

ATTITUDE AND POSITION CONTROL OF A QUADROTOR USING
ONBOARD VISION SYSTEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ŞERAFETTİN TÜZEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

DECEMBER 2019

Approval of the thesis:

**ATTITUDE AND POSITION CONTROL OF A QUADROTOR USING
ONBOARD VISION SYSTEM**

submitted by **ŞERAFETTİN TÜZEL** in partial fulfillment of the requirements for
the degree of **Master of Science in Mechanical Engineering Department, Middle
East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. M. A. Sahir Arıkan
Head of Department, **Mechanical Engineering**

Assoc. Prof. Dr. Ulaş Yaman
Supervisor, **Mechanical Engineering, METU**

Examining Committee Members:

Assoc. Prof. Dr. Melik Dölen
Mechanical Engineering, METU

Assoc. Prof. Dr. Ulaş Yaman
Mechanical Engineering, METU

Prof. Dr. M. Kemal Özgören
Mechanical Engineering, METU

Assist. Prof. Dr. Ali Emre Turgut
Mechanical Engineering, METU

Assist. Prof. Dr. Masoud Latifi Navid
Mechatronics Engineering, THK University

Date: 06.12.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Şerafettin Tüzel

Signature :

ABSTRACT

ATTITUDE AND POSITION CONTROL OF A QUADROTOR USING ONBOARD VISION SYSTEM

Tüzel, Şerafettin

M.S., Department of Mechanical Engineering

Supervisor: Assoc. Prof. Dr. Ulaş Yaman

December 2019, 119 pages

In this thesis, localization and trajectory following of a quadrotor is obtained with attitude and position control along with the help of an onboard monocular camera. Two control strategies are adopted for this aim, which are well-known PID and nonlinear backstepping controllers. For the PID algorithm, a cascaded structure is preferred so that angular rates are regulated as the inner loop of the attitude controller. At the outer loop of this system, angles are stabilized by producing rate references to the inner loop. PID is a proven method to control nonlinear systems, giving satisfactory system performances. Linearised or simplified system models are generally used to design PID controllers, and implementation is usually straight forward. On the other side, backstepping controller is selected since it does not cancel nonlinearities of the system model. The aim of this method is to obtain the final inputs by creating virtual system inputs in a recursive manner. Since it directly uses system equations of motion, nonlinearities like coupling between axes and actuator frictions are comprised by the controller. This also enables better performance in disturbance rejection. If the strategy is followed properly, backstepping gives satisfactory results.

To be able to design these controllers, quadrotor system is modelled within this study. Equations of motion are obtained and quadrotor behaviour is simulated by using Simulink. The responses of the controllers to given references are compared with the usage of this model representation.

A real platform is formed for the verification of the proposed control methods. The quadrotor uses a Pixhawk board which includes the main control algorithms. These algorithms are formed in Simulink and Embedded Coder is used to transfer the program to the Pixhawk. For the localization operation, RaspberryPi board is selected and also used with Simulink application. From the images obtained by the Raspberry, position errors are propagated so that attitude controllers on the Pixhawk can regulate the errors down to zero for stabilization. Serial communication is preferred as the connection interface and hardware and software schemes are explained along this study. As the main aim orients, designed localization and path tracking capabilities are shown on the target platform.

Keywords: Quadrotor, PID, Backstepping Controller, Path Tracking, Image Processing

ÖZ

4 PERVANELİ İHA’NIN YERLEŞİK GÖRÜNTÜ SİSTEMİ KULLANARAK TUTUM VE KONUM KONTROLÜ

Tüzel, Şerafettin
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ulaş Yaman

Aralık 2019 , 119 sayfa

Bu tez çalışmasında, dört rotorlu bir İHA’nın lokalizasyon ve yörünge takibi yerleşik bir monoküler kamera yardımı ile kontrol edilmiştir. Bu amaç için iyi bilinen PID ve doğrusal olmayan geri adımlamalı kontrol stratejileri benimsenmiştir. PID algoritması için kademeli bir yapı tercih edilmiş olup, böylece açısal hızlar durum kontrolcüsünün iç döngüsü olarak düzenlenmiştir. Bu sistemin dış döngüsünde, iç döngüye açısal oran referansları üreterek açılar stabilize edilir. PID, doğrusal olmayan sistemleri kontrol etmek için tatmin edici bir sistem performansı sağlayan kanıtlanmış bir yöntemdir. Doğrusallaştırılmış veya basitleştirilmiş sistem modelleri genellikle PID denetleyicilerini tasarlamak için kullanılır ve uygulaması genellikle basittir. Diğer tarafta geri adımlamalı kontrol yöntemi, sistem modelinin doğrusal olmayan elemanlarını ihmal etmediğinden tercih edilmiştir. Bu yöntemin amacı, yinelemeli bir şekilde sanal sistem girdileri oluşturarak son girdileri elde etmektir. Doğrudan sistem hareket denklemlerini kullandığından, eksenler arasındaki ilişkiler ve aktüatör sürtünmeleri gibi doğrusal olmayan etkenler kontrolcüler tarafından ele alınabilir. Bu aynı zamanda bozucu etkilere karşı daha iyi performans sağlar. Stratejinin doğru bir şekilde takip

edilmesi durumunda, geri adımlamalı kontrolcü tatmin edici sonuçlar verir.

Bu kontrolcöleri tasarlayabilmek için bu çalışma kapsamında dört rotorlu İHA sistemi modellenmiştir. Hareket denklemleri elde edilmiş ve Simulink kullanılarak dört rotorlu İHA davranışı simüle edilmiştir. Kontrolcölerin verilen referanslara verdiği cevaplar, bu model gösteriminin kullanımı ile karşılaştırılmıştır.

Önerilen kontrol yöntemlerinin doğrulanması için gerçek bir platform oluşturulmuştur. Dört rotorlu İHA, ana kontrol algoritmalarını içeren bir Pixhawk kartı kullanmaktadır. Bu algoritmalar Simulink'te oluşturulmaktadır ve programı Pixhawk'a aktarmak için Embedded Coder kullanılmıştır. Pozisyonlama işlemi için RaspberryPi kartı seçilmiş ve Simulink uygulaması ile de kullanılmıştır. Raspberry tarafından elde edilen görüntülerden, konum hataları hesaplanır ve böylece Pixhawk üzerindeki yönelim kontrolcöleri bu hataları sıfıra götüreceğ şekilde kontrol sağlayabilir. Bağlantı arayüzü olarak seri iletişim tercih edilmiş ve bu çalışmada donanım ve yazılım programları açıklanmıştır. Ana hedefin gösterdiği doğrultuda, tasarlanan lokalizasyon ve yol izleme yetenekleri hedef platformda gösterilmiştir.

Anahtar Kelimeler: 4 Pervaneli İnsansız Hava Aracı, PID, Geri Adımlamalı Kontrolcü, Yörünge İzleme, Görüntü İşleme

To my family...

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to Dear Ulaş Yaman, my thesis supervisor who has guided and encouraged me throughout my study.

I would like to express my sincere love to my family members, especially my beloved wife Rana Tüzel and my brother Fatih Tüzel who endured me patiently throughout my work. Thanks to them I was able to continue this study until the last moment.

I would like to thank my colleague Ersin Gönül in my company who enlightened me with his unique knowledge about the quadrocopter platform and the hardware I was not familiar with before.

I am also grateful to my colleagues Mustafa Gürlü and Ayşe İlgen Bayrak for their advice on control algorithm design.

Finally, I would like to express my gratitude to Semih Arslan, Ali Mert Türker and Mert Ergüven for their help throughout the thesis.

TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENTS	x
TABLE OF CONTENTS	xi
LIST OF TABLES	xiv
LIST OF FIGURES	xv
LIST OF ABBREVIATIONS	xx
LIST OF SYMBOLS	xxi
CHAPTERS	
1 INTRODUCTION	1
1.1 Motivation and Objective	4
1.2 Structure of the Thesis	6
2 LITERATURE SURVEY	7
2.1 Modelling and System Identification	7
2.2 Control Theory	8
2.3 Vision and Pose Estimation	11
3 QUADROTOR MODELLING	15
3.1 Definition of Control Inputs	16

3.2	Reference Frames and Transformation Matrices	18
3.2.1	Transformation Matrices for Translational Velocities	20
3.2.2	Transformation Matrices for Angular Velocities	22
3.3	Nonlinear Dynamic Model	23
3.3.1	Forces and Moments Acting on the Quadrotor	24
3.3.2	Rotational Equations of Motion	26
3.3.3	Translational Equations of Motion	28
3.3.4	Rotor Dynamics	29
3.3.5	State Variables and Equations	31
4	SYSTEM HARDWARE AND SOFTWARE	35
4.1	Hardware Overview	35
4.1.1	Quadrotor Frame	36
4.1.2	Propulsion System	38
4.1.3	Pixhawk	40
4.1.4	Raspberry Pi	41
4.1.5	RC Transmitter and Receiver	42
4.1.6	Lidar	43
4.1.7	Lipo Battery and DC-DC Converters	44
4.1.8	System Layout	45
4.2	Software Overview	46
4.3	Physical Parameters of the Quadrotor	56
4.3.1	Mass Moment of Inertia	57
4.3.2	Propulsion System Parameters	59

4.4	Quadrotor Test Bench	64
5	CONTROLLER DESIGN AND APPLICATIONS ON QUADROTOR	67
5.1	PID Controller	67
5.1.1	Attitude Control of Quadrotor	68
5.1.2	Altitude and Position Control of Quadrotor	76
5.1.3	Attitude Performance of PID Controller on Test Bench	79
5.2	Backstepping Controller	82
5.2.1	Backstepping Control of Rotational Motions	84
5.2.2	Backstepping Control of Translational Motions	87
5.2.3	Attitude Performance of Backstepping Controller on Test Bench	90
5.3	Controller Applications on Quadrotor	94
6	CONCLUSION	111
	REFERENCES	113

LIST OF TABLES

TABLES

Table 4.1	Motor Parameter Identification Test for 100% Battery	60
Table 4.2	Motor Parameter Identification Test for 60% Battery	61

LIST OF FIGURES

FIGURES

Figure 1.1	Quadrotor in construction [1]	3
Figure 1.2	3D printer quadrotor [2]	4
Figure 2.1	Quadrotor and ground robot system [3]	12
Figure 2.2	PIXHAWK controlled quadrotor [4]	13
Figure 3.1	Quadrotor flight configurations; plus configuration (on the left) and cross configuration (on the right) [5]	17
Figure 3.2	Quadrotor forces and moments [1]	17
Figure 3.3	Reference Frames	19
Figure 4.1	Quadrotor platform used in this study	36
Figure 4.2	eCalc - RC Calculator [6]	37
Figure 4.3	S500 Quadrotor Frame [7]	37
Figure 4.4	Motor and propeller pair [8]	38
Figure 4.5	T-motor ESC [8]	39
Figure 4.6	Pixhawk [9]	40
Figure 4.7	Raspberry Pi 3 Model B [10]	41
Figure 4.8	Raspberry Pi Camera [11]	42

Figure 4.9	Taranis X9d Plus Transmitter and Receiver [12]	42
Figure 4.10	Tf Mini Lidar Range Sensor [13]	43
Figure 4.11	Gens Ace 11.1V 3S 4000mAh Lipo Battery [14]	44
Figure 4.12	Diagram of quadrotor components	45
Figure 4.13	Simulink blocks provided by the Support Package [15]	47
Figure 4.14	Pixhawk Simulink model created within the scope of this study .	48
Figure 4.15	Raspberry Pi Simulink model created within the scope of this study [16]	50
Figure 4.16	Data packeting and serial sent in Pixhawk Simulink model . . .	51
Figure 4.17	Counter Difference values for Pixhawk and Raspberry Pi boards	51
Figure 4.18	Loop Back Performance of Pixhawk and Raspberry Pi	52
Figure 4.19	Yaw Angle vs time	52
Figure 4.20	ArucoMarker examples [17]	53
Figure 4.21	ArucoMarker used in this study	54
Figure 4.22	Example for ArucoMarker detection via Raspberry Pi camera . .	55
Figure 4.23	List of programs or threads that are currently operated by Linux Kernel of Raspberry Pi	55
Figure 4.24	Diagram of software architecture	56
Figure 4.25	Setup for Bifular Pendulum Theory [18]	57
Figure 4.26	Yaw angle response to given oscillations in pendulum setup . . .	58
Figure 4.27	Roll and Pitch Angle vs time	59
Figure 4.28	Thrust measurement setup for quadrotor motor-propeller pair . .	60
Figure 4.29	Lipo voltage and thrust relation	61

Figure 4.30	Relation of thrust and the square of propeller speed	62
Figure 4.31	Relation of torque and the square of propeller speed	63
Figure 4.32	Relation of PWM and propeller speed	63
Figure 4.33	Single axis quadrotor test bench	64
Figure 4.34	Three Axes Test Bench	65
Figure 5.1	Roll rate step response	69
Figure 5.2	Closed loop roll rate Bode plot	70
Figure 5.3	Roll angle step response	71
Figure 5.4	Closed loop roll angle Bode plot	71
Figure 5.5	Pitch angle step response	72
Figure 5.6	Closed loop pitch angle Bode plot	73
Figure 5.7	Cascaded PID control diagram for pitch and roll axes	73
Figure 5.8	Yaw angle step response	74
Figure 5.9	Closed loop yaw angle Bode plot	75
Figure 5.10	Cascaded PID control diagram for yaw axis	75
Figure 5.11	Response of X-Position, X-Velocity and Pitch Angle of quadrotor to commanded position reference	77
Figure 5.12	Cascaded PID diagram for x/y position control	78
Figure 5.13	PID diagram for altitude control	78
Figure 5.14	Response of quadrotor to given altitude reference	79
Figure 5.15	Simulink model used for quadrotor simulations	80

Figure 5.16	Response of quadrotor to ramp-wise inputs given in pitch and roll axes	81
Figure 5.17	Response of quadrotor to step-wise inputs given in pitch and roll axes	81
Figure 5.18	Response of quadrotor to sinusoidal inputs given in pitch and roll axes	82
Figure 5.19	Response of roll angle to given step input for changing controller parameters	91
Figure 5.20	Commanded PWM to motors by backstepping controller	92
Figure 5.21	Response of quadrotor to stepwise inputs given in pitch and roll axes	92
Figure 5.22	Response of quadrotor to ramp-wise inputs given in pitch and roll axes	93
Figure 5.23	Control structure used for quadrotor stabilization	94
Figure 5.24	Yaw angle reference management during marker detection	95
Figure 5.25	Roll angle step response	95
Figure 5.26	Closed loop roll angle Bode plot	96
Figure 5.27	Response of quadrotor system to given pilot inputs in pitch and roll axes	97
Figure 5.28	Quadrotor velocity in x/y directions during position hold over marker	98
Figure 5.29	Response of quadrotor system during position hold over marker	99
Figure 5.30	Position and altitude of quadrotor during position hold over marker	100
Figure 5.31	Statistical data for the quadrotor that performs position hold over marker	101

Figure 5.32	Quadrotor velocity in x/y directions during position hold over marker (second flight)	102
Figure 5.33	Response of quadrotor system during position hold over marker (second flight)	103
Figure 5.34	Position and altitude of quadrotor during position hold over marker (second flight)	104
Figure 5.35	Statistical data for the quadrotor that performs position hold over marker (second flight)	105
Figure 5.36	Position and altitude of quadrotor when a sinus reference is commanded for x-position	106
Figure 5.37	Statistical data for the quadrotor when a sinus reference is commanded for x-position	107
Figure 5.38	Position and altitude of quadrotor when a square path reference is commanded	108
Figure 5.39	Statistical data for the quadrotor when a square path reference is commanded	109
Figure 5.40	Photo of the quadrotor during path follow	110

LIST OF ABBREVIATIONS

3D	3 Dimensional
AC	Alternating Current
AM	Additive Manufacturing
BLDC	Brushless DC Motor
CAD	Computer Aided Design
DC	Direct Current
ESC	Electronic Speed Controller
FPV	First Person View
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
IMU	Inertial Measurement Unit
KV	Constant Velocity (of a motor)
LQ	Linear Quadratic
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
RC	Radio Control
RGB	Red Green Blue
RPM	Revolutions Per Minute
RTOS	Real Time Operating System
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Air Vehicle
UDP	User Datagram Protocol
VTOL	Vertical Takeoff and Landing

LIST OF SYMBOLS

b	Aerodynamic thrust contribution
B_ω	Bandwidth
C_τ	Aerodynamic thrust coefficient
C_D	Aerodynamic drag coefficient
d	Aerodynamic drag contribution
e_i	Tracking error for i_{th} state
F_B	Body reference frame
F_E	Earth reference frame
F_{ext}	Sum of external forces acting on the quadrotor
$F_{grav,B}$	Gravitational force expressed in F_B
$F_{grav,E}$	Gravitational force expressed in F_E
F_i	Force generated by i_{th} rotor
$F_{prop,B}$	Total force generated by the propeller expressed in F_B
g	Gravitational acceleration constant
I_{xx}, I_{yy}, I_{zz}	Products of inertia
J	Inertia matrix
J_r	Rotor inertia
K_d	Derivative gain
K_i	Integral gain
K_p	Proportional gain
l	Length of the moment arm
m	Mass of quadrotor
M_{ext}	Sum of external moments acting on the quadrotor
$M_{gyro,B}$	Gyroscopic moments produced by propellers expressed in F_B
M_i	Moment exerted on the quadrotor by i_{th} rotor
$M_{prop,B}$	Total moment acting on the quadrotor system expressed in F_B
r	Motor gear box reduction ratio

R	Transformation matrix
R_{BE}	Transformation matrix from F_E to F_B
R_{EB}	Transformation matrix from F_B to F_E
R_r	Angular transformation matrix from F_E to F_B
T_i	Torque generated by i_{th} rotor
U_1, U_2, U_3, U_4	Control inputs
$V(e_i)$	Lyapunov function for i_{th} tracking error
V_B	Translational velocity vector of quadrotor expressed in F_B
V_E	Translational velocity vector of quadrotor expressed in F_E
x_i	i_{th} system state
x_E, y_E, z_E	Earth reference frame axes
x_B, y_B, z_B	Body reference frame axes
p, q, r	Body angular velocities
u, v, w	Body velocities
α_i	i_{th} control coefficient in backstepping
ϕ	Euler roll angle
θ	Euler pitch angle
ψ	Euler yaw angle
$\dot{\phi}, \dot{\theta}, \dot{\psi}$	Euler rates
ξ	Position Vector of Quadrotor in F_E
η	Orientation Vector of Quadrotor in F_E
τ	Motor time constant
ω	Angular velocity vector of quadrotor
ω_m	Angular speed of the motor
Ω	Relative speed of the propellers
Ω_i	Angular velocity of i_{th} rotor

CHAPTER 1

INTRODUCTION

Quadrotor is a vertical takeoff and landing (VTOL) aerial vehicle that has four rotors. This aerial vehicle could change its attitude and altitude by controlling the angular velocity of each rotor, which are consist of a motor and a propeller group. Most of the quadrotors have symmetric shapes. In general, four symmetric rotor arms are combined with a central board, where all the hardware and functional stuff are located. This symmetry in the shape also simplifies the control of the vehicle, and quadrotors gain well hovering and vertical movement capabilities.

In recent years, with the development of hardware platforms such as microcontrollers, motors and sensors, quadrotor systems have gained huge commercial potential and became worldwide available. Although the first quadrotors were manned, today most of these robots are unmanned and used with simple controller units. This rapid evolution lies in the advantages of these flying robots in comparison to the conventional helicopters and aircrafts. These are

- to be able to move in every direction with high manoeuvrability,
- simple mechanical elements, small sizes and less aerodynamic affects,
- simplicity in vehicle control,
- low cost.

Thanks to these factors, the interest in quadrotor systems has grown rapidly, and nowadays quadrotors have entered our lives within different usage areas. These robots were initially used for simple photography and video shooting, but now they are used

for aerial delivery, communication, mapping, surveillance, search and rescue, border control, mine detection, and etc. There are lots of commercial products available on the market such as the products of Chinese and French companies DJI and Parrot. Some of them have emerged in parallel to the field of use, but a significant portion has appeared because people liked flying these robots. Since the companies have realized that the use of drone is fun, they produce different products and continue to grow the industry. First Person View (FPV) Drones and Racer Drones are the results of this market.

Although the entertainment business is one of the most important factors shaping the market, experimental studies could create new features in continuous development process. From the very beginnings of the quadrotor developments, researchers and developers have studied things that has not been tried before. This has added different usage areas to the quadrotors, and will continue to contribute.

In 2011, Lindsey et al. [1], [19] have created a system in which a team of quadrotors, that have had grippers on the bottom, could have form a structure by using 2.5D truss-like elements. The quadrotor could be seen in Figure 1.1. They have performed several simulations and experiments with real drones on how to build a feasible structure by considering the assembly constraints, and have brought out an algorithm that could have construct any special cubic structure. At that point, people have questioned whether construction with quadrotors are possible or not? That is why these works are important in shaping the course of events.

Another example could be given on additive manufacturing (AM), where the rapid evolution of small scale printing have been seen [20]. For the large scale one however, different research fields should contribute to the subject. There are many suggestions such as giant printers, cable-suspended printers, swarm robots and automated assembly robots, but some researchers have touched on upon the aerial printing [21] [22], [23], [24], [20]. Gerling et al. [23] have proposed the idea of self-organized flying robot systems for the purpose of construction, and have considered the quadrotors as ideal platforms due to their flexibility and precise control. However, major limitation comes with the limited battery life and payload capacity as stated in [2]. Thus, there could be a need of a division of labor between grounded and airborne units [23].

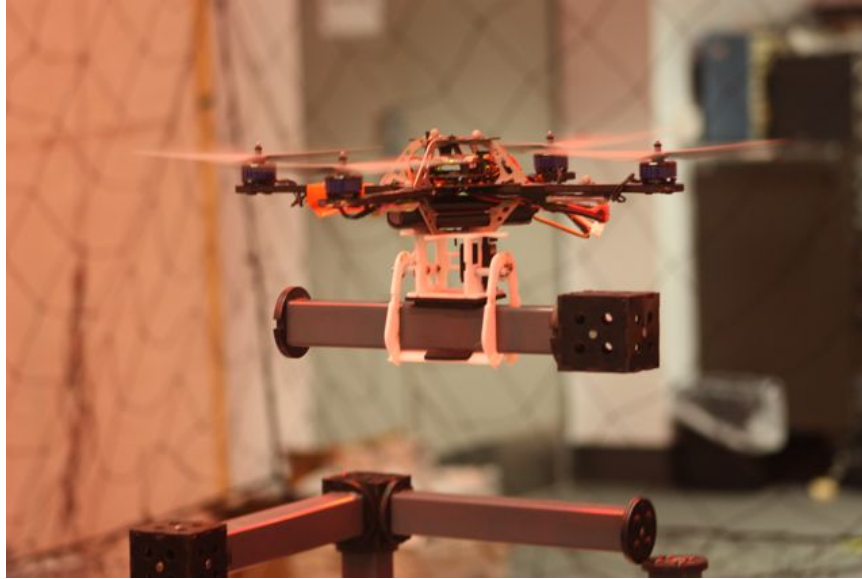


Figure 1.1: Quadrotor in construction [1]

Apart from these reviews, there are direct studies and experiments that have been done on aerial printing [2], [25], [26]. Hunt et al. [2] have combined additive layer manufacturing with aerial robotics. They have examined the feasibility of flying 3D printer robots. The designed quadrotor, which can be seen in Figure 1.2, has deposited polyurethane expanding foam during the flight. The authors have demonstrated the capabilities of the system, and have mentioned the potential usage areas. In another work [25], researchers have formed a cementitious paste solution to be used by the aerial robots so that these machines can create and repair buildings. They have successfully demonstrated the cementitious paste solution with an extrusion-printing mechanism. There is also a patent on aerial printing subject [27], in which drones have been defined to carry and deposit 3D printing material to construct structures. As it can be seen, these works have purposed directly aerial printing.

If it is desired to use a stationary printer, the product must be smaller in size compared to the machine used to form it [26]. Many researchers see this as a major limitation, which ensures continuous research activities including the aerial printing. There may also be mixed solutions that combines aerial machines with ground robots or even with humans. Pothole identification process could be given as an example to this labour division. In their study, Pan et al. [28] have distinguished the normal pavement and potholes, damages in the roads due to ageing and deterioration, by using machine

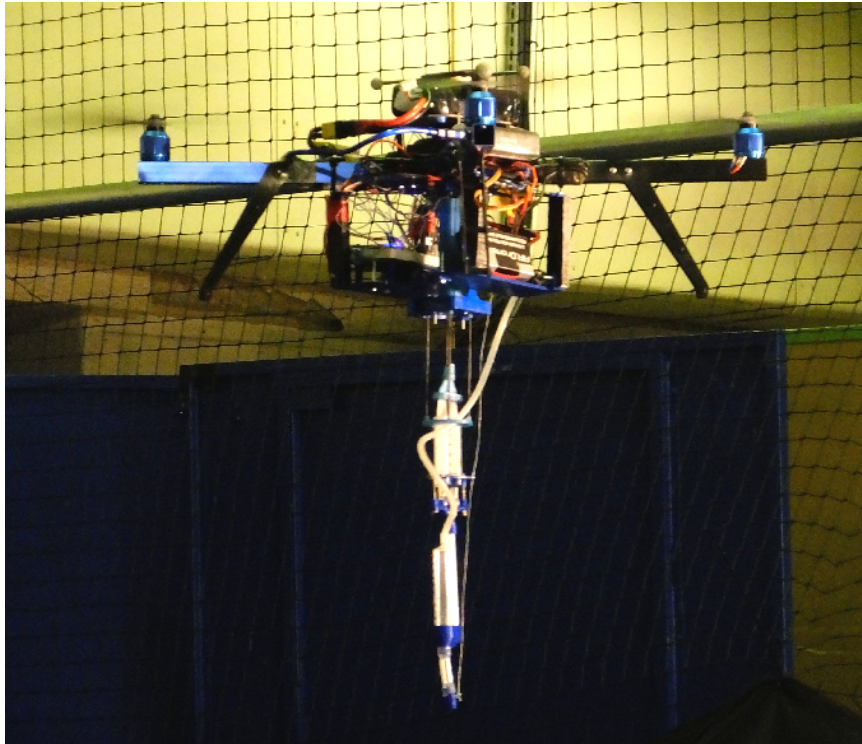


Figure 1.2: 3D printer quadrotor [2]

learning algorithms. They have obtained accurate results on identifying the potholes so that a quick repair action can be taken. This action might be a ground robot or a person. It is also predictable that there can be repair drones on highways in the near future. This is all about the area that the technology will flow into.

Whether drones are used in direct construction or not, people will continue to use them even for terrain and surface investigations. Potential applications include repair, construction of inaccessible areas, and painting. Although battery life and load capacity are the main limitations of quadrotor usage, autonomous control and precise positioning issues can shape the application areas.

1.1 Motivation and Objective

Nowadays, quadrotors appear in different fields with enhanced application capabilities. New interests like aerial printing, painting and object detection require intelligent

pose estimation and robust stabilization abilities. Thus, design of control algorithms takes an important role in designing these autonomous skilled systems. As other researchers focused on this issue, this thesis puts main effort on controller algorithm design and implementation on a real platform.

A successful control approach can be obtained with a good understanding of the UAV dynamics. Thus, the initial goal of this study is to obtain the mathematical expressions of the quadrotor dynamics. Obtained equations can be used to form a simulation model that represents the motion of the quadrotor in air. There is a real quadrotor platform made up for this study. The parameters of this platform will be examined and projected on the simulation models so that a proper algorithm design can be performed.

The stabilization of a quad-rotor helicopter forms the second aim of this work. There are different control strategies on the literature. By investigating flight regime and system dynamics, a controller selection can be made. Two different control methods are independently intended in our quadrotor's control. One of the methods is a non-linear control technique called as *Backstepping*. The other method is the well known linear technique, *PID*. For the applications like aerial printing, painting and monitoring as an assistant system, there is a limit in the speed of application and also the precision is important. Above mentioned methods will be implemented separately and performances will be evaluated for low-speed and hover-like conditions.

Control strategy is important in shaping the motion of quadrotor, however stabilization is possible with a reasonable localization. There are different techniques for UAV systems. Our intention is not to use GPS so that the platform can be used both indoors and outdoors. To detect the target place and perform a trajectory follow task, a vision system will be used for this thesis study. As the most significant objective, both hovering and trajectory following missions are aimed to be demonstrated with the quadrotor.

Usage of camera feedback in pose estimation will allow localization around a selected hover point. Our intention is to check the feasibility of aerial printing with an on-board monocular vision system.

1.2 Structure of the Thesis

The contents of the next chapters are summarized below.

- Chapter 2. A concise information about the recent studies will be given.
- Chapter 3. Dynamical model of the quadrotor will be represented.
- Chapter 4. The hardware components and used software will be introduced with their relation for the purpose of control.
- Chapter 5. Design steps of the controllers will be explained. The results of real flight performance will be compared for controller implementations and vision tracking.
- Chapter 6. In the final part, the conclusions of the thesis study will be given.

CHAPTER 2

LITERATURE SURVEY

In this chapter, a review of studies related to this thesis work has been investigated. For hover and trajectory following capabilities, an accurate representation of the quadrotor dynamics should be obtained in the first place. Then, a proper control law algorithm can be selected to stabilize the quadrotor in desired environments. Since a monocular camera will be used to detect the location of the platform, available studies should also be investigated. There are many different studies and experiments on the quadrotor platforms in literature. Related researches can be summarized under three sections.

2.1 Modelling and System Identification

Having a good dynamical model representation of a system is important in designing controllers. The controller could perform as expected if only the mathematical model represents the real system in a similar behavior. Especially for the nonlinear controller designs, the role of system dynamical model increases. Quadrotor is a system having relatively simple dynamic interactions such that the simply reduced models could estimate the behavior in a pretty good manner. There are some studies on obtaining the quadrotor dynamical model [29], [30]. Although these mathematical expressions could represent quadrotor's dynamics, parameters of them depend on the experimental findings. Thus, they should also be validated.

In some of the studies, researchers have identified and validated their dynamical model instead of finding those parameters with an experiment. As an example, Gremillion et al. [31] have estimated the linear dynamical model of a quadrotor vehicle by

using time domain system identification technique. With a couple of indoor flights, both IMU and Vicon system data have been collected to obtain model parameters and the pole locations of the quadrotor vehicle. In the end, researchers have obtained an accurate estimation of the system.

Another frequency domain estimation have been performed by Niermeyer et al. [32], and they have used outdoor flights data to identify their linear model, where most of the researchers have been using indoor data. In the work, they have used only inertial data and GNSS measurements to estimate the hover dynamics of a quadrotor system. They have also used CIPHER software to validate the model, and have achieved individual state space models for yaw axis, vertical motion and combined pitch-roll behavior.

Sun et al. [33] have performed large-scale wind tunnel tests with a quadrotor to estimate the high-speed dynamics model of the vehicle. The authors have marked that high aerodynamic affects could have dominated the forces and moments on the vehicle body during high-speed regime. Thus, they have suggested a gray-box model identification, where they have used a model that included these aerodynamic interactions. Considering the results, they have achieved an accurate model, but limited to the flight regime of interest.

In a study [34], researchers have suggested a method of verifying and validating the accuracy of a quadrotor model in the linear flight regime. They have applied specific control inputs in a range of frequencies and tried to fit the known model parameters to the test result. In the end, researchers have successfully extracted the model parameters that could be used as the initial model during the design process.

2.2 Control Theory

A variety of control methods are applied to achieve the best precision. PID control method is the most familiar and easy to use method. In their work [35], Bouabdallah et al. have applied PID and LQ control strategies on a quadrotor restricted on a test bench. They have used a simple decoupled model for the PID design, and have used a more complex one for the LQ implementation. They have achieved better

performance with the classical controller, and have concluded this as the imperfection of the complex model.

There is also a study [36], in which the response of the PID controller has been validated through a series of flights.

As opposed to linear control methods, there are so many works related to nonlinear control of quadrotor UAV. The most common nonlinear control techniques used to control the quadrotors are backstepping, sliding mode, and feedback linearization.

In a study [37], backstepping and sliding-mode control strategies have been applied to a quadrotor on a test bench. The average performance of the sliding-mode have been based on the switching nature of controller, which actually has increased the vibrations. Backstepping controller has showed a fair performance holding the orientation of the quadrotor although there has been relatively high disturbances.

In another work [38], researchers have modelled quadrotor dynamics by considering the aerodynamic affects formed by the vehicle motion. They have used the model parameters, have designed an Integral-backstepping controller and have made direct flight experiments without re-tuning the model. The researchers equipped their OS4 quadrotor with a camera system and sonar distance sensors, and in addition they have also presented take-off, landing, hover and collision avoidance performances of the quadrotor system.

There is also a study [39], where feedback linearization controller has been combined with a wind parameter estimation via a Lyapunov function.

Backstepping is based on the Lyapunov stability, and this has been shown in [40], where the authors have presented a robust backstepping controller design that has guaranteed the convergence and stability. They have showed that the maximum steady state error on Euler angle tracking has been in a desired bound.

For the nonlinear controllers, another comparison study has been done by Lee et al. [41]. In their paper, they have derived a feedback linearization controller that is derived from the conventional simplified model. Then, they have represented an adaptive sliding mode controller, which has been more robust to noise and uncertain-

ties. The difference has been based on the usage of reduced order of derivatives in the inputs. The researchers have also achieved a better performance in the estimation and toleration of ground effects.

In one of the most recent comparisons [42], the authors have designed feedback linearization, backstepping and sliding mode controllers, and have compared the performances of each of the proposed methods on a Qball-X4 quadrotor UAV. The authors have obtained better stability and robustness with sliding mode controller. They have also noted that backstepping strategy could have been used if decoupling of axes is required.

There are also less known, successful control techniques that can be used to control a quadrotor. One of them is the dynamic inversion method. In a study [43], the researchers have designed a two-loop dynamic inversion controller with an aim of handling underactuation and coupling between axes. They stabilized the internal control loop with a robust controller and have suggested a modified inversion loop to enhance stability and tracking performance. They have explained the stability based on Lyapunov, and have showed simulation results of the designed system.

Model predictive controller is another algorithm to stabilize the aircraft, where they are used less due to their high computation power demands compared to simple linear controllers. Nevertheless, they show successful results when a proper dynamical model is used. As an example, a switching model predictive controller has been suggested in [44] to reject the wind disturbances that a quadrotor faces during the flight. The designed controller has used the piecewise model of the quadrotor dynamics, in which turbulence affects has been included as a disturbance. The authors verified their algorithm with experimental flights, and have showed the capability of quadrotor's attitude tracking while subjected to the wind disturbances.

Adaptive controllers are the other strategies having high robustness to the disturbances. In an experimental study [45], authors have designed a direct approximate-adaptive control using a nonlinear approximator. The proposed method has updated the adaptive parameters such that the quadrotor could have hold unknown payloads and have provided robustness to the disturbances. The authors have performed an on-line training by varying the weights of the payloads, and have aimed approximately

the same outputs. They have also formed a test stand and have validated the performance of the proposed controller obtaining successful results.

Another adaptive controller study has been performed in [46], where the authors have designed a direct and indirect model reference adaptive control based on the Lyapunov stability. They have carried out indoor flight tests, and have concluded that combined model reference adaptive control has enabled smoother parameter estimates and has increased the performance of the quadrotor. They have also noted that these controllers could have tolerate the motor faults and thrust anomalies.

2.3 Vision and Pose Estimation

Vision is one of the most important subjects in pose estimation of a quadrotor. With the increase in interest on the autonomous flying machines, the need of accurate pose estimation has emerged. Today, vision systems are being used as the primary estimation sensors to detect objects, determine the device position and holding the orientation. Thus, there are pretty good number of studies about vision and pose estimation.

Markable first reference on this subject dating back to 2002, when Altuğ et al. [47] have controlled a cross-type quadrotor using the feedback of a stationary ground camera. They have formed a platform to restrict the vertical and yaw motion, and applied backstepping and feedback linearization control techniques to stabilize the quadrotor. They have also concluded that the quadrotor could not be made fully autonomous with a single camera, and second onboard camera shall be implemented.

In another work [48], the researchers have used a ground camera to compute the X-Y positions and yaw angle of the quadrotor aerial vehicle. The visual localization results have been shown as very stable.

Li et al. [3] have developed a vision guided quadrotor system following a ground robot carrying a visible marker on, Figure 2.1. They have obtained an accurate path following performance both indoors and outdoors using the onboard camera and IMUs. A pose estimation algorithm has been proposed and validated with the path following experiments.



Figure 2.1: Quadrotor and ground robot system [3]

Not all researchers have used monocular cameras, but some of them have tried other commercial products. In a study carried out in 2011, the researchers have used Microsoft Kinect Sensor to control the altitude of the quadrotor [49]. With the usage of depth map feedback from the sensor, a successful altitude control performance has been obtained.

Mier et al. [4] has made a significant contribution by providing both hardware and software system for the micro-air vehicles. They have created an open-source system, the PIXHAWK seen in Figure 2.2, that contains microcontroller, IMUs, cameras, GPS and external interfaces like radio communication so that the system could have

localization, obstacle avoidance and pattern recognition features. The system has had a support for different aerial vehicles such as quadrotors, helicopters and planes. The developers of PIXHAWK have obtained successful results on localization and path following during the experiments.

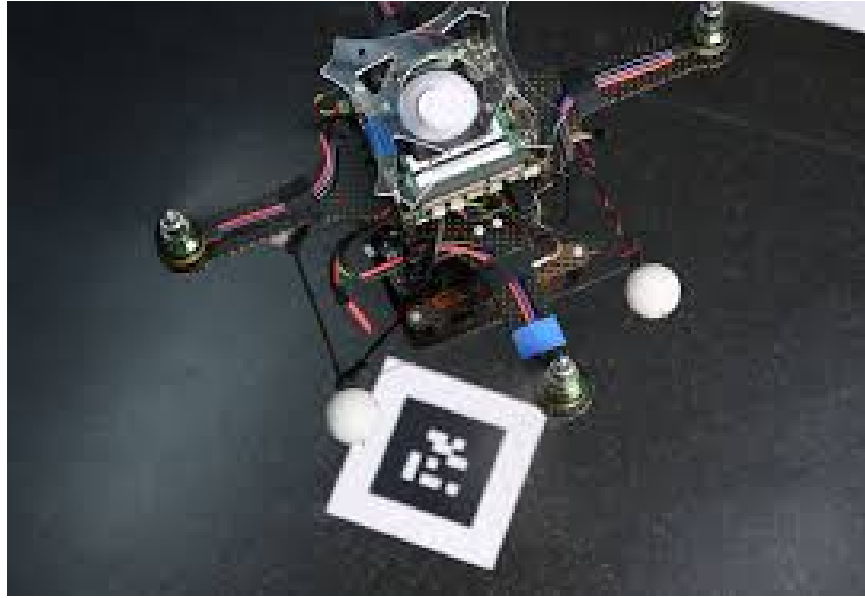


Figure 2.2: PIXHAWK controlled quadrotor [4]

In another work [50], onboard vision system of the commercial AR.Drone has been used to get X and Y direction feedbacks. Due to the computational restrictions of the quadrotor, a color based identification method has been adopted to recognize the marker on the ground. It was shown from the target tracking results that the system is sensitive and could be easily affected by the noise and disturbance.

Yang et al. [51] have performed a precise position hold and landing to a landing pad marked with 'H' letter in their work. By having an onboard vision system, 6 degrees of freedom pose estimation of the vehicle relative to the landing pad has been obtained in a precise manner.

In another work [52], trajectory tracking and positioning of commercial AR.Drone has been performed with a Kalman Filter implementation. AR.Drone have sent the inertial and visual data to a central computer, where the filter fuses the data and feedback to the control algorithm. They have used a stability proven nonlinear controller, and the positioning and path following results have been shown as precise.

Pose estimation is generally desired over a target. However, there could be unknown environments where a respective navigation is needed. In some of the studies [53], [54], autonomous waypoint navigation and obstacle avoidance has been performed by the usage of on board cameras and matching algorithms implemented on on-board controllers.

In a different study [55], indoor localization of the quadroter has been based on Ultra-WideBand technology, where the vision is in the second stage. The authors have used the ranging measurements to reduce the IMU errors and have obtained an accuracy of 10 cm.

CHAPTER 3

QUADROTOR MODELLING

In this chapter, main dynamic model of the quadrotor is provided. Two different methods can be used to obtain the governing mathematical expressions. One is the Lagrangian formalism as used in [29]. The other one is the Newton-Euler formalism that is used by most of the researchers [30], [56], [57], [58], [59], [37], [36], [60]. Since it is more comprehensible, Newton-Euler formalism is followed in the derivation of quadrotor's nonlinear expressions.

There are some assumptions prior to the derivation. These are listed below.

- The quadrotor's centre of gravity coincides with the origin of the body fixed frame.
- The quadrotor's structure is rigid and symmetrical.
- Propellers and motors are rigid.
- Thrust and drag forces are proportional to the square of propeller's speed.

These assumptions are actually important in terms of simplifying the mathematical model that will be created. However, when considering this simplicity, the dynamics to be neglected should be taken into consideration. The assumption of rigid body is one of the most general assumptions. It is assumed that the body is not deformed under applied forces. Quadrotor housing is rigid and does not deform. If there is an elastic part in the system, analytical solutions should be applied because there would be different behaviours to the deflections on the body. The closest part to this elasticity is the propellers. In fact, the propellers deflect from the connection point to the end point due to the force they carry. This deflection changes throughout the

quadrotor flight envelope and causes different thrust forces. However, this change is negligible for small parts such as the quadrotor propeller which also carry relatively little loads. For a helicopter blade, large deflections can be mentioned and this change should also be considered in modelling. With Blade Element Momentum Theory, this phenomenon can be calculated and reflected in the model, but it causes additional parameters to be created and complexity of the model. As in the case of other assumption, if the centre of gravity does not conflict with the origin of the body fixed frame, a series of operations will be required to express the movement in the body frame. In order to avoid this burden, adjustments are made such that the centre of gravity is in the centre of the quadrotor. The small differences are negligible.

The model of the quadrotor should include aerodynamic effects of the propeller rotation, gyroscopic effects due to change in rigid body orientation, inertial effects such as variation in propeller speed and gravity effect. Before describing these effects, however, control inputs should be defined in detail.

3.1 Definition of Control Inputs

The quadrotor's motion is shaped by the lift forces generated by 4 identical propeller and DC motor pairs. Two opposite side motors rotate clockwise, while the other pair turn counter-clockwise. This also removes the need of a tail rotor. Used quadrotor has fixed-pitch propellers which means angle of attack is constant. Thus, motion can be controlled by only varying the propeller speeds. At that point, there are two possible control orientations for the quadrotor system. One is the plus configuration where quadrotor advances in the direction of single rotor, and the other one is the cross configuration where the system moves in the direction of two rotors. These configurations have been described in [5] and can be seen in Figure 3.1.

Both of the configurations have been used in previous studies [43], [39], [41], [52], [47], however a cross type configurations is preferred in this thesis work. This is because the central board of the quadrotor is a rectangle that extends in the cross directions. This breaks down the symmetry in inertia calculations if the plus configuration is selected. With the cross configuration, body frame is also appointed in the

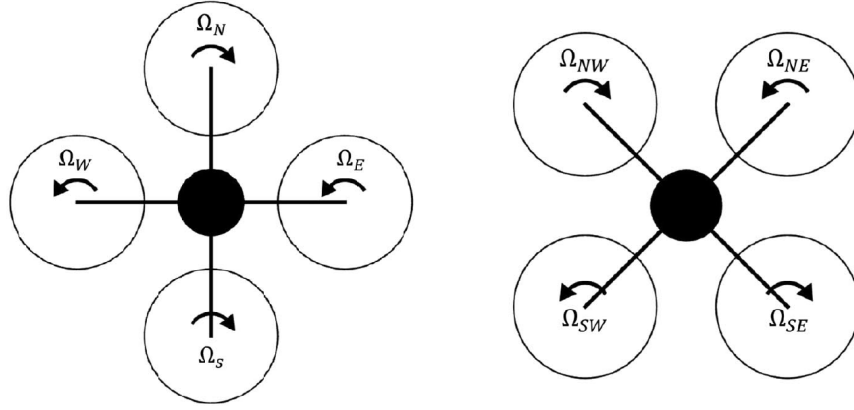


Figure 3.1: Quadrotor flight configurations; plus configuration (on the left) and cross configuration (on the right) [5]

line of choice. This assignment can be seen in Figure 3.2.

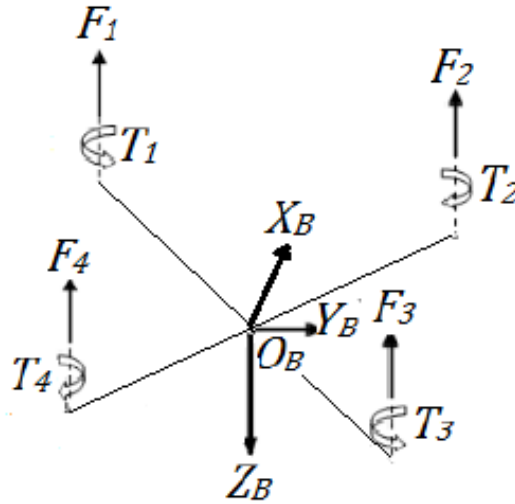


Figure 3.2: Quadrotor forces and moments [1]

There are 4 rotors of the quadrotor, which means 4 control inputs can be defined. In the previous studies [35], [38], [36], [42], [40] and [44], controller designs have been made individually for basic body motions such as pitch motion, roll motion and yaw motion. At the end, researchers have combined the outputs of the controllers with a mixing function to obtain the desired propeller speeds. This brings simplicity in designing the controller of respective motion since the number of inputs is reduced to one. With that approach, number of rotors could also be increased because controller

will be the same, but the outputs can be distributed over the rotors.

For this thesis work, the same strategy is adopted and control inputs are defined as

- U_1 : throttle input, which is the total force generated by the propellers.
- U_2 : roll input, which is the moment difference around x-axis due to left-side and right-side propellers.
- U_3 : pitch input, which is the moment difference around y-axis due to front and rear propellers.
- U_4 : yaw input, which is the net torque exerted on the system around z-axis by the 4 propellers.

In accordance with the above descriptions, these inputs are expressed with the following equations:

$$\begin{aligned}
 U_1 &= F_1 + F_2 + F_3 + F_4 \\
 U_2 &= l(F_1 - F_2 - F_3 + F_4) \\
 U_3 &= l(F_1 + F_2 - F_3 - F_4) \\
 U_4 &= -M_1 + M_2 - M_3 + M_4
 \end{aligned} \tag{3.1}$$

In Equation 31, l denotes the length of the moment arm for each rotor.

3.2 Reference Frames and Transformation Matrices

There are two coordinate frames to be defined for the ongoing modelling section. The first one should be an inertial frame so that dynamics model can be derived with Newton's laws. This frame can be selected as the Earth reference frame which is fixed at the ground. The axes of this frame are notated with N, E and D symbols which indicates that the axes extends in North, East and Downwards directions. Our quadrotor will fly over this fixed reference frame, and the orientation of it can be expressed relative to that frame with the usage of Euler angles (ϕ, θ, ψ) . Second frame is the body fixed reference frame that is attached to the centre of gravity of our quadrotor. The axes of this frame have already been placed with the choice of cross type

configuration as explained in Section 3.1. These two reference frames can be seen in Figure 3.3.

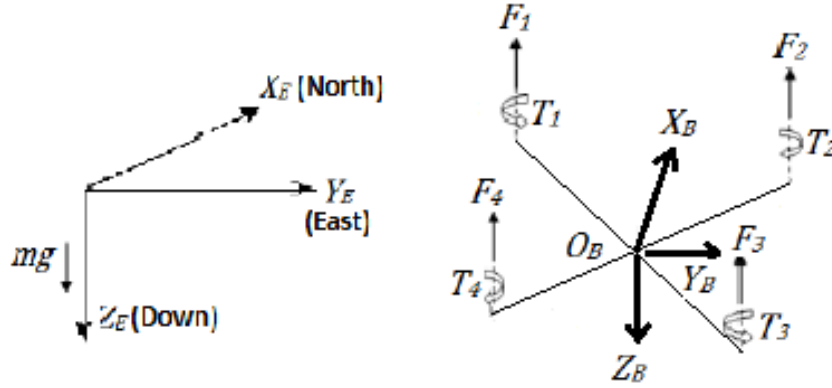


Figure 3.3: Reference Frames

It is better to represent equations of the quadrotor dynamics in the fixed body frame because the inertia matrix is time invariant and control inputs are defined on the body fixed frame [30]. The on-board sensor readings can be easily converted to the body fixed frame for the calculations. At that point, some transformation matrices can be defined to relate the body fixed frame to the Earth frame.

The position and orientation of the quadrotor can be defined relative to the ground with the following notations:

$$\begin{aligned}\xi &= \begin{bmatrix} x & y & z \end{bmatrix}^T \\ \eta &= \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T\end{aligned}\tag{3.2}$$

In Equation 3.2, ξ is expressed in Earth frame (F_E) and represents the absolute position of the quadrotor centre of gravity relative to the Earth reference frame. Here, orientation of the quadrotor relative to the Earth frame is also represented with η as expressed in (F_E).

By using these vectors, the positions and orientation of the quadrotor relative to the Earth reference frame can be identified. In addition to these, one can use $\dot{\xi}$ and $\dot{\eta}$ as the generalized translational and angular velocity vectors expressed with respect to F_E . If the velocities expressed in the body frame are required, some transformations should be performed.

Derivation can be started with the notations of the velocities (wrt F_E) in the body fixed frame, which are expressed in Equation 3.3 below.

$$\begin{aligned} V_B &= \begin{bmatrix} u & v & w \end{bmatrix}^T \\ \omega &= \begin{bmatrix} p & q & r \end{bmatrix}^T \end{aligned} \quad (3.3)$$

Although the process looks the same, there is a conversion from inertial to body fixed frame, and there are differences in obtaining the transformation matrices of angular and translational velocities. This is due to the fact that not all of the Euler rates are measured in the inertial frame [60]. It is known that ψ is measured in the inertial frame, but ϕ and θ are measured in intermediate reference frames.

3.2.1 Transformation Matrices for Translational Velocities

If the angular velocity case is considered for later, translational one can be investigated. The translational velocity of the body fixed frame can be expressed in terms of the translational velocity of the Earth frame by making three successive rotations. These rotations, in combination, describe the orientation of the body frame with respect to the Earth frame. The order of rotations are important, and "yaw, pitch, roll" order will be followed as used by the researchers in [60], [61], [62] and [30].

As noted in [60], 1→2→3 rotation is expressed as $R = R_3 R_2 R_1$ in the matrix multiplication order. Thus, $R_\phi R_\theta R_\psi$ matrix multiplication order should be used to represent the desired rotation. Following equation can be written down to find the transformation matrix R_{BE} that transforms Earth frame (F_E) to body fixed frame (F_B):

$$R_{BE} = R_\phi R_\theta R_\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & s(\phi) \\ 0 & -s(\phi) & c(\phi) \end{bmatrix} \begin{bmatrix} c(\theta) & 0 & -s(\theta) \\ 0 & 1 & 0 \\ s(\theta) & 0 & c(\theta) \end{bmatrix} \begin{bmatrix} c(\psi) & s(\psi) & 0 \\ -s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

For simplicity, *cosine* and *sine* functions are denoted with c and s symbols in long expressions like Equation 3.4. By simplifying Equation 3.4, the final form of the

transformation matrix R_{BE} is obtained as follows:

$$R_{BE} = \begin{bmatrix} c(\theta)c(\psi) & c(\theta)s(\psi) & -s(\theta) \\ s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & s(\phi)c(\theta) \\ c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) & c(\phi)c(\theta) \end{bmatrix} \quad (3.5)$$

Now, translational velocity of the quadrotor, as expressed in Earth frame F_E , can be represented with the following equation:

$$V_B = R_{BE}V_E = R_{BE}\dot{\xi} \quad (3.6)$$

In another case, one may need to go from the body fixed frame representation to the one of Earth. This means, there is a need of an inverse relation as shown in Equation 3.7.

$$\dot{\xi} = V_E = R_{EB}V_B \quad (3.7)$$

Let's try to generate the new matrix by manipulating Equation 3.6. By multiplying both sides of the equation with the inverse of the rotation matrix found in 3.5, an expression similar to 3.7 is obtained.

$$V_E = R_{BE}^{-1}V_B \quad (3.8)$$

From Equations 3.7 and 3.8, following result is concluded:

$$R_{EB} = R_{BE}^{-1} \quad (3.9)$$

It is known that the transformation matrix in Equation 3.5 is non-singular, thus the reverse of it exists, which gives a solution for Equation 3.9. Instead of finding the reverse of the matrix, the transpose of the matrix can be simply used as its reverse. This is a useful property of orthogonal matrices. Since the matrices in 3.4 are orthogonal, resulting transformation matrix is also orthogonal, and thus, this property may be used to find matrix R_{EB} :

$$R_{EB} = R_{BE}^T = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \quad (3.10)$$

3.2.2 Transformation Matrices for Angular Velocities

As mentioned earlier, not all of the Euler rates are measured in the inertial frame. Thus, the rotations described in Section 3.2.1 cannot be used directly. Body angular velocities $[p \ q \ r]$ can be related with Euler rates $[\dot{\phi} \ \dot{\theta} \ \dot{\psi}]$ by using the following relation [60]:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = R_\phi R_\theta R_\psi \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R_\phi R_{\dot{\theta}} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_\phi \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \quad (3.11)$$

R_ϕ , R_θ and R_ψ are already defined in Equation 3.4. Besides, $R_{\dot{\phi}}$, $R_{\dot{\theta}}$ and $R_{\dot{\psi}}$ can be accepted as a unit matrix I since $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ are small. By substituting them into Equation 3.11, the following expression is obtained:

$$\begin{aligned} \begin{bmatrix} p \\ q \\ r \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\ &+ \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -\dot{\phi} - \sin(\theta)\dot{\psi} \\ \cos(\phi)\dot{\theta} + \sin(\phi)\cos(\theta)\dot{\psi} \\ -\sin(\phi)\dot{\theta} + \cos(\phi)\cos(\theta)\dot{\psi} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \end{aligned} \quad (3.12)$$

By calling the angular transformation matrix with R_r notation, relation of angular velocities can be expressed as follows:

$$\omega = R_r \dot{\gamma} \quad (3.13)$$

where,

$$R_r = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{bmatrix} \quad (3.14)$$

Euler rates can also be represented in terms of body angular velocities. Simply multiplying both sides of the Equation 3.13 with R_r^{-1} , the following relation is obtained:

$$\dot{\eta} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix} \omega \quad (3.15)$$

There is an orthogonal vector of body angular velocities, however it is not true for Euler rates vector. This brings the following:

$$\dot{\eta} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \neq \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_I \quad (3.16)$$

This non-orthogonality brings a singularity in the transformation matrix of Equation 3.16. Singularity can be seen when $\theta = \pm 90^\circ$. To avoid this situation, pitch angle can be limited or a quaternion based vector rotation can be used.

Equation 3.15 may also be simplified because the quadrotor will mainly move around the hover point, where the angles ϕ and θ are close to zero. By using the small angle assumption, R_r simplifies to an identity matrix $I_{3 \times 3}$. Therefore, the relation between body angular velocities and Euler rates become:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (3.17)$$

Up to this point, reference frames have been described and necessary transformation matrices have been obtained. Now, the derivation can be carried on with dynamics modelling.

3.3 Nonlinear Dynamic Model

In order to obtain the representation of quadrotor's nonlinear model, Newton-Euler formalism is adopted. Newton's equations are based on the inertial reference frames. For our case, additional terms come up due to the selection of body fixed reference

frame. Newton's equations of motion describe the combined rotational and translational dynamics of a rigid body. These equations can be investigated under two subsystems: rotational subsystem (roll, pitch and yaw) and the translational one (altitude, x and y). According to these, dynamics of the rigid body are expressed under the following generalized form [62], [60]:

$$J\dot{\omega} + \omega \times J\omega = \sum M_{ext} \quad (3.18)$$

$$m\dot{V}_B + \omega \times mV_B = \sum F_{ext} \quad (3.19)$$

At the right hand side of Equations 3.18 and 3.19, external forces and moments acting on the quadrotor body are expressed. It is better to start with describing these effects.

3.3.1 Forces and Moments Acting on the Quadrotor

Quadrotor's motion is shaped by the external forces and moments exerted on the system. These effects can be expressed in the body fixed reference frame. The main contribution comes with the propeller rotation. Besides, some of the external effects can be neglected to form a simpler model. Let's start with the propeller force and moments.

As the propellers rotate, aerodynamic forces and moments are produced and exerted on the system. In Figure 3.2, force and moment of each rotor is indicated. By specifying each rotor with "i" subscript, the following expressions can be written [57], [35], [29], [40]:

$$\begin{aligned} F_i &= b\Omega_i^2 \\ M_i &= d\Omega_i^2 \end{aligned} \quad (3.20)$$

In Equation 3.20, force and moments of the i^{th} rotor are expressed. b and d terms represent the aerodynamic thrust and drag contributions. These factors depend on rotor blade radius and area, air density, speed of the rotor and experimental aerodynamics coefficients C_τ and C_D . Since the quadrotor's altitude variation is somehow limited, it can be assumed that air density is constant. Although b and d are not constant over the entire flight regime, they are mainly dependent on varying propeller speed Ω_i . Thus,

these terms can be experimentally determined for propulsion system by scanning the propeller speed range.

In Section 3.1, the control inputs are defined as U_1 , U_2 , U_3 and U_4 and their relations have been given in Equation 31. Now, these inputs can be put into vector form to be able to use in the equations of motion.

The total force generated by the propellers can be written in frame F_B as:

$$F_{prop,B} = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \quad (3.21)$$

There is also an expression for the moments acting on the quadrotor system, $M_{prop,B}$. If “ l ” is considered as the moment arm length for each rotor, the following equation can be written:

$$M_{prop,B} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} F_1 l - F_2 l - F_3 l + F_4 l \\ F_1 l + F_2 l - F_3 l - F_4 l \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix} = \begin{bmatrix} bl(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ bl(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \quad (3.22)$$

In addition to propeller force and moments, there is a contribution of the gravitational force. It can be expressed in Earth frame F_E as follows:

$$F_{grav,E} = m \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (3.23)$$

This gravitational force can also be expressed in body fixed frame F_B by using the rotation matrix R_{BE} found in Equation 3.5.

$$F_{grav,B} = m R_{BE} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (3.24)$$

Another moment contribution comes from the gyroscopic effects produced by the propellers. Although quadrotor propellers rotate in opposite directions to balance the torque, net moment will not be zero for most of the time. There is a fact that pitch and

roll rates will not be zero, and thus, the inertia of the rotors J_r should be considered. Following equation can be written to describe the gyroscopic moment in F_B [63], [59], [30]:

$$M_{gyro,B} = -\omega \times \begin{bmatrix} 0 \\ 0 \\ J_r \Omega \end{bmatrix} \quad (3.25)$$

In Equation 3.25, Ω defines the propeller's relative speed and given as follows:

$$\Omega = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \quad (3.26)$$

In addition to the gyroscopic moment, there is also aerodynamic drag force. In this thesis study, quadrotor will make position hold most of the time and it will not reach high speeds. Thus, the drag force might be neglected.

Now, Equations 3.21, 3.22, 3.24 and 3.25 can be combined to obtain $\sum F_{ext}$ and $\sum M_{ext}$.

$$\sum F_{ext} = \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} + mR_{BE} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (3.27)$$

$$\sum M_{ext} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} - \omega \times \begin{bmatrix} 0 \\ 0 \\ J_r \Omega \end{bmatrix} \quad (3.28)$$

3.3.2 Rotational Equations of Motion

Equation 3.18 will be used to derive the rotational equations of motion. On the left hand side of the equation, ω is already defined in Equation 3.3. $\dot{\omega}$ can also be expressed as follows:

$$\dot{\omega} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \quad (3.29)$$

Since the derivation has been made in the body fixed frame, a time independent inertia matrix is obtained. It has been already assumed a symmetric frame. Thus, this will

bring a diagonal matrix, whose off-diagonal elements are zero. Finally, quadrotor's inertia matrix can be defined as:

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (3.30)$$

In Equation 3.30, I_{xx} , I_{yy} and I_{zz} represent the moments of inertia around principle axes.

The rotational equations of motion can be written with the previously introduced vectors as:

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ J_r \Omega \end{bmatrix} \quad (3.31)$$

By rearranging Equation 3.31, the final form of the rotational equations of motion are obtained as follows:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})qr/I_{xx} \\ (I_{zz} - I_{xx})pr/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} \end{bmatrix} + \begin{bmatrix} U_2/I_{xx} \\ U_3/I_{yy} \\ U_4/I_{zz} \end{bmatrix} + \begin{bmatrix} -J_r q \Omega / I_{xx} \\ J_r p \Omega / I_{yy} \\ 0 \end{bmatrix} \quad (3.32)$$

With the integration of Equation 3.32, body angular rates $[p, q, r]$ are obtained. To obtain Euler angles, the body rates should be used with transformation matrix R_r that is found in Equation 3.15:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi) \tan(\theta) & \cos(\phi) \tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3.33)$$

Now, Equation 3.33 can be integrated to get the Euler angles $[\phi, \theta, \psi]$. With this step, the rotational dynamic model of the quadrotor have been obtained with Equations 3.32 and 3.33.

Although the rotational equations of motion have been found successfully, they seem quite complex to use in control algorithm design. Therefore, one might want to simplify these equations. In Section 3.2.2, small angle assumption has been made and

Equation 3.17 have been obtained. By using the equality represented there, $[p, q, r]$ can be replaced with $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$. With that change, the simplified version of the rotational equations of motion can be found as:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})\dot{\theta}\dot{\psi}/I_{xx} \\ (I_{zz} - I_{xx})\dot{\phi}\dot{\psi}/I_{yy} \\ (I_{xx} - I_{yy})\dot{\phi}\dot{\theta}/I_{zz} \end{bmatrix} + \begin{bmatrix} U_2/I_{xx} \\ U_3/I_{yy} \\ U_4/I_{zz} \end{bmatrix} + \begin{bmatrix} -J_r\dot{\theta}\Omega/I_{xx} \\ J_r\dot{\phi}\Omega/I_{yy} \\ 0 \end{bmatrix} \quad (3.34)$$

In the simulations, Equations 3.32 and 3.33 will be used to represent the rotational dynamics of the quadrotor. However, Equation 3.34 will be adopted in control law design process due to its simpler form.

3.3.3 Translational Equations of Motion

Equation 3.19 will be used to derive the rotational equations of motion. $\sum F_{ext}$ has already been derived on the right hand side of the equation. The only unknown is the \dot{V}_B term that can be found by differentiating Equation 3.7 with respect to time.

$$\ddot{\xi} = \dot{R}_{EB}V_B + R_{EB}\dot{V}_B \quad (3.35)$$

There is a need of a representation for \dot{R}_{EB} , which has emerged in Equation 3.35. Since the transformation matrix R_{EB} is orthogonal, the properties of orthogonality [63] can be used, [62] so that \dot{R}_{EB} can be expressed as the dot product of R_{EB} with the skew symmetric matrix of ω . Then, \dot{R}_{EB} becomes:

$$\dot{R}_{EB} = R_{EB}\tilde{\omega} \quad (3.36)$$

It is also known that dot product of any vector with a skew symmetric matrix $\tilde{\omega}$ can be expressed as a cross product of ω with that vector [63], [61], [62]. This can be expressed with Equation 3.7:

$$\tilde{\omega}S = \omega \times S \quad (3.37)$$

By substituting Equations 3.36 and 3.37 into Equation 3.35, the following expression is obtained:

$$\ddot{\xi} = R_{EB}(\omega \times V_B) + R_{EB}\dot{V}_B \quad (3.38)$$

Finally, \dot{V}_B term can be pulled from Equation 3.38 as follows:

$$\dot{V}_B = R_{EB}^{-1} \ddot{\xi} - \omega \times V_B \quad (3.39)$$

Now, by substituting Equations 3.39 and 3.27 into Equation 3.19, translational equations of motion are obtained as:

$$\begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} + m R_{BE} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = m(R_{EB}^{-1} \ddot{\xi} - \omega \times V_B) + \omega \times m V_B \quad (3.40)$$

By multiplying both sides of the Equation 3.40 with R_{EB} and simply dividing both sides with constant m , the following simplified form is obtained:

$$R_{EB} \begin{bmatrix} 0 \\ 0 \\ -U_1/m \end{bmatrix} + R_{EB} R_{BE} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = R_{EB} R_{EB}^{-1} \ddot{\xi} \quad (3.41)$$

In Equation 3.41, $R_{EB} R_{EB}^{-1}$ and $R_{EB} R_{BE}$ terms are simplified to yield an identity matrix $I_{3 \times 3}$. Thus, equation can be rewritten as follows:

$$\ddot{\xi} = R_{EB} \begin{bmatrix} 0 \\ 0 \\ -U_1/m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (3.42)$$

At the last step, the final version of translational equations of motion can be obtained by putting $[\ddot{x}, \ddot{y}, \ddot{z}]$ into $\ddot{\xi}$ as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R_{EB} \begin{bmatrix} 0 \\ 0 \\ -U_1/m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (3.43)$$

3.3.4 Rotor Dynamics

In Sections 3.3.2 and 3.3.3, rotational and translational equations of motion have been obtained for the quadrotor. These equations are based on the forces and moments produced by the propellers. However, there is not a linear relationship between

commanded and obtained control inputs on the real platform. Control law algorithm will calculate the required rotational speed for each rotor to maintain the attitude of the quadrotor. However this reference cannot be followed by the rotors instantly due to the dynamics of the rotor.

Today, most of the quadrotor platforms uses Brushless DC Motors (BLDC) as the hearth of their propulsion system. These motors are practical to use because of high torque capacity and low friction. In general, DC motors are controlled by varying the DC voltage on the motor armature. There are two or more permanent magnets that impose a magnetic field that enables rotor's rotation. By applying a DC current to the armature, generated magnetic forces rotate the rotor hub. For the BLDC case, however, there is a difference in control mechanism. These motors need a electronic speed controller unit (ESC) to feed the stator unit. These ESC systems create three-phase AC output (instead of DC power) to rotate the rotor. Although they give AC output, these units take DC power input. That is why BLDC has slightly different dynamics than standard brushed DC motor. Even so, the dynamics of a BLDC at steady state are quite similar to the one of DC motors [64]. Therefore, they can be modelled similar to DC motors.

In studies [59], [64], [29], [37] and [30], BLDC motor is modeled with the following equation:

$$\begin{aligned}\dot{\omega}_m &= -\frac{1}{\tau}\omega_m - \frac{d}{\eta_g r^3 J}\omega_m^2 + \frac{1}{k_m \tau}u \\ \frac{1}{\tau} &= \frac{k_m^2}{RJ}\end{aligned}\tag{3.44}$$

where:

- ◇ ω_m : motor angular speed
- ◇ τ : motor time constant
- ◇ d : drag factor
- ◇ η_g : gear box efficiency
- ◇ r : gear box reduction ratio
- ◇ J : propeller inertia

- ◇ k_m : torque constant
- ◇ u : motor input voltage
- ◇ R : motor internal resistance

Equation 3.44 describes the relation between the input voltage and rotor speed. This relation adds difficulty to the control algorithm design. In some of the studies [30], [36], [59] however, corresponding dynamics are neglected. They have considered a linear relationship between rotor speed and the input voltage, which is a Pulse Width Modulation Signal (PWM) in most of the cases. This can be done if the system responds very fast to the applied input. For this thesis study, the response of the motors should be checked so that these dynamics might be neglected. A transfer function representing the rotor dynamics can be obtained, so that it can be used in the simulation models.

3.3.5 State Variables and Equations

Rotational and translational equations of motion have been obtained in Sections 3.3.2 and 3.3.3. They can also be represented in a state space form. There are 12 states for a quadrotor system, and the state vector can be defined as;

$$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}^T \quad (3.45)$$

In simplified rotational equations of motion, Euler angles and their rates are used. For translational dynamics, the orientation of quadrotor is directly defined with x , y and z . Thus, the states can be written as;

$$X = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & z & \dot{z} & x & \dot{x} & y & \dot{y} \end{bmatrix}^T \quad (3.46)$$

Control input vector should also be expressed so that the state space model of the system can be formed. Control inputs have already been defined in Equation 31, therefore control vector U can be represented as:

$$U = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix}^T \quad (3.47)$$

By gathering the control input expressions from Equations 3.21 and 3.22, the control vector U can be written as follows:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ bl & -bl & -bl & bl \\ bl & bl & -bl & -bl \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \quad (3.48)$$

Control inputs have been defined in terms of rotor speeds, however there should also be a reverse relation such that desired propeller speeds can be found corresponding to the inputs calculated by the control algorithm. The inverse relation can be written as follows:

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4b} & \frac{1}{4bl} & \frac{1}{4bl} & -\frac{1}{4d} \\ \frac{1}{4b} & -\frac{1}{4bl} & \frac{1}{4bl} & \frac{1}{4d} \\ \frac{1}{4b} & -\frac{1}{4bl} & -\frac{1}{4bl} & -\frac{1}{4d} \\ \frac{1}{4b} & \frac{1}{4bl} & -\frac{1}{4bl} & \frac{1}{4d} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \quad (3.49)$$

Now, derived equations of motion can be revisited. For the rotational dynamics, the simplified version in Equation 3.34 should be used so that obtained state space model is simple enough to work on control law design.

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})\dot{\theta}\dot{\psi}/I_{xx} \\ (I_{zz} - I_{xx})\dot{\phi}\dot{\psi}/I_{yy} \\ (I_{xx} - I_{yy})\dot{\phi}\dot{\theta}/I_{zz} \end{bmatrix} + \begin{bmatrix} U_2/I_{xx} \\ U_3/I_{yy} \\ U_4/I_{zz} \end{bmatrix} + \begin{bmatrix} -J_r\dot{\theta}\Omega/I_{xx} \\ J_r\dot{\phi}\Omega/I_{yy} \\ 0 \end{bmatrix} \quad (3.34)$$

Separate equations can be written down from each row of Equation 3.34 as follows:

$$\begin{aligned} \ddot{\phi} &= \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\psi}\dot{\theta} + \frac{U_2}{I_{xx}} - \frac{J_r}{I_{xx}} \dot{\theta}\Omega \\ \ddot{\theta} &= \frac{I_{zz} - I_{xx}}{I_{yy}} \dot{\phi}\dot{\psi} + \frac{U_3}{I_{yy}} - \frac{J_r}{I_{yy}} \dot{\phi}\Omega \\ \ddot{\psi} &= \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\phi}\dot{\theta} + \frac{U_4}{I_{zz}} \end{aligned} \quad (3.50)$$

Equation 3.50 can be further simplified by defining new variables as follows:

$$\begin{aligned} \ddot{\phi} &= a_1 x_4 x_6 + b_1 U_2 - a_2 x_4 \Omega \\ \ddot{\theta} &= a_3 x_2 x_6 + b_2 U_3 - a_4 x_2 \Omega \\ \ddot{\psi} &= a_5 x_2 x_4 + b_3 U_4 \end{aligned} \quad (3.51)$$

where:

$$a_1 = \frac{I_{yy} - I_{zz}}{I_{xx}}, a_2 = \frac{J_r}{I_{xx}}, a_3 = \frac{I_{zz} - I_{xx}}{I_{yy}}, a_4 = \frac{J_r}{I_{yy}}, a_5 = \frac{I_{xx} - I_{yy}}{I_{zz}}$$

$$b_1 = \frac{1}{I_{xx}}, b_2 = \frac{1}{I_{yy}}, b_3 = \frac{1}{I_{zz}}$$

For the translational dynamics, Equation 3.43 can be revisited for state space conversion.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R_{EB} \begin{bmatrix} 0 \\ 0 \\ -U_1/m \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad (3.43)$$

where:

$$R_{EB} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \quad (3.10)$$

Separate equations can be written down from each row of Equation 3.43 as follows:

$$\begin{aligned} \ddot{x} &= -\frac{U_1}{m} \left(\cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi) \right) \\ \ddot{y} &= -\frac{U_1}{m} \left(\cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi) \right) \\ \ddot{z} &= g - \frac{U_1}{m} \left(\cos(\phi) \cos(\theta) \right) \end{aligned} \quad (3.52)$$

Equation 3.52 can be further simplified by defining new variables as follows:

$$\begin{aligned} \ddot{x} &= -\frac{U_1}{m} \left(\cos(x_1) \sin(x_3) \cos(x_5) + \sin(x_1) \sin(x_5) \right) \\ \ddot{y} &= -\frac{U_1}{m} \left(\cos(x_1) \sin(x_5) \sin(x_3) - \sin(x_1) \cos(x_5) \right) \\ \ddot{z} &= g - \frac{U_1}{m} \left(\cos(x_1) \cos(x_3) \right) \end{aligned} \quad (3.53)$$

As a remark for Equation 3.53, u_x and u_y terms can also be defined to yield the following:

$$\begin{aligned} u_x &= \cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi) \\ u_y &= \cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi) \end{aligned} \quad (3.54)$$

Finally, the complete state space representation of the quadrotor system can be represented as follows:

$$\dot{x} = f(x, U) = \begin{bmatrix} x_2 \\ a_1 x_4 x_6 + b_1 U_2 - a_2 x_4 \Omega \\ x_4 \\ a_3 x_2 x_6 + b_2 U_3 - a_4 x_2 \Omega \\ x_6 \\ a_5 x_2 x_4 + b_3 U_4 \\ x_8 \\ g - \frac{U_1}{m} \left(\cos(x_1) \cos(x_3) \right) \\ x_{10} \\ -\frac{U_1}{m} \left(\cos(x_1) \sin(x_3) \cos(x_5) + \sin(x_1) \sin(x_5) \right) \\ x_{12} \\ -\frac{U_1}{m} \left(\cos(x_1) \sin(x_5) \sin(x_3) - \sin(x_1) \cos(x_5) \right) \end{bmatrix} \quad (3.55)$$

CHAPTER 4

SYSTEM HARDWARE AND SOFTWARE

In this chapter, the software elements used for the implementation of the control system and the hardware elements of the quadrotor platform are explained in detail. Quadrotor platform is a mechatronic system in which multiple electronic devices work in harmony. The foundation of the system is based on the electrical energy and the movement of the system is ensured by the rotation of the electric motors. System uses different sensors for motion control and has a central control board. There are different products and software on the market in terms of actuator, sensor and controller unit. This chapter describes the hardware and software preferences for the platform created within the scope of this study.

4.1 Hardware Overview

The quadrotor in Figure 4.1 was created for this thesis study, and has been formed by the following hardware elements:

- Quadrotor Frame
- Propulsion System
- Pixhawk
- Raspberry Pi
- RC Transmitter and Receiver
- Lidar

- Lipo Battery and DC-DC Converters



Figure 4.1: Quadrotor platform used in this study

The hardware of the Quadrotor system has been selected according to the needs of this study. At the beginning of this work, it was aimed to design a platform that is able to fly at least 15min with a maximum payload of 500g. In this respect, the compatibility of the quadrotor components, especially the propulsion system, with other components is very important. Each component integrated in the system changes the weight and inertia, which results in both reduced payload and flight time. In order to achieve the expected performance, weight-flight time optimization is required so that the program called eCalc [6] was used. This program gets propulsion system, battery, frame size, and weight as inputs and produce flight time, power consumption and etc. as the output. As a result of the optimizations made with this program, a large part of the equipment to be used in our platform was decided. Entry page eCalc is shown in Figure 4.2.

At this point, explaining the selected equipments and their properties will provide an understanding of the effects on the results of this study.

4.1.1 Quadrotor Frame

Quadrotor frame consists of 4 arms, a plate which acts as a connector and also serves as a power distribution board, and a variety of connecting elements. It is possible to select two configurations, cross or plus shaped. As explained in Section 3.1, our

General	Model Weight: 850 g incl. Drive 30 oz	# of Rotors: 4 flat	Frame Size: 400 mm 15.75 inch	FCU Tilt Limit: no limit	Field Elevation: 500 m ASL 1640 ft ASL	Air Temperature: 25 °C 77 °F	Pressure (QNH): 1013 hPa 29.91 inHg	
Battery Cell	Type (Cont. / max. C) - charge state: select... - normal	Configuration: 3 S 1 P	Cell Capacity: mAh mAh total	max. discharge: 85%	Resistance: Ohm	Voltage: V	C-Rate: C cont. C max.	Weight: g oz
Controller	Type: select...	Current: A cont. A max.	Resistance: Ohm	Weight: g oz	Accessories Current drain: 0 A			Weight: g oz
Motor	Manufacturer - Type (Kv) - Cooling: select... - select... good	KV (w/o torque): rpm/V Prop-Kv-Wizard	no-load Current: A @ V	Limit (up to 15s): W	Resistance: Ohm	Case Length: mm inch	# mag. Poles:	Weight: g oz
Propeller	Type - yoke twist: select... - 0°	Diameter: 10 inch 254 mm	Pitch: 4.7 inch 119 mm	# Blades: 2	PConst / TConst: 1.2 / 1.0	Gear Ratio: 1 : 1	calculate	

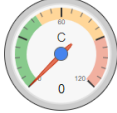





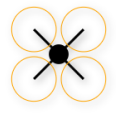
						
Load:	Hover Flight Time:	electric Power:	est. Temperature:	Thrust-Weight:	specific Thrust:	Configuration

Figure 4.2: eCalc - RC Calculator [6]

quadrotor is configured as cross shaped. The most important requirements are the sturdiness and having a light-weight frame. Considering the electronic boards and batteries to be placed on the system, a medium size frame had to be chosen. Initially, the F450 frame that is used by many researchers was considered, but it was realized that the hardware elements would remain very close to the propellers, and therefore the larger frame S500 was preferred. The centre plate of this frame, seen in Figure 4.3, offers more space for hardware placement and larger motor and propeller configurations might be preferred. In this way, usage of a propulsion system that can provide a reasonable flight time is possible. In addition, some 3d printed parts are attached to the frame for the placement of electronic boards and a custom frame shape has been obtained.



Figure 4.3: S500 Quadrotor Frame [7]

4.1.2 Propulsion System

The most important equipment group of the Quadrotor system is the propulsion system. The flight performance varies hugely depending on the selected brushless motor, electronic speed controller (ESC) and propeller trio. T-motor [8] products were preferred for the quadrotor platform used in this thesis. The propulsion system, created with optimizations from the eCalc program, consists of the following elements:

- T-motor Navigator Type MN3110 KV700 brushless motor (Figure 4.4)
- T-motor 40A 600Hz ESC (Figure 4.5)
- T-motor 12*4 Carbon Fiber Propeller

The motors used on the quadrotor are placed on each arm of the frame so that each neighbouring motor rotates in the opposite direction to the other. This is to cancel the net torque around the centre of gravity. Depending on the orientation of the quadrotor around its axes, each motor is driven with different speed commands. In this way, a balanced flight is aimed to be achieved.



Figure 4.4: Motor and propeller pair [8]

Brushless Motors are classified according to their stator size. There are also different 'KV' marked motors of the same size. Here, KV means the constant velocity of a motor. It is measured by the number of revolutions per minute (RPM) when a motor turns when 1V (one volt) is applied with no load attached to that motor [65]. With this stator size and your choice of KV, one can choose how fast the motor will turn depending on the applied voltage. Of course, there will be different measurement readings when the motor is loaded. When the propeller is connected, it will generate

different thrust forces depending on the propeller properties. For this reason, companies generally publish the performance values of their motors attached with different sets of propellers.

Propellers, on the other hand, usually have 2 descriptive numbers. The first one specifies the diameter of the propeller, while the second denotes the pitch value. Together, they form the propeller characteristics. Using eCalc, both motor and propeller have been selected at the same time. With this carbon fibre propeller, the motor is able to produce a thrust of 890g at full throttle with 3S Lipo battery. In this case, 4 engines can reach up to a thrust value of 3.5kg. For a quadrotor aimed as 1.6kg, the motors will rotate with almost half throttle in the case of hover condition. The important thing here is to keep the throttle value in the hover situation as low as possible, since the speed of the quadrotor motors are limited to the full throttle and reaching to that point will bring the loss of control due to speed saturation.



Figure 4.5: T-motor ESC [8]

The last remaining propulsion system component is the electronic speed controller (ESC). These ESCs are devices that control the rotational speed of the motor. Quadrotor autopilot board sends commands to each motor depending on the stability of the system. These commands are transmitted in the form of Pulse Width Modulation (PWM) signals. ESC units get these pulses as input, supply the necessary AC Voltage to the motors and allow them to rotate at the targeted speed. Today's modern ESCs are often more sensitive with back emf measurement. The ESC used in this study can provide continuous current of 40A, and has an update rate of 600Hz. One of the most vital parameters for the quadrotor system is the response time of the motors. In order to get a good performance and to increase the actuator bandwidth as much as possible, low response time values should be obtained. That's why, T-motor products have been preferred in this study.

4.1.3 Pixhawk

The main focus of this study is the orientation control of a quadrotor. To do this, there is a need for a microcontroller unit and various sensors for orientation data. There are a lot of products on the market with different features and prices. Our goal was to use a board that accommodates several interfaces and supports different software platforms. That's why Pixhawk autopilot board, in Figure 4.6, has been chosen as the main element of this platform.



Figure 4.6: Pixhawk [9]

Pixhawk is an open-hardware autopilot board that is used by researchers, hobby communities and developers. This board is based on 32-bit ARM Cortex M4 microcontroller equipped with accelerometer, gyroscope, compass and etc. In addition to having internal sensors, UART serial port, I2C, CAN and S.BUS interfaces are supported. Multi-functional hardware structure enables communication with different sensors and boards. In this study, the sensor data on Pixhawk is used as the primary control source. Besides, Raspberry Pi board and Lidar connections are implemented through serial interfaces and system orientation and position control is achieved.

This board supports various flight stacks such as Px4 and ArduPilot. Since it is a Linux based board, it also offers the possibility to develop our own flight stack. One of the main reasons why this board has been preferred is to be able to run Simulink models onboard. A Simulink Support Package is available for Pixhawk [15]. In this thesis, the control algorithms designed in Simulink have been embedded on Pixhawk autopilot by using provided Support Package. Details about this process will be ex-

plained in Section 4.2.

4.1.4 Raspberry Pi

Another focus of this study is to achieve position control with image feedback. The Pixhawk board is an accomplished board for orientation control, but it does not have the capability of image processing. It is only possible to control the speed of the quadrotor using the optical flow kit that can be attached through Pixhawk serial port. At this point, a second hardware is required to gather position information. For a precise position control, a reasonable image processing algorithm is necessary. Currently, there are many open source algorithms and libraries created both in Python and C++. However, a person may need a small sized computer to use them onboard. In this context, the Raspberry Pi 3 Model B, in Figure 4.7, was the choice for image processing root.

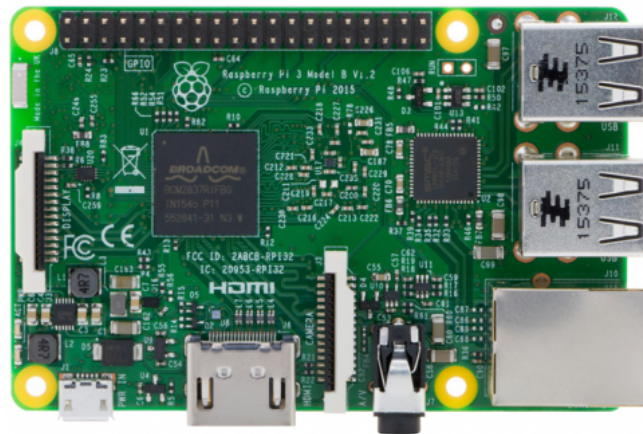


Figure 4.7: Raspberry Pi 3 Model B [10]

Raspberry Pi has a very fast microprocessor that you can use with its own camera or an external one. Both Linux and Windows operating systems are possible to use with this powerful board. Within the scope of this study, the camera of the Raspberry Pi, given in Figure 4.8, has been used. Just like Pixhawk, there is a Simulink Support Package for Raspberry Pi hardware [16], which allows you to benefit from the simplicity of the Simulink environment. Raspberry Pi also provides serial communication through its GPIO pins, which indeed enabled contact between boards.

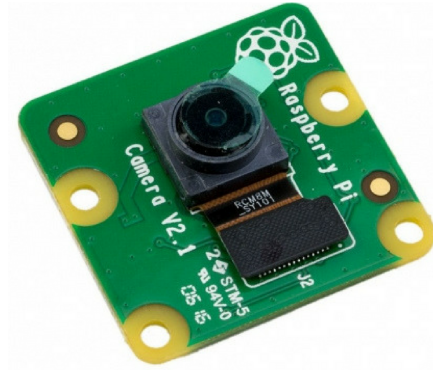


Figure 4.8: Raspberry Pi Camera [11]

4.1.5 RC Transmitter and Receiver

Radio Control (RC) systems are devices that are used to control a remote device, and are based on radio signals. The reference signals are transmitted and converted into radio signals via the transmitter. The receiver on the other side converts these signals into meaningful data and transmits them to the relevant device. In this study, Taranis X9d Plus Transmitter and Receiver pair, given in Figure 4.9, which are used frequently by hobby community have been preferred.



Figure 4.9: Taranis X9d Plus Transmitter and Receiver [12]

The quadrotor system built within the scope of this study has the ability to perform autonomous hover and path following with directives of orientation and position controllers. However, pilot intervention may be necessary in some cases due to security considerations. In addition, some commands like "integral reset" and "mode change"

are transmitted via the transmitter. Taranis RC transmitter is an advanced device with its knobs and switches. In this research, events such as starting the system, stopping the motor movement, initiating data logging and activating hover / trajectory tracking have been managed by switches located on Taranis. In addition, it is possible to update the gains for particular controller in the air.

4.1.6 Lidar

In order to check the height of the quadrotor from the ground, it needs a sensor that can measure this height properly. The Pixhawk autopilot has an embedded barometer. However, this sensor output was not used as a primary source in this study, since it can vary directly with air pressure and temperature, and thus the target altitude may alter depending on the conditions. Another way of obtaining height data is by means of image processing. In this study, position of the quadrotor is obtained by identifying the Aruco Marker left on the ground. Since there is no reference information for height control at the point where the marker exits the camera frame, it was considered unreasonable and it was decided to use an external device. For this reason, a small and lightweight lidar, which is the Tf Mini Lidar given in Figure 4.10, has been used as the altitude source. This laser range sensor has a sampling accuracy of $\pm 1\text{cm}$. It is not possible to perform an altitude control with millimetre precision, but quadrotor may stay within a few centimetres of band.



Figure 4.10: Tf Mini Lidar Range Sensor [13]

4.1.7 Lipo Battery and DC-DC Converters

Another product that is important as the propulsion system for the Quadrotor is the battery. Lipo batteries are commonly used as power supply in UAV applications. What makes them preferable is that they can provide a constant voltage over a long period of time. They can also provide high current values. Although the size of the battery means more flight time, large batteries are not preferred because of their weight.



Figure 4.11: Gens Ace 11.1V 3S 4000mAh Lipo Battery [14]

Formed platform can be used with 3S or 4S Lipo when the properties of Pixhawk, power module and motor are considered. For this system, a 3S (11.1V) 4000mAh Gens Ace product Lipo, in Figure 4.11 was preferred with an eCalc analysis. Fully charged Lipo can supply 12.6V, while it gives 9.6V close to discharge. This voltage difference cause the motors slow down and a noticeable amount of thrust is lost towards the end of the flight. Therefore, voltage variation is also included in the control algorithm.

The platform also includes a power brick to feed the Pixhawk autopilot module and a DC-DC converter to feed the Raspberry Pi board. The lipo is directly connected to the power distribution board of the quadrotor frame. Other hardware, especially ESCs, are supplied from this board.

4.1.8 System Layout

The placement of the equipment on the quadrotor frame was made considering the centre of gravity and inertia. To minimize the energy consumed by the system and improve control performance, the centre of gravity needs to be in the frame origin. Otherwise, some motors have to produce more thrust, which increases energy consumption. For this reason, the Pixhawk, Raspberry Pi and the Lipo battery were placed on top of each other using 3d printed parts, which can be seen in Figure 4.1.

After the installation, the hardware was connected to each other with necessary connectors and soldering. The Pixhawk board is powered by a power brick, and the Raspberry Pi is powered by a DC-DC converter module. Taranis receiver and Lidar power up directly from the Pixhawk. ESCs are directly connected to Lipo battery via the power distribution board. The Pixhawk board communicates with Raspberry Pi and Lidar via the UART serial port. The receiver module, on the other hand, is connected to the Pixhawk autopilot with S.BUS. In addition, PWM signals are transferred to ESCs using Pixhawk's Main Aux ports. This component scheme and connections can be seen in Figure 4.12.

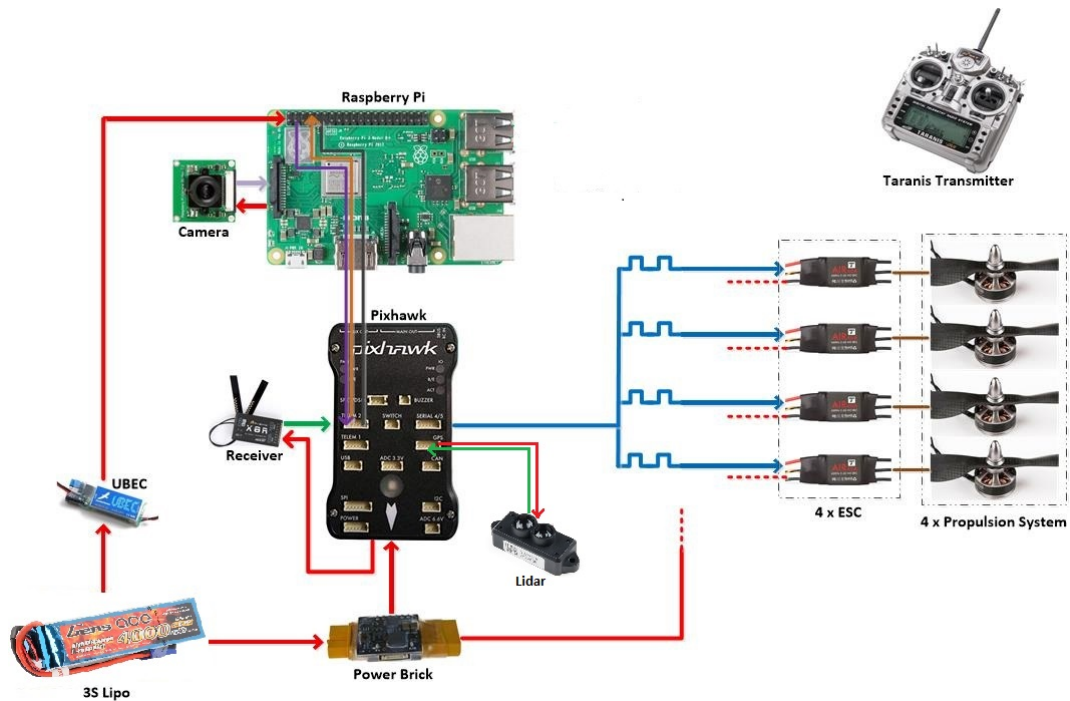


Figure 4.12: Diagram of quadrotor components

4.2 Software Overview

As mentioned in Section 4.1, Pixhawk and Raspberry Pi boards were used in this thesis study. In order to accurately evaluate the system results, it is important to understand the models running on the two boards and their communication.

Explanations can be initiated with the Pixhawk autopilot module, which is the root part of the system. The Pixhawk board is a linux based system and can work with different open source flight stacks. For example, a drone enthusiast can create a quadrotor platform, and then use a Pixhawk to fly it rapidly. All it needs is to install a flight stack like Px4 or Ardupilot. Since these are open source software platforms, any user can modify the design to reflect their idea. What important is to be aware of the software architecture.

Pixhawk software functionality is divided into many layers. There is an overall scheduler that enable each application run on different layers. Due to this threading, each application can run at different cycle periods, and communicates by a message protocol with others. At the base task layer, there are drivers and RTOS files are running. Operations such as Attitude, Position, and Trajectory control work on the upper layers. For example, if a user wants to implement a different controller, he/she only needs to change the control applications, if unless the desired inputs reach there. For more extensive changes, it is necessary to learn this threaded structure.

There is also a Simulink Support Package besides to the open source flight stacks. What this package does is to translate the model you designed in Simulink into a C++ script and implement it on a stable version of Px4. Then, one can easily transfer the built files into Pixhawk via a USB cable. Since the current controller design and system modeling is carried out in Simulink, it is a great convenience to stay in this environment. At this point, the Simulink needs access to sensor data and interfaces on the Pixhawk. For this purpose, some Simulink blocks have been created and ready to use in the Support Package.

By simply adding these blocks to the Simulink model, a user can read Euler angles, provide serial communication, read receiver data, and generate PWM outputs for the motors. By giving different priorities to the created Simulink model, model running

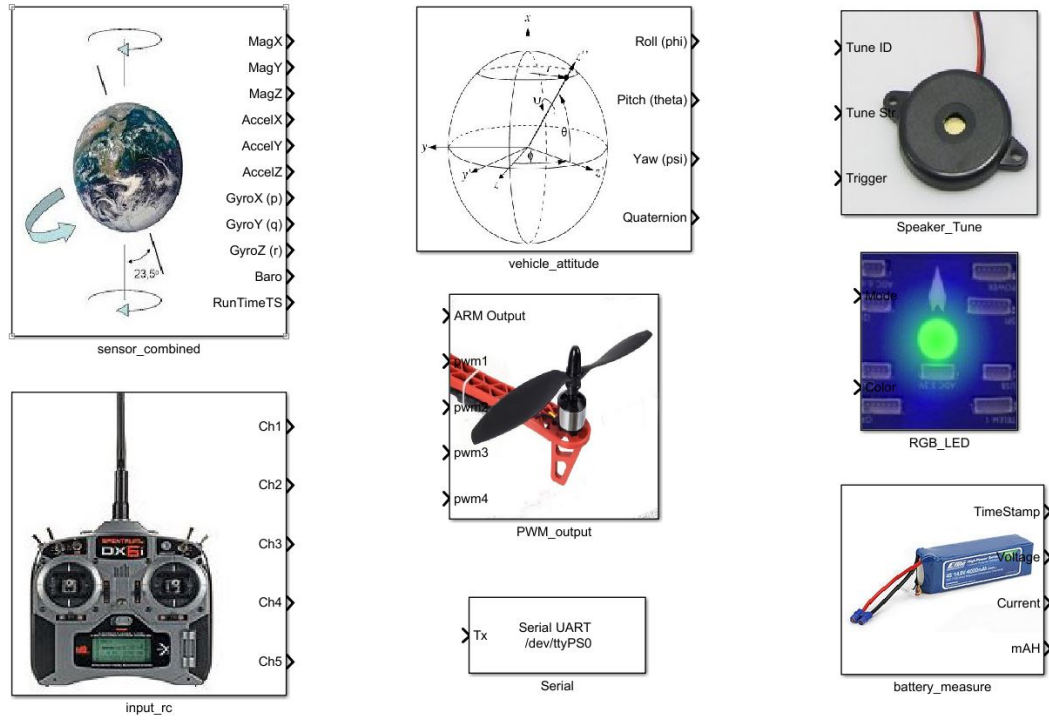


Figure 4.13: Simulink blocks provided by the Support Package [15]

order in Pixhawk thread can be managed, and it can be run at a sample time determined by the user. In Figure 4.13, the Simulink blocks for the model are given, and in Figure 4.14, there is the Simulink model created by using these blocks within the aim of this study.

The created model works at a sample time of 100Hz on Pixhawk board. Euler angles and body rates are read from the sensor blocks. The commands from the transmitter such as system start, rotate/stop motors and algorithm mode change are fed from the receiver block into the subsystems of algorithm. In addition, the instantaneous voltage of the battery and how much current the system draws are obtained from the corresponding blocks. At the centre of the model, attitude controller takes place. There is a cascaded PID structure for pitch, roll and yaw channels. Angle reference is supplied to these controllers. Depending on the selected system mode, these values can be given from Taranis transmitter (if the pilot wants to control), but mainly gathered from the position controller. The position controller also has a cascaded structure that takes x, y and speed data as inputs. Position data is also produced in Raspberry Pi, but sent to Pixhawk via serial communication. For this reason, data is transmitted from

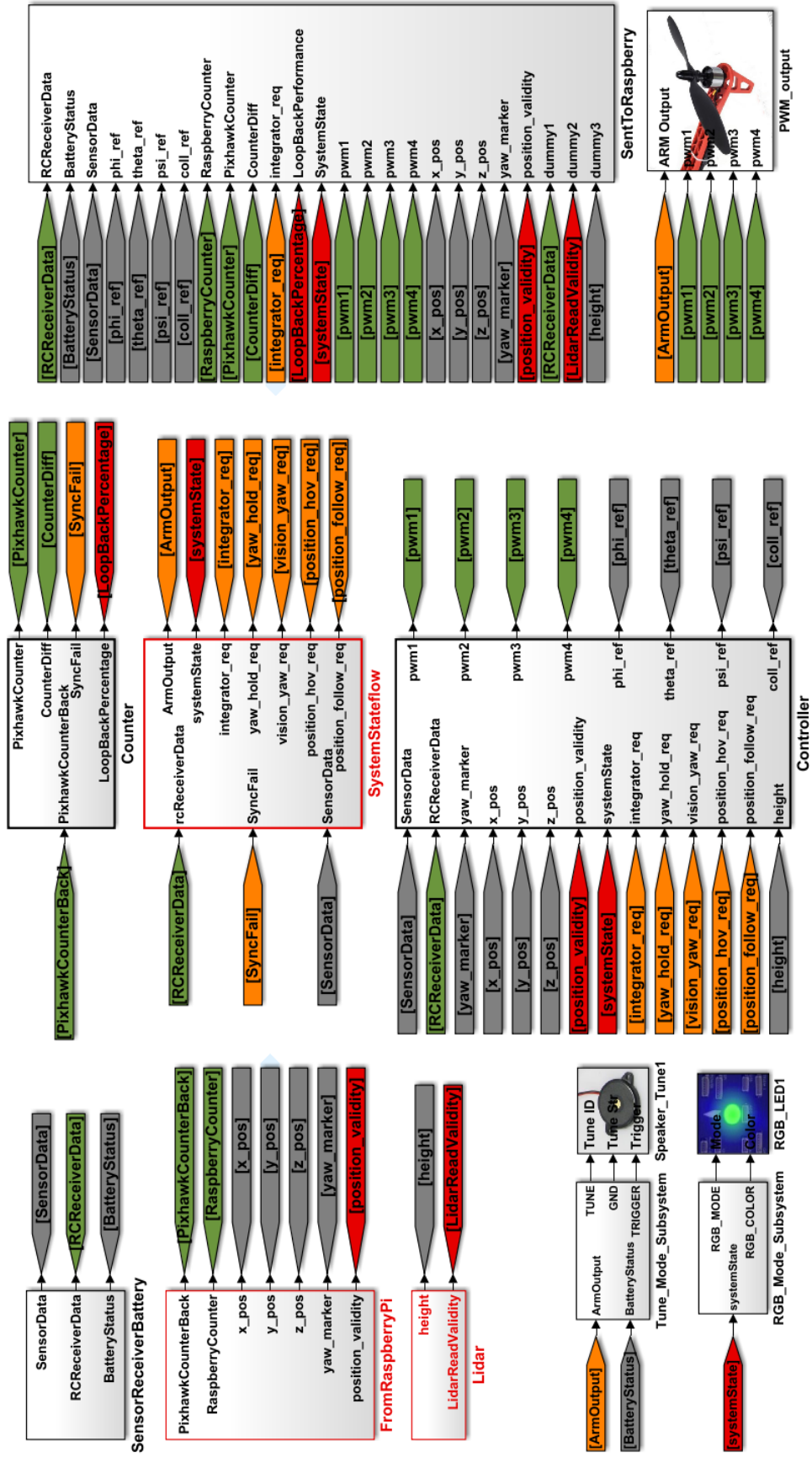


Figure 4.14: Pixhawk Simulink model created within the scope of this study

serial UART block of Simulink model. Attitude controllers convert the commands they produce into PWM, and these signals are transmitted to the motors via PWM output block. There is also a buzzer in the platform such that mode transitions, low voltage warning and etc. can be reflected to the user. Pixhawk also has the ability to save data to the SD card. However, since it is easier to get the data from Raspberry Pi, the data logging part was done on it. For this reason, all the important signals generated in Pixhawk are sent to Raspberry Pi via serial UART block.

While the Pixhawk is as explained, Raspberry Pi also has a similar Support Package [16]. A simulink model was created for the Raspberry Pi as it is easy to log data on Simulink and review them again. In this model, the data streamed from Pixhawk can be saved to the SD card in the form of a mat-file. Since Raspberry Pi has an integrated Wi-Fi module, it is possible to import the mat-files produced without a physical connection to the board. Furthermore, the Simulink model can be run externally on the Raspberry Pi, and it is possible to display the signals on the desktop while the model is running.

Serial ports were used for the communication of these two boards. However, the devices are not synchronous, and the Raspberry Pi is not a microcontroller. It uses a software clock, therefore it can be said that it does not support real time. Because of these differences, some buffer functions were written to handle the data sent by the two boards so that all data packets can be captured.

With these functions, a header of 4 bytes is added per packaged data and 2 bytes of checksum is attached to the end of the packet (Figure 4.16). In this way, other board obtain the correct data package by controlling both the header and checksum. Although Pixhawk has a buffer reset mechanism in itself, Raspberry Pi does not have this such that buffer keeps filling up when the model is not running. This causes a big delay, for which a mechanism is formed to interrupt package delivery in case of one-sided model action.

A counter mechanism has been established to check whether communication is healthy or delayed. Each board produces a counter, sends it to the other board and checks the difference between the data it sends and receives within a feedback loop. In a healthy communication, this difference is between 3 to 5 steps, which means a maximum

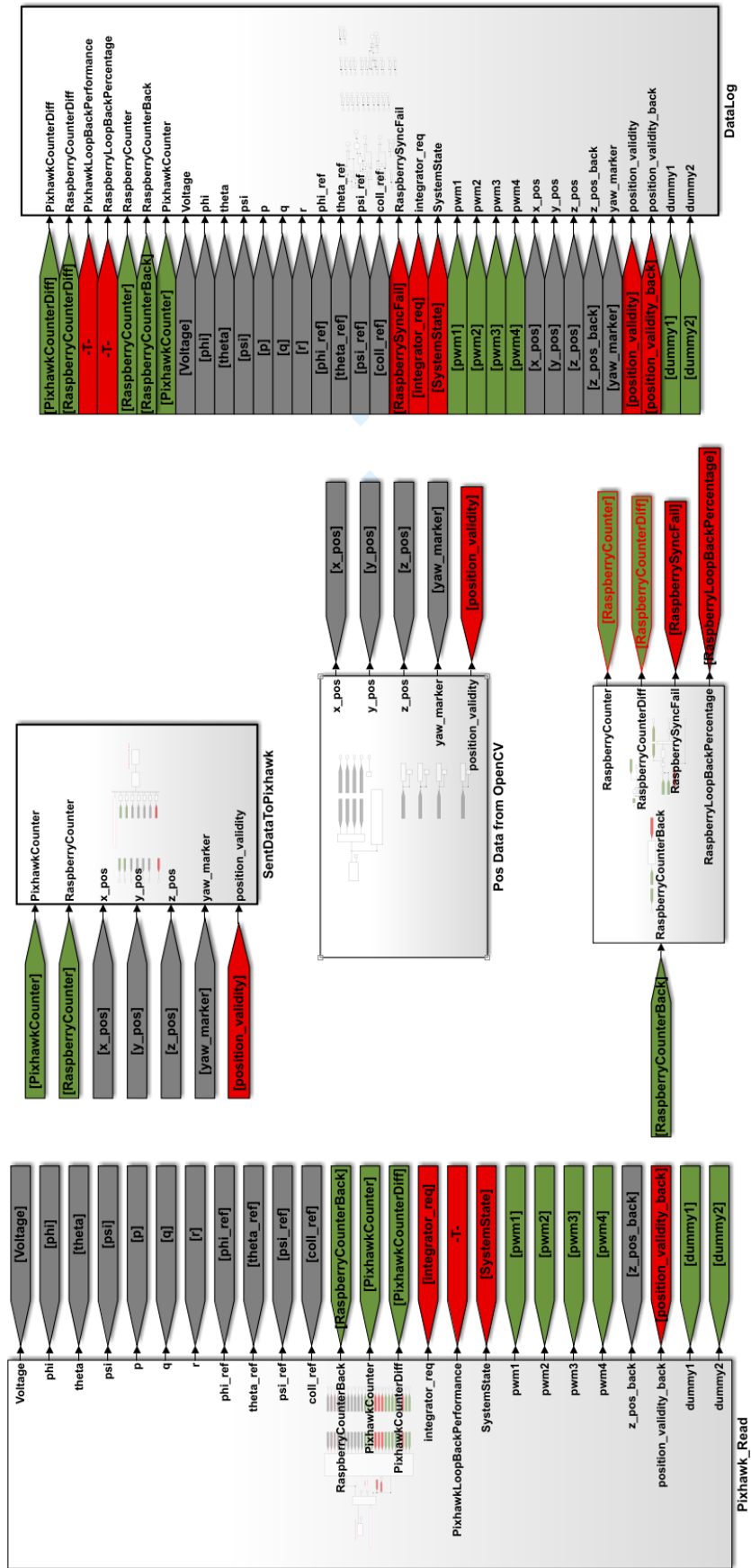
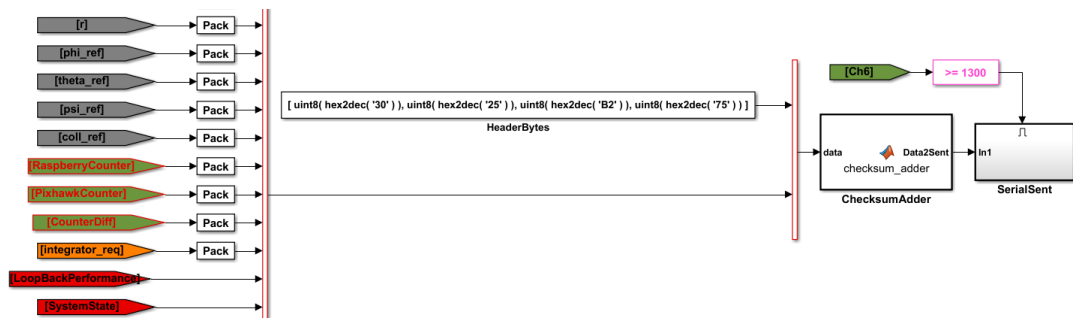


Figure 4.15: Raspberry Pi Simulink model created within the scope of this study [16]



delay of 50ms for the model running at 100Hz. Only the position data comes from Raspberry Pi, and the latency of these data is not a big problem since the system dynamics is slower in position level.

Figure 4.17: Counter Difference values for Pixhawk and Raspberry Pi boards

In some cases, data packet loss has been observed in communication, and thus a second counter mechanism has been established. This functionality increases its own counter in case of a counter value coming from the other computer within 1 second periods, and produces a value about communication health by looking at the expected counter and reached counter ratio at the end of the period. In a healthy communication, this value should be 100%, but in some cases, the Raspberry Pi model was found to be missing data packets due to non-real-time operation as seen in Figure 4.18.

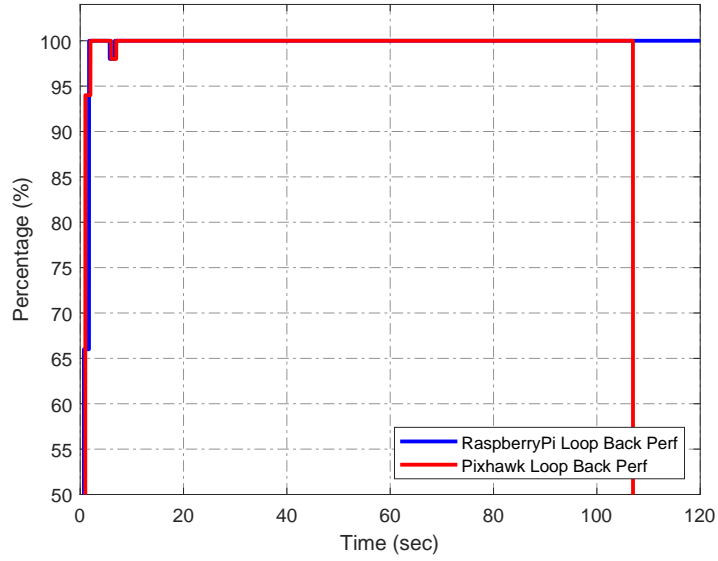


Figure 4.18: Loop Back Performance of Pixhawk and Raspberry Pi

Pixhawk board, which is used as a master, evaluates both loop back performance and communication delay, generates a sync fail command and interrupts position control with audible warning while leaving the controls to the pilot. In Figure 4.19, 'sync-fail' signal formation can be seen at the system startup.

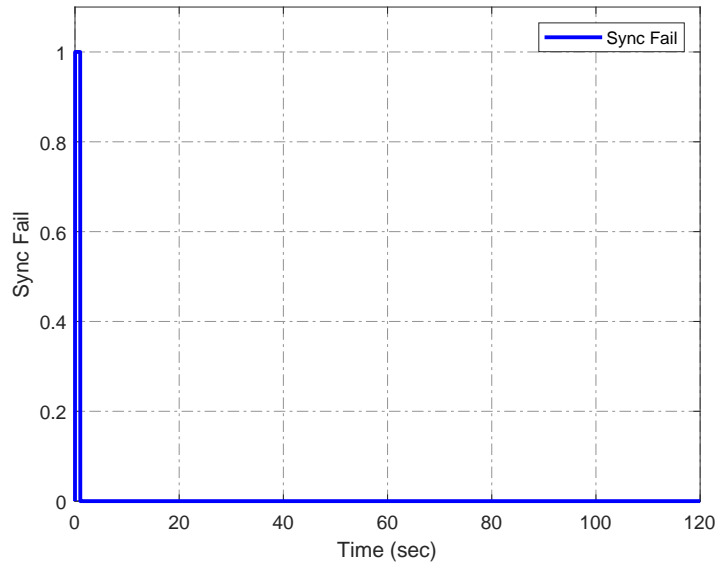


Figure 4.19: Yaw Angle vs time

It is already mentioned that Simulink is used since it is easy to record data on Raspberry Pi. However, the main software that runs on Raspberry is not the Simulink

model, but the Python script used for image processing. Although the Simulink environment enables us to use the camera, it is only possible to obtain data with simple RGB-based filtering. Since there is no advanced library structure in this environment, Python environment was preferred. There are many open source vision libraries, like OpenCV, available for both Python and C++. OpenCV is an advanced resource that includes many examples based on vision and machine learning.

In this thesis, a square shaped marker named ArucoMarker, composed of black and white coded squares representing a binary identifier, is used. Examples of these markers can be seen in Figure 4.20. A user can detect an ArucoMarker, its orientation and position by using OpenCV platform, and is able to publish this data to other sources. By the help of ArucoMarker libraries coded in Python [17], it was aimed to have a quadrotor able to detect an ArucoMarker and regulate the position with respect to that stationary marker reference.

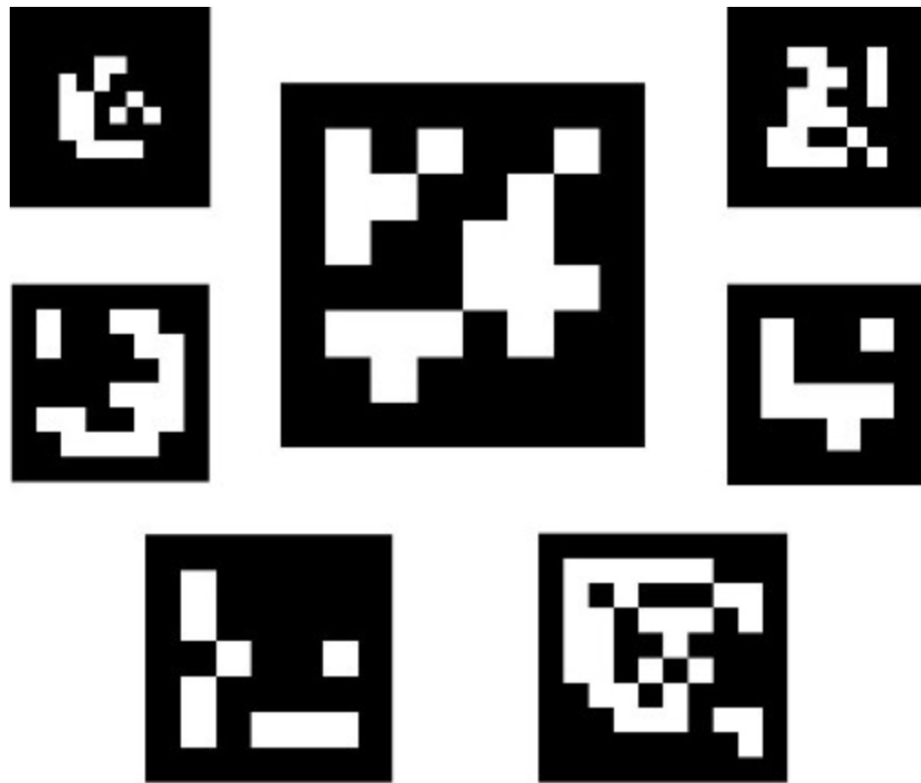


Figure 4.20: ArucoMarker examples [17]

There are different sized ArucoMarkers, but an eight by eight marker is used within the scope of this study. This marker is represented in Figure 4.21.



Figure 4.21: ArucoMarker used in this study

Using OpenCV library, marker detection is performed and position information is gathered according to the camera frame. For quadrotor position control, it is necessary to obtain the position information which is expressed in ground frame. Therefore, a conversion from camera frame to marker frame is performed. As an example for marker detection and position stream, the output of the Python script in Figure 4.22 can be visited. With this script, both position and heading information is propagated. Used vision algorithm has a sampling frequency between 20Hz and 30Hz.

The Raspberry Pi board runs both the Simulink model and the Python script simultaneously in different threads. In Figure 4.23, working programs are represented including Python script and Simulink model. The models on both Pixhawk and Raspberry Pi are set to operate in the device power-up, and can be stopped/started via the Linux terminal whenever desired. The position and heading information generated from the Python script is transmitted to the Simulink model via local UDP communication. Then, this data packet is sent to Pixhawk with Simulink serial blocks of Raspberry Pi model.

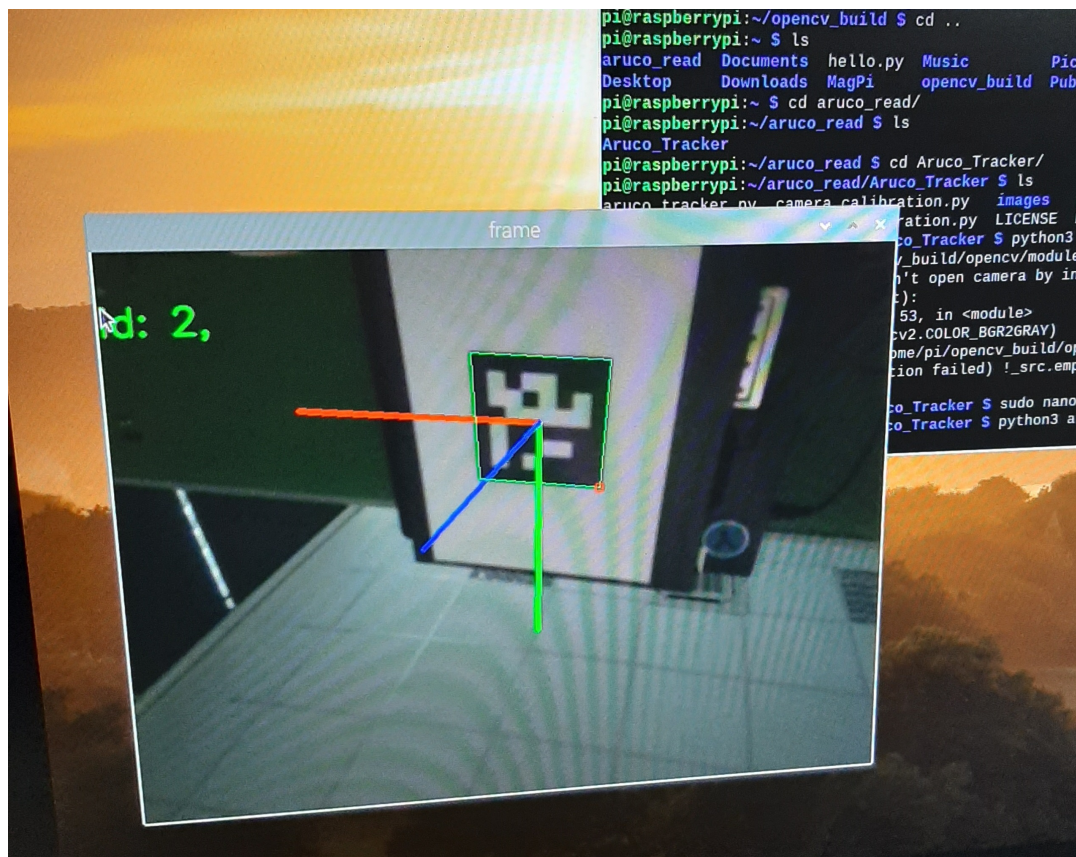


Figure 4.22: Example for ArucoMarker detection via Raspberry Pi camera

```

pi@raspberrypi: ~
top - 23:55:38 up 0 min, 2 users, load average: 1.58, 0.43, 0.15
Tasks: 138 total, 1 running, 137 sleeping, 0 stopped, 0 zombie
%Cpu(s): 44.2 us, 9.2 sy, 0.0 ni, 46.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 874.5 total, 513.0 free, 133.7 used, 227.8 buff/cache
MiB Swap: 100.0 total, 100.0 free, 0.0 used, 682.7 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 434 root        20   0  236896  63956  39620 S 208.3   7.1   0:52.99 python3
 433 root        rt   0   22076   2184   1856 S   2.3   0.2   0:00.74 raspberrypi_mod
    7 root        20   0         0        0        0 I   1.7   0.0   0:00.17 kworker/u8:0-mmial-vchiq
   64 root        1 -19         0        0        0 S   0.7   0.0   0:00.22 vchiq-slot/0
  114 root        20   0   18632   6444   5560 S   0.3   0.7   0:00.57 systemd-journal
  216 root        20   0         0        0        0 I   0.3   0.0   0:00.04 kworker/3:3-events
  777 pi          20   0   10328   2896   2536 R   0.3   0.3   0:00.11 top
    1 root        20   0   15148   7780   6288 S   0.0   0.9   0:03.15 systemd
    2 root        20   0         0        0        0 S   0.0   0.0   0:00.00 kthreadd
    3 root        0 -20         0        0        0 I   0.0   0.0   0:00.00 rcu_gp
    4 root        0 -20         0        0        0 I   0.0   0.0   0:00.00 rcu_par_gp
    5 root        20   0         0        0        0 I   0.0   0.0   0:00.00 kworker/0:0-events
    6 root        0 -20         0        0        0 I   0.0   0.0   0:00.00 kworker/0:0H-mmc_complete
    8 root        0 -20         0        0        0 I   0.0   0.0   0:00.00 mm_percpu_wq
    9 root        20   0         0        0        0 S   0.0   0.0   0:00.02 ksoftirqd/0
   10 root        20   0         0        0        0 I   0.0   0.0   0:00.05 rcu_sched
   11 root        20   0         0        0        0 I   0.0   0.0   0:00.00 rcu_bh
   12 root        rt   0         0        0        0 S   0.0   0.0   0:00.02 migration/0

```

Figure 4.23: List of programs or threads that are currently operated by Linux Kernel of Raspberry Pi

Explained software architecture and communication interfaces described in this section are also shown in the diagram of figure 4.24.

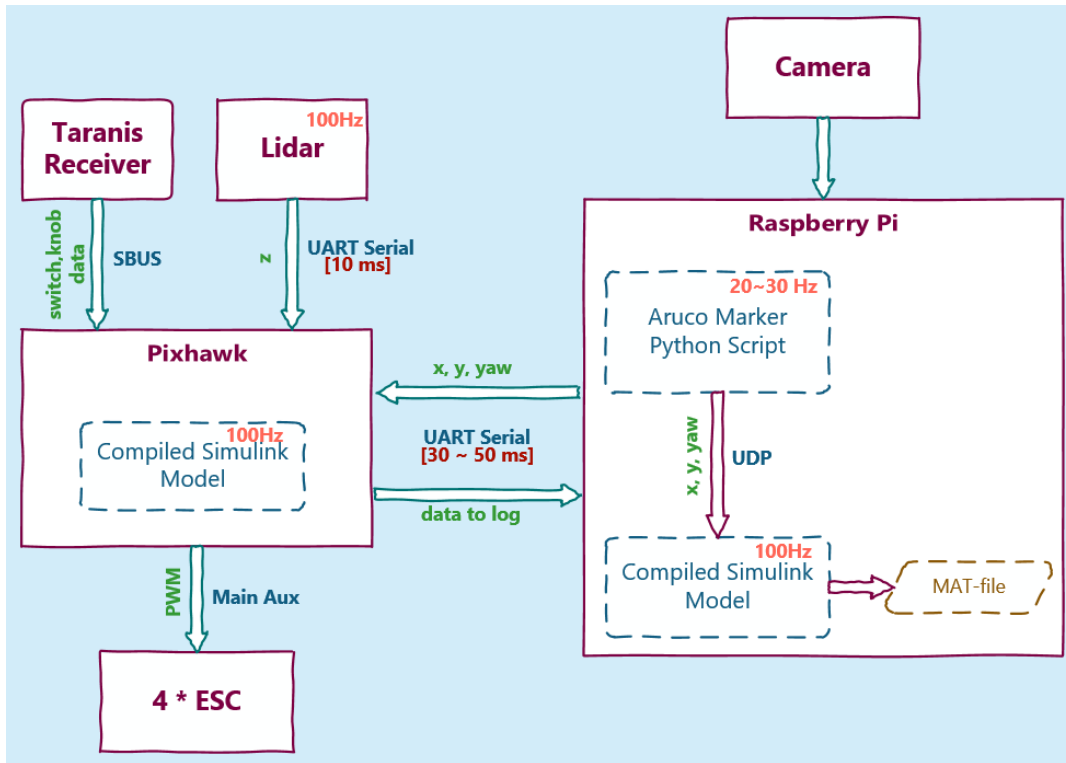


Figure 4.24: Diagram of software architecture

4.3 Physical Parameters of the Quadrotor

To be able prepare a controller for a quadrotor system, a person needs a mathematical model that reflects the system dynamical behavior. Closer the model is to reality, the better the performance of the designed controller on the quadrotor. In addition, the created algorithm must use basic parameters, such as motor characteristics, to produce logical outputs. For this reason, some of the basic quadrotor parameters should be properly determined and reflected in the mathematical model and control algorithm.

4.3.1 Mass Moment of Inertia

The Quadrotor system has a relatively simple dynamics compared to other aircrafts. The most dominant element of this dynamic behaviour is the mass moment of inertia. The most reasonable way to determine the inertias around principal axes is to construct the CAD model of the quadrotor, and obtain the values from here. However, this solution requires detailed modeling, where too many cables and connectors exist, and requires a big effort. For this reason, experimental measurement method was preferred. In this method, called Bifilar Pendulum Theory, quadrotor system is suspended with parallel ropes in order to form oscillations around the axis to be measured [66], [18]. In Figure 4.25, an example pendulum setup can be seen. The oscillation period of the suspended object varies depending on the rope length and the distance between the ropes.

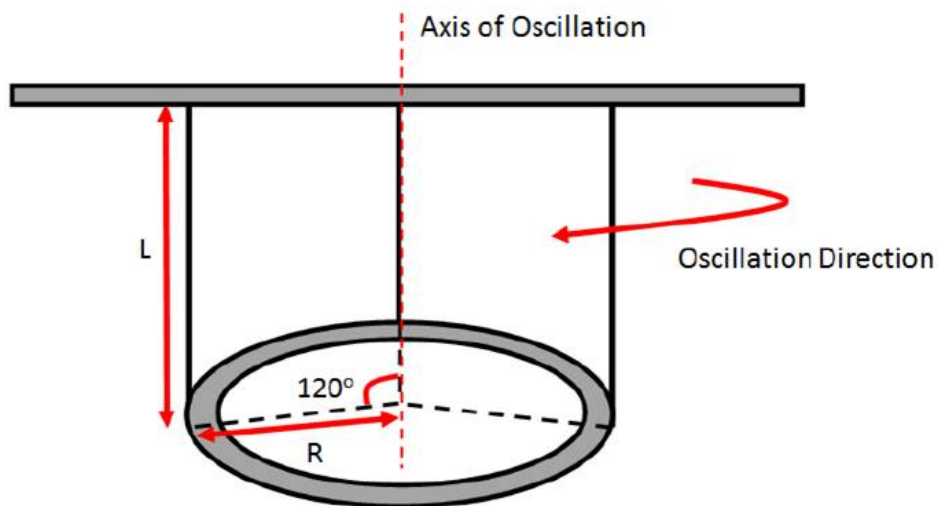


Figure 4.25: Setup for Bifilar Pendulum Theory [18]

The period of oscillation is noted after providing random swings with a setup of known rope length and rotation radius, and a formula is used to calculate the moment of inertia around that axis. Using this method, the inertia around each axis can be experimentally calculated.

Quadrotor mass moment of inertia around its principal axes can be calculated using

Equation 4.1:

$$I_{xx,yy,zz} = \frac{mgr^2 T_{x,y,z}^2}{4\pi^2 l} \quad (4.1)$$

where:

- ◇ $I_{xx,yy,zz}$: mass moment of inertia of quadrotor around x/y/z axis (kgm^2)
- ◇ $T_{x,y,z}$: period of oscillation (sec)
- ◇ m : mass of quadrotor (kg)
- ◇ g : acceleration of gravity (m/s^2)
- ◇ r : radius of rotation (m)
- ◇ l : length of suspension ropes (m)

The quadrotor platform used in this thesis was also suspended from 3 axes separately and random oscillations were provided. Figure 4.26 shows the oscillations formed in yaw angle when the quadrotor is suspended from the z-axis.

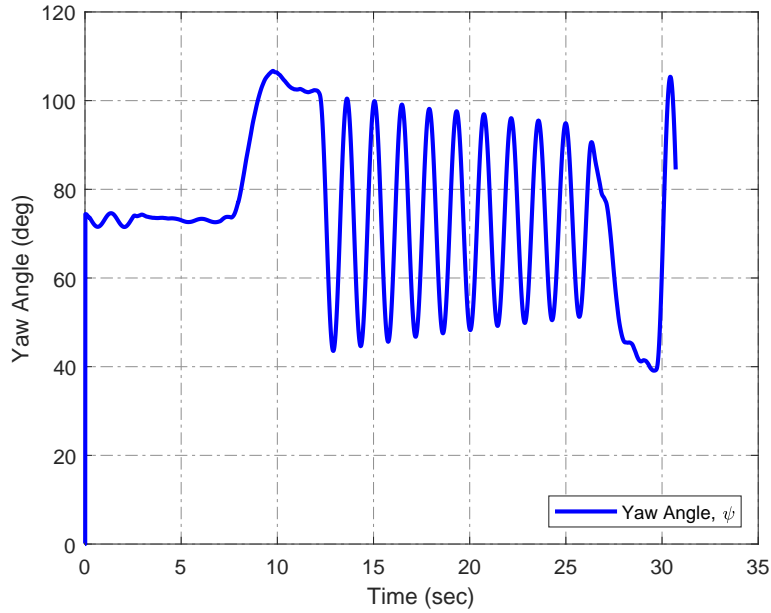


Figure 4.26: Yaw angle response to given oscillations in pendulum setup

It is not easy to give this oscillation from only one axis, and small swings in pitch and roll dynamics is also observed as seen in Figure 4.27.

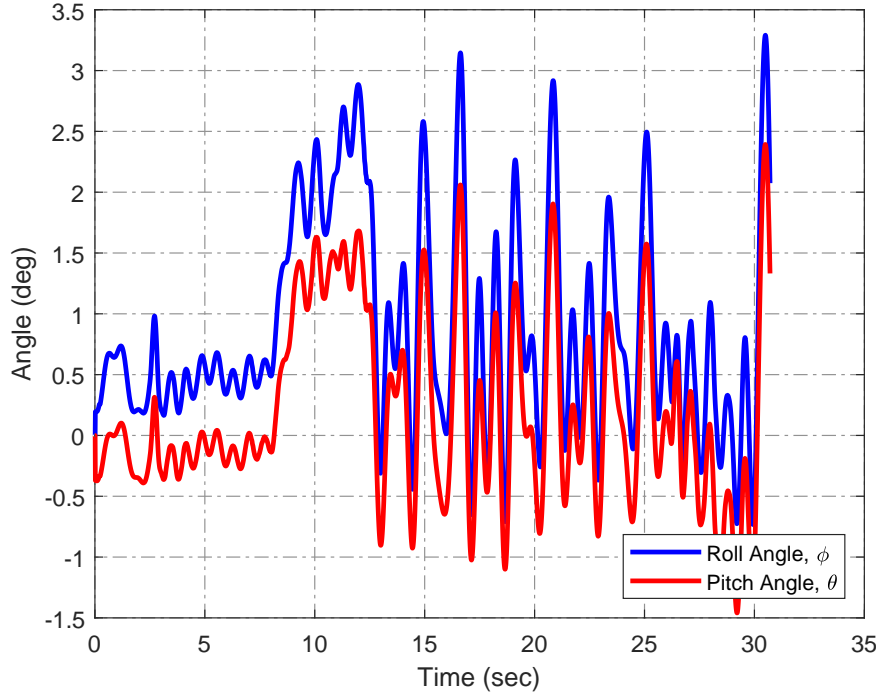


Figure 4.27: Roll and Pitch Angle vs time

After a few trials, the mass moment of inertia values were found as; $I_{xx} = 0.186 \text{kgm}^2$, $I_{yy} = 0.235 \text{kgm}^2$ and $I_{zz} = 0.112 \text{kgm}^2$.

4.3.2 Propulsion System Parameters

The quadrotor control algorithm generates thrust and moment commands for each principal axis to stabilize the platform. However, the only communication channel of the autopilot module with motors is PWM signals. Therefore, the PWM values corresponding to these signals must be found in order for the motor to reach the desired thrust values in the generation of these force and moments. As explained in section 4.1.2, the motor speed varies depending on the voltage applied. Therefore, Lipo voltage must also be taken into account when determining the PWM-Thrust relationship, since the battery voltage decreases along the flight envelope. In order to get the propulsion system properties, a thrust test bench was prepared as seen in Figure 4.28.

In this setup, varying PWM signals were given at different voltage values, and the motor speed, current drawn by the system and the produced thrust values were measured.



Figure 4.28: Thrust measurement setup for quadrotor motor-propeller pair

In Table 4.1 and 4.2, test results under 12.6V and 11.4V Lipo voltage are shown. In order to perform healthy experiments, a DC power supply was used instead of Lipo battery, and PWM signal was not increased beyond 1600 due to 5A current limit of the DC source.

Table 4.1: Motor Parameter Identification Test for 100% Battery

PWM	Voltage (V)	Speed (RPM)	Current (A)	Thrust (N)
1150	12.6	1282	0.45	0.38
1200	12.6	1565	0.56	0.59
1300	12.6	2211	0.94	1.21
1400	12.6	2870	1.66	2.07
1500	12.6	3580	2.82	3.26
1600	12.6	4192	4.35	4.55

Table 4.2: Motor Parameter Identification Test for 60% Battery

PWM	Voltage (V)	Speed (RPM)	Current (A)	Thrust (N)
1150	11.4	1157	0.44	0.33
1200	11.4	1437	0.53	0.50
1300	11.4	1988	0.85	0.99
1400	11.4	2603	1.44	1.70
1500	11.4	3275	2.44	2.70
1600	11.4	3858	3.70	3.83

For these results, many graphs, showing the relationship of different parameters, can be plotted. As shown in Figure 4.29, higher motor speed and thrust values are obtained for the same PWM signal under higher voltage values.

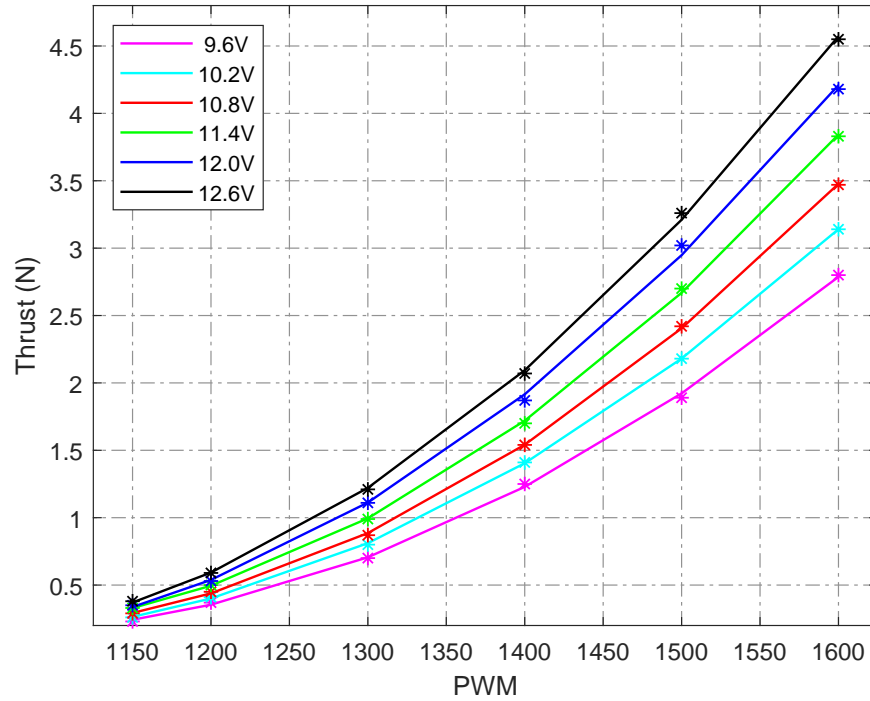


Figure 4.29: Lipo voltage and thrust relation

In Chapter 3, the relationship between forces and moments causing the movement of the quadrotor and the motor speed is expressed by Equation 3.48, which is also

presented below. As it can be seen here, the forces and moments are directly related to the square of the motor speeds. In this equation, parameters b (named 'thrust coefficient') and d (named 'torque coefficient') must be found, and the controller outputs should be subjected to the solution of this equation such that desired motor speeds shall be obtained.

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ bl & -bl & -bl & bl \\ bl & bl & -bl & -bl \\ -d & d & -d & d \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \quad (3.48)$$

Figures 4.30 and 4.31 show the variation of motor speed square with the generated thrust and torque. As it can be seen here, there is a linear relationship between them and the parameters b and d are found to be: $b = 2.368 * 10^{-5} N s^2$ and $d = 4.553 * 10^{-7} N m s^2$.

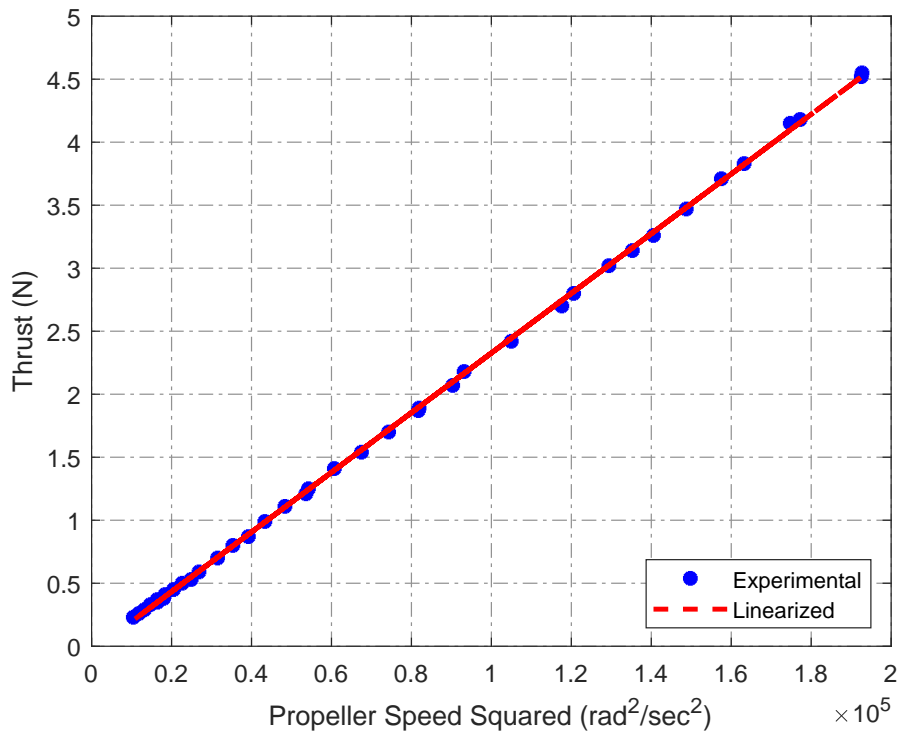


Figure 4.30: Relation of thrust and the square of propeller speed

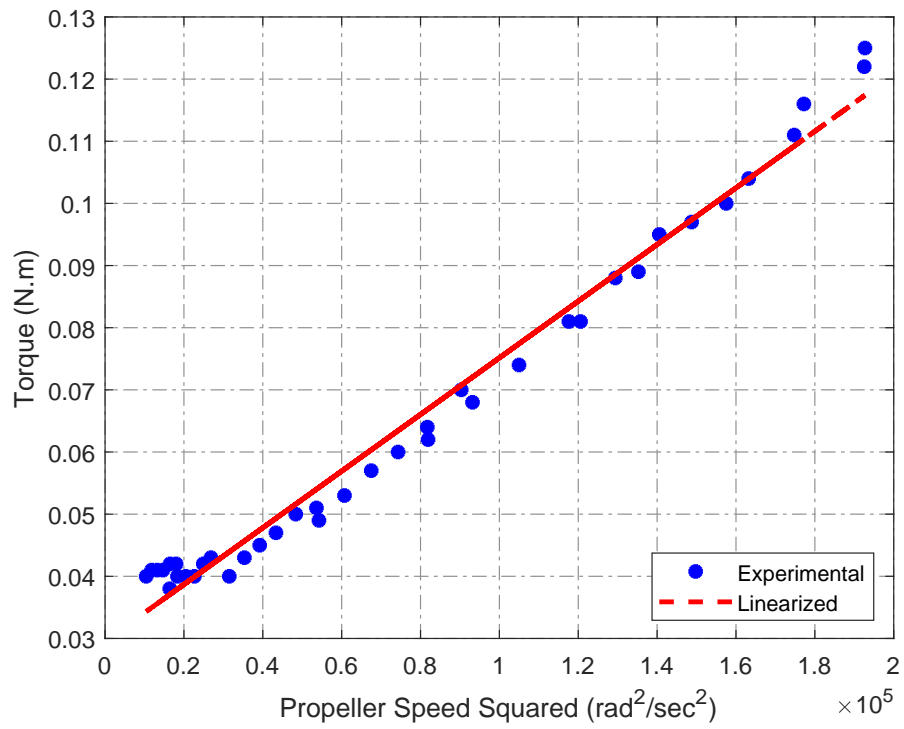


Figure 4.31: Relation of torque and the square of propeller speed

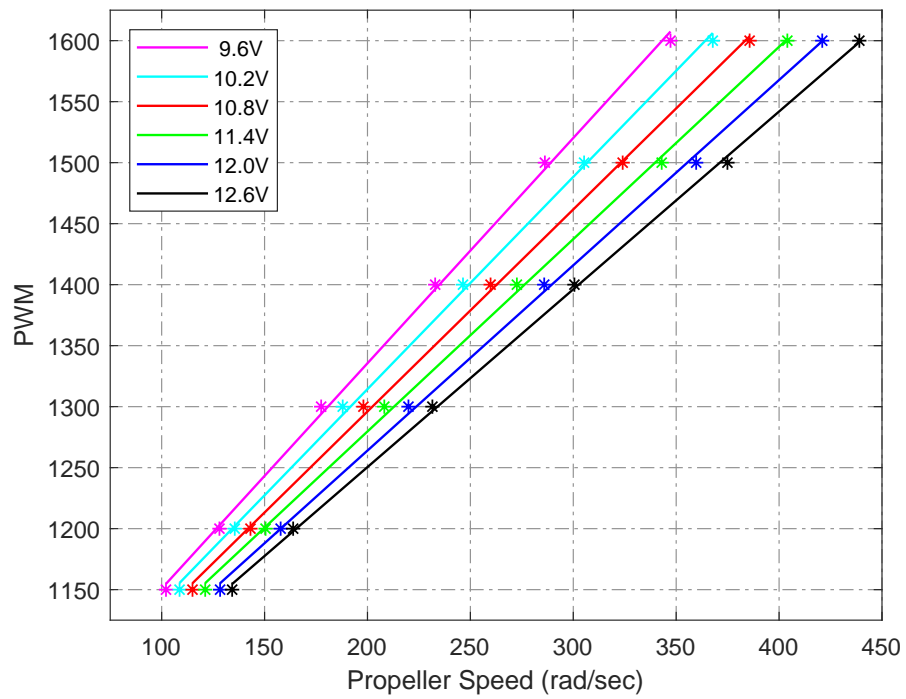


Figure 4.32: Relation of PWM and propeller speed

The force and torque outputs generated by the controller shall now be generated as a motor speed command, and if the ESCs can capture these motor speeds, the system will perform reasonably well for stabilization. Finally, the relation between motor speed and PWM, which is shown in Figure 4.32, should be used and PWM values should be commanded to the ESCs. In this final relation, voltage drop of Lipo battery is also taken into consideration.

4.4 Quadrotor Test Bench

Two test benches have been created in order to test whether the designed controller can really keep this quadrotor in a stable regime. With this way, breakage of quadrotor after a bad flight can be prevented. First, the setup shown in Figure 4.33, which allows the quadrotor to rotate only about one axis (pitch/roll motion), was created. The first performance of the controllers designed with the mathematical model was tested on this bench for pitch and roll motion.



Figure 4.33: Single axis quadrotor test bench

As the second test bench, a design that allows the quadrotor to move about 3 axes was preferred, which holds the drone with a universal joint at its centre. Pitch, roll and yaw movements were tested on this bench at the same time, and pre-flight algorithm experiments were performed.



Figure 4.34: Three Axes Test Bench

CHAPTER 5

CONTROLLER DESIGN AND APPLICATIONS ON QUADROTOR

This chapter describes the design of two control techniques, one is the linear PID controller and the other one is the nonlinear Backstepping technique. Two different control loops can be introduced for the control of quadrotor system. The first one is attitude hold and the other one is position hold. For the quadrotor system's ability to hold an accurate position, a well-designed attitude controller is necessary that is because the quadrotor system provides position control with thrust vectoring. In this context, ϕ and θ angles should be tracked as perfect as possible.

For the design of PID and Backstepping Controllers, mathematical expressions that reflect quadrotor dynamics have been used, which are already derived in Chapter 3. In the initial design phase, a Simulink model including the quadrotor dynamical model was created, and the first experiments were performed in this environment. These trials were then carried out on the three axis test bench to see both the accuracy of the model being created and the ability of the designed controller to stabilize this quadrotor system. The main point here was to test designed controllers before a catastrophic flight that might break down the quadrotor frame.

5.1 PID Controller

Well-known PID controller is the first method tried for the quadrotor system. In this control strategy, the system regularly tries to bring the error, difference between the reference signal and the measured value, to zero. This control method uses 3 elements. The first item, Proportional (P), produces an output proportional to the error, and increasing the value of this term decreases the rise time, which enables quicker

reduction in error. The Integral (I) term of the PID controller responds proportionally to the integral of the error, and reduces the steady state error. Derivative (D) term, on the other hand, is used to accelerate the system and create a more stable loop.

The structure of PID controller can be expressed with, reference signal $r(t)$, system output $y(t)$, error $e(t)$, and the controller output $u(t)$ as follows:

$$\begin{aligned} e(t) &= r(t) - y(t) \\ u(t) &= K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt} \end{aligned} \quad (5.1)$$

In Laplace domain, this well-known equation can be rewritten according to 5.2:

$$U(s) = (K_P + \frac{K_I}{s} + sK_D)E(s) \quad (5.2)$$

5.1.1 Attitude Control of Quadrotor

The quadrotor attitude dynamics allows the quadrotor to be directly controlled by a PID designed for angle stabilization. However, if such a controller is preferred, the quadrotor body rates will be uncontrolled, which may result in high rate values in angle stabilization. Cascaded PID structure has been preferred for this quadrotor system in order to control the fast rate dynamics and to make angle control easier. In this context, a PID controller is selected for body rate dynamics and a PI controller is used for the angle control as in the outer loop.

Design of the attitude controller of the quadrotor can be initiated with a reminder of rotational equations of motion, which are given according to 3.32.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (I_{yy} - I_{zz})qr/I_{xx} \\ (I_{zz} - I_{xx})pr/I_{yy} \\ (I_{xx} - I_{yy})pq/I_{zz} \end{bmatrix} + \begin{bmatrix} U_2/I_{xx} \\ U_3/I_{yy} \\ U_4/I_{zz} \end{bmatrix} + \begin{bmatrix} -J_r q \Omega/I_{xx} \\ J_r p \Omega/I_{yy} \\ 0 \end{bmatrix} \quad (3.32)$$

As it can be seen from the equations, roll, pitch and yaw dynamics of the quadrotor are coupled so that it is difficult to design a controller for them. Thus, the expressions can be simplified in a manner keeping the most dominant dynamical element. Inertia is the most important property that shapes the quadrotor's rotational behavior. Starting

with the roll axis, simplified equation and its Laplace form can be written according to 5.3:

$$\dot{p} = \frac{U_2}{I_{xx}} \implies P(s) = \frac{U_2(s)}{I_{xx}s} \quad (5.3)$$

The transfer function $G_p(s)$ between input $U_2(s)$ and output $p(s)$ is defined according to 5.4 as follows:

$$G_p(s) = \frac{p(s)}{U_2(s)} = \frac{1}{I_{xx}s} \quad (5.4)$$

The PID structure, for which Laplace form is given in 5.2, will be used for the roll rate control of the quadrotor system. In this case, the closed loop transfer function will be obtained according to 5.5:

$$G_c = \frac{KG}{1 + KG} \quad (5.5)$$

Thus, closed loop transfer function for the roll rate subsystem is expressed according to 5.6 as follows:

$$G_{cp}(s) = \frac{K_d s^2 + K_p s + K_i}{(K_d + I_{xx})s^2 + K_p s + K_i} \quad (5.6)$$

With a selection of PID gains as $K_p = 1.2$, $K_i = 0.8$ and $K_d = 0.04$, step response in Figure 5.1 is obtained.

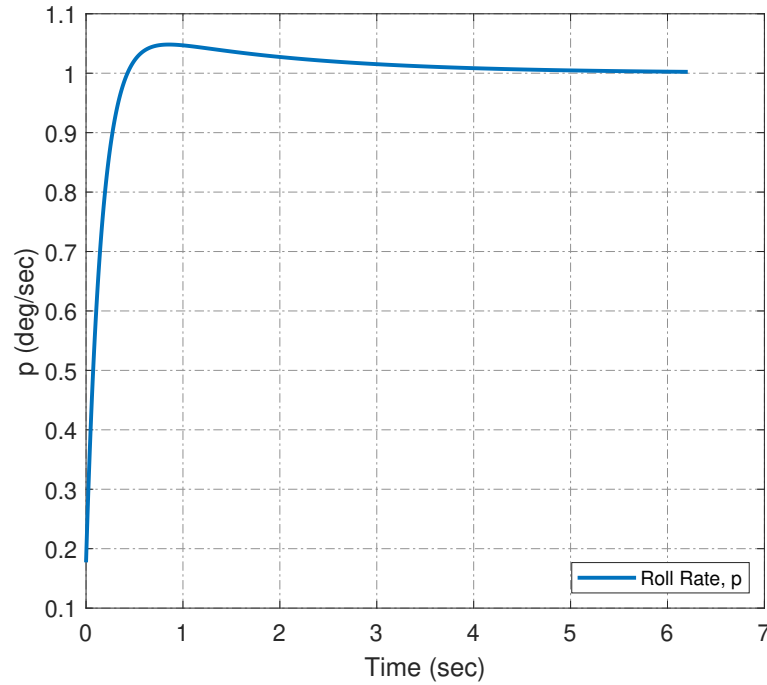


Figure 5.1: Roll rate step response

With the selected gains, the system will have a bandwidth of 1.14Hz as it can be seen from the Bode Plot (Figure 5.2) of the closed loop system.

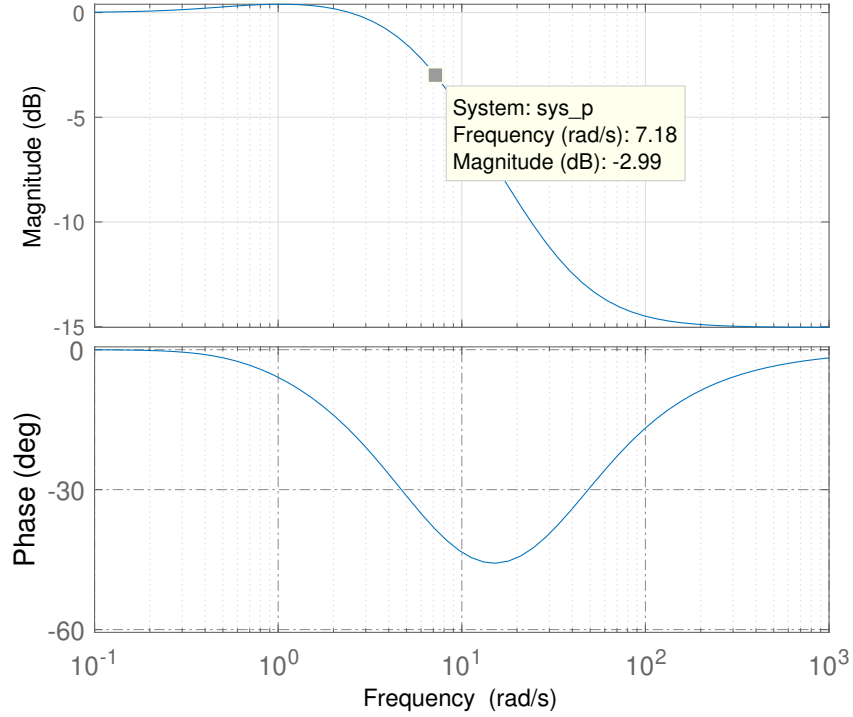


Figure 5.2: Closed loop roll rate Bode plot

Now, the outer angle loop can be closed. For the relation of the roll angle ϕ and roll rate p , the simple expression given in 5.7 can be used. Here, it should be reminded that p can be used instead of $\dot{\phi}$ as expressed in Section 3.3.2.

$$\dot{\phi} = p = \frac{\phi(s)}{p(s)} = \frac{1}{s} \quad (5.7)$$

For this outer loop of the cascaded structure, PI controller, whose Laplace form is given according to 5.8, was selected.

$$K_{\phi}(s) = K_p + \frac{K_i}{s} \quad (5.8)$$

With a selection of PI gains as $K_p = 4.0$ and $K_i = 1.0$, step response in Figure 5.3 is obtained.

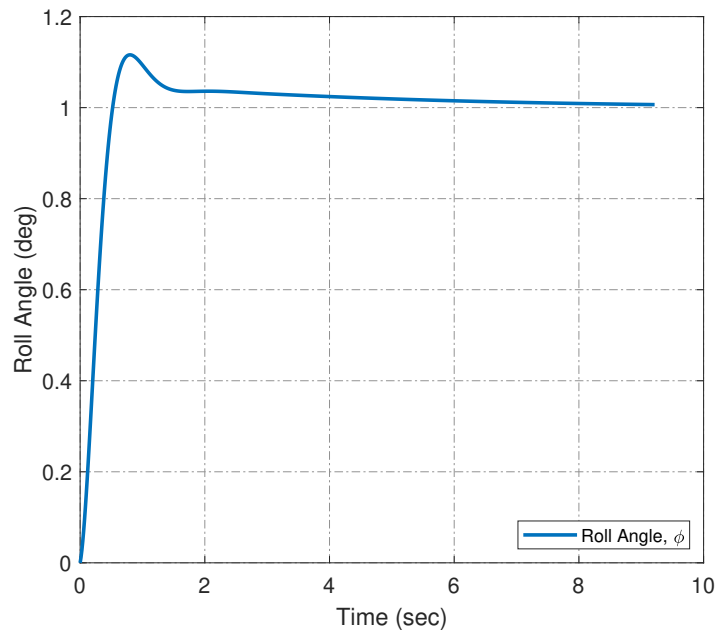


Figure 5.3: Roll angle step response

With the selected gains, the system will have a bandwidth of 0.89Hz as it can be seen from the Bode Plot (Figure 5.4) of the closed loop system.

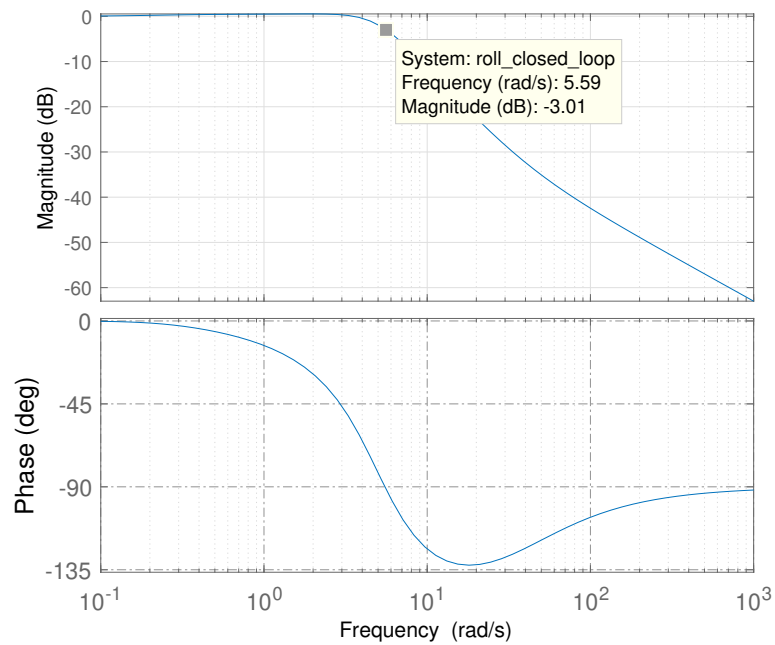


Figure 5.4: Closed loop roll angle Bode plot

The three axis test setup has a metallic structure, and it has been observed that vibrations are formed in the quadrotor frame with high gains of PID controllers. These vibrations cannot be simulated in the Simulink model, and therefore relatively slow dynamics was preferred with these PID gains to be able to compare the simulation and test setup results within a healthy procedure. These gains have been updated to their final values with a couple of flight testing.

Similar to the roll subsystem, the same equations were used for pitch dynamics with the same control gains since there is no big difference between the inertia values of the respective axes. In this context, step response graph for pitch angle θ and Bode Plot of the closed loop system can be seen in Figures 5.5 and 5.6, respectively.

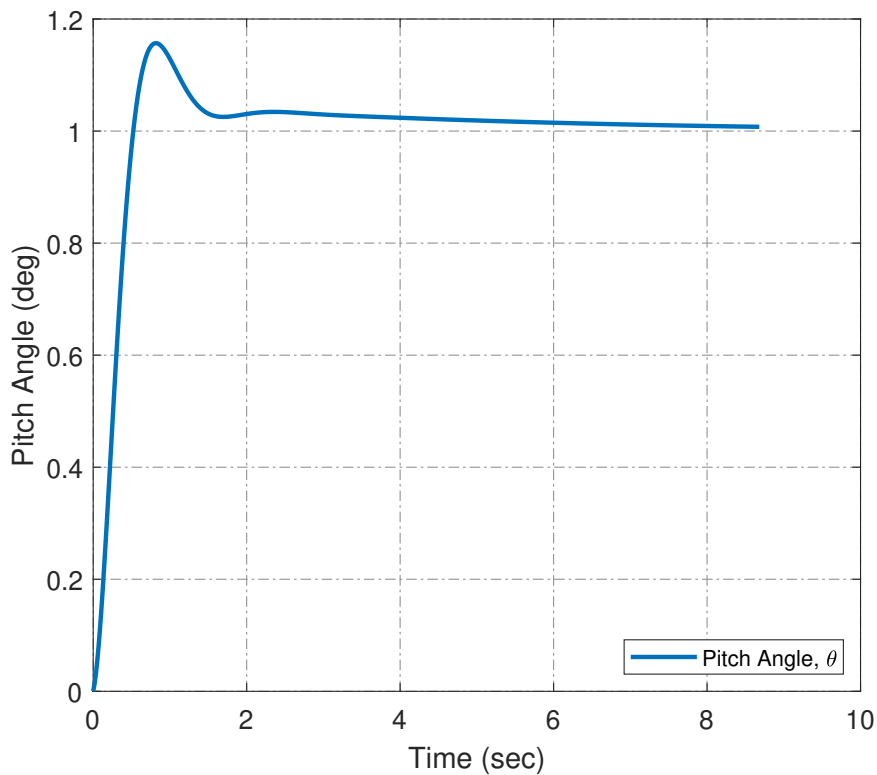


Figure 5.5: Pitch angle step response

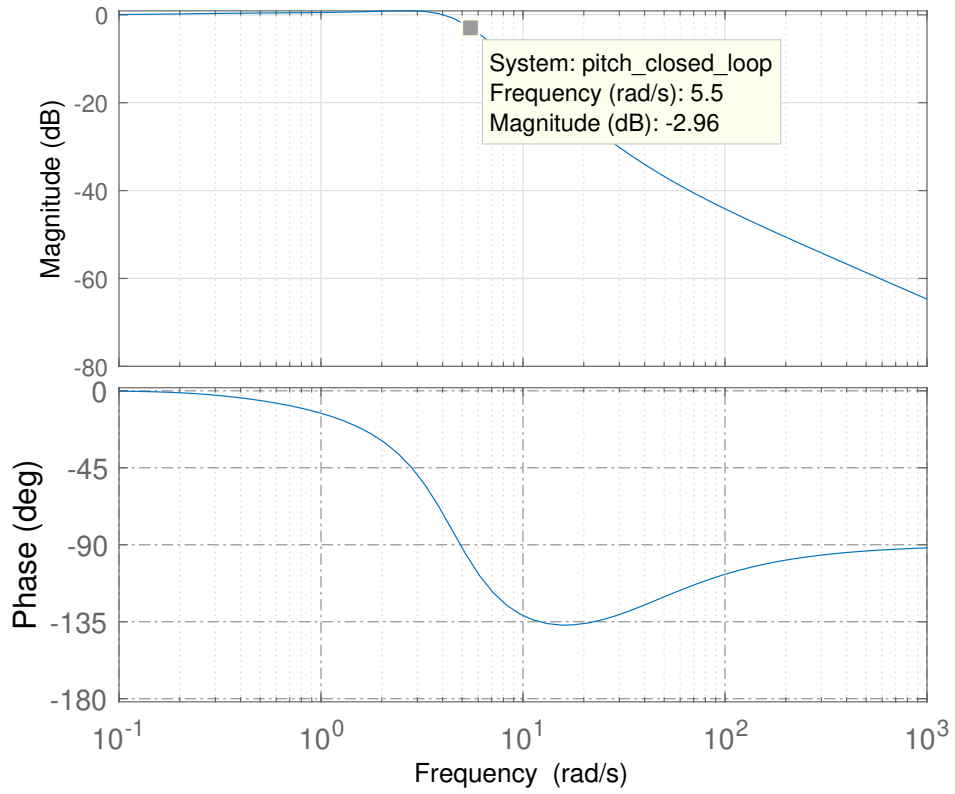


Figure 5.6: Closed loop pitch angle Bode plot

The designed Cascaded PID structure is implemented in the Simulink model as shown in Figure 5.7. Although not used in simulations, the corresponding external reset signals are connected to the integral blocks in order to reset them at the beginning of the flight and in the necessary situations of real trials.

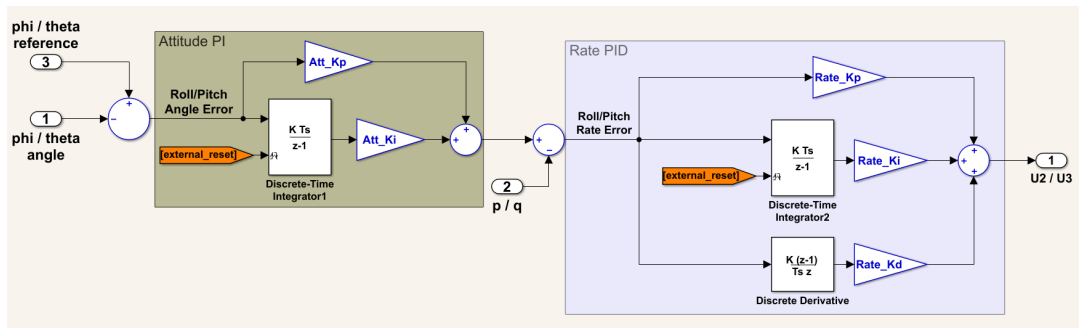


Figure 5.7: Cascaded PID control diagram for pitch and roll axes

The controller of yaw subsystem was designed in the same way as pitch and roll dynamics. However, a slower dynamic response was preferred for this axis. As mentioned earlier, the quadrotor movement is realized with changes in pitch, roll and yaw angles. For the position control, quadrotor does not need to change its heading, and can move in 2D-space without rotating about yaw axis. In this study, the yaw axis was mostly used for the conservation of the existing heading. Therefore, a rapid reference follow-up was not needed and a simpler structure was preferred.

For yaw rate control, PI controller was preferred with gains $K_p = 0.2$ and $K_i = 0.05$. For the outer angle loop, only a Proportional control is applied with a gain $K_p = 2.75$. Step response of yaw angle and Bode Plot of the closed loop yaw angle system can be seen in Figures 5.8 and 5.9, respectively.

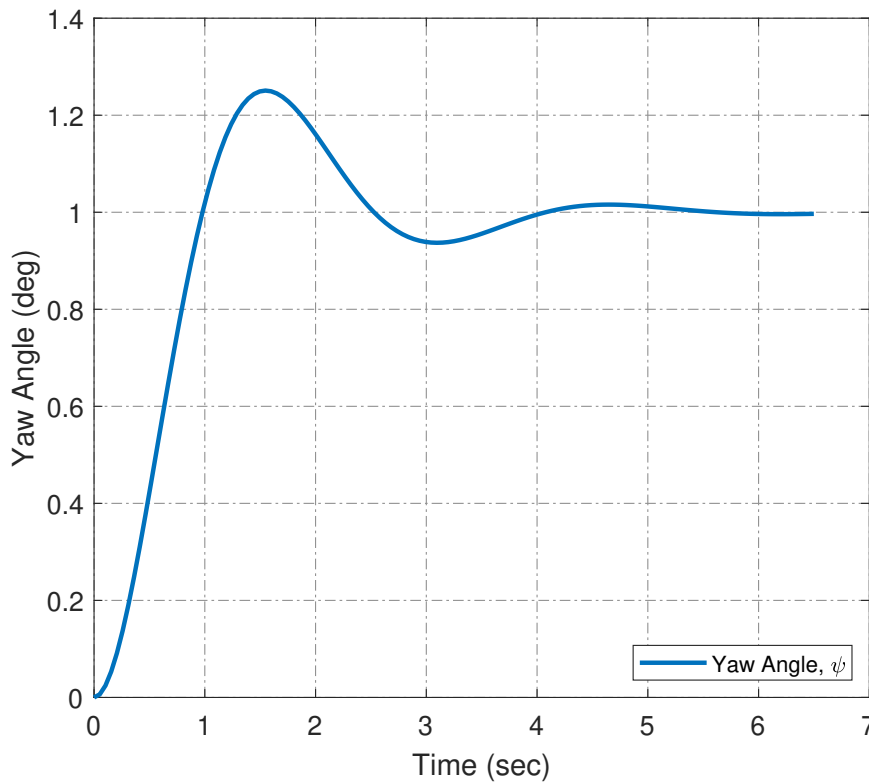


Figure 5.8: Yaw angle step response

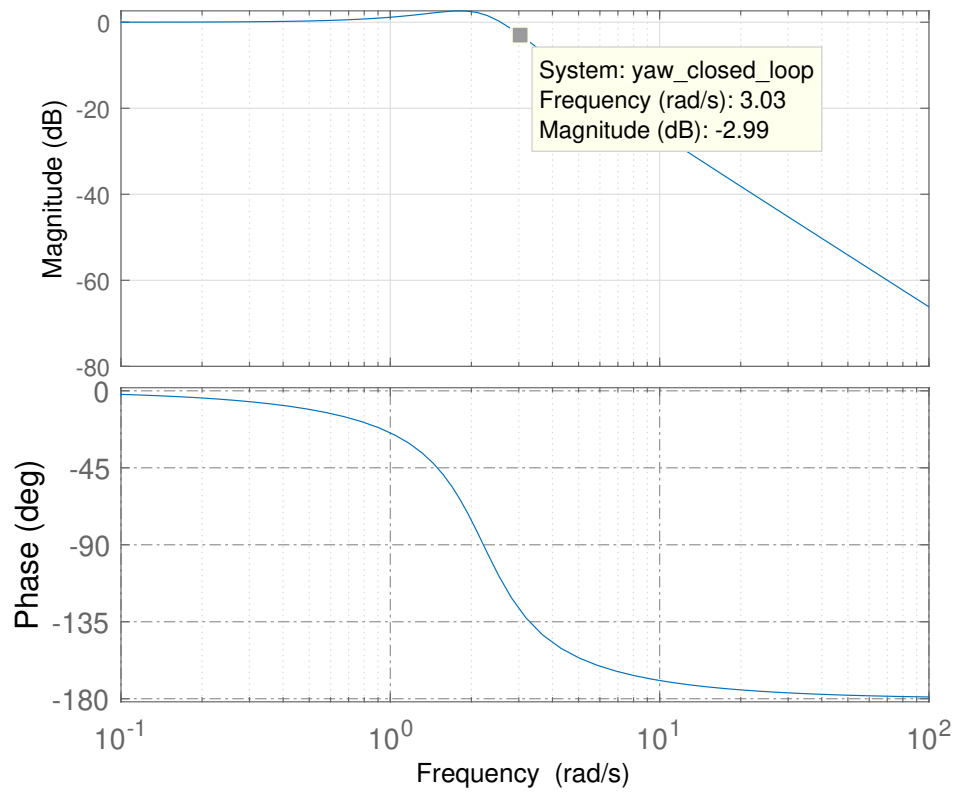


Figure 5.9: Closed loop yaw angle Bode plot

The designed Cascaded control structure is implemented in the Simulink model as shown in Figure 5.10.

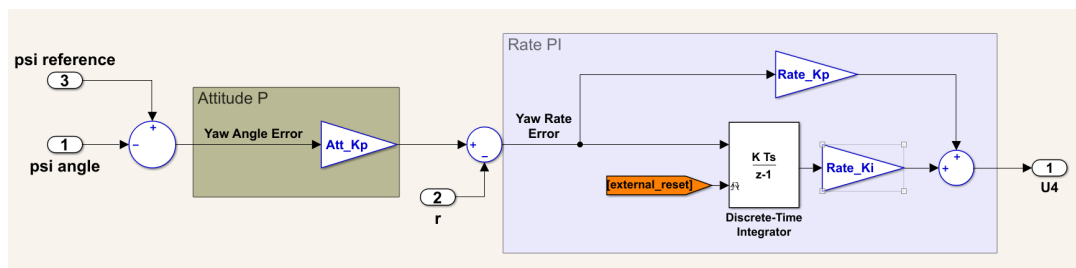


Figure 5.10: Cascaded PID control diagram for yaw axis

5.1.2 Altitude and Position Control of Quadrotor

Design of the position controllers of the quadrotor can be initiated with a reminder of translational equations of motion, which are given according to 3.52.

$$\begin{aligned}\ddot{x} &= -\frac{U_1}{m} \left(\cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi) \right) \\ \ddot{y} &= -\frac{U_1}{m} \left(\cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi) \right) \\ \ddot{z} &= g - \frac{U_1}{m} \left(\cos(\phi) \cos(\theta) \right)\end{aligned}\quad (3.52)$$

As it can be seen from the equations, X and Y motion is governed by thrust vectoring, which means the inclination of the U_1 component by ϕ and θ angles gives the relative motion. Here, ψ angle term can be omitted since the quadrotor will align its heading with the marker.

If a linearization is performed for X and Y motion equations, the simplified velocity forms are obtained according to 5.9:

$$\begin{aligned}\dot{u} &= -g\theta \implies \frac{u(s)}{\theta(s)} = \frac{-g}{s} \\ \dot{v} &= g\phi \implies \frac{v(s)}{\phi(s)} = \frac{g}{s}\end{aligned}\quad (5.9)$$

There is a need of a velocity loop to be able to control the position in an accurate manner. This velocity controller should produce angle reference to the attitude controllers. The outer position controller, as in the same manner with velocity, should produce velocity reference. With this cascaded structure, the velocity and angle variations of the quadrotor can also be limited.

Unlike the attitude controller design, velocity expressions in 5.9 does not directly depend on the quadrotor behavior, but depend on the gravitational acceleration. This is not convenient to design the outer control loop since the results will be different on the real system. Thus, controller design for position loop will be done using the closed loop simulation model formed in Simulink.

With linearization, the trigonometric relation between angles and the translational velocity is lost. Since the design will be based on the control model, this nonlinearity

can be introduced into controller with an inverse relation. There are studies [67], [68] that consider this relation in their controller design. This relation can be given according to 5.10 as expressed in [67]:

$$\begin{aligned}\phi_d &= \arcsin \frac{u_y}{g} \\ \theta_d &= -\arcsin \frac{u_x}{g}\end{aligned}\tag{5.10}$$

For the velocity loop, a PI controller is used with gains $K_p = 0.5$ and $K_i = 0.1$. For the outer position loop, only a proportional control is applied with a gain $K_p = 1.0$. To see the performance of the controllers, a sinusoidal input is given as X position reference, and pitch angle θ , X velocity v_x and X position are monitored as it can be seen in Figure 5.11.

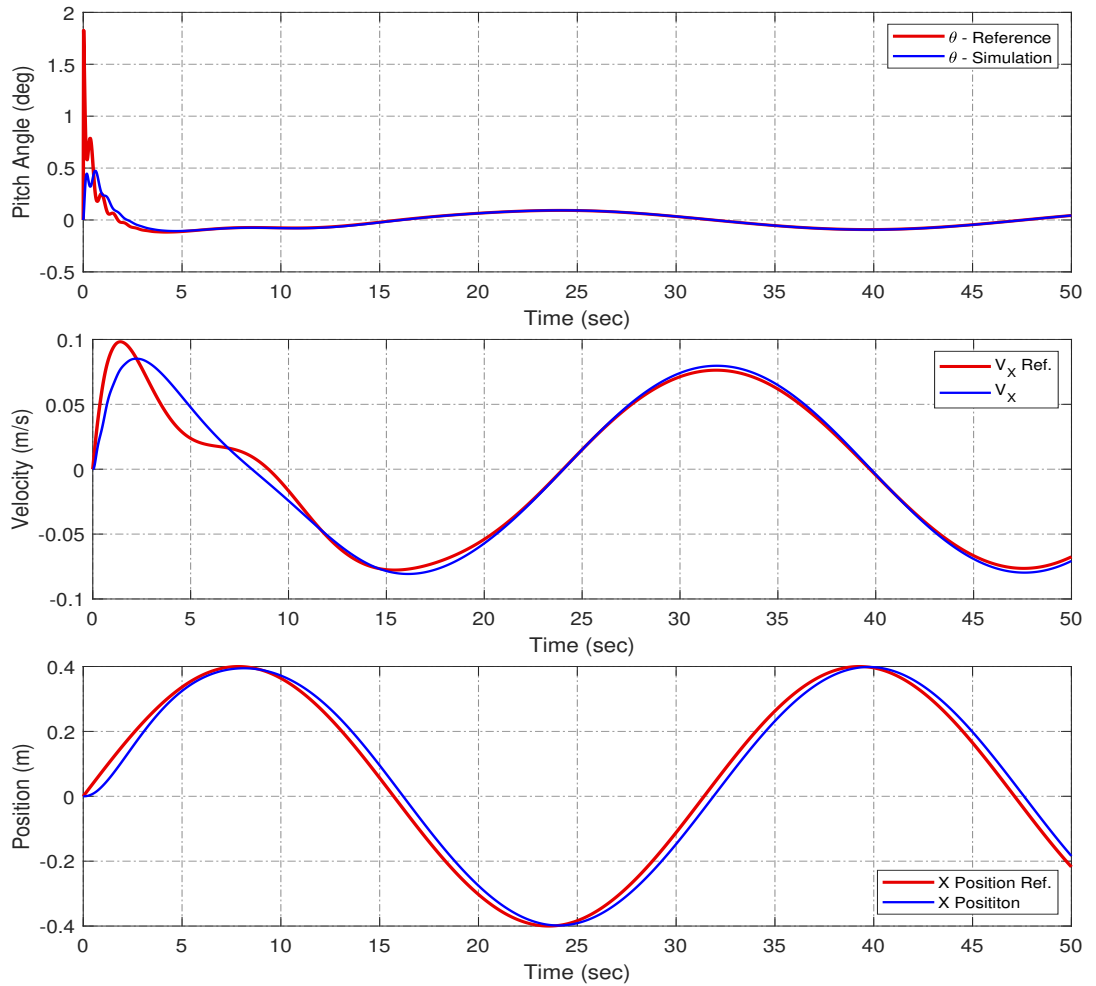


Figure 5.11: Response of X-Position, X-Velocity and Pitch Angle of quadrotor to commanded position reference

As it can be seen from the responses, both velocity and angle subsystems can track the commanded references by enabling a proper position control. For the position control, the diagram shown in Figure 5.12 is applied on Simulink model.

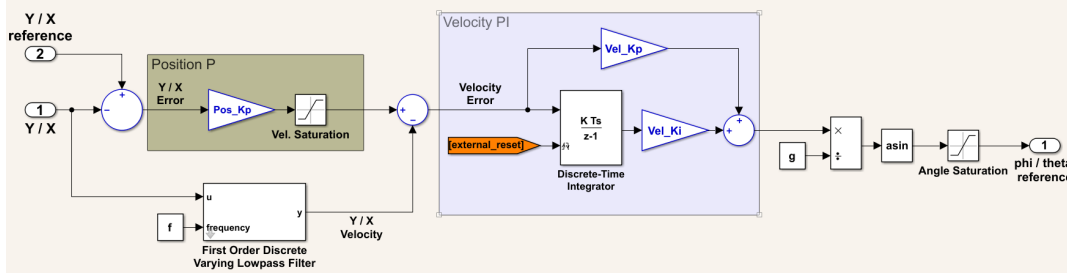


Figure 5.12: Cascaded PID diagram for x/y position control

For the altitude control, the same procedure followed in position control can be preferred. For the altitude axis, there is no velocity - angle relation, and thus a PID controller can be directly applied through this axis. At the hover condition, the quadrotor should always balance its weight in a way that the controller should apply a constant output in addition to the output corresponding to the altitude error. This can be seen from the linearized relation given in 5.11:

$$\ddot{z} = -\frac{U_1}{m} + g \quad (5.11)$$

By considering the gravity term as a constant disturbance, a feed-forward control structure is preferred. This diagram is shown in Figure 5.13.

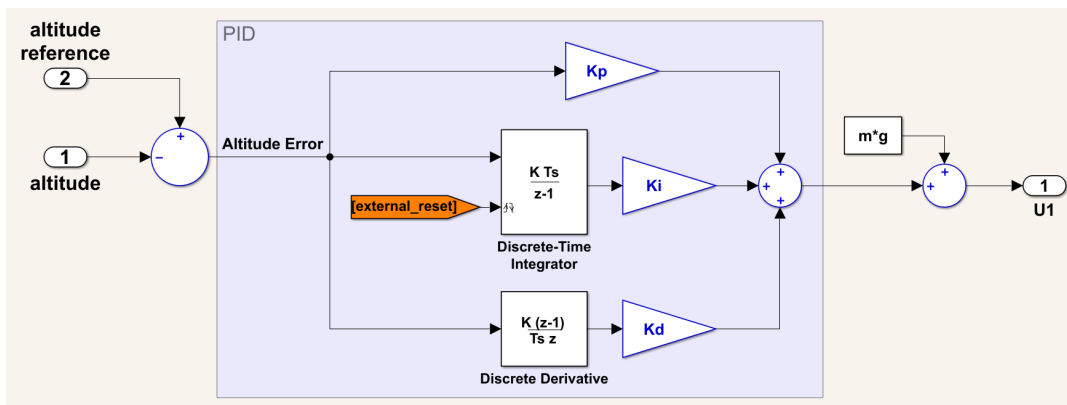


Figure 5.13: PID diagram for altitude control

The PID controller of the altitude system has gains of $K_p = 6$, $K_i = 8$ and $K_d = 4$. The response of quadrotor to given altitude reference is given in Figure 5.14.

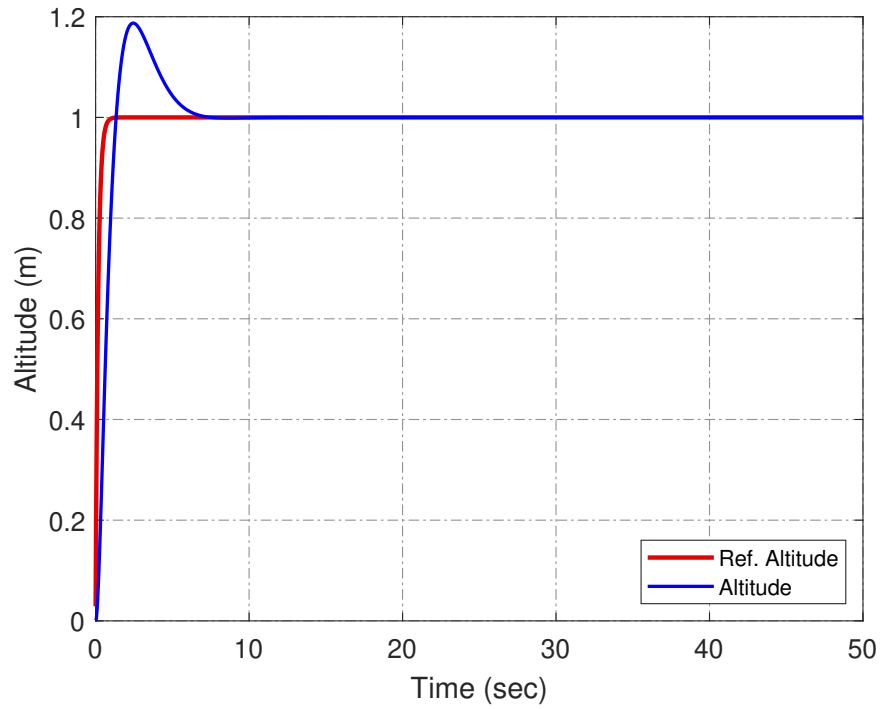


Figure 5.14: Response of quadrotor to given altitude reference

5.1.3 Attitude Performance of PID Controller on Test Bench

Attitude controller design of the quadrotor was the first step of study, and the first version of the controller was created by using the expressions coming from the quadrotor mathematical modeling. The design was then reflected in the Simulink model (Figure 5.15) and the attitude dynamics of the quadrotor were simulated. In order to prove both the accuracy of the mathematical model and the controller's ability to stabilize this quadrotor, some trials were performed on the three axis test bench.

Within the scope of this campaign, 3 main trials were conducted. In the first one, ramp-wise inputs were given at different times in the pitch and roll axes. While stepwise inputs are given on the same axes being as the second trials, sinusoidal inputs were applied in the last attempt. In these three tests, no input was given on the yaw channel because the quadrotor does not require heading orientation for position control.

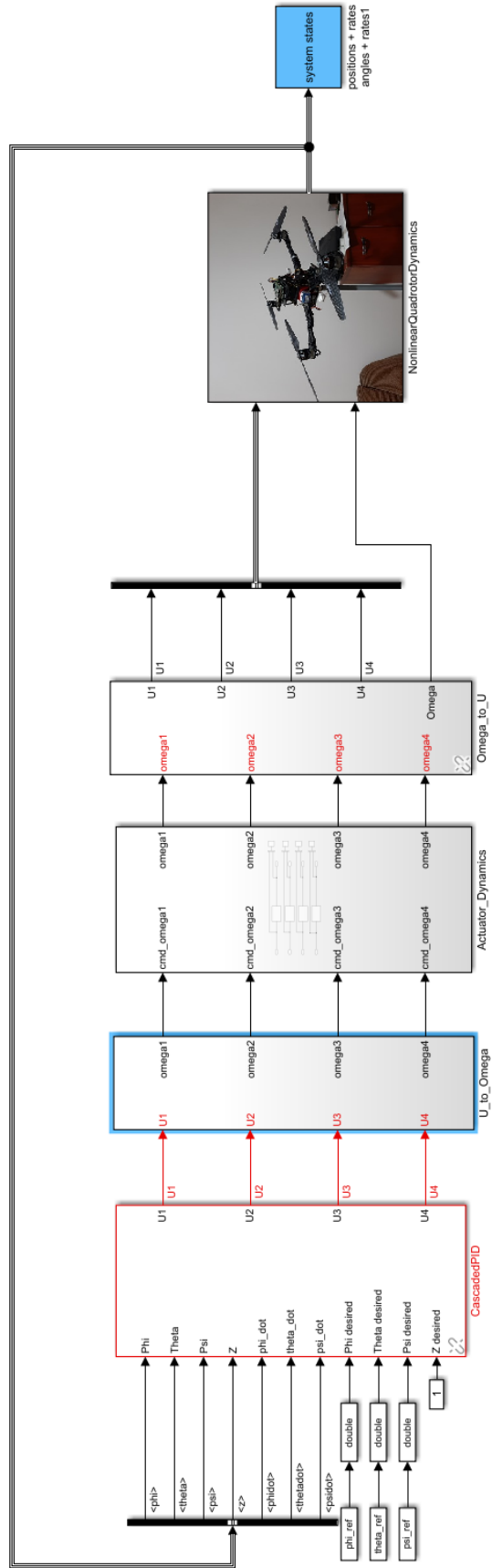


Figure 5.15: Simulink model used for quadrotor simulations

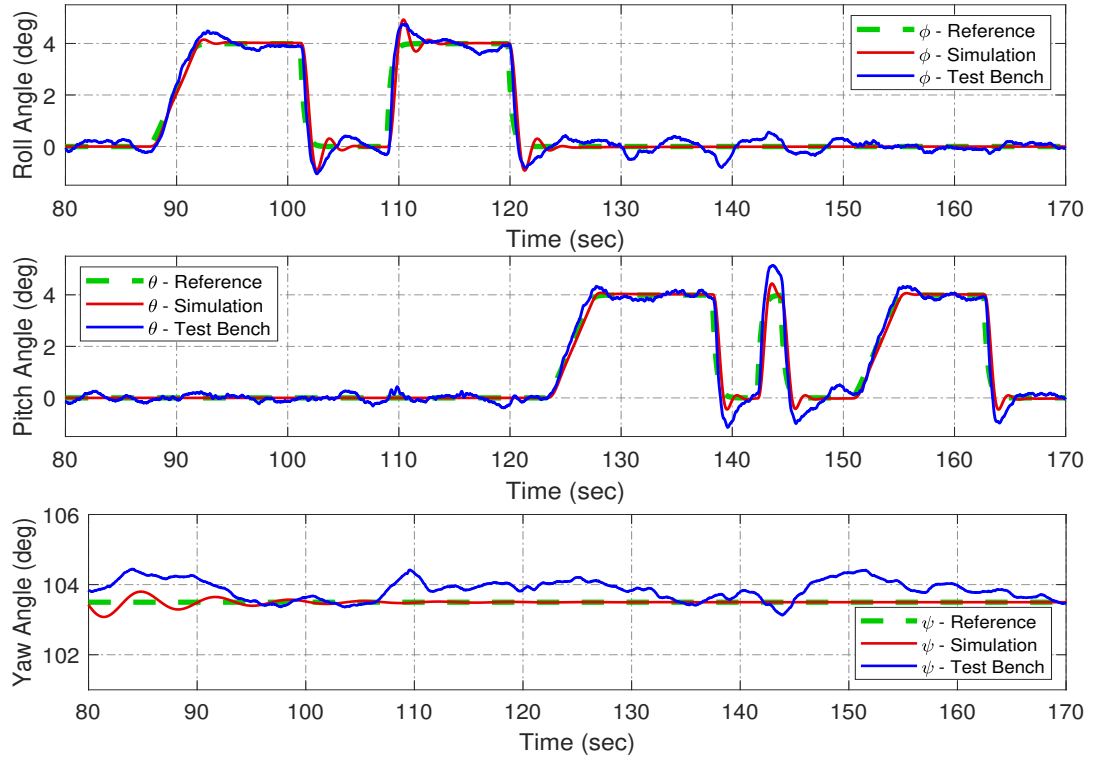


Figure 5.16: Response of quadrotor to ramp-wise inputs given in pitch and roll axes

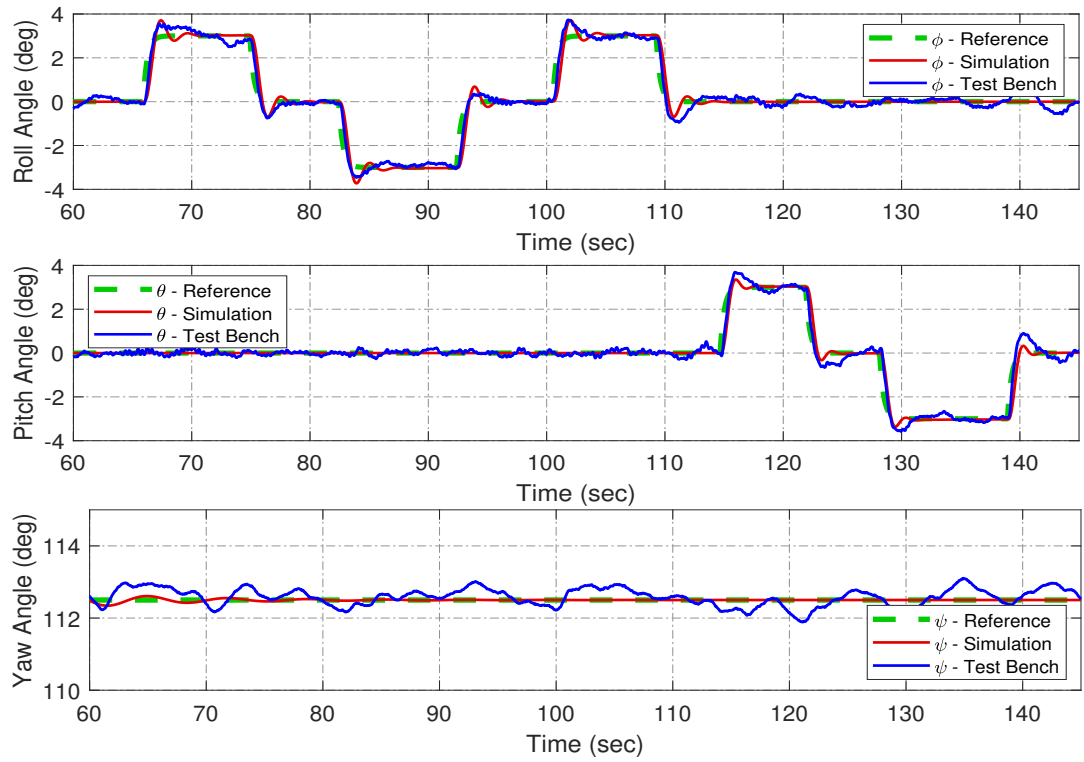


Figure 5.17: Response of quadrotor to step-wise inputs given in pitch and roll axes

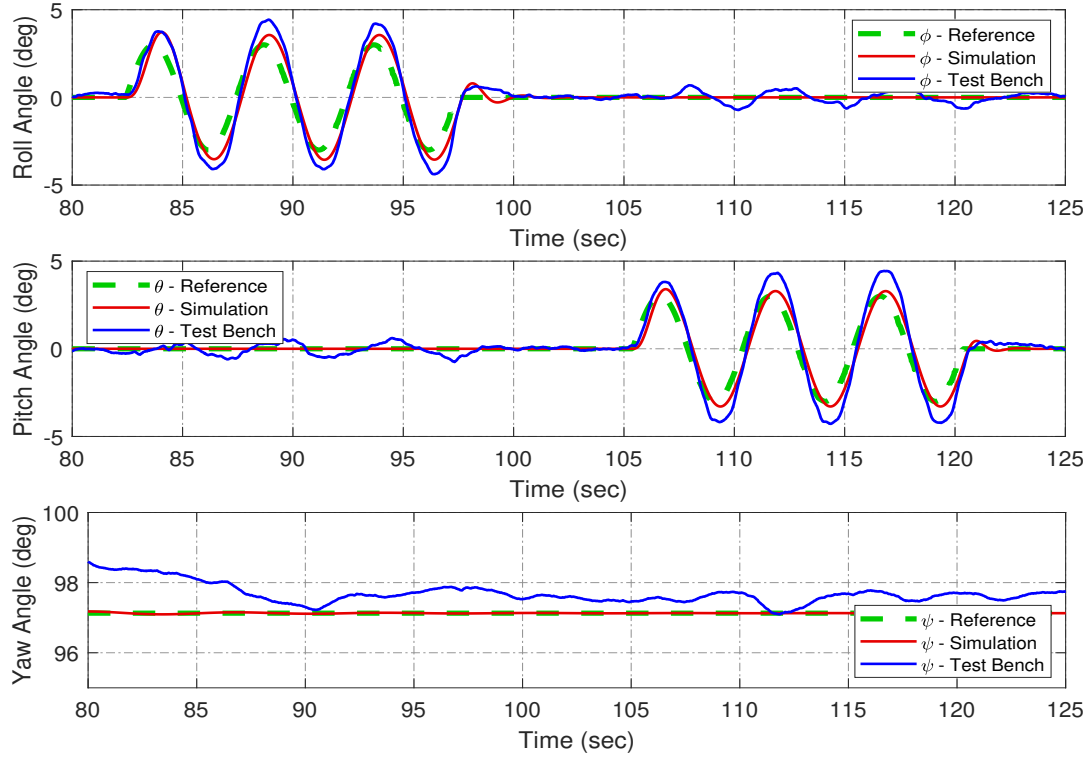


Figure 5.18: Response of quadrotor to sinusoidal inputs given in pitch and roll axes

As shown in Figures 5.16, 5.17 and 5.18, the designed attitude controllers were able to follow the given reference commands properly. Since the three axis test setup has a metallic structure, vibrations were observed in the quadrotor frame if the gains are increased. For this reason, the experiments were carried out by keeping the gains at this level. In addition to vibrations, observed overshoot is due to the fact that the quadrotor cannot be connected to the test setup from the center of gravity, but is connected 6cm below. For this reason, high overshoot values were observed in high frequency inputs. The results of the Simulink model were found to be consistent with the results obtained from the test bench, and flight experiments were started with these controllers.

5.2 Backstepping Controller

Backstepping is a recursive design process where intermediate control inputs are defined to stabilize subsystems, that are clusters of system states. The procedure gener-

ally starts with the smallest feasible subsystem in the whole state-space model. After stabilizing the smallest cluster, outer subsystems are balanced backwards in an iterative manner. The operation finalizes when the most outer system is stabilized with the virtual control input.

For most of the control methods, linearization brings simplicity. In the case of backstepping, however, there is no need for such simplification and nonlinearities in the system equations can be used within the process. This also allows the control system to react to these nonlinear phenomena. As an example, cross-coupling effects between axes can be canceled out with the proper implementation of the backstepping controller. There are couple of works on the application of backstepping for quadrotor systems [37], [38], [40], [42] and [58]. In this chapter, the derivations that is proposed by these studies will be followed.

The process can be initiated by recalling the equations of motion that have been obtained in Section 3. Complete state space representation of the quadrotor system has been found in Equation 3.55 as:

$$\begin{aligned}
\dot{x}_1 &= \dot{\phi} = x_2 \\
\dot{x}_2 &= \ddot{\phi} = a_1 x_4 x_6 + b_1 U_2 - a_2 x_4 \Omega \\
\dot{x}_3 &= \dot{\theta} = x_4 \\
\dot{x}_4 &= \ddot{\theta} = a_3 x_2 x_6 + b_2 U_3 - a_4 x_2 \Omega \\
\dot{x}_5 &= \dot{\psi} = x_6 \\
\dot{x}_6 &= \ddot{\psi} = a_5 x_2 x_4 + b_3 U_4 \\
\dot{x}_7 &= \dot{z} = x_8 \\
\dot{x}_8 &= \ddot{z} = g - \frac{U_1}{m} \left(\cos(x_1) \cos(x_3) \right) \\
\dot{x}_9 &= \dot{x} = x_{10} \\
\dot{x}_{10} &= \ddot{x} = -\frac{U_1}{m} \left(\cos(x_1) \sin(x_3) \cos(x_5) + \sin(x_1) \sin(x_5) \right) \\
\dot{x}_{11} &= \dot{y} = x_{12} \\
\dot{x}_{12} &= \ddot{y} = -\frac{U_1}{m} \left(\cos(x_1) \sin(x_5) \sin(x_3) - \sin(x_1) \cos(x_5) \right)
\end{aligned} \tag{3.55}$$

It is clear that equations representing the rotational motion do not depend on translational elements x , y , z and their derivatives. As opposed to this, equations of trans-

lational motion strictly depend on the rotational components ϕ , θ and ψ . Thus, the overall system can be considered as made up of two subsystems that are rotational and translational ones. Since the angular system is not dependent on the translational terms, it can be considered as the inner loop of the design and angular system can be stabilized by taking the desired Euler angles as reference. These angular references can be produced by the outer translational subsystem, which will also be regulated to hold the quadrotor's position.

5.2.1 Backstepping Control of Rotational Motions

Backstepping design process can be initiated with the first two equations seen in the state space representation of the equations of motion.

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= a_1 x_4 x_6 + b_1 U_2 - a_2 x_4 \Omega\end{aligned}\tag{5.12}$$

In Equation 5.12, roll dynamics appears as a simple double integrator system [40]. This is also the case for pitch and yaw dynamics. They all hold the control inputs U in the expressions so that they can be extracted to find the control inputs.

x_1 should be stabilized in the first step, and then x_2 can be further regulated. The tracking error for the roll angle ϕ can be defined as:

$$\begin{aligned}e_1 &= \phi_d - \phi \\ &= x_{1d} - x_1\end{aligned}\tag{5.13}$$

In order to ensure the stability of the equilibrium point where $e_1 = 0$, Lyapunov theorem can be used. A positive definite Lyapunov function should be selected for e_1 whose derivative with respect to time is negative semi-definite.

$$V(e_1) = \frac{1}{2}e_1^2\tag{5.14}$$

Taking the first derivative of Equation 5.14, the following representation is obtained:

$$\begin{aligned}\dot{V}(e_1) &= e_1 \dot{e}_1 \\ &= e_1 (\dot{x}_{1d} - \dot{x}_1) \\ &= e_1 (\dot{x}_{1d} - x_2)\end{aligned}\tag{5.15}$$

With a positive definite Lyapunov function selection, its derivative can be obtained as negative semi-definite to guarantee the stability of the system [69]. For that purpose, a virtual control input x_{2v} can be defined that will make $\dot{V}(e_1) = 0$.

$$x_{2v} = \dot{x}_{1d} + \alpha_1 e_1 \quad \text{with } \alpha_1 > 0 \quad (5.16)$$

Here, if the real state x_2 follows the virtual control input x_{2v} , following is obtained;

$$V(e_1) = -\alpha_1 e_1^2 \leq 0 \quad (5.17)$$

Thus, a stable system is obtained for x_1 for the case x_2 converges to x_{2v} as defined in Equation 5.16. There should be an outer control loop for that to be happen. It can be done by introducing the following tracking error as the origin of outer system.

$$\begin{aligned} e_2 &= x_2 - x_{2v} \\ &= x_2 - \dot{x}_{1d} - \alpha_1 e_1 \end{aligned} \quad (5.18)$$

Now, the Lyapunov function defined in Equation 5.14 should be expanded so that it can enclose both tracking errors e_1 and e_2 .

$$V(e_1, e_2) = \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 \quad (5.19)$$

Before writing the derivative of the new function, derivatives of the tracking errors can be specified for the ease of calculation:

$$\begin{aligned} \dot{e}_1 &= \dot{x}_{1d} - \dot{x}_2 = (\dot{x}_2 - e_2 - \alpha_1 e_1) - \dot{x}_2 = -e_2 - \alpha_1 e_1 \\ \dot{e}_2 &= \dot{x}_2 - \ddot{x}_{1d} - \alpha_1 \dot{e}_1 = a_1 x_4 x_6 + b_1 U_2 - a_2 x_4 \Omega - \ddot{x}_{1d} + \alpha_1 (e_2 + \alpha_1 e_1) \end{aligned} \quad (5.20)$$

Then, the first derivative of Lyapunov function becomes:

$$\begin{aligned} \dot{V}(e_1, e_2) &= e_1 \dot{e}_1 + e_2 \dot{e}_2 \\ &= e_1 (-e_2 - \alpha_1 e_1) + e_2 (\dot{x}_2 - \ddot{x}_{1d} - \alpha_1 \dot{e}_1) \\ &= e_1 (-e_2 - \alpha_1 e_1) \\ &\quad + e_2 (a_1 x_4 x_6 + b_1 U_2 - a_2 x_4 \Omega - \ddot{x}_{1d} + \alpha_1 (e_2 + \alpha_1 e_1)) \end{aligned} \quad (5.21)$$

In Equation 5.21, \ddot{x}_{1d} term can be accepted as 0 since the reference will be changing slowly.

In order to satisfy $\dot{V}(e_1, e_2) < 0$, the control input U_2 is extracted as follows:

$$U_2 = \frac{1}{b_1}(e_1 - \alpha_2 e_2 - a_1 x_4 x_6 + a_2 x_4 \Omega - \alpha_1(e_2 + \alpha_1 e_1)) \quad (5.22)$$

As Equation 5.22 is put into Equation 5.21, the following expression is obtained:

$$\dot{V}(e_1, e_2) = -\alpha_1 e_1^2 - \alpha_2 e_2^2 \quad (5.23)$$

which is negative definite for $\alpha_1 > 0$ and $\alpha_2 > 0$. Thus, asymptotic stability is obtained for the control of roll angle ϕ with input U_2 .

Pitch and yaw dynamics equations are very similar to the ones of roll. Thus, the same order can be easily followed to obtain related control inputs for the remaining rotational motions. Let's first remind the equations of motion for pitch and yaw.

$$\begin{aligned} \dot{x}_3 &= x_4 \\ \dot{x}_4 &= a_3 x_2 x_6 + b_2 U_3 - a_4 x_2 \Omega \\ \dot{x}_5 &= x_6 \\ \dot{x}_6 &= a_5 x_2 x_4 + b_3 U_4 \end{aligned} \quad (5.24)$$

As the next step, the tracking errors for inner and outer systems of θ and ψ can be introduced.

$$\begin{aligned} e_3 &= \theta_d - \theta = x_{3d} - x_3 \\ e_4 &= x_4 - x_{4v} = x_4 - \dot{x}_{3d} - \alpha_3 e_3 \\ e_5 &= \psi_d - \psi = x_{5d} - x_5 \\ e_6 &= x_6 - x_{6v} = x_6 - \dot{x}_{5d} - \alpha_5 e_5 \end{aligned} \quad (5.25)$$

Similar to the Lyapunov function defined in Equation 5.19, candidate functions for pitch and yaw can be defined as:

$$\begin{aligned} V(e_3, e_4) &= \frac{1}{2}e_3^2 + \frac{1}{2}e_4^2 \\ V(e_5, e_6) &= \frac{1}{2}e_5^2 + \frac{1}{2}e_6^2 \end{aligned} \quad (5.26)$$

Time derivative of Lyapunov function for the pitch motion is obtained as follows:

$$\begin{aligned}
\dot{V}(e_3, e_4) &= e_3 \dot{e}_3 + e_4 \dot{e}_4 \\
&= e_3(-e_4 - \alpha_3 e_3) + e_4(\dot{x}_4 - \ddot{x}_{3d} - \alpha_3 \dot{e}_3) \\
&= e_3(-e_4 - \alpha_3 e_3) \\
&\quad + e_4(a_3 x_2 x_6 + b_2 U_3 - a_4 x_2 \Omega - \ddot{x}_{3d} + \alpha_3(e_4 + \alpha_3 e_3))
\end{aligned} \tag{5.27}$$

Finally, let's extract the pitch control input U_3 from Equation 5.27.

$$U_3 = \frac{1}{b_2}(e_3 - \alpha_4 e_4 - a_3 x_2 x_6 + a_4 x_2 \Omega - \alpha_3(e_4 + \alpha_3 e_3)) \tag{5.28}$$

For the case of yaw motion, taking the time derivative of Lyapunov function yields:

$$\begin{aligned}
\dot{V}(e_5, e_6) &= e_5 \dot{e}_5 + e_6 \dot{e}_6 \\
&= e_5(-e_6 - \alpha_5 e_5) + e_6(\dot{x}_6 - \ddot{x}_{5d} - \alpha_5 \dot{e}_5) \\
&= e_5(-e_6 - \alpha_5 e_5) \\
&\quad + e_6(a_5 x_2 x_4 + b_3 U_4 - \ddot{x}_{5d} + \alpha_5(e_6 + \alpha_5 e_5))
\end{aligned} \tag{5.29}$$

As similar to roll and pitch inputs, yaw input can be obtained from Equation 5.29.

$$U_4 = \frac{1}{b_3}(e_5 - \alpha_6 e_6 - a_5 x_2 x_4 - \alpha_5(e_6 + \alpha_5 e_5)) \tag{5.30}$$

Obtained control inputs U_3 and U_4 ensures asymptotic stability by assuring both $\dot{V}(e_3, e_4) < 0$ and $\dot{V}(e_5, e_6) < 0$ for which coefficients α_3 , α_4 , α_5 and α_6 are bigger than zero.

5.2.2 Backstepping Control of Translational Motions

Translational control of the quadrotor system is composed of altitude and horizontal plane controls. For the case of altitude, there is a direct control input U_1 for the purpose of regulation. In the case of position control, however, there is no direct control input and thrust vectoring yields the desired motion in the $x - y$ plane. There is a need to find the desired angles ϕ_d , θ_d and ψ_d so that they can be fed to the attitude controllers.

Let's start with a recall of equations representing the altitude motion.

$$\begin{aligned}\dot{x}_7 &= x_8 \\ \dot{x}_8 &= g - \frac{U_1}{m} (\cos(x_1) \cos(x_3))\end{aligned}\tag{5.31}$$

The tracking error is defined as follows:

$$e_7 = z_d - z = x_{7d} - x_7\tag{5.32}$$

With the virtual control input $x_8 = \dot{x}_{7d} + \alpha_7 e_7$, the outer loop tracking error is also defined as:

$$e_8 = x_8 - \dot{x}_{7d} - \alpha_7 e_7\tag{5.33}$$

Let's introduce the Lyapunov candidate for the altitude control system:

$$V(e_7, e_8) = \frac{1}{2} e_7^2 + \frac{1}{2} e_8^2\tag{5.34}$$

Taking the derivative of Equation 5.34, the following is obtained:

$$\begin{aligned}\dot{V}(e_7, e_8) &= e_7 \dot{e}_7 + e_8 \dot{e}_8 \\ &= e_7 (-\dot{x}_8 - \alpha_7 \dot{e}_7) + e_8 (\dot{x}_8 - \ddot{x}_{7d} - \alpha_7 \dot{e}_7) \\ &= e_7 (-\dot{x}_8 - \alpha_7 \dot{e}_7) \\ &\quad + e_8 (g - \frac{U_1}{m} (\cos(x_1) \cos(x_3)) - \ddot{x}_{7d} + \alpha_7 (e_8 + \alpha_7 e_7))\end{aligned}\tag{5.35}$$

At the last step, the control input U_1 can be extracted from Equation 5.35 so that $\dot{V}(e_7, e_8) < 0$ condition is met.

$$U_1 = \frac{m}{\cos(x_1) \cos(x_3)} (-e_7 + \alpha_8 e_8 + g + \alpha_7 (e_8 + \alpha_7 e_7))\tag{5.36}$$

With Equation 5.36, all four control inputs of the quadrotor system, that are explained in Section 3.1, have been obtained. Now, let's navigate to the virtual control inputs that should be generated for position control. In Equation 3.54 of Section 3.3.5, u_x and u_y virtual inputs have already been defined as follows:

$$\begin{aligned}u_x &= \cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi) \\ u_y &= \cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi)\end{aligned}\tag{3.54}$$

Since u_x and u_y have very similar equation structure, it is enough to get a solution for one of them. Let's continue with definitions of tracking errors.

$$\begin{aligned} e_9 &= x_d - x &= x_{9d} - x_9 \\ e_{10} &= x_{10} - x_{10v} &= x_{10} - \dot{x}_{9d} - \alpha_9 e_9 \end{aligned} \quad (5.37)$$

" $x - y$ " motion equations, that are already represented in Equation 3.55, can also be further simplified with a state-space form.

$$\begin{aligned} \dot{x}_9 &= \dot{x} = x_{10} \\ \dot{x}_{10} &= \ddot{x} = -\frac{U_1}{m} u_x \\ \dot{x}_{11} &= \dot{y} = x_{12} \\ \dot{x}_{12} &= \ddot{y} = -\frac{U_1}{m} u_y \end{aligned} \quad (5.38)$$

Defining the Lyapunov function as $V(e_9, e_{10}) = \frac{1}{2}(e_9^2 + \frac{1}{2}e_{10}^2)$, the derivative can be investigated.

$$\begin{aligned} \dot{V}(e_9, e_{10}) &= e_9 \dot{e}_9 + e_{10} \dot{e}_{10} \\ &= e_9(-e_{10} - \alpha_9 e_9) + e_{10}(\dot{x}_{10} - \ddot{x}_{9d} - \alpha_9 \dot{e}_9) \\ &= e_9(-e_{10} - \alpha_9 e_9) \\ &\quad + e_{10}\left(-\frac{U_1}{m} u_x - \ddot{x}_{9d} + \alpha_9(e_{10} + \alpha_9 e_9)\right) \end{aligned} \quad (5.39)$$

It can be easily shown that the virtual control input

$$u_x = \frac{m}{U_1}(-e_9 + \alpha_{10} e_{10} + \alpha_9(e_{10} + \alpha_9 e_9)) \quad (5.40)$$

gives us $\dot{V}(e_9, e_{10}) < 0$, yielding stability with $\alpha_9 > 0$ and $\alpha_{10} > 0$.

For the case of u_y , the control input can be written directly using the similarity to u_x as follows:

$$u_y = \frac{m}{U_1}(-e_{11} + \alpha_{12} e_{12} + \alpha_{11}(e_{12} + \alpha_{11} e_{11})) \quad (5.41)$$

Here, $\alpha_{11} > 0$ and $\alpha_{12} > 0$ conditions should also be satisfied.

With Equations 5.40 and 5.41, virtual control inputs that transforms x and y reference values into control action have been obtained. This action should result in the desired ϕ and θ angles so that attitude system can regulate these to ensure position stabilization. Since u_x and u_y have been obtained, they can be used in Equation 3.54 to obtain ϕ_d and θ_d . This trigonometric solution have been given in [40] as follows.

$$\begin{aligned}\phi_d = x_{1d} &= \arctan\left(\frac{u_x \sin(x_5) - u_y \cos(x_5)}{(1 - \sin^2(x_5)u_x^2 + 2 \cos(x_5)u_x u_y \sin(x_5) + u_y^2 \sin^2(x_5) - u_y^2)^{\frac{1}{2}}}\right) \\ \theta_d = x_{3d} &= \arcsin\left(\frac{u_x \cos(x_5) + u_y \sin(x_5)}{(1 - \sin^2(x_5)u_x^2 + 2 \cos(x_5)u_x u_y \sin(x_5) + u_y^2 \sin^2(x_5) - u_y^2)^{\frac{1}{2}}}\right)\end{aligned}\quad (5.42)$$

With the usage of Equation 5.42, desired Euler angles can be found for the attitude controllers to be able to follow $x - y$ position references.

5.2.3 Attitude Performance of Backstepping Controller on Test Bench

Backstepping design is based on the Lyapunov Stability. This process yields a controller with constant parameters, which guarantees stability if the parameters are greater than zero. In this process, thus, change of the constants does not disturb the stability (small values may not bring stability due to simplified model and unknown disturbances), but it affects the performance of the system. For this reason, the parameters should be chosen in a way that is best for the quadrotor response.

For the roll angle dynamics of the quadrotor, an example of parameter change is given in Figure 5.19. By increasing two controller constants, the response of the system can be improved.

However, this is not always true in the real application field. Backstepping controller becomes more sensitive to errors and noise when the constants are increased. For the test bench experiments, Backstepping controller parameters are selected as $\alpha_1 = 4.0$ and $\alpha_2 = 4.0$ for pitch, roll and yaw dynamics. PWM values commanded to the first and second motors are shown in Figure 5.20. As it can be seen from the figure, controller outputs are noisy, which blocks the increase of controller parameters. This is not the case in PID control scheme because of the differences among the controllers. PID control directly depends on the error term. For a rotation axis, there are just angle

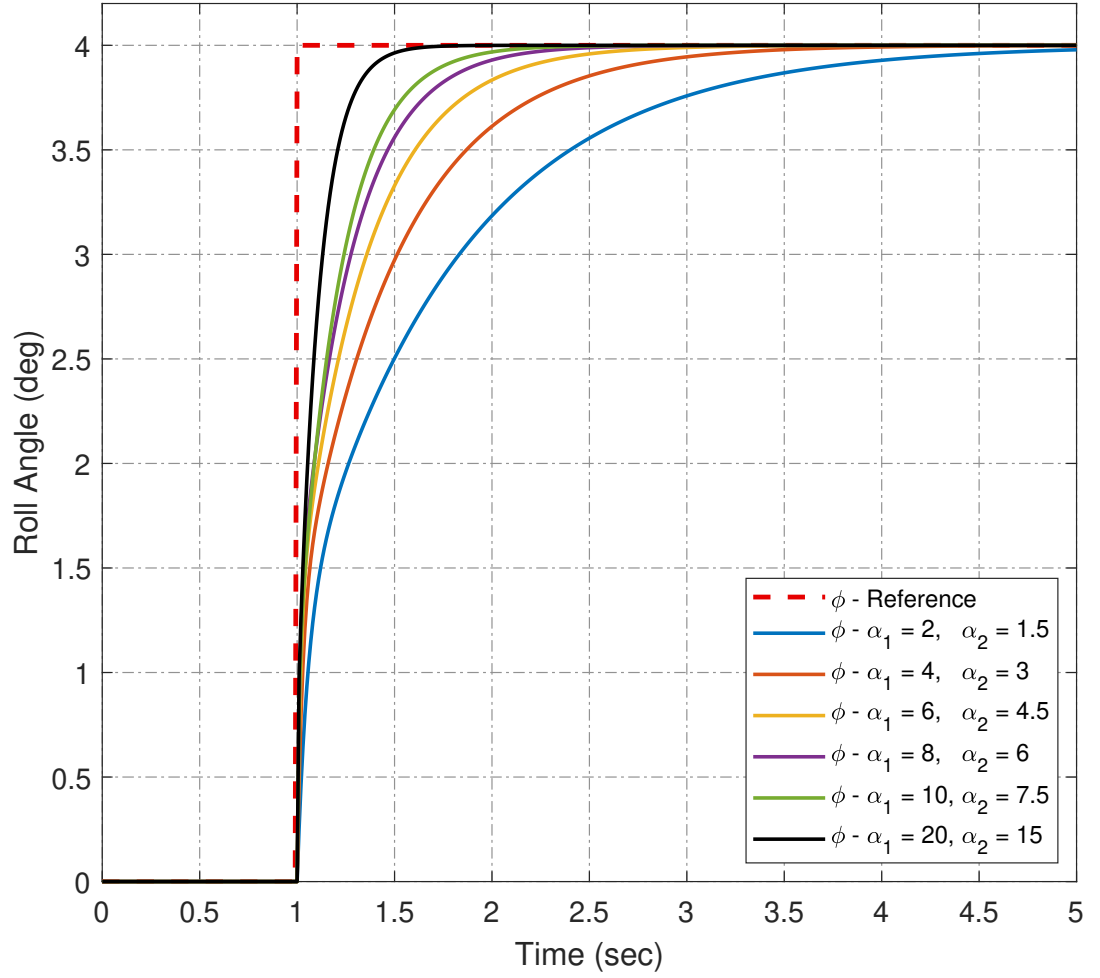


Figure 5.19: Response of roll angle to given step input for changing controller parameters

and body rate errors fed into the controller. On the Backstepping however, the control scheme differs based upon the selection of Lyapunov function. This function can be simple enough to yield stability or a complex representation, which directly affects the output. Besides, Backstepping includes cross-coupling effects and gyroscopic terms that comes from the modeling part. These terms are included in the control scheme to improve the behavior of the quadrotor, but this also increases the fluctuations in the output since the controller try to balance more phenomenon. This situation restricts the increase of control parameters.

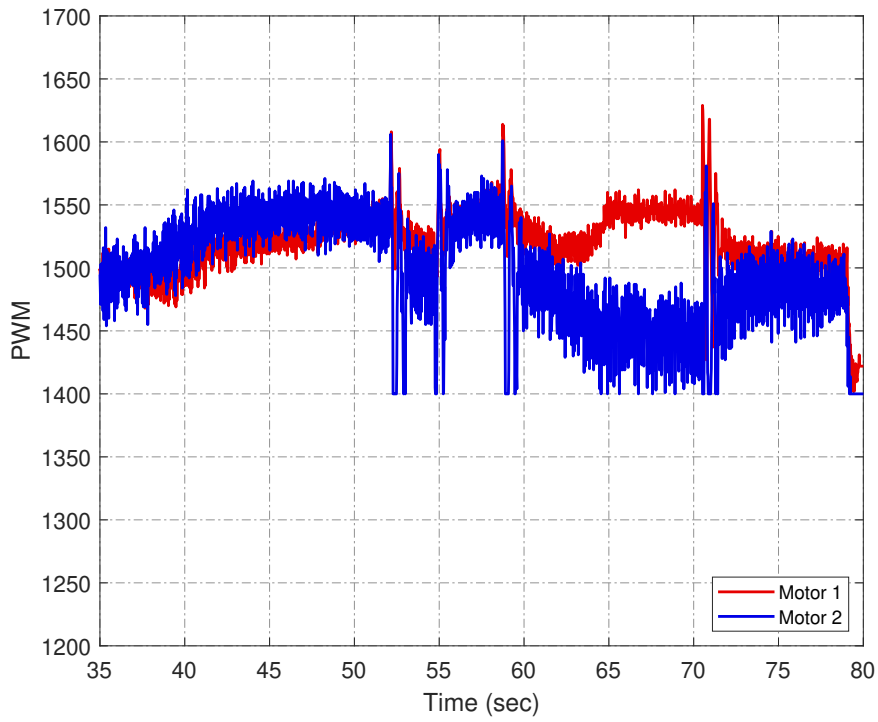


Figure 5.20: Commanded PWM to motors by backstepping controller

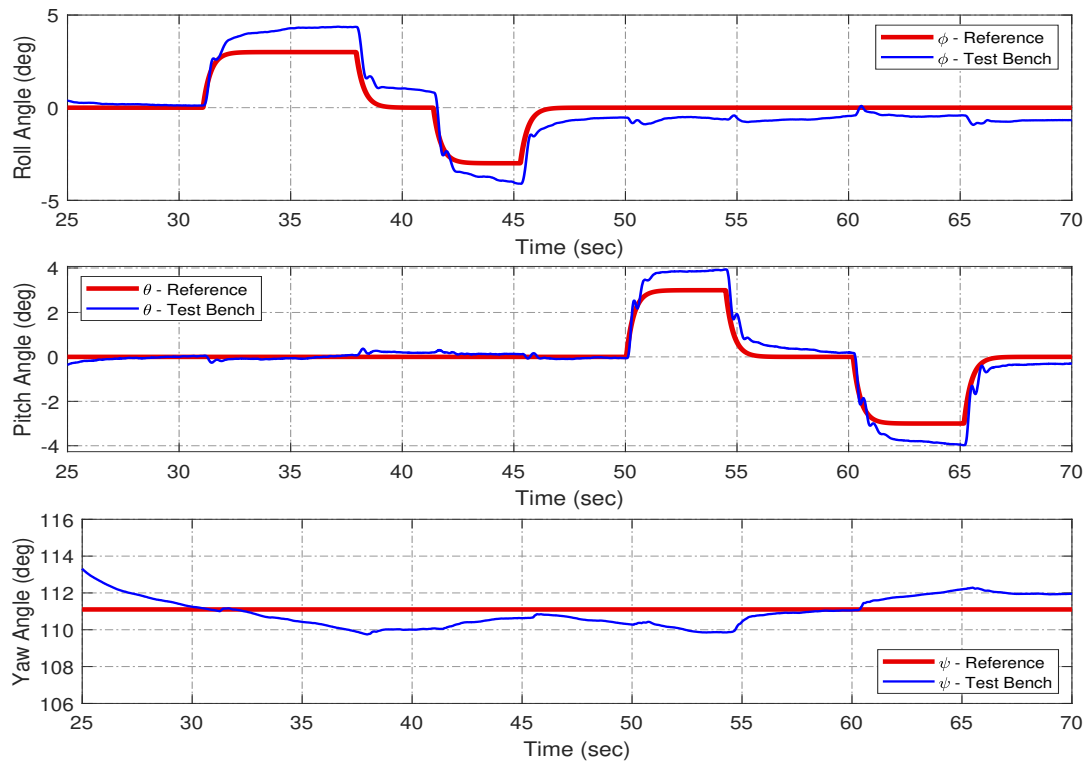


Figure 5.21: Response of quadrotor to stepwise inputs given in pitch and roll axes

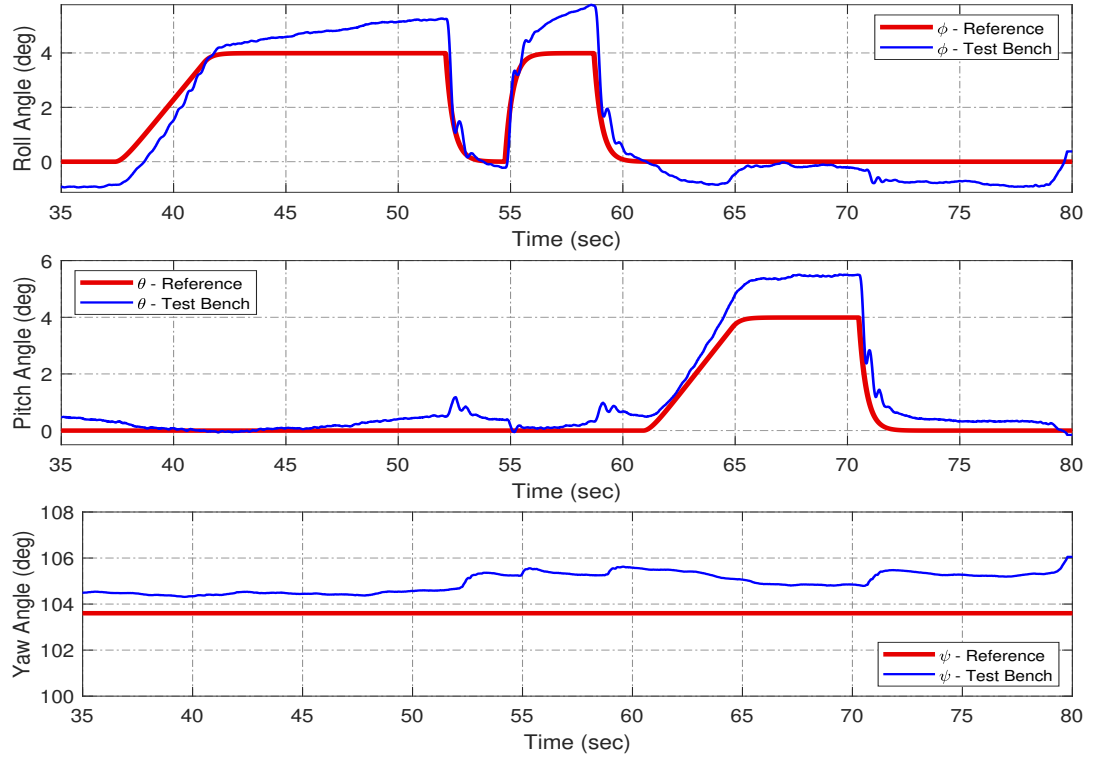


Figure 5.22: Response of quadrotor to ramp-wise inputs given in pitch and roll axes

As already done in PID controller tests, stepwise and ramp-wise inputs are given to pitch and roll channels of the quadrotor for the tests of the Backstepping controller. As it can be seen from Figures 5.21 and 5.22, there are high overshoots exist on both channels. This might be dependent on the far center of gravity connection of the quadrotor to the test setup, which was also seen in PID controller trials. However, main problem at this issue is the fact that there is a noticeable steady-state error after a big overshoot. Backstepping controller is like a PD control strategy and does not contain an integral term. Researchers have added integral terms to the Backstepping, which is called Integral Backstepping to eliminate the steady-state error. With this scheme, accurate position control cannot be reached due to weak attitude design. Thus, position control experiments were conducted with PID controller.

5.3 Controller Applications on Quadrotor

After the design steps of attitude and position controllers, real flight test were performed with the quadrotor platform. Pixhawk and Raspberry Pi boards were used as specified in the Chapter 4, and the position control was performed with a reference of ArucoMarker. The controller diagram on Pixhawk autopilot board is shown in Figure 5.23.

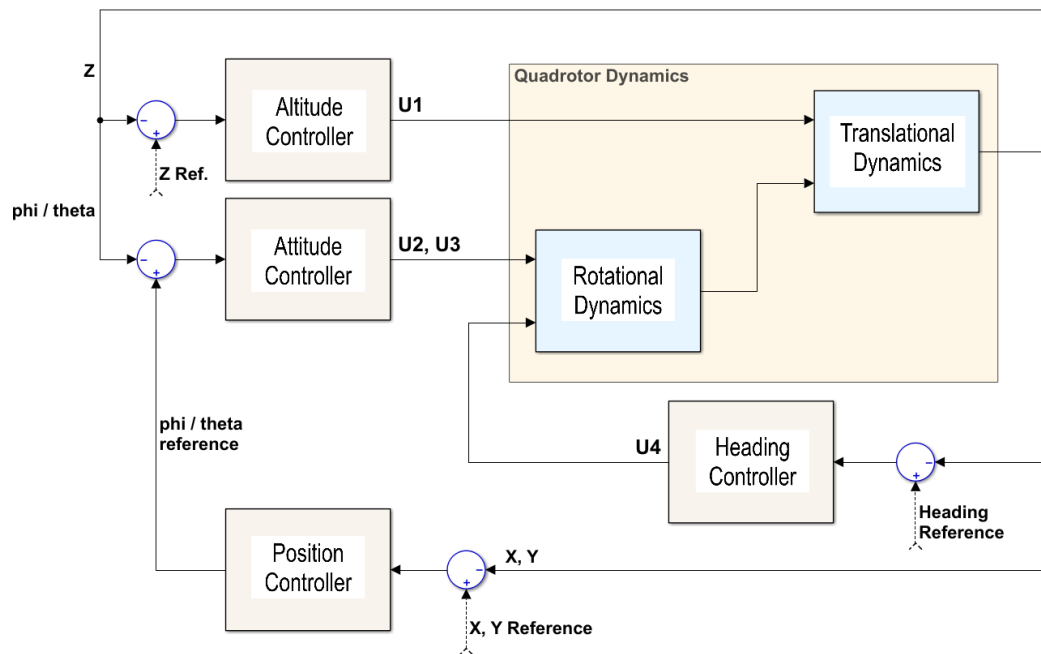


Figure 5.23: Control structure used for quadrotor stabilization

Yaw attitude controller was designed to hold the initial heading of the quadrotor. To rotate the quadrotor in the direction of the marker, yaw angle reference is managed by controlling heading difference between quadrotor and marker. This diagram is shown in Figure 5.24. The reference value is filtered to slow-down so that quick yaw responses can be eliminated. In this structure, quadrotor holds the last valid yaw angle value if the marker goes out of the camera focus.

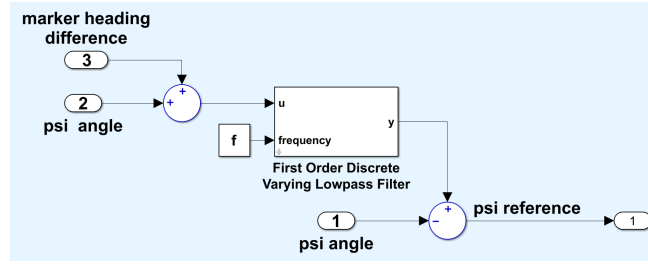


Figure 5.24: Yaw angle reference management during marker detection

In the real flights, it was observed that the frame vibrates less compared to the test bench situation, thus the controller gains were increased. Final gains of the PID Attitude Controllers are listed below:

- ϕ motion: $K_{p,\phi} = 20.0$, $K_{i,\phi} = 3.0$, $K_{p,p} = 3.0$, $K_{i,p} = 3.0$, $K_{d,p} = 0.025$
- θ motion: $K_{p,\theta} = 18.0$, $K_{i,\theta} = 3.0$, $K_{p,q} = 3.0$, $K_{i,q} = 4.0$, $K_{d,q} = 0.03$
- ψ motion: $K_{p,\psi} = 6.0$, $K_{p,r} = 0.2$, $K_{i,r} = 0.1$

As an example result of the new controllers, roll angle ϕ step response can be given as shown in Figure 5.25.

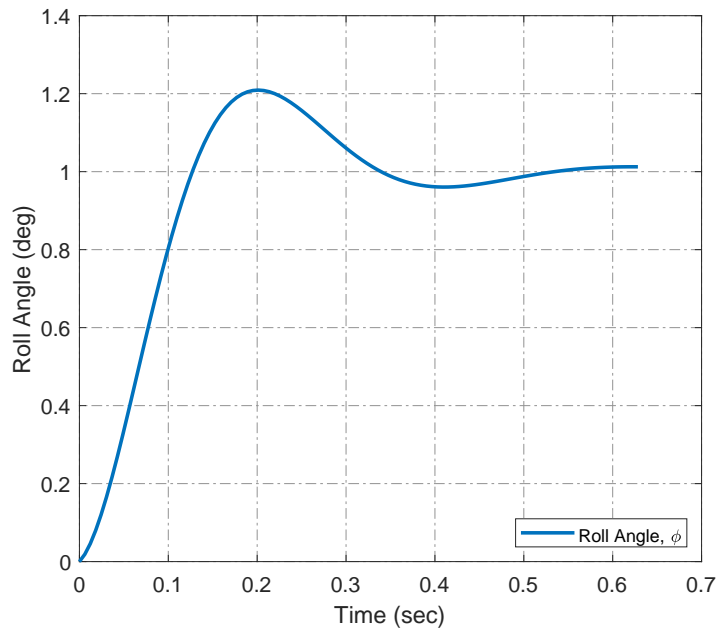


Figure 5.25: Roll angle step response

With the selected gains, the system will have a bandwidth of $3.56Hz$ as it can be seen from the Bode Plot (Figure 5.26) of the closed loop system.

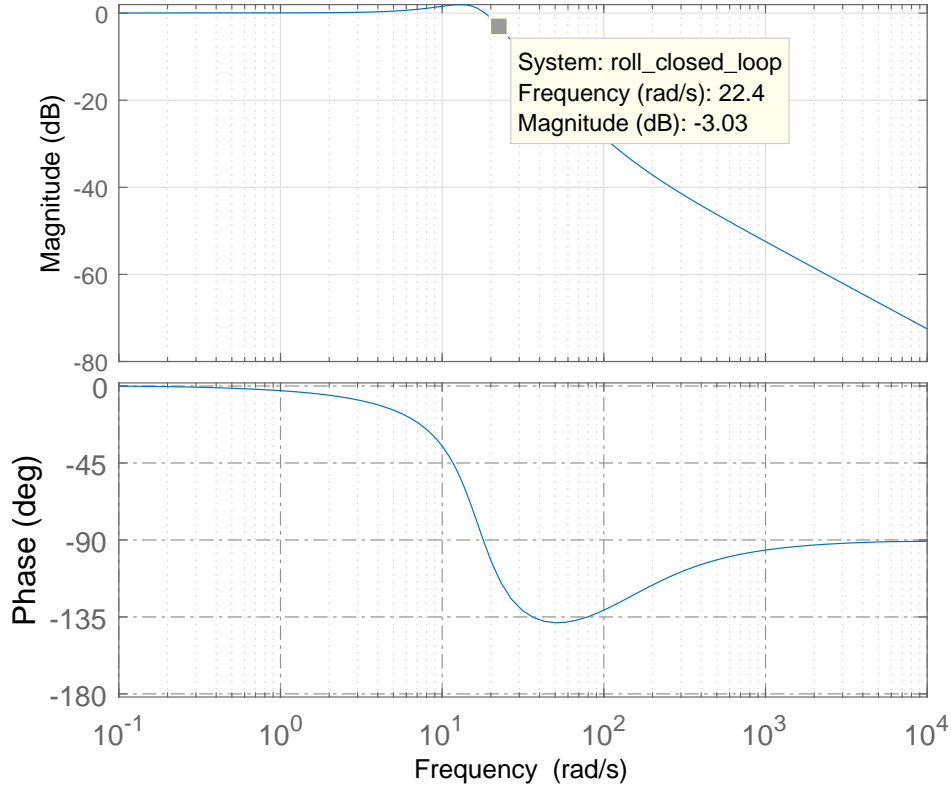


Figure 5.26: Closed loop roll angle Bode plot

For the altitude and position control, selected PID gains can be listed as follows:

- x motion: $K_{p,x} = 0.15$, $K_{p,v_x} = 1.5$, $K_{i,v_x} = 1.0$
- y motion: $K_{p,y} = 0.10$, $K_{p,v_y} = 1.0$, $K_{i,v_y} = 1.0$
- z motion: $K_p = 6.0$, $K_i = 8.0$, $K_d = 4.0$

To test attitude controller performance of the quadrotor, pilot inputs are given from the Taranis transmitter. In addition to that, altitude controller was used to hold the quadrotor height. In Figure 5.27, test results of the pilot controlled quadrotor are demonstrated.

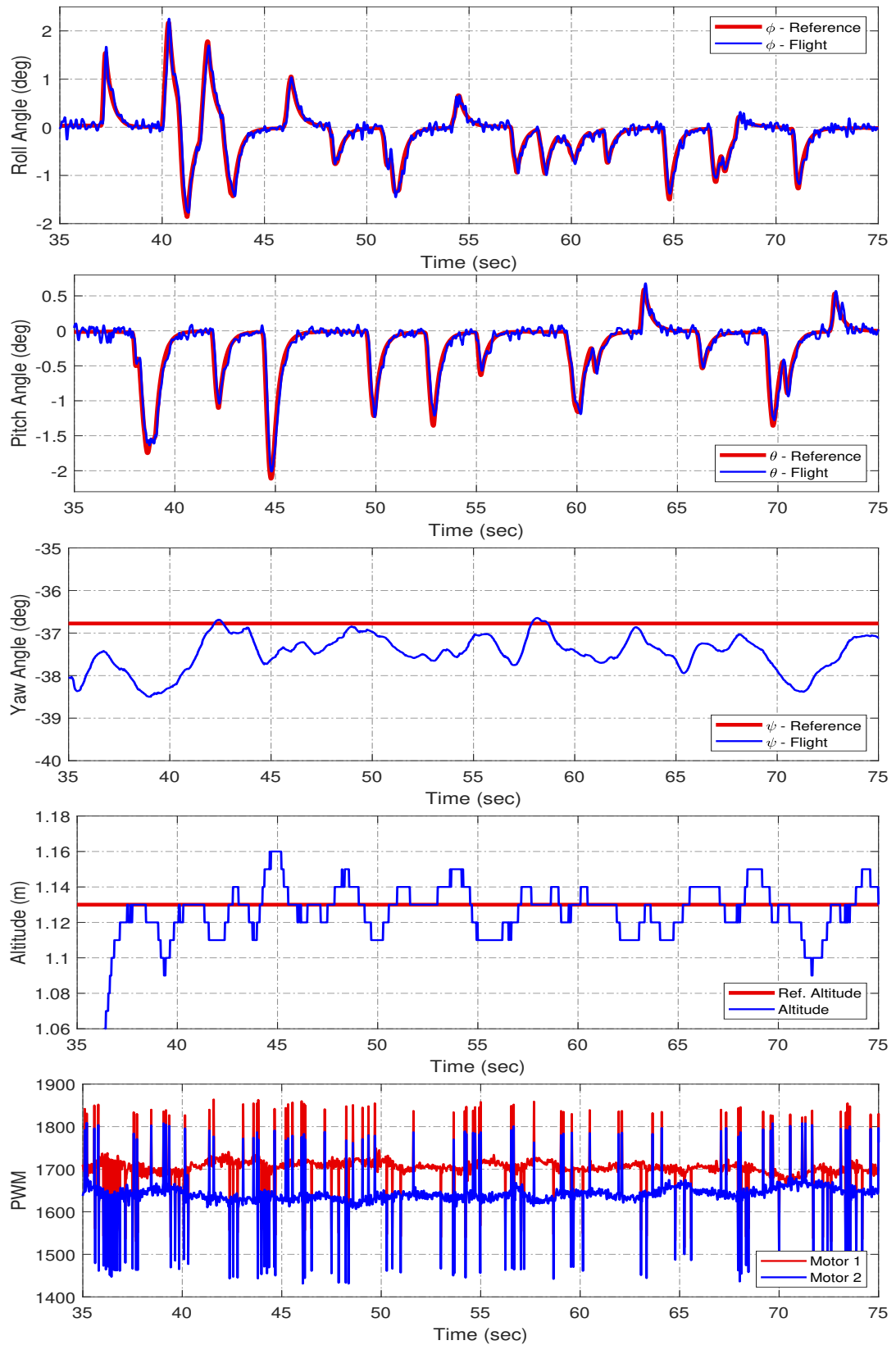


Figure 5.27: Response of quadrotor system to given pilot inputs in pitch and roll axes

As it can be seen from the results, quadrotor can track angle references with a good performance. In addition to attitude, altitude control performance is satisfactory with a resolution of $\pm 3cm$. This accuracy is acceptable since the Lidar output resolution is $\pm 1cm$.

After this experiment, position control performance of the system was observed. Two experiment results will be presented in this section. Both flights were conducted in a room of $25m^2$ area by placing the marker at the middle of the space. In both of them, quadrotor was expected to detect the marker at the ground and position hold over marker was aimed. For the first position hold trial, commanded and obtained velocity in x and y directions are represented in Figure 5.28. As it can be seen from the figures, obtained velocity data is noisy, and the velocity controller cannot track commanded reference like it was in the attitude controllers.

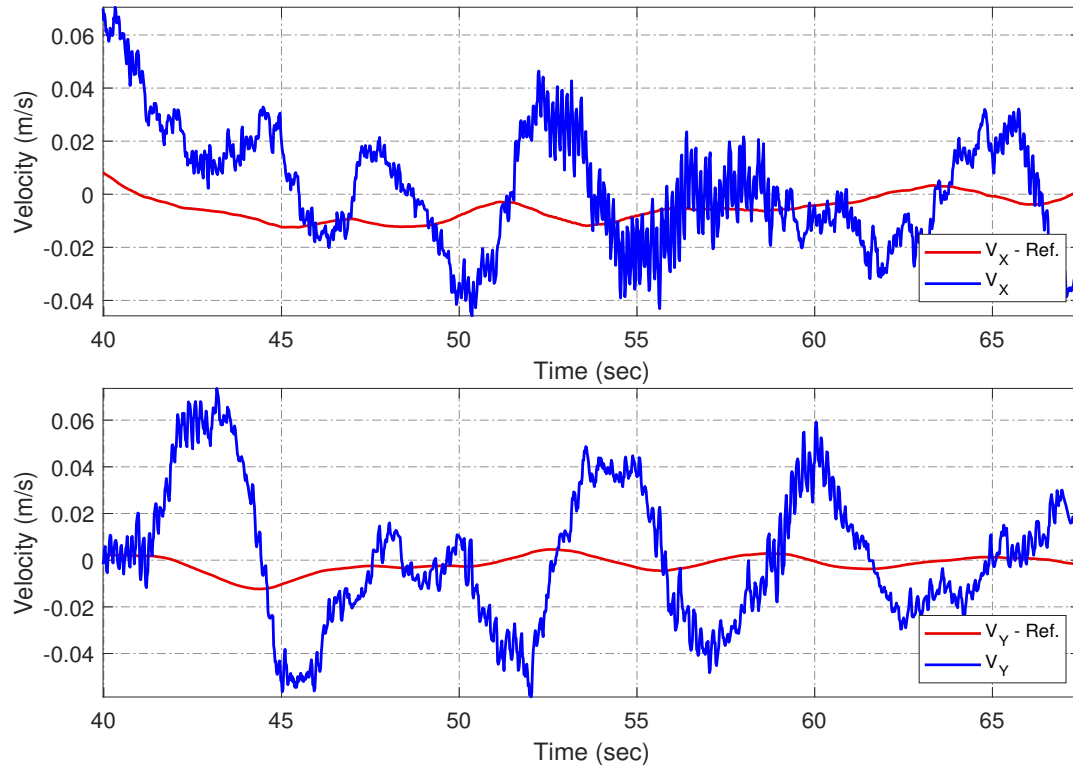


Figure 5.28: Quadrotor velocity in x/y directions during position hold over marker

In Figure 5.29, attitude and altitude results of the quadrotor are demonstrated for the first position hold demonstration. Since the velocity data is noisy, velocity controller output reference is noisy, which brings vibrations in the Euler angles by decreasing

flight performance. On the other hand, yaw angle and altitude responses are satisfactory.

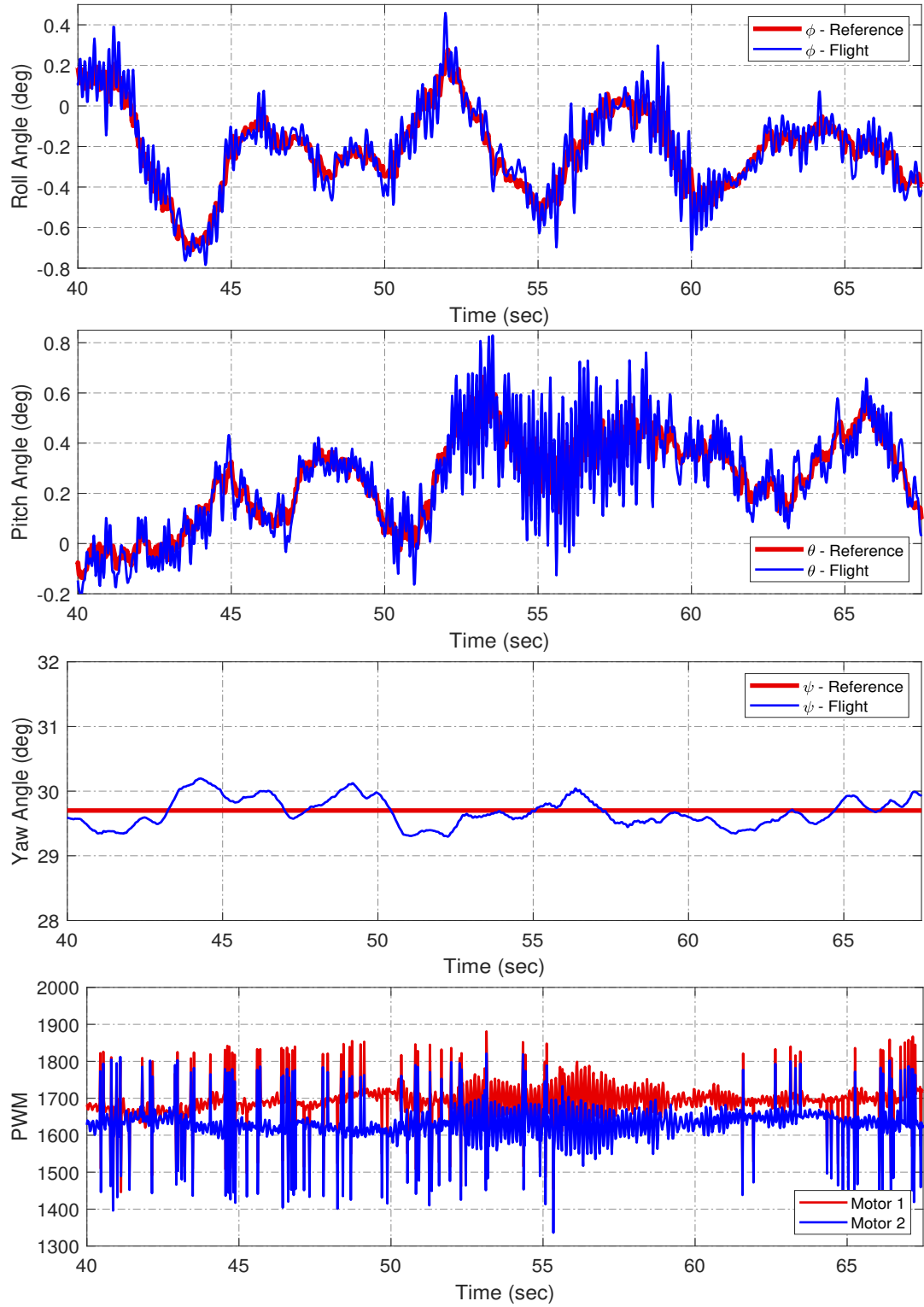


Figure 5.29: Response of quadrotor system during position hold over marker

As the main result of this trial, position trajectory of the performed flight is demonstrated in Figure 5.30. It can be seen that the quadrotor cannot perform an accurate position control and walk in a square of 15cm side-length.

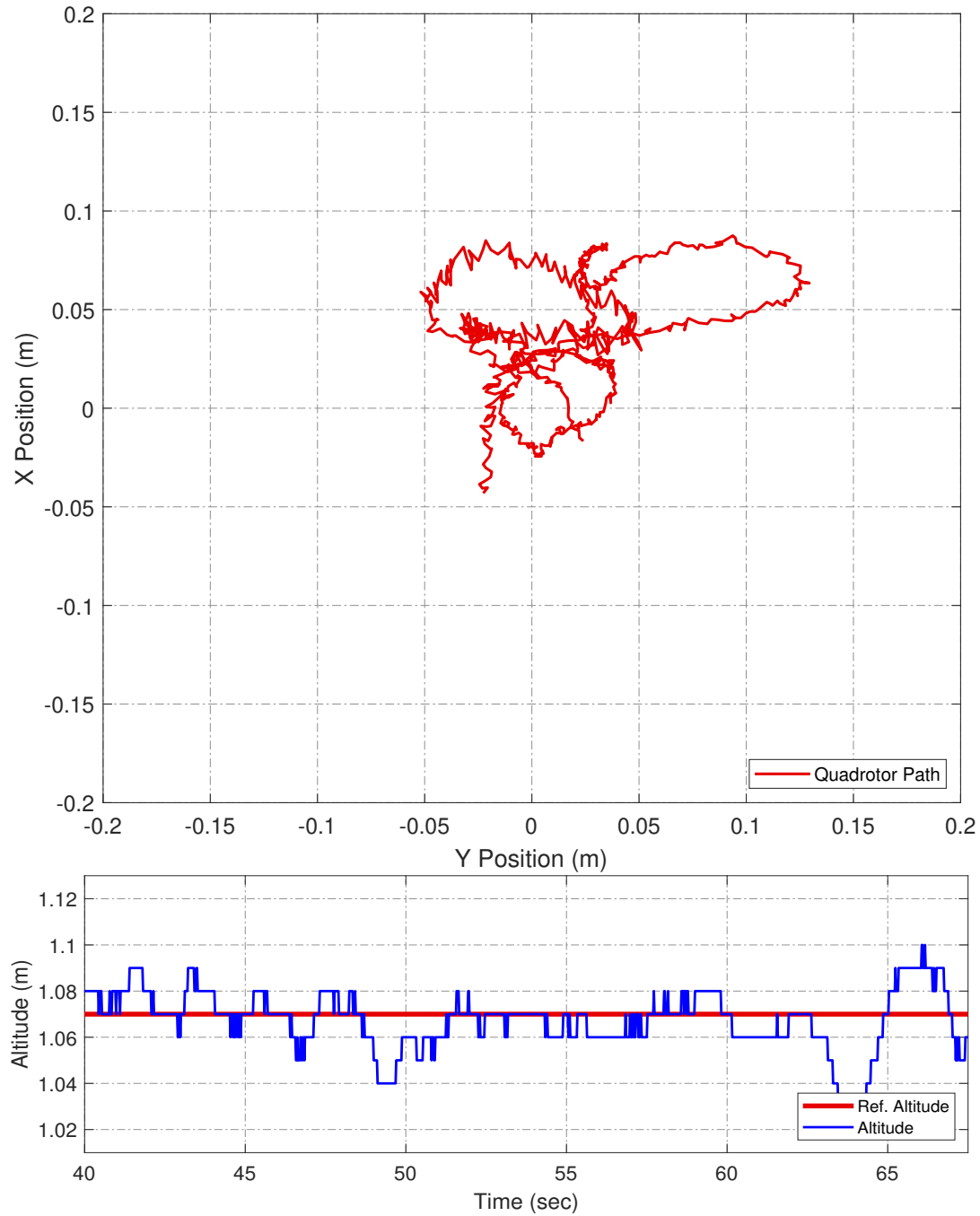


Figure 5.30: Position and altitude of quadrotor during position hold over marker

At this point, some statistical data can be represented for each motion axis. In Figure 5.31, minimum, maximum and mean values are represented on the plots. There is also the standard deviation value that has been placed in the text boxes.

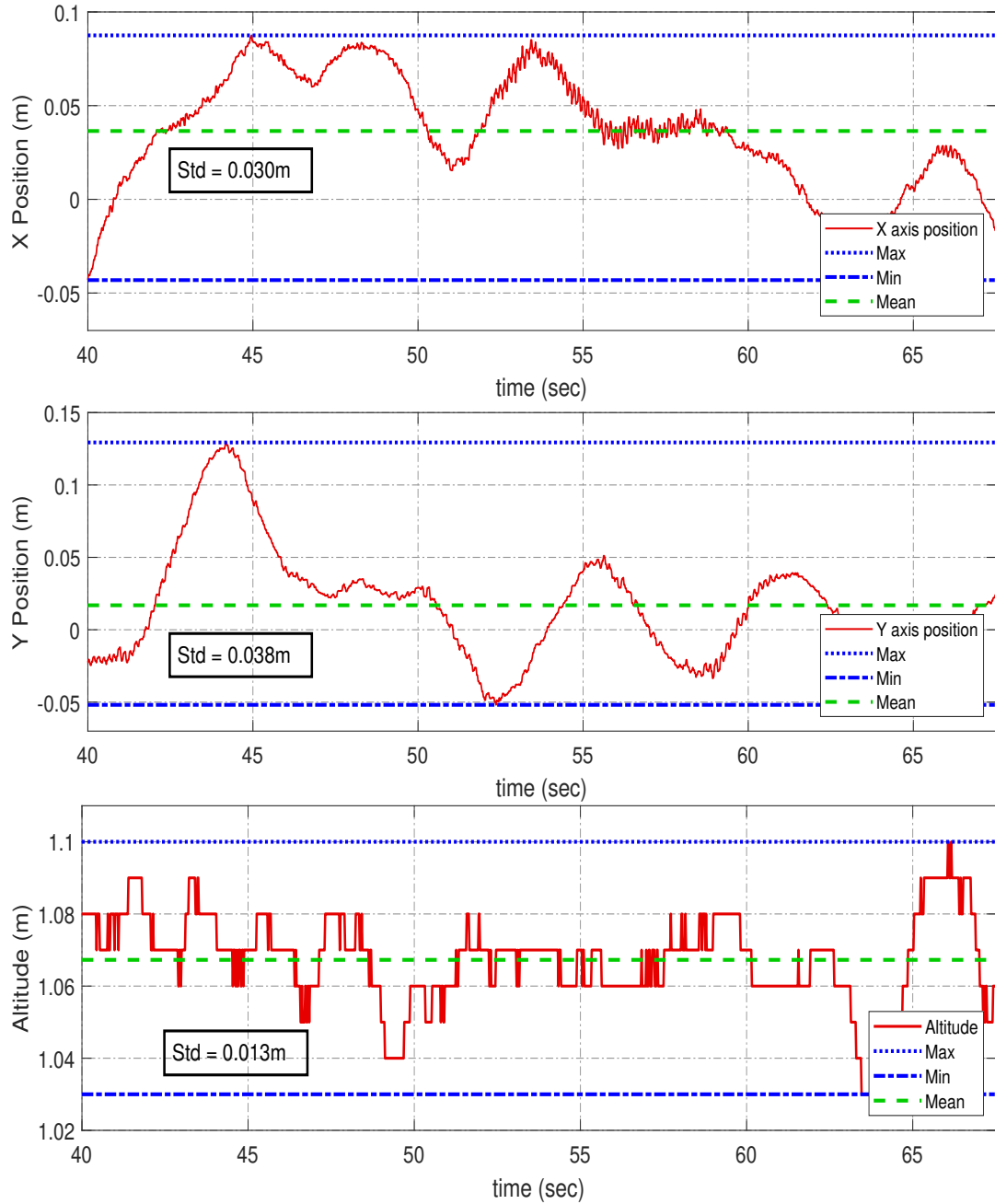


Figure 5.31: Statistical data for the quadrotor that performs position hold over marker

As it can be seen from the plots, deviation value obtained for the z-axis is relatively different compared to the one of positions, that is because altitude and position mo-

tions are governed by different sensor sources being as Lidar and Camera.

For the second position hold trial, commanded and obtained velocity in x and y directions are represented in Figure 5.32. As it can be seen from the figures, obtained velocity data is less noisy compared to the first flight attempt. However, controller performance is not satisfactory.

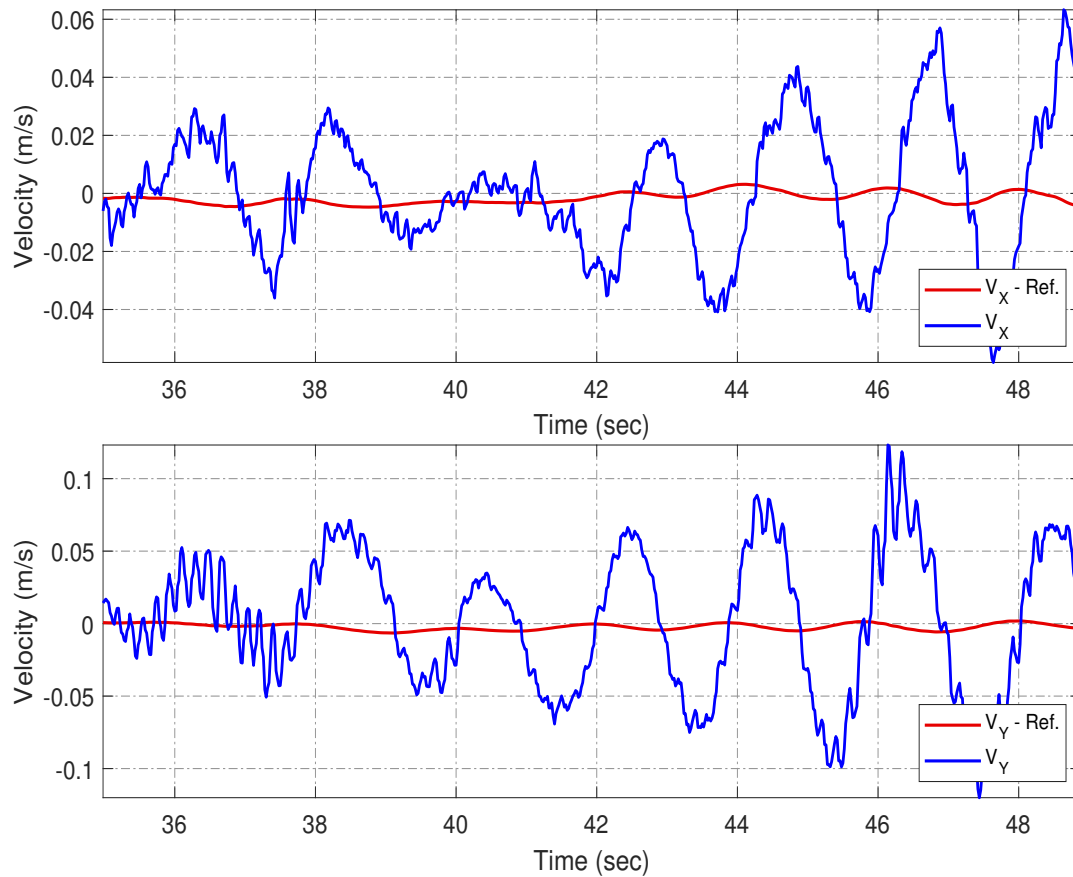


Figure 5.32: Quadrotor velocity in x/y directions during position hold over marker (second flight)

In Figure 5.33, attitude and altitude results of the quadrotor are demonstrated for the second position hold demonstration. Since the velocity data is less noisy, better attitude performance is observed compared to the first flight.

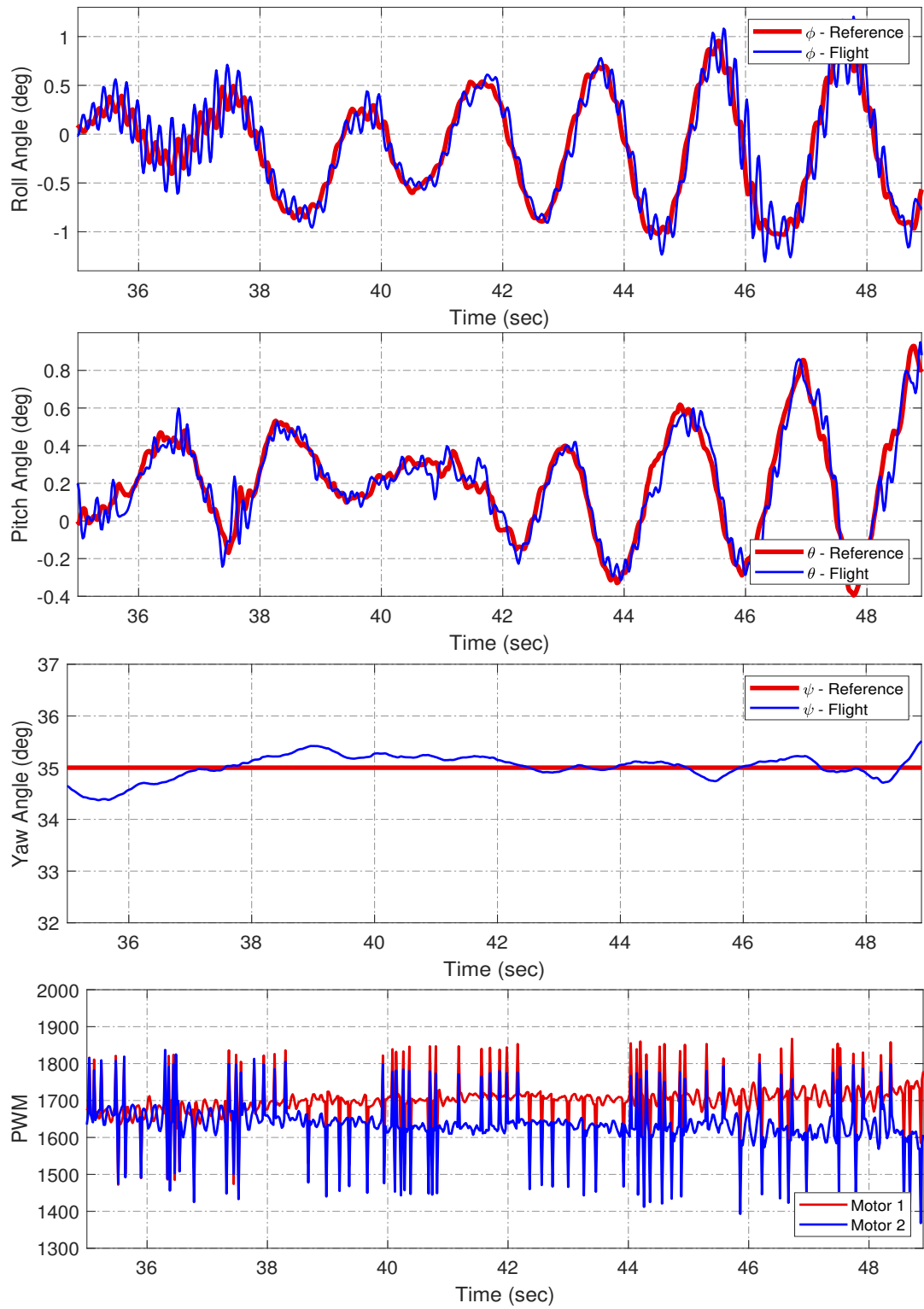


Figure 5.33: Response of quadrotor system during position hold over marker (second flight)

As the root result of second trial, position trajectory of the performed flight is demonstrated in Figure 5.34. It can be seen that the quadrotor can perform a relatively satisfactory performance by walking in a circle of 10cm diameter.

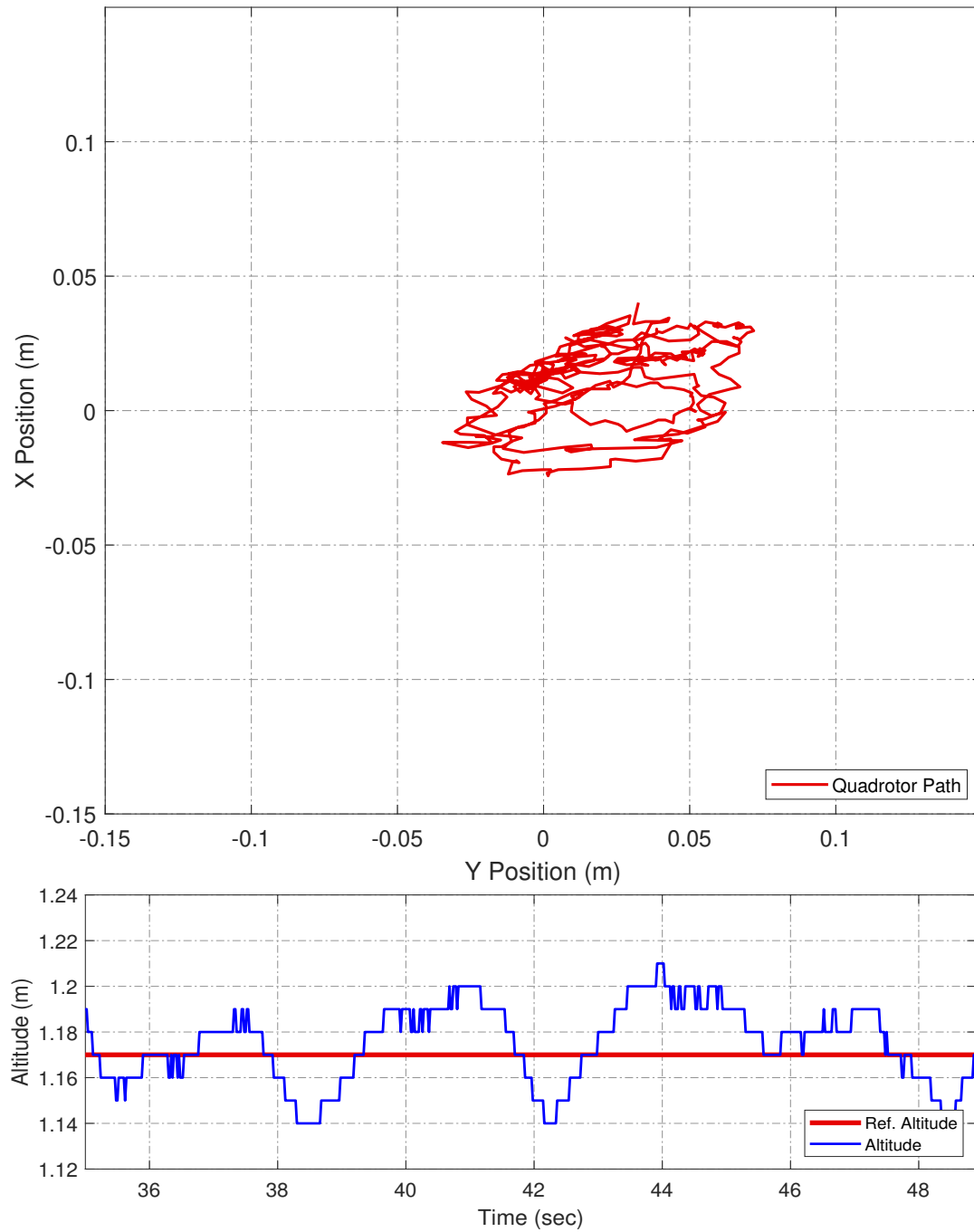


Figure 5.34: Position and altitude of quadrotor during position hold over marker (second flight)

For the second position hold trial, statistical data is also represented as it can be seen in Figure 5.35. In this trial, standard deviation of the X motion is the lowest among the axes. This flight demonstrates better results of position hold capability.

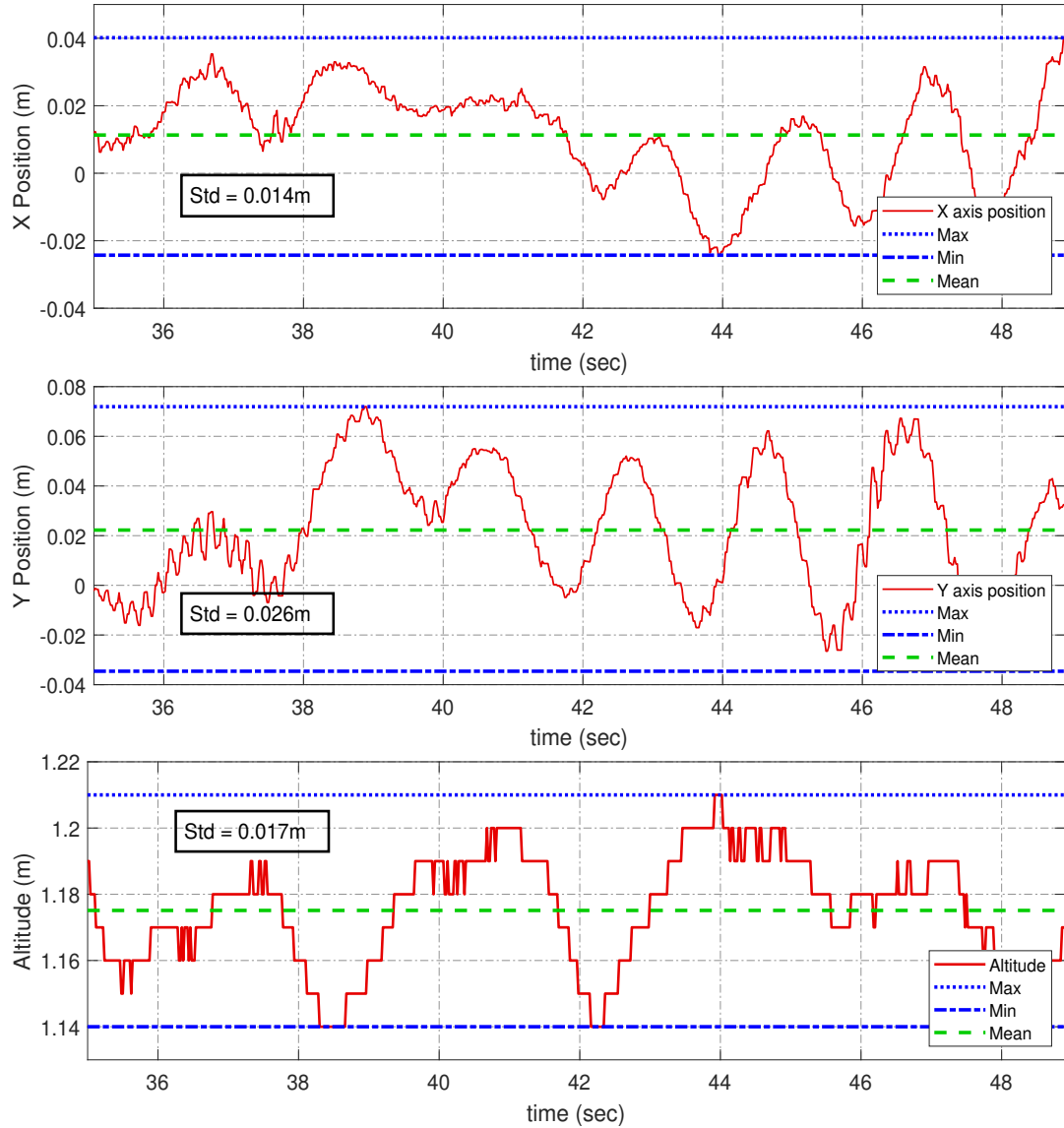


Figure 5.35: Statistical data for the quadrotor that performs position hold over marker (second flight)

In the next flight experiment, a sinusoidal input has been given for the X axis motion so that the path following capability of the quadrotor can be represented. In Figure 5.36, resultant position and altitude response of the quadrotor can be seen.

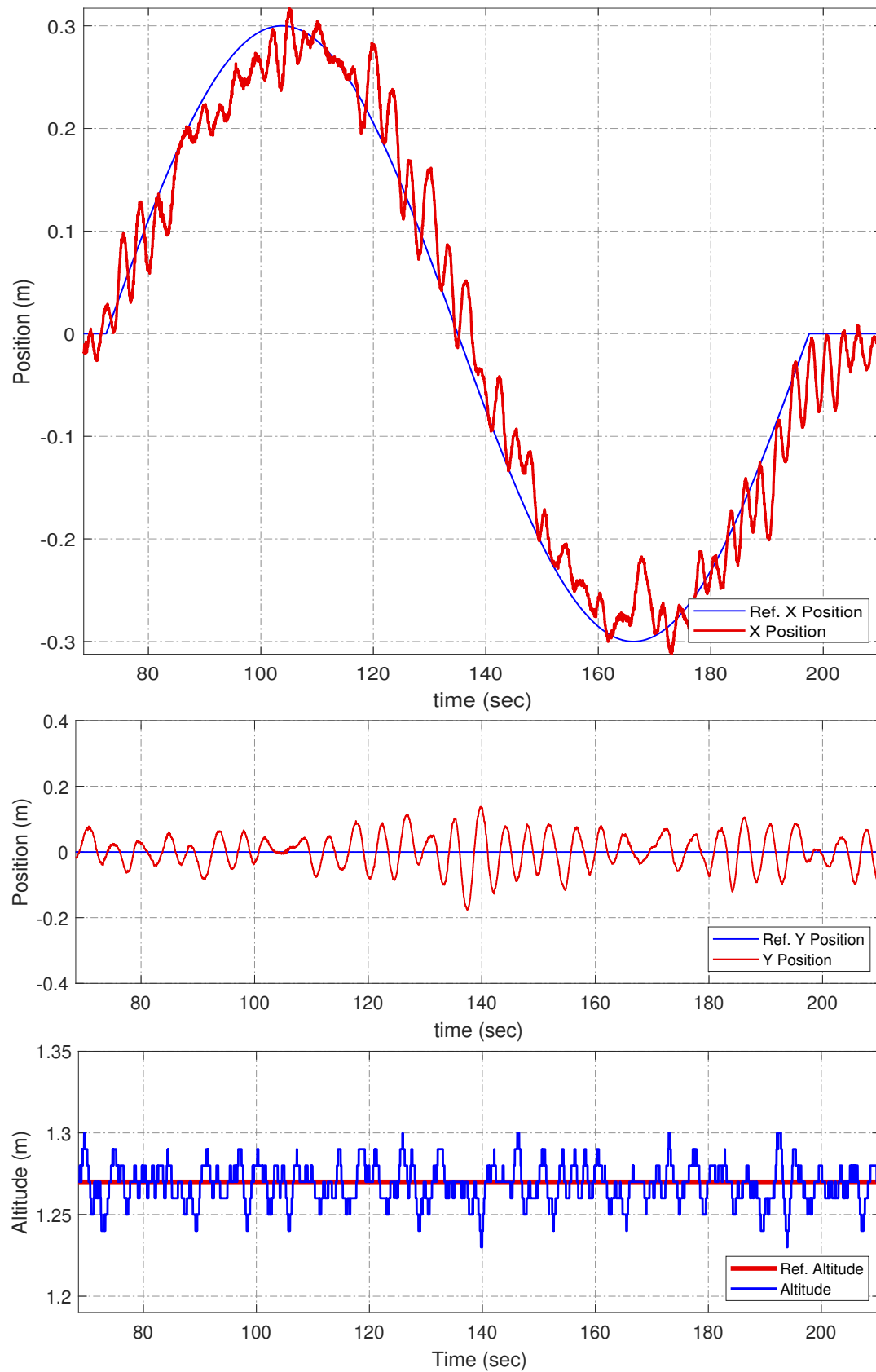


Figure 5.36: Position and altitude of quadrotor when a sinus reference is commanded for x-position

As it can be seen from the statistical data, Figure 5.37, commanded shape is also visible because of the noticeable delay formed in the velocity control loop.

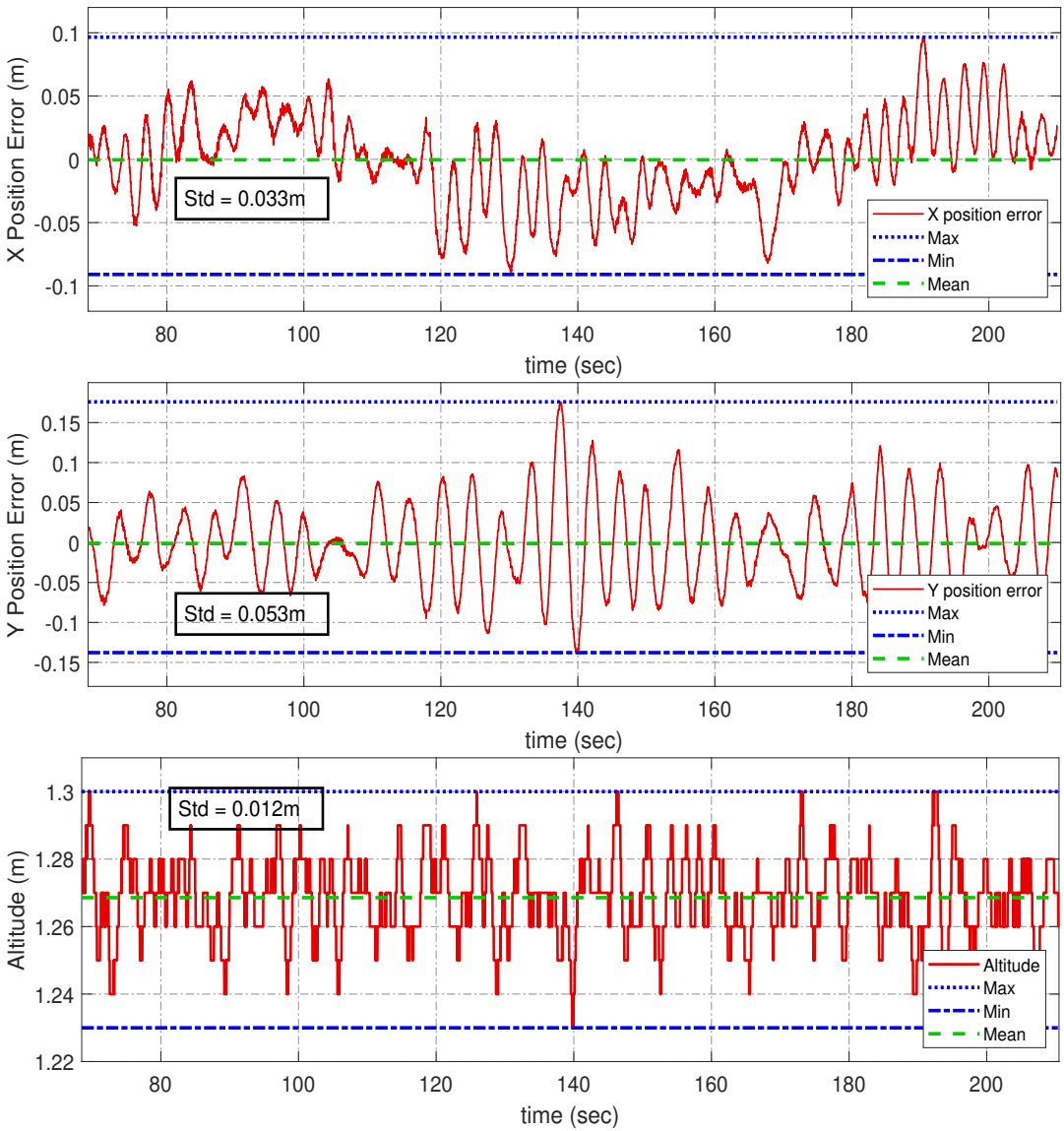


Figure 5.37: Statistical data for the quadrotor when a sinus reference is commanded for x-position

As the final flight trial, a square orbit was commanded for the quadrotor for which position and altitude data can be seen in Figure 5.38.

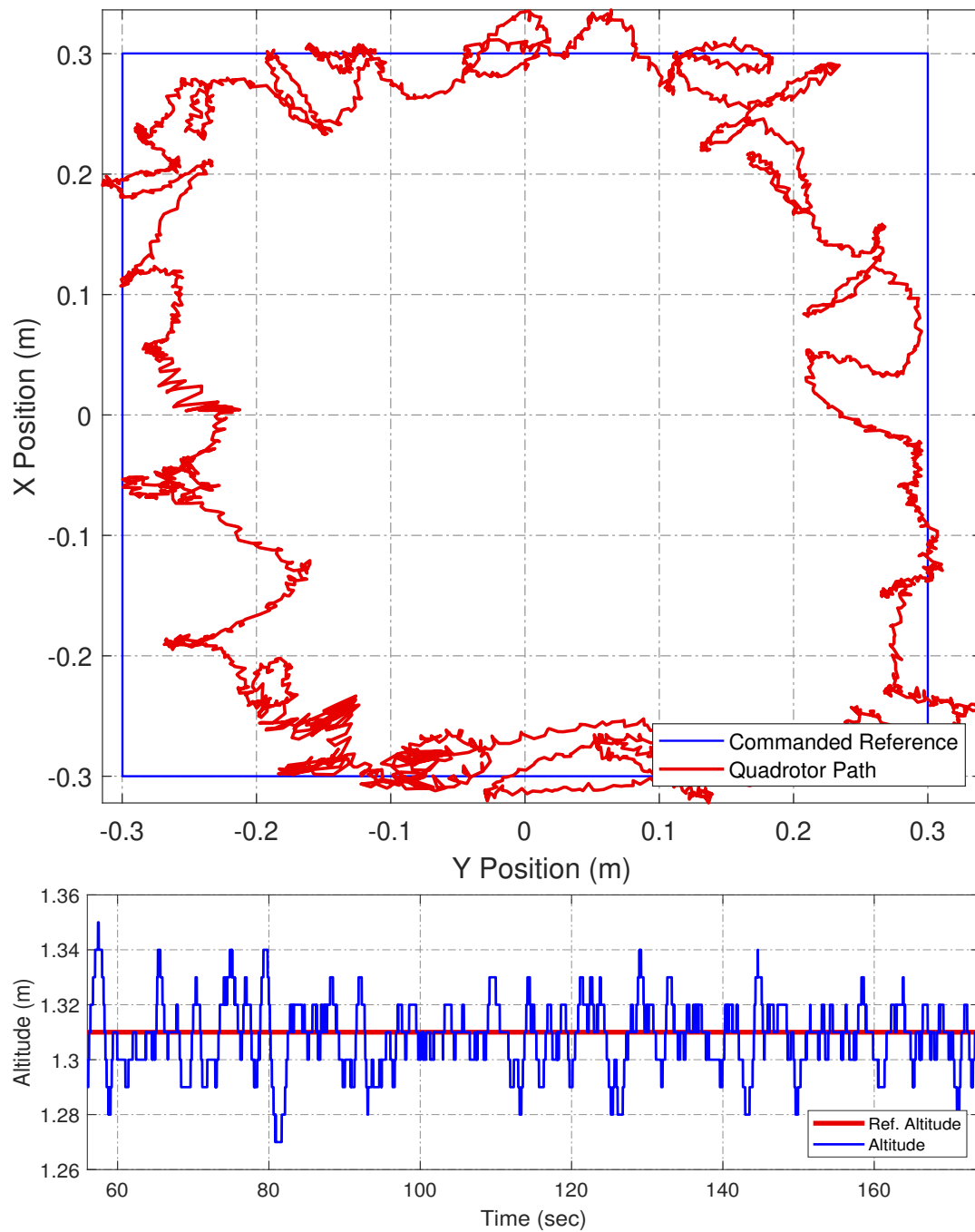


Figure 5.38: Position and altitude of quadrotor when a square path reference is commanded

As it can be seen from the plots, quadrotor is capable of following a closed orbit and able to come back to the start point. However, the accuracy of the formed path is not satisfactory for which the primary cause is the ineffective use of velocity data.

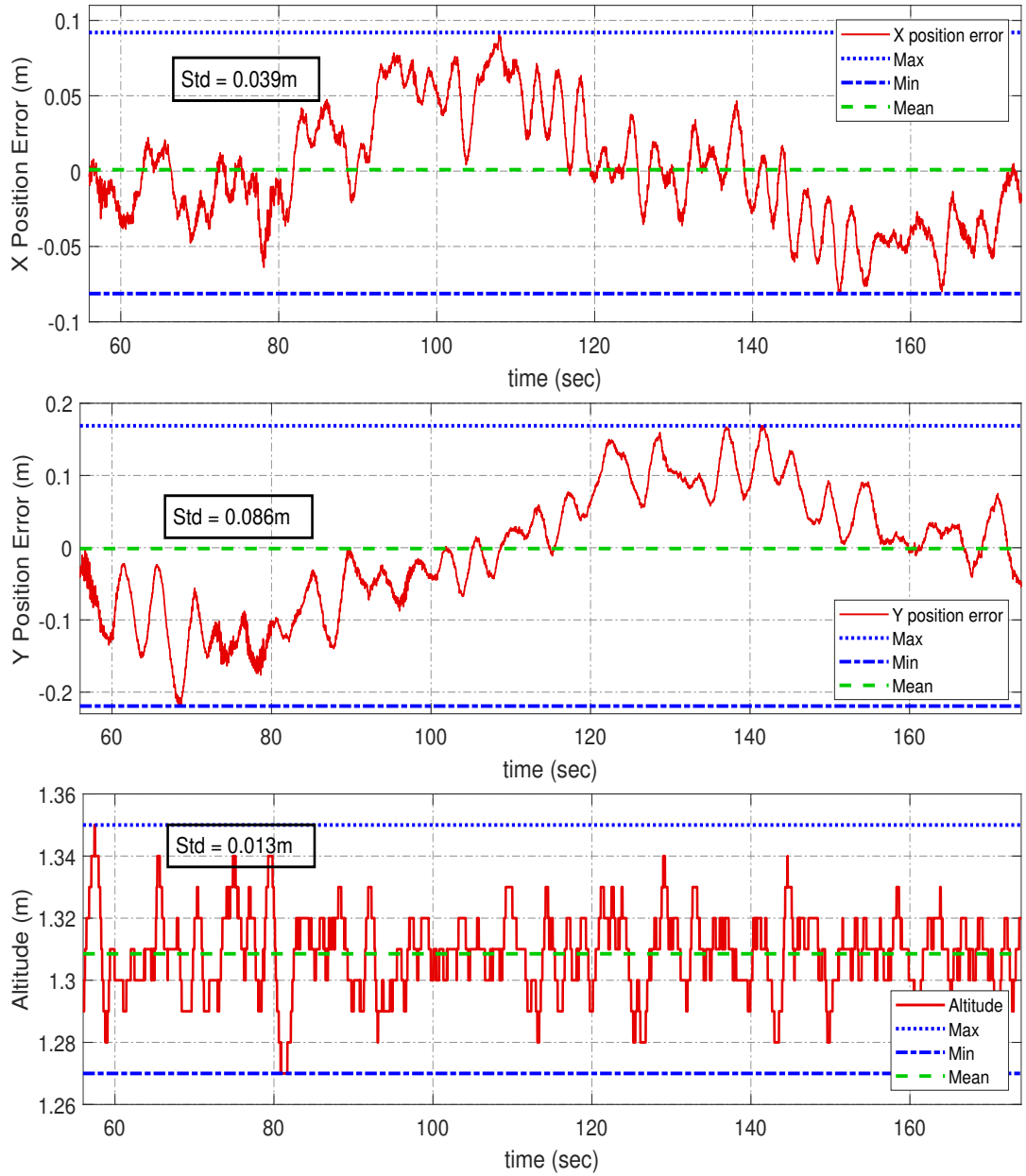


Figure 5.39: Statistical data for the quadrotor when a square path reference is commanded

As it can be seen from Figure 5.39, this accuracy has also been reflected in the statistical data. Since the reference line rotates sharply from the corners of the square, the quadrotor has already turned before the corners and this error is reflected in the plots. The reason for the apparent difference between the X and Y axes position follow capabilities is that different inertia values were obtained for pitch and roll channels, and

therefore the roll axis gives a worse response.

For the final flight experiment, a photo of the flying quadrotor is also demonstrated in Figure 15.40.



Figure 5.40: Photo of the quadrotor during path follow

CHAPTER 6

CONCLUSION

The main focus of this thesis was to test the aerial printing capability of a quadrotor system, that is capable of image processing with a monocular camera. A quadrotor platform was created for this purpose. The equipment of the platform was selected by monitoring the flight time and payload concerns. Pixhawk and Raspberry Pi boards have been used as the main components of the control system. The Pixhawk board have included the control model, while the Raspberry Pi have operated logging and image processing algorithms. Simulink environment have been preferred both for controller design and implementation of scripts running on the boards.

Before the design of the controllers, the mathematical model of the quadrotor dynamics was created. In order for the model to reflect the actual system, the inertia of the quadrotor was measured and the motor parameters were identified. A Simulink model is formed to be able to simulate quadrotor behavior and test the controllers.

Linear PID and nonlinear Backstepping methods were preferred for this quadrotor platform. For both of the strategies, attitude and position controllers were designed. The attitude performance of the implemented controllers were tested on a three axis test bench, and the flight was initiated after this step. The designed Backstepping controller was found to leave steady-state error with the given reference inputs, and was therefore not tested in real flight, considering that it could not provide an accurate position hold performance. As the PID controller performed well for the attitude behavior, the experiments continued with PID controller. The low gains of PID, held that way because of vibrations in quadrotor frame, have been updated in real flight, and a satisfactory attitude performance has been achieved.

The reference point required for position control is provided by an ArucoMarker placed on the ground. The orientation and position of this marker is detected by Raspberry Pi and shared with Pixhawk via serial communication. In the flight trials, the position data obtained through image processing was found to be noisy. The system does not have a second corrective camera or a speed measuring hardware. For this reason, the velocity data is obtained by derivative action followed by filtering. At this point, the performance of the altitude controller was found to be satisfactory because the Lidar has a resolution of $\pm 1cm$.

It is observed that the quadrotor is able to hold the position within an area of $15cm$. This accuracy is not sufficient for aerial printing. This system can carry material in an outdoor environment, can be used to mark a location, paint a floor or wall, but cannot be used in printing.

Two main factors can be considered as the reason for the lack of capability. First, the system does not have a hardware to measure velocity. Velocity measurement can be corrected by using a separate equipment, such as an optical flow sensor, and the failed velocity controller can be improved. The second reason is the delay created by serial communications and filters. There is a delay of $30 - 50ms$ during communication between Pixhawk and Raspberry Pi boards. Due to the noisy position and velocity data, the delay amount in the control loop is increased with addition of low-pass filters. Therefore, velocity and position oscillations are evaluated as expected. The hardware selection can be changed for which the image processing and control algorithm meet on the same platform. In addition, healthier images can be obtained from the camera by placing a gimbal under the quadrotor frame.

In conclusion, the quadrotor platform have performed successfully in attitude and altitude control, but could not reach the desired levels in terms of velocity and position hold due to above mentioned reasons. It can be stated that the system will gain aerial printing capability with the specified regulations.

REFERENCES

- [1] Q. Lindsey, D. Mellinger, and V. Kumar, “Construction of Cubic Structures with Quadrotor Teams,” in *Robotics: Science and Systems VII*, Robotics: Science and Systems Foundation, jun 2011.
- [2] G. Hunt, F. Mitzalis, T. Alhinai, P. A. Hooper, and M. Kovač, “3D Printing with Flying Robots,” in *2014 IEEE International Conference on Robotics & Automation (ICRA)*, (Hong Kong), pp. 4493–4499, 2014.
- [3] W. Li, T. Zhang, and K. Kuhnlenz, “A vision-guided autonomous quadrotor in an air-ground multi-robot system,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 2980–2985, IEEE, may 2011.
- [4] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “PIXHAWK: A system for autonomous flight using onboard computer vision,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 2992–2997, IEEE, may 2011.
- [5] R. Niemiec and F. Gandhi, “Multirotor Controls, Trim, and Autonomous Flight Dynamics of Plus- and Cross-Quadcopters,” *Journal of Aircraft*, vol. 54, pp. 1910–1920, sep 2017.
- [6] “eCalc - RC Calculator.” <https://www.ecalc.ch/>. Date Accessed: 2019-11-26.
- [7] “S500 Quadrotor Frame.” <https://www.amazon.com/Readytosky-Quadcopter-Stretch-Version-Landing/dp/B01N0AX1MZ>. Date Accessed: 2019-11-26.
- [8] “T-Motor Propulsion System.” <http://store-en.tmotor.com/>. Date Accessed: 2019-11-26.
- [9] “Pixhawk.” <https://pixhawk.org/>. Date Accessed: 2019-11-26.

- [10] “Raspberry Pi 3 Model B.” <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Date Accessed: 2019-11-26.
- [11] “Raspberry Pi Camera V2.” <https://www.raspberrypi.org/products/camera-module-v2/>. Date Accessed: 2019-11-26.
- [12] “FrSKY Taranis X9D Plus.” <https://www.frsky-rc.com/product/taranis-x9d-plus-2019/>. Date Accessed: 2019-11-26.
- [13] “TF Mini LiDAR Laser Range Sensor.” <https://www.dfrobot.com/product-1702.html>. Date Accessed: 2019-11-26.
- [14] “Gens ace 11.1V 4000mAh 3S Lipo Battery.” <https://www.gensace.de/>. Date Accessed: 2019-11-26.
- [15] “Pixhawk Pilot Support Package User Guide V2.1,” *MathWorks*, p. 71, feb 2017.
- [16] “Simulink Support Package for Raspberry Pi Hardware.” <https://www.mathworks.com/help/supportpkg/raspberrypi/index.html>. Date Accessed: 2019-11-26.
- [17] “OpenCV, Detection of Aruco Markers.” https://docs.opencv.org/trunk/d5/dae/tutorial_aruco_detection.html. Date Accessed: 2019-11-26.
- [18] M. Behniapoor, “Development and Modeling of a Micro Aerial Vehicle (MAV),” Master’s thesis, North Carolina A&T State University, 2017.
- [19] Q. Lindsey, D. Mellinger, and V. Kumar, “Construction with quadrotor teams,” *Autonomous Robots*, vol. 33, pp. 323–336, oct 2012.
- [20] H. Al Jassmi, F. Al Najjar, and A.-h. I. Mourad, “Large-Scale 3D Printing: The Way Forward,” *IOP Conference Series: Materials Science and Engineering*, vol. 324, p. 012088, mar 2018.
- [21] “Behavioural Production, Autonomous Swarm-Constructed Architecture,” pp. 54–59, 2015.
- [22] N. Labonnote, A. Rønnquist, B. Manum, and P. Rüther, “Additive construction: State-of-the-art, challenges and opportunities,” *Automation in Construction*, vol. 72, pp. 347–366, dec 2016.

- [23] V. Gerling and S. Von Mammen, “Robotics for Self-Organised Construction,” in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, pp. 162–167, IEEE, sep 2016.
- [24] V. M. Pawar, R. Stuart-Smith, and P. Scully, “Toward autonomous architecture: The convergence of digital design, robotics, and the built environment,” *Science Robotics*, vol. 2, p. eaan3686, apr 2017.
- [25] B. Dams, Y. Wu, P. Shepherd, and R. J. Ball, “Aerial Additive Building Manufacturing of 3D printed Cementitious Structures,” in *37th Cement and Concrete Science Conference*, no. 055, 2017.
- [26] P. Shepherd and C. Williams, “Shell Design Considerations for 3D Printing with Drones,” in *IASS Annual Symposium*, (Hamburg), 2017.
- [27] J. E. Bostick, C. Park, J. M. Ganci, M. G. Keen, and S. K. Rakshit, “Patent Application Publication,” 2017.
- [28] Y. Pan, X. Zhang, G. Cervone, and L. Yang, “Detection of Asphalt Pavement Potholes and Cracks Based on the Unmanned Aerial Vehicle Multispectral Imagery,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, pp. 3701–3712, oct 2018.
- [29] S. Bouabdallah, *Design and Control of Quadrotors With Application To Autonomous Flying*. PhD thesis, 2007.
- [30] T. Bresciani, “Modelling, Identification and Control of a Quadrotor UAV,” Master’s thesis, Lund University, oct 2008.
- [31] G. Gremillion and J. Humbert, “System Identification of a Quadrotor Micro Air Vehicle,” in *AIAA Atmospheric Flight Mechanics Conference*, vol. 56, (Toronto), American Institute of Aeronautics and Astronautics, aug 2010.
- [32] P. Niermeyer, T. Raffler, and F. Holzapfel, “Open-Loop Quadrotor Flight Dynamics Identification in Frequency Domain via Closed-Loop Flight Testing,” in *AIAA Guidance, Navigation, and Control Conference*, no. January, (Reston, Virginia), pp. 1–14, American Institute of Aeronautics and Astronautics, jan 2015.

- [33] S. Sun, C. C. de Visser, and Q. Chu, “Quadrotor Gray-Box Model Identification from High-Speed Flight Data,” *Journal of Aircraft*, vol. 56, pp. 645–661, mar 2019.
- [34] A. C. Schang, A. Hebert, B. L. Berry, M. T. Block, and R. E. Sherrill, “Quadrotor Performance Model Verification and Validation,” in *2018 AIAA Modeling and Simulation Technologies Conference*, no. January, (Reston, Virginia), pp. 1–26, American Institute of Aeronautics and Astronautics, jan 2018.
- [35] S. Bouabdallah, A. Noth, and R. Siegwart, “PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor,” pp. 2451–2456, 2004.
- [36] J. Li and Y. Li, “Dynamic analysis and PID control for a quadrotor,” in *2011 IEEE International Conference on Mechatronics and Automation*, pp. 573–578, IEEE, aug 2011.
- [37] S. Bouabdallah and R. Siegwart, “Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor,” *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 2247–2252, 2005.
- [38] S. Bouabdallah and R. Siegwart, “Full Control of a Quadrotor,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. TuA5.5, (San Diego), pp. 153–158, 2007.
- [39] A. Mokhtari and A. Benallegue, “Dynamic Feedback Controller of Euler Angles and Wind parameters estimation for a Quadrotor Unmanned Aerial Vehicle,” in *Proceedings - IEEE International Conference on Robotics and Automation*, (New Orleans), pp. 2359–2366, 2004.
- [40] L. Pollini and A. Metrangolo, “Simulation and Robust Backstepping Control of a Quadrotor Aircraft,” in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, (Honolulu), pp. 1–18, American Institute of Aeronautics and Astronautics, aug 2008.
- [41] D. Lee, H. Jin Kim, and S. Sastry, “Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter,” *International Journal of Control, Automation and Systems*, vol. 7, pp. 419–428, jun 2009.

- [42] T. Li, Y. Zhang, and B. Gordon, “Investigation, Flight Testing, and Comparison of Three Nonlinear Control Techniques with Application to a Quadrotor Unmanned Aerial Vehicle,” in *AIAA Guidance, Navigation, and Control Conference*, no. August, (Reston, Virginia), American Institute of Aeronautics and Astronautics, aug 2012.
- [43] F. Lewis, A. Das, and K. Subbarao, “Dynamic inversion with zero-dynamics stabilisation for quadrotor control,” *IET Control Theory & Applications*, vol. 3, pp. 303–314, mar 2009.
- [44] K. Alexis, G. Nikolakopoulos, and A. Tzes, “Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances,” *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.
- [45] C. Nicol, C. J. MacNab, and A. Ramirez-Serrano, “Robust adaptive control of a quadrotor helicopter,” *Mechatronics*, vol. 21, no. 6, pp. 927–938, 2011.
- [46] Z. T. Dydek, A. M. Annaswamy, and E. Lavretsky, “Adaptive Control of Quadrotor UAVs: A Design Trade Study With Flight Evaluations,” *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 1400–1406, jul 2013.
- [47] E. Altuğ, J. P. Ostrowski, and R. Mahony, “Control of a Quadrotor Helicopter Using Visual Feedback,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, no. May, pp. 72–77, 2002.
- [48] J. Kim, M.-S. Kang, and S. Park, “Accurate Modeling and Robust Hovering Control for a Quad-rotor VTOL Aircraft,” *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 9–26, jan 2010.
- [49] J. Stowers, M. Hayes, and A. Bainbridge-Smith, “Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor,” in *2011 IEEE International Conference on Mechatronics*, pp. 358–362, IEEE, apr 2011.
- [50] C. T. Dang, H. T. Pham, T. B. Pham, and N. V. Truong, “Vision Based Ground Object Tracking Using AR.Drone Quadrotor,” *2013 International Conference on Control, Automation and Information Sciences, ICCAIS 2013*, pp. 146–151, 2013.

- [51] S. Yang, S. A. Scherer, and A. Zell, “An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 69, no. 1-4, pp. 499–515, 2013.
- [52] L. V. Santana, A. S. Brandao, M. Sarcinelli-Filho, and R. Carelli, “A trajectory tracking and 3D positioning controller for the AR.Drone quadrotor,” in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 756–767, IEEE, may 2014.
- [53] S. Shen, N. Michael, and V. Kumar, “Autonomous multi-floor indoor navigation with a computationally constrained MAV,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 20–25, IEEE, may 2011.
- [54] K. Schmid, T. Tomic, F. Ruess, H. Hirschmuller, and M. Suppa, “Stereo vision based indoor/outdoor navigation for flying robots,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3955–3962, IEEE, nov 2013.
- [55] A. Benini, A. Mancini, and S. Longhi, “An IMU/UWB/Vision-based Extended Kalman Filter for Mini-UAV Localization in Indoor Environment Using 802.15.4a Wireless Sensor Network,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 70, no. 1-4, pp. 461–476, 2013.
- [56] K. Alexis, *Control of Cooperative Unmanned Aerial Vehicles*. PhD thesis, University of Patras, 2011.
- [57] M. Yıldız, “Effects of Active Landing Gear on the Attitude Dynamics of a Quadrotor,” Master’s thesis, Atılım University, 2015.
- [58] E. C. Suiçmez, “Trajectory Tracking of a Quadrotor Unmanned Aerial Vehicle (UAV) via Attitude and Position Control,” Master’s thesis, Middle East Technical University, 2014.
- [59] C. Balas, “Modelling and Linear Control of a Quadrotor,” Master’s thesis, Cranfield University, 2007.
- [60] R. Beard, *Quadrotor Dynamics and Control Rev 0.1*. Brigham Young University, 2008.

- [61] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2nd ed. ed., 2003.
- [62] B. Etkin, *Dynamics of Atmospheric Flight*. New York: Dover Publications, Inc., 2000.
- [63] J. H. Ginsberg, *Advanced Engineering Dynamics*. Cambridge University Press, second edi ed., 1998.
- [64] H. t. M. ElKholy, “Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches,” Master’s thesis, The American University in Cairo, 2014.
- [65] “Constant Velocity of Brushless Motors, KV.” <https://www.rotordronepro.com/understanding-kv-ratings>. Date Accessed: 2019-11-26.
- [66] W. Zhong, “Implementation of Simulink controller design on Iris+ quadrotor,” Master’s thesis, Naval Postgraduate School, 2015.
- [67] M. Ireland, A. Vargas, and D. Anderson, “A Comparison of Closed-Loop Performance of Multirotor Configurations Using Non-Linear Dynamic Inversion Control,” *Aerospace*, vol. 2, pp. 325–352, jun 2015.
- [68] D. F. G. Herrera, “Design, Development and Implementation of Intelligent Algorithms to Increase Autonomy of Quadrotor Unmanned Missions,” 2017.
- [69] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, third edit ed., 2002.