Design and Implementation of Flight Dynamics Control Strategies for a Smartphone-based Quadrotor

Thesis for obtaining the degree of

MASTER OF SCIENCE IN ENGINEERING
with emphasis in Automation

Alejandro Astudillo Vigoya
alejandro.astudillo@correounivalle.edu.co

School of Electrical and Electronic Engineering
UNIVERSIDAD DEL VALLE
Cali, COLOMBIA

December 4, 2017

Supervised by:

Dr.-Ing. Esteban Rosero
Industrial Control Research Group - GICI
School of Electrical and Electronic Engineering
Universidad del Valle

Bladimir Bacca Ph.D.
Perception and Intelligent Systems Research Group - PSI
School of Electrical and Electronic Engineering
Universidad del Valle

Abstract

Aerial robotics is a field in constant growth, both within the scientific community and in commercial applications. There are more and more existing development platforms, with which an unmanned aerial vehicle can be built without making large economic investments. On the other hand, smartphones have become devices with great computing and sensing capacity, allowing various scientific and research projects to be developed using these devices. It is then possible to think about integrating a smartphone as a processing and sensing device within an unmanned aerial system

This work aims to develop and implement control and estimation strategies which can be executed in a smartphone in order to control the flight dynamics of a rotary-wing unmanned aerial vehicle, specifically a quadrotor. Initially, the concept of unmanned aerial vehicle, its characteristics and classifications are introduced. Subsequently, the mathematical model of a quadrotor dynamics is obtained based on the Newton-Euler and Euler-Lagrange approaches. The components of the smartphone-based quadrotor are detailed and their specific parameters are experimentally identified. Two optimal control strategies are designed using the linearized model of the quadrotor. These control strategies are the linear quadratic regulator with integral feedback (LQI) and the H_∞ controller. In addition, a Kalman filter is designed to estimate the non-measurable states of the system. After verifying using simulations, that the designed controllers are functional within the quadrotor, the flight controller is implemented in Android. This flight controller includes the control, state estimation and communication algorithms. Additionally, a desktop application is implemented to monitor and configure the quadrotor during its flight. Finally, the quadrotor is subjected to multiple tests in its flight modes, using both implemented controllers. The results suggest that the H_∞ controller performs better than the LQI controller in the developed smartphone-based quadrotor prototype.

Keywords— Quadrotor, Smartphone, Optimal control, H_∞ controller, Linear quadratic controller

Resumen

La robótica aérea es un campo en constante crecimiento, tanto dentro de la comunidad científica, como en aplicaciones comerciales. Cada vez son más las plataformas de desarrollo existentes, con las cuales puede construirse un vehículo aéreo no tripulado sin realizar grandes inversiones económicas. Por otra parte, los teléfonos inteligentes se han convertido en dispositivos con gran capacidad computacional y de sensado, permitiendo que se desarrollen diversos proyectos científicos y de investigación utilizando estos dispositivos. Es posible pensar en integrar un smartphone como dispositivo de procesamiento y sensado dentro de un sistema aéreo no tripulado.

Este trabajo tiene como objetivo desarrollar e implementar estrategias de control y estimación de estados que puedan ser ejecutadas en un teléfono inteligente con el fin de controlar las dinámicas de vuelo de un vehículo aéreo no tripulado de ala rotatoria, específicamente un quadrotor. Inicialmente, se introduce el concepto de vehículo aéreo no tripulado, sus características y clasificaciones. Posteriormente, se obtiene el modelo matemático de las dinámicas de un quadrotor basandose en las aproximaciones de Newton-Euler y Euler-Lagrange. Los componentes del quadrotor basado en smartphone son detallados y sus parámetros específicos son identificados experimentalmente. Dos estrategías de control óptimo son diseñadas a partir del modelo linealizado del quadrotor. Estas estrategías de control son el regulador cuadrático lineal con realimentación integral (LQI) y el controlador H_∞. Además, se diseña un filtro de Kalman para estimar los estados no medibles del sistema. Tras verificar por medio de simulaciones, que los controladores diseñados son funcionales dentro del quadrotor, se implementa en Android el controlador de vuelo. Este controlador de vuelo incluye los algoritmos de control, de estimación de estados y de comunicación. Adicionalmente, se implementa una aplicación de escritorio para el monitoreo y configuración del quadrotor durante el vuelo. Finalmente, el quadrotor es sometido a múltiples pruebas en sus modos de vuelo, utilizando ambos controladores implementados. Los resultados sugieren que el controlador H_∞ tiene un mejor desempeño que el controlador LQI en el prototipo desarrollado de quadrotor basado en smartphone.

Palabras Clave— Quadrotor, Smartphone, Control óptimo, Controlador H_∞, Controlador lineal cuadrático

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the interest in unmanned aircraft vehicles (UAVs) research, has increased substantially. This is due to the several potential new services that this type of robotic devices offers, such as search and rescue, observation, mapping, inspection, etc. On the other hand, smartphones have become essential devices for humans and easily acquirable development tools. The interaction between these two technologies allows the development of low cost aerial robots based on an everyday item such as smartphones, facilitating both the distribution of its control software and its implementation by other researchers.

1.1 Motivation

Quadrotor control is a difficult and interesting problem. As established by [1] and [2], quadrotor dynamics are affected by nonlinearity, parameters perturbations, uncertainties and disturbances: this include unknown and variable payloads, aerodynamical parameters of the system, wind changes, and sensors inaccuracies. Numerous studies have been developed in designing optimal and robust controllers that allow UAVs to fly and accomplish missions rejecting disturbances and being robust to parameter uncertainties as seen in [3, 4, 5, 6].

Although there are embedded systems with high computational capacity that can serve as controllers of a quadrotor, smartphones are available, easily accessible for people and also have a large computational capacity, hence multiple instrumentation and communication elements integrated in the same device. In the last years, computing capacity and sensor technology in smartphones have decreased in price but increased in performance. Smartphones have become an inexpensive tool capable of commanding an UAV. The challenge then, is to use smartphones as quadrotor flight

controllers for autonomous flights following specific missions, taking advantage of the fact that the phones today are very powerful computers that include elements of sensing, processing and signal communication.

This research project aims to design and implement algorithms that will be executed in a smartphone to estimate and control the dynamics of a quadrotor. This project confronts several challenges such as using a smartphone as a hardware development platform, trying to use a non real-time operating system for real-time applications, designing optimal and high order controllers using Java, and executing that controllers in a smartphone.

1.2 Unmanned Aircraft Vehicles (UAV)

An UAV is an aircraft capable of flying without having a human on board. This aircrafts are commonly known as 'drones' due to the similarity of the sound they produce with that of a male bee (drone). Although they can perform flight missions in a completely autonomous way, the UAVs are complemented with remote display and control systems (Ground Control Station - GCS), and communication systems between the UAV and the GCS. These three components describe what is known as Unmanned Aircraft Systems (UAS).

The UAS were initially designed for military applications, where the presence of a human could be of high risk [7]. However, their use has been widely extended to multiple applications such as education [8], agriculture [9], search and rescue [10], delivery [11], research [12], among others.

UAVs are modelled as rigid bodies with freedom of movement in a three-dimensional space, hence have six degrees of freedom (DoF), three translational (position) and three rotational (attitude). These aircrafts are classified mainly according to the propulsion direction exerted by the actuators. Thus, there are two groups of UAVs: Fixed-wing and Rotary-wing UAVs, which are described below.

1.2.1 Fixed-Wing UAVs

A fixed-wing UAV is an aircraft which consists of an airframe attached to a rigid wing that generate lift using the UAVs forward airspeed and geometry. This type of UAVs need a constant forward movement in order to produce air flow around its wing, which can be generated by the thrust of propeller pushing the air flow in opposite direction of the UAV movement, or flying against to the wind. An example

of a fixed-wing UAV is shown in Fig. 1.1.

Figure 1.1: Skywalker X8 fixed-wing UAV[1]

Since they need a minimum airspeed to achieve their lift in flight, these aircrafts require a runway or catapult for take-off and landing. However, its geometry allows to achieve long-term flights (greater than 60 minutes) at high speeds (greater than 20 m/s) by taking advantage of its aerodynamic efficiency. Also, they are capable of carrying big and heavy payloads without significantly affecting the duration of the flight.

The six DoF of a fixed-wing UAV are controlled using elevators, ailerons and rudders, which are control surfaces built in the wing. The use of the control surfaces allows rotations to be made about any of the three perpendicular axes that intersect at the UAV center of gravity (CoG), and therefore control the rotational DoF. The translational DoF are directly affected by the rotations that the UAV suffers and its forward airspeed, and therefore can be indirectly controlled using the aforementioned control surfaces.

1.2.2 Rotary-Wing UAVs

Rotary-wing UAVs consist of one or multiple propellers attached to an airframe, generating an air flow and therefore generating the necessary thrust to move while overcoming the gravitational force. This UAVs are capable of doing a vertical take-off and landing (VTOL), so it is not necessary to use a catapult or runway for it. The number of rotors in the UAV determines the classification to which they belong, which usually includes: helicopter (one rotor), trirotor (three rotors), quadrotor (four rotors), hexarotor (six rotors), and octarotors (eight rotors), as the one shown in Fig. 1.2.

[1]Fixed-wing UAV image taken from https://goo.gl/LNEriU

Figure 1.2: Altavian octorotor rotary-wing UAV[2]

The main advantage of the rotary-wing UAVs is that they do not need to be constantly moving forward in order to have enough lift that sustain the aircraft in flight, as their propellers rotation generates the lift force by itself. This make the UAV capable of sustaining itself in a desired position.

Rotary-wing UAVs do not have control surfaces that change the direction of the air flow in order to control its six DoF. Instead, it uses the unbalance of thrust exerted by its motors and propellers to control the rotations of the rigid body and indirectly control its horizontal translational movement. As the motors thrust is always pushing the airframe vertically, the vertical translational movement is independently controlled by the total thrust exerted by all the UAVs motors.

Figure 1.3: Quadrotor aircraft X650 [3]

[2]Rotary-wing UAV image taken from https://goo.gl/WPVYAg
[3]Quadrotor XAircraft X650 image taken from https://goo.gl/ug5FJs

Due to the simplicity in its construction, the most common small-size rotary-wing UAV type developed by manufacturers and hobbyists, is the quadrotor. The quadrotors, also known as quadcopters, are typically based on a light airframe built with fiberglass or carbon fiber, and four motors that serve as actuation devices, as shown in Fig. 1.3. This aircrafts can be built in a relatively simple and inexpensive manner, besides being easy to control and model, when being compared with other multirotors. This is the reason why a quadrotor was established as the experimental platform in this project.

1.3 Literature Review

1.3.1 Control Strategies and State Estimation in Quadrotors

Since UAVs represent a challenge for the modelling and design of controllers, from basic to advanced, these aircrafts are widely used in the teaching and research of control systems theory. For instance, [13] implemented a PID controller (with two tuning methods) and a Linear-Quadratic regulator (LQR) for altitude control in a quadrotor, comparing them and establishing the advantages of using a PID controller tuned using the LQR equations. Moreover, [14] exposes the develop of a low cost quadrotor attitude controler based on a PID control strategy with saturation constraints, achieving the quadrotor attitude stabilization with set-points of less than 0.23 rad. Another comparison between PID and LQR control strategies is done in [15]. In this case, both controllers are used for trajectory tracking in a Qball-X4 quadcopter, getting smoother results with the LQR implementation.

In [16], a quaternion-based LQR gain scheduling controller is designed and simulated, successfully achieving six DoF control of the quadrotor. A position and heading regulation system using a LQR is also presented in [17]. Here, the implementation of the designed controller is carried out, in which the altitude regulation with oscillations of 0.1 m is observed. On the other hand, [18] designed a feed-forwarding LQR, which achieve to control quadrotor attitude with errors of less than 0.08 rad.

A trajectory tracking controller based on a LQR and an Extended Kalman Filter (EKF) is shown in [19]. This approach use the non-linear quadrotor dynamics model to estimate the quadrotor states and calculte the control signals based on the estimations.

Another quadrotor control strategy found in literature is the Model Predictive Control (MPC). In [20], a MPC controller for trajectory tracking is compared to a PID controller, getting an improvement in setting time and overshoot. [21] proposed a non-linear quaternion based control law complemented with an active disturbance rejection strategy that attempts to control the attitude of a quadrotor.

In [22], a Linear Parameter-Varying (LPV) control strategy is developed. Here, an adaptive control is implemented in order to control independently a family of multi-sized quadrotors using the same controller approach. On the other hand, [23] proposes a fuzzy adaptive sliding mode controller that aims to overcome actuators fault within the quadrotor attitude control system. Another adaptive control strategy was developed in [24], [25] and [26], where parameters uncertainties were taken into account in order to control variable payload quadrotors.

Finally, in [27] and [28], the authors designed H_∞ controllers for reference tracking in quadrotors, evidencing the advantages of the implementation of this controller in terms of robustness.

Regarding about state estimation in quadrotors, after carrying out the literary review, it was found that the Kalman filter (KF) is a widely used estimation algorithm in all its different variants. For instance, in[29], a KF for altitude estimation based on the model of a moving particle was developed. A two stage KF for both quadrotor state and parameter estimation is presented in [30]. This KF is designed in order to detect and identify faults in the quadrotor actuators. As the quadrotor dynamic model is strictly non-linear, the EKF is a popular variation of the KF used in multiple ongoing research, such as[31], [32], and [33]. Other variation of the KF is the Unscented Kalman Filter (UKF) which uses a deterministic sampling approach in order to estimate the state of a non-linear model. In [34], an UKF is used to estimate the orientation of a quadrotor on a test bench, comparing its estimations with the measurements of an incremental encoder and the estimations of a standard KF. The results of this comparison show that the UKF estimations are more similar to the measured orientation, than the estimates of KF.

In [35] a model-free technique implemented within a Thau observer is presented. This state estimator aims to compensate uncertainties and unmodeled dynamics. Also, other projects have used on-board cameras, for the acquisition of extra data that improve the state estimation, as seen in [36], [37] and [38], demonstrating that the use of vision-based odometry improves the quadrotor state estimation despite depending on inaccurate sensors.

1.3.2 Quadrotor Flight Modes

In quadrotors, the on-board flight controllers keep some of the quadrotors DoF in a desired value autonomously in order to allow pilots to perform tasks during a flight. This controllers have different modes that can control from three to six DoF depending on the will of the pilot. Flight modes commonly found in commercial flight controllers may be as basic to only control its attitude or as complex to let the quadrotor follow a complex trajectory with multiple waypoints [39].

The main flight modes, widely used in commercial flight controllers, are described bellow according to the number of controlled DoF, in ascending order.

- **Stabilize Mode**

  This mode allows the pilot to fly the quadrotor manually while the flight controller self-levels the quadrotor attitude and regulate its current heading. Thus, the stabilize mode attempts to control three DoF of the quadrotor.

  The attitude references can be set or changed by the pilot using the remote control, but their default value is 0 rad. On the other hand, the quadrotor heading is simply set to be regulated in its current state, enabling its rate using the remote control.

  As the stabilize mode do not take into account the control of the quadrotor position, the pilot needs to regularly change the attitude references manually to keep the quadrotor in a desired position, as it is affected by wind disturbances. Also, the pilot needs to regularly adjust the quadrotor thrust, so a desired altitude is maintained.

- **Altitude Hold Mode**

  The altitude hold mode adds automatic altitude control to the Stabilize mode. This way, in addition to controlling the attitude, the quadrotor thrust is set by the flight controller in order to maintain the quadrotor in a desired altitude, getting four controlled DoF.

  In this mode, the pilot can remotely control the rate of change of the altitude (with a default value of 0 m/s), as well as the attitude references.

- **GNSS-Dependent Flight Modes**

  The Global Navigation Satellite System (GNSS)-Dependent flight modes, are those that automatically attempt to regulate the six DoF of the quadrotor in order to maintain a desired position, heading and altitude during a flight.

The main GNSS-Dependent Flight Modes are: Loiter mode, Auto Mode, and Return-To-Launch (RTL) Mode.

In Loiter mode, the quadrotor attitude is self-leveled, while the position and altitude reference can be modified by the pilot using the remote controller. The position and altitude references are initialized using the current quadrotor position and altitude when this mode is set.

The Auto mode attempts to make a quadrotor follow automatically a preprogrammed path connecting multiple position and heading waypoints. This mode use the same controller as the Loiter mode, but its references are set automatically following a waypoints list.

During a flight mission, the home location is set as the position and altitude where the quadrotor took off. The RTL mode is used in case of emergency or when a the last waypoint is reached within a flight mission. This mode is equivalent to the Auto mode, but only has two waypoints. The first waypoint consists in the position of the home location with a previously set security altitude (RTL altitude) greater than the take-off altitude. When this waypoint is reached, the home location is set as the following waypoint so the quadrotor starts its landing, while keeping the position controlled.

1.3.3 Smartphones in Control Systems

Current smartphone processors are able to perform complex calculations such as those required in the implementation of real time control strategies. Multiple research projects have been carried out using smartphones as tools for development of control systems in robotics.

A survey about the trend of using smartphones as main processing component in robotics for research and education was carried out in [40]. In [41], it is proposed a low cost differential robot controlled using a smartphone as processing and sensing device, taking advantage of the ROS framework for Android. [42] expose the development of an adaptive cruise control algorithm for an object-following robot controlled by a smartphone on board. Also, an autonomous smartphone-based robot platform for football competitions, exposed in Fig. 1.4, was designed and shown in [43].

Figure 1.4: Smartphone-based ball tracking robot [4]

There are many ongoing research related to the possibility of using smartphones to implement control strategies, such as [44], as configuration and monitoring interfaces in control systems as seen in [45, 46, 47], and as a tool in both education and design of control strategies seen in [48, 49]. Following this trend, in the Universidad del Valle, it was developed a smartphone-based platform for monitoring, control and communication in portable laboratories, where a controller for a pendulum, based in the Lego Mindstorms EV3 platform, was implemented [50].

1.3.4 Smartphone-based Quadrotors

Multiple attempts to unite the technologies of quadrotors and smartphones have been made. Some of this attempts are described below.

In [51], the possibility of using old discarded smartphones as sensors and processor in a quadrotor was studied. In [52], a smartphone was used as mission planner for a quadrotor using a commercial flight controller on board. [53] implemented a flight control system in a smartphone, using its sensors and computational power to stabilize the quadrotor attitude and control its altitude. Also, in the University of Pennsylvania [54], a Google Tango smartphone was used as a quadrotor flight controller, including an image-based positioning system based on RGB-D images captures. The state estimation algorithms, control and planning were firstly implemented in a ODROID-XU board with additional sensors, but then, in [55], this

[4] Smartphone-based ball tracking robot image taken from [43]

algorithms were ported to the processor of the updated smartphone used in the project. The quadrotor developed in this project is shown in Fig. 1.5.

Figure 1.5: UPenn's smartphone-based quadrotor [5]

In [56], a PID controller aimed to be executed in a smartphone on board a quadrotor, which is also responsible for estimating the rotational dynamics of the quadrotor using the measurements made by the smartphone, was implemented. Following this project, [57] exposes in detail the development of a model-free PID controller for a smartphone-based quadrotor, including the processing of delayed feedback in the controller. After that, in [58], a Linear-Quadratic-Integral (LQI) controller was implemented and compared with the model-free PID controller set before. Finally, [59] developed a non-linear complementary filter for position and attitude estimation in a smartphone-based quadrotor.

Smartphone-based Quadrotor Limitations

The idea of using a smartphone as flight controller in an UAV opens the possibility of a quick and inexpensive development [59]. A smartphone offers other advantages compared with off-the-shelf flight controllers, for instance its powerful quad, hexa or octa-core processors and communications interfaces. However, smartphones and the Android operating system have some limitations that set challenges when implementing a control system in it.

Android is not a real-time operating system and therefore, can not assure execution of algorithms, like estimation and control, with a constant sample time. Furthermore, the sensors embedded in commercial smartphones are made for applications that do not have high requirements of accuracy nor precision, and therefore may

[5] UPenn smartphone-based quadrotor image taken from [55]

not be appropriate for sensing quadrotor dynamics. Nonetheless, as explained by [60], due to its computing capabilities, smartphones can overcome this limitations while using a temporized thread to execute the control system algorithms and implementing a sensor fusion technique to improve the states estimation reliability. This thread must be executed with a lower sample time compared to the one of the sensors embedded in the smartphone. This will ensure that the execution is not delayed by the sensors acquisition process.

1.4 Research Problem

The existing research challenges include how to develop and implement efficient control algorithms for smartphones using the Android operating system, and assess, adapt and develop the appropriate communication, sensing and performance technologies with smartphones in the execution of missions using quadrotors.

Then, the question to be answered is: how to develop control strategies in a smartphone in order to control the flight dynamics of a quadrotor so it can develop flight missions while the instrumentation and computing capacity of the smartphone is used?

1.5 Objectives

In order to find a solution for the research problem, the following general and specific objectives are proposed:

General Objective

Design and implement algorithms for control and estimation of flight dynamics executed in a smartphone for the quadrotor of the Industrial Control Research Group.

Specific Objectives

1. Conduct a study and analysis of the state of the art related to the control and estimation of states of quadrotors.

2. Integrate the existing quadrotor with a smartphone that contains the appropriate sensors for the control and estimation of states.

3. Obtain a dynamic model of the quadrotor.

4. Design and implement the algorithms for control and estimation of states for the quadrotor.

5. Integrate the experimentation platform with the control and estimation algorithms in the smartphone.

6. Evaluate the performance of the control strategies.

1.6 Outline

This thesis is organized as follows.

In Chapter 1, the definition and classification of UAVs are given. After that, a short literature review regarding control and estimation in quadrotors, flight modes, and smartphones in control systems, is detailed. Finally, the motivation and scope of this project are presented.

The dynamic model of the quadrotor is described in Chapter 2. Here, the nonlinear and linearized quadrotor model are obtained taking into account the main quadrotor geometry configurations and their inputs setting.

Chapter 3 focuses on the smartphone-based quadrotor prototype description, detailing all its components and their interactions, as well as the specific parameters of the prototype.

The control and estimation algorithms design is presented in Chapter 4. In this chapter, specific controllers for each quadrotor flight mode are designed and simulated.

In Chapter 5, the results of the implementation of the control and estimation algorithms, as well as the description of the Android application and the GCS are detailed.

Finally, in Chapter 6, the thesis conclusions and suggestions for future developments and improvements, are shown.

Chapter 2

Dynamic Model of the Quadrotor

In this chapter, the dynamics of the quadrotor and the calculation of the mathematical model through two different approaches, are provided. In addition, the necessary considerations for the choice of a geometry configuration are exposed. This dynamic model and its inputs setting are needed to design the quadrotor flight controllers.

Section 2.1 presents the description of two of the main used quadrotor geometry configurations. Also, this section presents what it means to choose one or the other configuration for the setting of the quadrotor inputs.

The dynamic model of the quadrotor, obtained using the Newton-Euler and Euler-Lagrange approaches, neglecting the gyroscopic effects produced by the propellers, is shown in Section 2.2.

Finally, Section 2.3 describes an overview of the Jacobian linearization method applied to the quadrotor dynamic model and the thrust compensation needed to be implemented in a real quadrotor.

2.1 Quadrotors Configurations

The term 'quadrotor' refers to a rotary-wing UAV which thrust is generated using four motors and propellers. Quadrotors can be build in multiple ways. There are two basic configurations which are widely used by commercial manufacturers and hobbyists. These two configurations are the '+' and the 'X'.

2.1.1 The '+' Configuration

The geometry used in quadrotors built in '+' configuration is shown in Fig. 2.1.

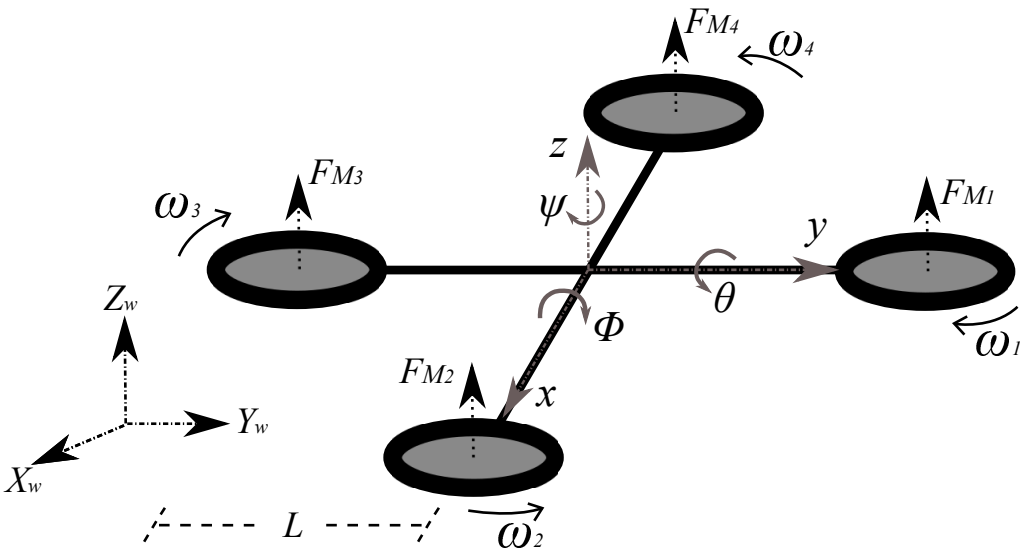


Figure 2.1: Quadrotor geometry in '+' configuration

In Fig. 2.1, (ϕ, θ, ψ) are the angular deviations (pitch, roll and yaw, respectively) of the quadrotor about the body-frame; (x, y, z) are the body-frame axes; (X_W, Y_W, Z_W) are the earth-frame axes; L is the distance from the quadrotor center of gravity (CoG) to the motors center; F_{M_i} is the thrust force exerted by each motor M_i; and ω_i is the angular velocity of M_i, with $i = 1, 2, 3, 4$. In this configuration, the body-frame axes x and y, coincide with the lines that connect motors of opposite sides in the quadrotor frame.

The quadrotor is a system with six DoF. Since a quadrotor has four actuators, it can be considered an underactuated system. This means that, it is only possible to reach a desired state for four DoF. However, it is possible to indirectly control the two remaining DoF choosing the control signals properly. These control signals, or inputs, represent the quadrotor basic movements and are described below.

- **Thrust** T_u $[N]$

  The thrust T_u is the total thrust force exerted parallel to the body-frame z-axis by the four motors. This thrust, can also generate accelerations in the direction of the X_W and Y_W axes. This happens when $|\theta| > 0$ or $|\phi| > 0$.

This control signal affects the speed of rotation of all motors, and therefore their thrust, in equal magnitude, and is set as

$$T_u = \sum_{i=1}^{4} F_{M_i}. \tag{2.1}$$

- **Yaw Torque** τ_ψ $[N \cdot m]$

From Fig. 2.1, M_1 and M_3 have a clockwise (CW) rotation while M_2 and M_4 rotate counter-clockwise (CCW). This configuration of opposite pairs rotational directions allows the system to control its conservation of momentum, and thus change its yaw angle in a controlled manner. This is done unbalancing the total momentum around the z-axis and without the need of a tail rotor used in the standard helicopter structure [61]. In '+' configuration, the fact that the pitch and roll angles are controlled using only two motors that rotate in the same direction, leads to large changes in the thrust force of the other two motors to achieve conservation of momentum.

This conservation of momentum is controlled using the torque generated by each motor (τ_{M_i}) around the body-frame z-axis. It produces a turn in the opposite direction to the rotation of the motor. Taking into account that the torque τ_ψ is positive when it generates a clock-wise rotation around the z-axis, only M_2 and M_4 contribute positively to it. While the other tow motors have a negative contribution to τ_ψ. Thus, the total torque around the z-axis is depicted as

$$\tau_\psi = \tau_{M_2} + \tau_{M_4} - \tau_{M_1} - \tau_{M_3}. \tag{2.2}$$

Each torque τ_{M_i} has a linear relationship with the thrust applied by the motor, with K_M being the proportional constant and

$$\tau_{M_i} = K_M F_{M_i}. \tag{2.3}$$

Replacing (2.3) in (2.2) the total torque τ_ψ dependence on the forces F_{M_i} is got as

$$\tau_\psi = K_m(F_{M_2} + F_{M_4} - F_{M_1} - F_{M_3}). \tag{2.4}$$

- **Roll Torque** τ_θ $[N \cdot m]$

The roll torque τ_θ is described as the torque exerted on the x-axis and about the y-axis. In the '+' configuration, the only motors that affect the quadrotor rotation with respect to the x-axis are M_2 and M_4. These motors do not affect the rotation with respect to the y-axis. In this case, the torques are generated by the forces F_{M_2} and F_{M_4} being applied at a distance L from the quadrotor CoG. Considering a positive τ_θ as the one causing a counter clock-wise rotation about the y-axis, the roll torque in '+' is set as

$$\tau_\theta = L(F_{M_4} - F_{M_2}). \tag{2.5}$$

- **Pitch Torque** τ_ϕ $[N \cdot m]$

  Unlike the roll torque, the pitch torque τ_ϕ is the one exerted on the y-axis about the x-axis. As Fig. 2.1 shows, M_1 and M_3 are the motors applying the forces that create this torque. Being τ_ϕ positive when clock-wise rotation about the x-axis is caused, it is defined for a '+' configuration as

$$\tau_\phi = L(F_{M_3} - F_{M_1}). \tag{2.6}$$

Inputs Setting in '+' Configuration

Summarizing, the '+' configuration quadrotor inputs are established as linear combinations of the motors forces F_{M_i}. Equations (2.1), (2.4), (2.5) and (2.6) integrate a linear equations system where the vector of inputs $\mathbf{u}_{(+)}$ is

$$\mathbf{u}_{(+)} = \begin{bmatrix} T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -K_m & K_m & -K_m & K_m \\ 0 & -L & 0 & L \\ -L & 0 & L & 0 \end{bmatrix} \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ F_{M_3} \\ F_{M_4} \end{bmatrix}. \tag{2.7}$$

2.1.2 The 'X' Configuration

Following the same nomenclature used in the '+' configuration, in 'X' configuration, the quadrotor frame is rotated $\pi/4\ rad$ about the z-axis in the body-frame, as shown in Fig. 2.2.

In this case, the front-line in the quadrotor is set between M_1 and M_4. In 'X' configuration, the roll and pitch torques are applied using the forces exerted by all the motors at a distance $L_X = L \cdot \cos(\pi/4)$ [62]. This geometry is shown in Fig. 2.2.

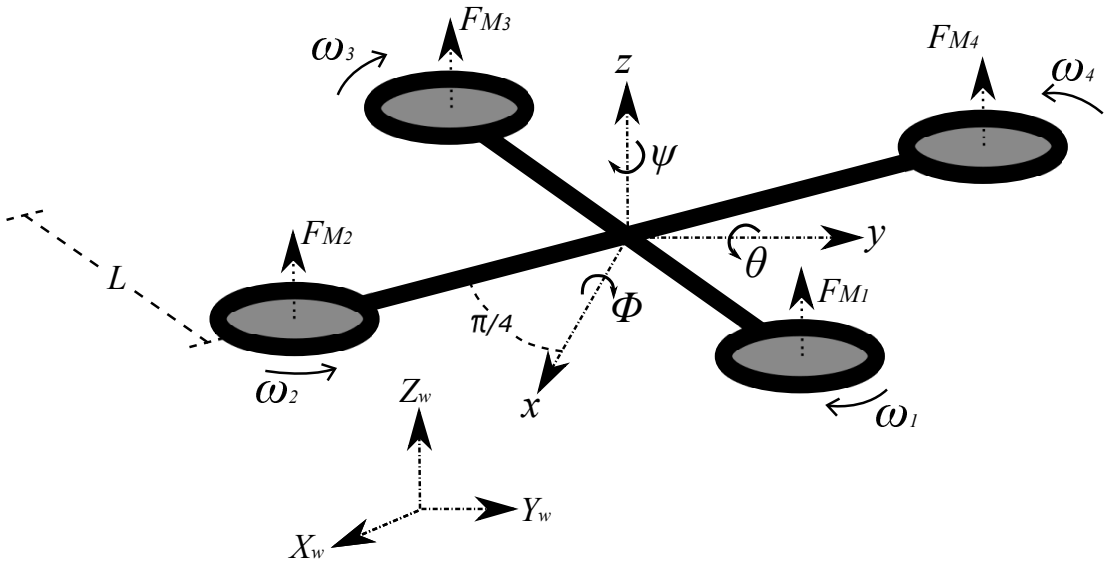


Figure 2.2: Quadrotor geometry in 'X' configuration

The inputs description for the quadrotor 'X' configuration is defined as follows.

- **Thrust** T_u $[N]$

  The total thrust exerted by the four motors in the quadrotor is not changed between the '+' and 'X' configurations. Hence, the thrust T_u is defined as

$$T_u = \sum_{i=1}^{4} F_{M_i}. \tag{2.8}$$

- **Yaw Torque** τ_ψ $[N \cdot m]$

  As the z-axis is not modified between the '+' and 'X' configurations, and the motors rotate in the same direction as in the other configuration, the yaw torque τ_ψ is set as

$$\tau_\psi = K_m(F_{M_2} + F_{M_4} - F_{M_1} - F_{M_3}). \tag{2.9}$$

- **Roll Torque** τ_θ $[N \cdot m]$

  As shown in Fig. 2.2, roll and pitch torques are affected by the effects of

all the quadrotor motors. For the roll torque τ_θ, the motors M_3 and M_4 contribute positively, while M_1 and M_2 have a negative contribution to it. In this case, the roll torque is depicted as

$$\tau_\theta = L_X(F_{M_3} + F_{M_4} - F_{M_2} - F_{M_1}). \tag{2.10}$$

- **Pitch Torque** τ_ϕ $[N \cdot m]$

  Due to the CW positive direction of the ϕ angle, and considering the sign of the contribution exerted by each force F_{M_i}, the pitch torque for the 'X' configuration is defined as

$$\tau_\phi = L_X(F_{M_2} + F_{M_3} - F_{M_1} - F_{M_4}). \tag{2.11}$$

Inputs Setting in 'X' Configuration

In the 'X' configuration, the quadrotor inputs remain the same regarding the '+' configuration. However both τ_θ and τ_ϕ are affected by the interaction of all the motors and the change in the distance of the point of application of the forces F_{M_i} on the x and y axes. The linear equations system that shows the inputs setting for a quadrotor in 'X' configuration is defined as

$$\mathbf{u}_{(\mathbf{X})} = \begin{bmatrix} T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -K_m & K_m & -K_m & K_m \\ -L_X & -L_X & L_X & L_X \\ -L_X & L_X & L_X & -L_X \end{bmatrix} \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ F_{M_3} \\ F_{M_4} \end{bmatrix}. \tag{2.12}$$

Maximum Torque About x and y axes in 'X' Configuration

In '+' configuration, the torques around the x and y axes are set using just two motors applying a force at a distance L from the quadrotor CoG. This implies that, for instance, in the case of roll torque τ_θ the maximum torque $\tau_{\theta max(+)}$ is achieved when $F_{M_4} = F_{M_i max}$ and $F_{M_3} = 0$, where $F_{M_i max}$ is the maximum thrust of the motors. Then, in '+' configuration, the $\tau_{\theta max(+)}$ is set as

$$\tau_{\theta max(+)} = L \cdot F_{M_i max}. \tag{2.13}$$

On the other hand, the torques around the x and y axes in 'X' configuration depend on the forces of the four motors in the quadrotor, which are applied at a distance $L_X = L \cdot \cos(\pi/4)$ from the quadrotor CoG. Continuing with the example of the roll torque, in 'X' configuration the maximum torque $\tau_{\theta max(X)}$ is achieved when

$F_{M_3} = F_{M_4} = F_{M_i max}$ and $F_{M_1} = F_{M_2} = 0$. Hence, the maximum torque about the y-axis in the 'X' configuration is

$$\begin{aligned} \tau_{\theta max(X)} &= L \cdot \cos(\pi/4) \cdot 2 \cdot F_{M_i max}, \\ \tau_{\theta max(X)} &= 2 \cdot \cos(\pi/4) \cdot \tau_{\theta max(+)}. \end{aligned} \tag{2.14}$$

Thereby, the quadrotor in 'X' configuration has $2 \cdot \cos(\pi/4)$ times more available torque to rotate about the x and y axes, when compared with the '+' configuration, and therefore it can achieve 41.42 % more rotational acceleration about the x and y axes.

2.2 Non-linear Model

This section describes the dynamic modeling used to develop the quadrotor control, based on the study carried out in [61] and [7]. This model represents the quadrotor as a solid symmetrical object, subject to a total thrust (T_u) and three torques (τ_ψ, τ_θ and τ_ϕ), assuming that the quadrotor CoG coincides with the origin of the body-frame and without considering the dynamics of the actuators. The modeling of the quadrotor system is done by two different methods. First, the Newton-Euler approach which is based on the quadrotor body-frame; and second, the Euler-Lagrange approach which is based on the earth-frame for its translation while keeping its rotational equations related to the body-frame.

2.2.1 Newton-Euler Approach

Following the quadrotor geometry shown in Fig. 2.2, the quadrotor position vector ($\mathbf{\Xi}$), composed by the translational position ($\mathbf{\Gamma_W}$ $[m]$) and rotational position ($\mathbf{\Theta_W}$ $[rad]$) regarding the earth-frame, is defined as

$$\mathbf{\Xi} = \begin{bmatrix} \mathbf{\Gamma_W} \\ \mathbf{\Theta_W} \end{bmatrix} = \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ \psi \\ \theta \\ \phi \end{bmatrix}. \tag{2.15}$$

On the other hand, the quadrotor velocity vector (ν) is composed by the translational ($\mathbf{V_B}$ $[m{\cdot}s^{-1}]$) and rotational ($\mathbf{\Omega_B}$ $[rad{\cdot}s^{-1}]$) velocities with respect to the body-frame

as

$$\nu = \begin{bmatrix} \mathbf{V_B} \\ \mathbf{\Omega_B} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix}. \tag{2.16}$$

Therefore exists a generalized matrix

$$\zeta_{\mathbf{\Theta}} = \begin{bmatrix} \mathbf{R_b^w} & \mathbf{0_{3\times 3}} \\ \mathbf{0_{3\times 3}} & \mathbf{T_b^w} \end{bmatrix}, \tag{2.17}$$

that ensures that

$$\dot{\mathbf{\Xi}} = \zeta_{\mathbf{\Theta}} \nu, \tag{2.18}$$

where $\mathbf{0_{3\times 3}}$ is a 3×3-matrix filled with zeros, $\mathbf{R_b^w}$ is the rotation matrix and $\mathbf{T_b^w}$ the transfer matrix from the body to the world-frame, defined as

$$\mathbf{R_b^w} = \begin{bmatrix} c_\theta c_\psi & c_\psi s_\theta s_\phi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta s_\psi & s_\psi s_\theta s_\phi + c_\phi c_\psi & c_\phi s_\psi s_\theta - s_\phi c_\psi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \tag{2.19}$$

$$\mathbf{T_b^w} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix}, \tag{2.20}$$

with $s_\theta = \sin(\theta)$, $c_\theta = \cos(\theta)$, and $t_\theta = \tan(\theta)$.

As the quadrotor is assumed to be a rigid body of 6 DoF, its dynamics consider the mass (m $[kg]$) and the inertia matrix ($\mathbf{J}$ $[kg \cdot m^2]$) of it, and are described as

$$\mathbf{M_B}\dot{\nu} + \mathbf{S}_\nu = \mathbf{\Lambda}, \tag{2.21}$$

where

$$\begin{aligned}
\mathbf{M_B} &= \begin{bmatrix} m\mathbf{I_{3\times 3}} & \mathbf{0_{3\times 3}} \\ \mathbf{0_{3\times 3}} & \mathbf{J} \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{zz} & 0 & 0 \\ 0 & 0 & 0 & 0 & J_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & J_{xx} \end{bmatrix}, \\
\dot{\nu} &= \begin{bmatrix} \dot{\mathbf{V}}_\mathbf{B} \\ \dot{\mathbf{\Omega}}_\mathbf{B} \end{bmatrix}, \\
\mathbf{S}_\nu &= \begin{bmatrix} \mathbf{\Omega_B} \times m\mathbf{V_B} \\ \mathbf{\Omega_B} \times \mathbf{J\Omega_B} \end{bmatrix} = \begin{bmatrix} m(\ddot{x} + \dot{z}\dot{\theta} - \dot{y}\dot{\phi}) \\ m(\ddot{y} + \dot{x}\dot{\phi} - \dot{z}\dot{\psi}) \\ m(\ddot{z} + \dot{y}\dot{\psi} - \dot{x}\dot{\theta}) \\ J_{zz}\dot{\psi} + \dot{\theta}\dot{\phi}(J_{xx} - J_{yy}) \\ J_{yy}\dot{\theta} + \dot{\psi}\dot{\phi}(J_{zz} - J_{xx}) \\ J_{xx}\dot{\phi} + \dot{\psi}\dot{\theta}(J_{yy} - J_{zz}) \end{bmatrix}, \\
\mathbf{\Lambda} &= \begin{bmatrix} \mathbf{F_B} \\ \tau_\mathbf{B} \end{bmatrix} = \begin{bmatrix} F_x & F_y & F_z & \tau_z & \tau_y & \tau_x \end{bmatrix}^T,
\end{aligned} \tag{2.22}$$

being $\mathbf{J}$ diagonal due to the assumption of a perfect symmetric quadrotor body, $\mathbf{I}$ the identity matrix, $\dot{\mathbf{V}}_\mathbf{B}$ is the quadrotor translational acceleration in the body-frame, $\dot{\mathbf{\Omega}}_\mathbf{B}$ is the quadrotor angular acceleration in the body-frame, $\mathbf{F_B}$ is the quadrotor force vector and $\tau_\mathbf{B}$ is the quadrotor torques vector.

The forces and torques vector $\mathbf{\Lambda}$ results from the effect of the gravitational force (represented in $\mathbf{G_\Lambda}$), the quadrotor inputs created by the motor forces (represented in $\mathbf{U_\Lambda}$), and the gyroscopic effects produced when the motors propellers are rotating (represented in $\mathbf{P_\Lambda}$). However in this project, for simplicity in the dynamic model, it is not taken into account the $\mathbf{P_\Lambda}$ contribution to the $\mathbf{\Lambda}$ vector and thus $\mathbf{P_\Lambda} \approx \mathbf{0_{6\times 1}}$.

The gravitational force affects just the $\mathbf{F_B}$ component in $\mathbf{\Lambda}$, proportionally to its magnitude $|\vec{g}| = g = 9.807\ m/s^2$. $\mathbf{G_\Lambda}$ is the contribution of the gravitational force to the vector $\mathbf{\Lambda}$ and is expressed as

$$\mathbf{G_\Lambda} = \begin{bmatrix} \hat{\mathbf{F}}_\mathbf{Gb} \\ \mathbf{0_{3\times 1}} \end{bmatrix} = \begin{bmatrix} \mathbf{R_b^w F_{G\xi}} \\ \mathbf{0_{3\times 1}} \end{bmatrix} = \begin{bmatrix} \mathbf{R_w^b} \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \\ \mathbf{0_{3\times 1}} \end{bmatrix} = \begin{bmatrix} mg\sin(\theta) \\ mg\cos(\theta)\sin(\phi) \\ -mg\cos(\theta)\sin(\phi) \\ \mathbf{0_{3\times 1}}, \end{bmatrix} \tag{2.23}$$

were $\mathbf{F_{G\xi}} = \mathbf{R_b^w}\hat{\mathbf{F}}_\mathbf{Gb}$ is the translational force due to gravity in the earth-frame, and $\mathbf{R_w^b} = (\mathbf{R_b^w})^{-1}$ is the rotation matrix from the world to the body-frame.

From Section 2.1, it follows that the quadrotor inputs are proportional to the motor forces F_{M_i}, and do not affect the x and y components of $\mathbf{F_B}$. These inputs, depend on the quadrotor configuration ('+' or 'X') and are set as

$$\mathbf{u_\Lambda} = \begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix}, \tag{2.24}$$

where T_u, τ_ψ, τ_θ and τ_ϕ depend on the configuration of the quadrotor as exposed in (2.7) and (2.12).

Thus, (2.21) can be redefined as

$$\mathbf{M_B}\dot{\nu} + \mathbf{S}_\nu = \mathbf{G_\Lambda} + \mathbf{u_\Lambda}. \tag{2.25}$$

The quadrotor acceleration vector $\dot{\nu}$ in the body-frame is then isolated from (2.25), getting

$$\dot{\nu} = \mathbf{M_B}^{-1}(-\mathbf{S}_\nu + \mathbf{G_\Lambda} + \mathbf{u_\Lambda}), \tag{2.26}$$

or expressed as a system of equations

$$\begin{aligned}
\ddot{x} &= \dot{y}\dot{\phi} - \dot{z}\dot{\theta} + g\sin(\theta), \\
\ddot{y} &= \dot{z}\dot{\psi} - \dot{x}\dot{\phi} + g\cos(\theta)\sin(\phi), \\
\ddot{z} &= \dot{x}\dot{\theta} - \dot{y}\dot{\psi} - g\sin(\theta) + \frac{u_1}{m}, \\
\ddot{\psi} &= \dot{\phi}\dot{\theta}\frac{J_{xx} - J_{yy}}{J_{zz}} + \frac{u_2}{J_{zz}}, \\
\ddot{\theta} &= \dot{\phi}\dot{\psi}\frac{J_{zz} - J_{xx}}{J_{yy}} + \frac{u_3}{J_{yy}}, \\
\ddot{\phi} &= \dot{\theta}\dot{\psi}\frac{J_{yy} - J_{zz}}{J_{xx}} + \frac{u_4}{J_{xx}},
\end{aligned} \tag{2.27}$$

with the inputs set as

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -K_m & K_m & -K_m & K_m \\ 0 & -L & 0 & L \\ -L & 0 & L & 0 \end{bmatrix} \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ F_{M_3} \\ F_{M_4} \end{bmatrix}, \tag{2.28}$$

for quadrotors in '+' configuration, and

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -K_m & K_m & -K_m & K_m \\ -L_X & -L_X & L_X & L_X \\ -L_X & L_X & L_X & -L_X \end{bmatrix} \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ F_{M_3} \\ F_{M_4} \end{bmatrix} \tag{2.29}$$

for quadrotors in 'X' configuration.

Defining the state vector as

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T, \tag{2.30}$$

the quadrotor non-linear dynamics model $\mathbf{f}(\mathbf{x}, \mathbf{u})$ got with a Newton-Euler approach is

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{x} \\ \dot{y}\dot{\phi} - \dot{z}\dot{\theta} + g\sin(\theta) \\ \dot{y} \\ \dot{z}\dot{\psi} - \dot{x}\dot{\phi} + g\cos(\theta)\sin(\phi) \\ \dot{z} \\ \dot{x}\dot{\theta} - \dot{y}\dot{\psi} - g\sin(\theta) + \dfrac{u_1}{m} \\ \dot{\psi} \\ \dot{\phi}\dot{\theta}\dfrac{J_{xx} - J_{yy}}{J_{zz}} + \dfrac{u_2}{J_{zz}} \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}\dfrac{J_{zz} - J_{xx}}{J_{yy}} + \dfrac{u_3}{J_{yy}} \\ \dot{\phi} \\ \dot{\theta}\dot{\psi}\dfrac{J_{yy} - J_{zz}}{J_{xx}} + \dfrac{u_4}{J_{xx}} \end{bmatrix}. \tag{2.31}$$

2.2.2 Euler-Lagrange Approach

The general coordinates representing the position and attitude of the quadrotor are defined as

$$\mathbf{\Xi} = \begin{bmatrix} \xi_\mathbf{W} & \mathbf{\Theta_B} \end{bmatrix}^T, \tag{2.32}$$

where $\xi_\mathbf{W} = \begin{bmatrix} X_W & Y_W & Z_W \end{bmatrix}^T$ is the vector representing the position of the CoG of the quadrotor relative to the earth-frame shown in Fig. 2.2 and $\mathbf{\Theta_B} = \begin{bmatrix} \psi & \theta & \phi \end{bmatrix}^T$ represent the quadrotor attitude.

The Lagrangian of the quadrotor is defined by

$$L(\boldsymbol{\Xi}, \dot{\boldsymbol{\Xi}}) = K_{trans} + K_{rot} - E_{pot}, \tag{2.33}$$

where K_{trans} is the quadrotor translational kinetic energy, K_{rot} is the quadrotor rotational kinetic energy, E_{pot} is the quadrotor potential energy, and z is the quadrotor altitude. Hence, the Lagrangian in (2.33) is rewritten as

$$L(\boldsymbol{\Xi}, \dot{\boldsymbol{\Xi}}) = \frac{m}{2}\dot{\xi}_{\mathbf{W}}^{T}\dot{\xi}_{\mathbf{W}} + \frac{1}{2}\dot{\boldsymbol{\Theta}}_{\mathbf{B}}^{T} J \dot{\boldsymbol{\Theta}}_{\mathbf{B}} - mgz. \tag{2.34}$$

The dynamic model of the quadrotor is derived from the Euler-Lagrange equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\boldsymbol{\Xi}}} - \frac{\partial L}{\partial \boldsymbol{\Xi}} = \begin{bmatrix} F_{\xi} \\ \tau \end{bmatrix}, \tag{2.35}$$

where $F_{\xi} = R_b^w \hat{F}_b$ is the translational force applied to the quadrotor by its four motors, and $\tau = \begin{bmatrix} \tau_{\psi} & \tau_{\theta} & \tau_{\phi} \end{bmatrix}^T$.

In the quadrotor body-frame, the translational force $\hat{F}_b$ is only applied in the z-axis, and thus it is represented by

$$\hat{F}_b = \begin{bmatrix} 0 \\ 0 \\ T_u \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_{M_i} \end{bmatrix}. \tag{2.36}$$

The system of equations that represent the dynamics of the quadrotor got using the Euler-Lagrange approach (2.35) is

$$\begin{aligned}
\ddot{X}_W &= \frac{u_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)), \\
\ddot{Y}_W &= \frac{u_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)), \\
\ddot{Z}_W &= \frac{u_1}{m}(\cos(\phi)\cos(\theta)) - g, \\
\ddot{\psi} &= \dot{\phi}\dot{\theta}\frac{J_{xx} - J_{yy}}{J_{zz}} + \frac{u_2}{J_{zz}}, \\
\ddot{\theta} &= \dot{\phi}\dot{\psi}\frac{J_{zz} - J_{xx}}{J_{yy}} + \frac{u_3}{J_{yy}}, \\
\ddot{\phi} &= \dot{\theta}\dot{\psi}\frac{J_{yy} - J_{zz}}{J_{xx}} + \frac{u_4}{J_{xx}},
\end{aligned} \tag{2.37}$$

with the inputs set as

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -K_m & K_m & -K_m & K_m \\ 0 & -L & 0 & L \\ -L & 0 & L & 0 \end{bmatrix} \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ F_{M_3} \\ F_{M_4} \end{bmatrix}, \tag{2.38}$$

for quadrotors in '+' configuration, and

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} T_u \\ \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -K_m & K_m & -K_m & K_m \\ -L_X & -L_X & L_X & L_X \\ -L_X & L_X & L_X & -L_X \end{bmatrix} \begin{bmatrix} F_{M_1} \\ F_{M_2} \\ F_{M_3} \\ F_{M_4} \end{bmatrix}, \tag{2.39}$$

for quadrotors in 'X' configuration [63, 64].

Defining the state vector as

$$\mathbf{x} = \begin{bmatrix} X_W & \dot{X_W} & Y & \dot{Y_W} & Z_W & \dot{Z_W} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T, \tag{2.40}$$

the quadrotor non-linear dynamics model got with an Euler-Lagrange approach, can be represented as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{X}_W \\ \frac{u_1}{m}(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)) \\ \dot{Y}_W \\ \frac{u_1}{m}(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)) \\ \dot{Z}_W \\ \frac{u_1}{m}(\cos(\phi)\cos(\theta)) - g \\ \dot{\psi} \\ \dot{\phi}\dot{\theta}\frac{J_{xx} - J_{yy}}{J_{zz}} + \frac{u_2}{J_{zz}} \\ \dot{\theta} \\ \dot{\phi}\dot{\psi}\frac{J_{zz} - J_{xx}}{J_{yy}} + \frac{u_3}{J_{yy}} \\ \dot{\phi} \\ \dot{\theta}\dot{\psi}\frac{J_{yy} - J_{zz}}{J_{xx}} + \frac{u_4}{J_{xx}} \end{bmatrix}. \tag{2.41}$$

2.3 Linearized Model

2.3.1 Jacobian Linearization

The linearization of a non-linear system is done about an equilibrium point $\overline{\mathbf{x}}$ achieved with a specific input called equilibrium input $\overline{\mathbf{u}}$ where $\mathbf{f}(\overline{\mathbf{x}}, \overline{\mathbf{u}}) \approx \mathbf{0}$. In the quadrotor, such an equilibrium point can be the hover state where $\mathbf{\Omega_B} \to \mathbf{0_{3\times 1}}$, $\dot{\mathbf{\Omega}}_\mathbf{B} \to \mathbf{0_{3\times 1}}$, and $\mathbf{V_B} \to \mathbf{0_{3\times 1}}$. This is

$$\begin{aligned}\overline{\mathbf{x}} &= \begin{bmatrix}\overline{x} & \dot{\overline{x}} & \overline{y} & \dot{\overline{y}} & \overline{z} & \dot{\overline{z}} & \overline{\psi} & \dot{\overline{\psi}} & \overline{\theta} & \dot{\overline{\theta}} & \overline{\phi} & \dot{\overline{\phi}}\end{bmatrix}^T \\ &= \begin{bmatrix}\overline{x} & 0 & \overline{y} & 0 & \overline{z} & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^T,\end{aligned} \tag{2.42}$$

where $\overline{x}$, $\overline{y}$ and $\overline{z}$ define a constant desired position in the body and earth-frame. When the quadrotor state is near the equilibrium point, the body and earth frames are assumed to coincide and thus $\dot{\xi}_\mathbf{W} \to \mathbf{V_B}$, $X_W \to x$, $Y_W \to y$, and $Z_W \to z$ [65].

The equilibrium point shown in 2.42, is obtained using a constant input value $\overline{\mathbf{u}}$ where only the thrust that overcomes gravity is applied, as

$$\overline{\mathbf{u}} = \begin{bmatrix}T_u & \tau_\psi & \tau_\theta & \tau_\phi\end{bmatrix}^T = \begin{bmatrix}mg & 0 & 0 & 0\end{bmatrix}^T. \tag{2.43}$$

The Jacobian linearization is based on the fact that the quadrotor is not exactly at the equilibrium point, but it is close to it with a small deviation $\delta_\mathbf{x} = \mathbf{x} - \overline{\mathbf{x}}$, due to an input deviation $\delta_\mathbf{u} = \mathbf{u} - \overline{\mathbf{u}}$ [66]; Then, the non-linear system is represented by

$$\dot{\delta_\mathbf{x}} = \mathbf{f}(\overline{\mathbf{x}} + \delta_\mathbf{x}, \overline{\mathbf{u}} + \delta_\mathbf{u}). \tag{2.44}$$

Using the first order Taylor polynomial from (2.44) and considering that $\mathbf{f}(\overline{\mathbf{x}}, \overline{\mathbf{u}}) \approx \mathbf{0}$, the following expression is obtained

$$\dot{\delta_\mathbf{x}} \approx \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\Big|_{\substack{\mathbf{x}=\overline{\mathbf{x}} \\ \mathbf{u}=\overline{\mathbf{u}}}} \delta_\mathbf{x} + \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{x}=\overline{\mathbf{x}} \\ \mathbf{u}=\overline{\mathbf{u}}}} \delta_\mathbf{u}. \tag{2.45}$$

The equation (2.45) describes a linear time-invariant representation of the non-linear dynamics of the quadrotor near an equilibrium point $\overline{\mathbf{x}}$ and with an input that tends to be $\overline{\mathbf{u}}$, which is established as a state space model

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t), \tag{2.46}$$

where

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T,$$

$$\mathbf{u} = \begin{bmatrix} T_u & \tau_\psi & \tau_\theta & \tau_\phi \end{bmatrix}^T,$$

$$A = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\Big|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.47}$$

$$B = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\Big|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{zz}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{yy}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{xx}} \end{bmatrix}^T.$$

2.3.2 Thrust Compensation

The forces F_{M_i} applied by the motors in the quadrotor are parallel to the body-frame z-axis and not to the earth-frame Z_W-axis. A compensated thrust input u_1^* must be set so that its projection u_1 (third component of F_ξ) on the Z_W-axis is such that it allows to keep the inputs of the system close to the equilibrium input, even though there are deviations in θ or ϕ.

Given the linearization consideration where the z and Z_W axes coincide, if a linear controller is designed to control the quadrotor, it takes the projection u_1 of $u_1^* = T_u$ on the Z_W-axis as its thrust control signal. From linear algebra, and following the geometry seen in Fig. 2.2, the projection u_1 is defined as

$$u_1 = u_1^* \cos(\theta)\cos(\phi), \tag{2.48}$$

for any $|\theta|$ and $|\phi|$ magnitudes greater than zero. Thus, after each iteration of the control algorithm the real thrust input u_1^* is calculated as

$$u_1^* = \frac{u_1}{\cos(\theta)\cos(\phi)}, \tag{2.49}$$

keeping the desired vertical thrust, despite any deviation about the x or y axes.

For simplicity, the rest of the document assumes that $u_1 = u_1^*$, however during the real implementation the thrust compensation is taken into account.

2.4 Conclusions

This chapter describes the dynamic model of the quadrotor system, based on its reference frame and geometry configuration. The '+' and 'X' quadrotor configurations are detailed, including the setting of the quadrotor inputs and its dependence on the quadrotor motors forces. The dynamic model is obtained using the Newton-Euler approach, for a body-frame base model, and an Euler-Lagrange approach, for a hybrid earth-frame-position/body-frame-attitude model. In order to design linear controllers for the flight dynamics of the quadrotor, a Jacobian linearization about an equilibrium point is implemented. This linearized model is valid for low translational and rotation speed flights with small angular deviations, where both of the non-linear models, got from different approaches, tend to be equivalent. This equilibrium point is achieved by an equilibrium input, which must be added to any control action set by the controller during its implementation. Finally, it is considered the decompensation of the force set by the controller to be exerted by the motors on the earth-frame Z_W-axis, so that it can be compensated in the real quadrotor implementation.

Chapter 3

Smartphone-based Quadrotor Prototype

In this chapter, the quadrotor prototype is presented. All the components that are used to build the prototype are described with the purpose of detailing the function that each component fulfils within the quadrotor. Furthermore, the specific parameters of the built quadrotor are shown, and the procedure carried out to find them experimentally is explained. This is of great importance for the design of the controllers that allow the flight of the built system.

In Fig. 3.1, a small overview of the hardware related to the electrical signals within the quadrotor is presented. This overview shows the interrelation between the main components detailed below.

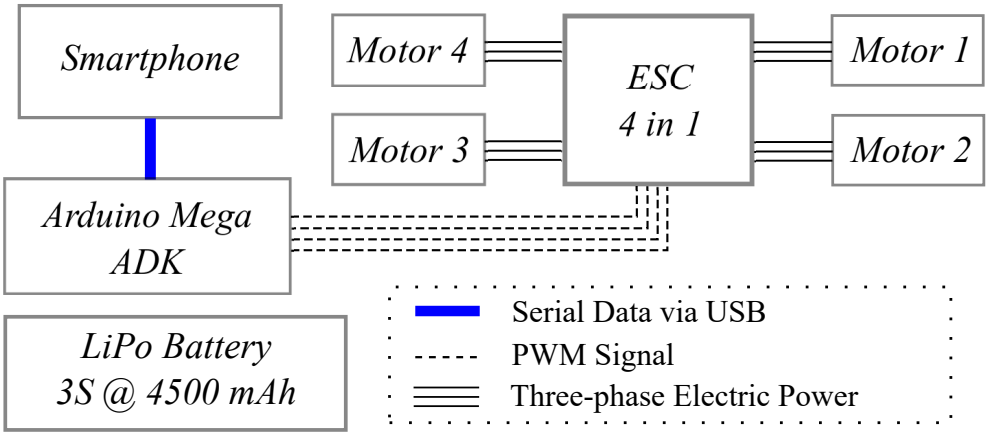


Figure 3.1: Quadrotor prototype hardware overview

3.1 Quadrotor Components

The hardware used to build the quadrotor prototype is detailed in this section. The main goal of this research project is to control a quadrotor using just a smartphone on-board. This smartphone executes the state estimation algorithms and defines the controls signals that will command the actuators. The control signals are processed in the smartphone and then sent, through a gateway, to the Electronic Speed Controllers (ESCs) that will set the angular speed of the motors depending on the control signal they receive.

3.1.1 Frame

The quadrotor's frame is responsible for carrying all the other components needed in the system. Additionally, it must be able to support the weight of any possible payload that the quadrotor will carry. The multirotors frames are usually built using fiberglass or carbon fiber as main material, being the carbon fiber frames stiffer and lighter than equally built fiberglass frames.

For this project, the requirement to build a quadrotor between 0.20 and 0.30 cm in radius was defined. This requirement is based on the fact that the quadrotor must carry a smartphone and possibly some mission-related payloads, which weight can be greater than 100 g. Taking that requirement and the literature review into account, in addition to the weight and stiffness advantages of the carbon fiber, a LJI 500-X4 carbon fiber frame was selected.

Figure 3.2: LJI 500-X4 carbon fiber frame[1]

This frame (Fig. 3.2) has a weight of 431 g and a radius L of 0.244 m measured between the frame center and each of the four rotor axis holes. Its landing gear

[1]LJI 500-X4 frame image taken from https://goo.gl/hHfHQR

has a height of 0.170 m, allowing the payload to be placed in the lower part of the quadrotor.

3.1.2 Smartphone

In this project, the smartphone takes the place of the quadrotor's flight controller. The basic instrumentation needed for a flight controller includes a triaxial accelerometer, a triaxial gyroscope, a triaxial magnetometer, a barometer and a GNSS receiver that can process signals from the GPS and GLONASS satellites constellations. Additionally, the processor of a flight controller must be powerful enough to execute the control and estimation algorithms within the sample time of the control system.

The LG Nexus 5X, shown in Fig. 3.3, is a smartphone developed by Google and assembled by LG, released in 2015 with the Android 6.0.1 operating system. This phone has a Hexa-core Qualcomm MSM8992 Snapdragon 808 CPU that includes four Cortex-A53 and two Cortex-A57 cores with 1.4 GHz and 1.8 GHz clock rate respectively and an Adreno 418 GPU. Its features also include 2 GB of RAM memory, 32 GB of Flash memory, total mass of 136 g and a 12.3 MP camera.

Figure 3.3: LG Nexus 5X, smartphone used as flight controller[2]

This smartphone features all the needed instrumentation for a flight controller, specified previously. The maximum sample rates of the instrumentation contained in the LG Nexus 5X are described in Table 3.1.

[2]LG Nexus 5X image taken from https://goo.gl/RxyDJd

Table 3.1: Sample rates of the sensors in the smartphone

Sensor	Sample rate $[Hz]$
Triaxis accelerometer	400
Triaxis gyroscope	200
Triaxis magnetometer	50
Barometer	10
GNSS	1

The LG Nexus 5X smartphone was selected in this project mainly due to its computational and instrumentation capabilities in addition to its communication interfaces (Bluetooth 4.0, Wireless LAN, USB Type-C and GSM). Its powerful CPU handling a GPU enables the possibility of developments using the camera for visual odometry, and controllers whose execution represents a high computational load, which exceeds the limits of a standard microcontroller such as the ATmega 2560.

As the quadrotor's flight controller, the smartphone receives remote orders from the quadrotor's pilot using wireless communication, while executing the sensorial, estimation and control algorithms. After the control signals are calculated, they are sent to the actuators as a serial data frame using the USB interface between the smartphone and a gateway that converts the serial data into PWM signals.

3.1.3 Motors and Electronic Speed Controllers

The actuation system of the quadrotor is composed by R/C-type brushless motors, propellers and ESCs. The LJI 500-X4 frame manufacturer recommends using motors with a RPM constant of 810 KV, which means that these motors can rotate at a speed of 810 RPM for each Volt applied to it. This kind of motors are used from medium to big sized multirotors due to its thrust efficiency and thrust capacity.

Following this recommendation, the EMAX MT2216II motors were selected. These motors, shown in Fig. 3.4a, are labelled by its manufacturer as 810 KV motors with a maximum current consumption of 9.8 A and a maximum thrust of 6.6 N, when using a 10-inch propeller.

(a) EMAX MT2216II motor

(b) EMAX 4in1 ESC

Figure 3.4: Motors and ESC used in the Quadrotor[3]

The quadrotor brushless motors are powered using a three-phase electric signal generated by the ESC. Each ESC sets one motor rotational velocity depending on a PWM signal input, and must be able to handle the maximum current consumption of the motor. Mainly due to the ease of handling and positioning within the frame, the EMAX 4in1 ESC was selected for this project (Fig. 3.4b). This 4in1 ESC contains four ESCs with a maximum current supply of 30 A each. It also has a DC-DC converter that outputs an isolated 5 V DC supply for any additional electronics. The power supply of the ESC is fully delivered by the quadrotor's battery.

3.1.4 Smartphone-to-ESC Gateway

As the smartphone can not generate any PWM signal, and each ESC needs a PWM signal input to set the rotational velocity of a motor, a gateway between the smartphone and the ESCs must be used. In order to avoid interference in the electromagnetic spectrum and delays that could lead the control system to instability while using wireless communications, the smartphone wired serial bus (USB interface) is defined as the only channel of communication through which the control signals are sent to the gateway.

The Arduino Mega ADK, is shown in Fig. 3.5, is a development board based on the Atmel 8-bit AVR RISC-based ATmega2560 microcontroller. This board supports the Android Open Accesory (AOA) protocol, which allows external hardware to exchange data with Android devices.

[3]Taken from https://goo.gl/6qD6Qg and https://goo.gl/fDHiUp

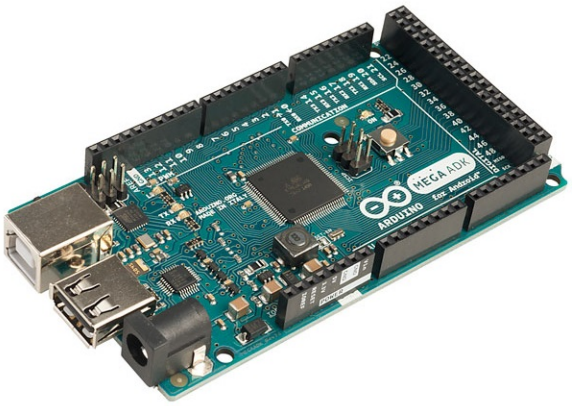


Figure 3.5: Arduino Mega ADK[4]

Although there are other boards that support the AOA protocol, as the IOIO board, the Arduino Mega ADK offers the possibility of executing tasks in the ATmega2560 microcontroller without any intervention of the Android device. This feature is very useful in case of future developments that include additional hardware to the smartphone-based quadrotor.

The Arduino Mega ADK is selected as the smartphone-to-ESC gateway. The workflow of this board is set as follows: the microcontroller receives a serial data frame from the smartphone through the USB port, each of the four PWM signal widths is extracted from the data frame, the PWM signals are sent to the ESCs, finally the cycle starts again. This workflow is executed each 2 ms and if the gateway does not receive any updated data frame from the smartphone, it keeps the last PWM signals set.

3.1.5 Battery

The quadrotor's battery must supply power to all the active components in the quadrotor. The maximum power consumption of these components, considering the nominal voltage value of a 3-cells lithium-ion polymer (LiPo) battery (11.1 V), is shown in Table 3.2.

The maximum current consumption in the quadrotor reaches 42.08 A, so the battery must be able to deliver current amplitudes greater than that value. A Floureon 3-cells LiPo battery with a nominal capacity of 4500 mAh and a nominal voltage of 11.1 V, was selected for this prototype and is shown in Fig. 3.6. This battery can deliver up to 30 times its nominal current, this is 135 A, continuously.

[4]Arduino Mega ADK image taken from https://goo.gl/ejeQeX

Table 3.2: Maximum power consumption of the quadrotor components

Component	Voltage $[V]$	Max. Current $[A]$	Max. Power $[W]$
Smartphone	5	0.75	3.75
Arduino Mega ADK	11.1	0.75	8.33
4 x ESCs	11.1	1.38	15.32
4 x Motors	11.1	39.2	435.12

Figure 3.6: LiPo battery that powers the quadrotor[5]

3.1.6 3D-printed Parts

The smartphone, the Arduino Mega ADK and the 4in1 ESC need to be easily placed and protected when installed on the frame. For that reason, multiple support components and one case were designed and 3D-printed using polylactic acid (PLA) filament. These 3D-printed objects are detailed below.

Smartphone Support

In order to place the smartphone near the *CoG* in the quadrotor and enable the possibility of capturing nadir photos or videos, the smartphone support was designed in such a way that it can be placed under the quadrotor's *CoG* but above the landing gear of the frame. Additionally, the support includes a free area that allows to use the complete field of view of the camera without being obstructed by it. The smartphone support is shown in Fig. 3.7.

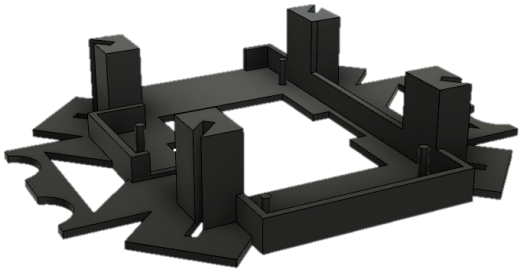
Arduino Mega ADK and ESC supports

The Arduino Mega ADK and the Emax 4in1 ESC are placed on the frame, above the quadrotor's *CoG*. Given the limited space available to locate these components

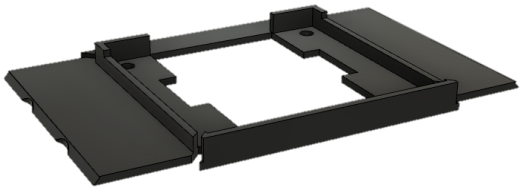
[5]Floureon 3S LiPo battery image taken from https://goo.gl/anC9M2

Figure 3.7: Smartphone support

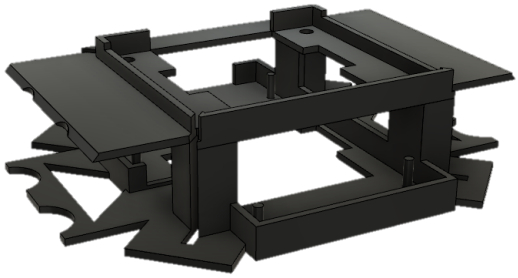
on the top of the frame, their supports are designed to be placed one on top of each other, as shown in Fig. 3.8.



(a) Arduino Mega ADK support



(b) ESCs support



(c) Arrangement of the Gateway and ESCs Support as a Whole

Figure 3.8: Arduino Mega ADK and EMAX 4in1 ESCs designed supports

Dome

A dome that covers and encloses the Arduino Mega ADK and the ESCs, is designed. This dome, shown in Fig. 3.9, totally encloses the Mega ADK and ESCs supports restricting their movement and protecting them in case of a shock or hit.

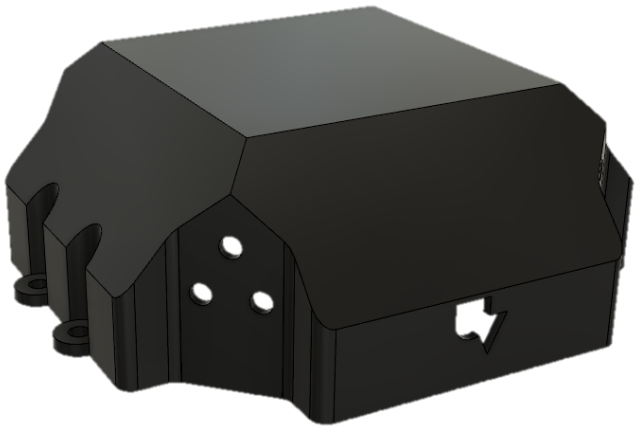


Figure 3.9: 3D designed dome

3.1.7 Assembled Smartphone-based Quadrotor

The smartphone-based quadrotor prototype is assembled using all the components exposed in this section, and is shown in Fig. 3.10.

Figure 3.10: Assembled smartphone-based quadrotor prototype

3.2 Quadrotor Parameters

In this section, the quadrotor parameters are detailed. These parameters define the specific dynamic model for the smartphone-based quadrotor prototype and the conversion of control signals to correctly set the motors thrust.

3.2.1 Mass

The mass of each quadrotor's component is measured using a kitchen scale that has an uncertainty of ± 1 g. The results are shown in Table 3.3.

Table 3.3: Mass values of all the quadrotor components

Component	Mass $[kg]$
Smartphone	0.136
Frame	0.431
Battery	0.351
ESC	0.110
Arduino Mega ADK	0.330
Motors (4)	0.140
Propellers (4)	0.380
Smartphone support	0.105
ADK support	0.590
ESC support	0.350
Dome	0.130
Total	1.568

The total mass of the quadrotor m is then calculated as the sum of the weight of each of the components in the quadrotor, being $m = 1.568$ kg.

3.2.2 Moments of Inertia

As exposed by [67], in a quadrotor, the moment of inertia matrix $\mathbf{J}$ is set as

$$\mathbf{J} = \begin{bmatrix} J_{xx} & -J_{xy} & -J_{xz} \\ -J_{yx} & J_{yy} & -J_{yz} \\ -J_{zx} & -J_{zy} & J_{zz} \end{bmatrix}. \tag{3.1}$$

Taking into account that the symmetry of the quadrotor with respect to the x and y axes is assumed, the $\mathbf{J}$ matrix can be approximated to

$$\mathbf{J} \approx \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}. \tag{3.2}$$

The J_{xx}, J_{yy} and J_{zz} values can be obtained using multiple methods including: using a CAD model of the quadrotor and obtaining the inertia values from a 3D design software [68]. This is done, approximating the shape of the quadrotor components to cylinders, cubes and other basic shapes to simplify the mathematical calculation of inertia [69], and developing the bifilar pendulum experiment on the quadrotor [70]. For this project, it was decided to obtain the quadrotor parameters experimentally, so the bifiliar pendulum experiment was developed.

In the bifilar pendulum experiment, an object is hung from two parallel ropes of length r and separated by a distance $2l$, being allowed to rotate freely around a the axis that is parallel to the ropes. In Fig. 3.11, the geometry of the experiment to get J_{zz}, is shown. For the J_{xx} and J_{yy} inertias experiment, it is necessary to place the quadrotor hanging in such a way that the x and y axes are pointing paraller to the ropes, respectively.

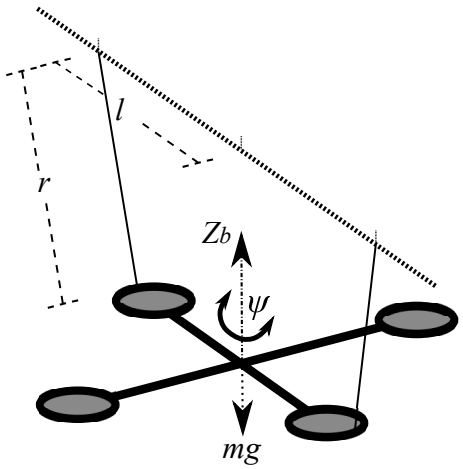


Figure 3.11: Bifilar pendulum experiment geometry for inertia identification

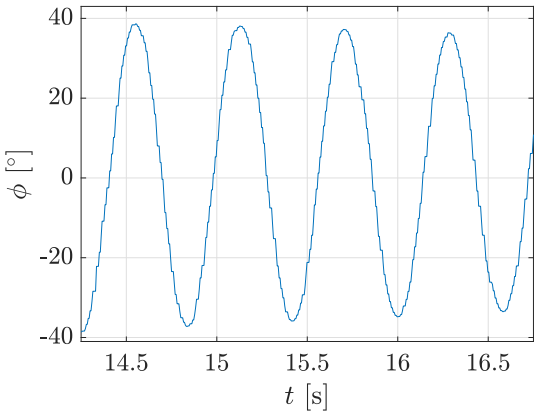
Once the quadrotor is hanging from the ropes, a small torque is applied manually to the quadrotor making it rotate about the vertical axis. The tension in the ropes makes the quadrotor swing, with a period of T_{osc} $[s]$, about the rotation axis.

The moment of inertia J is then calculated using the bifiliar equation

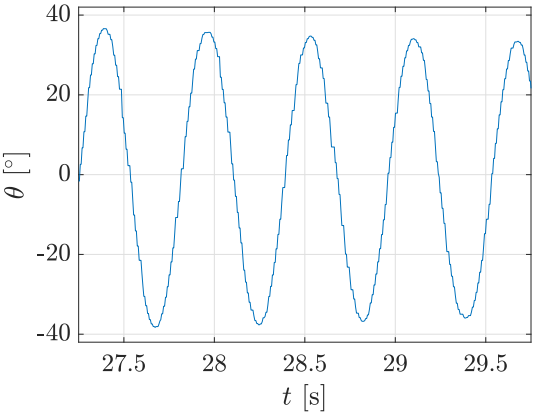
$$J = \frac{mgT_{osc}^2 l^2}{4\pi^2 r} \; [kg \cdot m^2], \tag{3.3}$$

where $m = 1.568\ kg$ is the quadrotor's total mass and $g = 9.807\ m/s^2$ is the gravity acceleration magnitude [71].

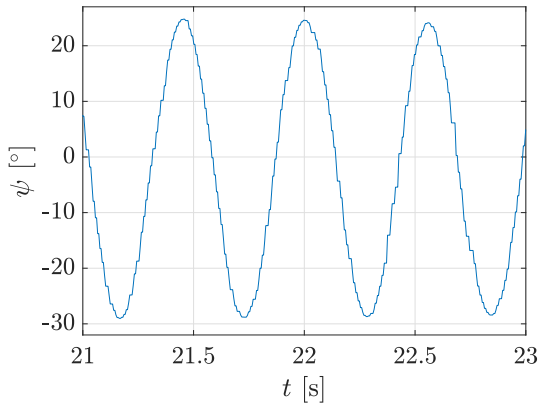
In Fig. 3.12, it is shown the excursion of the rotation angles, ϕ, θ and ψ, about the x, y and z axes respectively, while performing the bifilar pendulum experiment separately.



(a) Rotation about x axis, J_{xx} experiment



(b) Rotation about y axis, J_{yy} experiment



(c) Rotation about z axis, J_{zz} experiment

Figure 3.12: Rotation about x, y and z axes during the bifilar pendulum experiments

The resulting data got after the execution of the experiment around the three components of the quadrotor body frame, are shown in Table 3.4.

Table 3.4: Bifilar pendulum experiment results

Rotation Axis	$r\ [m]$	$l\ [m]$	$T_{osc}\ [s]$	Inertia value $[kg \cdot m^2]$
x	1.18	0.173	1.168	$J_{xx} = 0.0135$
y	1.10	0.173	1.080	$J_{yy} = 0.0124$
z	1.025	0.265	1.122	$J_{zz} = 0.0336$

3.2.3 Motor Thrust

As seen in Section 3.1, the motors rotational velocity ω_i is set by the ESC, which receives a PWM signal input. However, the inputs of the quadrotor (T_u, τ_ψ, τ_θ, and τ_ϕ) depend on the thrust force F_{M_i} applied by each quadrotor motor, as exposed in Section 2.1.

In order to correctly set the desired force F_{M_i} in each motor during a flight, it is necessary to characterize the motor; and thus know how the force applied by the motor behaves with respect to the input PWM signal. This characterization is carried out by means of a thrust test.

In the thrust test, the motor and its corresponding propeller are fixed pointing up over a calibrated scale, as shown in Fig. 3.13. The motor is connected to the ESC, and then the PWM signal is increased with steps of 10, with 0 being the minimum width of PWM signal and 255 the maximum.

With each PWM signal width increment, a scale reading is made. Since the readings m_s are obtained in units of mass (kg), the F_{M_i} must be calculated as

$$F_{M_i} = m_s g\ [N]. \tag{3.4}$$

This test was developed twice with each motor and its results are shown in Fig. 3.14a.

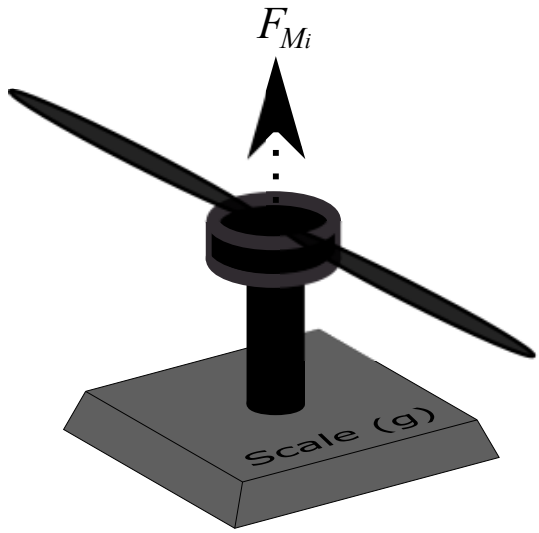
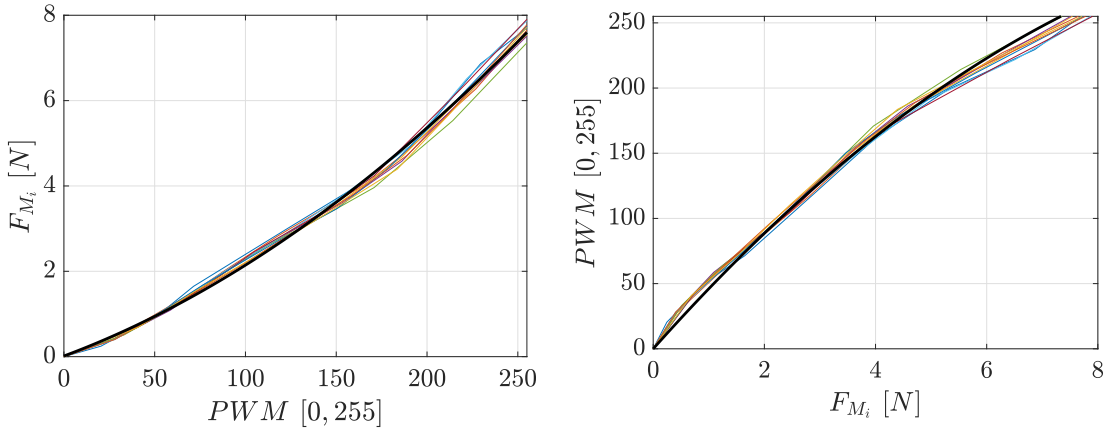


Figure 3.13: Thrust test configuration



(a) Motors characterization

(b) Inverse motors characterization

Figure 3.14: Motors thrust test results

In Fig. 3.14b, the inverse characterization is exposed. In this case, the PWM signal is defined as the dependent variable, and F_{M_i} as independent. This inverse characterization defines the PWM width setting that is sent from the smartphone to the Arduino Mega ADK.

The trend equations resulting from the thrust test are

$$F_{M_i} = (5.441 \times 10^{-5})(PWM)^2 + 0.01586(PWM) + 0.014808, \tag{3.5}$$

$$PWM = -1.983F_{M_i}^2 + 47.84F_{M_i} + 3.835, \tag{3.6}$$

where F_{M_i} is given in N, and PWM is a value between 0 and 255.

3.2.4 Motor Torque

The quadrotor suffers the application of a torque τ_{M_i} around the z axis when a motor M_i rotates. This torque affects the ψ angle indirectly by adding to the τ_ψ torque proportionally to the thrust force F_{M_i}, being $\tau_{M_i} = K_M F_{M_i}$, as shown in (2.3).

The torque τ_{M_i} is generated due to the conservation of momentum, and its unbalance is used to rotate the quadrotor about the z axis in the opposite direction to the rotation of the motor that generates it as exposed in Section 2.1. In order to properly control the mentioned unbalance, it is necessary to find the value of the constant K_m.

The constant K_m can be found through a steady-state torque experiment, as stated by [72]. In this experiment, the z axis in the quadrotor is located parallel to the ground, leaving the rotation about this axis as the only unblocked DoF. Using the geometry seen in Fig. 3.15, it is necessary to measure the force F_s that is generated when the motor M_i is rotating.

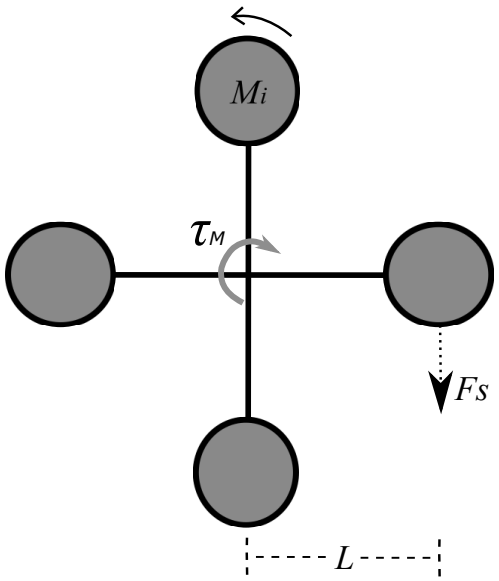


Figure 3.15: Motor torque experiment configuration

Taking into account that when the motor rotates clockwise, the quadrotor will tend to rotate counter-clockwise and vice versa, a scale is located to indirectly obtain the generated force F_s. This is done using the reading of the weight m_s as $F_s = m_s|\vec{g}|$ $[N]$. Using (3.5) and the PWM width steps used in the motors thrust experiment, multiple F_{M_i}-dependent F_s readings are obtained. The torques τ_M are

then calculated by

$$\tau_M = F_s L \ [N \cdot m], \tag{3.7}$$

and their results are shown in Fig. 3.16.

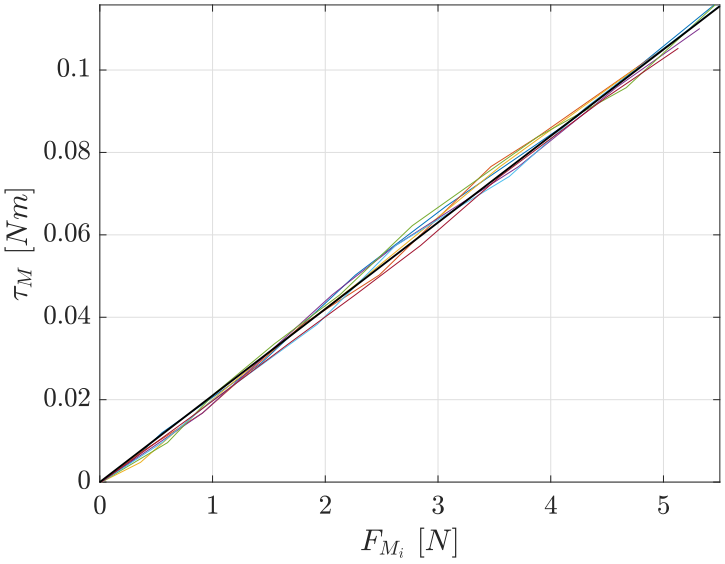


Figure 3.16: Motors torque experiment results

The relation between τ_M and F_{M_i} is linear as expected, and its slope defines the variable K_m as

$$K_m = 0.021 \ [m]. \tag{3.8}$$

3.3 Conclusions

This chapter presented the physical composition and parametrization of the smartphone-based quadrotor prototype. In the first part, all the components of the quadrotor are detailed. A medium-size carbon fiber frame is used to support the complete quadrotor system. The LG Nexus 5X is selected as the smartphone where the control and estimation algorithms will be executed, which sends the control signals to an Arduino Mega ADK in order to generate PWM signals that set the motors rotational velocities through the ESC. These components are supported and protected by 3D-printed objects that are attached to the carbon fiber frame. On the other hand, the system is parameterized in its entirety using experimental methods. Here, the total mass and moments of inertia of the system are defined using a scale and the bifilar pendulum experiment respectively, while the motor's thrust and torque about the z-axis are found applying multiple PWM signals to the ESCs and obtaining the corresponding variable with the help of the scale.

Chapter 4

Control Strategies and State Estimation

This project has the aim to evaluate control strategies designed for a quadrotor, test its performance in simulation, and then implement them in the smartphone-based quadrotor prototype. In this chapter, the design procedure of the control and estimation algorithms for the quadrotor is shown. This algorithms are based on the linearized model of the quadrotor, detailed in Section 2.3. Also, all the simulations were carried out in MATLAB to verify the proper functioning of the designed algorithms.

The state-space representation of the system, and the concept of controllability and observability, are briefly introduced in Section 4.1. Then, Section 4.2 shows the theoretical basis required for the design of the controllers used in this project. These controllers are the Linear Quadratic Integral (LQI) controller and the H_∞ controller.

Section 4.3 presents the considerations made for the design of the two types of controllers according to the flight mode of the quadrotor. In addition, this section shows the simulated response of the quadrotor being controlled by both types of controllers in each flight mode. All the simulations are done using the non-linear dynamics model of the quadrotor. Finally, the state estimation algorithm is detailed in Section 4.4.

4.1 Concept and Generalities

4.1.1 State Space Representation

Recalling from Section 2.3, the quadrotor model has 6 DoF and it can be represented by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t), \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t), \end{aligned} \tag{4.1}$$

where A is the system matrix; B is the input matrix; C is the output matrix; D is the feed-through matrix; $\mathbf{x}$ is the states vector of size n_x; $\mathbf{u}$ is the inputs vector of size n_u; and $\mathbf{y}$ is the outputs vector of size n_y.

For simplicity, the state-space model representation shown in (4.3), can be represented by the quartet of matrices $\mathcal{G}$ as

$$\mathcal{G} = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right], \tag{4.2}$$

For the design of each controller, only the dynamics that are required to control, depending on the flight mode in which the quadrotor is set, are taken into account. Therefore, system $\mathcal{G}$ will vary for each flight mode.

Since the control system is implemented in a smartphone, whose algorithms are executed with a sample time of $t_k = 0.01\ s$, the system $\mathcal{G}$ is discretized. The discrete system $\mathcal{G}_k$ is got using the Zero Order Hold (ZOH) equivalent method, and the sample time $t_k = 0.01\ s$. The discrete state-space model $\mathcal{G}_k$ is represented by

$$\begin{aligned} \mathbf{x}(k+1) &= A_k\mathbf{x}(k) + B_k\mathbf{u}(k), \\ \mathbf{y}(k+1) &= C_k\mathbf{x}(k) + D_k\mathbf{u}(k). \end{aligned} \tag{4.3}$$

Thus, $\mathcal{G}_k$ is represented by the quartet

$$\mathcal{G}_k = \left[\begin{array}{c|c} A_k & B_k \\ \hline C_k & D_k \end{array}\right], \tag{4.4}$$

which is the system for which the controllers are developed.

4.1.2 Controllability and Observability

The concept of controllability is related to the question of whether or not there exists a sequence of $\mathbf{u}$ capable of changing the states $\mathbf{x}$ from an initial value $\mathbf{x_0}$, to

a desired final value $\mathbf{x_f}$, in a finite time. On the other hand, observability refers to the possibility of inferring, in finite time, the initial value of the states $\mathbf{x_0}$ knowing just the system dynamics $\mathcal{G}$ and its outputs $\mathbf{y}$ [73].

Typically, the controllability and observability of a system are determined using the so-called controllability and observability matrices, $\mathcal{C}_\mathcal{G}$ and $\mathcal{O}_\mathcal{G}$ respectively. This matrices depend on the system matrices A, B and C, and are set as

$$\begin{aligned} \mathcal{C}_\mathcal{G} &= \begin{bmatrix} B & AB & \dots & A^{k-1}B \end{bmatrix}, \\ \mathcal{O}_\mathcal{G} &= \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-1} \end{bmatrix}. \end{aligned} \tag{4.5}$$

Thus, the system $\mathcal{G}$ is defined as controllable if $\mathcal{C}_\mathcal{G}$ have n_x linear independent rows (or columns). This is, the rank of $\mathcal{C}_\mathcal{G}$ is equal to n_x. Analogous to the controllability, the observability of $\mathcal{G}$ is checked if the rank of the matrix $\mathcal{O}_\mathcal{G}$ is equal to n_x.

One equivalent way to check the controllability and observability features of $\mathcal{G}$ is using the controllability and observability Grammians $\mathcal{W}_c$ and $\mathcal{W}_o$ defined as the solutions to the Lyapunov equations

$$\begin{aligned} 0 &= A\mathcal{W}_c + \mathcal{W}_c A^T + BB^T, \\ 0 &= A^T\mathcal{W}_o + \mathcal{W}_o A + C^T C, \end{aligned} \tag{4.6}$$

where

$$\begin{aligned} \mathcal{W}_c &= \int_0^\infty e^{At} BB^T e^{A^T t} dt, \\ \mathcal{W}_o &= \int_0^\infty e^{A^T t} C^T C e^{At} dt. \end{aligned} \tag{4.7}$$

In this case, if the matrices $\mathcal{W}_c$ and $\mathcal{W}_o$ are positive definite, the system is both controllable and observable [74].

4.2 Control Strategies

This section exposes the controllers design procedure. Here, the mathematical procedure to design a linear quadratic regulator with integral feedback (linear quadratic integral controller) is described. The design process of a H_∞ controller is also shown, taking into account weighting sensitivities.

4.2.1 Linear Quadratic Integral (LQI) Controller

Optimal Problem Solution

The design of optimal controllers seeks that a dynamic system can be controlled achieving a minimum cost. The cost function is determined by the control designer [75]. With a finite horizon, a linear quadratic regulator (LQR) is set while looking for the minimization of the cost function $\mathcal{V}$ as

$$\mathcal{V} = \int_0^T \mathcal{L}(\mathbf{x}, \mathbf{u}, t)\ dt + \Psi(\mathbf{x}, \mathbf{u}, t), \tag{4.8}$$

where

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{u}, t) &= \mathbf{x}^T \mathcal{Q}\mathbf{x} + \mathbf{u}^T \mathcal{R}\mathbf{u}, \\ \Psi(\mathbf{x}, \mathbf{u}, t) &= \mathbf{x}^T(T)\mathcal{S}\mathbf{x}(T), \end{aligned} \tag{4.9}$$

$\mathcal{R}$ is a positive definite matrix; and $\mathcal{Q}$ and $\mathcal{S}$ are positive semi-definite matrices. These matrices penalize the inputs, states and the terminal cost, respectively. Thus, the optimal controller design problem turns into an optimization problem where it is necessary to find an input $\mathbf{u}^*$ such that minimizes $\mathcal{V}$ for a controllable system subject to

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \approx A\mathbf{x}(t) + B\mathbf{u}(t), \\ \mathbf{x}(0) &= \mathbf{x_0}, \quad t \in [0, T]. \end{aligned} \tag{4.10}$$

This minimization is achieved using the Pontryagin maximum principle [76], where

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \lambda_\mathcal{P}, t) = \mathcal{L}(\mathbf{x}, \mathbf{u}, t) + \lambda_\mathcal{P}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{4.11}$$

is the Hamiltonian function, with $\lambda_\mathcal{P}$ being the vector of co-state variables of size n_x, and

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \left(\frac{\partial \mathcal{H}(\mathbf{x}, \mathbf{u}, \lambda_\mathcal{P}, t)}{\partial \lambda_\mathcal{P}}\right)^T = A\mathbf{x}(t) + B\mathbf{u}(t), \\ -\dot{\lambda}_\mathcal{P}(t) &= \left(\frac{\partial \mathcal{H}(\mathbf{x}, \mathbf{u}, \lambda_\mathcal{P}, t)}{\partial \mathbf{x}}\right)^T = \mathcal{Q}\mathbf{x}(t) + A^T \lambda_\mathcal{P}(t), \\ 0 &= \frac{\partial \mathcal{H}(\mathbf{x}, \mathbf{u}, \lambda_\mathcal{P}, t)}{\partial \mathbf{u}} = \mathcal{R}\mathbf{u}(t) + \lambda_\mathcal{P}^T(t) B. \end{aligned} \tag{4.12}$$

Thereby, from (4.12), considering the Pontryagin maximum principle, the optimal solution of $\mathbf{u}$ that minimizes $\mathcal{H}(\mathbf{x}, \mathbf{u}, \lambda_\mathcal{P}, t)$ and the cost function $\mathcal{V}$, is

$$\mathbf{u}^*(t) = -\mathcal{R}^{-1} B^T \lambda_\mathcal{P}(t). \tag{4.13}$$

Assuming that $\lambda_{\mathcal{P}}(t) = \mathcal{P}(t)\mathbf{x}(t)$, the optimal input $\mathbf{u}^*$ depends directly on a feedback vector, which is the state vector $\mathbf{x}$, such that

$$\begin{aligned} \mathbf{u}^*(t) &= \mathbf{K}_{\mathbf{lqr}}(t)\mathbf{x}(t), \\ \mathbf{K}_{\mathbf{lqr}}(t) &= -\mathcal{R}^{-1}B^T\mathcal{P}(t). \end{aligned} \tag{4.14}$$

Here, matrix $\mathcal{P}(t)$ is a solution of the Riccati Equation [77]

$$-\dot{\mathcal{P}} = \mathcal{P}A + A^T\mathcal{P} + \mathcal{Q} - \mathcal{P}B\mathcal{R}^{-1}B^T\mathcal{P}. \tag{4.15}$$

When $T \to \infty$, an infinite-time regulator is designed. Here, aiming for a steady-state solution, the terminal cost $\Psi(\mathbf{x}, \mathbf{u}, t)$ is eliminated, and the matrix $\mathcal{P}$ is assumed to be constant. Thus, $\mathcal{P}$ is a solution of the Algebraic Riccati Equation (ARE)

$$0 = \mathcal{P}A + A^T\mathcal{P} + \mathcal{Q} - \mathcal{P}B\mathcal{R}^{-1}B^T\mathcal{P}. \tag{4.16}$$

Consequently, in the infinite-time LQR, the optimal input $\mathbf{u}^*$ is achieved by multiplying the state vector $\mathbf{x}$ with the constant feedback gain matrix

$$\mathbf{K}_{\mathbf{lqr}} = -\mathcal{R}^{-1}B^T\mathcal{P}. \tag{4.17}$$

As the feedback gain is not a dynamical system, but a static matrix, the order of the closed-loop dynamics is the same as the one of the system dynamics $\mathcal{G}$. The closed-loop dynamics are written as

$$\dot{\mathbf{x}} = (A + B\mathbf{K}_{\mathbf{lqr}})\mathbf{x}. \tag{4.18}$$

When the desired state value $\mathbf{x}_{\mathbf{des}}$ is different from zero, these reference value is applied to the dynamics through the feedback gain matrix as

$$\dot{\mathbf{x}} = (A + B\mathbf{K}_{\mathbf{lqr}})\mathbf{x} - B\mathbf{K}_{\mathbf{lqr}}\mathbf{x}_{\mathbf{des}}, \tag{4.19}$$

due to the fact that the error $\mathbf{e}_{\mathbf{x}} = \mathbf{x} - \mathbf{x}_{\mathbf{des}}$ is used as the feedback vector of the LQR.

Reference Tracking

For the quadrotor position control, a controller capable of reference tracking is needed. However, the LQR controller is not made for reference tracking, but just for regulation. This is overcame making use of a Linear Quadratic Integral (LQI) controller, which is a LQR with an additional integral feedback. The integral action ensures that a zero steady-state error is achieved.

The approach of using integral feedback with a LQR to provide tracking capabilities

to the control system, is based on the augmentation of the state vector $\mathbf{x}$ with a vector

$$\mathbf{x_i} = \int \mathbf{e}\ dt, \quad \mathbf{x_i}(0) = \mathbf{0}, \quad \mathbf{x_i} \in \mathbb{R}^{n_y}, \tag{4.20}$$

containing the integral of the error signal $\mathbf{e} = r - \mathbf{y}$, where r is the reference signal of size n_y. The augmented state vector $\mathbf{x_{lqi}}$ is defined as

$$\mathbf{x_{lqi}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{x_i} \end{bmatrix}, \tag{4.21}$$

with which the augmented system to be controlled is obtained as

$$\begin{aligned} \dot{\mathbf{x}}_{\mathbf{lqi}} &= \bar{A}\mathbf{x_{lqi}} + \bar{B}\mathbf{u} + B_i r, \\ \mathbf{y} &= \bar{C}\mathbf{x_{lqi}}, \end{aligned} \tag{4.22}$$

where

$$\begin{aligned} \bar{A} &= \begin{bmatrix} A & \mathbf{0}_{n_x \times n_y} \\ -C & \mathbf{0}_{n_y \times n_y} \end{bmatrix}, \\ \bar{B} &= \begin{bmatrix} B \\ \mathbf{0}_{n_y \times n_u} \end{bmatrix}, \\ B_r &= \begin{bmatrix} \mathbf{0}_{n_x \times n_y} \\ \mathcal{I}_{n_y \times n_y} \end{bmatrix}, \\ \bar{C} &= \begin{bmatrix} C & \mathbf{0}_{n_y \times n_y} \end{bmatrix}. \end{aligned} \tag{4.23}$$

Given the augmented system in (4.22), the controller is designed using the Optimal Problem Solution detailed before, obtaining a control law of the form

$$\mathbf{u} = \mathbf{K_{lqi}}\mathbf{x_{lqi}} = \begin{bmatrix} \mathbf{K_{lqr}} & \mathbf{K_i} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x_i} \end{bmatrix}, \tag{4.24}$$

where $\mathbf{K_{lqr}}$ and $\mathbf{K_i}$ are submatrices which correspond to the feedback gain matrices that multiply $\mathbf{x}$ and $\mathbf{x_i}$ respectively, as shown in Fig. 4.1.

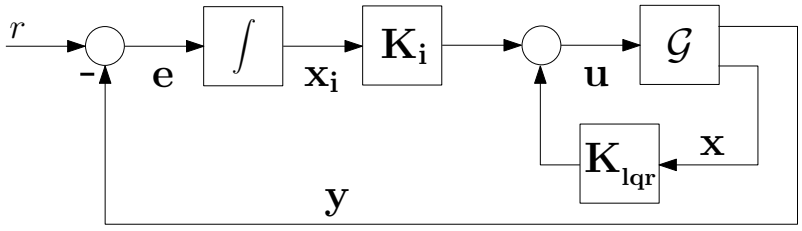


Figure 4.1: Closed-loop system with LQI controller for reference tracking

4.2.2 H_∞ Controller

H_∞ Synthesis for Controller Design

The design of the H_∞ controller is based on a representation of the closed-loop system including the generalized plant P_H and the controller K_H, shown in Fig. 4.2.

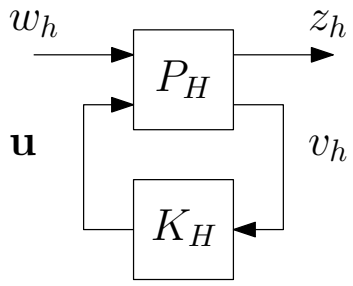


Figure 4.2: Control loop with generalized plant

The generalized plant P_H corresponds to the system

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B_w w_h + B\mathbf{u}, \\ z_h &= C_z\mathbf{x} + D_{zw} w_h + D_{zu}\mathbf{u}, \\ v_h &= -C\mathbf{x} + D_{vw} w_h + D_{vu}\mathbf{u}. \end{aligned} \tag{4.25}$$

where w_h are the external inputs, including command inputs, disturbances and noises; z_h is a fictitious output vector used to express design specifications; and v_h are the measured outputs of the system [74].

The H_∞ design procedure aims to synthesize a dynamic controller K_H, with input $v_h = -\mathbf{y}$ and output $\mathbf{u}$, such that the closed-loop system is stabilized and the fictitious output z_h is minimized.

The system in Fig. 4.2, is rearranged for easy interpretation, in the way of Fig. 4.3.
In Fig. 4.3, $\mathcal{G}$ represents the quadrotor dynamics; K_H the controller; r is the system reference; $\mathbf{e}$ is the error; $\mathbf{u}$ is the control input; $z_h = \begin{bmatrix} Z_s & Z_k \end{bmatrix}$; and W_s, W_k are weighting filters that must satisfy

$$\gamma = \left\| \begin{bmatrix} W_s S \\ W_k K_H S \end{bmatrix} \right\|_\infty < 1, \tag{4.26}$$

where S is the sensitivity function and $K_H S$ is the control sensitivity defined as

$$S = (\mathcal{I} + \mathcal{G}K_H)^{-1}, \quad K_H S = K_H(\mathcal{I} + \mathcal{G}K_H)^{-1}. \tag{4.27}$$

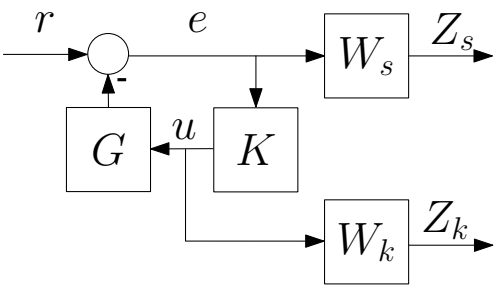


Figure 4.3: Generalized plant with the weighting filters W_s and W_k.

The H_∞ norm is defined only for proper and stable systems, so the weighting filters must satisfy that condition. The weighting filter W_s impose an upper bound on the sensitivity and is designed taking into account that it is desired to have integral action in the closed-loop. This ensures a zero steady-state error. On the other hand, the filter W_k impose an upper bound on the control sensitivity and must have a high-pass behaviour.

The weighting filters for the sensitivity and the control sensitivity are chosen as

$$\begin{aligned} W_s &= \frac{w_s/M_s}{s+w_s} * \mathcal{I}_{n_y \times n_y} \ , \\ W_k &= \frac{c_k}{M_k} \frac{s+w_k}{s+c_k w_k} * \mathcal{I}_{n_u \times n_u} \ , \end{aligned} \tag{4.28}$$

where M_s is small constant that sets an upper bound on the sensitivity at low frequencies; and w_s is a small constant that ensures that W_s does not have a pole at the origin; M_k is a constant that sets the upper bound on the control sensitivity at low frequencies; w_k is a small constant that place the zero of W_k near the origin; and c_k is a large constant that place the pole of W_k at a frequency above the bandwidth.

If the resulting value of γ is greater than one, the design procedure is iterated normalizing the weighting filters with γ.

In this project, the synthesized optimal H_∞ controllers K_H are computed using the Robust Control Toolbox of MATLAB.

H_∞ Controller Order Reduction

The designed H_∞ controller K_H is a dynamic model with n_u outputs, n_y inputs and order $n_H = n_x + n_y + n_u$. To reduce the computational load that the execution of a high-order dynamic system entails, it is necessary to find a reduced order controller that behaves similarly to the full order controller K_H.

In a dynamic system, the states with small singular value energy in $\mathcal{W}_c$ show a weak response to a control input, while states with small singular value energy in $\mathcal{W}_o$ have weak influence on the observed output. In order to check which states of K_H have little influence both in terms of controllability and observability, the singular values of the Hankel matrix

$$H_k = \mathcal{O}_{\mathcal{K}_\mathcal{H}} \mathcal{C}_{\mathcal{K}_\mathcal{H}}, \tag{4.29}$$

are analysed. The matrices $\mathcal{C}_{\mathcal{K}_\mathcal{H}}$ and $\mathcal{O}_{\mathcal{K}_\mathcal{H}}$ represent the controllability and observability matrices of the system K_H, respectively. A Hankel singular value (HSV) with small energy indicates that a state has little influence both in terms of controllability and observability.

If $\hat{K}_H = (\hat{A}_H,\ \hat{B}_H,\ \hat{C}_H,\ \hat{D}_H)$ is the balanced realization of K_H with n_H state variables, and we have n_h significant states, so the last $n_H - n_h$ HSVs are small enough to be neglected without modifying the system dynamics [78].

4.3 Controllers Design and Simulation

This section shows the specific design for the LQI and H_∞ controllers in each of the quadrotor flight modes. Based on what is described on Section 4.2, the dynamic model for each flight mode is set. Then, the controllers are designed and simulated in order to check their proper operation.

4.3.1 Stabilize Mode

In stabilize mode, the dynamics related to the position of the quadrotor are neglected. Thus, the only *DoF* that are automatically controlled are ψ, θ, and ϕ.

Dynamic Model

The dynamic model exposed in 2.3, is reduced to a sixth order dynamic system ($n_x = 6$) with three outputs ($n_y = 3$), where

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T, \\ \mathbf{y} &= \begin{bmatrix} \psi & \theta & \phi \end{bmatrix}^T. \end{aligned} \tag{4.30}$$

The linear state-space representation (4.3) for the dynamics related to the stabilize mode is defined by the matrices

$$
\begin{aligned}
A &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\
B &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \dfrac{1}{J_{zz}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{1}{J_{yy}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dfrac{1}{J_{xx}} \end{bmatrix}^T, \\
C &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \\
D &= \mathbf{0_{3\times 4}}.
\end{aligned}
\tag{4.31}
$$

Both the controllability matrix $\mathcal{C}_\mathcal{G}$ and observability matrix $\mathcal{O}_\mathcal{G}$, have full rank ($rank(\mathcal{C}_\mathcal{G}) = rank(\mathcal{O}_\mathcal{G}) = 6$), and therefore the system is controllable and observable.

LQI Controller

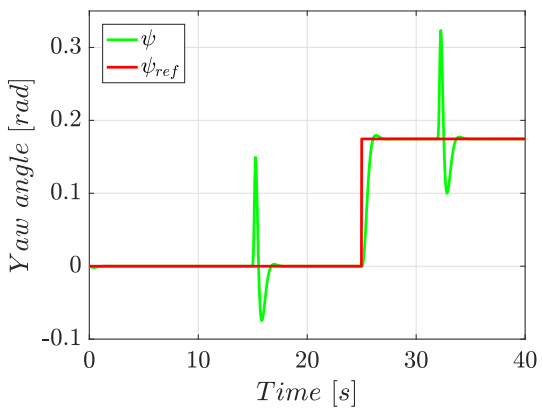
The design of the LQI controller is done using the penalization matrices $\mathcal{Q}$ and $\mathcal{R}$, that modify the cost function (4.8). Both matrices are set as diagonal matrices, so that there is only one parameter in the matrix affecting each state or input. The matrix $\mathcal{Q}$ must be a 9×9 matrix, since the size of $\mathbf{x_{lqi}}$ is $n_x + n_y$. Similarly, $\mathcal{R}$ must have four components in its diagonal, inasmuch as the number of inputs n_u remains four. In [58], the author suggests the matrices

$$
\begin{aligned}
\mathcal{Q} &= \mathcal{I}_{9\times 9} \begin{bmatrix} 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 10 & 40 & 40 \end{bmatrix}^T, \\
\mathcal{R} &= \mathcal{I}_{4\times 4} \begin{bmatrix} 3 & 3 & 3 & 3 \end{bmatrix}^T,
\end{aligned}
\tag{4.32}
$$

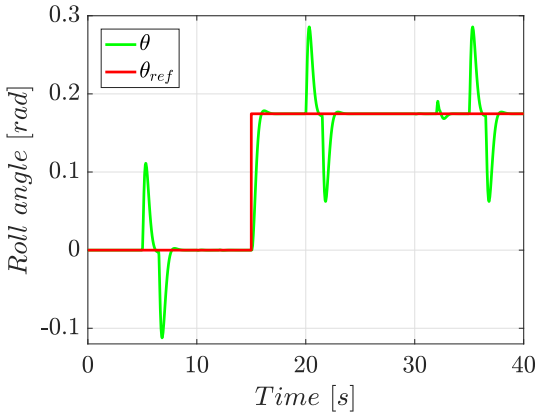
where the system outputs $\mathbf{y}$ have greater penalty than their derivatives ($\dot{\psi}$, $\dot{\theta}$, $\dot{\phi}$); $\mathbf{x_{lqi}}$ have greater penalties than $\mathbf{x}$; and the input vector $\mathbf{u}$ is slightly more penalized

than $\mathbf{x}$. If the penalty of $\mathbf{x_{lqi}}$ is small, the response of the system will tend to be slow. On the other hand, if the penalty of $\mathbf{x_{lqi}}$ is very high, the controller will over-compensate the system.

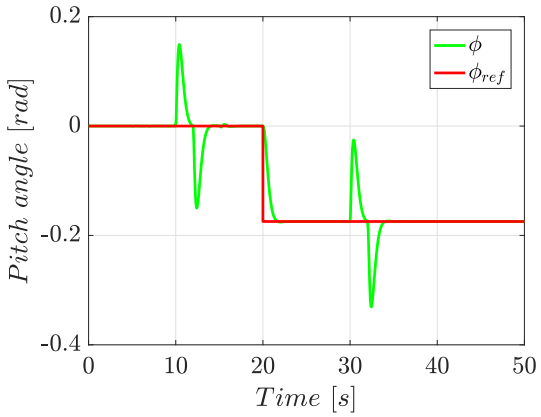
Based on the penalization matrices from (4.32), the feedback gain matrix $\mathbf{K_{lqi}}$ is calculated. The closed-loop response of the system shown in Fig. 4.1 is simulated using MATLAB, and is shown in Fig. 4.4.



(a) Yaw angle response



(b) Roll angle response



(c) Pitch angle response

Figure 4.4: Simulated closed-loop response of stabilize mode controlled by a LQI controller

Here, the system shows responses with a setting time of less than 1.5 s and overshoot of less than 5 %.

H_∞ Controller

The design of the H_∞ controller is based on the weighting filters W_s and W_k, which are set so that (4.26) is satisfied.

In the case of the stabilize mode, the filters are set as

$$W_s = \frac{(22 \cdot 10^{-7})/(10^{-4})}{s + (22 \cdot 10^{-7})} * \mathcal{I}_{3\times 3} \ ,$$
$$W_k = \frac{10^3}{20} \frac{s + 50}{s + (10^3 \cdot 50)} * \mathcal{I}_{4\times 4} \ , \tag{4.33}$$

with which $\gamma = 0.0150$ was obtained. Since the γ value is very far from 1, the filters are normalized with γ as

$$W_s = \frac{1}{\gamma} W_s,$$
$$W_k = \frac{1}{\gamma} W_k, \tag{4.34}$$

after which γ is recalculated, obtaining a value of $\gamma = 0.8542$. The filters W_s and W_k effectively set an upper bound for sensitivity S and the control sensitivity $K_H S$, as shown in Fig. 4.5.

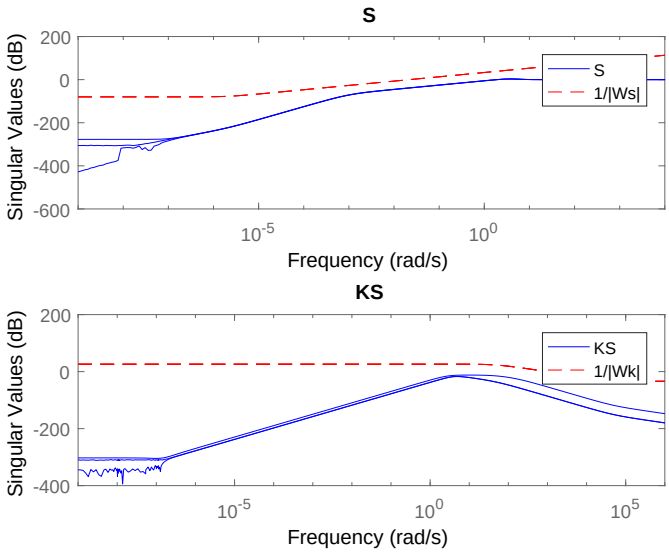


Figure 4.5: Upper bounded singular values of S and $K_H S$ in stabilize mode

The computed dynamic controller K_H is a 13^{th} order system. However, it is possible to find an equivalent system of lower order as seen in Section 4.2. The energy of the HSV of K_H, is shown in Fig. 4.6.

As shown in Fig. 4.6, the last ordered state of the balanced realization of K_H,

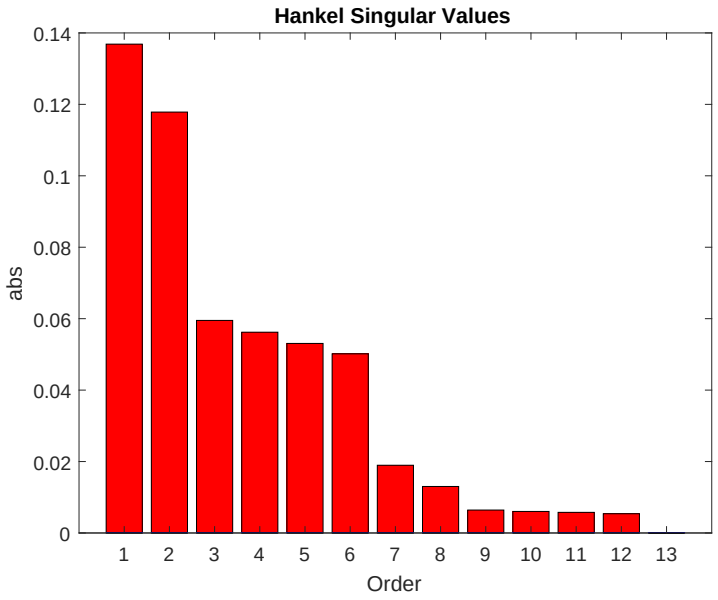


Figure 4.6: HSV energy histogram of K_H in stabilize mode

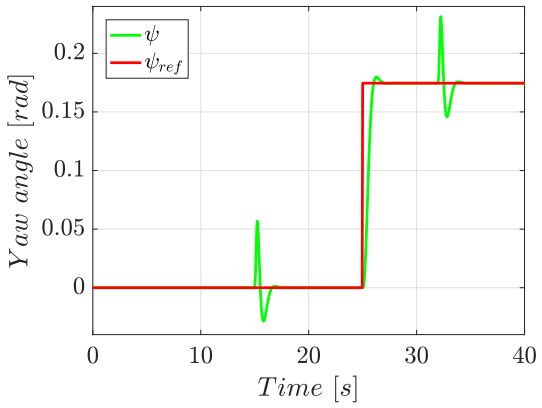
has unnoticeable energy when it is plotted; that means that this state can be truncated from the controller without modifying its dynamics. Thus, the reduced order controller K_H^* is a 12^{th} order system.

Once, the optimal H_∞ controller K_H^* is synthesized using MATLAB, its performance is simulated. For the simulation, the system is subjected to the same disturbances and reference changes set in the LQI simulation. The simulation response is shown in Fig. 4.7.

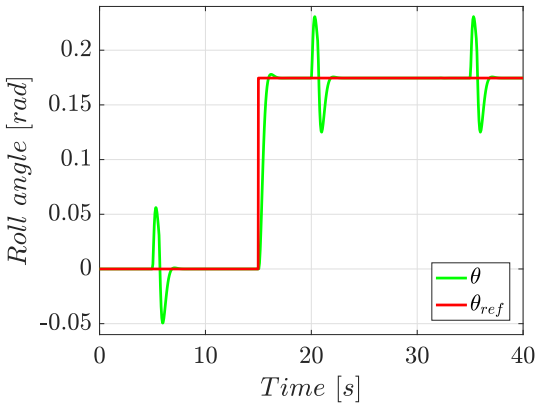
For the H_∞ controller designed for the quadrotor stabilize mode, the closed-loop performance is improved when compared to the LQI controller. The designed K_H^* controller achieves to get responses with less than 1.1 s of setting time and 3 % overshoot.

4.3.2 Altitude Hold Mode

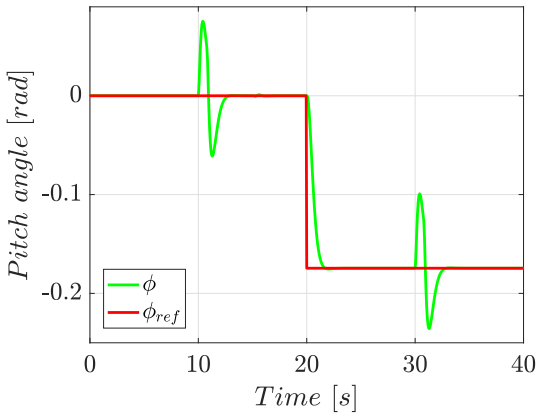
This mode adds the flight altitude of the quadrotor, represented by the variable z, to the quadrotor controlled DoF. Hence, it neglects the dynamics related to the x and y position.



(a) Yaw angle response



(b) Roll angle response



(c) Pitch angle response

Figure 4.7: Simulated closed-loop response of stabilize mode controlled by a H_∞ controller

Dynamic Model

The altitude hold mode, is represented by a dynamic model of 8^{th} order, which states and outputs vectors are

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T, \\ \mathbf{y} &= \begin{bmatrix} z & \psi & \theta & \phi \end{bmatrix}^T. \end{aligned} \tag{4.35}$$

This dynamic system, is represented by the matrices

$$
\begin{aligned}
A &= \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \\
B &= \begin{bmatrix}
0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{J_{zz}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{J_{yy}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{xx}}
\end{bmatrix}^T, \\
C &= \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}, \\
D &= \mathbf{0_{4\times 4}}.
\end{aligned}
\tag{4.36}
$$

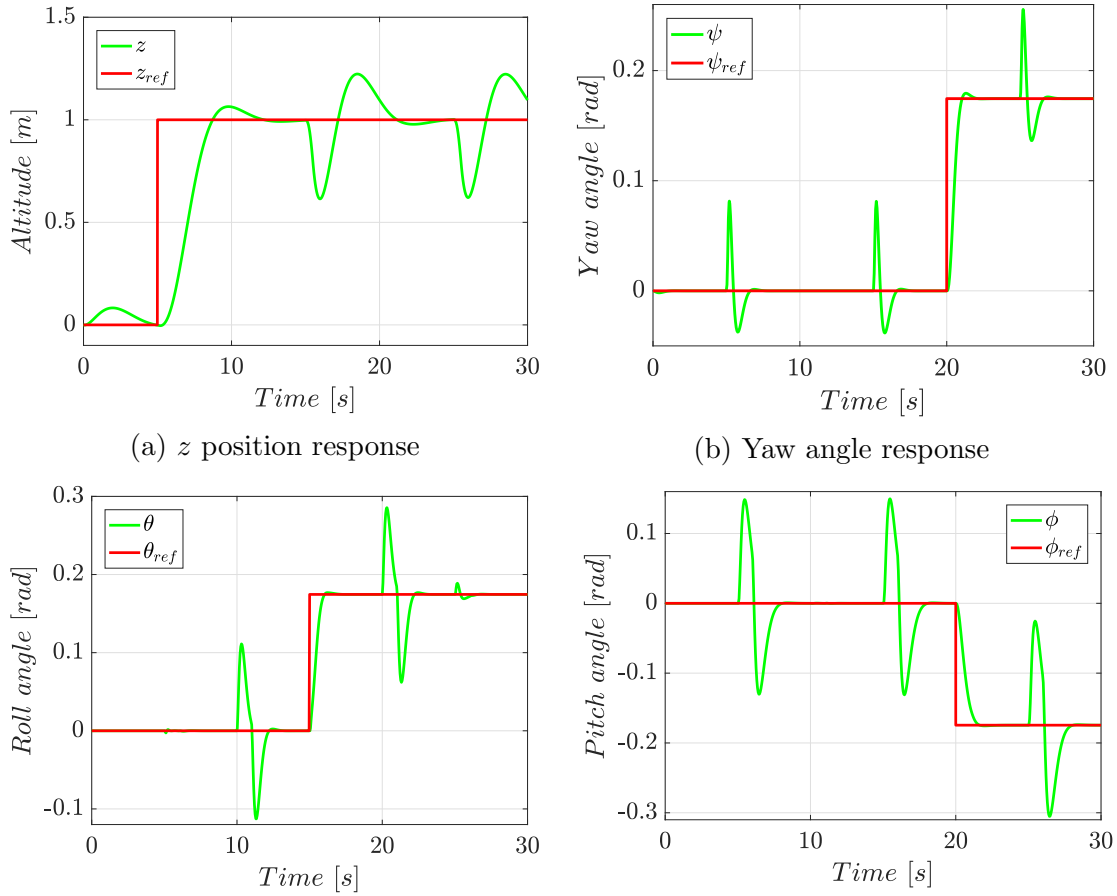
The controllability and observability features of the system are checked using the matrices $\mathcal{C}_\mathcal{G}$ and $\mathcal{O}_\mathcal{G}$. In the altitude hold dynamic model, both matrices have a *rank* of 8, which means that the system is controllable and observable.

LQI Controller

For the altitude hold mode, the penalization matrices $\mathcal{Q}$ and $\mathcal{R}$ are set as

$$
\begin{aligned}
\mathcal{Q} &= \mathcal{I}_{12\times 12}\begin{bmatrix} 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 10 & 10 & 40 & 40 \end{bmatrix}^T, \\
\mathcal{R} &= \mathcal{I}_{4\times 4}\begin{bmatrix} 3 & 3 & 3 & 3 \end{bmatrix}^T,
\end{aligned}
\tag{4.37}
$$

so that the state z has the same penalization as the other outputs, while the penalization of the unmeasured states remains being lower. The integral error of the quadrotor altitude is penalized the same way as the error of yaw, with a not so big penalization, as the pitch and roll dynamics must be prioritized.



(a) z position response

(b) Yaw angle response

(c) Roll angle response

(d) Pitch angle response

Figure 4.8: Simulated closed-loop response of altitude hold mode controlled by a LQI controller

In Fig. 4.8, the closed-loop response of the system $\mathcal{G}_k$ in altitude hold mode with LQI control, is shown. Here, the altitude has a response with a setting time of 11.5 s and an overshoot of 6 %. The attitude response remains the same as it is in the stabilize mode due to the use of the same penalties for the corresponding states and inputs.

H_∞ Controller

In altitude hold mode, the weighting filters which satisfy $\gamma < 1$ are

$$\begin{aligned} W_s &= \frac{(10^{-4})/(10^{-4})}{s + (10^{-4})} * \mathcal{I}_{4\times 4} \ , \\ W_k &= \frac{10^3}{20} \frac{s + 20}{s + (10^3 \cdot 20)} * \mathcal{I}_{4\times 4}. \end{aligned} \tag{4.38}$$

In this case, $\gamma = 0.5976$, forcing to iterate after normalizing the filters. After one iteration, the value of γ is 0.9939. The bounding of the S and $K_H S$ functions, is shown in Fig. 4.9

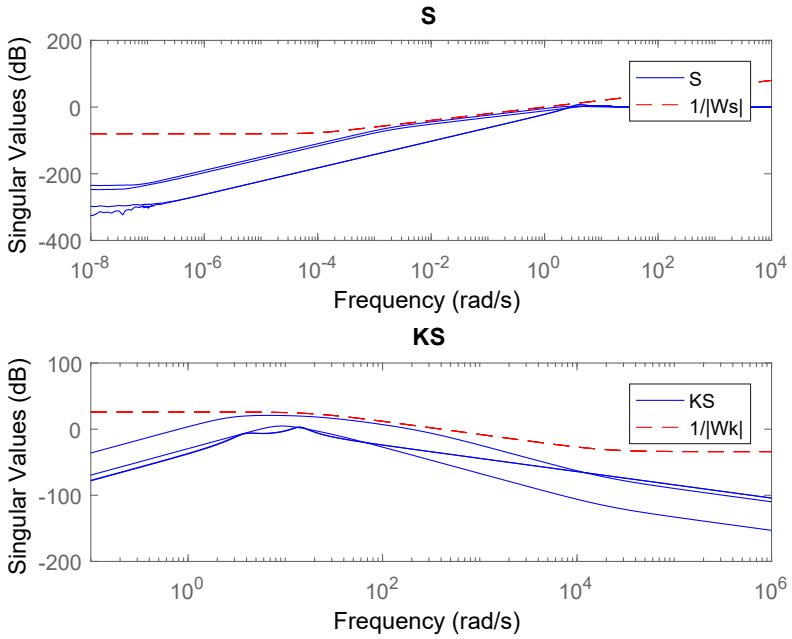


Figure 4.9: Upper bounded singular values of S and $K_H S$ in altitude hold mode

As the dynamic model for altitude hold is a 8^{th} order system with 4 inputs and 4 outputs, the resulting H_∞ controller is a 16^{th} dynamic system.

The K_H HSV energy, exposed in Fig. 4.10, shows that there are 13 states of K_H which are influenced in terms of controllability and observability, and make up a minimal realization K_H^* of K_H.

The simulated response of the designed control system (Fig. 4.11), shows an improvement in setting time and overshoot when compared to the LQI controller response in the same mode. Using the H_∞ controller, the setting time for z is 10.7 s, while the overshoot is 4.5 %.

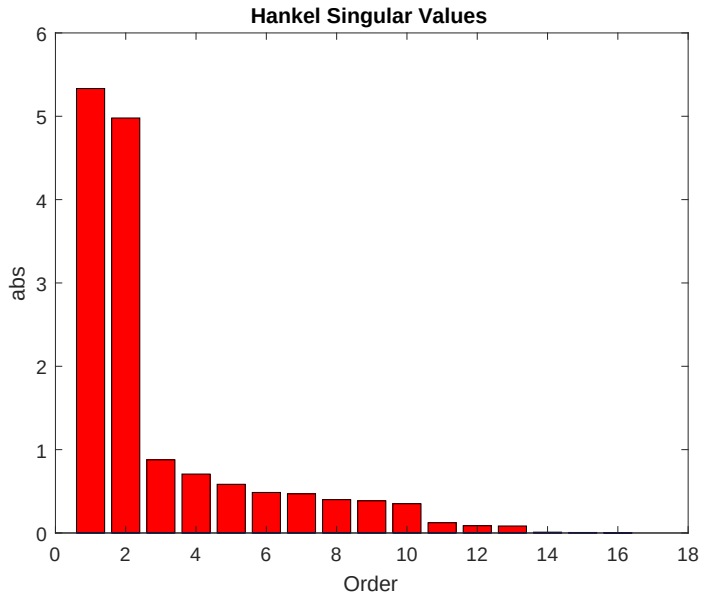
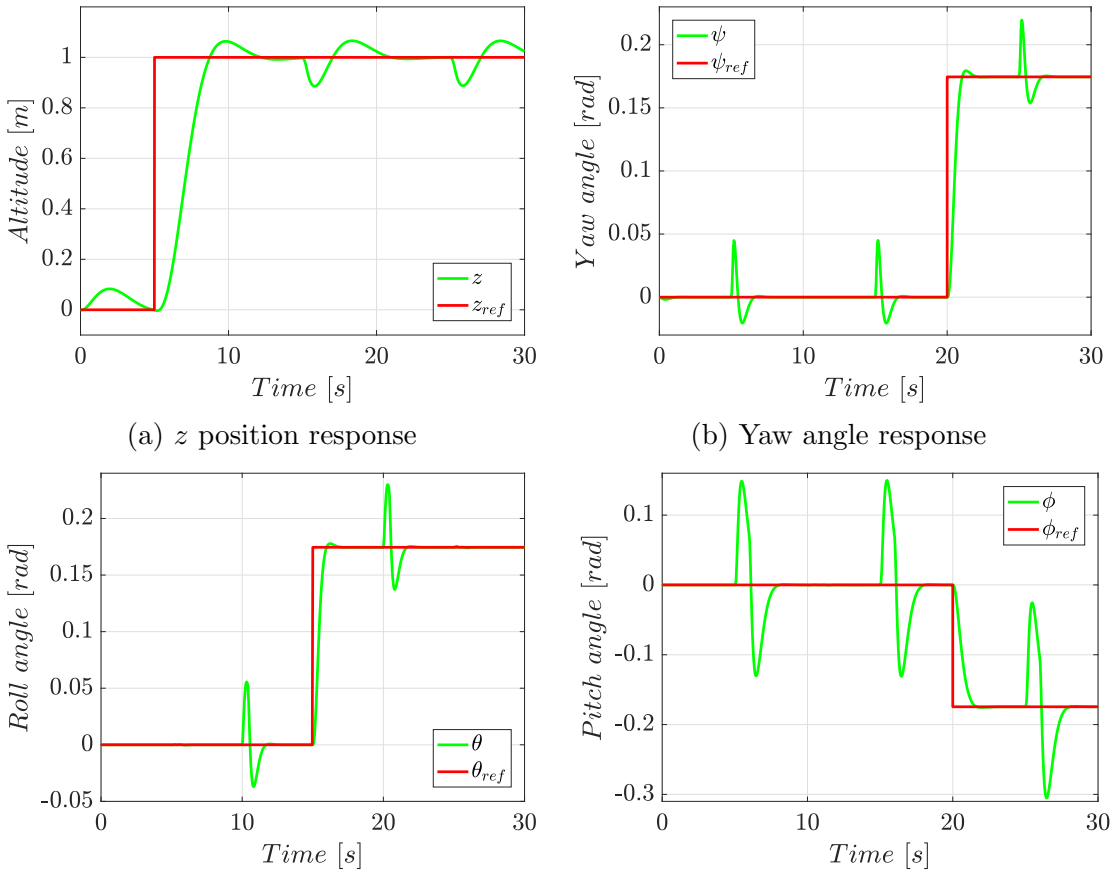


Figure 4.10: HSV energy histogram of K_H in altitude hold mode



(a) z position response

(b) Yaw angle response

(c) Roll angle response

(d) Pitch angle response

Figure 4.11: Simulated closed-loop response of altitude hold mode controlled by a H_∞ controller

4.3.3 GNSS-Dependent Flight Modes

The GNSS-Dependent flight modes take into account the full states dynamics of the quadrotor. These modes include loiter, auto and RTL mode. The designed controllers can be used in any of the three mentioned modes.

Dynamic Model

The dynamic model used for the design of controllers for the GNSS-Dependent flight modes, is the full 12^{th} system introduced in Section 2.3 with all the measurable outputs available, where

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T,$$

$$\mathbf{y} = \begin{bmatrix} x & y & z & \psi & \theta & \phi \end{bmatrix}^T,$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.39}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{m} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{zz}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{yy}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{xx}} \end{bmatrix}^T.$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{4.40}$$

$$D = \mathbf{0_{6\times 4}}.$$

Verifying the controllability and observability of the system using the matrices $\mathcal{C}_{\mathcal{G}}$ and $\mathcal{O}_{\mathcal{G}}$, it is confirmed that these matrices have full-rank. Hence, the system is controllable and observable.

LQI Controller

Similarly to the altitude hold mode, in the GNSS-Dependent modes, the penalization matrices are

$$\begin{aligned} \mathcal{Q} &= \mathcal{I}_{18\times 18} \begin{bmatrix} 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 1 & 0.1 & 10 & 10 & 10 & 10 & 40 & 40 \end{bmatrix}^T, \\ \mathcal{R} &= \mathcal{I}_{4\times 4} \begin{bmatrix} 3 & 3 & 3 & 3 \end{bmatrix}^T, \end{aligned} \tag{4.41}$$

where x and y have the same penalty as z, and analogously happens with $\dot{x}$, $\dot{y}$ and $\dot{z}$.

Using MATLAB, the quadrotor controlled by the designed LQI controller, is simulated. In this simulation, the quadrotor is commanded automatically through some waypoints, which describe a square figure with constant flight altitude.

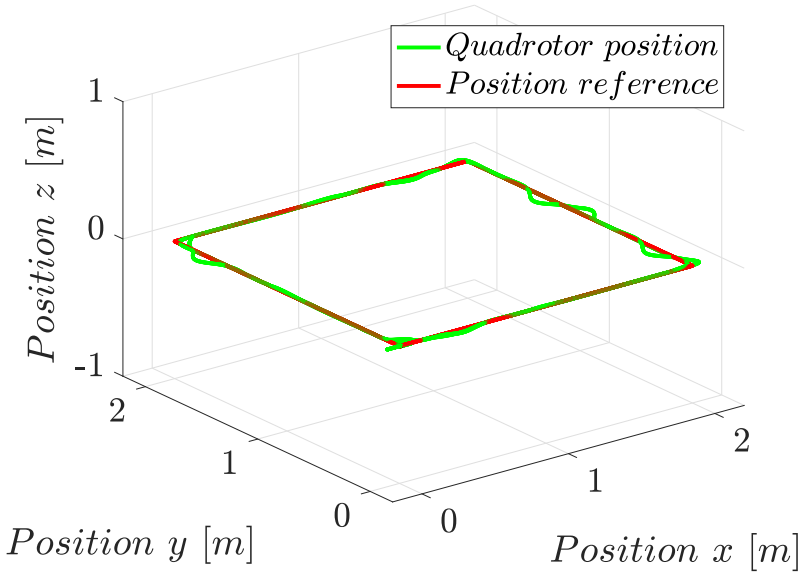
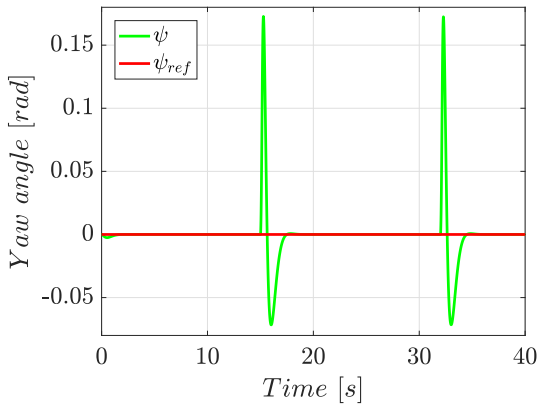
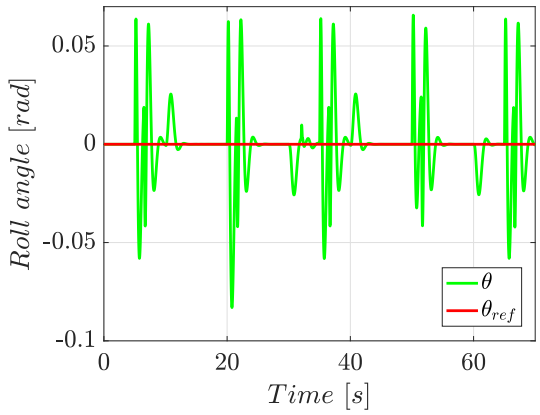


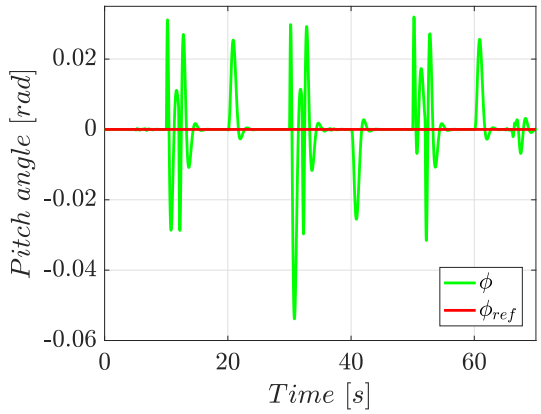
Figure 4.12: Simulated position response of the GNSS-Dependent modes with a LQI controller



(a) Yaw angle response



(b) Roll angle response



(c) Pitch angle response

Figure 4.13: Simulated attitude response of the GNSS-Dependent modes with a LQI controller

As can be seen in Fig. 4.12, the quadrotor is capable of following the desired trajectory while being controlled by the LQI controller in a GNSS-Dependent mode. In addition, the quadrotor manages to recover its track in less than 3.5 s, after suffer the actions of disturbances that periodically simulate the force of the wind.

H_∞ Controller

For the GNSS-Dependent flight modes, the weighting filter W_s is now a 6×6-matrix due to the existence of six output signals in $\mathbf{y}$. Thus, the weighting filters are set as

$$\begin{aligned} W_s &= \frac{(10^{-4})/(10^{-4})}{s + (10^{-4})} * \mathcal{I}_{6\times 6} \ , \\ W_k &= \frac{10^4}{20} \frac{s + 30}{s + (10^4 \cdot 30)} * \mathcal{I}_{4\times 4}. \end{aligned} \tag{4.42}$$

Wigh these weighting filters, $\gamma = 0.5976$. After one iteration, with normalized weighting filters, the γ value becomes equal to 0.999. Thus, the sensitivity and control sensitivity functions are closely upper bounded by the weighting filters, as shown in Fig. 4.14.

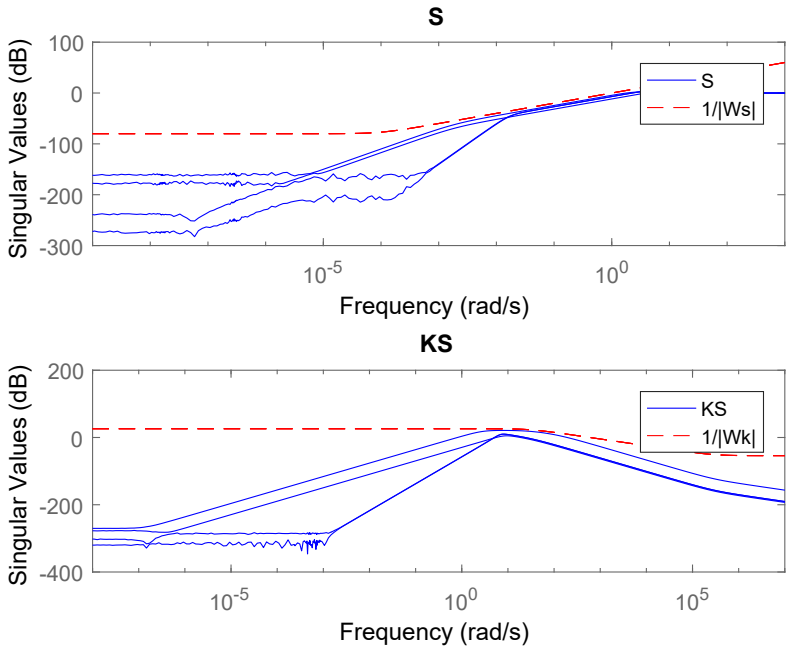


Figure 4.14: Upper bounded singular values of S and $K_H S$ in GNSS-Dependent modes

The synthesized H_∞ controller K_H for the GNSS-Dependent modes has 20 states.

An equivalent controller K_H^* of lower order is found, using the HSV energy shown in Fig. 4.15.

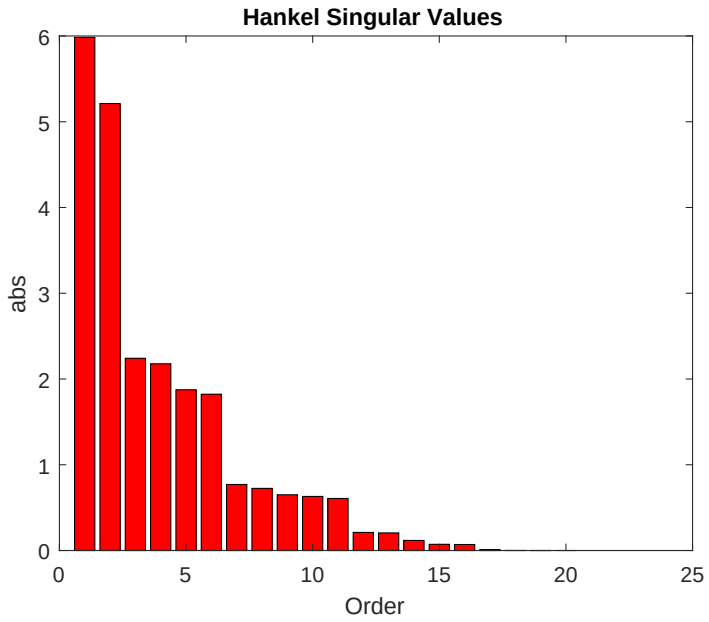


Figure 4.15: HSV energy histogram of K_H in GNSS-Dependent modes

The system K_H has four unnoticeable states in terms of controllability and observability, as seen in Fig. 4.15. Therefore, the reduced order controller K_H^* is the subsystem of K_H which includes just the 16 noticeable states.

The simulation of the designed H_∞ controller with the quadrotor non-linear dynamics, is also developed using a reference of waypoints describing a square with sides of 2 m, and a constant altitude.

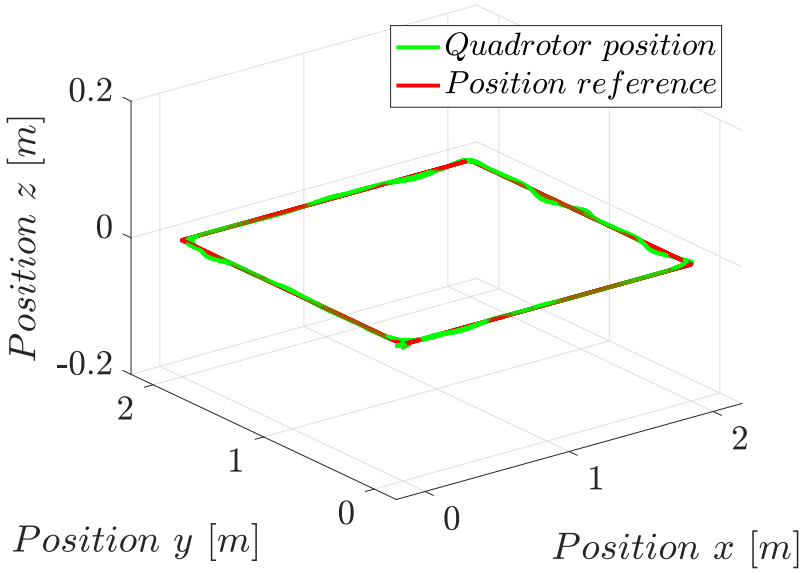
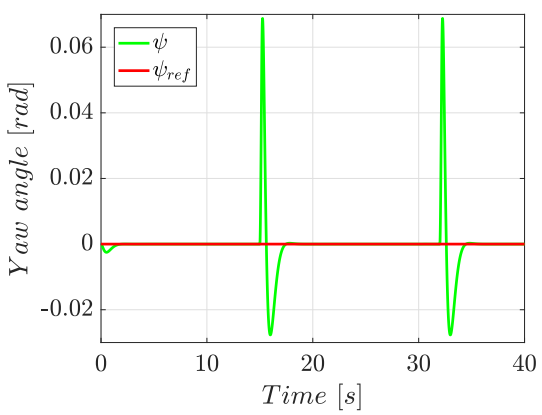
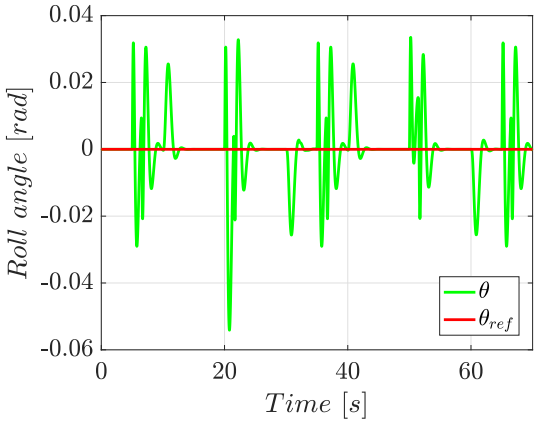


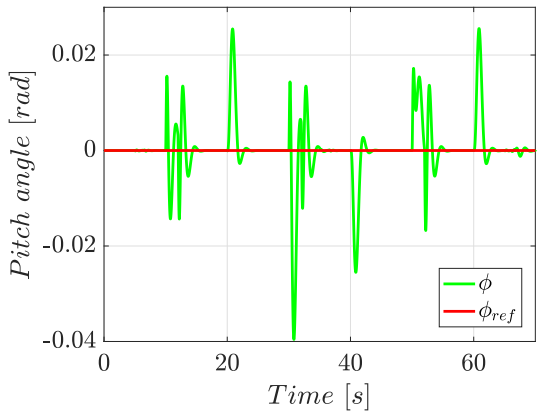
Figure 4.16: Simulated position response of the GNSS-Dependent modes with a H_∞ controller



(a) Yaw angle response



(b) Roll angle response



(c) Pitch angle response

Figure 4.17: Simulated attitude response of the GNSS-Dependent modes with a H_∞ controller

The simulated control system manages to follow the position reference, despite being affected by disturbances, as seen in Fig. 4.16. For disturbances, it shows a setting time of 2.8 s.

4.4 State Estimation Through Kalman Filter

The quadrotor dynamics are sensed exclusively using the on-board smartphone sensors. These sensors have different sample frequencies and poor accuracy. On the other hand, the LQI controller needs a full-state feedback, but it is not posible to get reliable measurements of all the components in $\mathbf{x}$ in a smartphone. Hence, it is necessary to use estimation algorithms, as a Kalman filter. The principle of separation allows the design of controllers and state estimators independently.

4.4.1 Attitude Estimation

The Android API implements a Kalman filter for attitude estimation using the raw data deliverd by the smartphone accelerometer, gyroscope and magnetometer, as exposed in [79]. Using the quaternion Q_s delivered by the Rotation virtual sensor included in the Android SDK, it is obtained an absolute orientation representation with respect to the Earth frame [80], with

$$\begin{aligned} Q_s &= \mathrm{e}^{(\alpha/2)(u\,\vec{i}+v\,\vec{j}+w\,\vec{k})} \\ &= \begin{bmatrix} \kappa_0 \sin(\alpha/2) \\ \kappa_1 \sin(\alpha/2) \\ \kappa_2 \sin(\alpha/2) \\ \cos(\alpha/2) \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \end{aligned} \tag{4.43}$$

where α is the amount of degrees the quaternion is rotated around the axis $\kappa_0\,\vec{i} + \kappa_1\,\vec{j} + \kappa_2\,\vec{k}$. The rotation matrix $\mathbf{R}_{\mathbf{b}}^{\mathbf{w}}$ can be defined using Q_s as

$$\mathbf{R}_{\mathbf{b}}^{\mathbf{w}} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_1q_2-q_0q_3) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_0q_1+q_2q_3) & 1-2(q_1^2+q_2^2) \end{bmatrix}. \tag{4.44}$$

Comparing (2.19) and (4.44), the Euler angles (ψ_m, θ_m, ϕ_m) are then obtained from the quaternion Q_s as

$$\begin{bmatrix} \psi_m \\ \theta_m \\ \phi_m \end{bmatrix} = \begin{bmatrix} atan2(2(q_3q_2+q_0q1), 1-2(q_1^2+q_2^2)) \\ arcsin(2(q_3q_1-q_2q_0)) \\ atan2(2(q_3q_0+q_1q2), 1-2(q_0^2+q_1^2)) \end{bmatrix}. \tag{4.45}$$

4.4.2 Position Measurement

The smartphone used in this project, can measure its position with respect to the Earth-frame. For this task, it counts with a GNSS receiver and a barometer. The x_m and y_m position measurements, with respect to the X_W and Y_W axes, are acquired using the GNSS receiver.

Each coordinates sample is initially set in an ellipsoidal representation of decimal degrees, following the WGS84 coordinate system. Then, the coordinates are converted to a bi-dimensional representation in meter units using the cartographic projection Magna-Sirgas (projection EPSG:3115).

The altitude measurements z_m are acquired using the barometric pressure sensor which delivers the pressure value p_k in hPa units. This signal is converted to meters as

$$z_m = 44330 \left(1 - \frac{p_k}{p_0}^{1/5.255}\right) \ [m], \tag{4.46}$$

where p_0 is the atmospheric pressure at sea level [81].

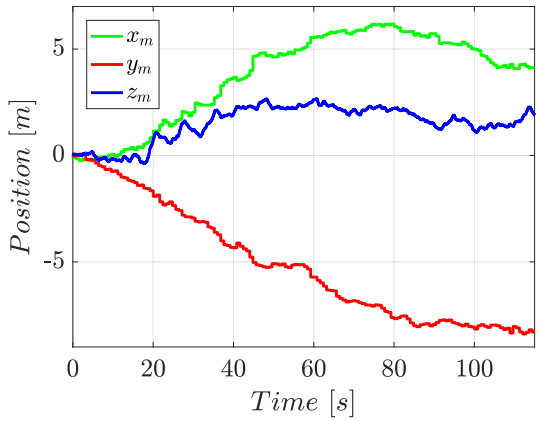
The smartphone GNSS receiver and barometer were tested statically, leaving the phone in a stable position for 100 minutes, with an ambient temperature of 22 $°C$ and a UV index of 3. The results of the test are shown in Fig. 4.18

As seen in Fig. 4.18a, even though the phone was kept in a static position, the x_m and y_m position signal received from the GNSS receiver shows a change of about 6 m in both the X_W and Y_W directions, after just 60 s. The z_m position, however, has sudden changes in short periods of time, but without reaching errors greater than 2.5 m. The attitude results, Fig. 4.18b, show stable signals for the three Euler angles. The ψ_m and θ_m angles suffer from a drift of about 0.04 and -0.02 rad respectively, while the initial drift suffered by ϕ_m is just -0.01 rad.

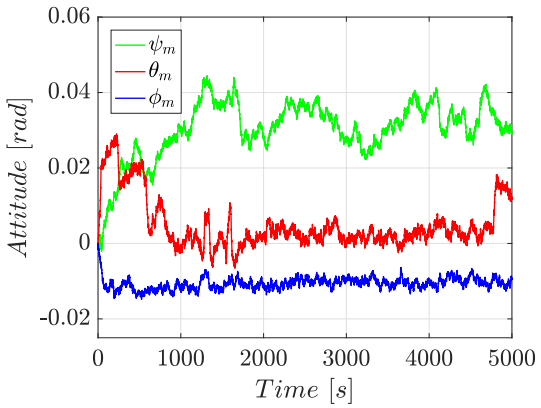
Despite being under a clear sky, where the smartphone could have visibility of more than six satellites of the GPS and GLONASS constellations, the GNSS accuracy value, delivered by the receiver in the smartphone, is always more than 9 m as seen in Fig. 4.18c.

4.4.3 States Estimation

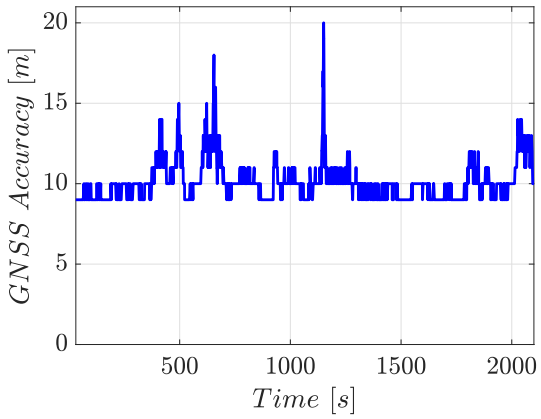
In order to estimate all the components of the state vector $\mathbf{x}$, a Kalman filter is designed. The Kalman filter is an iterative algorithm that estimates the observable



(a) Position drift



(b) Attitude drift



(c) GNSS receiver accuracy

Figure 4.18: Static test of the smartphone GNSS receiver and barometer

(and not measurable) states of a system, using the measurable variables, the inputs and the variances of the noises that affect the system.

This filter is designed using the linearized dynamic model of the quadrotor from Section 2.3, where $\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^T$, and using the discretized representation $\mathcal{G}_k$.

The algorithm starts calculating a prediction of the state $\hat{\mathbf{x}}_k^-$ and its covariance P_k^- as

$$\hat{\mathbf{x}}_k^- = A_k \hat{\mathbf{x}}_{k-1} + B_k \mathbf{u}, \tag{4.47}$$

$$P_k^- = A_k P_{k-1} A_k^T + Q_k, \tag{4.48}$$

where $\hat{\mathbf{x}}_{k-1} = \hat{\mathbf{x}}(k-1)$ is the previous estimated state, P_{k-1} the previous error covariance matrix and Q_k the process variance. The state prediction is then corrected

using the Kalman gain vector

$$K_k = P_k^- H^T (H P_k^- H^T + R_k)^{-1}, \tag{4.49}$$

and updating the state estimation $\hat{\mathbf{x}}_k$ and its covariance P_k, based on the measurements ϑ_k as

$$\begin{aligned} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + K_k(\vartheta_k - H\hat{\mathbf{x}}_k^-), \\ P_k &= (\mathcal{I} - K_k H)P_k^-, \end{aligned} \tag{4.50}$$

where R_k is the measurement covariance matrix, $\mathcal{I}$ is the identity matrix and H is the matrix that satisfies

$$\vartheta_k = H\mathbf{x}_k. \tag{4.51}$$

The measurements vector ϑ_k is set as

$$\vartheta_k = \begin{bmatrix} x_m & y_m & z_m & \psi_m & \theta_m & \phi_m \end{bmatrix}^T, \tag{4.52}$$

so matrix H is defined as

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.53}$$

This algorithm is executed once every sampling period, after which, the controller can use the vector of estimated states $\hat{\mathbf{x}}_k$ as feedback.

4.5 Conclusions

The control and state estimation algorithms design procedure is presented in this chapter. The concepts of representation of a system as a state space model, controllability and observability, were introduced in the first part. Given the need to achieve trajectory tracking with the quadrotor, a LQR controller with integral feedback LQI is designed. This controller ensures zero steady-state error. Also, a controller based on the H_∞ synthesis is designed. The performance of these controllers is simulated acting with the non-linear dynamic model of the quadrotor. As a result of these simulations, the H_∞ controller showed a better performance when subjected to exactly the same disturbances and reference changes as the LQI controller. Finally, due to the principle of separation, a state estimator based on a Kalman filter is designed taking into account the accuracy of the smartphone measurements.

Chapter 5

Implementation and Results

The smartphone-based quadrotor control system developed in this project, is subjected to flight tests. the results of which are shown in this chapter. This chapter describes the Android application developed for the execution of the flight controller, detailing its activities and functions. In order to build a complete UAS, a GCS is developed and a communication channel is established between the Android application executed on board, and the desktop application (GCS). The quadrotor is tested for each designed flight mode and controller.

In Section 5.1, the flight control application, developed to run on the Android operating system, is detailed. Here, the activities of which the application is composed, and its functions, are detailed.

The components and features of the GCS, are shown in Section 5.2. This section details the characteristics of the developed desktop application, in addition to the communication channel established with the Android application, and therefore with the quadrotor. Finally, in Section 5.3, the results obtained after the execution of the flight tests in the quadrotor, are presented. These results show the quadrotor performance after being subjected to disturbances and changes of references, while executing the designed control and estimation algorithms.

5.1 Android Application

The flight controller application, designed for being executed in an Android smartphone, is exposed in this section. The application is based on nine activities, which allow the user to perform tests and data acquisition with or without being connected to the smartphone-to-ESC gateway. The application activities composition and its navigation flow, are shown in Fig. 5.1.

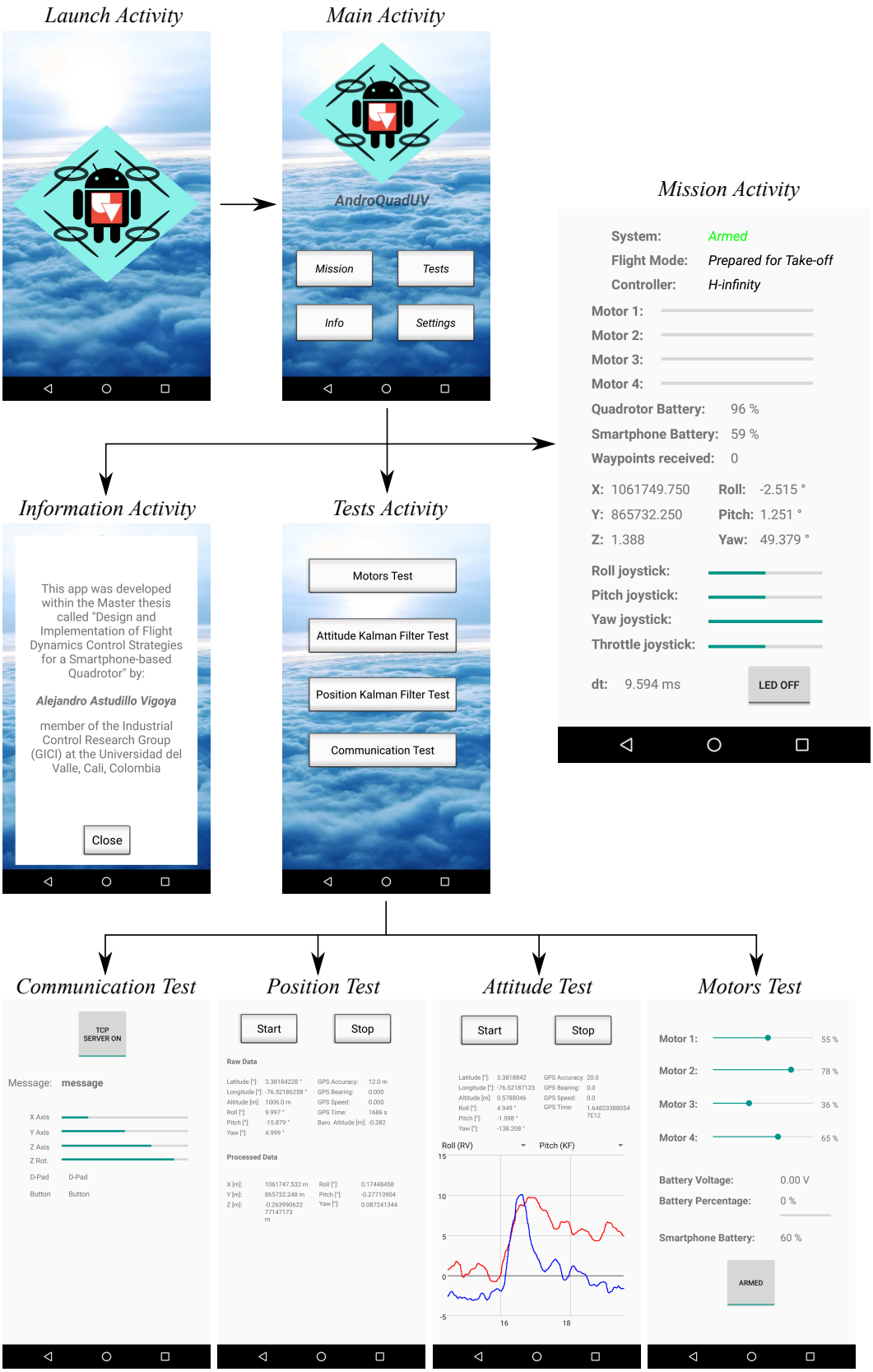


Figure 5.1: Activities diagram in the Android application

The activities description, based on the navigation flow shown in Fig. 5.1, is detailed below.

Launch Activity

The launch activity is the activity executed first when starting the application. This activity is executed for around 3 seconds. Here, the initial load of the application is completed, and the required permissions are requested. These are the location and storage permissions.

Main Activity

This activity is the main interface of the application. From here, the user can access the mission, tests and information activities, as well as change application settings. If the interface does not receive any command, and the screen is not touched after 10 seconds of execution, the mission activity is executed automatically.

Mission Activity

The mission activity is the activity responsible for the execution of the flight controller. During the execution of this activity, a TCP server is created, which waits for the remote connection of the GCS. Additionally, the sensors of the smartphone, and the control and estimation algorithms, are initialized and are available to the flight controller.

Once the communication channel with the GCS is established, the flight controller initiates the decoding of frames coming from the GCS, which include commands of: arming or disarming of motors, flight mode change, waypoints setting, references changes, remote controller commands, and quadrotor data query.

For debug purposes, the mission activity also has an interface that shows the flight mode in which the quadrotor is located, the current type of controller being executed, the status of the motors, current position and attitude, level of the smartphone and quadrotor batteries, number of received waypoints, and remote control commands. Also, the communication between the smartphone and the gateway can be tested using the button 'Led On/Off', which sends a command to turn on or off the built-in led of the Arduino Mega ADK.

Information Activity

This activity shows information about this project and its author, so that the user of the application knows its origin.

Tests Activity

The preliminary tests of the smartphone-based quadrotor system can be accessed from this activity. These tests are: the communication test, position sensing test, attitude estimation test, and quadrotor motors test.

Communication Test Activity

The communication test activity allows the user to check if the communication channel between the Android application and the GCS, is working. Here, the user can turn on or off the TCP server, and watch some commands sent by the remote controller, through the GCS.

Position Test Activity

The position test activity initializes the smartphone GNSS receiver and starts the GNSS data acquisition. This activity shows the raw GNSS data and its projection to the Magna-Sirgas coordinate system, in addition to the altitude value acquired from the barometer and the attitude from the rotation virtual sensor.

Attitude Test Activity

This activity shows the data obtained from the virtual rotation sensor, as well as the raw position of the GNSS, to the user. It also allows the simultaneous plot of two of these signals, which can be changed during the acquisition execution.

Motors Test Activity

Finally, the motors test activity allows the user to test the established communication between the smartphone and the Arduino Mega ADK. In this activity, the user can configure the PWM signal sent to each motor separately, in addition to checking the status of the quadrotor and smartphone batteries. The motors thrust and torque tests, shown in Section 3.2, are developed using this activity to command the motors thrust.

5.2 Ground Control Station (GCS)

The GCS is the land-based control station that adds monitoring and remote control functions for a UAV. For this project, the GCS software is designed using the Java programming language. The Java language is chosen in order to add portability to the application.

As mentioned earlier, the Android application creates a TCP server in order to let the GCS, acting as a TCP client, exchange data packets with the smartphone. The communication between the Android application and the GCS, is based on a packet-based communication using the TCP and IP protocols.

The user can interact with the GCS through the mouse or touchscreen of the computer, or using a control for video games (remote controller) connected through the USB interface of the computer. Using button sequences commands through the remote controller, the user can establish and finish the connection with the quadrotor, change the flight mode, and arm or disarm the motors. Also, the quadrotor position and attitude references can be changed using the remote controller joysticks.

The GCS interface, is composed by six main panels, shown in Fig. 5.2, and a Communication tab, which are are described below.

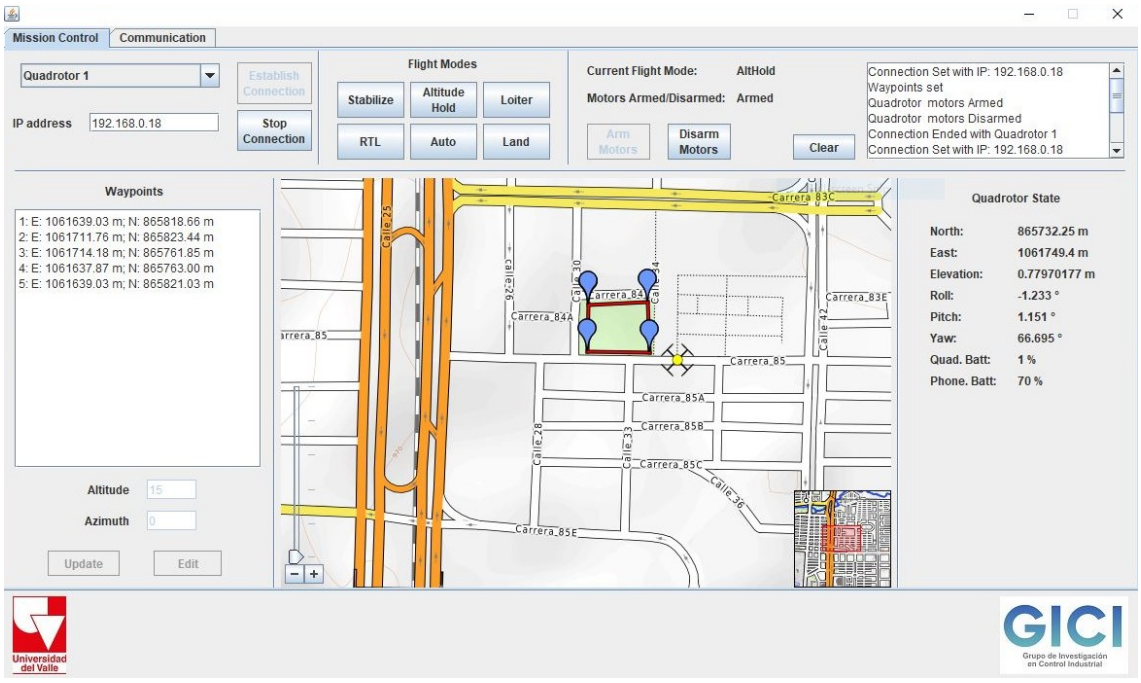


Figure 5.2: Ground Control System main interface

Connection Panel

Located at the upper left edge of the GCS interface, this panel allows the user to select the quadrotor to which the user wants to connect and establish its IP address. This is the only panel enabled after the execution of the GCS. Once the connection with the quadrotor is established, the other panels are enabled.

Flight Modes Panel

Once the connection is established, the user can select the desired flight mode using the buttons in the Flight Modes panel. After a flight mode button is pushed, the user is prompted to confirm its will to change the quadrotor flight mode. When a confirmation of flight mode change is received from the smartphone, the current flight mode, in the Status and Console Panel, is updated.

Status and Console Panel

The Status and Console panel is located in the upper right edge of the GCS interface. This panel, shows the current status of the motors and flight mode. Additionally, this panel includes buttons for arming/disarming the quadrotor motors, and a console area. This console, show relevant information regarding the connection, waypoints, flight mode, and motors status.

Map Panel

In order to check the current position of the quadrotor, a map is displayed in the Map Panel, located in the center of the GCS interface. This display is based on the JXMapViewer2 library and the Open Street Map (OSM) project. In the map, the user can select the waypoints that the quadrotor will visit, which will be automatically connected in the graph, and added to the waypoints list. Also, a quadrotor icon is displayed in the current geolocated position of the quadrotor so the user can track it. When the GCS has internet access, the map provider from OSM is set. Otherwise, the map is got from downloaded map tiles which are included in the GCS resources. The offline map is restricted to the area of the city of Cali, Colombia, while the online map is available for all the world. The map is projected using the Pseudo-Mercator projection.

Waypoints Panel

The Waypoints panel is located on the left flank of the Map panel. This panel shows the list of waypoints dynamically, so it is updated each time a waypoint is added

in the Map panel. Also, the waypoints coordinates can be edited by the user, and their position is automatically updated in the map.

Quadrotor State Panel

The quadrotor state data is composed by the quadrotor position, the quadrotor attitude, the quadrotor battery level and the smartphone battery level. This data is shown in the Quadrotor State panel, located on the right of the Map panel. The quadrotor state values are updated each 100 ms.

Communication Tab

Additionally, the GCS interface have a Communication tab, which includes a Controller Settings Panel shown in Fig. 5.3. This panel shows the connection status of the GCS with the remote controller. Also, it presents the current value of the remote controller joysticks and its buttons state.

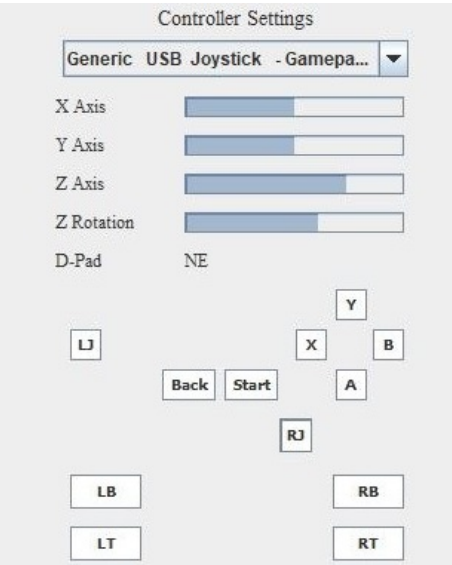


Figure 5.3: Controller settings in GCS

In order to obtain a wide range of wireless communication between the Android application and the GCS, an external high-gain WLAN antenna is used. Using this antenna, the quadrotor was tested to fly 200 meters from the GCS with line of sight, without failing to keep a constant communication with the GCS.

The complete UAS built in this project is shown in Fig. 5.4. This UAS is composed by a smartphone-based quadrotor UAV, a GCS executing a java application for monitoring and remote control, and a WLAN-based communication channel between both of them.

Figure 5.4: Complete implemented Smartphone-based UAS

5.3 Flight Tests

The control and estimation algorithms designed in Chapter 4, are implemented in the Android application that is executed in the smartphone on board the quadrotor. Using the smartphone-based quadrotor, the controllers are tested subjecting the system to disturbances and reference changes, within their flight modes. The results of these tests are shown in this section. The control signals $\mathbf{U} = \begin{bmatrix} T_u & \tau_\psi & \tau_\theta & \tau_\phi \end{bmatrix}^T$, and the *PWM* signals, logged during each of the tests, are shown in Appendix C.

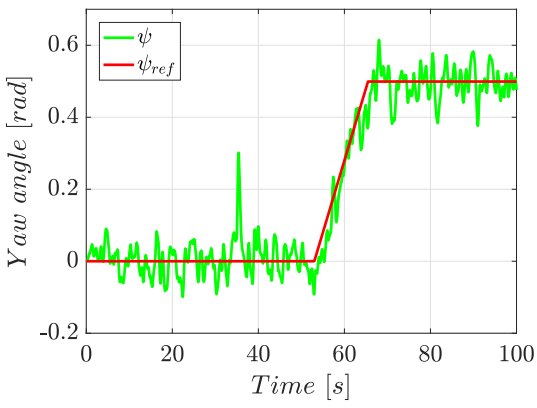
5.3.1 Stabilize Mode

The smartphone-based quadrotor, in its stabilizing flight mode, is subjected to tests, in which a test bench based on a tripod and a ball head is used. The stabilize mode test bench, shown in Fig. 5.5, was designed and built within the framework of this project. This test bench allows only rotational movements of the quadrotor, blocking its translational movements.

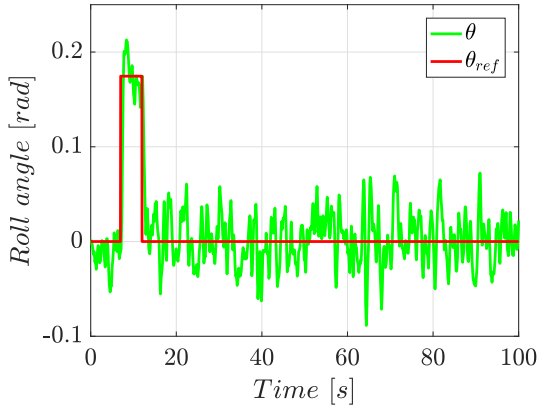
LQI Controller

The LQI controller in stabilize mode was tested by manually applying disturbances in the form of torques about the x, y and z axes, and successive reference changes from the remote controller.

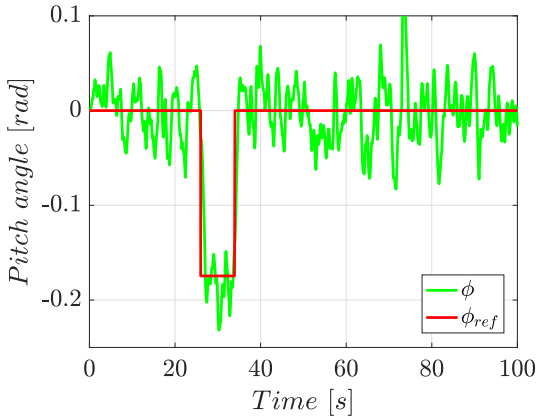
Figure 5.5: Stabilize mode test bench



(a) Yaw angle response



(b) Roll angle response



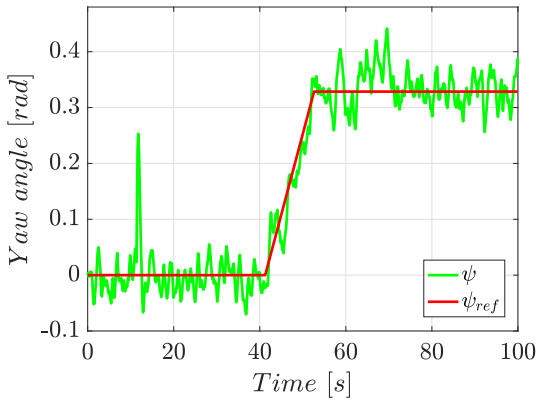
(c) Pitch angle response

Figure 5.6: Closed-loop response of stabilize mode controlled by a LQI controller

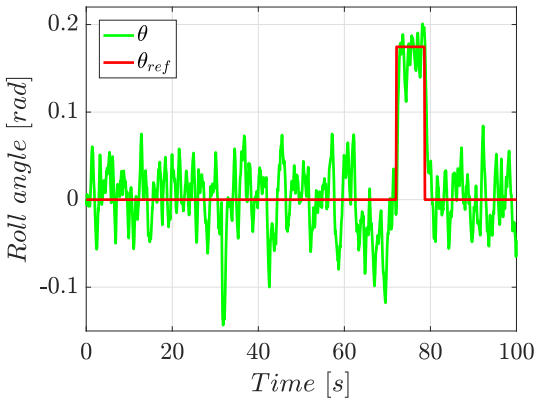
In Fig. 5.6, the quadrotor response to the tests in stabilize mode, is shown. The yaw angle ψ is subjected to a disturbance at $t = 34\ s$, and a slow reference change between $t = 53\ s$, to $t = 64\ s$. In the case of the roll angle θ, the reference change is done momentarily, after $t = 8\ s$, and a disturbance is applied at $t = 67\ s$. On the other hand, at $t = 72\ s$, a disturbance in the torque τ_ϕ is applied, and at $t = 24\ s$ the pitch angle ϕ reference is changed.

H_∞ Controller

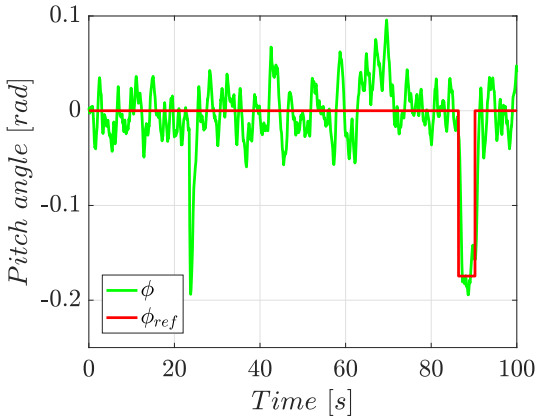
The same test, as with the LQI controller, is developed with the H_∞ controller.



(a) Yaw angle response



(b) Roll angle response



(c) Pitch angle response

Figure 5.7: Closed-loop response of stabilize mode controlled by a H_∞ controller

In this case, as seen in Fig. 5.7, the disturbances are applied at $t = 11\ s$ for ψ, $t = 23\ s$ for ϕ, and $t = 32\ s$ for θ. However, the reference of ψ is changed between $t = 41\ s$ and $t = 53\ s$, while the references change in θ and ϕ are applied at $t = 71\ s$ and $t = 86\ s$ respectively.

Performance

Both of the controllers were capable of stabilize the quadrotor attitude dynamics. For the stabilize mode tests, the performance of the controllers is evaluated taking into account the setting time and overshoot of the controlled signals. These indices are shown in Table 5.1.

Table 5.1: Performance indices - Stabilize mode tests

Controller	Controlled variable	Setting time [s]	Overshoot [%]
LQI	ψ	1.39	10.52
	θ	1.16	17.39
	ϕ	1.12	13.81
H_∞	ψ	1.18	9.84
	θ	1.03	8.40
	ϕ	1.29	5.67

The performance indices for the stabilize mode tests show that both the LQI and H_∞ controllers designed for the stabilize flight mode, have similar setting time, but different overshoot percentage. The H_∞ controller is the one with the lower overshoot and setting time, except for the setting time of the ϕ signal, which is lower when using the LQI controller.

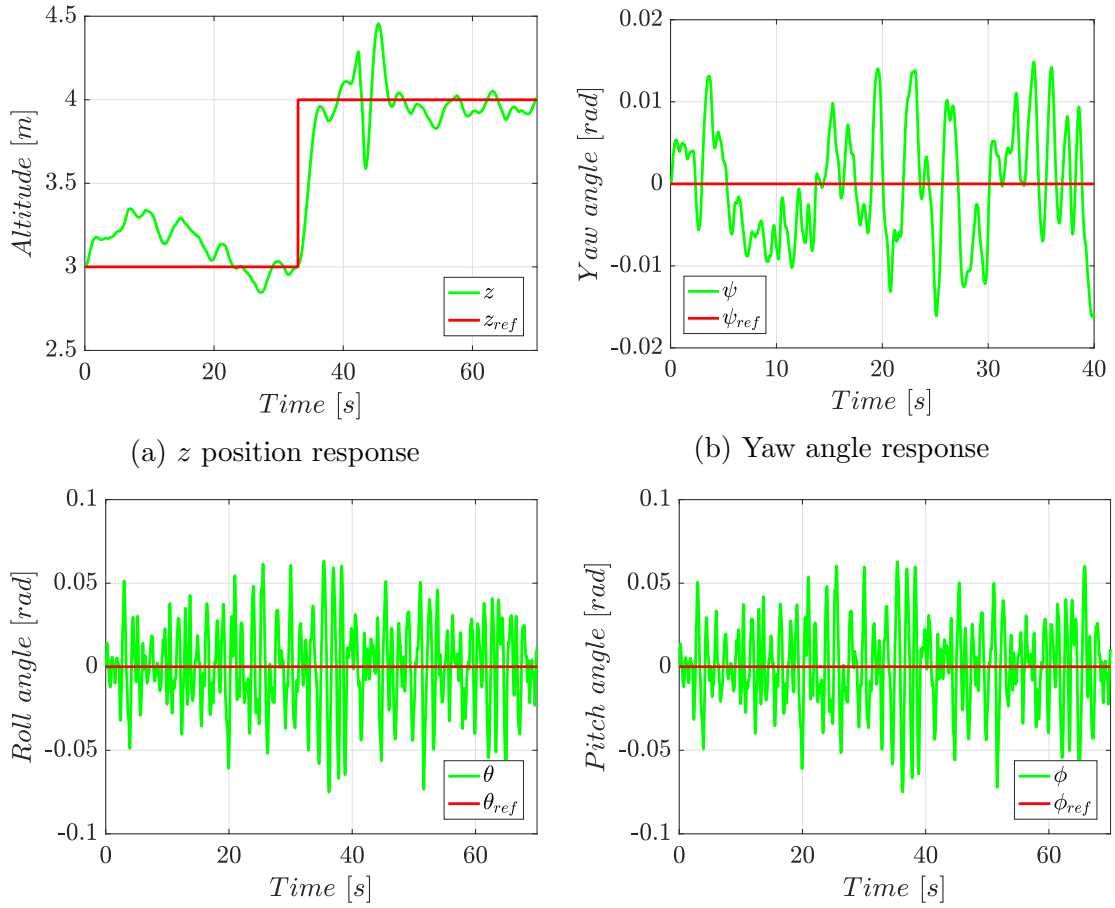
5.3.2 Altitude Hold Mode

The altitude flight mode tests were performed on a football field of the Universidad del Valle. This place was chosen because it is an open space free of obstacles, where the quadrotor has total freedom to move without being at risk of colliding with something or somebody.

In these tests, the only variable that was intentionally affected was the quadrotor altitude z. Hence, the references regarding the roll, pitch and yaw angles, are left in zero during the tests.

LQI Controller

As seen in Fig. 5.8, the quadrotor altitude reference was changed at $t = 33\ s$. A disturbance was applied by manually pulling the quadrotor down with a rope at $t = 42\ s$.



(a) z position response

(b) Yaw angle response

(c) Roll angle response

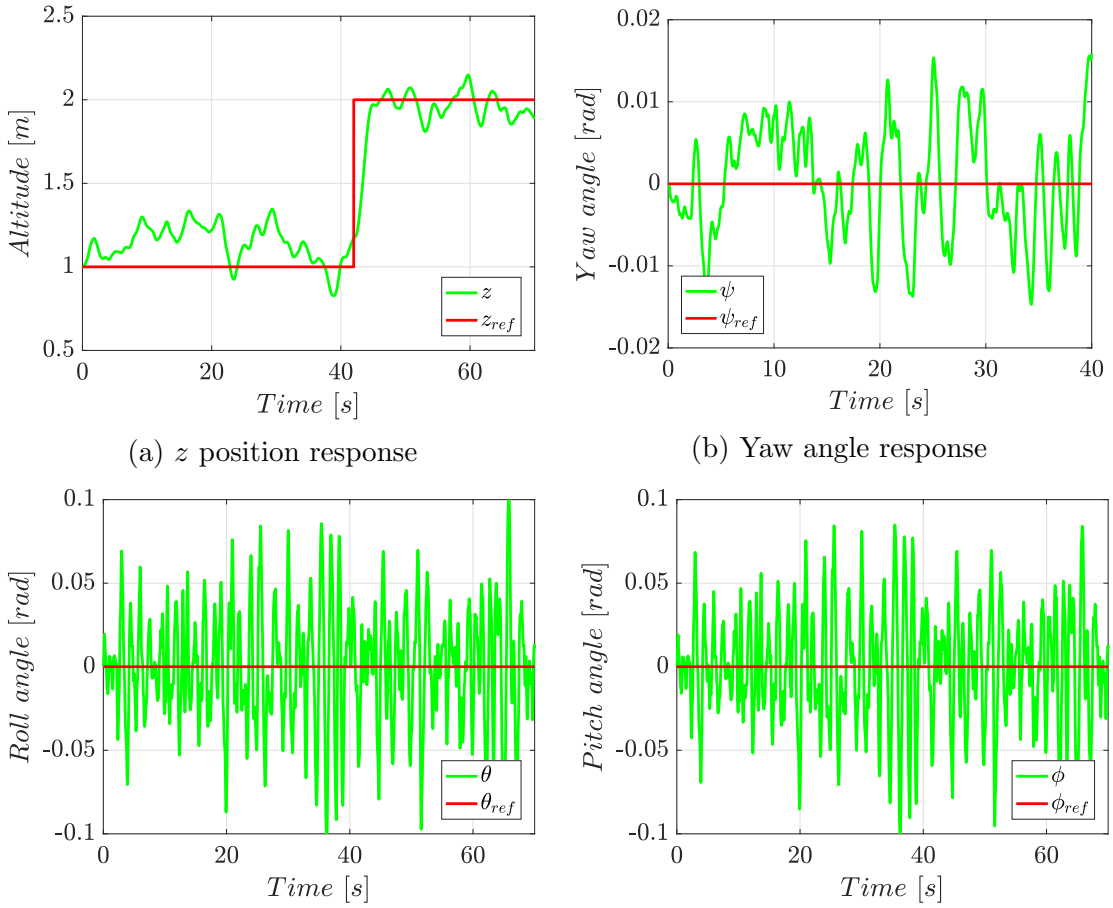
(d) Pitch angle response

Figure 5.8: Closed-loop response of altitude hold mode controlled by a LQI controller

H_∞ Controller

Under approximately the same environmental conditions, the altitude hold mode test is developed with the H_∞ controller being executed in the smartphone on board the quadrotor.

In Fig. 5.9 is seen that the disturbance was applied, pulling the rope, at $t = 49\ s$, while the altitude reference was changed at $t = 42\ s$.



(a) z position response

(b) Yaw angle response

(c) Roll angle response

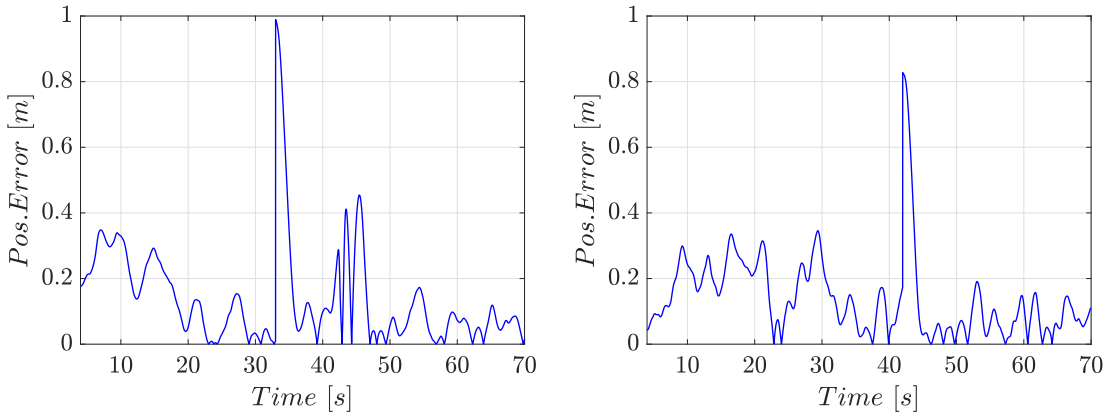
(d) Pitch angle response

Figure 5.9: Closed-loop response of altitude hold mode controlled by a H_∞ controller

Performance

The quadrotor altitude control is achieved when implementing both the LQI and H_∞ controllers. The position error is calculated as the Euclidean distance between the quadrotor position and the reference position in each time step. The quadrotor achieves errors less than 0.4 m during constant reference tracking, with both controllers. However, as shown in Fig. 5.10, during the reference change, the error is lower when executing the H_∞ controller.

In Table 5.2, the overshoot and setting time indices of the tests developed in the altitude hold mode, are shown. Regarding these indices, the H_∞ controller presents slightly better results compared with the LQI controller.



(a) Position error with LQI controller (b) Position error with H_∞ controller

Figure 5.10: Position errors in altitude hold mode

Table 5.2: Performance indices - Altitude hold mode tests

Controller	Controlled variable	Setting time $[s]$	Overshoot [%]
LQI	z	5.21	8.71
H_∞	z	4.49	5.36

5.3.3 GNSS-Dependent Modes

The tests regarding the GNSS-Dependent flight modes, are executed in the same scenario as the altitude hold mode tests. In this case, the variables of interest for the tests are the x, y and z positions.

The references used in these tests are selected in a way that the same signal excursions as those used in Section 4.3 are described. The performance of the controllers in the GNSS-Dependent modes is evaluated using the position error signal.

LQI Controller

This test was developed on the same football field as the altitude hold mode tests. During the execution of the test, there was no considerable presence of wind.

In Fig. 5.11, the results of the LQI controller test for GNSS-Dependent modes, are shown. As detailed in Section 4.3, the cyclic reference trajectory describes a square of two meters on each side, with constant altitude.

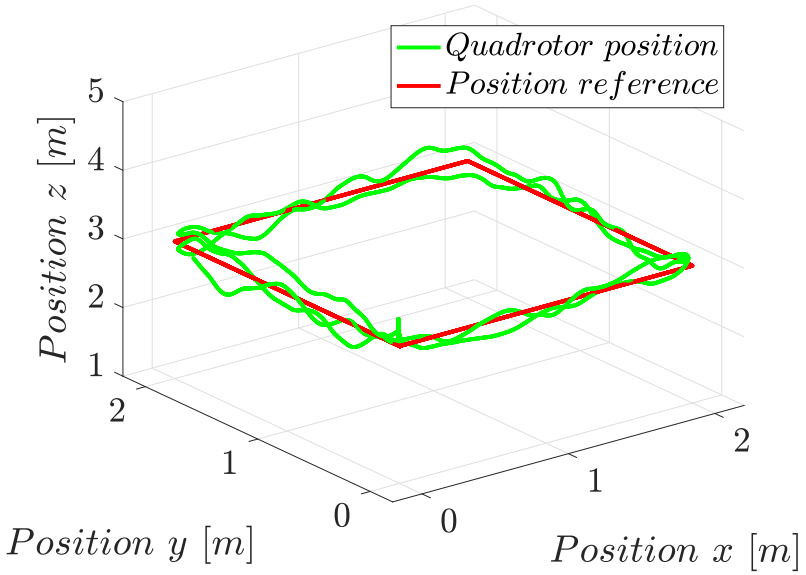


Figure 5.11: Position response of the GNSS-Dependent modes with a LQI controller

H_∞ Controller

The H_∞ controller test, in GNSS-Dependent modes, was developed approximately 60 minutes after the LQI controller test, with similar weather conditions, and the same reference trajectory. The results of these test are shown in Fig. 5.12.

Performance

Both the H_∞ and the LQI controller manage to make the quadrotor successfully follow a three-dimensional trajectory.

To compare the performance of both controllers, the measurement of the Euclidean distance between the position of the quadrotor and the instantaneous reference, is used again. This Euclidean distance defines the error in the control system.

As seen in Fig. 5.13, the error in the LQI controller test is slightly greater than the one in the H_∞ test. The error means, of each controller test, are shown in Table 5.3.

Based on the mean position error in each test, the H_∞ controller has a better performance when compared to the LQI controller in GNSS-Dependent modes.

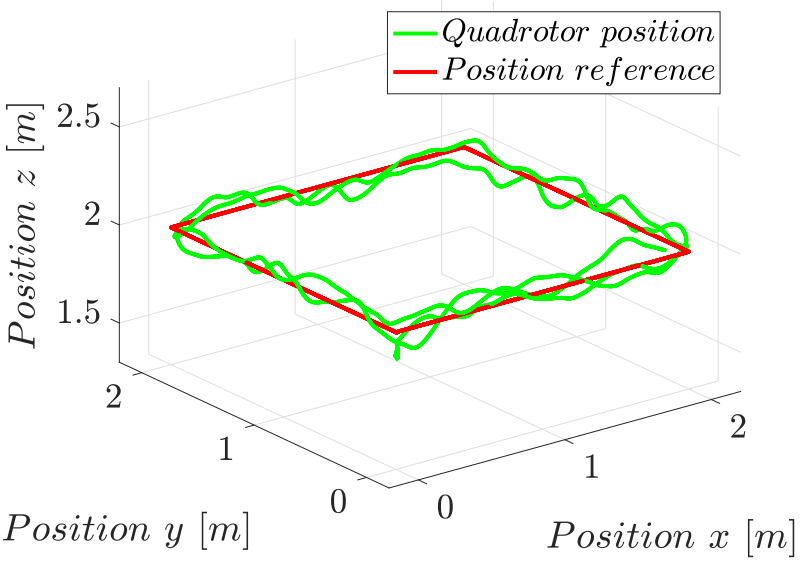
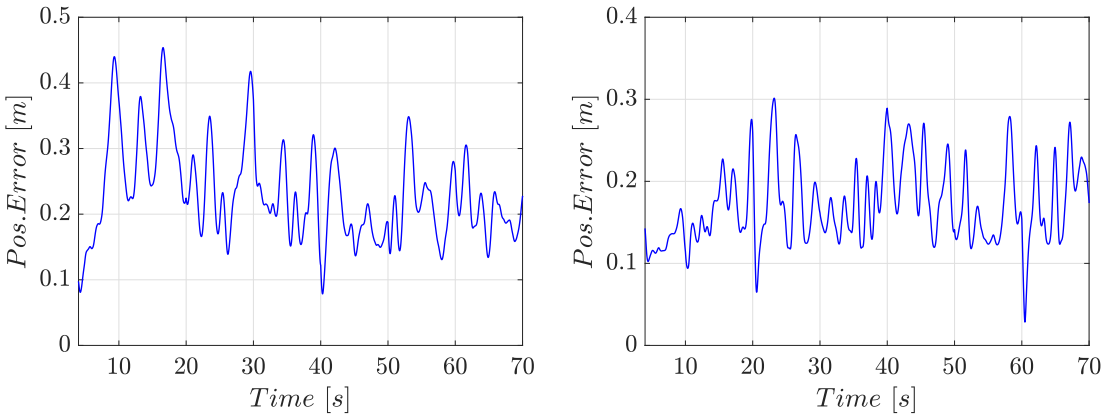


Figure 5.12: Position response of the GNSS-Dependent modes with a H_∞ controller



(a) Position error with LQI controller

(b) Position error with H_∞ controller

Figure 5.13: Position errors in GNSS-Dependent mode

Table 5.3: Error mean - GNSS-Dependent modes tests

Controller	Error mean $[m]$
LQI	0.234
H_∞	0.177

5.4 Conclusions

This chapter presents the implementation of the controllers designed for the quadrotor prototype. The flight controller, which executes the control and estimation algorithms, is implemented as an Android application that is executed in the smartphone on board the quadrotor. The unmanned aircraft system is complemented with a ground control station which is wirelessly connected to the smartphone using a packet-based communication. Using the ground control station, the user is allowed to configure and monitor the status of the quadrotor during the flight, being able to change the flight mode, and the reference signals of the control system. The results of the execution of the two control algorithms, H_∞ and LQI, for each flight mode are exposed. The feedback of the output signals and the states of the system is carried out using the signals estimated by the Kalman filter. The flight mode tests, using both controllers, show that they are able to stabilize the quadrotor and make it follow a given reference, being the H_∞ controller, the one that performs best in terms of error, overshoot and setting time.

Chapter 6

Conclusions and Outlook

6.1 Conclusions

The main objective of this thesis is to design and implement algorithms for control and estimation of flight dynamics executed in a smartphone for the quadrotor of the Industrial Control Research Group of the Universidad del Valle.

In the first stage of this thesis, the motivation and objectives of this project were defined. A literature review was conducted in order to analyse the recent developments regarding the control of quadrotors and the use of smartphones in control systems. Also, the main commercially used quadrotor flight modes were described.

The dynamic model of the quadrotor was derived using the Newton-Euler and Euler-Lagrange approaches. The model inputs are derived depending on the quadrotor geometry configuration, '+' or 'X', which were described. The dynamic model was linearized about an equilibrium state and input, using the Jacobian linearization.

After knowing the dynamic model of the quadrotor, the smartphone-based quadrotor, was detailed. This experimental platform is composed by a smartphone, motors, electronic speed controllers, an Arduino Mega ADK, a battery and a carbon fiber frame. The smartphone is the only computational tool in the quadrotor which executes the control and estimation algorithms that allow the quadrotor to fly. Some additional components were designed and 3D printed in order to support and protect the smartphone and the Arduino Mega ADK. Additionally, the quadrotor parameters, including mass, moments of inertia, and motors dynamics, were established experimentally.

Two types of controllers were designed aiming to have reference tracking capabilities in the smartphone-based quadrotor: a linear quadratic regulator with integral feed-

back (also known as linear quadratic integral - LQI), and a H_∞ controller. In order to have available estimations of all the components of the quadrotor state vector, a Kalman filter was designed. This Kalman filter is restricted to operate near the quadrotor equilibrium point, as it is based on the linearised quadrotor model. The simulation of these controllers, executed with the non-linear model of the quadrotor, showed the feasibility of its subsequent implementation in the real prototype.

Finally, the implementation phase included the development of an Android application that executes all the flight control system and is executed in the smartphone on board the quadrotor. Also, a desktop application for monitoring and configuring the quadrotor, was developed. This desktop application is the software component of the ground control station. Between the ground control station and the smartphone-based quadrotor, a packet-based wireless communication based on the TCP/IP protocols was established. The behaviour of the smartphone-based quadrotor being controlled by the designed control strategies is evaluated through flight tests. In these tests, both the H_∞ and LQI controllers showed the ability to keep the quadrotor stabilised and track a reference. The H_∞ controller showed a slightly better performance than the LQI controllers in terms of error, overshoot and setting time.

6.2 Outlook

The work done within the framework of this project, shows that it is possible for a smartphone to execute static and dynamic controllers, as well as state estimation algorithms. Therefore, the smartphone is able to serve as flight controller for a quadrotor and numerous options for future work can be discussed.

There are several aspects which can be improved to obtain better results. The state estimation, for instance, is a fundamental pillar in the proper performance of the implemented control system. This estimation can be improved using an Extended Kalman filter (EKF), which is a Kalman filter based on the non-linear dynamics of the system. Another approach for state estimation in aircraft systems is the GNSS/INS integration, were a Kalman filter based on the error dynamics is used. Taking advantage of the availability of the camera present in the smartphone, it is also possible to add estimation strategies that include visual odometry as a data source.

The control algorithms can be improved aiming to use robust controllers or non-linear controllers such as the linear parameter-varying control strategies. Also, the addition of the neglected quadrotor dynamics, as the propellers drag force and the ground effect, would improve the performance of the control system.

With the work developed in this thesis, it will be possible to implement a swarm of quadrotors based on smartphones, which could be used for research about multi-agent systems.

Appendix A

Publications

A. Astudillo, P. Muñoz, F. Alvarez and E. Rosero, "Altitude and attitude cascade controller for a smartphone-based quadcopter," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017, pp. 1447–1454. [Online]. Available: http://ieeexplore.ieee.org/document/7991400/

A. Astudillo, B. Bacca and E. Rosero, "Optimal and robust controllers design for a smartphone-based quadrotor," in *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)*

(Paper Submitted to Journal) A. Astudillo, P. Muñoz and E. Rosero, "Cascade Controller for Autonomous Flight of a Smartphone-based Quadrotor," in *Journal of Intelligent & Robotic Systems, SI: UAS-2017*.

Appendix B

Supplementary Material

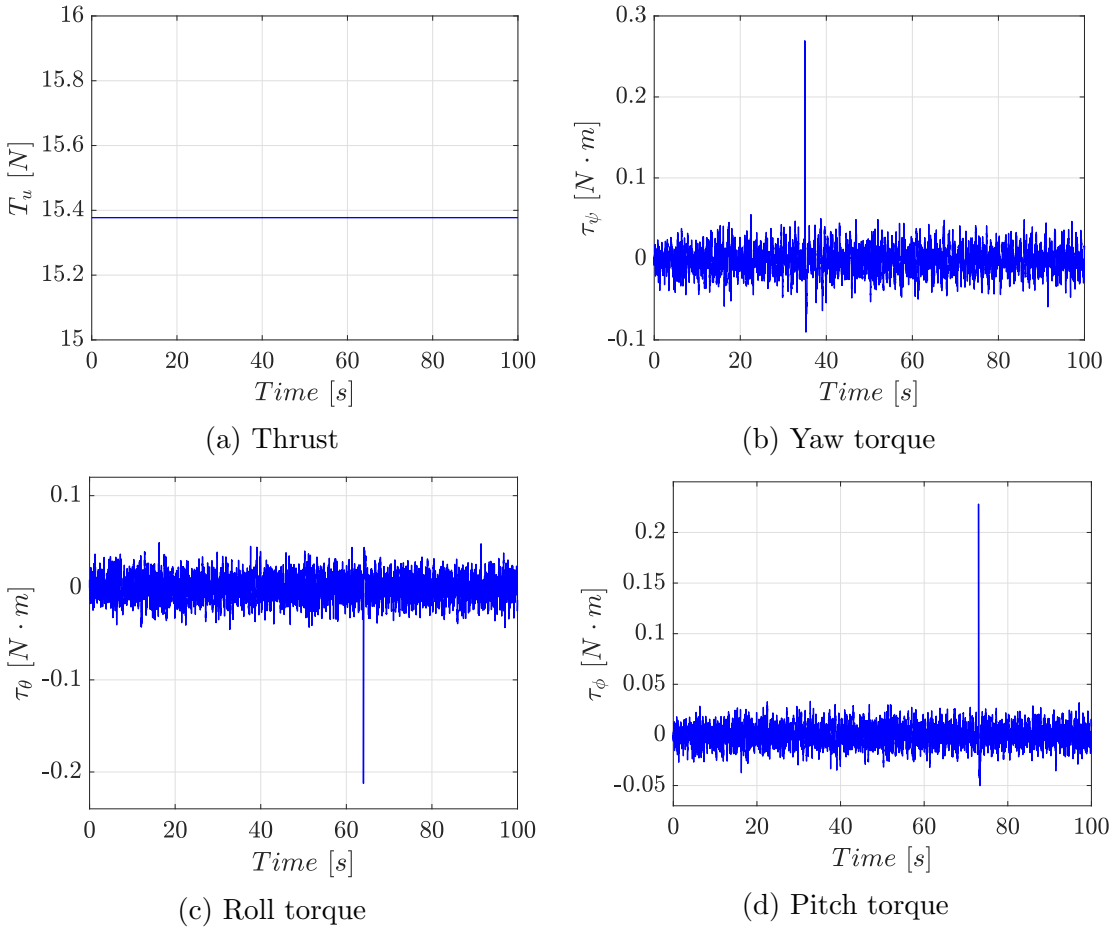
All supplementary material including the Android app code, GCS code, Arduino code, 3D objects, etc., is hosted in: https://goo.gl/U43bB6

Appendix C

Control Signals in Flight Tests

Stabilize Mode

LQI Controller



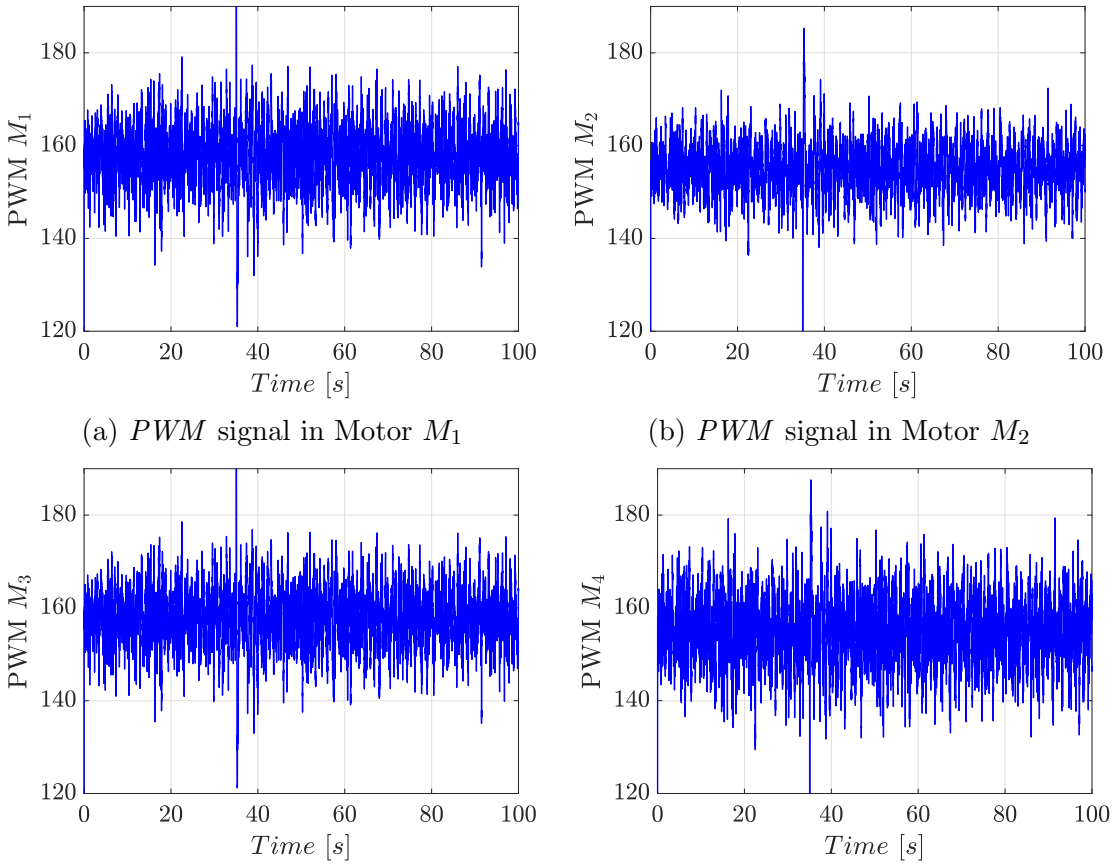
(a) Thrust

(b) Yaw torque

(c) Roll torque

(d) Pitch torque

Figure C.1: Control signals in LQI control test for stabilize mode



(a) PWM signal in Motor M_1

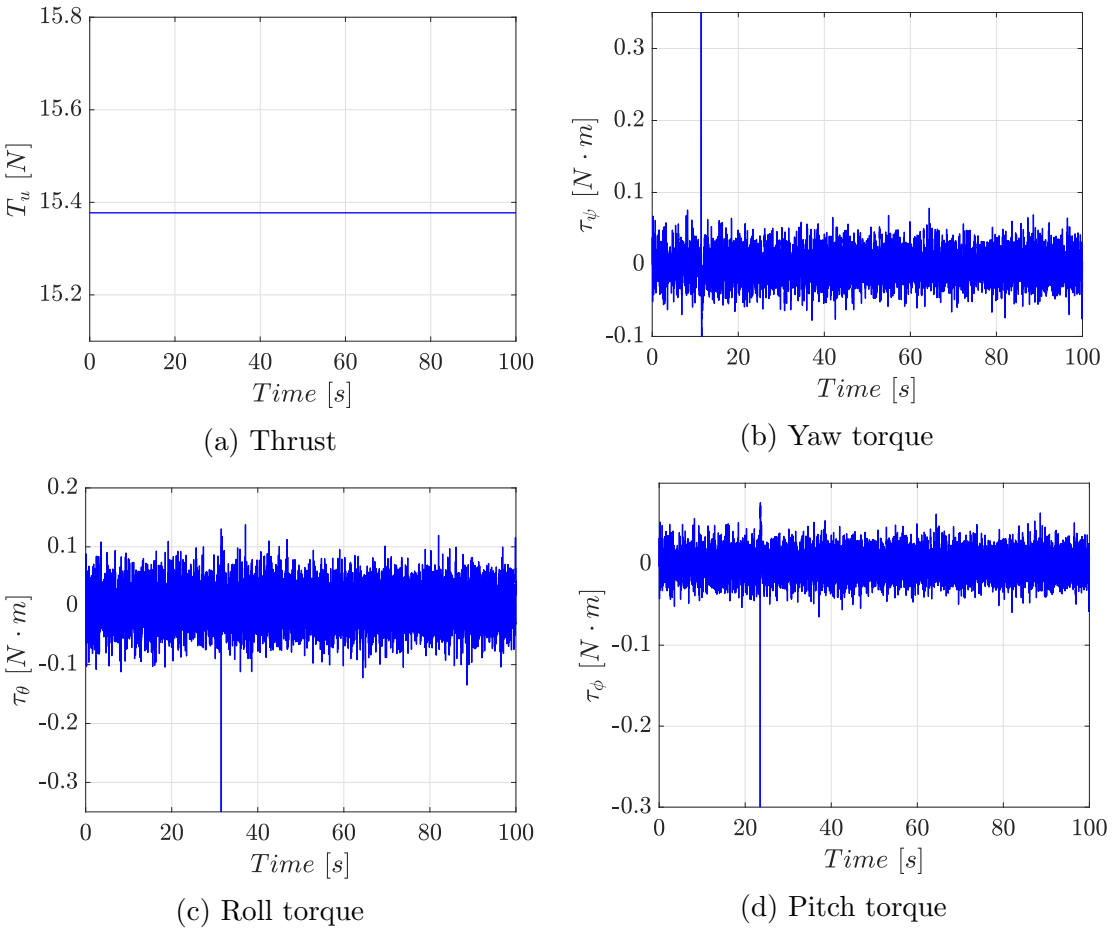
(b) PWM signal in Motor M_2

(c) PWM signal in Motor M_3

(d) PWM signal in Motor M_4

Figure C.2: PWM signals in LQI control test for stabilize mode

H_∞ Controller



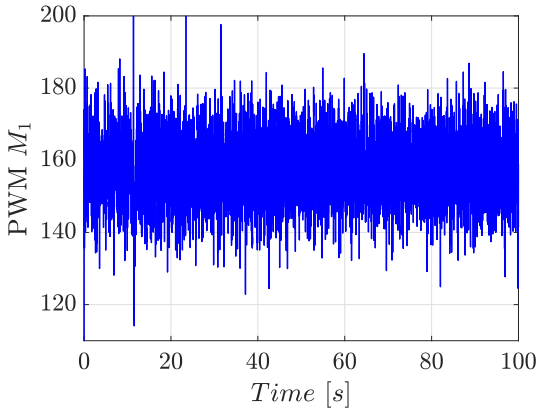
(a) Thrust

(b) Yaw torque

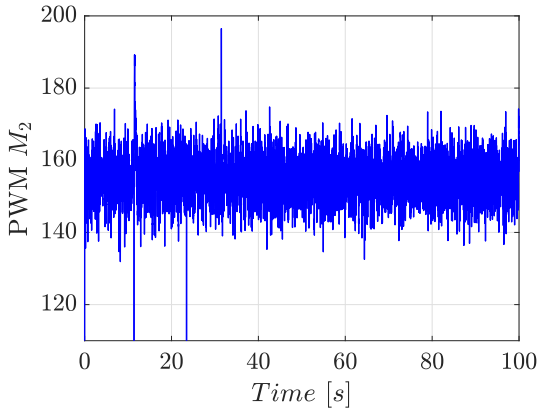
(c) Roll torque

(d) Pitch torque

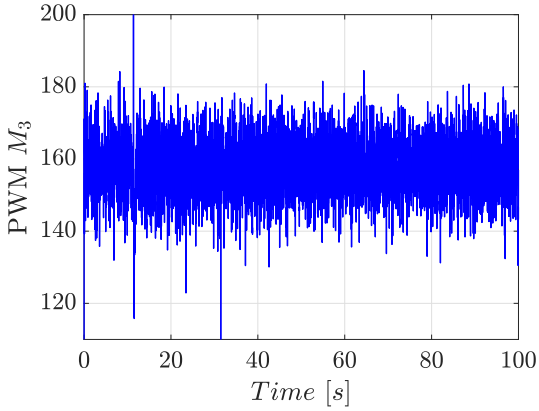
Figure C.3: Control signals in H_∞ control test for stabilize mode



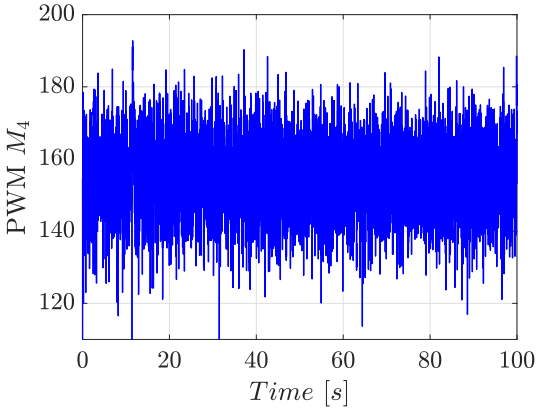
(a) PWM signal in Motor M_1



(b) PWM signal in Motor M_2



(c) PWM signal in Motor M_3

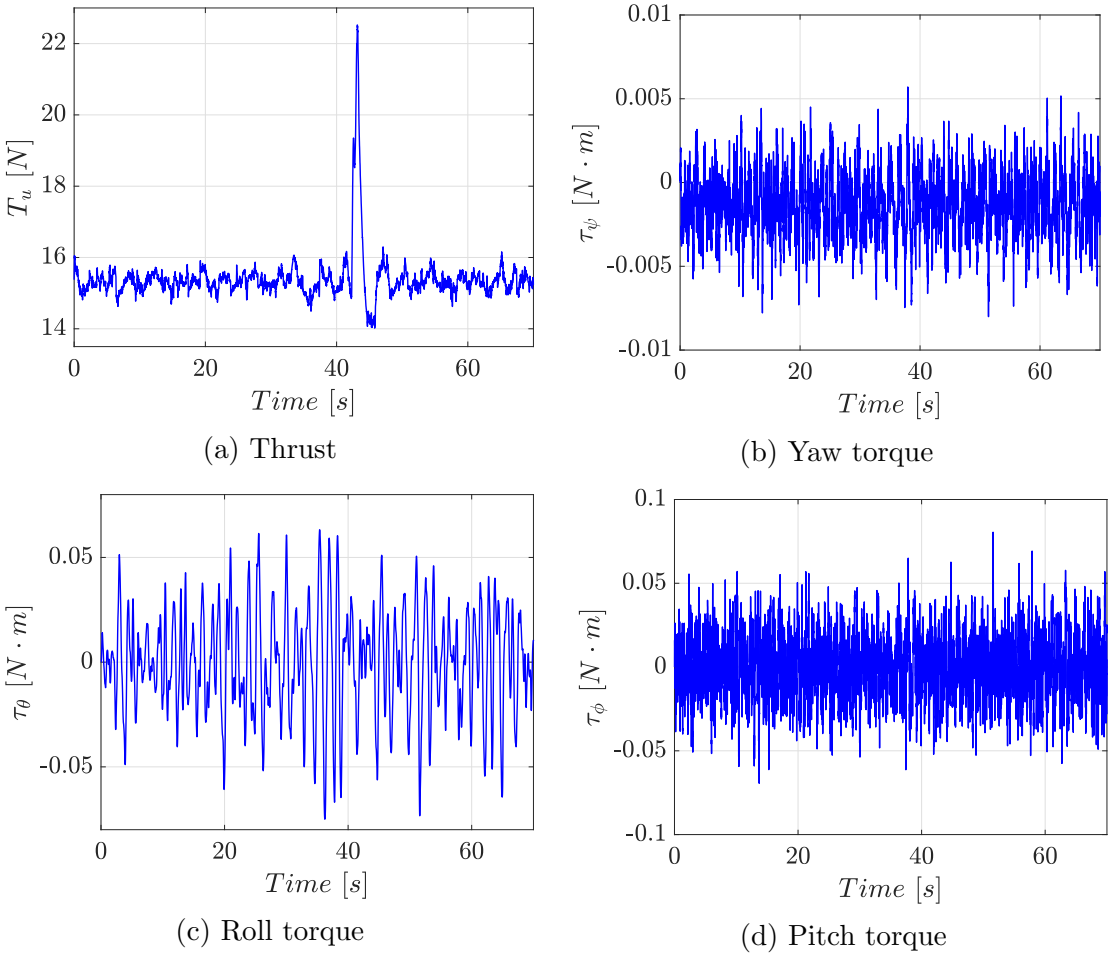


(d) PWM signal in Motor M_4

Figure C.4: PWM signals in H_∞ control test for stabilize mode

Altitude Hold Mode

LQI Controller



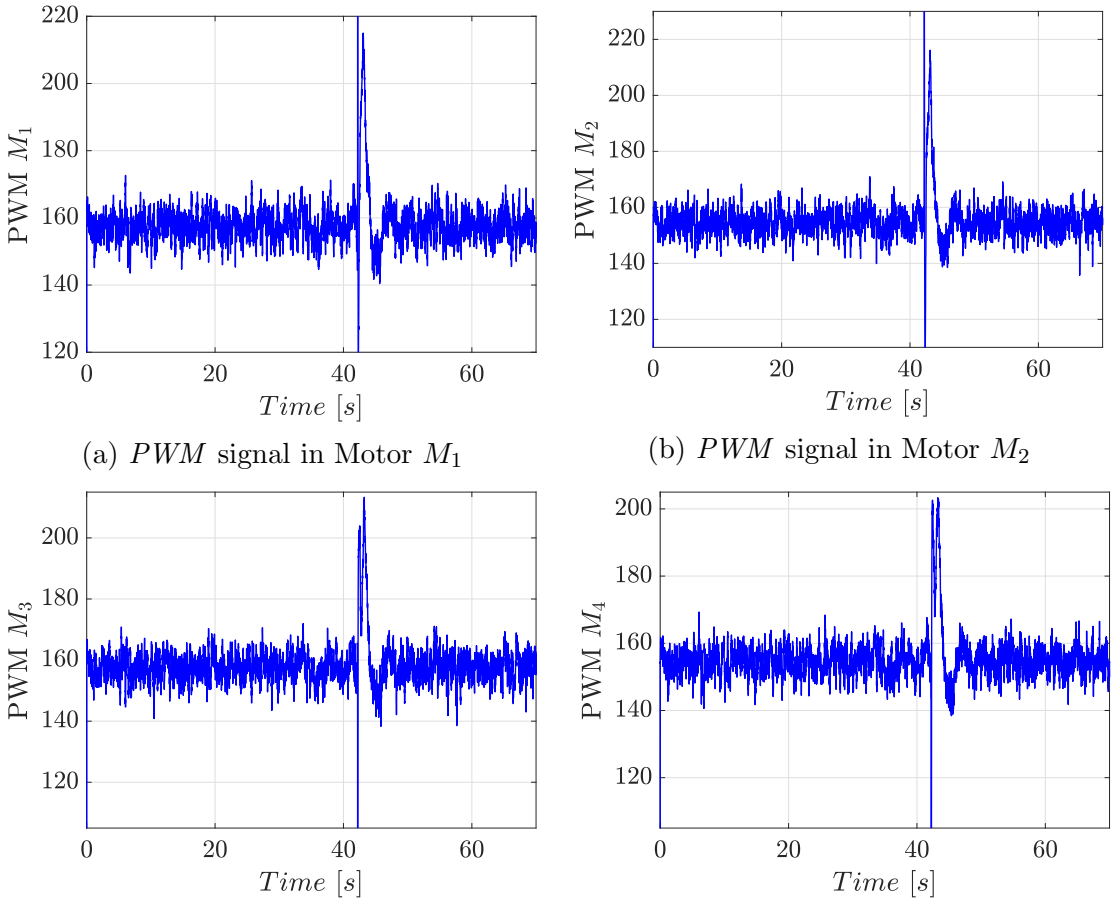
(a) Thrust

(b) Yaw torque

(c) Roll torque

(d) Pitch torque

Figure C.5: Control signals in LQI control test for altitude hold mode



(a) *PWM* signal in Motor M_1

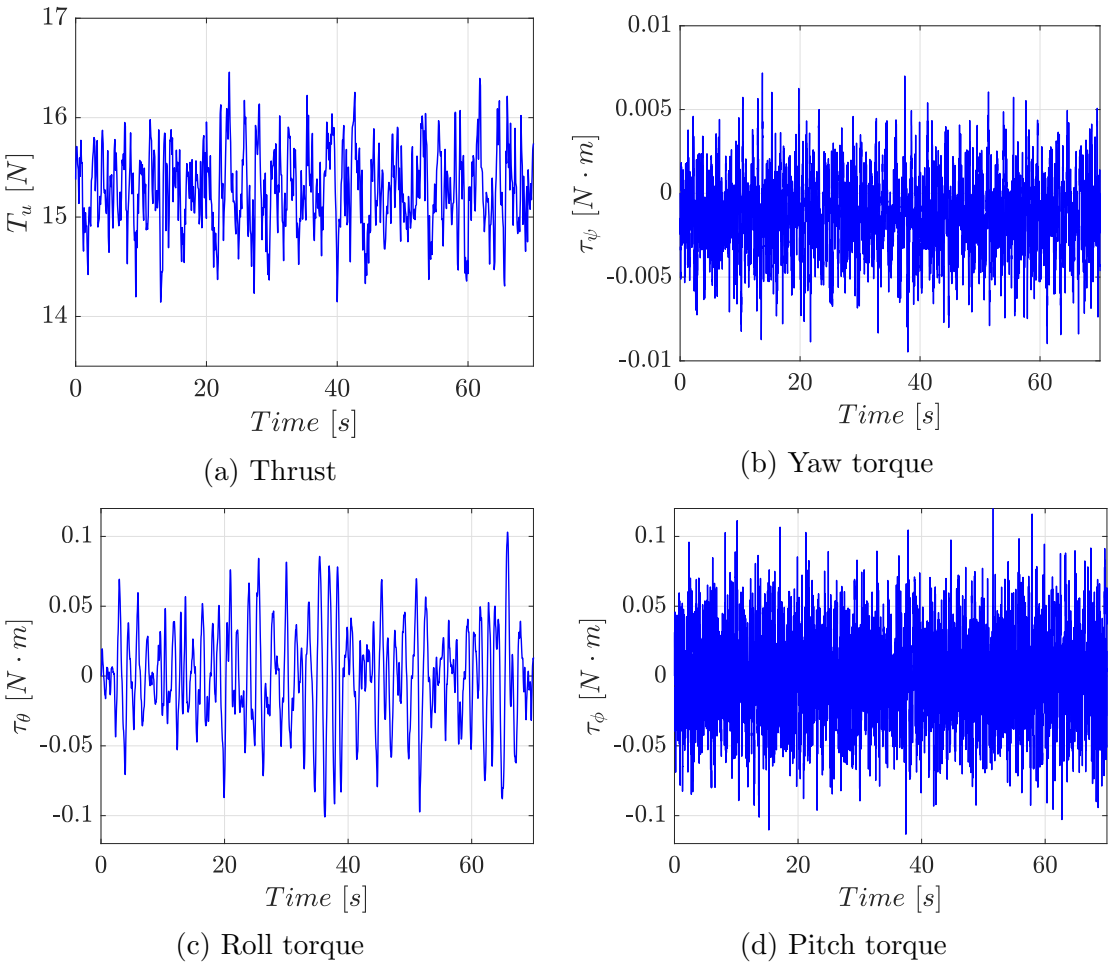
(b) *PWM* signal in Motor M_2

(c) *PWM* signal in Motor M_3

(d) *PWM* signal in Motor M_4

Figure C.6: *PWM* signals in LQI control test for altitude hold mode

H_∞ Controller



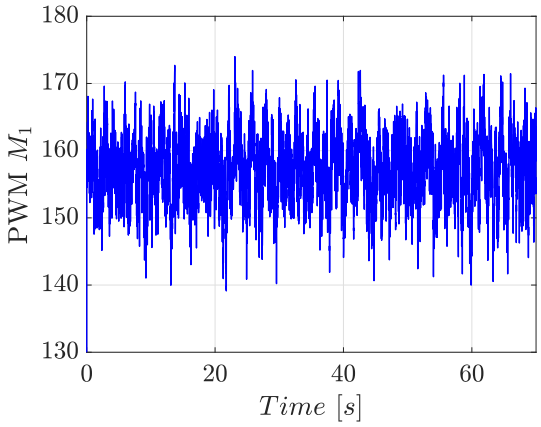
(a) Thrust

(b) Yaw torque

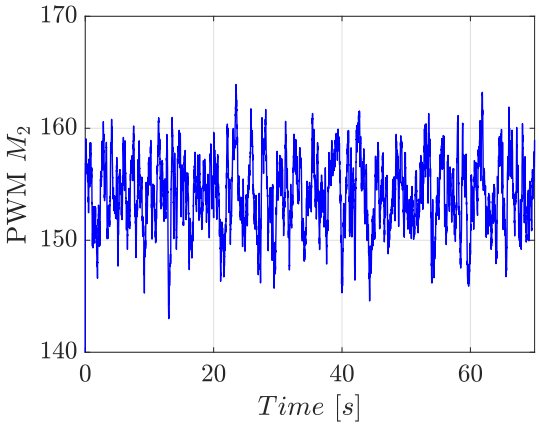
(c) Roll torque

(d) Pitch torque

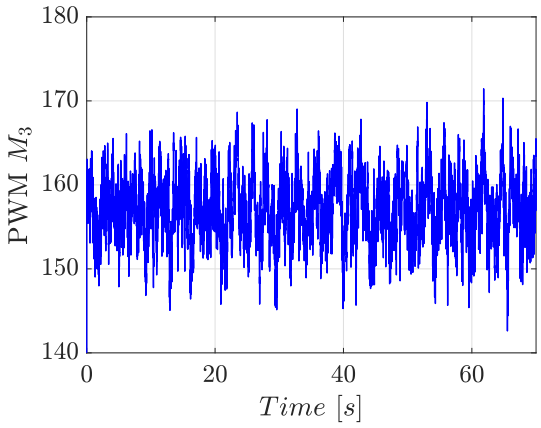
Figure C.7: Control signals in H_∞ control test for altitude hold mode



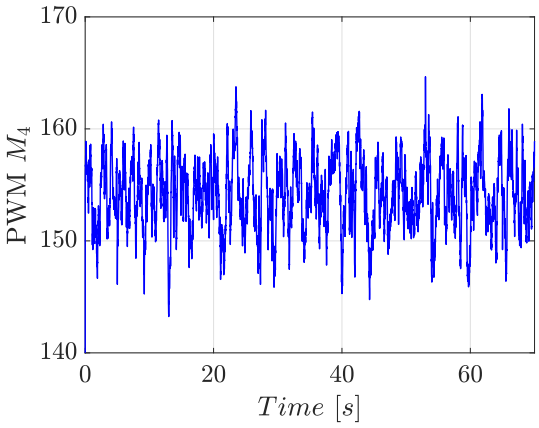
(a) PWM signal in Motor M_1



(b) PWM signal in Motor M_2



(c) PWM signal in Motor M_3

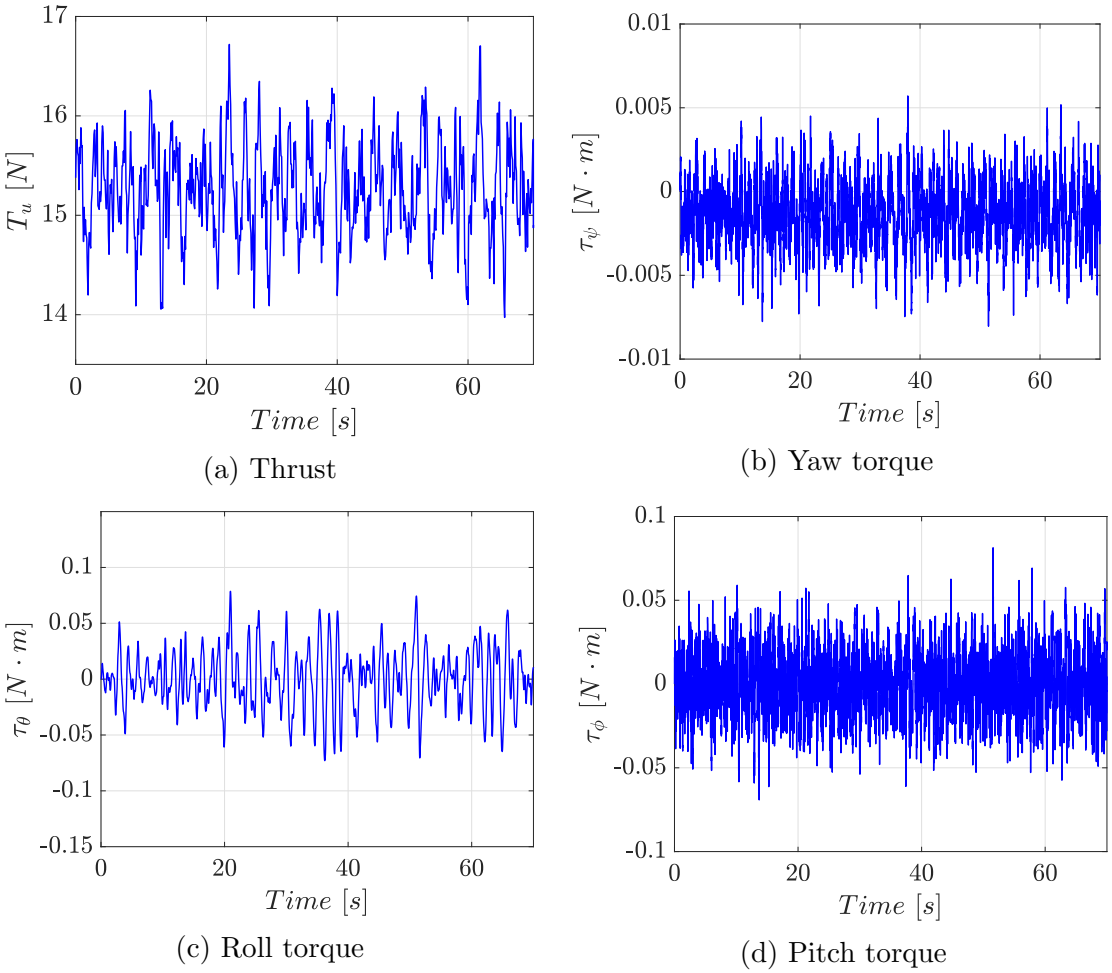


(d) PWM signal in Motor M_4

Figure C.8: PWM signals in H_∞ control test for altitude hold mode

GNSS-Dependent Mode

LQI Controller



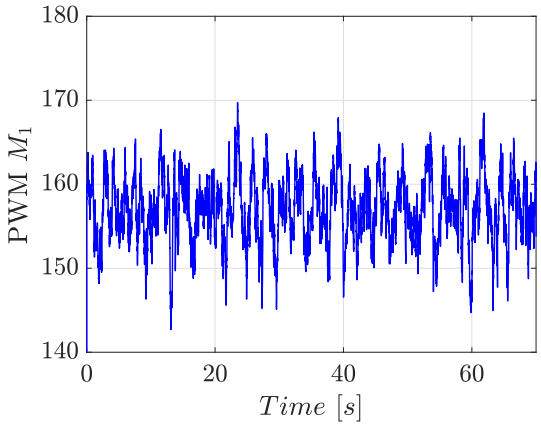
(a) Thrust

(b) Yaw torque

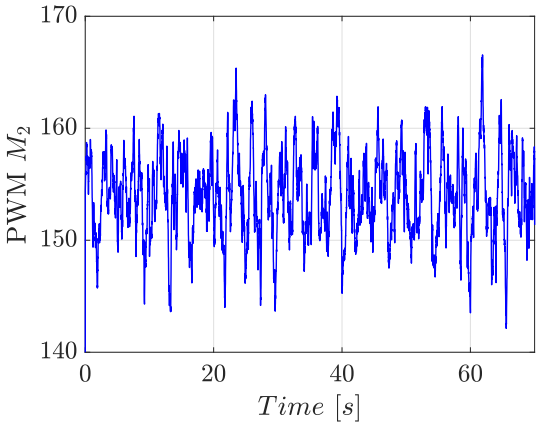
(c) Roll torque

(d) Pitch torque

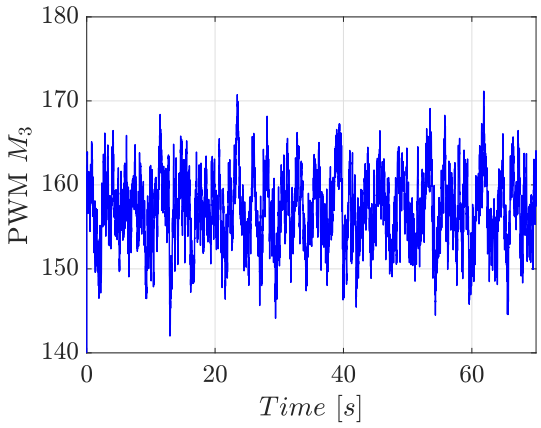
Figure C.9: Control signals in LQI control test for GNSS-Dependent modes



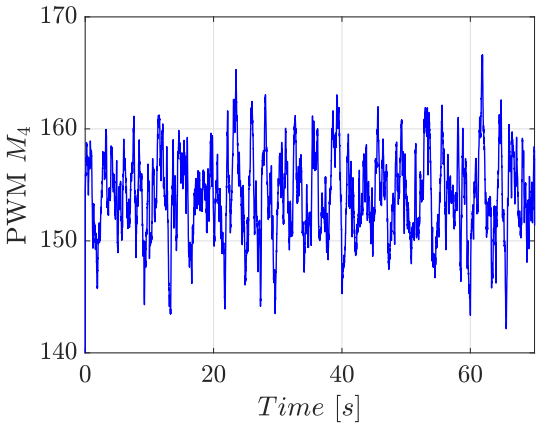
(a) PWM signal in Motor M_1



(b) PWM signal in Motor M_2



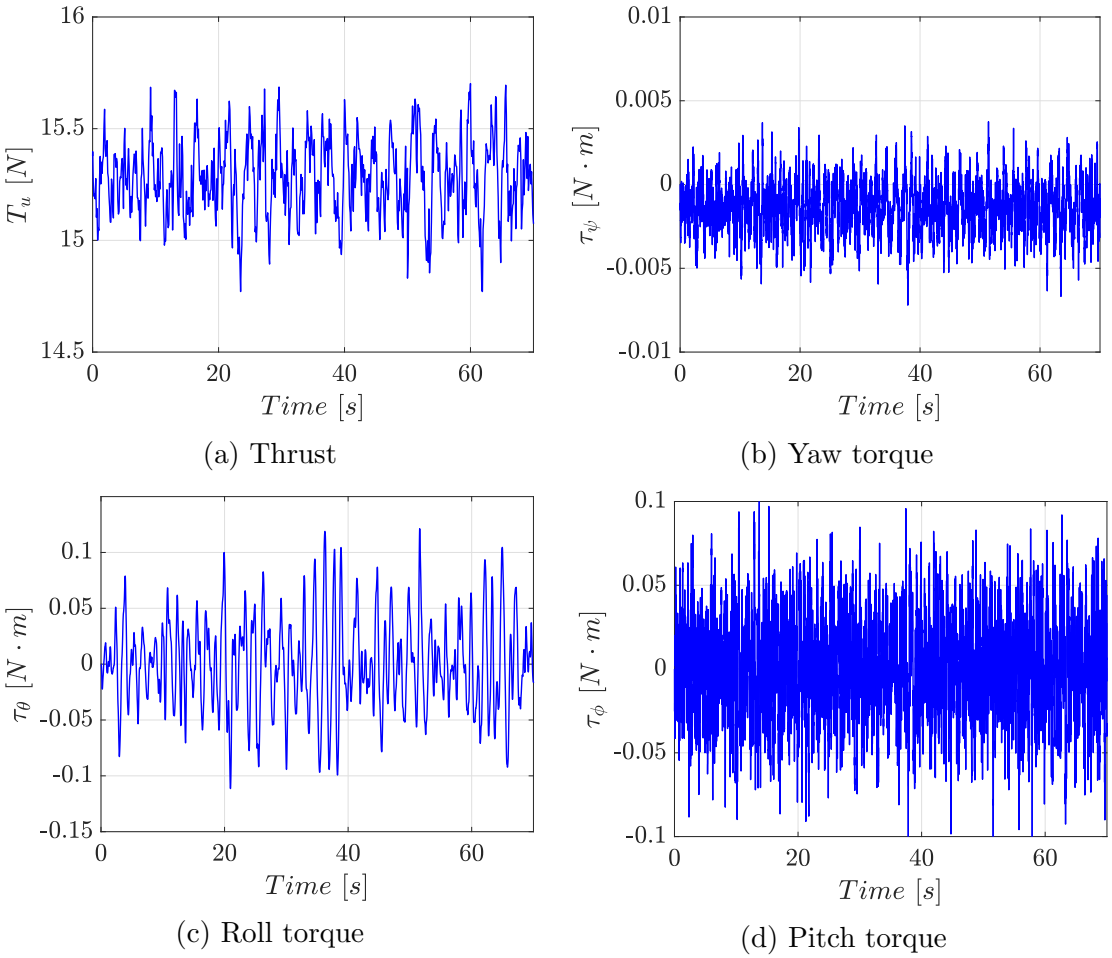
(c) PWM signal in Motor M_3



(d) PWM signal in Motor M_4

Figure C.10: PWM signals in LQI control test for GNSS-Dependent modes

H_∞ Controller



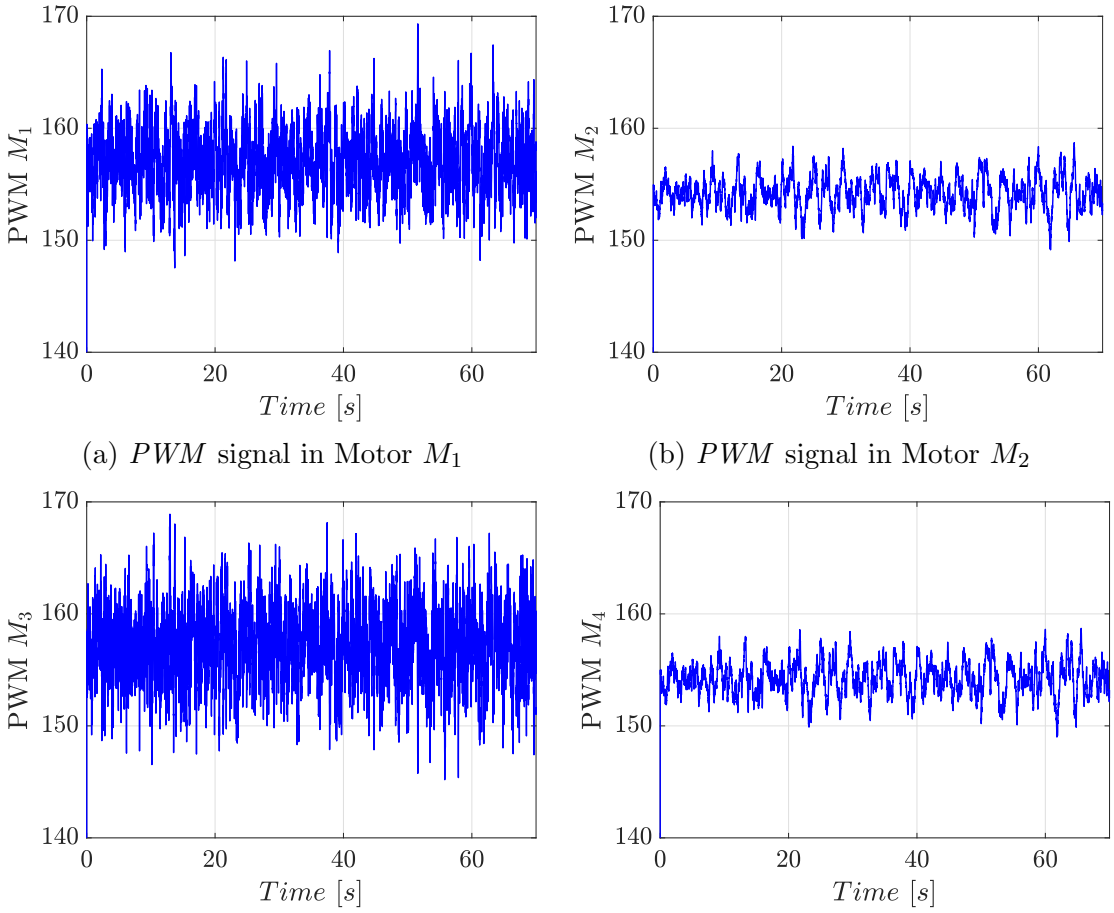
(a) Thrust

(b) Yaw torque

(c) Roll torque

(d) Pitch torque

Figure C.11: Control signals in H_∞ control test for GNSS-Dependent modes



(a) PWM signal in Motor M_1

(b) PWM signal in Motor M_2

(c) PWM signal in Motor M_3

(d) PWM signal in Motor M_4

Figure C.12: PWM signals in H_∞ control test for GNSS-Dependent modes

Bibliography

[1] H. Liu and X. Wang, "Quaternion-based robust attitude control for quadrotors," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 920–925.

[2] R. Lopez, I. Gonzalez-Hernandez, S. Salazar, A. E. Rodriguez, J. J. Ordaz, and A. Osorio, "Disturbance rejection for a Quadrotor aircraft through a robust control," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 409–415.

[3] J. Jung, Y. Jung, D. You, and D. H. Shim, "A flight control system design for highly unstable unmanned combat aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2014, pp. 1117–1125.

[4] S. Kohno and K. Uchiyama, "Design of robust controller of fixed-wing UAV for transition flight," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2014, pp. 1111–1116.

[5] B. Shang, J. Liu, T. Zhao, and Y. Chen, "Fractional order robust visual servoing control of a quadrotor UAV with larger sampling period," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, vol. 1, no. 209. IEEE, jun 2016, pp. 1228–1234.

[6] S. Salazar, I. Gonzalez-Hernandez, R. Lopez, and R. Lozano, "Simulation and robust trajectory-tracking for a Quadrotor UAV," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, may 2014, pp. 1167–1174.

[7] S. Bouabdallah, "Design and Control of Quadrotors With Application To Autonomous Flying," Ph.D. dissertation, École Polytechnique Fédérale de Lausanne, 2007.

[8] M. Rahman, A. Ani, S. Yahaya, Z. Hussain, R. Boudville, and A. Ahmad, "Implementation of quadcopter as a teaching tool to enhance engineering courses," in *2016 IEEE 8th International Conference on Engineering Education (ICEED)*. IEEE, dec 2016, pp. 32–37.

[9] C. E. García and F. A. Herrera, "Percepción remota en cultivos de caña de azúcar usando una cámara multiespectral en vehículos aéreos no tripulados," *Anais XVII Simpósio Brasileiro de Sensoriamento Remoto - SBSR*, pp. 4450 – 4457, 2015.

[10] S. S. Kumar and A. S. Sundar, "The Rescue Mission with SQUADCOPTER by Real time GPS 3D surveillance," *Journal of Basic and Applied Engineering Research*, vol. 1, no. 5, pp. pp. 60–63, 2014.

[11] V. Gatteschi, F. Lamberti, G. Paravati, A. Sanna, C. Demartini, A. Lisanti, and G. Venezia, "New Frontiers of Delivery Services Using Drones: A Prototype System Exploiting a Quadcopter for Autonomous Drug Shipments," in *2015 IEEE 39th Annual Computer Software and Applications Conference*, vol. 2. IEEE, jul 2015, pp. 920–927.

[12] L. F. Gonzalez, M. P. G. Castro, and F. Tamagnone, "Multidisciplinary Design and Flight Testing of a Remote Gas/Particle Airborne Sensor System," *28th International Congress of the Aeronautical Sciences*, pp. 1–13, 2012.

[13] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "PID, LQR and LQR-PID on a quadcopter platform," in *2013 International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, may 2013, pp. 1–6.

[14] M. F. Silva, A. C. Ribeiro, M. F. Santos, M. J. Carmo, L. M. Honório, E. J. Oliveira, and V. F. Vidal, "Design of angular PID controllers for quadcopters built with low cost equipment," *2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings*, pp. 216–221, 2016.

[15] C. Liu, J. Pan, and Y. Chang, "PID and LQR trajectory tracking control for an unmanned quadrotor helicopter: Experimental studies," in *2016 35th Chinese Control Conference (CCC)*. IEEE, jul 2016, pp. 10 845–10 850.

[16] E. Reyes-Valeria, R. Enriquez-Caldera, S. Camacho-Lara, and J. Guichard, "LQR control for a quadrotor using unit quaternions: Modeling and simulation," in *CONIELECOMP 2013, 23rd International Conference on Electronics, Communications and Computing*. IEEE, mar 2013, pp. 172–178.

[17] Y. Dong, J. Fu, B. Yu, Y. Zhang, and J. Ai, "Position and heading angle control of an unmanned quadrotor helicopter using LQR method," *Chinese Control Conference, CCC*, vol. 2015-Septe, pp. 5566–5571, 2015.

[18] B. Fan, J. Sun, and Y. Yu, "A LQR controller for a quadrotor: Design and experiment," *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*, pp. 81–86, 2017.

[19] K. Zhang, J. Chen, Y. Chang, and Y. Shi, "EKF-based LQR tracking control of a quadrotor helicopter subject to uncertainties," *IECON Proceedings (Industrial Electronics Conference)*, pp. 5426–5431, 2016.

[20] G. Ganga and M. M. Dharmana, "MPC controller for trajectory tracking control of quadcopter," in *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT)*. IEEE, apr 2017, pp. 1–6.

[21] A. Castillo, R. Sanz, P. García, and P. Albertos, "A quaternion-based and active disturbance rejection attitude control for quadrotor," *2016 IEEE International Conference on Information and Automation, IEEE ICIA 2016*, no. August, pp. 240–245, 2016.

[22] Y. Ameho, F. Niel, F. Defay, J.-M. Biannic, and C. Berard, "Adaptive control for quadrotors," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, may 2013, pp. 5396–5401.

[23] H. Gao, C. Liu, D. Guo, and J. Liu, "Fuzzy adaptive PD control for quadrotor helicopter," in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, no. JUNE. IEEE, jun 2015, pp. 281–286.

[24] B. Wang, L. Mu, and Y. Zhang, "Adaptive robust control of quadrotor helicopter towards payload transportation applications," in *2017 36th Chinese Control Conference (CCC)*. IEEE, jul 2017, pp. 4774–4779.

[25] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, "Robust adaptive control design for quadcopter payload add and drop applications," *Chinese Control Conference, CCC*, vol. 2015-Septe, pp. 3252–3257, 2015.

[26] B. Wang, L. Mu, and Y. Zhang, "Adaptive robust tracking control of quadrotor helicopter with parametric uncertainty and external disturbance," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017, pp. 402–407.

[27] A. Prayitno, V. Indrawati, and C. Arron, "H-Infinity Control for Pitch-Roll AR.Drone," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 3, p. 963, 2016.

[28] J. P. Ortiz, L. I. Minchala, and M. J. Reinoso, "Nonlinear Robust H-Infinity PID Controller for the Multivariable System Quadrotor," *IEEE Latin America Transactions*, vol. 14, no. 3, pp. 1176–1183, mar 2016.

[29] P. Muñoz, "Control de Elevación Para Cuadricóptero Usando Un Celular Inteligente," Bachelor Thesis, Universidad del Valle, Cali, 2017.

[30] M. Moghadam and F. Caliskan, "Actuator and Sensor Fault Detection and Diagnosis of Quadrotor Based on Two-Stage Kalman Filter," *2015 5th Australian Control Conference (AUCC)*, vol. 4, no. 1, pp. 2–7, 2015.

[31] F. A. Goodarzi and T. Lee, "Extended Kalman filter on SE(3) for geometric control of a quadrotor UAV," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, no. 3. IEEE, jun 2016, pp. 1371–1380.

[32] K.-H. Oh and H.-S. Ahn, "Extended Kalman filter with multi-frequency reference data for quadrotor navigation," in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, no. Iccas. IEEE, oct 2015, pp. 201–206.

[33] K. D. Sebesta and N. Boizot, "A real-time adaptive high-gain EKF, applied to a quadcopter inertial navigation system," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 495–503, 2014.

[34] J. Goslinski, W. Giernacki, and S. Gardecki, "Unscented Kalman Filter for an orientation module of a quadrotor mathematical model," in *2013 9th Asian Control Conference (ASCC)*. IEEE, jun 2013, pp. 1–6.

[35] Y. Al Younes, H. Noura, M. Muflehi, A. Rabhi, and A. El Hajjaji, "Model-free observer for state estimation applied to a quadrotor," *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pp. 1378–1384, 2015.

[36] N. Xie, X. Lin, and Y. Yu, "Position estimation and control for quadrotor using optical flow and GPS sensors," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, nov 2016, pp. 181–186.

[37] Q. Yu, F. He, Y. Zhang, and J. Ma, "Stereo-vision based obstacle motion information acquisition algorithm for quadrotors," in *2017 36th Chinese Control Conference (CCC)*. IEEE, jul 2017, pp. 5590–5595.

[38] Y. Liu and M. Z. Q. Chen, "A novel three-axis visual attitude estimation algorithm with application to quadrotors," in *2017 29th Chinese Control And Decision Conference (CCDC)*. IEEE, may 2017, pp. 5436–5441.

[39] ArduPilot Dev Team, "Copter - Flight Modes," 2016. [Online]. Available: http://ardupilot.org/copter/docs/flight-modes.html

[40] N. Oros and J. L. Krichmar, "Smartphone Based Robotics: Powerful, Flexible and Inexpensive Robots for Hobbyists, Educators, Students and Researchers," 2013.

[41] J. P. de A. Barbosa, F. do P. de C. Lima, L. dos S. Coutinho, J. P. R. Rodrigues Leite, J. Barbosa Machado, C. Henrique Valerio, and G. Sousa Bastos, "ROS,

Android and cloud robotics: How to make a powerful low cost robot," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, jul 2015, pp. 158–163.

[42] M. A. Gunawan, F. Sulaiman, T. A. Putra, C. Daniel Hasudungan, and R. F. Sari, "Object-following robot using adaptive cruise control algorithm with IOIO," in *2014 International Conference on Intelligent Green Building and Smart Grid (IGBSG)*. IEEE, apr 2014, pp. 1–5.

[43] T. Tetzlaff, R. Zandian, L. Drüppel, and U. Witkowski, *Robot Intelligence Technology and Applications 2012*, ser. Advances in Intelligent Systems and Computing, J.-H. Kim, E. T. Matson, H. Myung, and P. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 208.

[44] A. Drumea, "Control of Industrial Systems Using Android-Based Devices," *36th Int. Spring Seminar on Electronics Technology*, pp. 405–408, 2013.

[45] K.-w. Lin, Z.-h. Wu, and Y.-l. Lin, "Applications of Temperature Control Based on Android Platform," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. II, 2014.

[46] N.-V. Truong and D.-L. Vu, "Remote monitoring and control of industrial process via wireless network and Android platform," *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, no. 1, pp. 340–343, 2012.

[47] Z. Lu, F. Nagata, K. Watanabe, and M. K. Habib, "iOS application for quadrotor remote control," *Artificial Life and Robotics*, vol. 22, no. 3, pp. 374–379, sep 2017.

[48] L. F. Aristizabal, D. F. Almario, and J. A. Lopez, "Development of an Android App as a learning tool of dynamic systems and automatic control," in *2014 III International Congress of Engineering Mechatronics and Automation (CIIMA)*. IEEE, oct 2014, pp. 1–5.

[49] S. Wu Wu, "Sintonización de controladores PI/PID mediante el desarrollo de una aplicación en Android," Ph.D. dissertation, Universidad de Costa Rica, 2013.

[50] J. García Téllez and D. F. Ochoa Calambás, "Desarrollo de una plataforma de monitorización, control y comunicación con teléfonos inteligentes, para laboratorios portátiles de soporte al área de señales y sistemas," Bachelor Thesis, Universidad del Valle, Cali, 2015.

[51] V. A. Isuru, M. A. I. M. Dharmadasa, D. V. B. C. Jayasinghe, and C. I. Keppitiyagama, "Reusing discarded-smartphone capabilities on quadcopters:

The rationale, benefits and issues," in *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, sep 2016, pp. 229–236.

[52] C. Pearce, M. Guckenberg, B. Holden, A. Leach, R. Hughes, C. Xie, M. Hassett, A. Adderley, L. E. Barnes, M. Sherriff, and G. C. Lewin, "Designing a spatially aware, automated quadcopter using an Android control system," in *2014 Systems and Information Engineering Design Symposium (SIEDS)*, vol. 00, no. c. IEEE, apr 2014, pp. 23–28.

[53] A. Bjälemark and H. Bergkvist, "Quadcopter control using Android based sensing," *Advances in Electrical and Computer Engineering*, pp. 15–21, 2014.

[54] G. Loianno, G. Cross, C. Qu, Y. Mulgaonkar, J. A. Hesch, and V. Kumar, "Flying Smartphones: Automated Flight Enabled by Consumer Electronics," *IEEE Robotics & Automation Magazine*, vol. 22, no. 2, pp. 24–32, jun 2015.

[55] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar, "Smartphones power flying robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2015, pp. 1256–1263.

[56] M. A. Alsharif and M. S. Holzel, "Estimation of a drone's rotational dynamics with piloted Android flight data," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, no. Cdc. IEEE, dec 2016, pp. 1199–1204.

[57] M. A. Alsharif, Y. E. Arslantas, and M. S. Holzel, "Advanced PID attitude control of a quadcopter using asynchronous android flight data," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017, pp. 1602–1607.

[58] ——, "A comparison between advanced model-free PID and model-based LQI attitude control of a quadcopter using asynchronous android flight data," in *2017 25th Mediterranean Conference on Control and Automation (MED)*. IEEE, jul 2017, pp. 1023–1028.

[59] L. Aldrovandi, M. Hayajneh, M. Melega, M. Furci, R. Naldi, and L. Marconi, "A smartphone based quadrotor: Attitude and position estimation," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 1251–1259.

[60] P. Bryant, G. Gradwell, and D. Claveau, "Autonomous UAS controlled by onboard smartphone," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2015, pp. 451–454.

[61] T. Bresciani, "Modelling, Identification and Control of a Quadrotor Helicopter," Master's thesis, Lund University, 2008.

[62] M. Faessler, D. Falanga, and D. Scaramuzza, "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight," *IEEE Robotics and Automation Letters*, vol. PP, no. 99, pp. 1–7, 2016.

[63] M. Emam and A. Fakharian, "Attitude tracking of quadrotor UAV via mixed H2/H∞ controller: An LMI based approach," in *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE, jun 2016, pp. 390–395.

[64] S. Badr, O. Mehrez, and A. E. Kabeel, "A novel modification for a quadrotor design," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2016, pp. 702–710.

[65] F. Sabatino, *Quadrotor control : modeling, nonlinear control design, and simulation*. Stockholm, Sweden: KTH Royal Institute of Technology in Stockholm, 2015, no. June.

[66] R. Vepa, *Nonlinear Control of Robots and Unmanned Aerial Vehicles: An Integrated Approach*, 1st ed. London: CRC Press, 2016.

[67] K. U. Lee, Y. H. Yun, W. Chang, J. B. Park, and Y. H. Choi, "Modeling and Altitude Control of Quad-rotor UAV," in *Proceedings of International Conference on Control, Automation and Systems*, 2011, pp. 1897–1902.

[68] M. A. Khodja, M. Tadjine, M. S. Boucherit, and M. Benzaoui, "Experimental dynamics identification and control of a quadcopter," in *2017 6th International Conference on Systems and Control (ICSC)*. IEEE, may 2017, pp. 498–502.

[69] T. Jiřinec, *Stabilization and control of unmanned quadcopter*. Prague, Czech Republic: Czech Technical University in Prague, 2011.

[70] G. A. Garcia, A. R. Kim, E. Jackson, S. S. Kashmiri, and D. Shukla, "Modeling and flight control of a commercial nano quadrotor," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017, pp. 524–532.

[71] Z. Mustapa, S. Saat, A. M. Darsono, and H. H. Yusof, "Experimental Validation of an Altitude Control for Quadcopter," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 6, pp. 3789–3795, 2016.

[72] M. D. L. Costa De Oliveira, *Modeling, Identification and Control of a Quadrotor Aircraft*. Czech Technical University in Prague, jun 2011.

[73] T. Koussiouris and G. Kafiris, "Controllability indices, observability indices and the hankel matrix," *International Journal of Control*, vol. 33, no. 4, pp. 773–775, 1981.

[74] H. Werner, *Lecture Notes - Control Systems Theory and Design*. Hamburg: Hamburg University of Technology, 2012.

[75] M. Steinbuch and G. Meinsma, *Design Methods for Control Systems*, 2007.

[76] R. M. Murray, *Optimization-Based Control*. California Institute of Technology, 2009.

[77] B. Anderson and J. Moore, *Optimal Control, Linear Quadratic Methods*, T. Kailath, Ed. Prentice Hall International, Inc., 1989, vol. 1.

[78] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*. John Wiley and Sons, 2001, vol. 2.

[79] A. Astudillo, P. Munoz, F. Alvarez, and E. Rosero, "Altitude and attitude cascade controller for a smartphone-based quadcopter," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, jun 2017, pp. 1447–1454.

[80] Android Developer, "SensorEvent," 2015. [Online]. Available: https://developer.android.com/

[81] K. S. Lauszus, "Flight Controller for Quad Rotor Helicopter in X-configuration," Bachelor Thesis, Technical University of Denmark, Kongens Lyngby, Denmark, 2015.