;
59 w_Z = 2;
60 W = w_t + w_cost + w_eta + w_m + w_V + w_Z;
61
62 tMAX = max(t_hov);
63 costMIN = min(cost_array);
64 etaMAX = max(eff_nl);
65 mMAX = max(mload_nl);
66 VMAX = max(V_max2);
67 ZMAX = max(Z_max);
```

```
68
69 p1 = w_t*t_hov/tMAX;
70 p2 = (w_cost*costMIN/cost_array)';
71 p3 = w_eta*eff_nl/etaMAX;
72 p4 = w_m*mload_nl/mMAX ;
73 p5 = w_V*V_max2/VMAX;
74 p6 =  w_Z*Z_max/ZMAX;
75
76 P = (p1+p2+p3+p4+p5+p6)/W;
77
78 [P_max, P_max_index] = max(P);
79
80 PRP_model = prpxmotxescxbat(P_max_index,4); %PROPELLER MODEL NUMBER
81 MOT_model = prpxmotxescxbat(P_max_index,3); %MOTOR MODEL NUMBER
82 ESC_model = prpxmotxescxbat(P_max_index,2); %ESC MODEL NUMBER
83 BAT_model = prpxmotxescxbat(P_max_index,1); %BATTERY MODEL NUMBER
84
85 fprintf('#########################TRICOPTER######################### \n');
86 fprintf('PRP: model %d\n', PRP_model);
87 fprintf('MOT: model %d\n', MOT_model);
88 fprintf('ESC: model %d\n', ESC_model);
89 fprintf('BAT: model %d\n\n', BAT_model );
90 fprintf('P_MAX = %3.4f\n\n', P_max);
91 fprintf('t_e: %3.1f min\n',t_hov(P_max_index));
92 fprintf('Efficiency: %3.2f %%\n ',eff_nl(P_max_index));
93 fprintf('T: %3.2f N\n', T_nl(P_max_index));
94 fprintf('m_{LOADMAX}: %2.3f kg\n', mload_nl(P_max_index));
95 fprintf('theta: %1.2f rad\n',pitch_nl(P_max_index));
96 fprintf('V_{MAX}: %2.2f m/s\n', V_max(P_max_index));
97 fprintf('Z_{MAX}: %4.2f m\n', Z_max(P_max_index));
98 fprintf('m: %1.3f kg\n', mass_array(P_max_index));
99 fprintf('cost: %4.2f USD\n', cost_array(P_max_index));
100
101 %PAYLOAD MASS FOR THE SET x TIME OF ENDURANCE OF THE SET
102 mass_payload = 0:0.01:mload_nl(P_max_index);
103 mass = mass_array(P_max_index) + mass_payload;
104 N = 60*sqrt(g*mass/(rho*num_of_prop*Dp(PRP_model)^4*CT(PRP_model))); ...
       %ANGULAR SPEED N (1)
105 M = rho*Dp(PRP_model)^5*CM(PRP_model)*(N/60).^2; %TORQUE M (2)
106 Im = M/KT(MOT_model) + Im0(MOT_model); %Im (3)
107 Um = KE(MOT_model)*N + Rm(MOT_model)*Im; %Um (4)
108 Ue0 = Um + Im*Re(ESC_model); %Ue0 (5)
109 sigma = Ue0/Ub(BAT_model); %sigma (6)
```

```
110 Ie = sigma.*Im; %Ie (7)
111 Ib = num_of_prop*Ie + Iother; %Ib (8)
112 Ue = Ub(BAT_model) - Ib*Rb(BAT_model); %Ue (9)
113 t_e = 0.06*(1-BMCF)*Cb(BAT_model)./Ib; %Thover (10)
114
115 hold on
116 figure(7)
117 plot(mass_payload, t_e, 'LineWidth', 2);
118 set(gca, 'FontSize', 12)
119 title('Time of Endurance x Payload Mass');
120 xlabel('Payload Mass m_{LOAD} [kg]');
121 ylabel('Time of Endurance t_e [min]');
122 legend('Trirotor','Quadrotor', 'Hexarotor');
123
124 hold on
125 figure(7)
126 plot(mass_payload, t_e,'LineWidth', 2);
127 set(gca, 'FontSize', 12)
128 xlabel('Payload Mass m_{LOAD} [kg]');
129 ylabel('Time of Endurance t_e [min]');
130 legend('No Cost Filter','Cost Filter');
131
132 %MULTIPLE BATTERIES
133 n_bat = 1:1:1000;
134 mass1 = maf(PRP_model) + 1.25*(n_bat*mb(BAT_model) + ...
        num_of_prop*me(ESC_model) + num_of_prop*mm(MOT_model) + ...
        num_of_prop*mp(PRP_model)); %TOTAL MASS m(it contains a 25% ...
        factor on the mass of the propulsor to account for the wires, ...
        sensors, microcontroller, etc)
135 mass = mass1 + mload_nl(P_max_index);
136 N = 60*sqrt(g*mass/(rho*num_of_prop*Dp(PRP_model)^4*CT(PRP_model))); ...
        %ANGULAR SPEED N (1)
137 M = rho*Dp(PRP_model)^5*CM(PRP_model)*(N/60).^2; %TORQUE M (2)
138 Im = M/KT(MOT_model) + Im0(MOT_model); %Im (3)
139 Um = KE(MOT_model)*N + Rm(MOT_model)*Im; %Um (4)
140 Ue0 = Um + Im*Re(ESC_model); %Ue0 (5)
141 sigma = Ue0/Ub(BAT_model); %sigma (6)
142 Ie = sigma.*Im; %Ie (7)
143 Ib = num_of_prop*Ie + Iother; %Ib (8)
144 Ue = Ub(BAT_model) - Ib*Rb(BAT_model); %Ue (9)
145 t_e = 0.06*(1-BMCF)*n_bat*Cb(BAT_model)./Ib; %Thover (10)
146
147 plot(n_bat,t_e);
```

APPENDIX G – CONTROL AND SIMULATION MATLAB CODE

Source: Author

```
%CONTROL-----------------------------------------------------------------
close all
clear
clc
n = 4;
m_af = 0.708;
m_p = 5/1000;
m_m = 36/1000;
m_e = 39/1000;
m_b = 478/1000;
m_c = 39/1000;

m = 0.8*m_af + n*(m_p + m_m) + 1.1*(m_e + m_b + m_c);
g = 9.81;
Ixx = 0.002 + 2*(m_m + m_p)*(.1293)^2;
Iyy = 0.004 + 2*(m_m + m_p)*(.1293)^2;
Izz = 0.005;

c_T = 1; %Corrected with PID parameters
c_M = 1; %Corrected with PID parameters
l = 182.9/1000;
f_max = n*1.758*g;
f_min = 0.1*m*g;
T_max = f_max*l;
tau_max = f_max*l;
tau_max_z = (2/n)*(2.925*10^-9/(1.4865*10^-7))*f_max;
w_max = sqrt(f_max/n/c_T);
```

```
theta_max = 81.93*pi/180;

%TUNNING OF CONTROL PARAMETERS----------------------------------------
xi_ang = 1;
omegan_ang = 9;

xi_lin = 1;
omegan_lin = 4;

kp_phi = Ixx*omegan_ang^2;
kp_theta = Iyy*omegan_ang^2;
kp_psi = Izz*omegan_ang^2;
kd_phi = 2*Ixx*xi_ang*omegan_ang;
kd_theta = 2*Iyy*xi_ang*omegan_ang;
kd_psi = 2*Izz*xi_ang*omegan_ang;
kd_x = 2*xi_lin*omegan_lin;
kd_y = 2*xi_lin*omegan_lin;
kd_z = 20*2*xi_lin*omegan_lin;
kp_x = omegan_lin^2;
kp_y = omegan_lin^2;
kp_z = 4*omegan_lin^2;
ki_x = 0.01;
ki_y = 0.01;
ki_z = 40*0.01;
T_m = 0.09;
Kd = [kd_x   0    0;
        0   kd_y  0;
        0    0   kd_z];
Kp = [kp_x   0    0;
        0   kp_y  0;
        0    0   kp_z];
Ki = [ki_x   0    0;
        0   ki_y  0;
        0    0   ki_z];
%TIME ARRAY DEFINITION-----------------------------------------------
t_step = 0.01;
t_init = 0;
t_end = 20;
t_array = t_init:t_step:t_end; %Time array

r0 = [0;0;-10];
dr0 = [0;0;0];
ddr0 = [0;0;0];
```

```
71 theta0 = 0;
72 phi0 = 0;
73 psi0 = 0;
74 dphi0 = 0;
75 dtheta0 = 0;
76 dpsi0 = 0;
77
78 w0 = [4000/60;4000/60;4000/60;4000/60];
79
80 %ALLOCATION-----------------------------------------------------------------
81 t = zeros(1,length(t_array)+1);
82
83 r = zeros(3,length(t_array)+1); %ACTUAL POSITION
84 dr = zeros(3,length(t_array)+1);
85 ddr = zeros(3,length(t_array)+1);
86
87 e_p = zeros(3,length(t_array)+1);
88 e_v = zeros(3,length(t_array)+1);
89 de_v = zeros(3,length(t_array)+1);
90 int_e_p = zeros(3,length(t_array)+1);
91
92 r_T = zeros(3,length(t_array)+1); %TRAJECTORY TO BE TRACKED
93 dr_T = zeros(3,length(t_array)+1);
94 ddr_T = zeros(3,length(t_array)+1);
95 dddr_T = zeros(3,length(t_array)+1);
96
97 ddr_c = zeros(3,length(t_array)+1); %COMMANDED ACCELERATION
98
99 phi_c = zeros(1,length(t_array)+1);
100 theta_c = zeros(1,length(t_array)+1);
101 psi_c = ones(1,length(t_array)+1)*psi0;
102 dphi_c = zeros(1,length(t_array)+1);
103 dtheta_c = zeros(1,length(t_array)+1);
104 dpsi_c = zeros(1,length(t_array)+1);
105
106 phi = zeros(1,length(t_array)+1);
107 theta = zeros(1,length(t_array)+1);
108 psi = psi_c;
109 dphi = zeros(1,length(t_array)+1);
110 dtheta = zeros(1,length(t_array)+1);
111 dpsi = zeros(1,length(t_array)+1);
112
113 u1_c = zeros(1,length(t_array)+1);
```

```
114 u2_c = zeros(3,length(t_array)+1);
115
116 w_des_sq = zeros(4,length(t_array)+1);
117 w_des = zeros(4,length(t_array)+1);
118 w = zeros(4,length(t_array)+1);
119
120 u = zeros(4,length(t_array)+1);
121
122 %INITIAL CONDITIONS-----------------------------------------------------
123 r(:,1) = r0;
124 dr(:,1) = dr0;
125
126 phi(1) = phi0;
127 theta(1) = theta0;
128 dphi(1) = dphi0;
129 dtheta(1) = dtheta0;
130
131 w(:,1) = sqrt(m*g/n/c_T)*ones(4,1);
132
133 for t_i = 1:length(t_array)
134 %TRAJECTORY TO BE TRACKED-----------------------------------------------
135     if t_array(t_i) < 5
136         r_T(:,t_i) = [t_array(t_i)/5*10;0;-10];
137     elseif (5≤ t_array(t_i))&&(t_array(t_i)< 10)
138         r_T(:,t_i) = [10;(t_array(t_i)-5)/5*10;-10];
139     elseif (10≤ t_array(t_i))&&(t_array(t_i)< 15)
140         r_T(:,t_i) = [10-(t_array(t_i)-10)/5*10;10;-10];
141     else
142         r_T(:,t_i) = [0;10-(t_array(t_i)-15)/5*10;-10];
143     end
144 end
145
146 %FIRST DERIVATIVE OF THE TRACKED POSITION
147 for t_i = 1:length(t_array)
148     dr_T(:,t_i) = (r_T(:,t_i+1)-r_T(:,t_i))/t_step;
149 end
150
151 %SECOND DERIVATIVE OF THE TRACKED POSITION
152 for t_i = 1:length(t_array)
153      ddr_T(:,t_i) = (dr_T(:,t_i+1)-dr_T(:,t_i))/t_step;
154 end
155
156 %THIRD DERIVATIVE OF THE TRACKED POSITION
```

```
157 for t_i = 1:length(t_array)
158      dddr_T(:,t_i) = (ddr_T(:,t_i+1)-ddr_T(:,t_i))/t_step;
159 end
160
161 for t_i = 1:length(t_array)
162 %CONTROLLED ACCERATIONS-------------------------------------------------
163     e_p(:,t_i) = r_T(:,t_i) - r(:,t_i);
164     e_v(:,t_i) = dr_T(:,t_i) - dr(:,t_i);
165     int_e_p(:,t_i) = ...
            [sum(e_p(1,1:t_i));sum(e_p(2,1:t_i));sum(e_p(3,1:t_i))];
166
167     ddr_c(:,t_i) = ddr_T(:,t_i) + Kd*e_v(:,t_i) + Kp*e_p(:,t_i) + ...
            Ki*int_e_p(:,t_i);
168 %COMMANDED ANGLES-------------------------------------------------------
169     phi_c(t_i) = sat_gd(1/g*( ddr_c(1,t_i)*sin(psi0) - ...
            ddr_c(2,t_i)*cos(psi0) ), theta_max);
170     theta_c(t_i) = sat_gd(1/g*( ddr_c(1,t_i)*cos(psi0) + ...
            ddr_c(2,t_i)*sin(psi0) ), theta_max);
171
172     if t_i ==1
173         dphi_c(t_i) = dphi0;
174         dtheta_c(t_i) = dtheta0;
175     else
176         dphi_c(t_i) = (phi_c(t_i)-phi_c(t_i-1))/t_step;
177         dtheta_c(t_i) = (theta_c(t_i)-theta_c(t_i-1))/t_step;
178     end
179 %INPUTS-----------------------------------------------------------------
180     u1_c(t_i) = sat_gd(m*(g + ddr_c(3,t_i))-(f_max+f_min)/2, ...
            (f_max-f_min)/2 ) + (f_max+f_min)/2;
181
182     u2_c(1:2,t_i)= sat_gd([ kp_phi* (phi_c(t_i) - phi(t_i)) + ...
            kd_phi*(dphi_c(t_i) - dphi(t_i) );
183                     kp_theta* (theta_c(t_i) - theta(t_i)) + ...
                        kd_theta*(dtheta_c(t_i) - dtheta(t_i) )],tau_max);
184     u2_c(3,t_i) = sat_gd(kp_psi* (psi_c(t_i) - psi(t_i)) + ...
            kd_psi*(dpsi_c(t_i) - dpsi(t_i) ),tau_max_z);
185
186 %CONTROL ALLOCATION-----------------------------------------------------
187     M_4 = [  c_T      c_T       c_T        c_T;
188              l*c_T/2  l*c_T/2  -l*c_T/2   -l*c_T;
189             -l*c_T/2  l*c_T/2   l*c_T/2   -l*c_T;
190              c_M     -c_M       c_M       -c_M];
191
```

```
192     if t_i==1
193         w_des_sq(:,t_i) = w(:,t_i).^2;
194     else
195         w_des_sq(:,t_i) = M_4\[u1_c(t_i); u2_c(:,t_i)];
196     end
197
198      w_des(:,t_i) = sqrt(w_des_sq(:,t_i) );
199
200 %MOTOR DYNAMICS-------------------------------------------------------------
201     w(:,t_i) = w_des(:,t_i).*(1-exp(-t_array(t_i)/T_m));
202
203     u(:,t_i) = M_4*w(:,t_i).^2;
204
205 %RIGID BODY DYNAMICS--------------------------------------------------------
206     ddr(1,t_i) = g*( phi(t_i)*sin(psi(t_i)) + ...
            theta(t_i)*cos(psi(t_i))  );
207     ddr(2,t_i) = g*(- phi(t_i)*cos(psi(t_i)) + ...
            theta(t_i)*sin(psi(t_i)) );
208     ddr(3,t_i) = u(1,t_i)/m - g;
209
210     dx1dt = @(t,x1)[x1(2); u(2,t_i)/Ixx]; % roll channel
211     dx2dt = @(t,x2)[x2(2); u(3,t_i)/Iyy]; % pitch channel
212     dx3dt = @(t,x3)[x3(2); (u(4,t_i) - ...
            dphi(t_i)*dtheta(t_i)*(Iyy-Ixx))/Izz]; % yaw channel
213     dx4dt = @(t,x4)[x4(2); g*( theta(t_i)*cos(psi(t_i)) + ...
            phi(t_i)*sin(psi(t_i)) ) ]; %x channel
214     dx5dt = @(t,x5)[x5(2); g*( theta(t_i)*sin(psi(t_i)) - ...
            phi(t_i)*cos(psi(t_i)) ) ]; %y channel
215     dx6dt = @(t,x6)[x6(2); u(1,t_i)/m - g]; %z channel
216
217     x10 = [phi(t_i), dphi(t_i) ];
218     x20 = [theta(t_i), dtheta(t_i)] ;
219     x30 = [psi(t_i), dpsi(t_i) ];
220     x40 = [r(1,t_i), dr(1,t_i)];
221     x50 = [r(2,t_i), dr(2,t_i)];
222     x60 = [r(3,t_i), dr(3,t_i)];
223
224     tLim = [t_array(t_i) - t_step, t_array(t_i) + t_step];
225
226     [t1Sol, x1Sol] = ode45(dx1dt, tLim, x10); %roll channel
227     [t2Sol, x2Sol] = ode45(dx2dt, tLim, x20); %pitch channel
228     [t3Sol, x3Sol] = ode45(dx3dt, tLim, x30); %yaw channel
229     [t4Sol, x4Sol] = ode45(dx4dt, tLim, x40); %x channel
```

```
230      [t5Sol, x5Sol] = ode45(dx5dt, tLim, x50); %y channel
231      [t6Sol, x6Sol] = ode45(dx6dt, tLim, x60); %z channel
232
233      %ATTITUDE CONTROLLER FEEDBACK
234      phi(t_i+1) = sum( x1Sol(:,1) )/length(t1Sol);
235      dphi(t_i+1) = sum( x1Sol(:,2) )/length(t1Sol);
236      theta(t_i+1) = sum( x2Sol(:,1) )/length(t2Sol);
237      dtheta(t_i+1) = sum( x2Sol(:,2) )/length(t2Sol);
238      psi(t_i+1) = sum( x3Sol(:,1) )/length(t3Sol);
239      dpsi(t_i+1) = sum( x3Sol(:,2) )/length(t3Sol);
240
241      %POSITION CONTROLLER FEEDBACK
242      r(:,t_i+1) = [sum(x4Sol(:,1) )/length(t4Sol); sum(x5Sol(:,1) ...
             )/length(t5Sol); sum(x6Sol(:,1) )/length(t6Sol) ];
243      dr(:,t_i+1) = [sum(x4Sol(:,2) )/length(t4Sol); sum(x5Sol(:,2) ...
             )/length(t5Sol); sum(x6Sol(:,2) )/length(t6Sol) ];
244  end
245
246  %GRAPHICAL ANALYSIS-------------------------------------------------------
247  figure(1)
248  plot(t_array,r(1,1:end-1),'r','LineWidth', 2 );
249  xlabel('Time [s]');
250  ylabel('x(t) [m]');
251  hold on
252  plot(t_array,r_T(1,1:end-1),'--k');
253  legend('Actual','Tracked')
254  set(gca,'FontSize',22)
255
256  figure(2)
257  plot(t_array,r(2,1:end-1),'g','LineWidth', 2 );
258  xlabel('Time [s]');
259  ylabel('y(t) [m]');
260  hold on
261  plot(t_array,r_T(2,1:end-1),'--k' );
262  legend('Actual','Tracked')
263  set(gca,'FontSize',22)
264
265  figure(3)
266  plot(t_array(1:end-1),r(3,1:end-1-1),'b','LineWidth', 2);
267  xlabel('Time [s]');
268  ylabel('z(t) [m]');
269  hold on
270  plot(t_array,r_T(3,1:end-1),'--k' );
```

```
271 legend('Actual','Tracked')
272 set(gca,'FontSize',22)
273
274 figure(4)
275 plot(t_array,dr(1,1:end-1),'r','LineWidth', 2  );
276 xlabel('Time [s]');
277 ylabel('$\dot{x}(t)$ [m/s]', 'interpreter', 'latex');
278 set(gca,'FontSize',22)
279 hold on
280 plot(t_array,dr_T(1,1:end-1) );
281 legend('Actual','Tracked')
282
283 figure(5)
284 plot(t_array,dr(2,1:end-1),'g','LineWidth', 2 );
285 xlabel('Time [s]');
286 ylabel('$\dot{y}(t)$ [m/s]', 'interpreter', 'latex');
287 set(gca,'FontSize',22)
288 hold on
289 plot(t_array,dr_T(2,1:end-1) );
290 legend('Actual','Tracked')
291
292 figure(6)
293 plot(t_array(1:end-1),dr(3,1:end-1-1),'b','LineWidth', 2 );
294 xlabel('Time [s]');
295 ylabel('$\dot{z}(t)$ [m/s]', 'interpreter', 'latex');
296 set(gca,'FontSize',22)
297 hold on
298 plot(t_array,dr_T(3,1:end-1) );
299 legend('Actual','Tracked')
300
301 figure(7)
302 plot(t_array,ddr(1,1:end-1) );
303 xlabel('time step');
304 ylabel('$\ddot{x}(t)$', 'interpreter', 'latex');
305 hold on
306 plot(t_array,ddr_T(1,1:end-1) );
307 legend('Actual','Tracked')
308
309 figure(8)
310 plot(t_array,ddr(2,1:end-1) );
311 xlabel('time step');
312 ylabel('$\ddot{y}(t)$', 'interpreter', 'latex');
313 hold on
```

```
plot(t_array,ddr_T(2,1:end-1) );
legend('Actual','Tracked')

figure(9)
plot(t_array,ddr(3,1:end-1) );
xlabel('time step');
ylabel('$\ddot{z}(t)$', 'interpreter', 'latex');
hold on
plot(t_array,ddr_T(3,1:end-1) );
legend('Actual','Tracked')

figure(10)
plot(t_array(1:end-1),phi(1:end-1-1) ,'r','LineWidth', 2 );
xlabel('Time [s]');
ylabel('$\phi(t)$ [rad]', 'interpreter', 'latex');
hold on
plot(t_array,phi_c(1:end-1), '--k');
legend('Actual','Commanded')
set(gca,'FontSize',22)

figure(11)
plot(t_array,theta(1:end-1),'g','LineWidth', 2 );
xlabel('Time [s]');
ylabel('$\theta(t)$ [rad]', 'interpreter', 'latex');
hold on
plot(t_array,theta_c(1:end-1),'--k' );
legend('Actual','Commanded')
set(gca,'FontSize',22)

figure(12)
plot(t_array,psi(1:end-1) ,'b','LineWidth', 2);
xlabel('Time [s]');
ylabel('$\psi(t)$ [rad]', 'interpreter', 'latex');
hold on
plot(t_array,psi_c(1:end-1),'--k' );
legend('Actual','Commanded')
set(gca,'FontSize',22)

figure(13)
plot(t_array(1:end-1),dphi(1:end-1-1),'r','LineWidth', 2 );
xlabel('Time [s]');
ylabel('$\dot{\phi}(t)$ [rad/s]', 'interpreter', 'latex');
set(gca,'FontSize',22)
```

```
357 hold on
358 plot(t_array,dphi_c(1:end-1) );
359 legend('Actual','Commanded')
360
361 figure(14)
362 plot(t_array,dtheta(1:end-1),'g','LineWidth', 2 );
363 xlabel('Time [s]');
364 ylabel('$\dot{\theta}(t)$ [rad/s]', 'interpreter', 'latex');
365 set(gca,'FontSize',22)
366 hold on
367 plot(t_array,dtheta_c(1:end-1) );
368 legend('Actual','Commanded')
369
370 figure(15)
371 plot(t_array,dpsi(1:end-1) ,'b','LineWidth', 2);
372 xlabel('Time [s]');
373 ylabel('$\dot{\psi}(t)$ [rad/s]', 'interpreter', 'latex');
374 set(gca,'FontSize',22)
375 hold on
376 plot(t_array,dpsi_c(1:end-1) );
377 legend('Actual','Commanded')
378
379 figure(16)
380 plot3(r(1,1:end-1),r(2,1:end-1),r(3,1:end-1), 'Color',[26 188 ...
       156]/255,'LineWidth', 2);
381 xlabel('x [m]')
382 ylabel('y [m]')
383 zlabel('z [m]')
384 xlim([-2 12])
385 ylim([-2 12])
386 zlim([-12 -8])
387 box on
388 hold on
389 plot3(r_T(1,1:end-1),r_T(2,1:end-1),r_T(3,1:end-1),'--k');
390 legend('Actual','Tracked')
391 set(gca,'FontSize',22)
392
393 figure(17)
394 plot(r(1,1:end-1),r(2,1:end-1), 'Color',[26 188 156]/255,'LineWidth', 2);
395 xlabel('x [m]')
396 ylabel('y [m]')
397 xlim([-2 12])
398 ylim([-2 12])
```

```
399 box on
400 hold on
401 plot(r_T(1,1:end-1),r_T(2,1:end-1), '--k');
402 legend('Actual','Tracked', 'Location', 'bestoutside')
403 set(gca,'FontSize',22)
404
405 figure(18)
406 plot(t_array(1:end-9),u1_c(1:end-10))
407 xlabel('Time step')
408 ylabel('u1')
409 hold on
410 plot(t_array(1:end-9),u(1,1:end-10))
411 legend('Controlled','Actual')
412
413 figure(20)
414 plot(t_array, w_des(1,1:end-1));
415 hold on
416 plot(t_array, w(1,1:end-1));
417 legend('Controlled','Actual')
418
419 %-------------------------------------------------------------------------
420 %DIRECTION GUARANTEED SATURATION FUNCTION-------------------------------
421 function u_sat = sat_gd(u,a)
422     mod_inf = max(abs(u));
423     if (mod_inf < a) || (mod_inf==a)
424         kappa = 1;
425     else
426         kappa = a/mod_inf;
427     end
428     u_sat = kappa*u;
429 end
430 %-------------------------------------------------------------------------
```

APPENDIX H – BASE ARDUINO CODE

Source: Author

```
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// INITIALIZATION
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
#include <Wire.h>//Importing the I2C library.
#include <Servo.h>
#include <PID_v1.h>

Servo M1, M2, M3, M4;

const int I2C_address_MPU = 0x68; //I2C Address of the MPU6050;
int16_t acc_x_raw, acc_y_raw, acc_z_raw; // Variables for the
Accelerometer sensor
int16_t gyro_x_raw, gyro_y_raw, gyro_z_raw; //Variables for the Gyroscope
int16_t temp_raw; //Variable in which the temperature is saved
float acc_x_offset, acc_y_offset, acc_z_offset; // offsetting values
float gyro_x_offset, gyro_y_offset, gyro_z_offset; //offsetting values
int16_t acc_x, acc_y, acc_z; // Variables for treated data
float gyro_x, gyro_y, gyro_z; //Variables for treated

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// FIXED QUANTITIES
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
const int n = 4;
const float m = 1.3420; //[kg]
const float g = 9.81; //[m/s^2]
const float Ixx = 0.0034; //[kg*m^2]
const float Iyy = 0.0054; //[kg*m^2]
const float Izz = 0.0050; //[kg*m^2]
const float l = 0.1829; //[m]

const float c_T = 1; //Corrected with PID parameters
const float c_M = 1; //Corrected with PID parameters
const float f_max = n*1.758*g;
const float f_min = 0.1*m*g;
const float T_max = f_max*l;
const float tau_max = f_max*l;
const float tau_max_z = (2/n)*(2.925*pow(10,-9)/(1.4865*pow(10,-7)
))*f_max;
const float w_max = sqrt(f_max/n/c_T);
const float theta_max = 81.93*PI/180;
const float  M_40[] = {.25,   .5,  -.5,   .25};
const float  M_41[] = {.25,   .5,   .5,  -.25};
const float  M_42[] = {.25,  -.5,   .5,   .25};
const float  M_43[] = {.25,  -.5,  -.5,  -.25};

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// TUNNING OF CONTROL PARAMETERS
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
const int xi_ang = 1;
const int omegan_ang = 9;
```

```
const int xi_lin = 1;
const int omegan_lin = 4;

const float kp_phi = Ixx*omegan_ang*omegan_ang;
const float kp_theta = Iyy*omegan_ang*omegan_ang;
const float kp_psi = Izz*omegan_ang*omegan_ang;
const float kd_phi = 2*Ixx*xi_ang*omegan_ang;
const float kd_theta = 2*Iyy*xi_ang*omegan_ang;
const float kd_psi = 2*Izz*xi_ang*omegan_ang;
const float kd_x = 2*xi_lin*omegan_lin;
const float kd_y = 2*xi_lin*omegan_lin;
const float kd_z = 20*2*xi_lin*omegan_lin;
const float kp_x = omegan_lin*omegan_lin;
const float kp_y = omegan_lin*omegan_lin;
const float kp_z = 4*omegan_lin*omegan_lin;
const float ki_x = 0.01;
const float ki_y = 0.01;
const float ki_z = 40*0.01;
const float T_m = 0.09;
const float Kd[][3] = { {kd_x,   0,    0},
                        {0   , kd_y,   0},
                        {0   , 0   ,kd_z} };
const float Kp[][3] = { {kp_x,   0,    0},
                        {0   , kp_y,   0},
                        {0   , 0   ,kp_z} };
const float Ki[][3] = { {ki_x,   0,    0},
                        {0   , ki_y,   0},
                        {0   , 0   ,ki_z}
};

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// INITIAL CONDITIONS
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
const float r0[] = {0, 0, 0};
const float dr0[] = {0, 0, 0};
const float ddr0[] = {0, 0, 0};
const float theta0 = 0;
const float phi0 = 0;
const float psi0 = 0;
const float dphi0 = 0;
const float dtheta0 = 0;
const float dpsi0 = 0;
const float w0[] = {4000/60, 4000/60, 4000/60, 4000/60};

////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// DEFINING VARIABLES
////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
unsigned long t0, t1;
float t_step;

unsigned long t0_int, t1_int;
float t_step_int;
```

```
float r[] = {}; //ACTUAL POSITION
float dr[] = {};
float ddr[] = {};
float dr_int[] = {};
float ddr_int[] = {};


float e_p[] = {};
float e_v[] = {};
float de_v[] = {};
float int_e_p[] = {};

float r_T[] = {}; //TRAJECTORY TO BE TRACKED
float dr_T[] = {};
float ddr_T[] = {};
float dddr_T[] = {};

float ddr_c[] = {}; //COMMANDED ACCELERATION

float phi_c;
float theta_c;
float psi_c = psi0;
float dphi_c;
float dtheta_c;
float dpsi_c;

float phi;
float theta;
float psi = psi_c;
float dphi;
float dtheta;
float dpsi;
float phi_c0;
float theta_c0;
float dphi_c0;
float dtheta_c0;
float dphi_int;
float dtheta_int;
float dpsi_int;

float u1_c;
float u2_c[] = {};
float u_c[] = {};

float w_des_sq[] = {};
float w_des[] = {};
float w[] = {};

float u[] = {};

float a;
float u_sat;
float kappa;
float mod_inf;

bool offset = true;
bool init_high = true;
```

```
bool init_low = true;

void setup() {
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(I2C_address_MPU); //Starting the I2C
transmission
  Wire.write(0x6B);
  Wire.write(0);
  Wire.endTransmission(true);

  M1.attach(3,1000,2000);  // (pin, min pulse width, max pulse width in
microseconds)
  M2.attach(4,1000,2000);  // (pin, min pulse width, max pulse width in
microseconds)
  M3.attach(5,1000,2000);  // (pin, min pulse width, max pulse width in
microseconds)
  M4.attach(6,1000,2000);  // (pin, min pulse width, max pulse width in
microseconds)

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
// ALLOCATING INITIAL CONDITIONS
////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
  for(int i = 0; i < 3; i++){
    r[i] = r0[i];
    dr[i] = dr0[i];

    r_T[i] = 0;
    dr_T[i] = 0;
    ddr_T[i] = 0;
  }

  phi = phi0;
  theta = theta0;
  dphi = dphi0;
  dtheta = dtheta0;
  dphi_c0 = dphi0;
  dtheta_c0 = dtheta0;

  for(int i = 0; i < 4; i++){
    w[i] = sqrt(m*g/n/c_T);
    w_des_sq[i] = w[i]*w[i];
  }

}

void loop() {
////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
// CONTROLLED ACCELERATIONS
////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////
   e_p[0] = r_T[0] - r[0];
   e_p[1] = r_T[1] - r[1];
   e_p[2] = r_T[2] - r[2];
```

```
   e_v[0] = dr_T[0] - dr[0];
   e_v[1] = dr_T[1] - dr[1];
   e_v[2] = dr_T[2] - dr[2];

   int_e_p[0] += e_p[0];
   int_e_p[1] += e_p[1];
   int_e_p[2] += e_p[2];

   ddr_c[0] = ddr_T[0] + Kd[0][0]*e_v[0]+ Kp[0][0]*e_p[0] +
Ki[0][0]*int_e_p[0];
   ddr_c[1] = ddr_T[1] + Kd[1][1]*e_v[1]+ Kp[1][1]*e_p[1] +
Ki[1][1]*int_e_p[1];
   ddr_c[2] = ddr_T[2] + Kd[2][2]*e_v[2]+ Kp[2][2]*e_p[2] +
Ki[2][2]*int_e_p[2];

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
// COMMANDED ANGLES
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

  t1 = millis();
  t_step = (t1-t0)/1000.0;
  t0 = millis();

  u_sat = 1/g*( ddr_c[0]*sin(psi0) - ddr_c[1]*cos(psi0) );
  a = theta_max;
  sat_gd();
  phi_c = u_sat;

  u_sat = 1/g*( ddr_c[0]*cos(psi0) + ddr_c[1]*sin(psi0) );
  a = theta_max;
  sat_gd();
  theta_c = u_sat;

  dphi_c = ( phi_c - phi_c0 )/t_step;
  dtheta_c = ( theta_c - theta_c0 )/t_step;
  Serial.print(phi_c0);
  Serial.println();

  phi_c0 = phi_c;
  theta_c0 = theta_c;

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
//INPUTS
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

  u_sat = m*(g + ddr_c[2])-(f_max+f_min)/2;
  a = (f_max-f_min)/2;
  sat_gd();
  u1_c = u_sat + (f_max+f_min)/2;

  u_sat = kp_phi* (phi_c - phi) + kd_phi*(dphi_c - dphi );
  a = tau_max;
```

```
  sat_gd();
  u2_c[0] = u_sat;

  u_sat = kp_theta* (theta_c - theta) + kd_theta*(dtheta_c - dtheta );
  a = tau_max;
  sat_gd();
  u2_c[1] = u_sat;

  u_sat = kp_psi* (psi_c - psi) + kd_psi*(dpsi_c - dpsi );
  a = tau_max_z;
  sat_gd();
  u2_c[2] = u_sat;

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
//CONTROL ALLOCATION - MOTOR
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////

  u_c[0] = u1_c;
  u_c[1] = u2_c[0];
  u_c[2] = u2_c[1];
  u_c[3] = u2_c[2];

  for (int i = 0; i < 4; i++){
    w_des_sq[0] += M_40[i]*u_c[i];
    w_des_sq[1] += M_41[i]*u_c[i];
    w_des_sq[2] += M_42[i]*u_c[i];
    w_des_sq[3] += M_43[i]*u_c[i];
    //Serial.print(tau_max);
    //Serial.println();
  }

  w_des[0] = sqrt(w_des_sq[0]);
  w_des[1] = sqrt(w_des_sq[1]);
  w_des[2] = sqrt(w_des_sq[2]);
  w_des[3] = sqrt(w_des_sq[3]);

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
//FEEDBACK
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
  if( offset == true){
    for(int i = 0; i < 2000; i++){
      read_MPU6050();
      acc_x_offset += acc_x_raw;
      acc_y_offset += acc_y_raw;
      acc_z_offset += acc_z_raw;
      //temp += temp_raw;
      gyro_x_offset += gyro_x_raw;
      gyro_y_offset += gyro_y_raw;
      gyro_z_offset += gyro_z_raw;
    }
    acc_x_offset /= 2000.00;
    acc_y_offset = abs( acc_x_offset/16384.00*(-g) );
    acc_y_offset /= 2000.00;
```

```
    acc_y_offset = abs( acc_y_offset/16384.00*(-g) );
    acc_z_offset /= 2000.00;
    acc_z_offset = abs( acc_z_offset/16384.00*(-g) );
    //temp /= 2000;
    gyro_x_offset /= 2000.00;
    gyro_x_offset = abs( gyro_x_offset/131.00);
    gyro_y_offset /= 2000.00;
    gyro_y_offset = abs( gyro_y_offset/131.00);
    gyro_z_offset /= 2000.00;
    gyro_z_offset = abs( gyro_z_offset/131.00);
    offset = false;
  }
    read_MPU6050();

//ACCELERATION CONTROLLER FEEDBACK
    ddr[0] = acc_x_raw/16384.00*(-g);
    ddr[0] += acc_x_offset;
    ddr[1] = acc_y_raw/16384.00*(-g);
    ddr[1] += acc_y_offset;
    ddr[2] = acc_z_raw/16384.00*(-g);
    ddr[2] += acc_z_offset;

//ATTITUDE CONTROLLER FEEDBACK

    t1_int = millis();
    t_step_int = (t1_int - t0_int)/1000.0;
    dphi = gyro_x_raw/131.00;
    dphi += gyro_x_offset;
    dphi *= PI/180;
    dtheta = gyro_y_raw/131.00;
    dtheta += gyro_y_offset;
    dtheta *= PI/180;
    dpsi = gyro_z_raw/131.00;
    dpsi += gyro_z_offset;
    dpsi *= PI/180;

    phi = (dphi - dphi_int)*t_step_int;
    theta = (dtheta - dtheta_int)*t_step_int;
    psi = (dpsi - dpsi_int)*t_step_int;

//POSITION CONTROLLER FEEDBACK
    dr[0] = (ddr[0] - ddr_int[0])*t_step_int;
    dr[1] = (ddr[1] - ddr_int[1])*t_step_int;
    dr[2] = (ddr[2] - ddr_int[2])*t_step_int;

    r[0] = (dr[0] - dr_int[0])*t_step_int;
    r[1] = (dr[1] - dr_int[1])*t_step_int;
    r[2] = (dr[2] - dr_int[2])*t_step_int;

    t0_int = millis();
    dphi_int = dphi;
    dtheta_int = dtheta;
    dpsi_int = dpsi;
    dr_int[0] = dr[0];
    dr_int[1] = dr[1];
    dr_int[2] = dr[2];
```

```
    ddr_int[0] = ddr[0];
    ddr_int[1] = ddr[1];
    ddr_int[2] = ddr[2];

//  Serial.print(w_des[0]);
//  Serial.println();
//  Serial.print(w_des[1]);
//  Serial.println();
//  Serial.print(w_des[2]);
//  Serial.println();
//  Serial.print(w_des[3]);
//  Serial.println();
//  Serial.print("----------------------------------------------------------
----------------");
//  Serial.println();
//  delay(1500);

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
// MOTOR COMMANDS
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////
  w_des[0] *= 1000;
  w_des[1] *= 1000;
  w_des[2] *= 1000;
  w_des[3] *= 1000;

  w_des[0] = (int) w_des[0];
  w_des[1] = (int) w_des[1];
  w_des[2] = (int) w_des[2];
  w_des[3] = (int) w_des[3];

  w_des[0] = map(w_des[0], 0, 1000, 0, 180);   // scale it to use it with
the servo library (value between 0 and 180)
  w_des[1] = map(w_des[1], 0, 1000, 0, 180);
  w_des[2] = map(w_des[2], 0, 1000, 0, 180);
  w_des[3] = map(w_des[3], 0, 1000, 0, 180);
//  Serial.print(w_des[0]);
//  Serial.println();
//  Serial.print(w_des[1]);
//  Serial.println();
//  Serial.print(w_des[2]);
//  Serial.println();
//  Serial.print(w_des[3]);
//  Serial.println();
//  Serial.print("----------------------------------------------------------
----------------");
//  Serial.println();
//  delay(1500);

  if(init_high == true){
    M1.write(180);    // Send the signal to the ESC
    M2.write(180);
    M3.write(180);
    M4.write(180);
    init_high = false;
    delay(8000);
```

```
    //Serial.print(1);
  }
  if(init_low == true) {
    M1.write(0);    // Send the signal to the ESC
    M2.write(0);
    M3.write(0);
    M4.write(0);
    init_low = false;
    delay(8000);
    //Serial.print(2);
  }
  else{
    M1.write(w_des[0]);    // Send the signal to the ESC
    M2.write(w_des[1]);
    M3.write(w_des[2]);
    M4.write(w_des[3]);
    delay(15);
    //delay(30);
  }

}

////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
// FUNCTIONS
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
// MPU6050
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
void read_MPU6050(){
  Wire.beginTransmission(I2C_address_MPU); //Starting the I2C
transmission
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(I2C_address_MPU, 7*2, true);
  acc_x_raw = Wire.read()<<8 | Wire.read();
  acc_y_raw = Wire.read()<<8 | Wire.read();
  acc_z_raw = Wire.read()<<8 | Wire.read();
  temp_raw = Wire.read()<<8 | Wire.read();
  gyro_x_raw = Wire.read()<<8 | Wire.read();
  gyro_y_raw = Wire.read()<<8 | Wire.read();
  gyro_z_raw = Wire.read()<<8 | Wire.read();
}

////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
// DIRECTION GUARANTEED SATURATION FUNCTION
////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
```

```
void sat_gd(){
  mod_inf = abs(u_sat);
  if ( (mod_inf < a) || (mod_inf == a) ){
      kappa = 1;
  }
  else{
      kappa = a/mod_inf;
  }
  u_sat = kappa*u_sat;
}
```

ANNEX A – PETG: TECHNICAL SHEET

Source: [28]

PET-G Easy Go Filament 1.75mm

PET-G is the best material for advanced users:

– Fast solidification, higher heat resistance (Glass transition temperature: 70ºC - 75ºC), very hard and flexible.

– Does not require Kapton tape.

– Heated bed needed at 60ºC-80ºC.
Important: *do not clean the bed with alcohol.*

– Due to the nature of the PET-G, we recommend regular cleaning of the hotend exterior and the nozzle.

– Keep in a dry environment and away from light.

PET-G is the best alternative to ABS: it is non-toxic, recyclable and releases fewer particles into the air.

Filament Diameter: 1.75 mm
Thickness: 1.27 g/cm³ (ASTM D792)
Weight: 1 kg
Spool Size: 175 mm x 77 mm
Enclosure: 187 x 187 x 83 mm
Spool axle diameter: 44 mm

Compatible with printers with heated bed and 1.75 mm filament.

SKU: F000174
8 434663 037059
Coal Black

SKU: F000173
8 434663 037042
Pure White