AALBORG UNIVERSITY

Network Communication in Autonomous Quadcopters in Formation Flight

Authors:
Jorge Val
Ricard Bordalba
Jeff Hindsborg

Supervisor:
Henrik Schiøler

Worksheets submitted as supplementary material to

Group 730
Department of Electronic Systems

December 16, 2014

AALBORG UNIVERSITET

AALBORG UNIVERSITY

Department of Electronic Systems
Frederik Bajers Vej 7B
9220 Aalborg Ø
http://www.es.aau.dk

Title:
Network Communication in Autonomous Quadcopters in Formation Flight

Theme:
Networked Control Systems

Project period:
2014: 2. September - 17. December

Project group:
Group 730

Members:

Jorge Val

Ricard Bordalba

Jeff Hindsborg

Supervisor:
Henrik Schiøler

No. of printed copies: 5

No. of appendix: 1

Total no. of pages: 104

Completed: December 16, 2014

Abstract:

This report studies the behaviour of a network in a quadcopter controller, designed for both single and formation flight applications. Different control strategies were investigated, including classical PID and state space methods, such as integral control and LQR. The LQR approach was applied, as the it yielded the most optimal results. The performance of the controller is tested in different scenarios of data loss probability and path-loss. The limits of the quadcopter controller caused by loss of data are found. Results shows that the simulated system is stable until package loss probability reaches approximately 50%. The maximum distance between receiver and transmitter, in terms of communication breakdown, is also simulated. Both studies can be applied to any networked systems with this control setup. Finally, flight formation restrictions are found. Additionally, a Kalman Filter was developed and implemented, along with other algorithms on the autopilot, applied to the quadcopter.

PREFACE

This worksheet has been composed by a group of students, studying Automation & Control at Aalborg University. The group, whose reference is 730, have been working on the project from September 2, to December 16, 2014. The focus of the project is modeling and controlling networked systems.

Reading Guide

Unless otherwise noted, this report uses the following notations:

- Cites and references to sources are denoted by square brackets containing a number, directing to the bibliography section. Detailed informations are written in the order showed below:
  - Books: Author(s), title, publisher, year, ISBN, page(s) or chapter(s).
  - Articles: Author(s), title, journal, year, page(s).
  - Homepages: Author(s), title, year, URL, date of visit.
- Figures and tables will be numbered according to chapters and sequence. For example the third figure in chapter four will be denoted as "Figure 4.3". Below every figure and table a short explanatory text is introducing the content along with a reference.
- Abbreviations will be introduced in their extended form the first time they appear.

CONTENTS

Part I

Preliminaries

CHAPTER 1

INTRODUCTION

1.1 Project Description

The fundamental basis of this project is the application of autonomous quadcopters as assisting instruments for search and rescue operations in disaster areas. The increasing rate of quadcopter's developments in the last few years has permitted unmanned aerial vehicles (UAV) to be used in many applications, such as flight formation. Flight formation combined with surveillance technology, applied in areas that are hardly accessible to humans, pose a great potential for fast and precise rescue operations, where time may be of the utmost importance to save human life. Flight formation allows the quadcopters to cover a larger area in a shorter period of time, and can evaluate location of possible points of interest using e.g computer vision. To accomplish this ability, a fast and reliable network communication between the quadcopters and the ground station is crucial.

In this project, the behaviour of a networked quadcopter flight formation will be investigated through different simulations, with respect to potential network disturbances. During e.g camera surveillance, the quadcopters will be flying in formation with a semi-constant relative distance between each other as shown in Fig. 1.1, such that no area remains unscanned. This distance vector will be maintained by a continuous inter-communication between the quadcopters, in a distributed real-time network, such that the quadcopters will not collide into each other. In addition, this relative distance between the quadcopters can impose a restriction, as to how close they are allowed to fly to one another, due to how fast the control response will be.

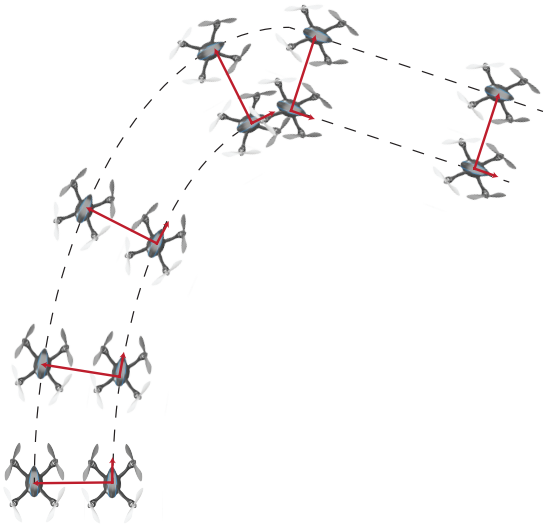

Figure 1.1: Flight formation representation.

All units will have a short range communication module, while one will have both short and long range in order to receive the global position from a ground station. This will limit the heavy power consumption to one quadcopter, since long range communication can be expensive in terms of battery. Communication failures may occur, depending on the physical landscape the quadcopter is located in. For example, if the signal have to go through more transmission mediums, the signal strength will be attenuated and slowed down. If other electrical devices are present, which applies the same frequency for transmission, it can cause interference as well. Lastly, the physical distance between the transmitter and receiver will also have a great impact on signal strength, and possibly interference (if other networks using the same frequency, are communicating inbetween). As opposed to the non-line-of-sight propagation, which infers the most signal disturbances, the line of sight is considered the ideal scenario.
Initially in this project, a controller will be designed, which will provide the means for autonomous flight. During the design of the controller, certain assumptions will be made, which may have an impact on the collected response of a formation of quadcopters. Furthermore, it will be investigated how the controller will behave in a network failure scenario. Lastly, the response of three quadcopters in a flight formation will be investigated, such that stability is maintained and collisions are avoided.

1.1.1 Project Requirements

The specifications of the project, and thereby the quadcopters will be listed and explained below.

1. **Modelling a Quadcopter -** assuming all quadcopters in the formation will be of the same size, weight etc.

2. **Control Design -** applying control theory, both classical and modern, to design a controller and find the most suitable design for deployment.

3. **Formation Flight -** a vector will define the relative distance between each quadcopter. The quadcopters in the flight formation will all have the same altitude.

4. **Distributed Real-Time Network -** should satisfy the following

    - Quadcopters communicate internally to maintain flight-formation.
    - All units except one will have short range communication to communicate internally, while one unit will communicate both short range with the quadcopters and long range with the ground station, i.e. master and slaves hierarchy.
    - Ground station will collect the position of the quadcopters from a motion capture system, which will be forwarded to the quadcopters through the wireless link.

1.1.2 Limitations

Limitations to this project has been listed up below.

- It is assumed that all units in the formation will have the same physical parameters.
- Three quadcopters will be applied.
- The angular position of the quadcopter is limited by the linearisation constraints.
- All units apply the same communication device.
- It is assumed that there are no sudden obstacles in the altitude of the quadcopters, thus only following a constant altitude related to the terrain (hills, or minor variations), since no sensors will evaluate obstacles right in its path.
- Network analysis will be investigated through simulations, using suitable software.
- The quadcopter controller will be tested in laboratory environment using motion capture, before outside flight.
- The study of the path-loss will be carried out through simulations, as it will be experimentally difficult to realize. Alternatively, the quadcopter can be equipped with a GPS, however it will not deliver position data as accurate as a motion capture system.

1.1.3 Project Overview

In accordance to the project requirements and limitations, the goal of this project is to investigate different control strategies that will lead to a proper controller for the quadcopter. Furthermore, network influence on the behaviour of the quadcopter is studied under certain circumstances, where communication breakdowns can occur. Finally an approach to flight formation is conducted and analysed.

Part II

Theory & Modelling

CHAPTER 2

THEORY

The theory chapter will provide the reader with the basic concepts and methods that has been applied throughout this project. If the reader is familiar with state space theory, Sec. 2.1 can be skipped.

2.1 Control Theory

Control based on state space is used throughout this project and a brief introduction is given.

2.1.1 State Space

The State Space Model or representation is a mathematical model that represents a physical system, consisting of input, output and state variables described through differential equations. Assuming p inputs, q outputs and n state variables, it can be expressed as

$$\dot{x} = Ax + Bu \tag{2.1}$$

$$y = Cx + Du \tag{2.2}$$

where

- x is the state vector, $\mathbb{R}^n$,
- y is the output vector, $\mathbb{R}^q$,
- u is the control vector (or input), $\mathbb{R}^p$,
- A is the state or system vector, $\mathbb{R}^{n\times m}$,
- B is the input matrix, $\mathbb{R}^{n\times p}$,
- C is the output matrix, $\mathbb{R}^{q\times n}$,

- D is the feedforward matrix, which is negligible if there is no direct feedthrough, $\mathbb{R}^{q\times p}$.

D will however be assumed to be zero.
Performing a Laplace transform and rearranging above equations, yields

$$x(s) = (sI - A)^{-1}Bu(s)$$
$$y(s) = C(sI - A)^{-1}Bu(s) + Du(s) = G(s)u(s) \tag{2.3}$$

The characteristic polynomial from $\det(\lambda I - A) = 0$ will result in the eigenvalue(s), λ, thus revealing the placement of the poles in the system [10].

State Feedback

For the state space model in Eq. 2.1, a state feedback is of the form

$$u = Fx \tag{2.4}$$

Combining the two equation will yield

$$\dot{x} = Ax + BFx = (A + BF)x. \tag{2.5}$$

Consequently implemented a feedback, results in a modified system with modified poles. A block diagram is illustrated in Fig. 2.1, illustrating the feedback.

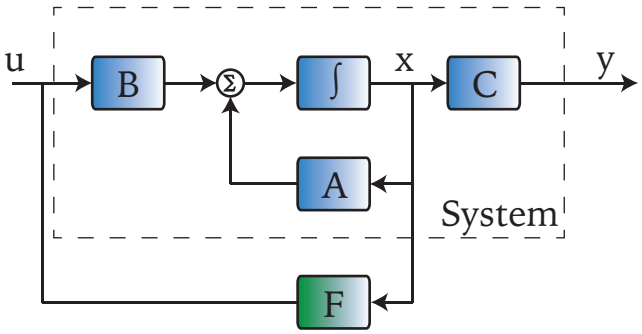

Figure 2.1: State Feedback block diagram.

Controllability

Definition : ***a system is controllable if it is possible to go from any initial state to any final state.***
Thus, in order to control a dynamic system under control input, it must be controllable. The aforementioned system is said to be controllable if and only if

$$\text{rank}\,(\mathbf{C}) = n$$

such that it has full rank, n. In the case where the amount of control inputs, $p = 1$, it reduces to

$$\det(\mathbf{C}) \neq 0$$

where $\mathbf{C}$ is the **controllability matrix**:

$$\mathbf{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix}. \tag{2.6}$$

Additionally, pole-placement to arbitrary eigenvalues in State Feedback is possible if and only if the system is controllable [10].

2.2 Coordinate Systems

While working with quadcopters or drones in 3D-space, two frames of reference are necessary to fully describe the system. The first coordinate system will throughout these worksheets be referred to as the 'global frame' (often called the earth fixed frame), while the other will be the 'local body frame'.
The global frame is a fixed reference frame and the local frame will follow the attitude of the quadcopter rotating. The quadcopter is assumed to be a rigid body, such that any two points on the body, will remain the same, regardless of how it rotates.

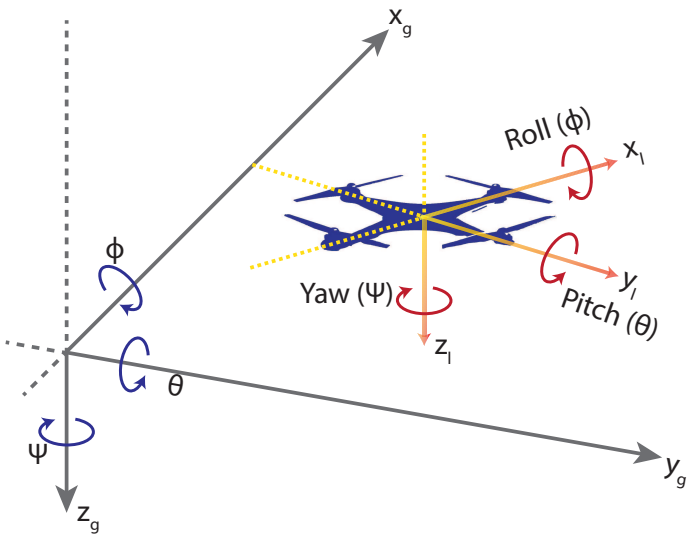

Figure 2.2: The coordinate system to the left, depicts the global reference frame, while the coordinate system on the quadcopter depicts the local frame.

Obtaining the orientation of the quadcopter, in terms of angles, these two set of frames will be linked and described in Sec. 2.3. In avionics, it's convention to define the Euler angles as depicted in Fig. 2.2. Roll, Yaw and Pitch will be described in relation to a quadcopter in a later section. The position of the quadcopter is described by $\mathbf{P} = [x, y, z]^T$, angular velocity of the local frame is denoted ω_l and translators velocity v_l. Subsequently a sub-index l or g on the right-side of the variable will determine whether it is in the local or global frame [2].

2.3 Orientation and Rotations

There are different ways to parametrize a moving rigid body, where some of the most widely used techniques are the Euler angle parametrization $\Phi = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$ and quaternion parametrization, $\mathbf{q} = \begin{bmatrix}q_o & q_1 & q_2 & q_3\end{bmatrix}^T$ [22].
Parametrization using Euler will involve different trigonometric functions and will consequently render the transformation non-linear.

Quaternion parametrization contains quadratic expressions, but no trigonometric. Thus, while not being completely linear, it will not cause singularities. [6]

2.3.1 Euler Angles

Euler angles, which builds on Cartesian coordinates, and is very intuitively understood compared to quaternions, is applied through rotation matrices. As illustrated in Fig. 2.2, angles in avionics are usually separated up into roll, yaw and pitch, the orientation of the angles is defined according the right-handed rule.
Calculating the rotation ϕ matrix around the x-axis, as in Fig. 2.3 is represented:

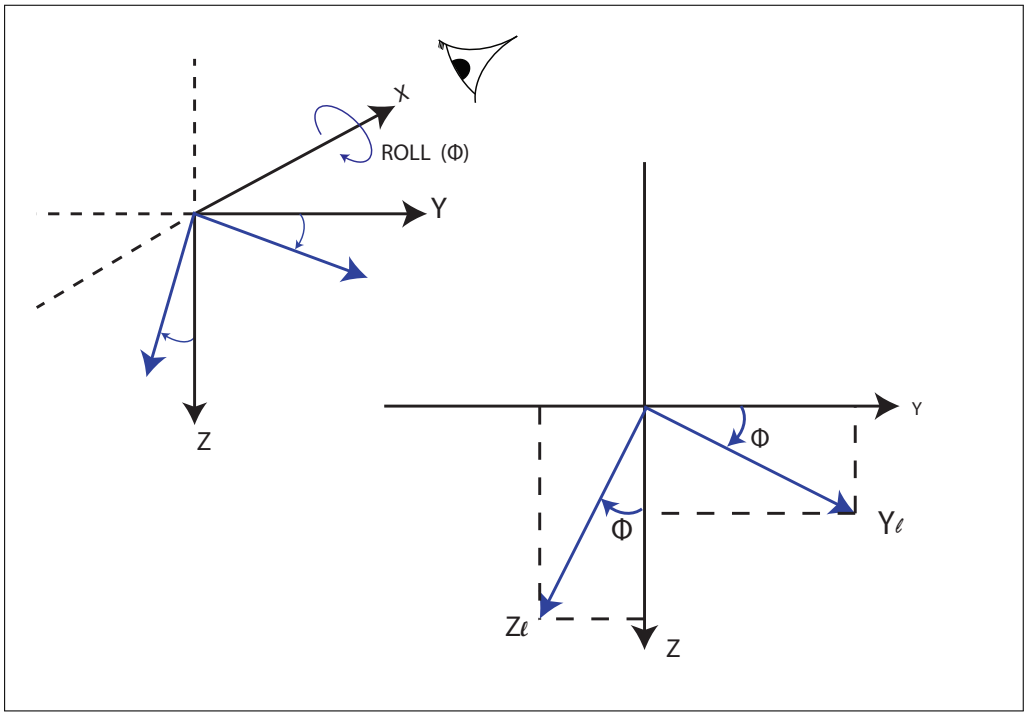

Figure 2.3: Example of frame rotation over x-axis through roll angle ϕ.

$$\text{Roll:} \qquad R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}. \tag{2.7}$$

Additionally, same procedure is follow to the pitch, angle around y-axis, and yaw, rotation around z-axis is:

$$\text{Pitch:} \qquad R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\phi) \end{bmatrix}. \tag{2.8}$$

and

$$\text{Yaw:} \qquad R_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.9}$$

which can be used to construct the rotational matrix from global to local frame such that

$$\mathbf{R}_{g-l}(\Phi) = R_x(\phi)R_y(\theta)R_z(\psi) \tag{2.10}$$

Which becomes

$$\mathbf{R}_{g-l}(\Phi) = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ -\cos(\phi)\sin(\psi)+\sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\psi)+\sin(\phi)\sin(\theta)\sin(\psi) & \sin(\phi)\cos(\theta) \\ \sin(\phi)\sin(\psi)+\cos(\phi)\sin(\theta)\cos(\psi) & -\sin(\phi)\cos(\psi)+\cos(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\cos(\theta) \end{bmatrix}. \tag{2.11}$$

that ultimately gives the total conversion from global to local frame. In some cases, it is necessary to apply the opposite conversion, which involves taking its inverse.
A trait of rotation matrices is its orthogonality, which allows for

$$\mathbf{R}^{-1}_{g-l}(\Phi) = \mathbf{R}^{T}_{g-l}(\Phi) \qquad \text{and} \qquad \det(\mathbf{R}(\Phi)) = 1. \tag{2.12}$$

that will allow for fast computation.
The opposite rotation, from local to global is given by [13]

$$\mathbf{R}_{l-g}(\Phi) = \mathbf{R}^{T}_{g-l}(\Phi) \tag{2.13}$$

2.4 Kinematics

Throughout this section, the description of the geometry of motion of the quadcopter will be described. The quadcopter is assumed to be a rigid body, and the kinematic differential properties such as velocity and acceleration will be described.

Starting with $\mathbf{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ the position of the quadcopter, the conversion of the corresponding velocities $\mathbf{v}$ from one frame to another can be determined by concatenating with the rotational matrix described in the previous section.

$$\mathbf{v}_g = \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix}_g = \mathbf{R}_{l-g}(\Phi) \begin{bmatrix} \mathbf{v}_x \\ \mathbf{v}_y \\ \mathbf{v}_z \end{bmatrix}_l \tag{2.14}$$

Consequently, the velocity in the local frame can be obtained using the same procedure with the inverse matrix:

$$\mathbf{v}_l = \mathbf{R}_{g-l}(\Phi)\mathbf{v}_g \tag{2.15}$$

Additionally, as the angular rates $\omega_l = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ around the three local frame axis, that are obtained from the IMU, will be given in the local frame and the quadcopter Euler angles $\Phi = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ and velocities $\omega_\Phi = \begin{bmatrix} \omega_\phi & \omega_\theta & \omega_\psi \end{bmatrix}^T$ are in the global frame. In order to know its orientation, it is important to be able to transform the angular velocities. Therefore, the conversion from global angular velocity to local frame using the rotation matrices is given by the following equation: [19]

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix} = \begin{bmatrix} \omega_x \\ 0 \\ 0 \end{bmatrix}_l + R_x(\phi) \begin{bmatrix} 0 \\ \omega_y \\ 0 \end{bmatrix}_l + R_x(\phi) R_y(\theta) \begin{bmatrix} 0 \\ 0 \\ \omega_z \end{bmatrix}_l = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_l \tag{2.16}$$

In order to obtain the opposite relationship, the inverse matrix can be computed:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_l = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.17}$$

The 3 angular and 3 linear velocities now determine the 6 Degrees Of Freedom (DOF) of the quadcopter.

2.5 Dynamics

Throughout this section the dynamics of the quadcopter will be described. Dynamics is the branch of mechanics that describes forces and torques and their impact on motion. Thus it can be divided into two parts, one describing the linear dynamics that pertains the quadcopter moving in a line, using physical quantities such as force, mass, displacement, velocity and acceleration. The second part is the rotational dynamics, which as the name infers, describes objects rotating or moving in a curved path, where quantities such as torque, moment of inertia, angular displacement, velocity and accelerations are included.

2.5.1 Newton-Euler Equations

The Newton-Euler equations are applied, as they describe both rotational and translational dynamics of a rigid body. Thus the quadcopter is considered to be a solid body with its centre of gravity (COG) at its geometrical centre. They state the following [9]:

$$\begin{bmatrix} F \\ \tau \end{bmatrix} = \begin{bmatrix} m\mathbb{I}_{3x3} & 0 \\ 0 & I_{CM} \end{bmatrix} \begin{bmatrix} \ddot{P} \\ \ddot{\Phi} \end{bmatrix} + \begin{bmatrix} \dot{\Phi} \times m\dot{P} \\ \dot{\Phi} \times I_{CM}\dot{\Phi} \end{bmatrix}, \tag{2.18}$$

where m the total mass of the rigid body, I_{3x3} a 3x3 identity matrix, F and τ are the vectors of forces and torques applied to the quadcopter.
Initially the local frame is used for the rotational movement [18], since

- The inertia matrix I_{CM} around the centre of masses is constant.
- Body symmetry simplifies the equations.
- Measurements from the gyroscope sensor are in the local frame.

On the other hand, the global frame is used in the linear movement, as the desired position of the quadcopter is in this frame. In developing the matrix of Eq. 2.18 which is the result of two separate equations, one defining the linear acceleration and the other the angular acceleration:

$$\dot{v} = \ddot{P} = \frac{F}{M} + \dot{\Phi} \times \dot{P} = \frac{F}{M} + \omega_\Phi \times v \tag{2.19}$$

$$\dot{\omega} = \ddot{\Phi} = {I_{CM}}^{-1}(\tau - \dot{\Phi} \times I_{CM}\dot{\Phi}) = {I_{CM}}^{-1}(\tau - \omega_\Phi \times I_{CM}\omega_\Phi) \tag{2.20}$$

Rearranging both equations, Eq. 2.21 and 2.22 are found. Equation for position is developed in the global frame and the angular equation in the local. Next step is to develop the vectorial product inside the matrix in order to see the acceleration in every axis.

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix}_g = \frac{1}{M} \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} + \begin{bmatrix} \omega_\psi v_y - \omega_\theta v_z \\ \omega_\phi v_z - \omega_\psi v_x \\ \omega_\theta v_x - \omega_\phi v_y \end{bmatrix} \tag{2.21}$$

where $[F_x, F_y, F_z]^T$ is the sum of forces in the global frame, M is the total mass of the quadcopter, $[v_x, v_y, v_z]^T$ is the velocity in the global frame and $[\omega_\phi, \omega_\theta, \omega_\psi]^T$ is the angular velocity

also in global frame.

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}_l = {I_{CM}}^{-1} \left(\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} - \begin{bmatrix} \omega_y \omega_z (I_{zz} - I_{yy}) \\ \omega_x \omega_z (I_{xx} - I_{zz}) \\ \omega_y \omega_x (I_{yy} - I_{xx}) \end{bmatrix} \right) = \begin{bmatrix} \frac{1}{I_{xx}} \tau_\phi \\ \frac{1}{I_{yy}} \tau_\theta \\ \frac{1}{I_{zz}} \tau_\psi \end{bmatrix} + \begin{bmatrix} \omega_y \omega_z \frac{I_{yy} - I_{zz}}{I_{xx}} \\ \omega_x \omega_z \frac{I_{zz} - I_{xx}}{I_{yy}} \\ \omega_y \omega_x \frac{I_{xx} - I_{yy}}{I_{zz}} \end{bmatrix} \tag{2.22}$$

where $[\omega_x, \omega_y, \omega_z]^T$ are the angular velocities around the local frame axis, $[\tau_x, \tau_y, \tau_z]^T$ are the torques in the local frame and $[I_{xx}, I_{yy}, I_{zz}]^T$ the inertial moments around each axis of the local frame.
The second term in both equations belongs to what is called the Coriolis effect.

2.5.1.1 Inertia matrix

Initially, the inertial matrix will be formulated. The quadcopter is modelled as a central sphere with radius r and mass m_0, while its motors are considered point masses, m_m, at a distance l from the centre of the sphere. [19]

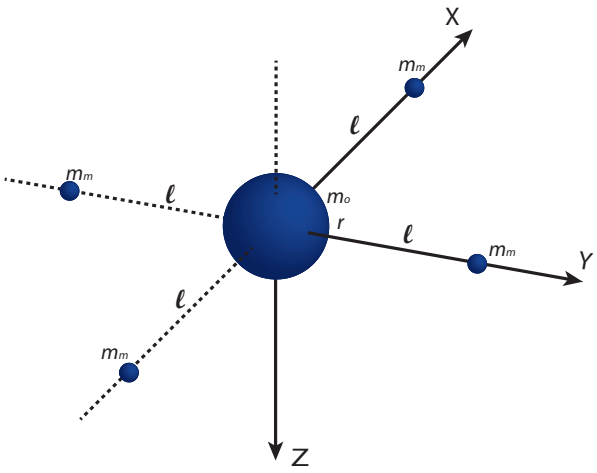

Figure 2.4: Quad-copter modelling to compute the inertial matrix

Consequently, the inertia matrix of the quadcopter becomes a diagonal matrix with the inertial moment around the 3 axes in the local frame

$$I_{CM} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.23}$$

In order to obtain the values for the matrix, the sphere and the 4 mass points are studied separately. As they are very common bodies, their moments of inertia can be easily found [16].

- **Solid sphere -** It is located at the centre of the 3 axis. Due to its symmetry, the moments of inertia are the same in the three axis:

$$I_{xx} = I_{yy} = I_{zz} = \frac{2m_o r^2}{5} \tag{2.24}$$

- **Point mass -** Its inertial moments are easily computed as a summation of the individual contributions to the total moment on each axis. While the front and back motors do not produce a moment of inertia around x as they are already on this axis, the left and right motors do not produce it around the y axis for the same reason. Then, the inertial moments are found as

$$I = \sum m_i r_i^2 \tag{2.25}$$

$$I_{xx} = m_f 0^2 + m_b 0^2 + m_l l^2 + m_r l^2 = 2m_m l^2 \tag{2.26}$$

$$I_{yy} = m_f l^2 + m_b l^2 + m_l 0^2 + m_r 0^2 = 2m_m l^2 \tag{2.27}$$

$$I_{zz} = m_f l^2 + m_b l^2 + m_l l^2 + m_r l^2 = 4m_m l^2 \tag{2.28}$$

Now, adding both the solid sphere and point masses inertial moments per each axis, the total inertia moment is obtained:

$$\begin{aligned} I_{xx} = I_{yy} &= \tfrac{2m_o R^2}{5} + 2m_m l^2 \\ I_{zz} &= \tfrac{2m_o R^2}{5} + 4m_m l^2 \end{aligned} \tag{2.29}$$

Its inverse matrix is found directly:

$$I_{CM}^{-1} = \begin{bmatrix} \frac{1}{I_{xx}} & 0 & 0 \\ 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix} \tag{2.30}$$

2.5.1.2 Forces and Torques

In extension to the results obtained in the beginning of the dynamics section, forces and torques will be formulated. In terms of this project, modelling of aerodynamic forces and moments are considered negligible. Thus the system is only affected by the gravitational force and the four motor forces (thrust) and its torques, while Coriolis is neglected.
These forces and torques are linearly related with the square angular velocity Ω_i^2, while these square angular velocities and the PWM signals sent to the motors by the microcontroller are linear with an offset [3]:

$$F_i = k_F \cdot \Omega_i^2 \tag{2.31}$$

$$\tau_i = k_\tau \cdot \Omega_i^2 \tag{2.32}$$

$$PWM_i = k_{PWM} \cdot \Omega_i^2 + PWM_0 \tag{2.33}$$

Where k_{PWM}, k_F and k_τ are proportional coefficients which depend on blade features like angle of attack, size of the blade and air density. On the other hand, PWM_0 is the minimum PWM signal needed to get the motors rotating and, as It is just an offset, the assumption that $PWM'_i = PWM_i - PWM_0$ such that it can be expressed without an offset:

$$PWM'_i = k_{PWM} \cdot \Omega_i^2 \tag{2.34}$$

$$\Omega_i^2 = \frac{1}{k_{PWM}} \cdot PWM'_i \tag{2.35}$$

These three constants are found experimentally as described in Appendix A.

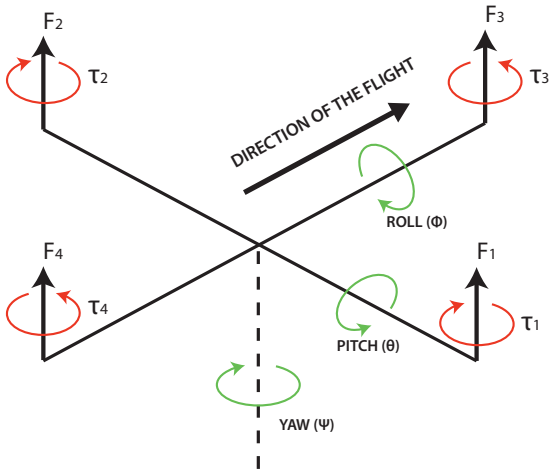

Figure 2.5: Representation of forces and torques acting on the quadrotor.

The forces affecting the quadcopter will be defined as follows.

- **Thrust Force** is the result of the sum of forces generated by each motor. In case the 4 of them are equal, there will be a vertical movement upwards or downwards depending on its total sum.

$$F_M = F_1 + F_2 + F_3 + F_4 \tag{2.36}$$

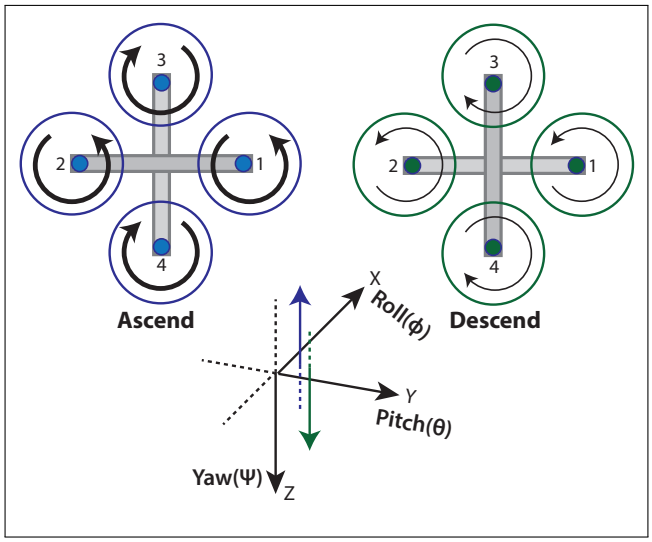

Figure 2.6: Throttle movement and its representation in the local frame considering all 4 forces are equal. Black arrows in the quadcopter motors represent the rotational direction of the propellers.

Its direction is always perpendicular to the propellers, so it will only have a z component in the local frame.

$$F_{motor_l} = \begin{pmatrix} 0 \\ 0 \\ F_M \end{pmatrix} \tag{2.37}$$

So It can be converted into global using the rotational matrix as previously defined,

$$F_{motor_g} = \mathbf{R}_{l-g}(\Theta) \begin{pmatrix} 0 \\ 0 \\ F_M \end{pmatrix} = \begin{pmatrix} (sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi))F_M \\ (-sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi))F_M \\ cos(\theta)cos(\phi)F_M \end{pmatrix} \tag{2.38}$$

- **Gravitational Force -** is the force affecting the mass of the quadcopter. Gravity is always perpendicular to the ground, so this force can be expressed directly on the global frame as

$$W_g = \begin{pmatrix} 0 \\ 0 \\ Mg \end{pmatrix} \tag{2.39}$$

where M is the total mass and g is the constant of gravity.

The total force in the global frame will be the summation of the these two opposite forces, such

that

$$F_g = W_g - F_{motor_g} = \begin{bmatrix} 0 \\ 0 \\ Mg \end{bmatrix} - \mathbf{R}_{l-g}(\Theta) \begin{bmatrix} 0 \\ 0 \\ F_M \end{bmatrix} = \begin{pmatrix} -(sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi))F_M \\ -(-sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi))F_M \\ Mg - cos(\theta)cos(\phi)F_M \end{pmatrix} \tag{2.40}$$

In addition, these 4 motor forces produce different torques to the quadcopter which affects the rotation dynamics in the local frame:

- **Roll** is generated by the torque of the left and right motors.

$$\tau_x = L(F_2 - F_1) \tag{2.41}$$

  where L is the arm length of the quadcopter. It will make the quadcopter move to the left or to the right depending on which force is larger, as illustrated in Fig. 2.7

  - If $f_2 > f_1$, the quadcopter will roll to the right.
  - If $f_2 < f_1$, then it will roll to the left.

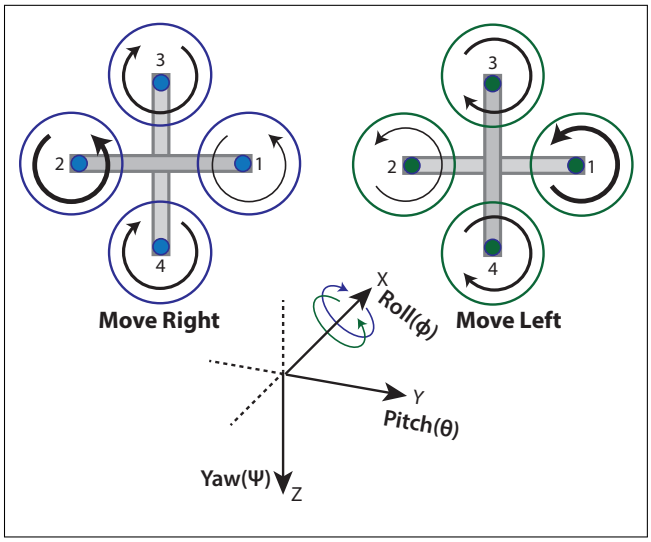

Figure 2.7: Roll movement and its representation in the local frame. Black arrows in the quadcopter motors represent the rotational direction of the propellers.

- **Pitch** is generated by the torque from the front and back motors.

$$\tau_y = L(F_3 - F_4) \tag{2.42}$$

  It will make the quadcopter move forward or backward depending on which force is bigger, as illustrated in Fig. 2.8

  - If $f_3 > f_4$, the quad-copter will move backwards.

– If $f_3 < f_4$, then it will move forwards.

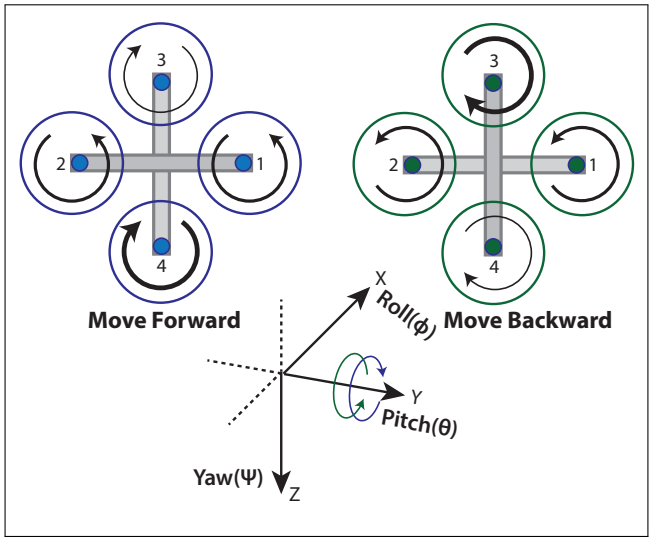

Figure 2.8: Pitch movement and its representation in the local frame. Black arrows in the quadcopter motors represent the rotational direction of the propellers.

- **Yaw** According to the third Law of Newton (Action-Reaction), the drag of the propeller generates a torque which is generated in the opposite direction of propeller rotation and makes the quadcopter spin around z axis.

$$\tau_z = -\tau_3 - \tau_4 + \tau_1 + \tau_2 \tag{2.43}$$

When the rotational speed of the front and back motor is increased and the left and right one is decreased, a movement counter-clockwise is produced. On the other hand, if the speed of the left and right motors is increased and the front and back one is decreased, the quadcopter will rotate clockwise, as illustrated on Fig. 2.9.

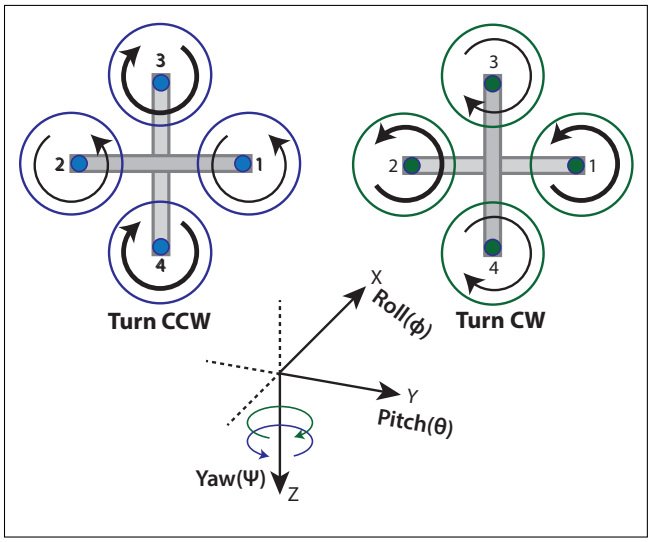

Figure 2.9: Yaw movement. Black arrows in the quadcopter motors represent the rotational direction of the propellers. Note that torques produced by the propellers are opposite to its rotation direction.

Lastly, considering that both forces, F_i, and torques, τ_i, are linearly related to the square of the angular speed of the blade, all the forces and torques will be represented as a function of this angular velocity Ω.

$$\begin{aligned} F_M &= K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \tau_x &= L \cdot K_F(\Omega_2^2 - \Omega_1^2) \\ \tau_y &= L \cdot K_F(\Omega_3^2 - \Omega_4^2) \\ \tau_z &= K_\tau(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \end{aligned} \tag{2.44}$$

2.6 Network

This section aims to describe the properties of the network communication for the quadcopters, specifically a wireless network. This kind of network allow the communication between devices using radio waves providing the quadcopter with enough mobility and still having a fast and reliable communication.
Despite these advantages,for the proper operation there is some aspects to consider when implementing a real-time communication through a wireless network since it could affect the performance of the controller in the quadcopter.

Real-Time Communication

A real-time system is a computer system in which the correctness of the system behaviour depends not only on the logical results of the computations, but also on the time when the results are produced. Real-time systems are usually in greatly connected to their physical environment. They receive data, process it, and return results in with certain time specifications [12].

Wireless Network Properties

A wireless network uses radio waves to transmit data from one module to another, where each module has an address. The basic operation of a wireless network consist on a computer module transforming the data in to a radio signal in order to be sent by an antenna (transmitter), this signal is received by another antenna module (receiver) with an specific address. Modules capable of both are called transceivers. During the transmission process, there are many factors, that can affect the connection, which should be considered to mitigate the problems they may cause.

Interferences

In contrast with other wired network, wireless networks can be affected by electromagnetic interferences (EMI). These interferences in a network are mostly generated by other networks or electronic devices. These disturbances may impact the connection from a small degradation of the transmission signal, to the total loss of the data packages.

Path loss

Path loss is defined as the attenuation of the power of electromagnetic waves along its propagation through space. Path loss is an important element to analyse, in the design of a wireless communication network, since it is influenced by many elements that will limit the network. To mention some of these parameters: the environment where the communication takes place, propagation mechanisms, the distance between transmitter and receiver [11] [5].Propagation mechanisms are known as the set of variations that a wave may be affected during the propagation process. Reflection, occurs when a wave impacts upon an object and changes the trajectory, this effect occurs particularly on metallic or conductive materials. On the contrary, when the wave impacts to other kind of material the effect caused is absorption, damping the

power of the wave and sometimes generating dead zones where the signal can not reach the receiver.

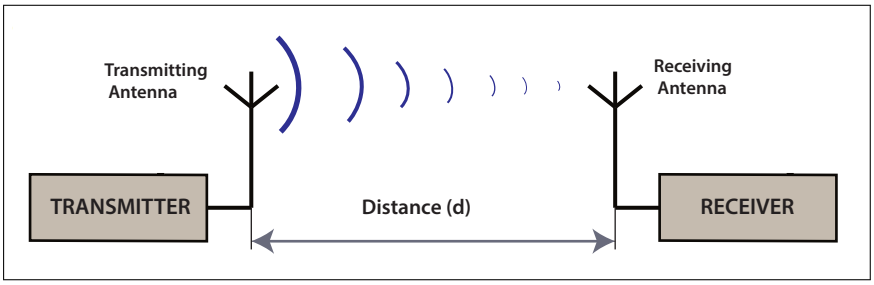

Figure 2.10: Path loss scheme

Fig. 2.10 illustrates a simple model of path loss. The power of the signal is reduced with the distance squared. Path loss function [21] is defined as:

$$P_{receiver} = \frac{1}{d^a} P_{transmitter} \tag{2.45}$$

Where P is the power, d is the distance in meters and a is the parameter chosen to model the environment, called the path-loss exponent. The path-loss exponent is $a = 2$ in an ideal free space and can reach values between 4 and 6 [23] when the transmission is taking place in an environment with many obstacles such as buildings or mountainous terrain, under circumstances where the exponent has values close to 6 the communication would be impractical.

2.7 Kalman Filtering

Sensor data in general contains noise, a method to reduce this noise is applying an estimation filter, such as a Kalman Filter (KF). An ordinary KF is a linear estimation, while the Extended Kalman Filter is the non-linear version.
A Kalman Filter runs recursively, taking in a stream of noisy sensor data and outputs variables that often is more precise than those picked up by the sensors itself.
While observer design can provide robustness to external disturbances, the KF explicitly incorporate a noise model, and are thus more appropriate and performs better in a stochastic system.
In short, the KF is an optimal estimator, in the sense that it minimizes the mean square error of the estimated parameters, if the noise is Gaussian. This report will not go into depth with the derivation of the KF, but merely sum it up for application.

Model

The KF will be based on the model [1]

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_k \\ z_k &= Cx_k \end{aligned} \tag{2.46}$$

where x_k, the signal value, can be found using the first equation which is a linear combination of its previous value, x_{k-1} and control value u_k. Additionally the measurement value z_k consists of an observation matrix and a signal vector.
Assuming the system in question satisfies this model, one only need to estimate the noise parameters, which will be explained later. It is important to note, that the better these noise parameters are estimated, the better the output estimates one will receive.

Process

The evaluation process is divided into two steps, with their respective equations. and both steps are applied in the k^{th} state.
where the parameters

1. Prediction Step - Time update

$$\begin{aligned} \text{State prediction} \quad & \bar{\hat{x}}_k = A\hat{x}_{k-1} + Bu_k \\ \text{Covariance prediction} \quad & \bar{P}_k = AP_{k-1}A^T + Q \end{aligned}$$

2. Correction Step - Measurement update

$$\begin{aligned} \text{Kalman Gain} \quad & K_k = \bar{P}_k H^T \left(C\bar{P}_k C^T + R\right)^{-1} \\ \text{State Update} \quad & \hat{x}_k = \bar{\hat{x}}_k + K_k(z_k - C\bar{\hat{x}}_k) \\ \text{Covariance Update} \quad & P_k = (I - K_k C)\bar{P}_k \end{aligned}$$

- $\hat{x}_k$ is the current estimation
- K_k is the Kalman gain
- P_k is a covariance giving the newest estimate of the average error for each part of the state
- C is an observation matrix, such that a state vector multiplied by the observation matrix, translates into a measurement vector.
- Q is the estimated process error covariance
- R is the estimated error covariance.

Often the $\bar{\hat{x}}_k$ variable is called the *prior estimate* and $\bar{P}_k$ the *prior error covariance*, while $\hat{x}_k$ and P_k are called *posterior* values. The algorithm will be executed as illustrated in Fig. 2.11.

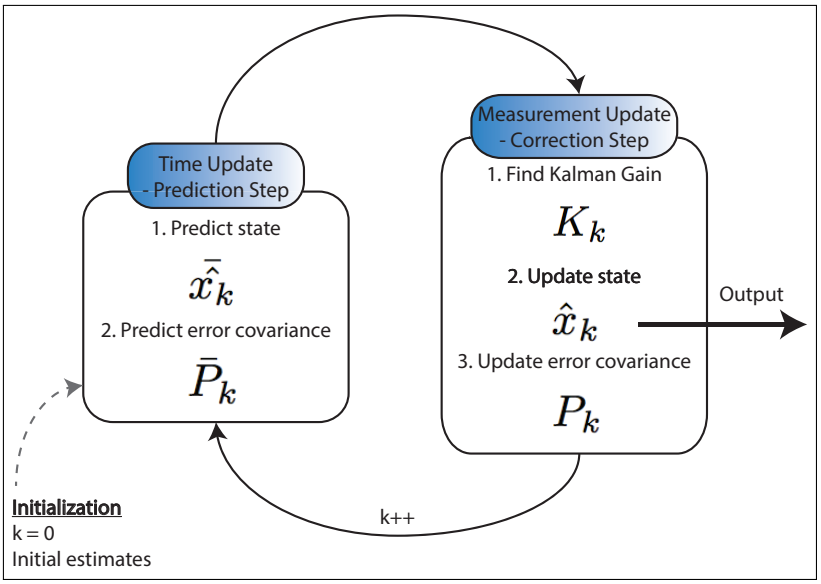

Figure 2.11: Kalman Filter algorithm.

2.7.1 Altitude & Vertical Velocity Estimation

In this project the system state variables, as previously described, are filtered and estimated using an Extended Kalman Filter (EKF). However, as the topic of EKF is outside the scope of this project, the existing algorithm that is already implemented on the platform, is applied. Instead a KF will be designed for estimating altitude and velocity in the z-direction.

Model representation

In this project some of the system state variables are filtered and estimated using an Extended Kalman Filter (EKF). However, as the topic of EKF is outside the scope of this project, the existing algorithm that is already implemented on the platform, is applied.
In respect to altitude and velocity in the z-direction, a KF will be designed for estimating them. Using the physics equation below

$$\begin{aligned} v_k &= v_{k-1} + a_{k-1} dt \\ h_k &= h_{k-1} + v_{k-1} dt + a_{k-1} dt^2 2 \end{aligned} \tag{2.47}$$

where dt is the sample period, one can design the system. The system state variables are

$$x = \begin{bmatrix} h \\ v \\ a \end{bmatrix} \qquad A = \begin{bmatrix} 1 & dt & \frac{dt^2}{2} \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix} \tag{2.48}$$

where the states are relative altitude, vertical velocity and acceleration, respectively. Expressed as in Eq. 2.46

$$x_k = Ax_{k-1} + Bu_k \tag{2.49}$$

$$\begin{bmatrix} h_k \\ v_k \\ a_k \end{bmatrix} = \begin{bmatrix} 1 & dt & \frac{dt^2}{2} \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_{k-1} \\ v_{k-1} \\ a_{k-1} \end{bmatrix} \tag{2.50}$$

and

$$y = Cx_{k-1}$$

$$\begin{bmatrix} h \\ a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_{k-1} \\ v_{k-1} \\ a_{k-1} \end{bmatrix} \tag{2.51}$$

Where there is no control matrix, and for $dt \neq 0$ the system is observable. ««««< .mine Incidentally, the model would be ready for implementation. Although, according to [17] the Kalman Filter will be very noisy when estimating the vertical velocity, due to tilting (roll and pitch). A proposed algorithm, is explained in [17], in order to compensate for tilt acceleration. This will however not be implemented in this project.

Vertical Acceleration Calculation

As the acceleration in the z direction is given in the local frame, it needs to be transformed into the global frame. This is done, using a rotation matrix, as described in the beginning of the theory chapter. The calculation can be computed as

$$a_z = R_{l-g}(\Phi)a_l \tag{2.52}$$

Consequently, the Euler angles from the EKF are needed. Additionally, the gravity affects the acceleration in the global frame, and can thus be subtracted as an offset.

$$a_{z,corr} = a_z - \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T \tag{2.53}$$

Altitude Calculation

The altitude state of the KF is measured through a barometer. As it returns pressure and temperature measurements, one have to calculate the altitude from such measurements.
As it is possible to calculate pressure from

$$p = p_0 \left(1 - \frac{Lh}{T_0}\right)^{gM/RL} \tag{2.54}$$

one can solve for the height, which is the absolute height above sea level.

$$h = -\left(\frac{p^{RL/gM}}{p_0} - 1\right)\frac{T_0}{L} \tag{2.55}$$

where the variables are

- p - atmospheric pressure at defined level
- p_0 - sea level standard atmospheric pressure
- L - temperature lapse rate
- T_0 - sea level standard temperature
- g - Earth's gravity acceleration constant
- M - molar mass of dry air
- R - Universal gas constant

It is worth noting that the temperature readings can cause problems, due to self-heating capabilities. However this stabilizes to a constant level and is compensated for in the driver settings. Finally one can find the relative altitude during flight as [17]

$$h_r = h_{abs} - h_0 \tag{2.56}$$

where h_0 is the initial reading in idle position and h_{abs} can be found as explained in Eq. 2.55. The altitude calculation is already implemented in the Pixhawk open source and can be read directly.

Noise Parameters

The noise parameters applied in the KF, is as follows

- The estimated process error covariance matrix, Q, is set to zero, as it is outside the scope of this project.
- The estimated sensor error covariance matrix R, is calculated in the idle state during initialization.
- The initial covariance P_0 of the states error is set to a diagonal matrix with high values, as this indicates high uncertainty. This matrix will eventually converge.

2.7.2 Estimation Algorithm

The overall algorithm will be carried out as follows

1. Estimation of attitude (given from EKF)
2. Measure pressure, vertical acceleration and temperature
3. Compute relative altitude, h_r
4. Altitude and vertical velocity estimation with the KF

Fig. 2.12 illustrates how the algorithm is carried out on the platform as in series to the EKF. Note that in this example, the estimates are not used in the control algorithm. This will applied in future work, when the quadcopter is not restricted to inside applications.

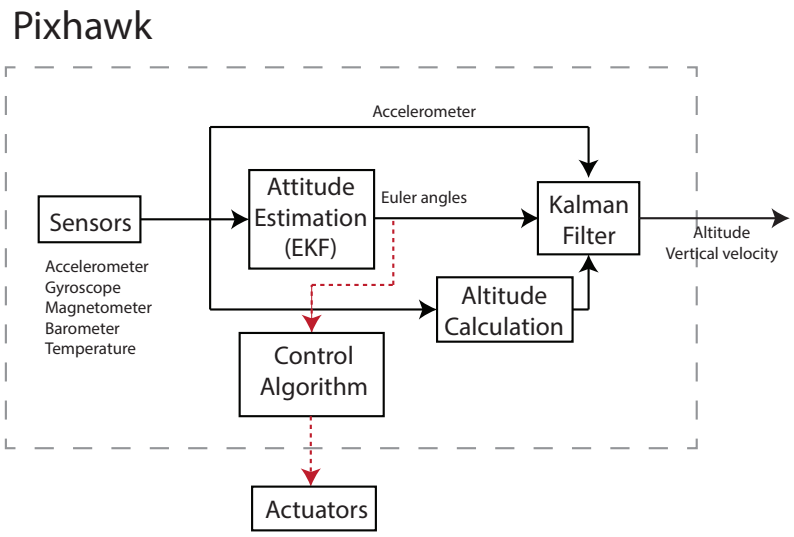

Figure 2.12: Implementation algorithm.

CHAPTER 3

MODELLING

The model used in this project is developed in the global frame as the desired quadcopter position is in this frame.

3.1 Full model

After the dynamics and kinematics study, Eq. 2.21 and 2.22 are applied, in order to develop the full model of the quadcopter. As $\ddot{P} = \dot{v}$ and $\ddot{\Phi} = \dot{\omega}_\Phi$ it will be expressed as first order derivatives in the equations. As such, the forces in the global frame and torques in the local, described in Sec. 2.5.1.2, are introduced.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix}_g = \frac{1}{M} \begin{bmatrix} -(sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi))F_M \\ (sin(\phi)cos(\psi) - cos(\phi)sin(\theta)sin(\psi))F_M \\ Mg - cos(\theta)cos(\phi)F_M \end{bmatrix} + \begin{bmatrix} \omega_\psi v_y - \omega_\theta v_z \\ \omega_\phi v_z - \omega_\psi v_x \\ \omega_\theta v_x - \omega_\phi v_y \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}_l = \begin{bmatrix} \frac{1}{I_{xx}} L(F_2 - F_1) \\ \frac{1}{I_{yy}} L(F_3 - F_4) \\ \frac{1}{I_{zz}}(-\tau_3 - \tau_4 + \tau_1 + \tau_2) \end{bmatrix} + \begin{bmatrix} \omega_y \omega_z (I_{zz} - I_{yy}) \\ \omega_x \omega_z (I_{xx} - I_{zz}) \\ \omega_y \omega_x (I_{yy} - I_{xx}) \end{bmatrix} \tag{3.2}$$

As It is desired to control the Euler angles, another equation is used based on the conversion from one to the other frame defined in Eq. 2.16:

$$\dot{\Phi} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix}_g = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_l \tag{3.3}$$

The relationship between position and velocity in the global frame is defined:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_g = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_g \tag{3.4}$$

Now, both position and orientation in the global frame are defined throughout these four equations which are used to build a Simulink model for the quadcopter.

3.1.1 Full Model in Simulink

The implementation of the full model in Simulink is illustrated in Fig. 3.1, in a simplified version. All the blocks will be explained in more detail below.

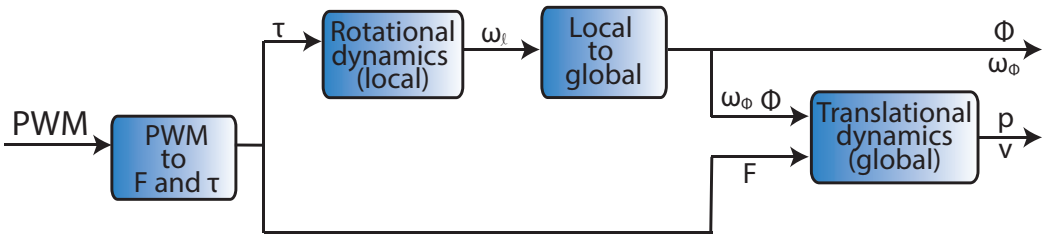

Figure 3.1: Scheme of the full model.

Incidentally, in the first block the forces and torques, which was described in Sec. 2.5.1.2, are calculated from the four PWM signals, that drives the four motors. Additionally, the constants of moments of inertia are computed.

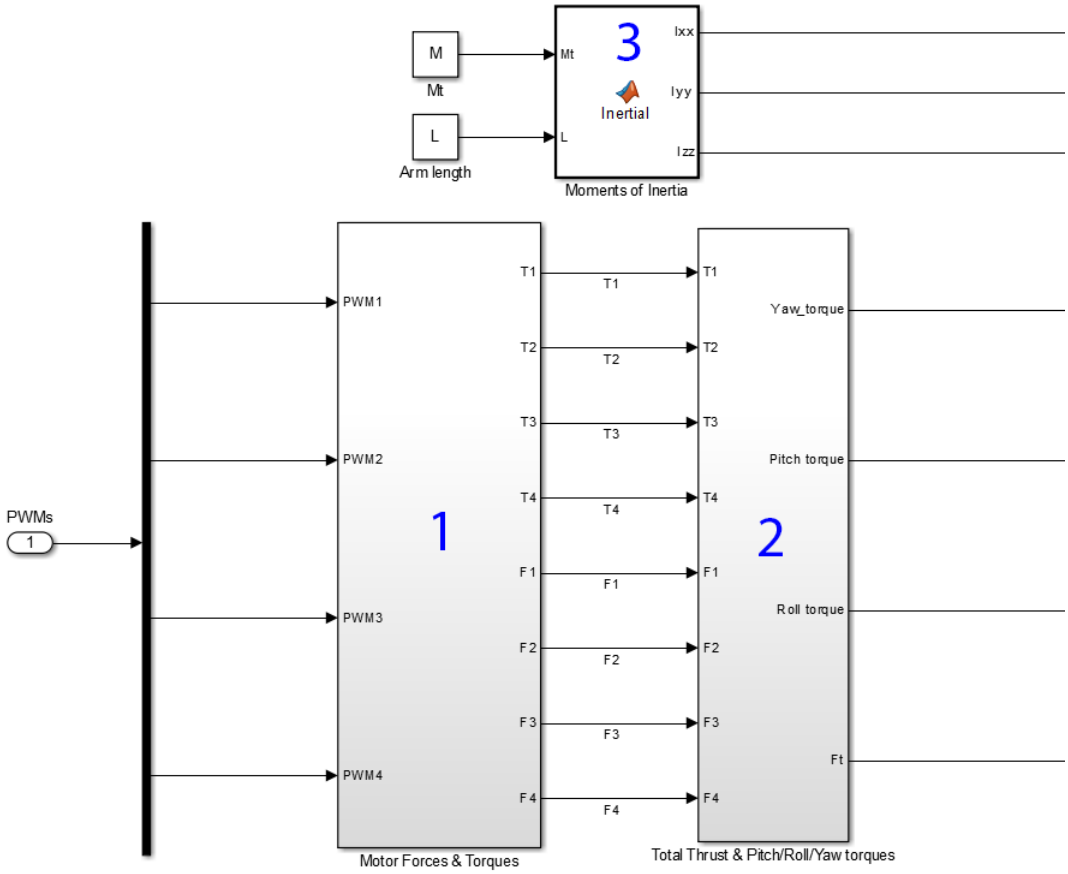

Figure 3.2: Forces, torques and moments of inertia in Simulink.

Fig. 3.2 illustrates the data flow, and and the blocks computes the following:

1) converts PWM signal to motor forces and torques using Eq. 2.31, Eq. 2.32 and Eq. 2.33.

2) computes the torques in the local frame defined in Eq. 2.41, Eq. 2.42 and Eq. 2.43. Also the total thrust defined in Eq. 2.36.

3) calculates the moments of inertia found in Eq. 2.29.

Now, angular acceleration in the local frame from Eq. 3.2 is computed by taking the outputs shown in Fig. 3.2:

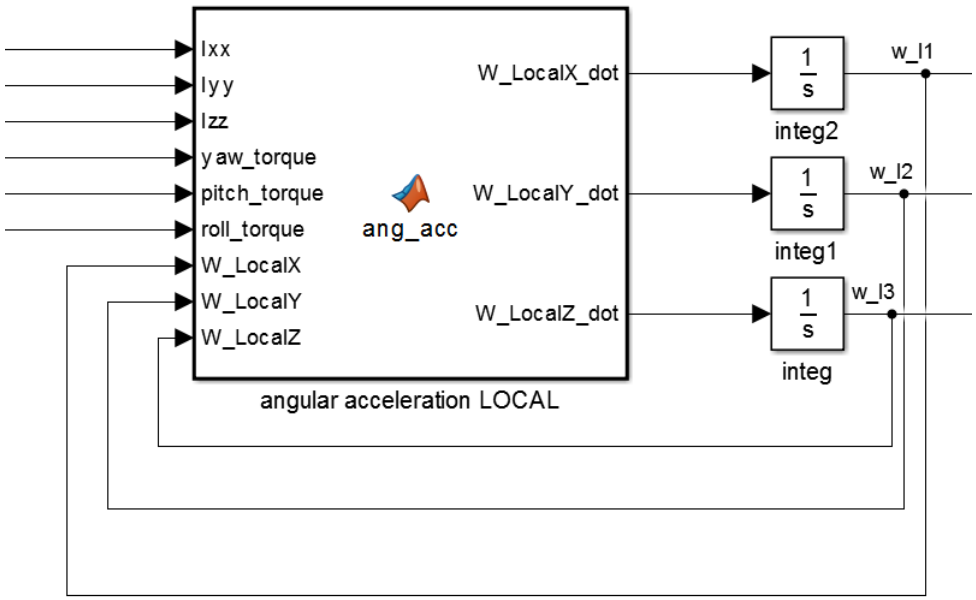

Figure 3.3: Angular acceleration, velocity and Euler angles in the local frame in Simulink.

Next step is to calculate the angular velocity in the global frame and the Euler angles as defined in Eq. 3.3 using the local angular velocity output, in Fig. 3.3:

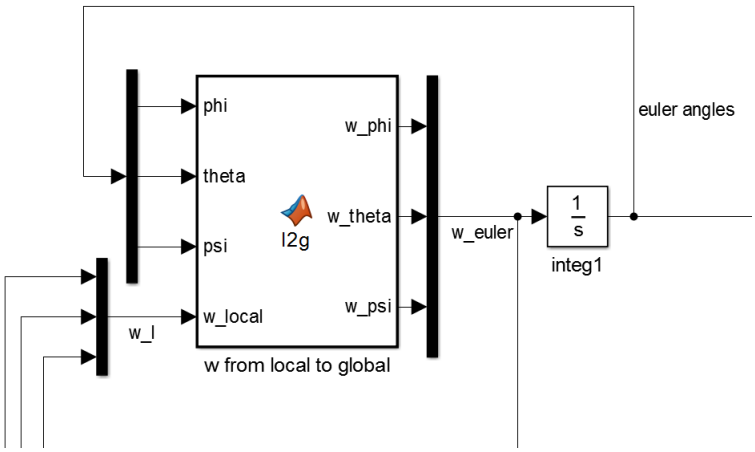

Figure 3.4: Angular velocity and Euler angles in the global frame in Simulink.

Finally, Eq. 3.1 and Eq. 3.4 are introduced in the simulation in order to find position in the global frame. A saturation block is used to avoid the z position going below zero in the simu-

lation. The total thrust found in Fig. 3.2, weight of the quadcopter, Euler angles and angular accelerations found in Fig. 3.3 are used as input for the calculation in Fig. 3.5

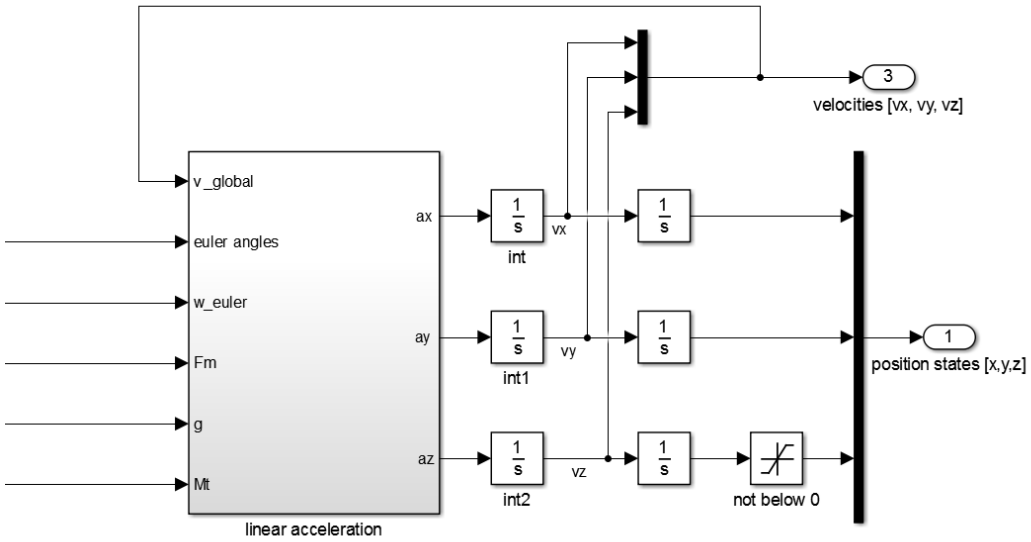

Figure 3.5: Linear acceleration, velocity and position in the global frame in Simulink.

These four parts of the simulink file are put together as shown in Fig. 3.6.

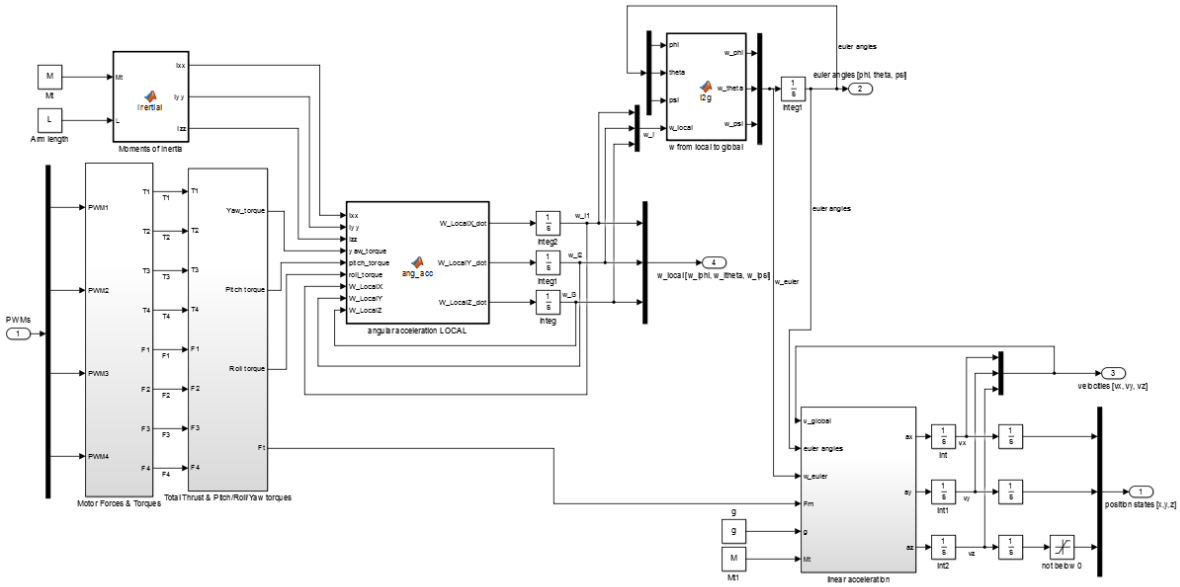

Figure 3.6: Full model in Simulink.

3.2 Simple model

The next step is to construct a simple model in order to design a controller for the quadcopter. Certain assumptions are made to achieve such, and is listed below.

- Initially, the quadcopter is assumed to move slow enough so the second term from both Eq. 3.1 and 3.2, which is the Coriolis effect, is neglected.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix}_g = \frac{1}{M} \begin{bmatrix} -(sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi))F_M \\ (sin(\phi)cos(\psi) - cos(\phi)sin(\theta)sin(\psi))F_M \\ Mg - cos(\theta)cos(\phi)F_M \end{bmatrix} \tag{3.5}$$

  and

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}_l = \begin{bmatrix} \frac{1}{I_{xx}} L(F_2 - F_1) \\ \frac{1}{I_{yy}} L(F_3 - F_4) \\ \frac{1}{I_{zz}}(-\tau_3 - \tau_4 + \tau_1 + \tau_2) \end{bmatrix} \tag{3.6}$$

- Both pitch (θ) and roll (ϕ) angles are considered to be very small and close to 0 during the flight. Consequently, the sine and cosine term from the linear acceleration equations can be linearised using the trigonometrical small-angle approximation. This is valid for sine when $\alpha < 14°$ and cosine when $\alpha < 38°$, as the approximation error at these angles is only 1%.

$$\sin\alpha \approx \alpha \quad \cos\alpha \approx 1 \tag{3.7}$$

- Furthermore, yaw angle (ψ) must be controlled and set to 0 as it simplifies the model, so $sin(\psi) = 0$ and $cos(\psi) = 1$. Now, the linear acceleration results in Eq. 3.8:

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix}_g \approx \begin{bmatrix} \frac{-F_M\theta}{M} \\ \frac{F_M\phi}{M} \\ g - \frac{1}{M}F_M \end{bmatrix} \tag{3.8}$$

- As $\dot{v}_x$ and $\dot{v}_y$ are not linear yet because they are defined by the multiplication of an input and a state (F_M with θ and ϕ), F_M will be considered to be constant and equal to the minimum force needed to hover as it is state the quadcopter will be in, during flight and there will not be many fluctuations. $\dot{v}_z$ will not be modified:

$$F_M = F_{hover} = Mg \tag{3.9}$$

- Now, Forces F_i and Torques τ_i will be expressed as a function of the propellers angular velocity as stated in Eq. 2.44. Also, $F_{hover} = Mg$ is put into the equation.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix}_g \approx \begin{bmatrix} -g\theta \\ g\phi \\ g - \frac{1}{M}K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{3.10}$$

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}_l = \begin{bmatrix} \frac{LK_F}{I_{xx}}(\Omega_2^2 - \Omega_1^2) \\ \frac{LK_F}{I_{yy}}(\Omega_3^2 - \Omega_4^2) \\ \frac{K_\tau}{I_{zz}}(-\Omega_3^2 - \Omega_4^2 + \Omega_1^2 + \Omega_2^2) \end{bmatrix} \tag{3.11}$$

- Then, the relationship between PWM and Ω^2 from Eq. 2.34 is used in order to have the PWM signal as input.

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \end{bmatrix}_g \approx \begin{bmatrix} -g\theta \\ g\phi \\ g - \frac{K_F}{M \cdot K_{PWM}}(PWM_1^2 + PWM_2^2 + PWM_3^2 + PWM_4^2) \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}_l = \begin{bmatrix} \frac{L \cdot K_F}{I_{xx} \cdot K_{PWM}}(PWM_2^2 - PWM_1^2) \\ \frac{L \cdot K_F}{I_{yy} \cdot K_{PWM}}(PWM_3^2 - PWM_4^2) \\ \frac{K_\tau}{I_{zz} \cdot K_{PWM}}(-PWM_3^2 - PWM_4^2 + PWM_1^2 + PWM_2^2) \end{bmatrix} \tag{3.13}$$

- Finally, as Eq. 3.3 is not linear yet, angular velocity is considered to be equal in both local and global frame as the deviation in the Euler angles during steady flight produced in the quadcopter movement are very small:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix}_g \approx \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_l \tag{3.14}$$

After simplifying the model, two approaches are done in order to design a proper controller which will be designed using Matlab and simulated in the full model in Simulink.

3.2.1 Single Input approach

First of all, from Eq. 3.10 and 3.11 four new input signals for thrust, pitch, roll and yaw are created that will be used to control $[x, y, z, \psi]^T$.

$$\begin{aligned} U_1 &= \Omega_2^2 - \Omega_1^2 \\ U_2 &= \Omega_3^2 - \Omega_4^2 \\ U_3 &= -\Omega_3^2 - \Omega_4^2 + \Omega_1^2 + \Omega_2^2 \\ U_4 &= \Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2 \end{aligned} \tag{3.15}$$

which can also be expressed in a matrix form:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{3.16}$$

After this consideration, there are 6 differential equations that can be expressed as 4 different systems with individual input signals:

$$\begin{aligned} \ddot{x} &= -g\theta \\ \ddot{y} &= g\phi \\ \ddot{z} &= g - \tfrac{K_F}{M} U_4 \\ \ddot{\phi} &= \tfrac{L \cdot K_F}{I_{xx}} \cdot U_1 \\ \ddot{\theta} &= \tfrac{L \cdot K_F}{I_{yy}} \cdot U_2 \\ \ddot{\psi} &= \tfrac{K_\tau}{I_{zz}} \cdot U_3 \end{aligned} \tag{3.17}$$

After this assumption, model for hover, x, y and yaw are described separately:

- **Hover**: z acceleration equation is taken and then Laplace transform is used in order to obtain transfer function.

$$\ddot{z} = g - \frac{K_F}{M} U_4 \tag{3.18}$$

$$\frac{Z(s)}{U_4(s)} = -\frac{K_F}{Ms^2} \tag{3.19}$$

  It can also be expressed as a state space model if $\dot{z} = v_z$ is assumed:

$$\begin{bmatrix} \dot{z} \\ \dot{v_z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ v_z \end{bmatrix} - \begin{bmatrix} 0 \\ \frac{K_F}{Ms^2} \end{bmatrix} U_4 + \begin{bmatrix} 0 \\ g \end{bmatrix} \tag{3.20}$$

- **x:** equations of θ and x acceleration are taken and also transformed into the Laplace domain.

$$\begin{aligned} \ddot{x} &= -g\theta \\ \ddot{\theta} &= \tfrac{LK_F}{I_{yy}} \cdot U_2 \end{aligned} \tag{3.21}$$

$$\begin{aligned} \tfrac{X(s)}{\theta(s)} &= -\tfrac{g}{s^2} \\ \tfrac{\theta(s)}{U_2(s)} &= \tfrac{LK_F}{I_{yy}s^2} \end{aligned} \tag{3.22}$$

  Then, multiplying these two, the transfer function for x is obtained:

$$\frac{X(s)}{U_2(s)} = \frac{X(s)}{\theta(s)} \cdot \frac{\theta(s)}{U_2(s)} = -\frac{LK_F g}{I_{yy}s^4} \tag{3.23}$$

  Considering that $[\dot{x}, \dot{\theta}]^T = [v_x, \omega_\theta]^T$, a state space model to control x is found:

$$\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \dot{v_x} \\ \dot{\omega_\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \theta \\ v_z \\ \omega_\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{LK_F}{I_{yy}s^2} \end{bmatrix} U_2 \tag{3.24}$$

- **y:** Equations of ϕ and y acceleration are taken and also transformed into the Laplace domain.

$$\begin{aligned} \ddot{y} &= \phi g \\ \ddot{\phi} &= \frac{LK_F}{I_{xx}} \cdot U_1 \end{aligned} \tag{3.25}$$

$$\begin{aligned} \frac{Y(s)}{\phi(s)} &= \frac{g}{s^2} \\ \frac{\phi(s)}{U_1(s)} &= \frac{LK_F}{I_{xx}s^2} \end{aligned} \tag{3.26}$$

Then, multiplying these two, the transfer function for x is obtained:

$$\frac{Y(s)}{U_1(s)} = \frac{Y(s)}{\phi(s)} \cdot \frac{\phi(s)}{U_1(s)} = \frac{LK_F g}{I_{xx}s^4} \tag{3.27}$$

Considering that $[\dot{y}, \dot{\phi}]^T = [v_y, \omega_\phi]^T$, a state space model to control x is found:

$$\begin{bmatrix} \dot{y} \\ \dot{\phi} \\ \dot{v_y} \\ \dot{\omega_\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \phi \\ v_y \\ \omega_\phi \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{LK_F}{I_{xx}s^2} \end{bmatrix} U_1 \tag{3.28}$$

- **Yaw:** ψ acceleration equation is taken and then Laplace transform is used in order to obtain transfer function.

$$\ddot{\psi} = \frac{K_\tau}{I_{zz}} U_3 \tag{3.29}$$

$$\frac{\psi(s)}{U_3(s)} = \frac{K_\tau}{I_{zz} \cdot s^2} \tag{3.30}$$

It can also be expressed as a state space model if $\dot{\psi} = \omega_\psi$ is assumed:

$$\begin{bmatrix} \dot{\psi} \\ \dot{\omega_\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi \\ \omega_\psi \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_\tau}{I_{zz}s^2} \end{bmatrix} U_3 \tag{3.31}$$

3.2.2 MIMO approach

The differential linearised equations described along Sec. 3.2 can also be expressed as a system with Multiple Inputs and Multiple Outputs (MIMO) where Ω^2 of the 4 motors are the inputs:

$$\begin{aligned} \dot{v_x} &= -\theta g \\ \dot{v_y} &= \phi g \\ \dot{v_z} &= g - \tfrac{1}{M}(K_F(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)) \\ \dot{\omega_\phi} &= \tfrac{1}{I_{xx}} L \cdot K_F(\Omega_2^2 - \Omega_1^2) \\ \dot{\omega_\theta} &= \tfrac{1}{I_{yy}} L \cdot K_F(\Omega_3^2 - \Omega_4^2) \\ \dot{\omega_\psi} &= \tfrac{1}{I_{zz}} K_\tau(-\Omega_3^2 - \Omega_4^2 + \Omega_1^2 + \Omega_2^2) \end{aligned} \tag{3.32}$$

Now, if $[\dot{x},\dot{y},\dot{z}]^T = [v_x, v_y, v_z]^T$ and $[\dot{\phi},\dot{\theta},\dot{\psi}]^T = [\omega_\phi, \omega_\theta, \omega_\psi]^T$, Eq. 3.32 can be written as a state space representation with a total of 12 states:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{\omega_\phi} \\ \dot{\omega_\theta} \\ \dot{\omega_\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ v_x \\ v_y \\ v_z \\ \omega_\phi \\ \omega_\theta \\ \omega_\psi \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{K_F}{M} & -\frac{K_F}{M} & -\frac{K_F}{M} & -\frac{K_F}{M} \\ \frac{-LK_F}{I_{xx}} & \frac{LK_F}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{LK_F}{I_{yy}} & \frac{-LK_F}{I_{yy}} \\ \frac{K_\tau}{I_{zz}} & \frac{K_\tau}{I_{zz}} & \frac{-K_\tau}{I_{zz}} & \frac{-K_\tau}{I_{zz}} \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ g \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.33}$$

Which can be illustrated as the diagram in Fig. 3.7.

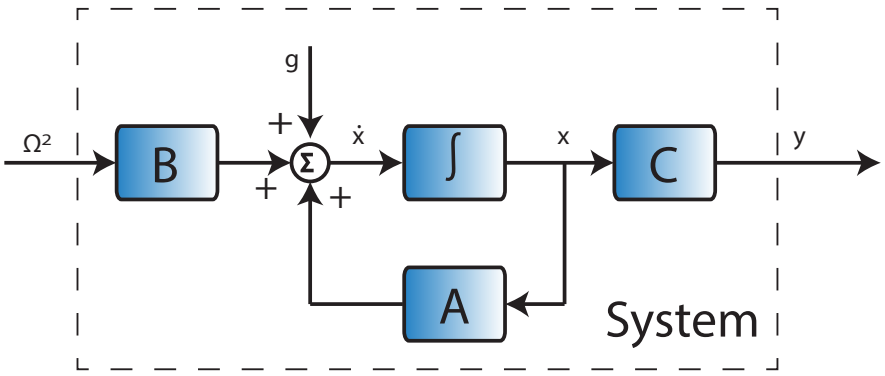

Figure 3.7: State Space Block Diagram.

The 12x12 matrix is A and the 12x4 matrix is B. Having a state space representation can ease the controller design, and will be applied in the control design.

Part III

Design

CHAPTER 4

CONTROLLER

In this section, the controllers suggested for the quadcopter are developed and explained. Initially, the four systems for x, y, z and ψ from Sec. 3.2.1 are intended to be controlled using classical control theory. Then, state space integral controllers are suggested for the state space model of these four systems. Finally, a LQR controller using the MIMO state space from Eq.3.33 is used.

4.1 Classical Control Theory

The classical control theory is sometimes the first and easy approach to control any system. As a single input and single output are needed, Eq. 3.19, 3.23, 3.27 and 3.30 for the x, y, z and ψ movement of the quadcopter are taken where $[U_1, U_2, U_3, U_4]^T$ will be the corresponding input signal.
The transfer function for hover and yaw are second order systems so a PID or PD is expected to work fine with them. In case of hover, a PID is used as the integral part can minimize the effect of gravity. On the other hand, a PD is sufficiently good for yaw. The form of these controllers are defined as [10]:

$$\begin{aligned} C(s) = PID(s) = K_p + sK_d + \tfrac{1}{s}K_i \\ C(s) = PD(s) = K_p + sK_d \end{aligned} \tag{4.1}$$

In respect to x and y, it has not been possible to find a working PI, PD or PID controller for them. It may be so because they are fourth order systems. An alternative to control them is given in Sec. 4.2 using the state space form of these two systems.

4.1.1 Classical Control in Simulink

The two working controllers are initially tuned with the linearised transfer function using Simulink. A PID is modelled as a block represented in Fig. 4.1 which will also be used for the PD controller.

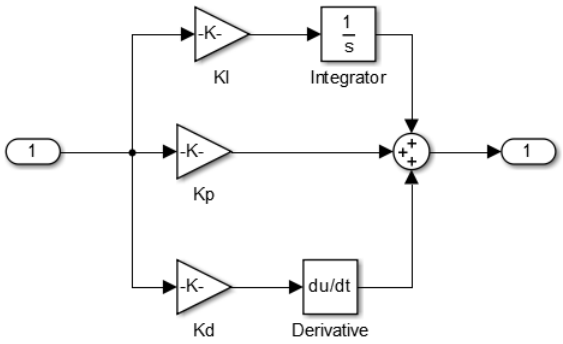

Figure 4.1: PID block created and used in Simulink.

Then, the control signals $[U_1, U_2, U_3, U_4]^T$ are coupled into $[PWM_1^2, PWM_2^2, PWM_3^2, PWM_4^2]^T$ and put in the non-linear model described in Sec. 3.1.1. These controllers are tested and a final manual tuning is done. Fig. 4.2 shows the controllers and the full model in simulink.

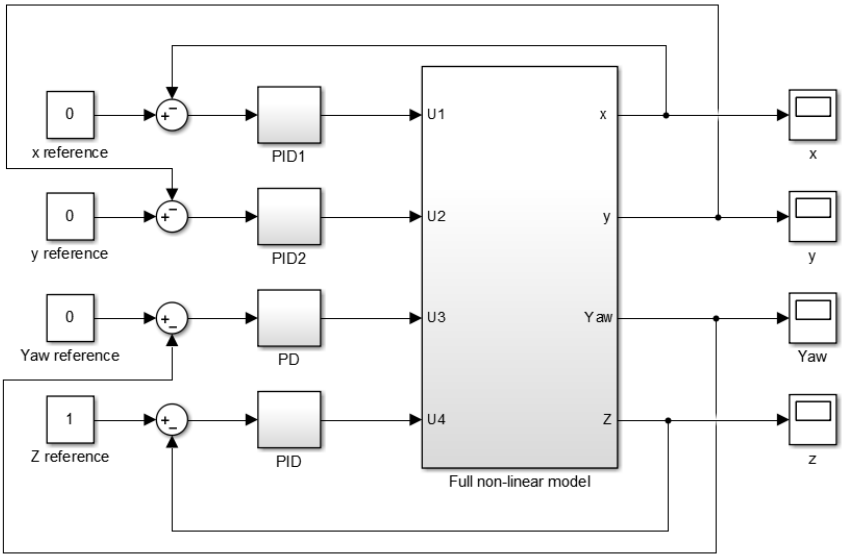

Figure 4.2: Simulink layout of the hover and yaw control.

Results in the full model verifies the hover and yaw controllers as shown in Fig. 4.3. The x and y were intended to be controlled, however results turned out to be unsuccessful, thus this method will not be investigated further.

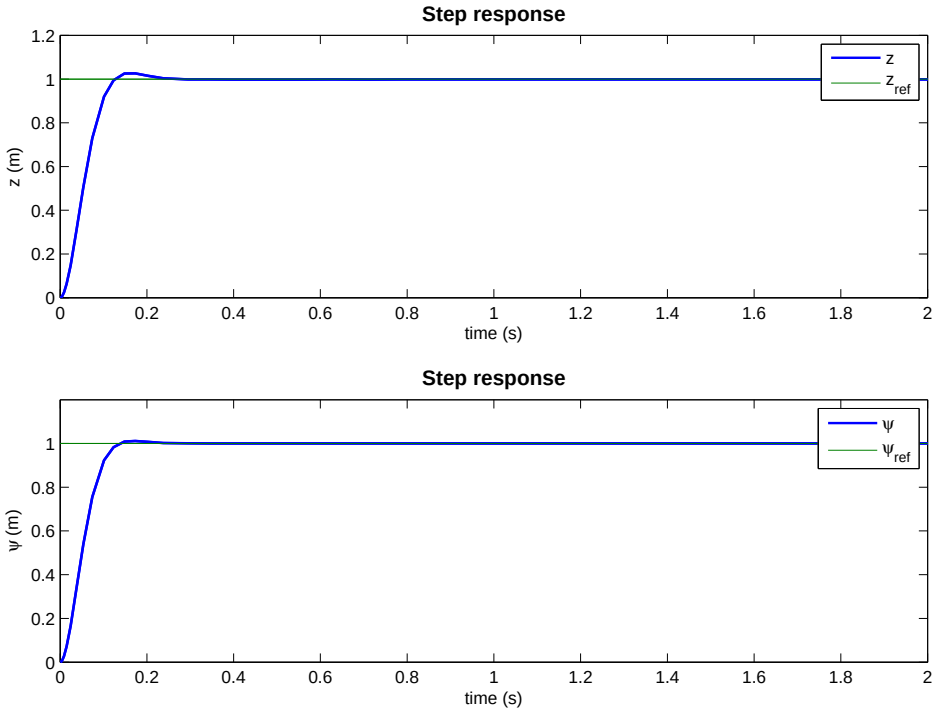

Figure 4.3: Step response .

4.2 Integral Control

As x and y could not be controlled through classical control, the four single input systems described in Sec. 3.2.1 are intended to be controlled using a state space integral control. The respective state space models found in Eq. 3.20, 3.24, 3.28 and 3.31 are used in order to control x, y, z and ψ. These controllers are studied in both the simplified and full model.

An integral control is introduced when a state feedback controller does not reject the effects of disturbances as it happens with the gravity. Likewise, the integral term will ensure unitary static gain. In other words, the response of the system will reach the desired position in steady-state. For convenience, the system error is defined as $e = y - r$ and the objective is to let $y - r$ reach zero.
When introducing the integral control, the feedback design is modified to [10]

$$u(t) = Fx(t) + F_I x_I(t) \tag{4.2}$$

where $x_I = \int_0^t y(\tau) - r(\tau) d\tau$, such that $\dot{x}_I = y(t) - r(t)$.

Consequently, the state space models can now be summarized to

$$\dot{x} = Ax + Bu$$
$$\dot{x}_I = y - r$$
$$y = Cx$$

or alternatively

$$\begin{bmatrix} \dot{x} \\ \dot{x}_I \end{bmatrix} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ x_I \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ -I \end{bmatrix} r \tag{4.3}$$

$$y = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x \\ x_I \end{bmatrix} \tag{4.4}$$

where

$$u = Fx + F_I x_I = \begin{bmatrix} F & F_I \end{bmatrix} \begin{bmatrix} x \\ x_I \end{bmatrix} \tag{4.5}$$

Now, this can be written as a conventional state feedback case, as

$$x_e = A_e x_e + B_e u$$
$$y = C_e x_e \tag{4.6}$$

for which the state feedback $u = F_e x_e$ of each system is designed using pole placement in Matlab. Where

$$F_e = \begin{bmatrix} F & F_I \end{bmatrix}, \quad x_e = \begin{bmatrix} x \\ x_I \end{bmatrix}$$
$$A_e = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}, \quad B_e = \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad C_e = \begin{bmatrix} C & 0 \end{bmatrix} \tag{4.7}$$

This system flow that will be used has been depicted in Fig. 4.4

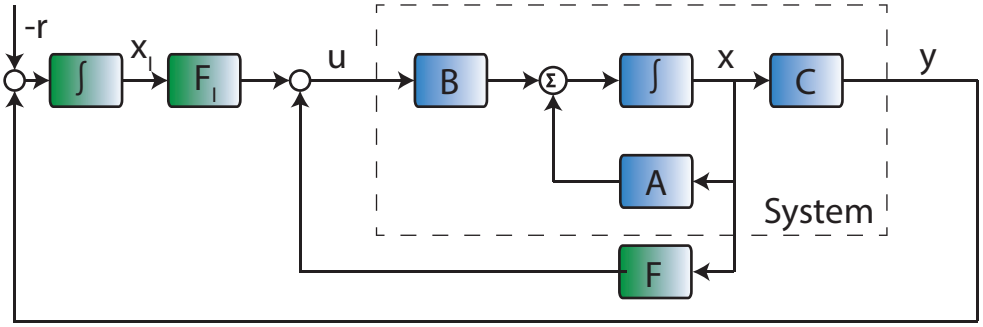

Figure 4.4: Integral Control block diagram.

4.2.1 Results

After designing the four different controller using pole placement, they are analysed separately with the state space systems, which are linearised and do not have cross effects, as shown in Fig. 4.5.

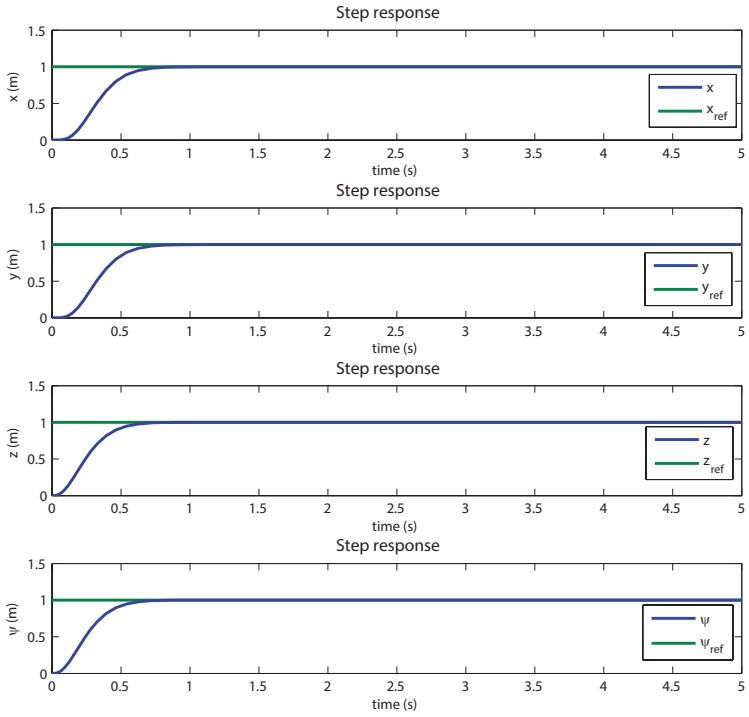

Figure 4.5: Step response of the four independent systems for x, y, z and yaw.

When the full model is introduced, the four controllers are coupled and tested. Unfortunately, as pitch and roll are not controlled with this method, the movement in the x and y can not be stable because the linearisation condition of the model are not assured, as both pitch and roll may increase too much during movement in x and y. So only yaw and z can be really controlled with this approach, whereas controlling x and y resulted the full model to unstable. As it is done in the classical control approach, both hover and yaw control, which are working, are shown in Fig. 4.6.

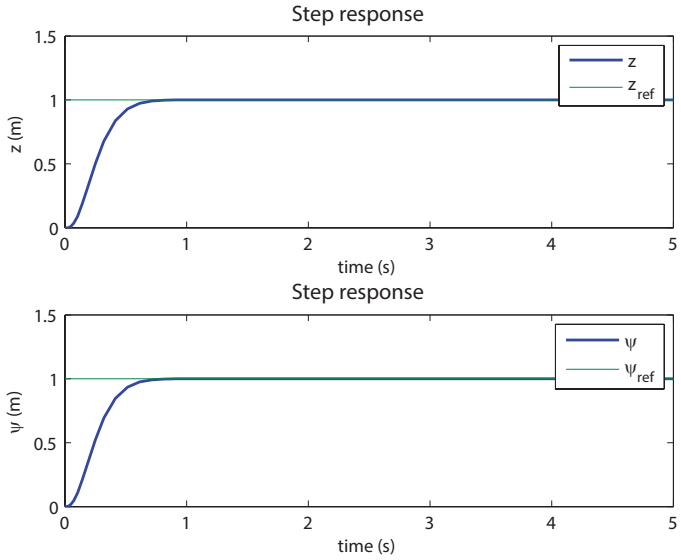

Figure 4.6: Step response of z and yaw in the full model.

No further work to correct this problem is done as it is preferred to have a single LQR controller, where multiple states can be controlled at once, instead of 4 individual ones. The location of the poles in the systems after tuning the controllers, are illustrated in Fig. 4.7 and Fig. 4.8.

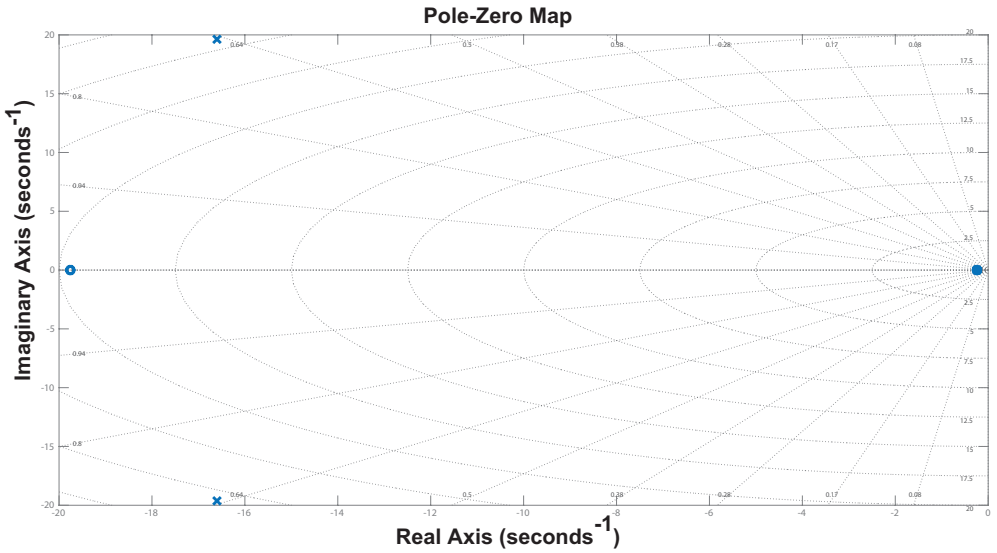

Figure 4.7: Pole locations of the system with hover control.

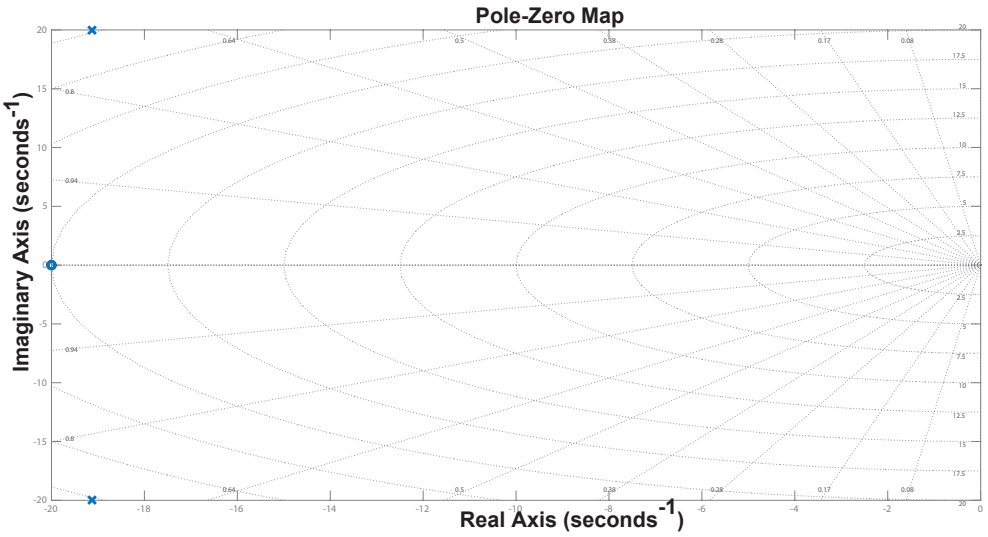

Figure 4.8: Pole and zero map of the system with yaw control.

4.3 LQR Control

A controller for a system with multiple inputs and multiple outputs is applied as it may solve the problems that appeared when coupling the quadcopter of four different systems into one.

In order to find a Linear Quadratic Regulator, state space model of the form $\dot{x} = Ax + Bu$ from Eq. 3.33 is taken. Gravity effect is not considered in the controller design and initially it is assumed that all states are available. A feedback gain K_d will be designed and implemented as $u[n] = -K_d(x[n] - x_{ref}[n])$, which is represented in Fig. 4.9, for the discrete-time state space model [14]:

$$x[n+1] = A_d x[n] + B_d u[n] \tag{4.8}$$

where K_d is a matrix with as many rows as inputs (4) and as many columns as states (12). Besides, A_d and B_d are the discrete matrices of the plant.

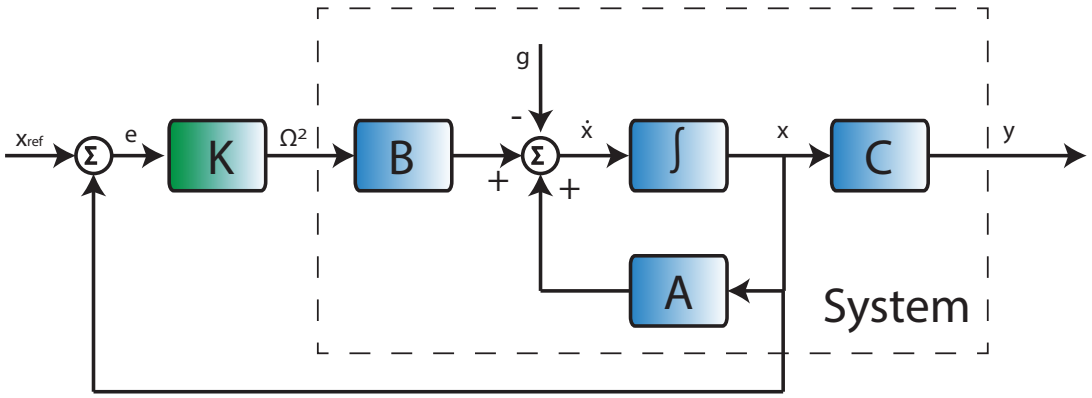

Figure 4.9: State Space Feedback Block Diagram.

The discrete LQR design determines the matrix K_d which minimizes the quadratic cost function of the following form [10]:

$$J(u) = \sum_{n=1}^{\infty} (x[n]^T Q x[n] + u[n]^T R u[n]) \tag{4.9}$$

where K_d is obtained by:

$$K_d = (B_d^T S B_d + R)^{-1} (B_d^T S A_d) \tag{4.10}$$

And S is found by solving the associated Riccati equation:

$$A_d^T S A_d - S - (A_d^T S B_d)(B_d^T S B_d + R)^{-1}(B_d^T S A_d) + Q = 0 \tag{4.11}$$

All these calculation are done in Matlab using the command *dlqr*(A_d, B_d, Q, R), which gives K_d that fulfils the previous statements. Moreover, a first choice for the matrices Q and R is done using Bryson's rule [15]:

$$Q = \begin{bmatrix} \frac{\alpha_1^2}{(x_1)_{max}^2} & & & \\ & \frac{\alpha_2^2}{(x_2)_{max}^2} & & \\ & & \ddots & \\ & & & \frac{\alpha_n^2}{(x_n)_{max}^2} \end{bmatrix} \tag{4.12}$$

$$R = \begin{bmatrix} \frac{\beta_1^2}{(u_1)_{max}^2} & & & \\ & \frac{\beta_2^2}{(u_2)_{max}^2} & & \\ & & \ddots & \\ & & & \frac{\beta_n^2}{(u_n)_{max}^2} \end{bmatrix} \tag{4.13}$$

where $(x_i)_{max}$ and $(u_i)_{max}$ are the maximum desired values of the states and the control inputs, while α and β are used to define the weighting of each state and control input in the controller design.

For the quadcopter, R is a square 4x4 matrix, while Q is a 12x12 matrix. In order to find them, the following considerations are done:

- As there are many states and the goal is not to control all of them, the weighting of the states v_x, v_y, v_z, ω_ϕ, ω_θ and ω_ψ is set to 0.
- Controlling position $[x, y, z]^T$ is not the main goal but $[\phi, \theta, \psi]^T$ is because it is essential to assure that pitch, roll and yaw angles are approximately 0 in order to fulfil the linearisation assumptions and to keep the controller working. Thus, the highest α coefficient is set to the states ϕ, θ and ψ.

After adding the feedback, given by the LQR, the poles of the system are located in the left-hand plane:

$$\begin{aligned}
&-22.3305 + 22.6874i \\
&-22.3305 - 22.6874i \\
&-22.3305 + 22.6874i \\
&-22.3305 - 22.6874i \\
&-3.3092 + 3.3092i \\
&-3.3092 - 3.3092i \\
&-0.9051 + 0.9051i \\
&-0.9051 - 0.9051i \\
&-0.9051 + 0.9051i \\
&-0.9051 - 0.9051i \\
&-2.6930 + 2.6930i \\
&-2.6930 - 2.6930i
\end{aligned}$$

They are illustrated in Fig. 4.10 as well.

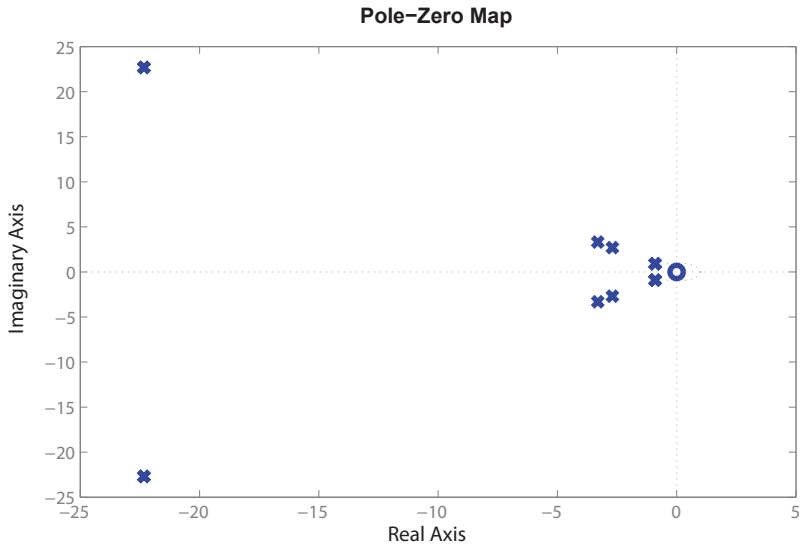

Figure 4.10: Poles and zero of the system, after applying a LQR.

4.3.1 LQR in Simulink

Simulink is used to investigate the behaviour of the discrete LQR controller that is expected to be programmed on an autopilot in future work. Pitch, roll and yaw references are set to 0 in order to have the quadcopter close to the model used, while $[x_{ref}, y_{ref}, z_{ref}]^T$ do not have any limitations. Finally, angular and linear velocity references are set arbitrarily to 0 as they are not taken into consideration in the control. The four PWMs of the motors are the inputs of the system and a saturation block will limit this control signal. This is illustrated in Fig. 4.11.

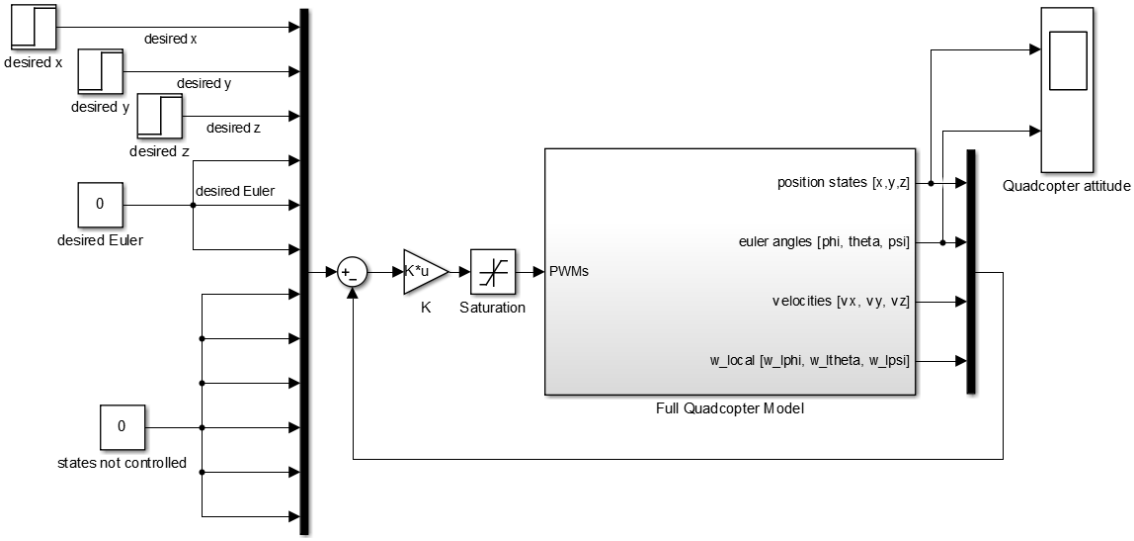

Figure 4.11: Simulink layout of the LQR controller used in the quadcopter full model.

The gain block *K* works as the discrete LQR controller. Its optimal sampling time is found by trial and error in simulation and this is how the frequency of the controller in the autopilot is found. A frequency below 20Hz is makes it impossible to control and over 35 Hz it starts to show a suitable behaviour as it can be seen in Fig. 4.12. A frequency of 50Hz is chosen for further studies.

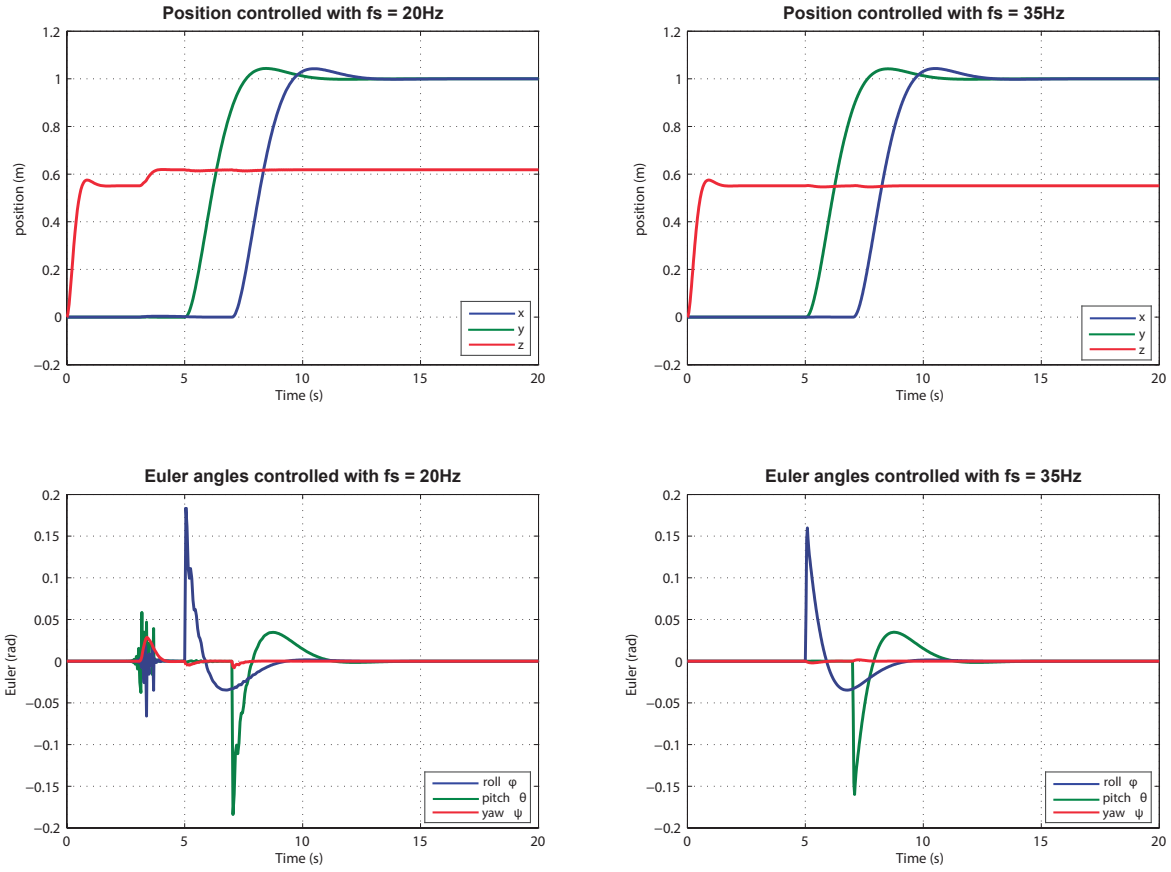

Figure 4.12: Comparison of the control considering a frequency of the controller of 20Hz and 35Hz.

Additionally to the frequency of the controller, the previous plots show the attitude of the quadcopter and it can be seen that z is not reaching the desired position of 1 as the model used for the controller design did not take gravity into consideration. Despite this stationary error, can not be fully corrected unless another model is used or integral action to z is introduced, some improvements can be done regarding the R matrix defined in Eq. 4.13.

The maximum desired value of the control signals $(u_i)_{max}$ that appears in the R matrix can be modified to give more or less strength to the PWMs. After some simulation, it is seen that giving more power to them will result in a smaller stationary error, but will also lead to more aggressive response to any reference as shown in Fig. 4.13.

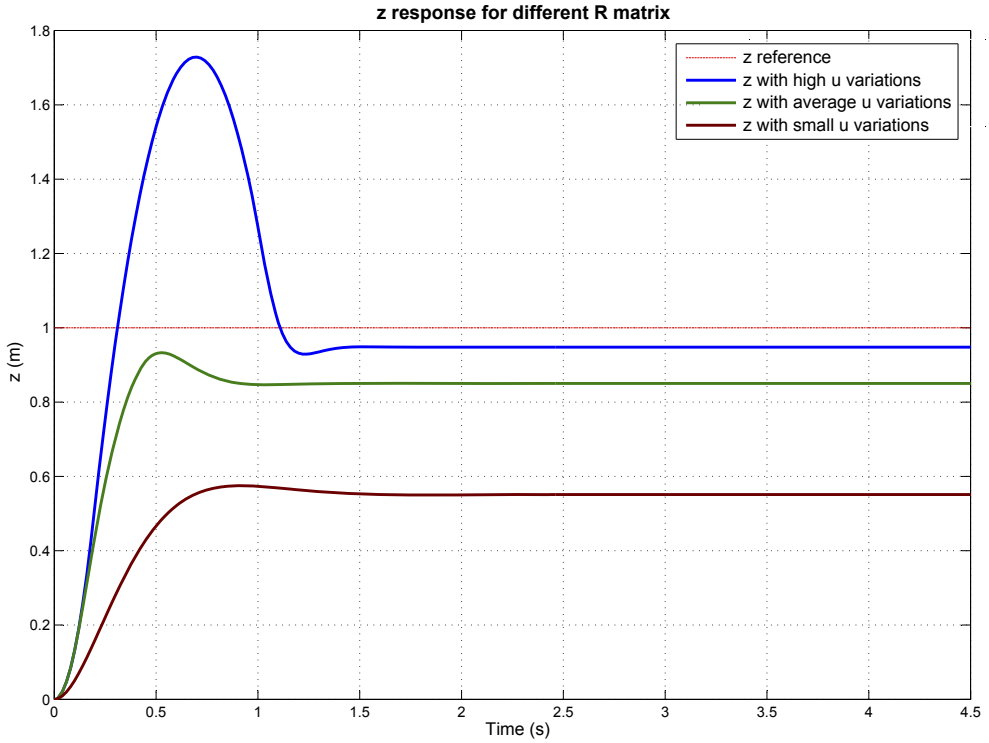

Figure 4.13: Comparison of the z control with different u variations.

Despite getting a better result with high and average u variations, the control signal is more aggressive and then Euler angles are more unstable. For this reason, small u variation are preferred in the control even if the z stationary error is higher.

4.4 Controller Conclusion and Analysis

An LQR controller is applied instead of four individual ones. Its operational range is determined by the limitations of the model used, which mainly needs to have Euler angles close to 0. Thus, pitch, roll and yaw are controlled altogether with the position in the three axis.
Additionally, a study about the discrete controller determines that the board should provide at least a frequency of 35 Hz. Otherwise, the stability of the controller is affected and the quadcopter may crash, since its response would be too slow. Likewise, it is noticed that gravity force produce a stationary error when controlling z. A first approach to partially correct it consists of designing the LQR controller with higher variations in the PWM signal. The error is reduced, but in return, the movement of the quadcopter is more aggressive and can lead to very high overshoots which is not desired. A solution proposed for future works is to design the LQR controller with integral action in z.
Conclusively, the LQR design is a good control strategy to begin with, but an improvement to the controller should be done in case a more precise z position is expected.

CHAPTER 5

NETWORK

From the mathematical model and the controller design, a simulation of the network communication is carried out, with the purpose of testing the behaviour of the quadcopter under certain circumstances where there might be communication breakdowns.

5.1 Network design

A wireless network simulates the connection between a ground station and the quadcopter where the ground station is used to send the desired position $[x_{ref}, y_{ref}, z_{ref}]^T$ and the current position $[x, y, z]^T$ to the quadcopter. Current position is measured with a motion capture system, such as a Vicon System, connected to the ground station as shown in Fig. 5.1 and the link between them is assumed to be fast enough that it will not be considered in the simulation. Finally, the data received is processed in the quadcopter autopilot, where an algorithm implementing the designed controller is coded. Thus, the feedback control process is completed.

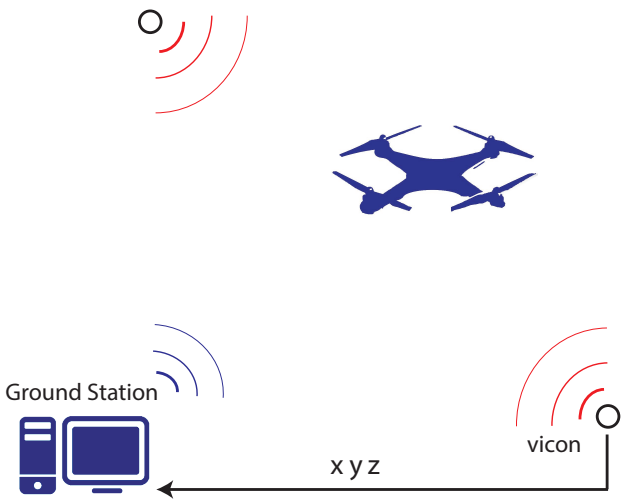

Figure 5.1: Motion capture of the quadcopter (Vicon) scheme.

In addition, a short range network is used for the communication between quadcopters during flight formation. This second wireless network consist of a master quadcopter transmitting desired and current position to each quadcopter, thus performing the flight formation. This configuration enables the quadcopter controllers to follow instructions from a ground station located far away. The scheme in Fig. 5.2 shows how a master quadcopter is responsible of receiving the control data corresponding to each quadcopter. This data contains message identification, references and current position of the receivers. Then, the master quadcopter processes the data and distribute through the short range network.

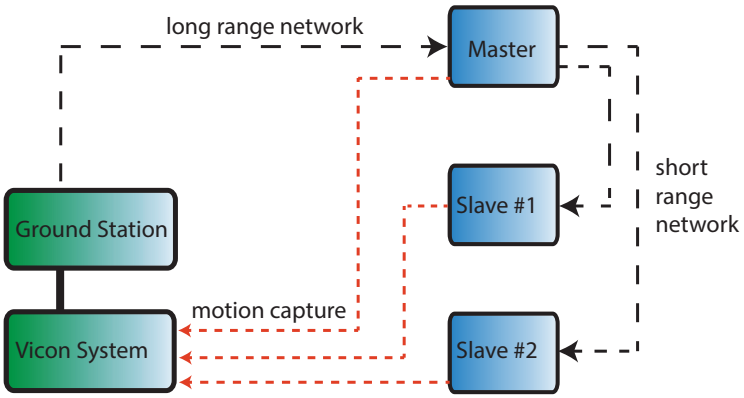

Figure 5.2: Scheme of the network design

5.1.1 Network simulation

The simulation is carried out, using TrueTime 2.0 Beta [4]. This represents the Real Time communication between the quadcopters and ground-station as shown in Fig. 5.2 and also includes the discrete system of the board where the computation of the input signal is made. The scheme in Fig. 5.3 presents the node distribution of the simulation in TrueTime, where the following blocks are represented:

1 The two wireless networks used in the system, short and long range.

2 The set of motion capture (Vicon) and ground station transmitter node.

3 The set of reception node and network handler node on master quadcopter.

4 The reception node of slave 1 quadcopter.

5 The reception node of slave 2 quadcopter.

6 The control and mathematical model of the quadcopter.

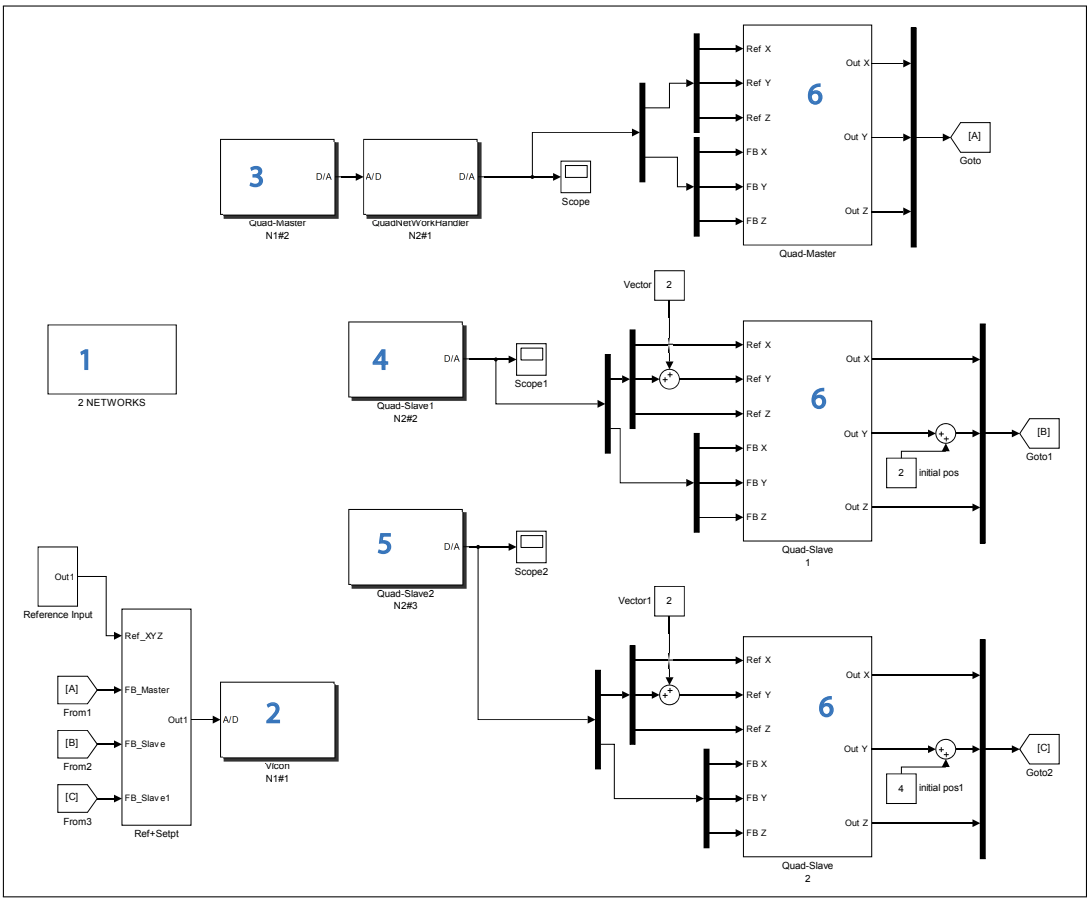

Figure 5.3: Full simulation of the network in True Time

Simulation intends to analyse the network communication and the performance of the quadcopter in different scenarios, where data loss probability and path-loss are studied and compared. The delay produced in the data received that affects the controller is observed. And finally, flight formation restrictions in a network are analysed.

5.1.1.1 Data loss probability

Simulations of the complete system are performed under circumstances where the network can suffer interferences or other communication breakdowns that lead to data loss. In the following graphs, the behaviour of the attitude controller of the quadcopter can be observed when 0%, 40% and 60% of the packages are lost, in Fig. 5.4, 5.5 and 5.6 respectively. Furthermore the maximum number of times a node will try to retransmit a message is set to zero.

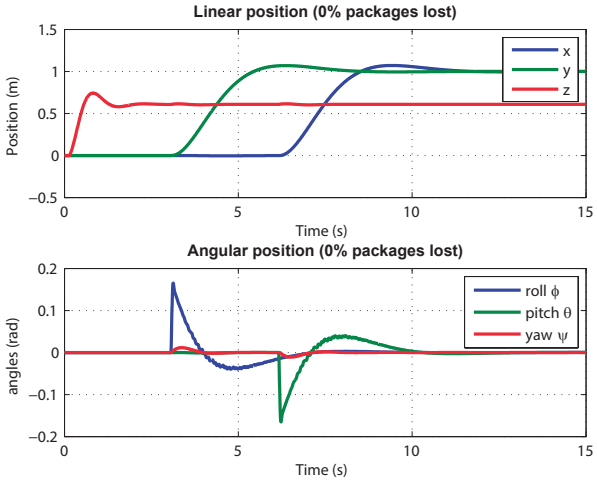

Figure 5.4: Attitude control in a 0% loss probability scenario.

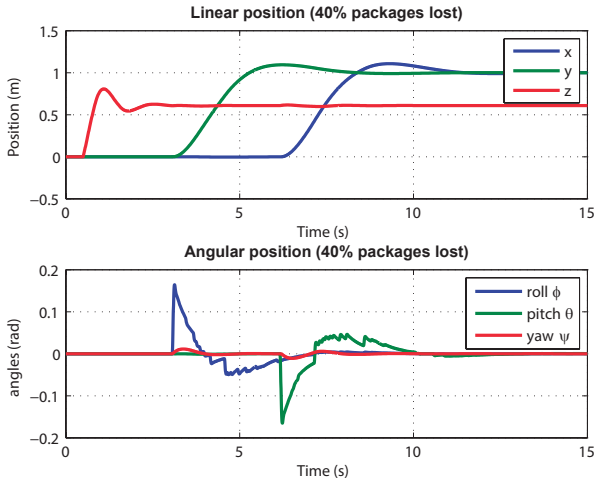

Figure 5.5: Attitude control in a 40% loss probability scenario.

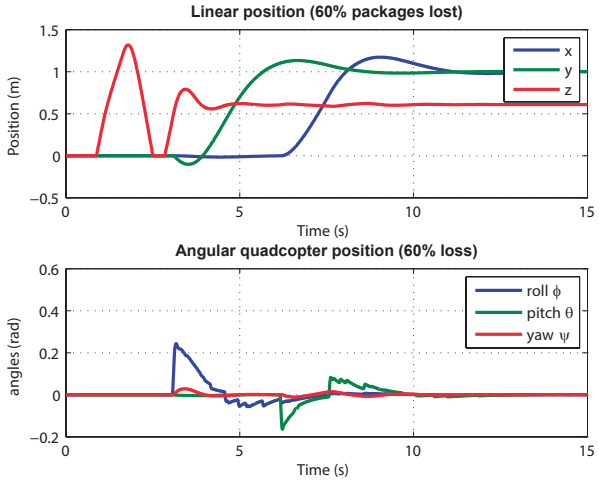

Figure 5.6: Attitude control in a 60% loss probability scenario.

These results show that the hover control is no longer stable when there is a data loss between 50% and 60% as it is not able to reach the reference signal. Additionally, Euler angles are also affected by the loss of packages and are hardly controllable with higher data losses. As can be observed in Fig. 5.5, data loss probability around 40% slightly disturbs the hover control, as well as Euler angles become less steady than under ideal circumstances but never leading to instability.

5.1.1.2 Path-loss

For this simulation, the path-loss function is defined as in Eq. 2.45 described in Sec. 2.6. This project will depict one of the worst cases scenario in terms of network interference. Hence, the simulations in TrueTime are carried out with the path loss exponent, $a = 4$. The protocol of transmission, power of the transmitter and receiver and data rate are modelled according to the datasheet specifications of the hardware chosen, distance d is a variable which depends directly on the current position of the quadcopter $[x, y]^T$. According to the framework selected for the wireless network ZigBee (802.15.4), the transmit power is 18.00 dBm, receiver threshold is -100.00 dBm and finally, the maximum signal reach is calculated to 890.25 m [7].

In the simulation, the quadcopter (receiver) is set to follow a sinusoidal reference at the time it starts flying away from the ground station (transmitter). The starting point is set at 875 meters in order to illustrate the communication failure, with a proper resolution. The failure occurs due to the dissipation of the signal.

In the first graph on Fig. 5.7 a breakdown in the communication occurs when the quadcopter surpass the maximum operating distance, which is approximately 890 meters. From there, the communication with the quadcopter is lost. On the second graph on Fig. 5.7 the behaviour of the quadcopter is shown with respect to the reference and current position. It can be observed that once it reaches areas where the controller does not received any data, the controller keeps processing control values according to the last data received. These data values remain constant in the controller after the breakdown in the communication. This fact force the quadcopter to fly permanently in the same direction it had when the communication was lost, since the feedback error (reference received - position received) remains constant. Thus, the ground station lose all control over the device.

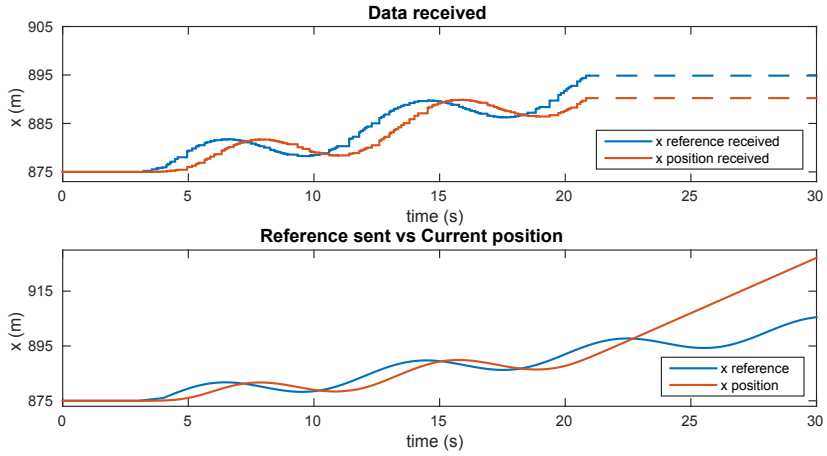

Figure 5.7: a) Data received by the quadcopter from the ground station. Comparing the reference signal and the current position measured. The point where the communication is lost is shown with a dashed line. b) Reference signal sent from ground station compared with the current position of the quadcopter

5.1.1.3 Flight formation analysis

Communication between three quadcopters in flight formation (one master and two slaves) and the ground station is simulated. The master is responsible for providing a proper reference signal, through a short range network that the slaves use to maintain the relative distance in the flight formation. Thus the master quadcopter compute and distribute reference values so each quadcopter is holding a certain safety distance respecting the closest quadcopter.

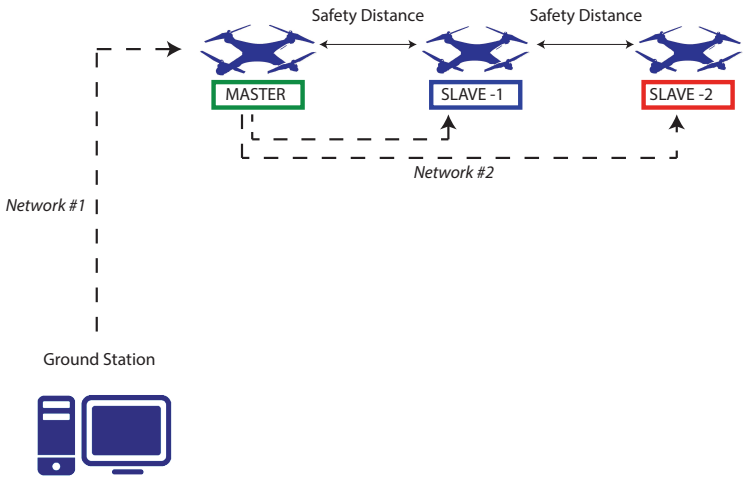

Figure 5.8: General scheme of the network distribution and flight formation.

For this analysis, the formation is an alignment over the y axis, where the distance between them in the this axis is set to 2 meters. The speed in the y axis is studied in order to find the flight formation restrictions, that allows the quadcopters to keep the formation without colliding.

The method used consists on giving a sinusoidal reference of the form $y_{ref}(t) = A \cdot sin(\omega t)$ to the master during a certain period of time in order to see the behaviour of the formation when there is a change in the y direction. The expected velocity is estimated to be the reference derivative:

$$\frac{dy_{ref}(t)}{dt} = A\omega \cdot cos(\omega t) \approx v_y(t) \tag{5.1}$$

The critical point of the flight, where they may collide, is when the distance between the quadcopters is minimal. This minimum is reached when the quadcopters are flying at the maximum velocity which is given in Eq. 5.1 when $cos(\omega t) = 1$.

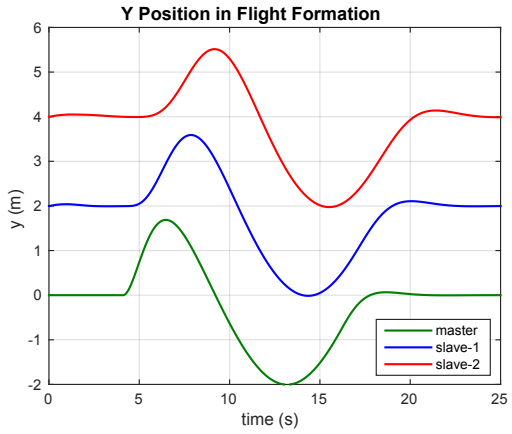

Figure 5.9: Position y of three quadcopters in flight formation reaching a maximum velocity of 0.8 m/s. Safety distance between them is still maintained during all flight.

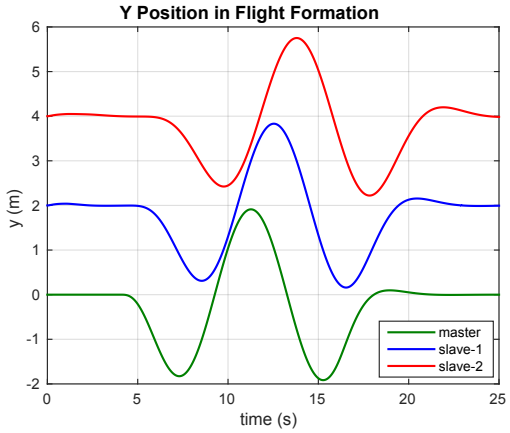

Figure 5.10: y position of 3 quadcopters in flight formation with a maximum velocity of 1.6 m/s. Safety distance is no longer maintained at the critical point.

Results show how the slaves are not able to keep the safety conditions when the velocity in the y axis is higher than $v_{max} = 1.4$ (m/s). In Fig. 5.9 flight is stable, while in Fig. 5.10 the quadcopters collides into each other.

5.2 Network Analysis

After studying the three subsections, data loss probability in 5.1.1.1, path loss in 5.1.1.2 and flight formation in 5.1.1.3, in a network between quadcopters and ground station, the results leads the limitations produced by a network.

The first study infer that the most suitable scenario for a working controller is to have rates of data loss below 50%, otherwise the quadcopter is no longer stable during flight. On the other hand, the path-loss study proves that the power of the signal sent by the ground station is lost when the receiver is 890 meters away from the transmitter. Hence, ground station lose the control of the quadcopter. This issue could be fixed implementing a control algorithm that make the quadcopter remain in the same point in hover position until new data is received. In addition, the use of flight formation implies a limitation to the velocity in the axis of the formation in order to avoid collisions. In case the speed in the axis of the flight formation is needed to be increased, the safety conditions must be scaled proportionally. In conclusion, results show how a network can affect the performance of the quadcopter controller and flight formation up to the point that collisions could occur.

This configuration of the wireless network consists of one single long range for the communication master quadcopter - ground station, and a short range for the communication between quadcopters. Other designs have been proposed to this system, in order to optimize the power resources of the devices. Another approach for future development, in terms of networking, can be introduced. A Wireless Mesh Networking (WMN) [20] maintains, in a

way, the reliability of the communication and an optimization of the energy used on the communication process is achieved. In this kind of configuration, the network connection is spread among many nodes sharing the network along a broad area. These nodes are programmed according to how they have to interact in the network, thus the data is sent from one point (master) to other (slave) the nodes automatically choose the best path.

The great advantage of introducing a mesh wireless network in the quadcopter communication is to reduce the power consumption on the designed network. It is planned to develop a wireless communication configuration, in which, not only one of the quadcopters has long range module (a master), but all of them are able to carry out this communication. Switching master role between the quadcopters, the power consumption in terms of networking is shared amongst all the quadcopters in the formation, since not all of them would transmit simultaneously. In addition, develop a mesh network in the short range contribute to reduce the power consumption considering that the distance of the transmission is reduced.

Path-loss has dependency on distance. Introducing more nodes, that are interacting in the network may reduces the distance of the transmission, as shown in Fig. 5.11. The signal is guided from one quadcopter to another, until it reaches the receiver.

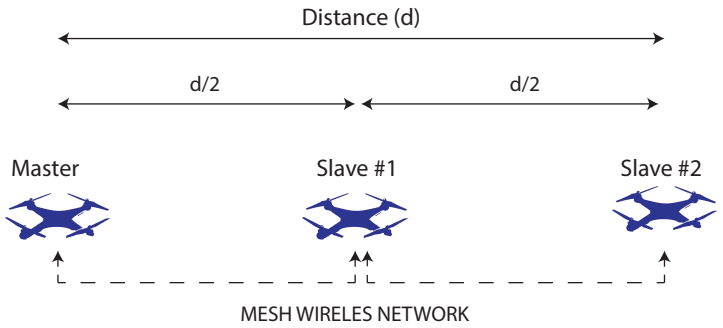

Figure 5.11: Scheme of the mesh wireless network during the flight formation communication.

In order to reach the receiver, the power of the signal sent is defined according to the Free Space Path Loss (FSPL) distance dependency [21]:

$$S = P_t \frac{1}{4\pi d^2} \tag{5.2}$$

where S is the power per unit area at distance d, P_t is the power of the transmitter and d is distance between transmitter and receiver.

In case data is sent directly from master to slave 2, as shown in Fig. 5.11, the power is com-

puted:

$$P_{total} = S \cdot 4\pi \cdot (d)^2 \tag{5.3}$$

When using mesh network, the total power of the transmission is calculated as the sum of the power of transmission between nodes:

$$P_{total} = S \cdot 4\pi \cdot (d/2)^2 + S \cdot 4\pi \cdot (d/2)^2 = \frac{S \cdot 4\pi \cdot (d)^2}{2} \tag{5.4}$$

As the distance is reduced, the total power of transmission is reduced as well.

Part IV

Implementation

CHAPTER 6

HARDWARE

The hardware that makes up a quadcopter is depicted in Fig. 6.1. Each module and its purpose is described throughout the sections in this chapter.
The Pixhawk, which will be described more in the next chapter, is connected to four electronic speed controllers (ESC), which in turn is connected to the motors. It should be noted that the motors are directly connected to the battery as well, even though that is not depicted in the image. The signal sent from the Pixhawk to the ESC is a PWM signal, which will drive the motors. The pins applied are one to four on the *Main Out* marked pinout section. The Pixhawk supports up to 8 motors (e.g. octocopter). The Telemtry port 1 is connected to the wireless Xbee module through serial connection, and will communicate with the ground station.

Ground Station

The ground station is computer, running a software such that an operator can communicate with the quadcopter. All data that is desired to be recorded for further analysis will be sent to the ground station. The ground station could also be connected to a motion capture setup. In addition to desired data, the ground station transmits waypoints and commands for safety concerns, e.g. landing procedure if something goes wrong or simply disarm motors and let it crash.

Wireless Link

Communication between the quadcopter and the ground station will go through radio communication. The wireless modules that is be applied in this project is the XBee Pro's (63mW), which ranges up to 1600 meters. One is connected to the ground station and another to the quadcopter using the serial port. The XBee has a RF data rate of 250 kbps.

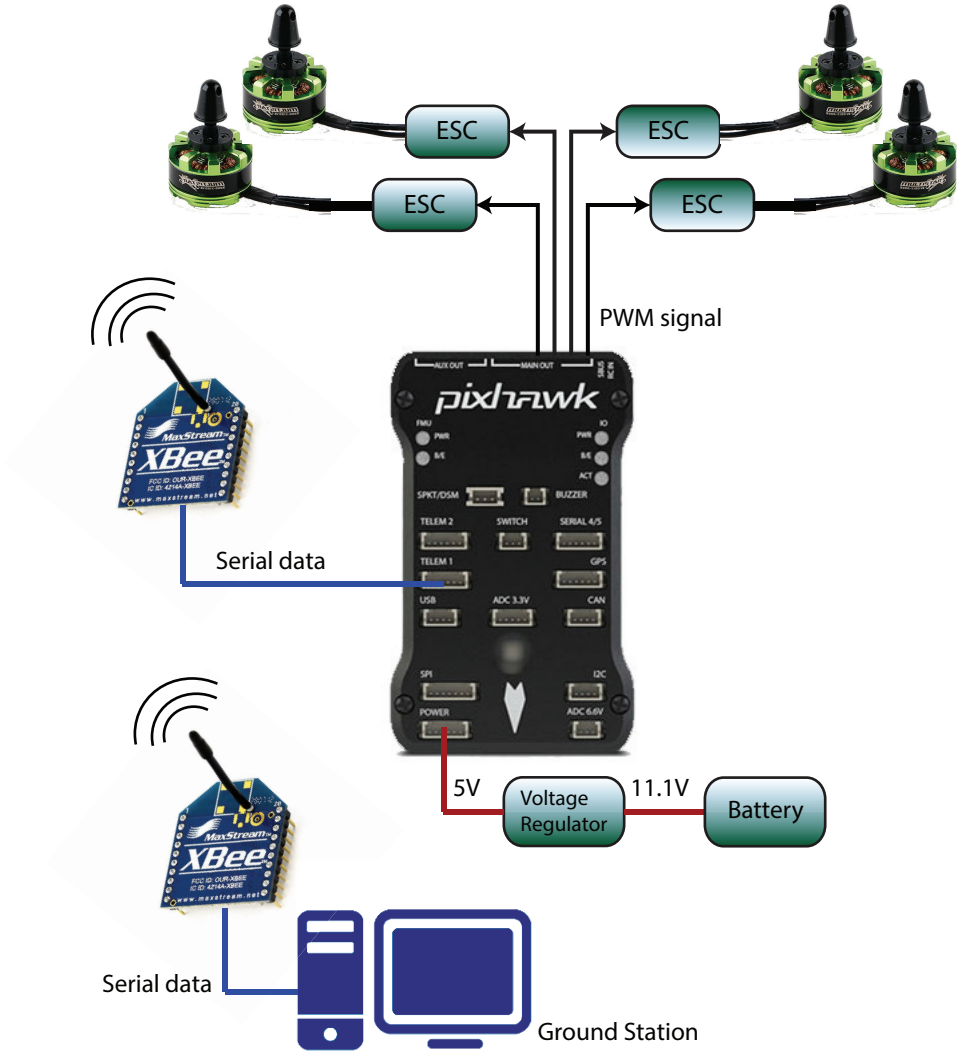

Figure 6.1: A diagram over the hardware section

Electronic Speed Controller (ESC)

The ESC installed on the quadcopter is the Turnigy Multistar 30 Amp Multi-rotor ESC. The main purpose of this module is to control the angular speed of the DC motor by varying the voltage. The ESC is directly connected to the battery. Four of these modules are attached to the quadcopter frame, one for each motor. The ESC's provides a very smooth and linear throttle control which will respond fast to the input throttle.

Brushless Motor

Brushless motors provides the quadcopter with the thrust force. The MT2213-935KV MultiStar Motor and the size of the propeller where selected with respect to the size of the frame which

is 450 mm in diameter. The motor has a kv (RPM/V) of 935 and is rated to a maximum of 15A. The size of the propellers are 10-4.5 inches were the two of them will rotate clockwise and the other two counter clock wise. Four of these modules are applied to the quadcopter frame.

Pixhawk

The 3DR Pixhawk is the autopilot that controls the quadcopter, and is based on open source software, which will be described in more detail later. The Pixhawk is a high-performance autopilot-on-module and is supports various of vehicles, such as the quadcopter (or multi-rotors in general), helicopters, cars, boats and any other robotic platform that can is in motion. It is targeted toward high-end research. Its key features are

- 168MHz Cortex M4F, 256kb RAM, 2MB Flash
- 14 PWM or Servo outputs, where 8 is with failsafe and manual override, 6 auxiliary and high-power compatible.
- Peripherals such as IC2, UART and CAN buses.
- External safety switch
- microSD card for high-rate logging.
- Sensor Modules
  - ST Micro L3GD20H 16 bit gyroscope
  - ST Micro LSM303D 14 bit accelerometer / magnetometer
  - MEAS MS5611 barometer

and more [8].

CHAPTER 7

SOFTWARE

The software section will describe the software applied in the autopilot, as well as algorithms that are used.

7.1 Pixhawk Open Source

PX4 is an open-source and open-hardware project that provides a high-end autopilot for academia. It serves as a full-package solution to hardware, software platforms, and is used by several Institutes, such at *Computer Vision and Geometry Lab of ETH Zurich*.
The Pixhawk autopilot applies a real-time operating system (RTOS), providing the means to create a multi-threading environment. The firmware can be updated using a USB bootloader [8].

7.1.1 Firmware Overview

The PX4 firmware comes with various applications and in-built functions. Inter-task communication is conducted using an application called micro Object Request Broker (uORB), that allows for a simple way to share data structures between threads and applications. The uORB application start up with the RTOS by default, and it is thus not necessary to initiate it.
A mixer is also implemented in the firmware, which is a set of pre-defined rules and parameters to produce a set of outputs. In short, it reads from control inputs and write to actuator outputs while obeying a set of rules. This option however was not tried implemented in the code implementation in this project, where low-level actuator control was used instead.
Applications can be implemented using Daemon, where its status can be queried during runtime from a shell and stopped.

7.1.2 Mavlink

Mavlink is an abbreviation for Micro Air Vehicle Communication Protocol, which is a lightweight, header-only message library for micro air vehicles. It has been extensively tested and applied on the Pixhawk. The protocol packs C-structs over serial channels and send these packets to ground station. Software such as QGroundcontrol and APM Planner supports this protocol.

7.1.3 Implementation

With the previously described background, the implementation code was developed and is explained in the coming subsections.

System

As a part of initializing the system, it is required to go through these steps in the NSH.

1. Stop unimportant tasks (e.g other control routines)
2. Arm the unit, pressing the button until 2 quick flashes occur periodically.
3. Start calibration of the ESC.
4. Start the application.

In Fig. 7.1 the main program is depicted. It consists of three main steps, where the first is to create a thread using Daemon, the second is the initialization sequence and the third is the main loop. The application will communicate with three modules.

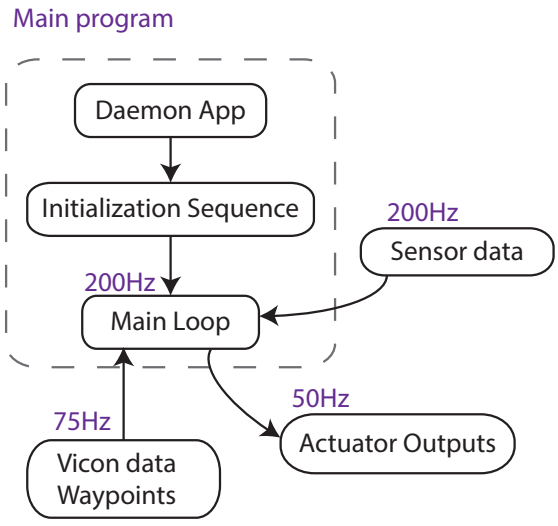

Figure 7.1: Application with its three steps and data sharing with outside features.

The first is receiving the sensor data and when using uORB as internal communication protocol it is possible to define a fixed interval of subscription which has been set to 200Hz in

this program, which also corresponds to the loop iteration frequency. The second module is the connection to the actuators, which will be set by the control output with 50Hz. Lastly the ground station transmits motion captured data and waypoints with approximately 75Hz.
The thread creation has been depicted in Fig. 7.2, where the programmer defines settings, such as priority and stack size.

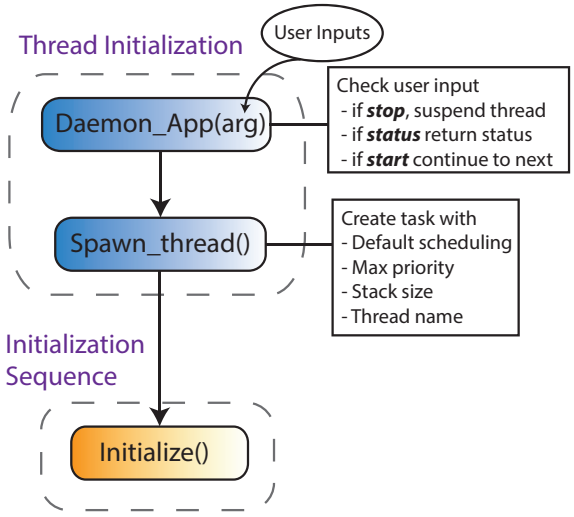

Figure 7.2: Thread creation using Daemon.

The user can communicate with the thread from the NSH, but only on a high level, where the thread can be terminated, started of check its status. Direct communication with the thread would have to take place using mavlink.
The second step in Fig. 7.1 is the initialization sequence, which is depicted in Fig. 7.3.

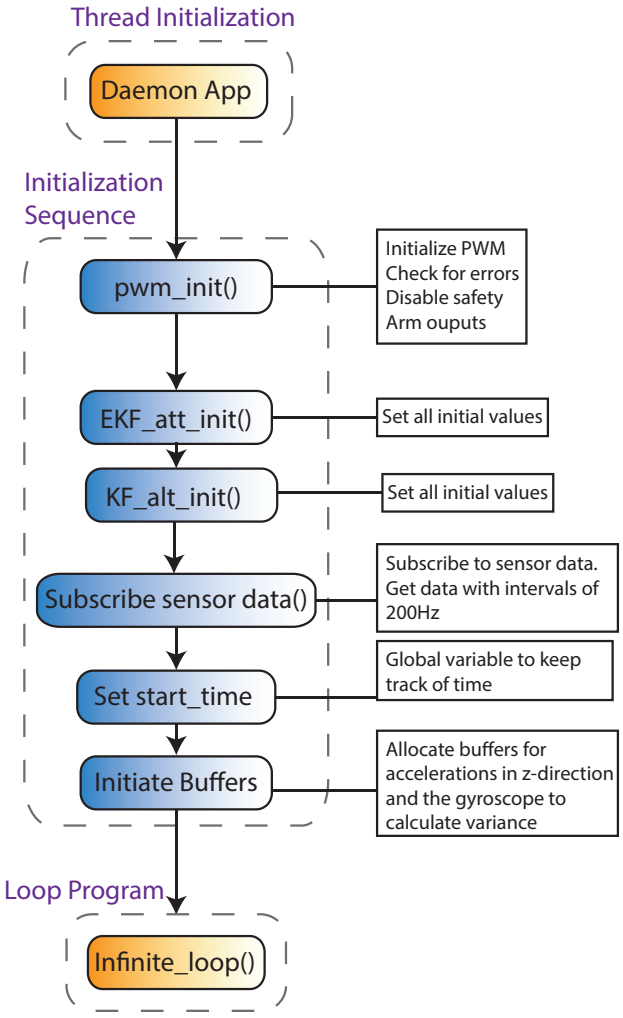

Figure 7.3: Initialization sequence that executes once before the main loop.

The PWM to the actuators are initialized in this sequence, such that it is armed and ready to process data. Afterwards the kalman filters are initialized, for both the inbuilt EKF and the KF programmed as a part of this project. The kalman initialization sets the initial values and covariance values. Followed by sensor subscription using uORB with 200Hz, like previously explained. Lastly a variable that holds the start time (in microseconds) is created, to keep track of time in the main loop while finding the offsets and variances.

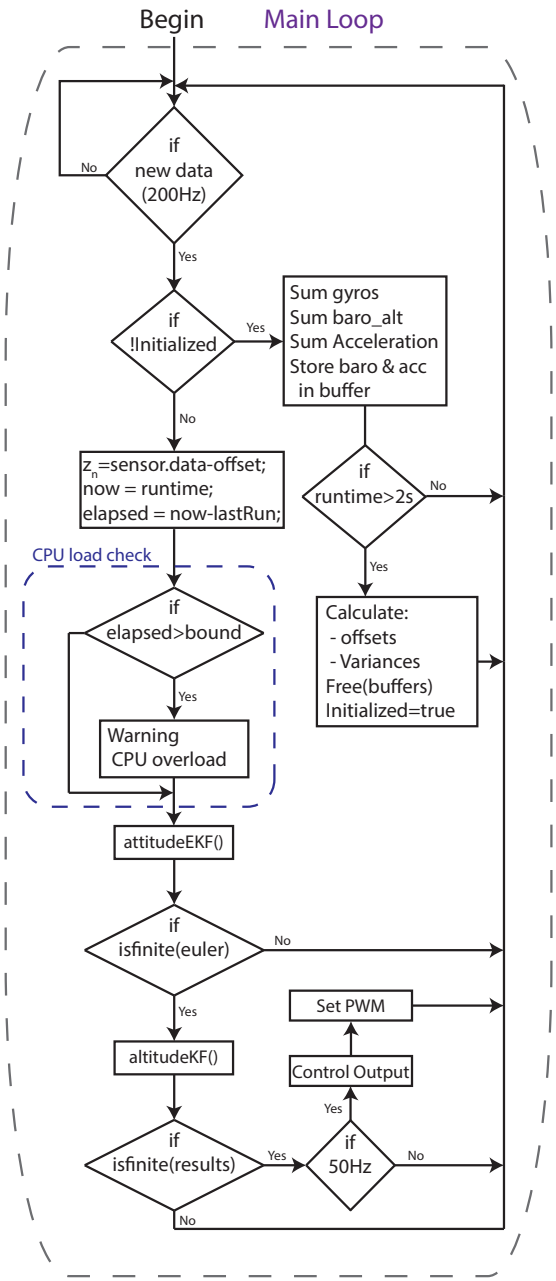

Figure 7.4: Main loop.

As the third step, the main loop will execute for as long as the Daemon application allows (e.g. user requests a termination). The first part of the main loop is sampling sensor data, which are to be used to determine offsets and variances. The loop initialization will iterate until the application has executed for 2 seconds, using the previously mentioned variable that holds the start time. When 2 seconds has surpassed the offsets and variances are calculated, the buffer space is released and the *Initialized* flag is changed, such that it will not execute again. The second part of the main loop is the actual main loop, where first the sensor data is adjusted

with its offset, as calculated in the loop initialization, and the time for last loop iteration is recorded. After which a short CPU check will commence, to confirm or disprove that the CPU is not overloaded. In the project code, it will only give warnings, but if repeatedly happening the code should terminate after a landing procedure.
After the CPU check the inbuilt extended Kalman filter will estimate the attitude. When finished the Euler values will be checked, e.g if a calculation returns NaN. If euler values are valid the constructed Kalman Filter, as described in Sec. 2.7 will execute, to estimate the vertical velocity and altitude. Lastly the output of the KF is checked again and the actuator sequence will commence, where control output will be performed with a 50Hz frequency.
In future work it would be desirable to create another thread, having only the control sequence, while subscribing to state variables estimated in the main loop as well as waypoints given through mavlink.

Kalman Filter

The Kalman Filter in Sec. 2.7 was implemented on the Pixhawk and an experiment was conducted to verify improvements of the sensor estimates. The Pixhawk was attached to a quadcopter and placed on various levels (black lines) on a shelf in resting position, while sampling the barometer data and its corresponding KF estimates in a terminal.

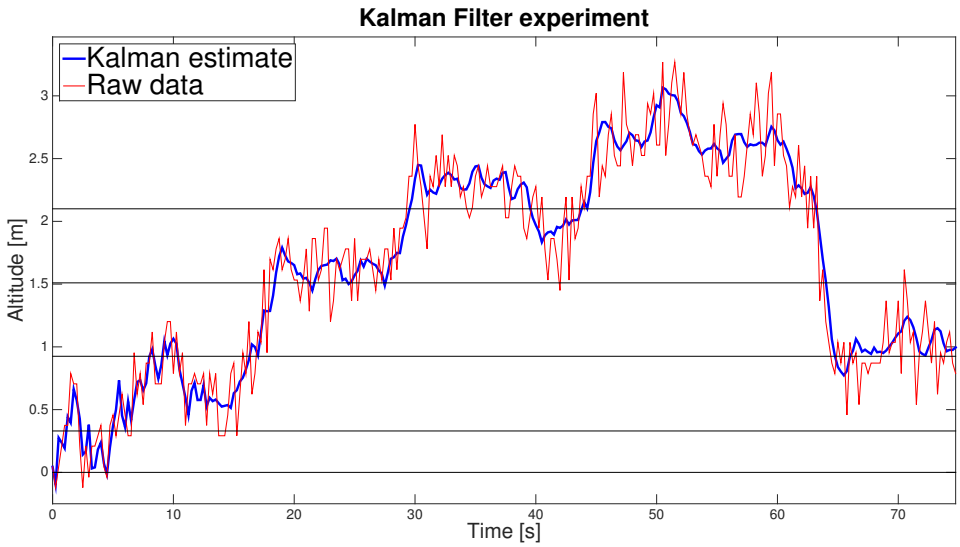

Figure 7.5: Altitude estimation, with raw data and KF estimates.

As the barometer have a small resolution (10cm) the estimates would not be applied in the control algorithm. The filter would possibly do well in an outside environment where 10cm is not of great importance. However in a motion capture lab, 10 cm tolerance could be an issue in terms of collisions.

Fig. 7.5 shows the raw sensor data (red) versus the filtered data (blue). The filtered data clearly indicates a smoother transition from data point to data point. To verify, a frequency spectrum analysis is shown in Fig. 7.6.

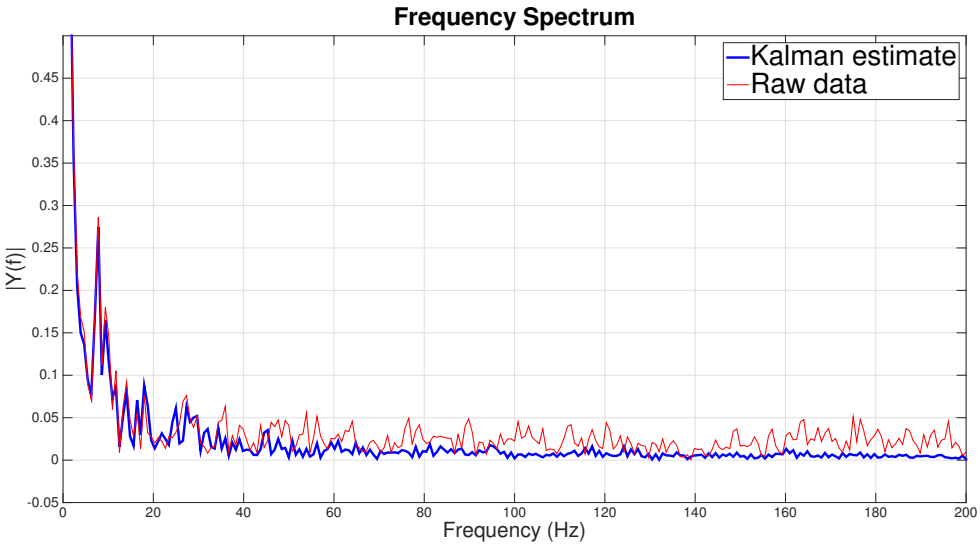

Figure 7.6: Single-sided Amplitude Spectrum of both signals.

Fig. 7.6 implies that the Kalman Filter attenuates the high frequency noise. Due to the sampling frequency of 400Hz and the Nyquist criteria, frequencies over 200Hz can not be analysed, so the analysis is strictly restrained as such.
The vertical velocity was also estimated, but was not tested in an experiment. Future work will include capturing the vertical velocity as with the altitude, but at the same time tracking its motion in a Vicon lab, where the actual vertical velocity can be tracked concurrently as a reference.

7.1.4 Software Summary

Due to the time constraints of the project, not all software, which is necessary for a in-flight control test, was implemented. The software that was successfully implemented is as follows

- Creating thread using Daemon
- Initialization
- Main loop
  - Data sharing using uORB
  - Initialization with offset and variance calculations
  - EKF for Attitude Estimation (open-source code from the firmware)

  - KF for Altitude Estimation
  - PWM output

The software that remains to be developed is as follows

- Mavlink routine - reads motion capture data and set points from ground station and publishes it.
- Control routine - calculates the control signal, while subscribing to the mavlink routine to acquire setpoints and motion capture data.
- Ground Station - Xbee running on ground station streaming position readings from the motion capture system while transmitting to the quadcopter with a specific frequency.

It should be noted that a working mavlink connection was created, but not fully developed to the purpose of the specific data to be shared, meaning that the main loop would not acquire usable data.

Part V

Closing

CHAPTER 8

CONCLUSION

The following chapter will summarize the results and corresponding conclusions to the work performed in this project.

Control Design

In the design phase, a controller for the quadcopter was needed to achieve autonomous flight. Initially, four systems for x, y, z and yaw were used to design independent controllers. First using PID and then integral control in state space. However, the results indicated instability in x and y when the control signals were coupled. Thus, an LQR controller for the multiple input multiple output system was designed and tested successfully.

Network and Flight Formation

After studying the path-loss and data loss probability in a network between quadcopter and ground station, the results infer that the most suitable scenario for a working controller is to have rates of data loss below 50% to maintain a fast and stable control output. On the other hand, the path-loss study proves that the power of the signal sent by the ground station is lost when the receiver is at a 890 meters away from the transmitter, when using the aforementioned Xbee setup. Hence the ground station loses control of the quadcopter. This issue could be compensated by implementing an algorithm that makes the quadcopter go into hover mode and thus, stay in its current position when the signal is lost. In addition, the use of flight formation implies a limitation to the velocity in the alignment axis of the formation. When the quadcopters reaches a velocity more than 1.4 m/s collisions may occur, with this specific controller and safety distance.

Software

The main algorithm for the quadcopter was designed and implemented, but a continuous communication with a ground station and a control algorithm remains to be coded. An open-source EKF was added to the code, followed by the previously designed Kalman Filter for altitude estimation. The Kalman Filter improved the sensor data, by removing high frequency noise. As the barometer resolution was too small, the control algorithm would not depend on this output. Finally, PWM output signals to the motors were successfully implemented.

Part VI

Appendix

APPENDIX A

LABORATORY TESTS

A.1 Introduction

As is indicated in the theory chapter, this appendix describes the tests carried out in a laboratory, in order to obtain coefficients which relates the angular velocity of the propellers and the forces generated.
This coefficients are dependant on many factors related to propeller features, such as size, angle of attack, air density. Due to the complexity of this analysis, a test of the aerodynamic behaviour is performed.

A.2 Thrust Test

The aim of this section is to estimate the coefficient that relates the angular velocity on the set motor-propeller with the force generated.

$$F_i = K_F \cdot \omega_i^2 \tag{A.1}$$

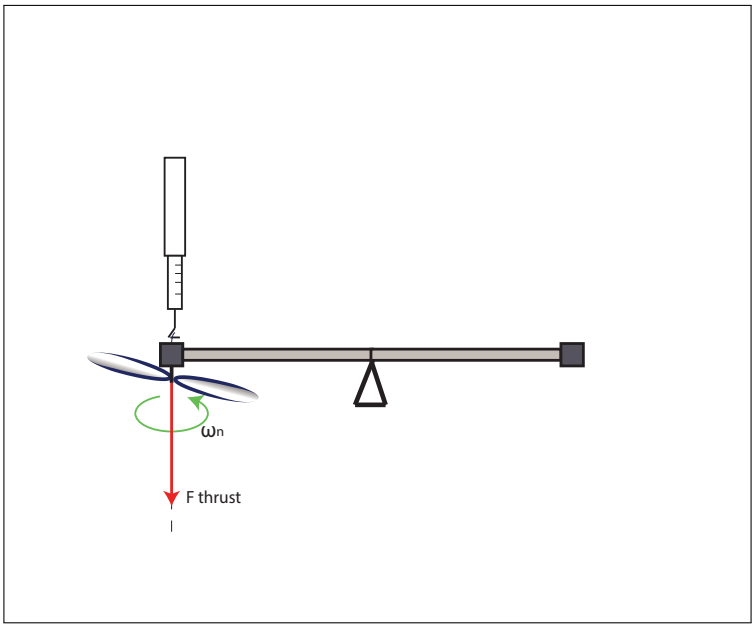

Figure A.1: Thrust force measurement

For this estimation the thrust force and the angular velocity are measured, among several samples from the minimum range of angular velocity to the maximum.

- The test is conducted on a stable metal frame that provided a high enough height to negligible the ground effects.
- Motor speed is regulated through the RC manual-mode from Pixhawk.
- To avoid measurement failures, the IMU from Pixhawk board must be kept equally balance in the 3 axis along the measurement procedure.

Instruments used in the measurement are stated as follows:

- Dynamo-meter
- Motor - MultiStar 2213-935
- PropellerCCW. 10x4.5R

A.2.1 K_F Calculation

The following subsection is an explanation of the estimation method used to calculate the K_F. A Matlab script has been used to plot the values and linearise the value of K_F according to the estimation of the slope. First of all, two plot compare the relationship between F and ω and ω^2. It is noticed a linearity when plotting ω^2 which fits the expectations.

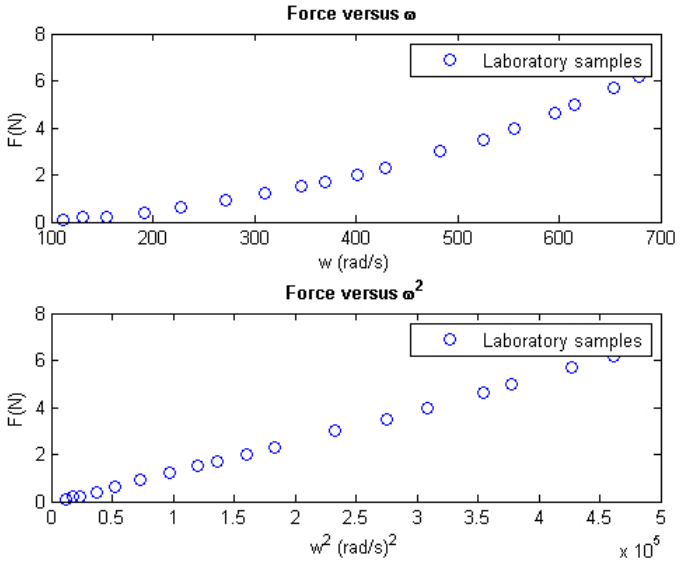

Figure A.2: Thrust force and ω^2

Matlab function *polyfit(x, y, n)* is used to compute the n^{th} order polynomial that fits with the samples. A first order polynomial (linear relationship) is used for ω^2 and 2nd order for ω.

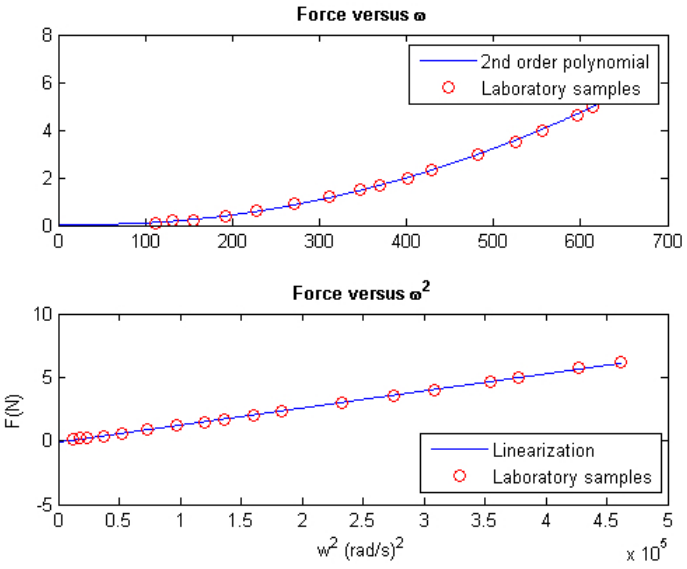

Figure A.3: Thrust force and ω^2 after linearisation

The linear approximation of the ω^2 plot gives the following form:

$$F = 1.3458 \cdot 10^{-5} \omega^2 \tag{A.2}$$

which leads to the desired constant: $K_f = 1.3458 \cdot 10^{-5}$.

A.3 Torque Test

The aim of this section is to estimate the coefficient that relates the angular velocity on the set motor-propeller with the torque generated from the drag force. For this estimation the torque generated by 2 motors placed on the same axis and the angular velocity are measured.

$$\tau_i = K_\tau \cdot \omega_i^2 \tag{A.3}$$

- To perform the measurements, the quadcopter is firmly attached to the torque test (some threads balance the set quadcopter-torque tester to accurate the measurements).
- The test is conducted far enough from any surface to consider negligible the ground effects.
- In order to measure the relation of PWM and ω, Motor speed is controlled applying directly to the ESC a PWM signal.
- PWM signal applied from 43% to 60 % with a frequency of 480Hz.
- Torque tester provide 5V-DC each $1N \cdot m$ applied.

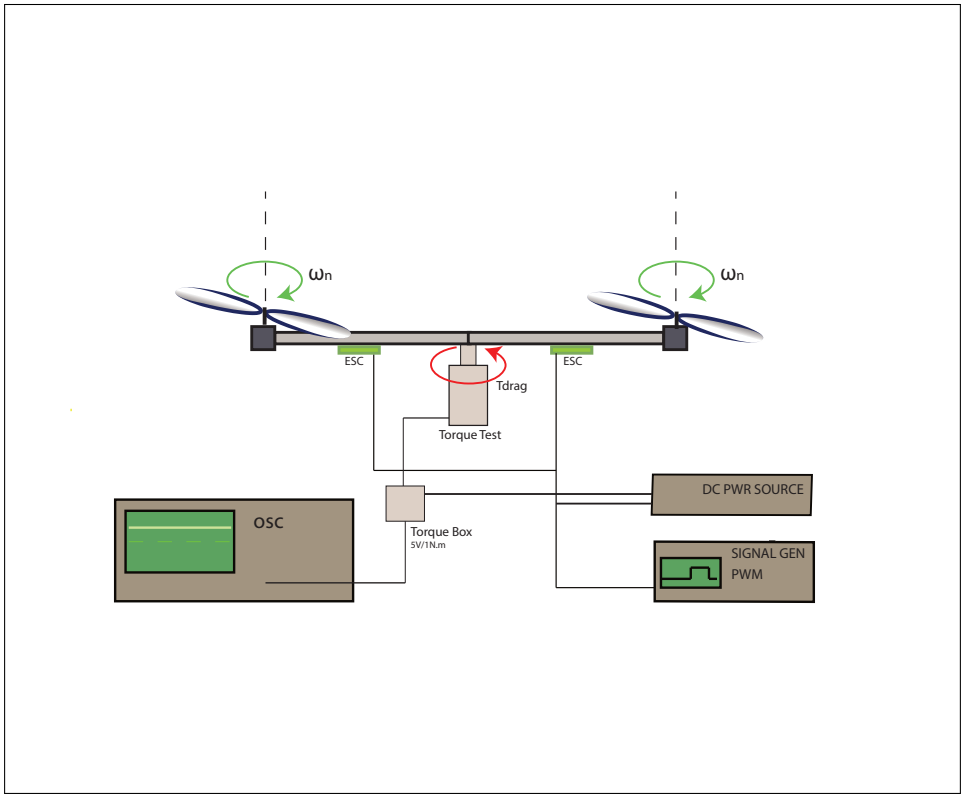

Figure A.4: Drag force measurement.

Instruments used in the measurement are stated as follows:

- Torque meter. HOLGER CLASEN. Typ MWA-W8-1-P SN 30690
- Oscilloscope. Agilent DSO6034A
- Signal generator. Agilent 33220A LXI
- Power Supply. HAMEG HM7042-3
- 2 ESC. MultiStar 30A
- 2 Motor quad.MultiStar 2213-935
- 2 Propellers CCW. 10x4.5R

A.3.1 K_τ Calculation

The following subsection is an explanation of the estimation method used to calculate the K_τ. A Matlab scrip has been used to plot the values and linearise the value of K_τ according to the estimation of the slope. The variation of ω and ω^2 respecting the torque is shown. It is noticed a linearity when plotting ω^2 which follows the expectations.

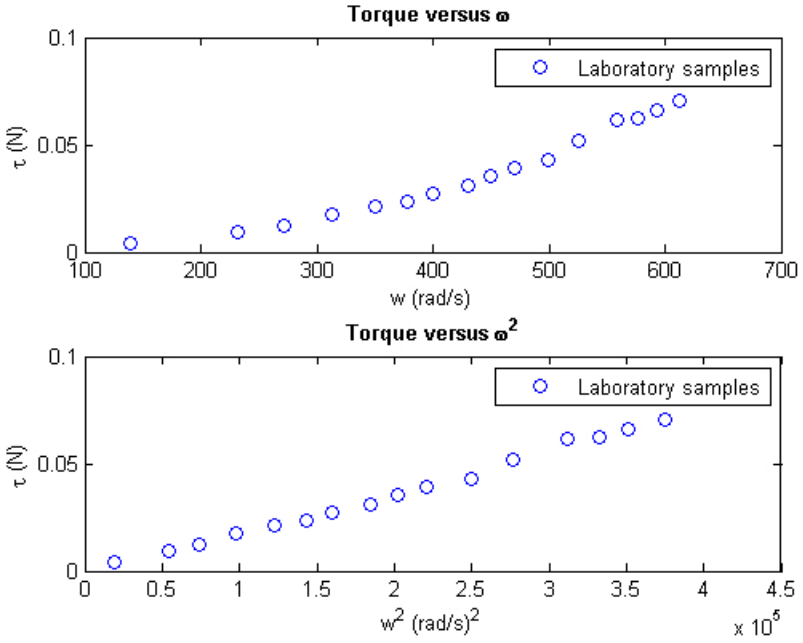

Figure A.5: Graph of the samples of torque with ω and ω^2

Matlab function *polyfit(x, y, n)* is used to compute the n^{th} order polynomial that fits with the samples. A first order polynomial (linear relationship) is used for ω^2 and 2nd order for ω.

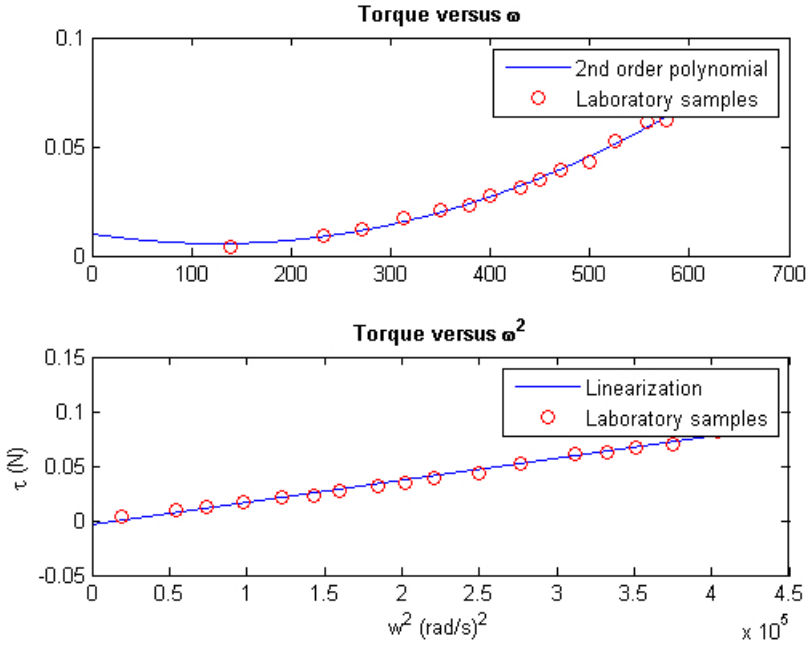

Figure A.6: Torque and ω^2 with its linear approximation

The linear approximation gives:

$$\tau = 2.0266 \cdot 10^7 \omega^2 \tag{A.4}$$

which leads to $K_\tau = 2.0266 \cdot 10^7$.

A.4 PWM Test

Last experiment consist on measuring PWM signal and the angular velocity produced. A Matlab script has been used to plot the values and find values K_τ and PWM_0 according to the estimation of the slope and offset of the first order polynomial that fits the samples. The variation of PWM respecting ω^2 is shown. A linear relation is observed.

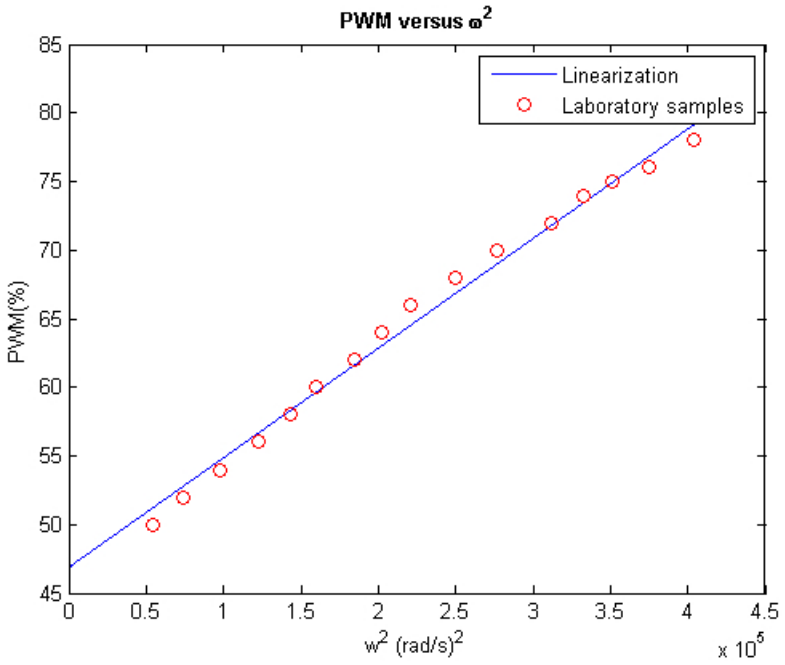

Figure A.7: PWM and ω^2

The polynomial found has the following form:

$$PWM = 7.9855 \cdot 10^{-5}\omega^2 + 46.8459 \tag{A.5}$$

which leads to $PWM_0 = 46.8459$ and $K_{PWM} = 7.9855 \cdot 10^{-5}$.

BIBLIOGRAPHY

[1] Kalman R. E. (1960). A new approach to linear filtering and prediction problems.

[2] Swokowski Earl (1979). Calculus with analytic geometry.

[3] S. Bouabdallah (2007). Design and control of quadrotors with applications to autonomous flying.

[4] Bo Lincoln Johan Eker-Karl-Erik Årzén Anton Cervin, Dan Henriksson. How does control timing affect performance analysis and simulation of timing using jitterbug and truetime.

[5] Fogarty.T Armoogum.V, Soyjaudah.K.M.S and Mohamudally.N. Comparative study of path loss using existing models for digital television broadcasting for summer season in the north of mauritius.

[6] Thomas Bak. Lecture notes - modeling of mechanical systems.

[7] Datasheet. Xbee pro rf modules. `https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf`.

[8] Pixhawk Developers. Pixhawk autopilot. `https://pixhawk.org/modules/pixhawk`.

[9] Roy Featherstone. Rigid body dynamics algorithms.

[10] Abbas Emami-Naeini (2010) Gene F. Franklin, J. David Powell. Feedback control of dynamic systems, 6ed.

[11] P. Dananjayan K.Ayyappan. Propagatin model for highway in mobile communication system.

[12] Hermann Kopetz. Design principles for distributed embedded applications.

[13] Steven M. Lavelle. Planning algorithms. `http://planning.cs.uiuc.edu/node102.html`.

[14] Franz S. Hover Michael S. Triantafyllou. Maneuvering and control of marine vehicles. `http://ocw.mit.edu/courses/mechanical-engineering/2-154-maneuvering-and-control-of-surface-and-underwater-vehicles-13-49-fall-2004/lecture-notes/1349_notes.pdf`.

[15] Department of Aeronautics and Astronautics. Pole placement approach, lecture notes, feedback control systems. http://ocw.mit.edu/courses/mechanical-engineering/2-154-maneuvering-and-control-of-surface-and-underwater-vehicles-13-49-fall-2004/lecture-notes/1349_notes.pdf.

[16] University of Guelph. Moment of intertia. https://www.physics.uoguelph.ca/tutorials/torque/Q.torque.inertia.html.

[17] J. Goslinski W. Giernacki P. Gasior, S. Gardecki. Estimation of altitude and vertical velocity for multirotor aerial vehicle using kalman filter.

[18] T. Perez. Ship motion control:course keeping and roll stabilisation using rudder and fins.

[19] Syed Ali Raza and Wail Gueaieb. Intelligent flight control of an autonomous quadrotor, motion control, isbn: 978-953-7619-55-8, intech.

[20] Dave (2007) Roos. How wireless mesh networks work. http://computer.howstuffworks.com/how-wireless-mesh-networks-work.htm.

[21] Bernard Sklar. Digital communications: Fundamentals and applications.

[22] Gregory G. Slabaugh. Computing euler angles from a rotation matrix. http://staff.city.ac.uk/sbbh653/publications/euler.pdf.

[23] Murat Torlak. Pathloss, telecom, switching and transmission.