Hochschule
Ravensburg-Weingarten
University of Applied Sciences

Master Thesis

Development and Implementation of a Control System for a quadrotor UAV

by

Yiting Wu

March 2009

1. **Supervisor:** **Prof. Dr. –Ing Holger Voos**
2. **Supervisor:** **Prof. Dr. –Ing Konrad Wöllhaf**

Acknowledgments

- I would like to thank my thesis supervisor, Prof. Dr. Voos, for his guidance and support throughout the duration of this thesis.

- Thank you to Achim Feucht for his active support in the lab and his experiences of electrical engineering always helps me a lot to solve the problems.

- Thank you to Martin Binswager for his help on explanation of his bachelor thesis which made the work on electrical board and IMU sensor much easier.

- Many thanks to my family and my girlfriend for their patience and understanding, no matter where they were. In particular, my girlfriend who studied industrial design for her bachelor helped me to design the figure of "course of events".

- Last but not least, I would also like to thank my friends and fellow students for the great time spent together in the lab. These experiences in the duration of my thesis will be a great treasure for my memory of the study in University of Applied Science Ravensburg-Weingarten in Germany.

Declaration

Hereby I declare that this report has never been accepted in substance of any degree. I have composed it myself as a result of my own investigations, except where otherwise stated.

Weingarten, Germany, 31st March 2009 _____

Yiting Wu

ABSTRACT

Development and Implementation of a Control System for a quadrotor UAV

Yiting Wu

Master of Science

Department of Electrical and Computer Engineering

University of Applied Science Ravensburg-Weingarten

Small quadrotor UAVs are considered as one important type of vehicle for future unmanned aerial missions. In spite of the four actuators of the system, the quadrotor is a dynamically unstable system that has to be stabilized by a suitable control system. During control systems design however, the nonlinear dynamics of the system has to be taken into account. Based on an already existing simulation model in Matlab/Simulink and already derived control concepts, a control system for a quadrotor UAV has to be developed in this Master thesis and implemented in a test bed. The test bed is a quadrotor vehicle which will be fixed on a spherical bearing in order to allow test flights.

The first task herein comprises the final construction of the test platform and the implementation of an inertial measurement unit (IMU, from Xsense) in the vehicle. Furthermore, the quadrotor vehicle with implemented IMU is connected with an ATMega2560™ microcontroller board. The microcontroller is able to command the rotational speed of the rotors and to get the measurements of the IMU as input signals. In the first step, the microcontroller board is placed outside of the vehicle. With the help of the test platform, the main parameters of the quadrotor vehicle are identified via some suitable experiments.

In a second step, three control algorithms are developed based upon the Matlab™/Simulink™ simulation model and the already available control concepts. The control algorithms then are optimized for implementation in the microcontroller through C/C++ language, taking the limitations of the hardware into account. The control algorithms are implemented in the test platform, evaluated and compared against each other. Finally, the best control algorithm is determined.

Table of Contents

Chapter 1

Introduction

This thesis work focuses on the attitude control of a Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicle (UAV). The proposed structure is four-rotor micro aerial robot, so called quadrotor.

The UAVs has seen a growing interest over the past decade because of the wide area of applications, e.g. near-area surveillance, crop dusting, fire fighting and exploration both in military and commercial in- and outdoor applications etc. The quadrotor is one of the most preferred types of UAV which can apply in above mentioned fields. The reason is the very easy construction and steering principle using four rotors in a cross configuration against the traditional helicopter construction using one main rotor and one tail rotor.

The first question one is asked about the quadrotor is how it stands out from the traditional one. Hence a short introduction about the quadrotor construction and steering principle is necessary. The quadrotor is a mechatronic system with four rotors that provide the lift and control. With respect to hover, the main difference is best explained by considering how the helicopters compensate from gyroscopic torques. Traditional helicopters basically compensate from the torque generated by the main rotor through the tail rotor. However the tail rotor compensation conducts a sideways displacement of the helicopter, thus counter steering by tilting the main rotor blades is necessary. In this way hover is an ongoing and complex process.

The quadrotor has four propellers driven by four motors in a cross configuration. While the front and the rear motor rotate counter-clockwise, the left and the right motor rotate clockwise, as long as the rotors rotate at the same speed the gyroscopic effects are nearly eliminated and the quadrotor essentially hovers. One additional advantage of the quadrotor compared to a traditional helicopter is the simplified rotor mechanics. By varying the speed of the single motors, the lift force can be changed and vertical and/or lateral motion can be created, see 4.1 Quadrotor Kinematics.

1.1 Motivation

Although the quadrotor has the advantages in easy mechanical construction against the traditional helicopter, but there are still issues that prevent it from being widely used in many of the suggested fields and application. First, the stabilizing control and guidance of the quadrotor is a difficult task because of the nonlinear dynamic behavior. Second, the small payload and the reduced processing power of the onboard electronics are further limitations for any control system implementation. This means, in order to implement more complex control task such as landing and target tracking, more adequate sensors might be equipped on the quadrotor which needs a fixed ground station for processing of all sensors related to control of the micro-UAV.

Since the attitude stabilization could be considered the single most important component of flight control for the quadrotor and this is also a precondition for further implementation of other functionalities in the vehicle, the main goal of this paper is to realize the attitude stabilization by using the Inertia Measurement Unit

(IMU) sensor equipped on quadrotor. There are some contributions in the literature that are concerned with control system design for quadrotor vehicles. Many of the proposed control systems are based on a linearized model and conventional PID- or state space control [5], [11], [15]while the other approaches apply SDRE or Nonlinear Feedback control [1] , [18]. In this paper the Nonlinear Feedback control [18] is studied and chosen to implement on the real quadrotor. In order to compare Nonlinear Feedback control with the conventional PID control technique, a PID controller is designed and implemented on the quadrotor. Finally according to the simulation and test results, the features of these two kinds of control techniques are summarized.

1.2 Course of events

This section will briefly give insight into the intermediate goals and objectives on the way towards achieving autonomous hovering flight. Figure 1.1 provides an overview of the course of events during the whole thesis period.

At the beginning of September work started from reading the corresponding literatures to provide a basic understanding of how the quadrotor operates, what kind of physics are involved and essentially how to combine this knowledge into a useful model for control purpose. The bold goal at that time was to be able to hover at the end of the thesis. After that, a lot of time and efforts were put on get related electronic unit ready to work, including the IMU configuration, motor controller design, USART interrupt routine design and so on. At the end, two controllers were implemented on the quadrotor and could control the attitude with small tolerances.

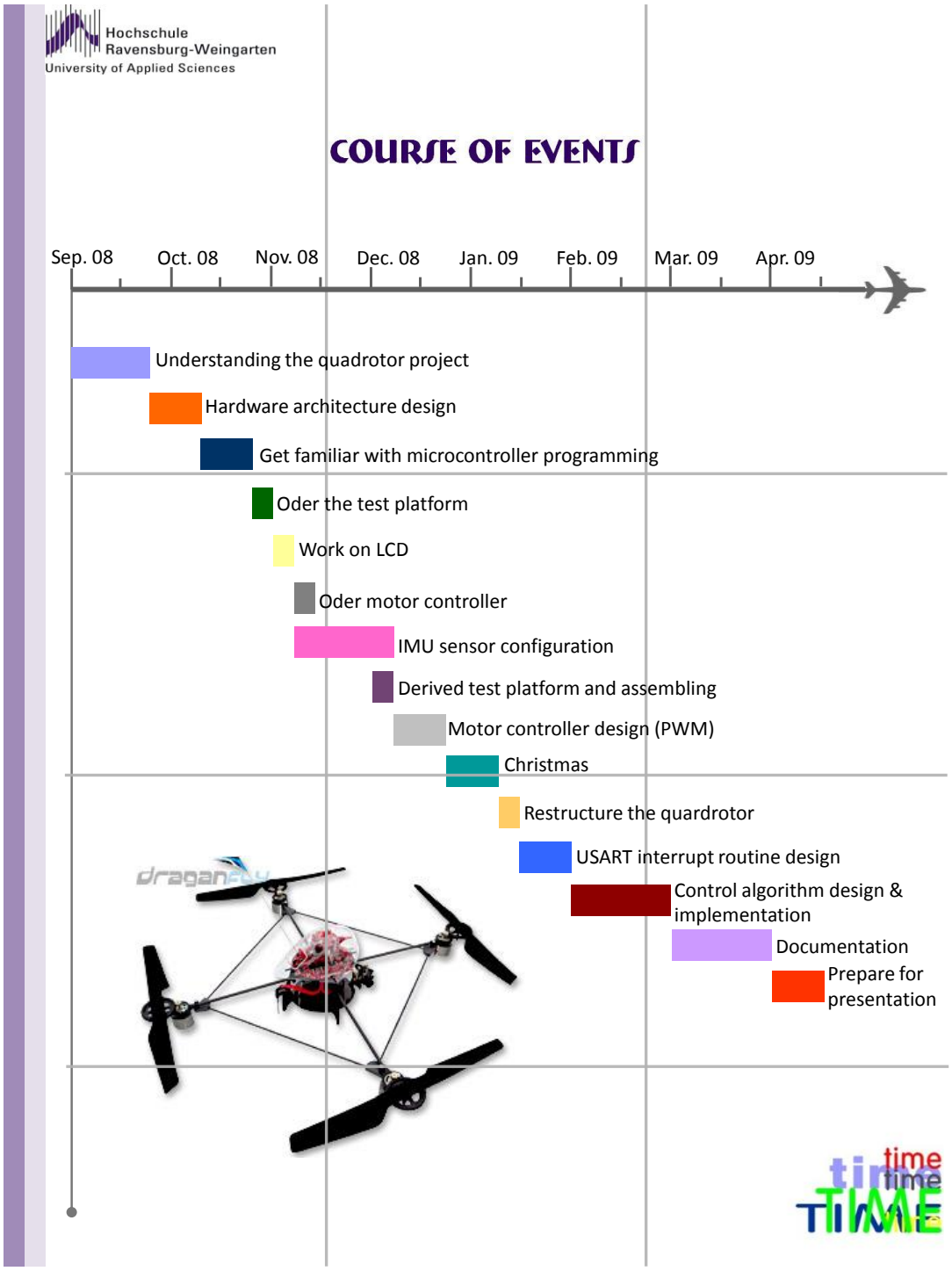


Figure 1.1 Timeline illustrating Course of Events

1.3 Outline of Thesis

Chapter 2 deals with the hardware design for the quadrotor and hardware specifications, which includes the following aspects, such as: mechanical structure design of the quadrotor, test platform, electronic units, sensors and actuators.

Chapter 3 gives the information about software preparation and specification in order to make the electronic board and IMU sensor ready to be used for later on the control algorithm implementation. This chapter is mainly about the microcontroller programming.

Chapter 4 provides the overall quadrotor model of kinematics and dynamics.

Chapter 5 focuses on the control algorithms design which are needed to stabilize the quadrotor. The nonlinear control using feedback-linearization and PID techniques are adopted in this work. Besides the design of the control algorithm, the implementation of the different controller in C language is also explained in this chapter.

Chapter 6 presents the simulation and experimental test results of the controllers designed in chapter 5.

Chapter 7 summarizes the control effect of the controllers based on the test results and also proposes solutions to improve performance of the attitude control.

Chapter 2

Hardware Design and Specification

Introduction

This chapter is mainly about hardware design for the quadrotor and hardware specifications, which includes the following aspects, such as: mechanical structure design of the quadrotor, test platform, electronic units, sensors and actuators. At the first beginning, the hardware architecture should be decided. Then according to the selected hardware architecture, the necessary components should be selected and prepared so the hardware architecture can be built. In order to make all the components are ready to work, the specification must be done for all the components which includes both the hardware and software sides.

2.1 Hardware Architecture

The hardware architecture shows the develop concept and describes the whole project. Many different hardware architectures have been used for developing the autonomous quadrotor, like real-time implementation includes hardware in the loop or microcontroller-on-board solution which is not real time. The first task of the project is that, based on the available electronic units and components and also the develop software, one proper develop concept should be chosen which is possible to be realized and implemented under the condition of the university lab.

2.1.1 Real-Time Implementation

A system is a real-time system when it can support the execution of applications with time constraints on that execution. Real-time control is a popular term for a certain class of digital controllers. For effective digital control, it is critical that sample time be constant. Real-time control achieves nearly constant sample time. The most used real time application tools are Matlab Real-Time Workshop™ or xPC Target™. By using these tools you can create a real-time application to let the system run while synchronized to a real-time clock. This allows the system to control or interact with an external system. Figure 2.1 shows the hardware architecture for the real-time control concept which is developed by student group from AALBORG University as their master thesis.[5]

As the Figure 2.1 shows it has been chosen to equip the quadrotor with sensors as GPS for absolute position estimate, Magnetometer for information about heading range finder to aid the GPS in getting an altitude estimate, IMU for the possibility to propagate position and attitude. Besides all these sensors there are a number of components related to manual flight and other safety feature. All together these transducers are connected to the main CPU, some via a slave processor, the Robostix board, which is built up around a 16 MHz Atemega128 processor. The Gumstix features a 400 MHz Intel XScale processor of the type PXA255. It has 16 MB flash memory and 64 MB of SDRAM. The embedded Linux system is running on this Gumstix. Wifistix is a fully configurable wireless board, following the 802.11(g) standard, which means that the bandwidth can be up to 54 ^{Mb}/s.

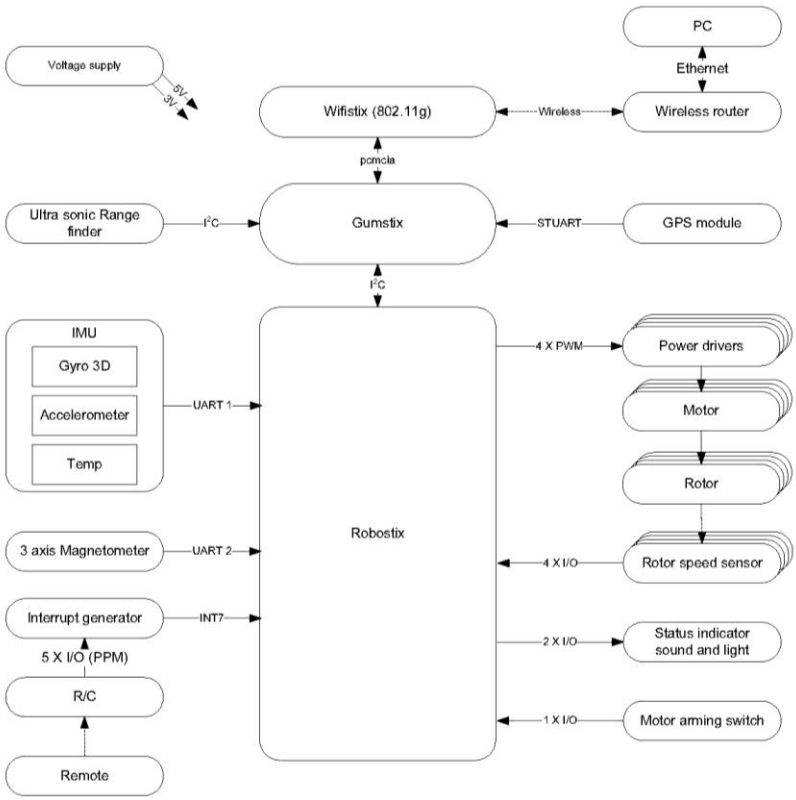


Figure 2.1 Hardware Architecture for real-time application [5]

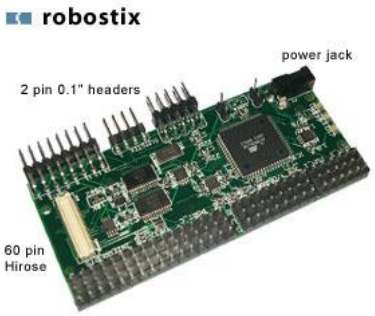


Figure 2.2 Robostix Board

Figure 2.3 Gumstix

Figure 2.4 Wifistix

First the Robostix is treated which has the main task of forwarding sensor data to the Gumstix. The low level code is written in C and cross compiled to generate a hex file which is loaded onto the Atmega128. Second the Gumstix software is treated which is written in high level C code. This software is also cross compiled to fit the system specific architecture of the Gumstix, namely the Intel Xscale PXA255 processor. Next the Development Host Machine software is described which is basically defined by a Linux Soft Real Time Target application in Matlab™/Simlink™. It is in this environment the controller will be derived and also implemented.[5] The interaction between the components and the main process is illustrated in Figure 2.5.

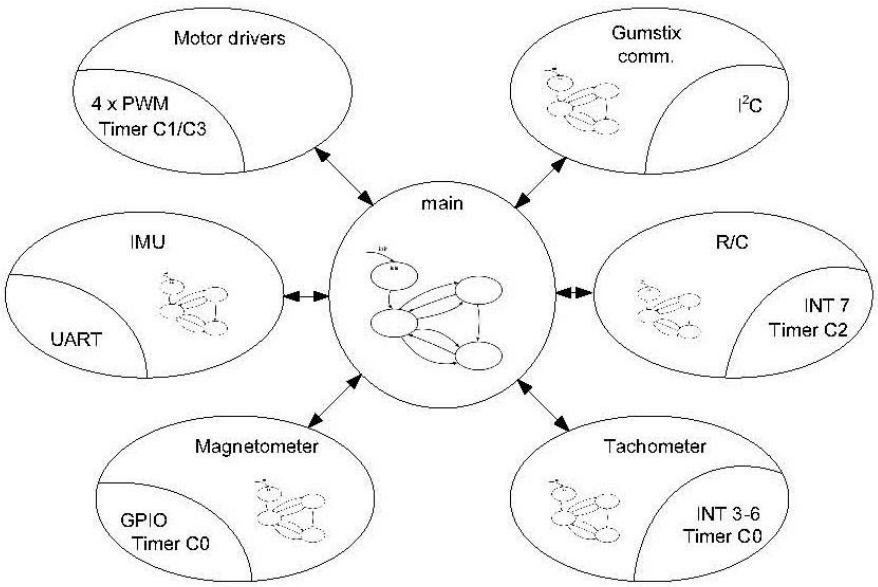


Figure 2.5 The main process interacting with peripheral components [5]

2.1.2 Microcontroller-on-Board Implementation

Microcontroller-on-Board implementation means that, the quadrotor is equipped with all the necessary sensors and electronic control board which is built with certain type of microcontroller, the task of the microcontroller is not only receiving the data from the sensor and sending the signal to the actuator, but also responsible for processing the sensor data and computing the needed value which can achieve the desired control task. The control algorithm is first developed based on the modeling and simulation of the physical system by using the simulation software like Matlab™/Simulink™. Then the designed algorithm should be translated into low level communication language like C/C++, which later can be built in HEX file format and flashed on the microcontroller. Compare to the real time implementation described above, this development can't allow the quadrotor be controlled in real time, which also means, during the flying of the quadrotor, there is no interaction between the quadrotor system and the external system like pc.

This approach is most used on the model building. One we have in the lab is named DraganFlyer, which is also commercially available, see www.rctoys.com.

Figure 2.6 DraganFlyer

2.1.3 Results

As two kinds of developing concept described above, finally the microcontroller-on-board implementation has been chosen.

The real time implementation described in 2.1.1 has the big advantage in rapid prototyping, which means using the Real Time Workshop™ or xPC target™ of Matlab™ to design the control algorithm on the host PC and then the c code is automatic generated and programmed on the quadrotor microcontroller. It saves developing time and makes the hardware in the loop test possible, in the other words; the tuning of the controller parameters during the test of quadrotor can be realized.

Although there are many advantages of real time implementation, but take into the hardware aspect into account, it needs the interface board which is supported by Matlab Real Time Workshop™ or xPC Target™ to communicate between microcontroller and Host PC, this increases the cost of whole project and since we don’t have these kind of Hardware, for ordering it takes also the time.

Considering the main goal of this thesis is only stabilize the quadrotor in order to realize that the quadrotor can hovering in the air without any remote control, the IMU sensor MTi from Xsens and the RN-Control Board 2560 have been selected for the project, these hardware are available in our lab. Figure 2.7 shows the hardware architecture used for this project, quadrotor attitude control.

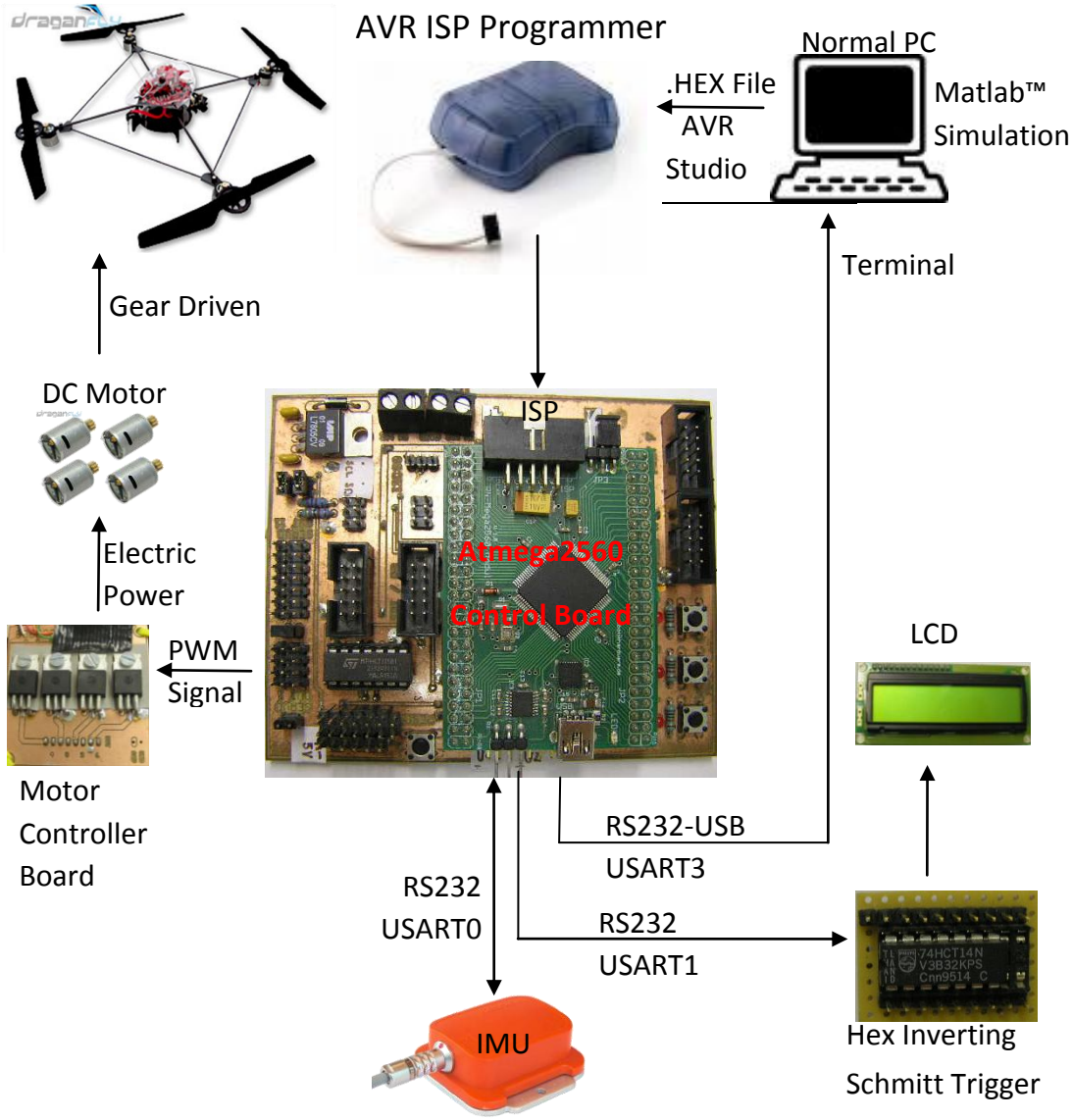
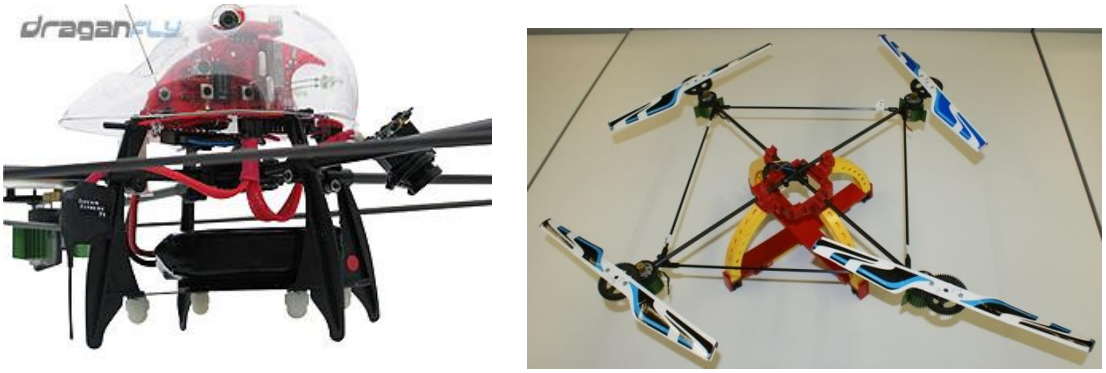


Figure 2.7 Hardware Architecture for Microcontroller-on-Board Implementation

2.2 Quadrotor Structure

In this project the Draganflyer V Ti Pro has been chosen as the basic structure of the quadrotor, since this quadrotor is available in the lab and there are many studies and researches based on this Draganflyer. One of the original vertical risers of the quadrotor was broken during the several test flies, in order to fix the quadrotor on the test platform, the new structure is demanded.

Figure 2.8 (a) shows the original structure of the Draganflyer and Figure 2.8 (b) is the new structure designed by myself. It's heavier than the original one and it needs the maximal rotational speed of the rotor to lift up. But for testing on the platform the lower parts can be removed. For attitude control testing, only the row, pitch and yaw angle of the quadrotor need to be controlled and to lift up the quadrotor quickly is not so important, so this structure can fulfill the requirement.



(a) (b)

Figure 2.8 Mechanical Structure of Quadrotor

2.3 Test Platform

The test platform is ordered from www.kdg-produkte.de. With this test platform it is possible to fly and test the quadrotor inside the lab without any dangerousness to people. The motion of the quadrotor is limited inside the test platform with the range of 100 mm in X and Y and 11 mm in Z. The freedom of the roll and pitch angel of the quadrotor is limited inside ±20 degrees through the sphere joint. Figure 2.9 shows the test platform and quadrotor fixed on the platform.

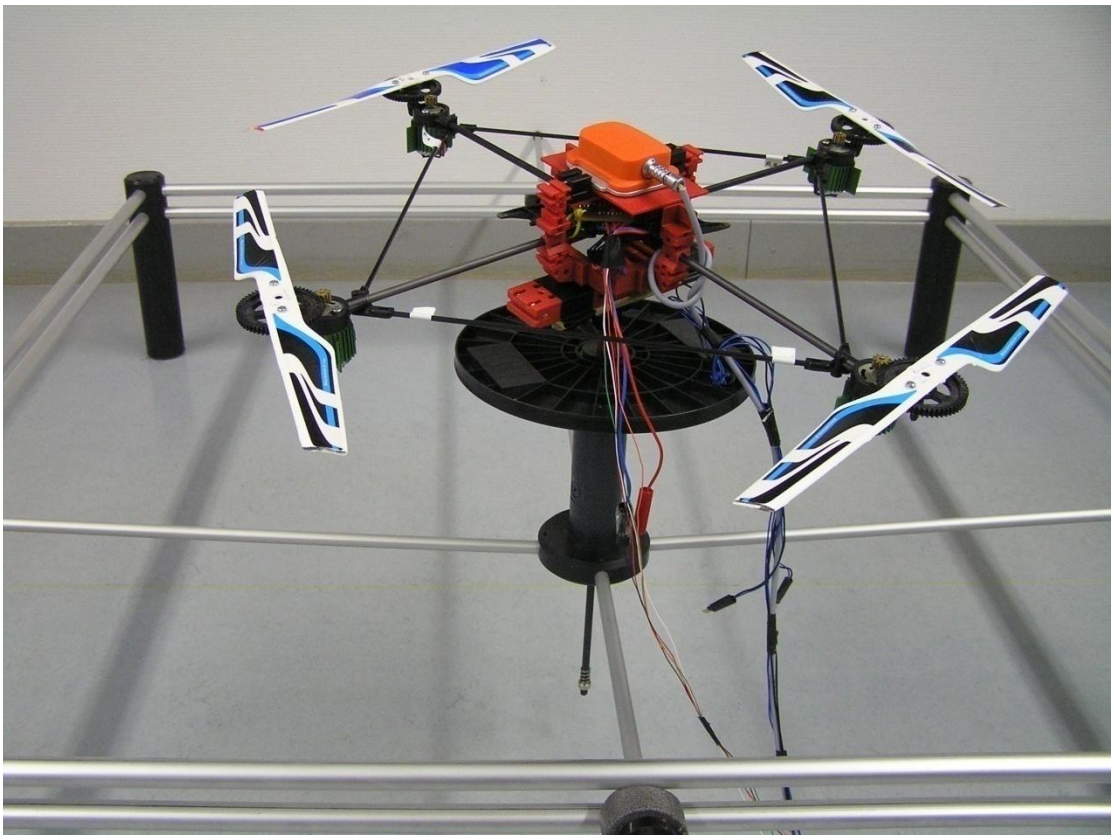


Figure 2.9 Test Platform

2.4 IMU

IMU stands for inertial measurement unit. The IMU sensor MTi which is used in this project is produced by Xsens™, more detailed product information can be found on www.xsens.com. The MTi is a miniature, gyro-enhanced Attitude and Heading Reference System (AHRS). Its internal low-power signal processor provides drift-free 3D orientation as well as calibrated 3D acceleration, 3D rate of turn (rate gyro) and 3D earth-magnetic field data. The MTi is an excellent measurement unit for stabilization and control of cameras, robots, vehicles and other equipment. Figure 2.10 shows the overview of the MTi sensor.

Figure 2.10 MTi Overview

2.4.1 Sensor Communication Features

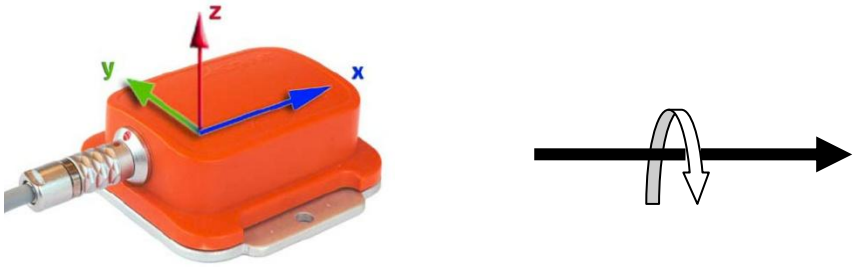
- Interface through COM-object API, capable to access the MTi directly in application software such as MATLAB™, LabVIEW™, Excel (Visual Basic). →Please refer to the **MT Software Development Kit Documentation**[6] for more information on this topic.

- Direct low-level communication with MTi (RS-232/422).

  →Please refer to the **MTi and MTx Low-level communication protocol Documentation**[6] for more information on this topic.

2.4.2 Co-ordinate systems

All calibrated sensor readings (accelerations, rate of turn, earth magnetic field) are in the right handed Cartesian co-ordinate system as defined in Figure 2.11 (a). This co-ordinate system is body-fixed to the quadrotor and is defined as the sensor co-ordinate system.



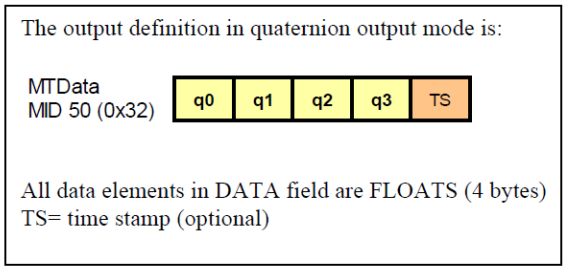
(a) (b)

Figure 2.11 MTi with sensor-fixed co-ordinate system overlaid

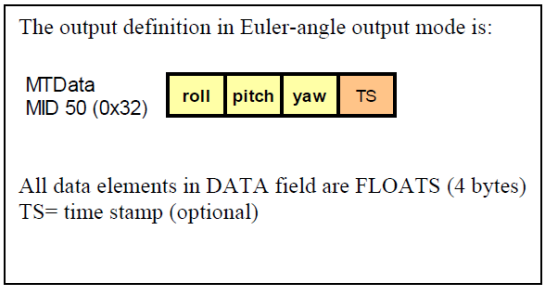
A positive rotation is always „right-handed“, i.e. defined according to the right hand rule (corkscrew rule). This means a positive rotation is defined as clockwise in the direction of the axis of rotation, Figure 2.11 (b).

2.4.3 Output Modes

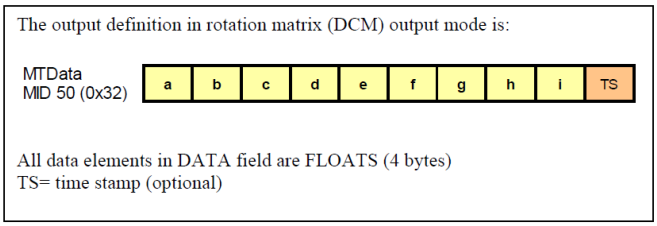
- Quaternion orientation output mode



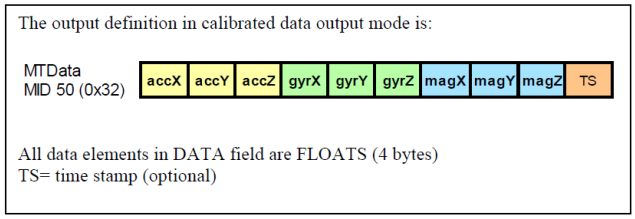
- Euler angles orientation output mode



- Rotation Matrix orientation output mode



- Calibrated data output mode

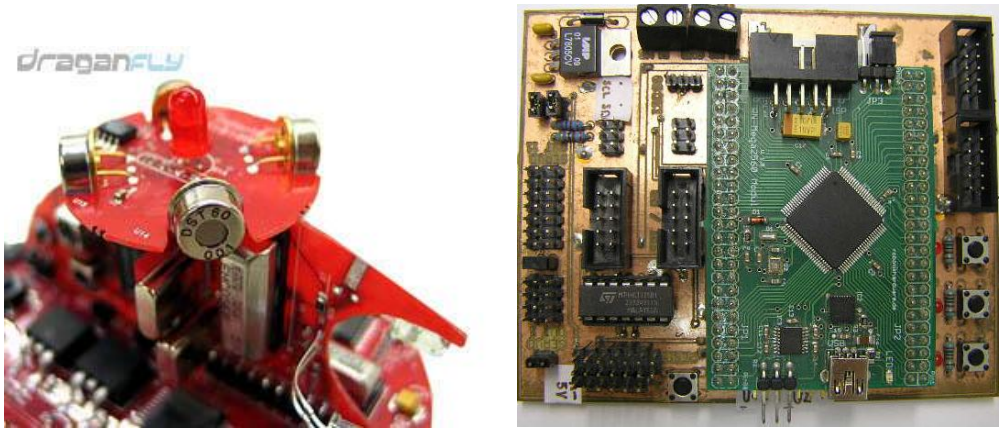


→Please refer to the **MTi and MTx User Manual**[6] for more information on this topic.

2.5 Microcontroller Board

The original electronic circuit board (Figure 2.12 (a)) of the commercial DraganFlyer is equipped with Radio Frequency (RF) control module, three Piezo electric gyro sensors in X, Y and Z direction which are used for stabilization, four infrared sensors which are used for thermal intelligence to maintain the altitude.

→Please refer to the **df5ti-manual**[6] for more information on this topic.



(a) (b)

Figure 2.12 Electronic circuit board

Since there is no interface between the microcontroller and PC, the microcontroller cannot be programmed by ourselves. So it is necessary to remove the original board and develop our own one. Thanks to Martin Binswanger who has developed one board for his bachelor project, this board is also competent for my quadrotor project. Thus my quadrotor project is based on Martin's electronic board. The board is shown in Figure 2.12 (b).

2.6 Motor Controller

The functions of the motor controller are that, a). give the power supply to the motors; b). amplify the motor speed control signal, here is PWM signal. Since the motor controller give the power supply to the motor, so the maximal load current should be taken into account. As from the measurement result, when the rotor rotates on full speed, the current in the wire is 2.5 A, so the maximal load current of the motor controller for each motor should bigger than 2.5 A. After searching on the market, finally the motor controller BTS 621 L1 from Siemens[10] has been chosen, whose maximal Load current (each) is 4.4 A. Then the motor controller board has been made with four motor controllers, where one controller controls only one motor. Figure 2.13 shows the motor controller board.

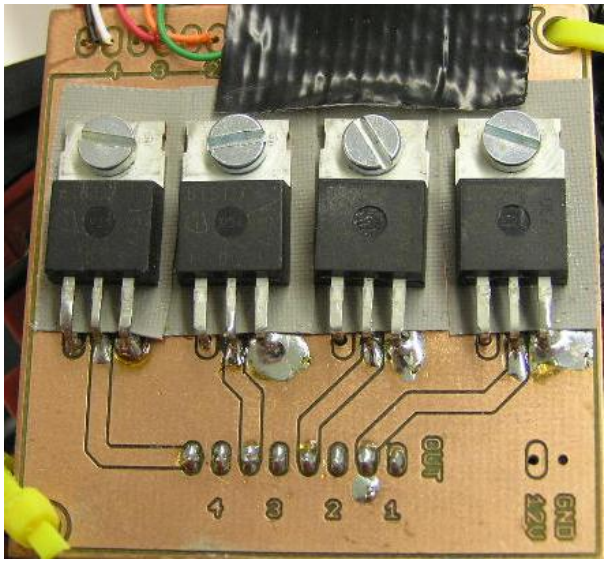
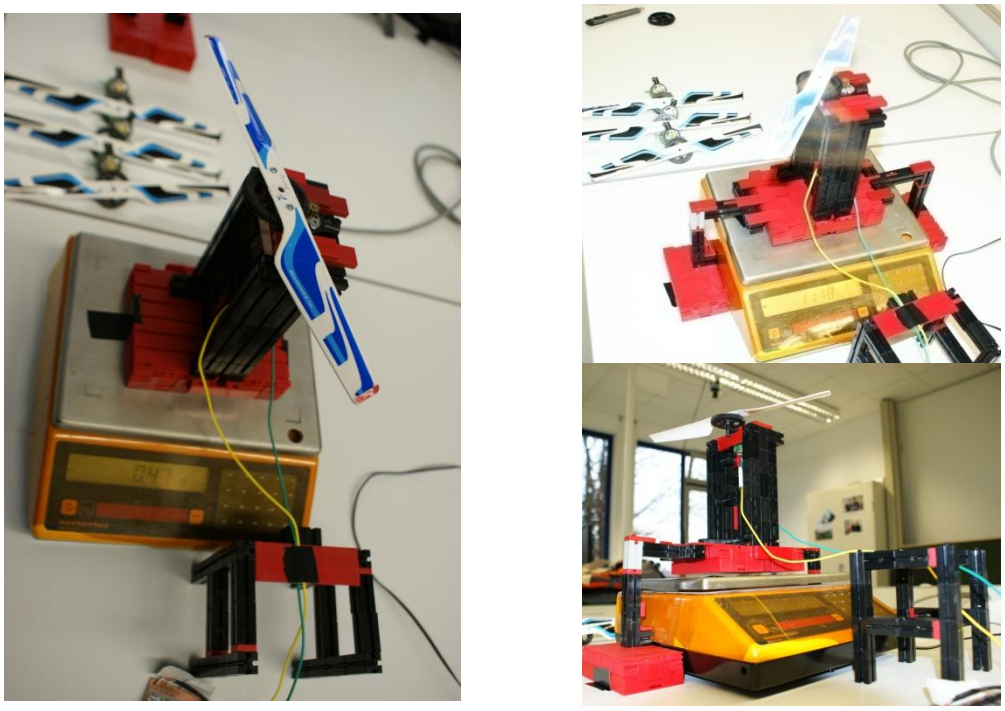


Figure 2.13 Motor controller Board

2.7 Motor and Propeller

The quadrotor is equipped with four DC motors RC-280SA-2865 produced by MABUCHI™ and four nylon propellers driven by the motors via pinion gear. For controlling the rotational speed of the propellers to stabilize the quadrotor, we need to know the relationship between the rotational speed of the propeller and the lift force generated by the propeller. Thus an experiment is designed for measuring the generated lift force by the propeller on certain rotational speed. Figure 2.14 shows the experiment structure.



(a) (b)

Figure 2.14 Motor Force Experiment

The experiment is divided into two parts. First, as Figure 2.14 (a) shows, the motor set is fixed on the assistant bracket which is pasted on the test platform of the balance by the adhesive tape. Then I give the motor PWM signal from 0 to 250 with incensement by 10 which corresponding to 0% and 100% of the maximal rotational speed of the propeller, at the same time, the balance measures the weight generated by the propeller (M_1: gram) where negative value stands for the lift direction. Second, as Figure 2.14 (b) shows, the assistant bracket is attached to the desk but not directly on the test platform of the balance where there is a small gap between two parts, so we can measure only the weight (M_2: gram) of the blowing down air flow generated by the propeller. Finally, the lift force (F_l: Newton) generated by the propeller on certain rotational speed is:

$$F_l = (M_2 - M_1) \times g \div 1000 \quad (2.1)$$

Where g=9.8 N/Kg, is the gravity of the earth. The diagram 2.1 shows the experiment results in graphic and table 2.1 shows the values of the experiment results.

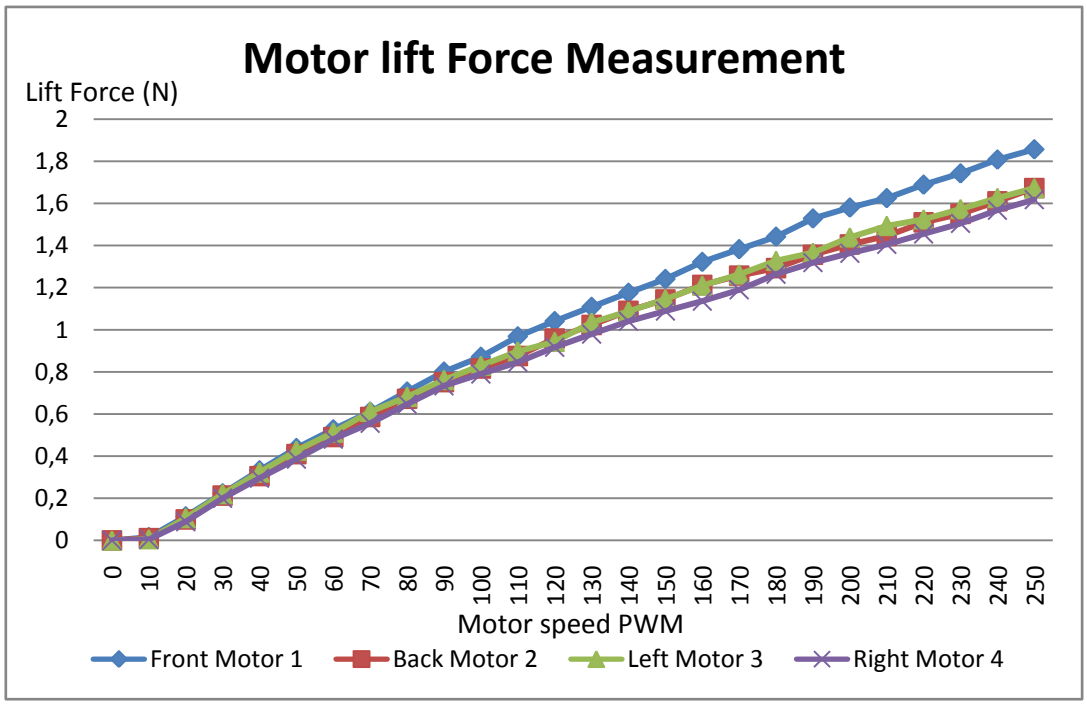


Diagram 2.1 Motor Lift Force Measurement

Table 2.1 Motor Lift Force Measurement

Motor Speed PWM	Front Motor 1 Lift Force (N)	Back Motor 2 Lift Force (N)	Left Motor 3 Lift Force (N)	Right Motor 4 Lift Force (N)
0	0	0	0	0
10	0,015484	0,01078	0,006174	0,004606
20	0,11417	0,099862	0,106428	0,088592
30	0,223636	0,212856	0,220108	0,200018
40	0,33271	0,30527	0,323106	0,295372
50	0,43855	0,409934	0,426692	0,38612
60	0,52724	0,490588	0,512246	0,481376
70	0,61103	0,58604	0,610834	0,556248
80	0,706678	0,67228	0,681786	0,6468
90	0,801444	0,75166	0,762146	0,734314
100	0,872102	0,81634	0,829766	0,790566
110	0,968828	0,875826	0,897974	0,846132
120	1,041348	0,958244	0,943348	0,91728
130	1,108968	1,02312	1,034782	0,979706
140	1,175804	1,08976	1,087506	1,040368
150	1,241366	1,14366	1,145522	1,088976
160	1,322412	1,21324	1,2103	1,13631
170	1,382878	1,25734	1,263514	1,189818
180	1,442364	1,29262	1,3279	1,264102
190	1,5288	1,35632	1,366512	1,31908
200	1,580446	1,40532	1,438934	1,363278
210	1,62533	1,4455	1,49352	1,405712
220	1,689226	1,51018	1,524782	1,4553
230	1,742734	1,55036	1,573488	1,50528
240	1,808394	1,61014	1,626114	1,568
250	1,8571	1,67286	1,67286	1,618176

From the experiment results as showed in Diagram 2.1, we take the range of the motor speed PWM signal from 100 to 250 and linearize this section, and then we can get the equation of thrust force F_i generated by rotor i, i =1, 2, 3, 4:

$$F_i = a_i + b_i \times PWM_i \qquad PWM_i \in (100,250) \tag{2.2}$$

where a_i and b_i are the thrust force factors.

2.8 Identification of the constants

To identify the constants of the quadrotor, there are different approaches to estimate a constant. The experiments to determine the constants are described in [11]. Since we use the same quadrotor mode (Figure 2.6) with the mode used in paper [11], so the parameters determined in [11] can be used here too. Table 2.2 gives the quadrotor constants.

Table 2.2 quadrotor constants

Parameter	Value	Unit	Remark
g	9.81	m/s^2	gravity
m	0.55	kg	Mass of the quadrotor
L	0.21	m	Length of the lever
$Ix = Iy$	$4.85 \cdot 10^3$	$kg \cdot m^2$	Inertias around X and Y axis
Iz	$8.81 \cdot 10^3$	$kg \cdot m^2$	Inertia around Z axis
I_R	$4.85 \cdot 10^3$	$kg \cdot m^2$	Rotor Inertia
b	$2.92 \cdot 10^{-6}$	$kg \cdot m^2$	Thrust factor
d	$1.12 \cdot 10^{-7}$	$kg \cdot m^2$	Drag factor
a_1	0.309	N	Thrust factor, Front Motor 1
a_1	0.345	N	Thrust factor, Back Motor 2
a_1	0.25	N	Thrust factor Left Motor 3
a_1	0.34	N	Thrust factor, Right Motor 4
b_1	0.0063	N	Thrust factor, Front Motor 1
b_2	0.0053	N	Thrust factor, Back Motor 2
b_3	0.006	N	Thrust factor, Left Motor 3
b_4	0.005	N	Thrust factor, Right Motor 4

Chapter 3

Software Preparation and Specification

3.1 IMU Initialization and Configuration

3.1.1 Message Structure

The communication protocol of the IMU with microcontroller, which is message based, enables the user to change the configuration of the Mti and to retrieve the data from the device. The used message has the form:

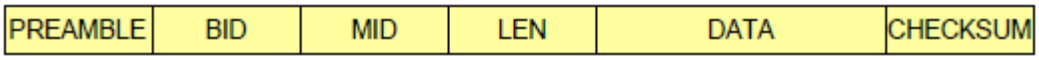


Figure 3.1 IMU Message Structure

Field	Field width	Description
Preamble	1 byte	Indicator of start of packet → 250 (0xFA)
BID	1 byte	Bus identifier / address → 255 (0xFF)
MID	1 byte	Message identifier
LEN	1 byte	Value equals number of bytes in DATA field Maximum value is 254 (0xFE). Value 255 (0xFF) is reserved.
DATA	0 – 254 bytes	Data bytes (optional)
Checksum	1 byte	Checksum of message

Table 3.1 IMU Message Structure

Every message will be closed with one CHECHSUM data field. This field is used for communication error-detection. If all message bytes excluding the preamble are summed and the lower byte value of the result equals zero, the message is valid and it may be processed.

The checksum value of the message can be calculated by the following equation.

$$Checsum = complement(BID + MID + LEN + DATA) + 1 \quad (3.1)$$

For example, we have the message of set output mode:

PREAMBLE	BID	MID	LEN	DATA1	DATA2	CHECKSUM
0xFA	0xFF	0xD0	0x02	0x00	0x06	0x29

For example:

SUM=0xFF+0xD0+0x02+0x00+0x06=0x1D7=111010111(binary)

complement (SUM)=000101000

CHECKSUM=complement+1=101001(binary)=0x29

First of all, the initialization of the IMU was executed with a power-cycle or reset of the microcontroller. Through the initialization of the IMU (subprogram: imu_init(…)[12]) the properties of the data output are configured.

Baudrate	115k2
Sampling Rate	100Hz
Output Rate	100Hz
Output Mode	Calibrated data--rate of turn and Euler angles

Table 3.2 The used configuration of IMU

After the initialization the IMU goes from Configuration State to the Measurement State. The IMU sends 100 data packets pro second through RS232 interface to the microcontroller. Every data packet has the size of 29 Bytes.

The measured data of rate of turn and Euler angles are coded in float number which is 4 Bytes long. The received Bytes are checked first and then stored in one Bufferarray(MH_rx_buffer[32])[12]. In Bufferarry the data are found again under following index n:

n	Data
0	Preamble
1	BID
2	MID
3	Length
4 to 7	4 Byte Float Number = Rate of turn around X-Axis
8 to 11	4 Byte Float Number = Rate of turn around Y-Axis
12 to 15	4 Byte Float Number = Rate of turn around Z-Axis
16 to 19	4 Byte Float Number = Euler Angle around X-Axis
20 to 23	4 Byte Float Number = Euler Angle around Y-Axis
24 to 27	4 Byte Float Number = Euler Angle around Z-Axis

Table 3.3 The Bufferarray for the IMU Data

In order to make the four single elements(each one is 1 byte) of the Bufferarrays as one float number for the software which is 4 bytes, the subprogram "float make_float(…)"[12] must be used.

3.1.2 Message usage in "imu_init()"

IMU-Reset:

{0xfa, 0xff, 0x40, 0x00, 0xC1};

IMU-WakeUp:

{0xfa, 0xff, 0x3f, 0x00, 0xC2};

SetBaudrate: 115k2

{0xfa,0xff,0x18,0x01,0x02,0xE6 };

ReqBaudrate:

{0xfa,0xff,0x18,0x00,0xE9};

SetPeriod: sets 10 ms to the sampling period of the device used in Measurement State

{0xfa,0xff,0x04,0x02,0x04,0x80,0x77};

SetSkipFactor: every MTData message is sent

{0xfa,0xff,0xD4,0x02,0x00,0x00,0x2B};

SetOutputMode: calibrated data output

{0xfa,0xff,0xD0,0x02,0x00,0x06,0x29};

SetOutputsettings: rate of turn and Euler angle enable, acceleration and magnetometer disable

{0xfa,0xff,0xD2,0x04,0x00,0x00,0x00,0x54,0xD7};

SetExtOutputMode: Analog Output Mode disable

{0xfa,0xff,0x86,0x02,0x00,0x00,0x79};

GotoMeasurement: enter the Measurement State

{0xfa,0xff,0x10,0x00,0xF1};

→ Please refer to the **MTi and MTx Low-level Communication**[6] for more information on this topic.

3.2 Pulse-Width Modulation (PWM)

Pulse-width modulation (PWM) uses a square wave whose pulse width is modulated resulting in the variation of the average value of the waveform. PWM is often used to control the supply of the electrical power to the another device such as speed control of electric motors. The microcontroller Atmega2560 we used for the project can support 8-bit Timer/Counter0 with different kinds of PWM operation modes. In my code I use 8 bit counter with Fast PWM mode. The PWM signal has the changing range between 0 and 255 because of the 8 bit counter0.

The fast Pulse Width Modulation or fast PWM mode provides a high frequency PWM waveform generation option. The fast PWM differs from the other PWM option by its single-slope operation. The counter counts from BOTTOM to TOP then restarts from BOTTOM. In fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the fast PWM mode is shown in Figure 3.2.

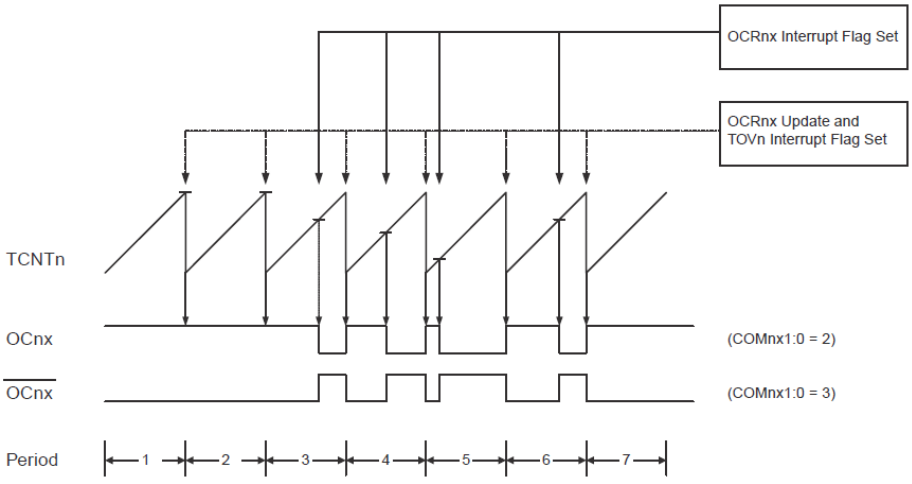


Figure 3.2 Fast PWM Mode, Timing Diagram

The PWM frequency for the output can be calculated by the following equation:

$$f_{OCnxPWM} = \frac{f_{clK\_I/O}}{N \cdot (1 + 255)} \tag{3.2}$$

$f_{clK\_I/O}$ represents clock frequency and the N variable represents the prescale factor (1, 8, 32, 64,128, 256, or 1024). Our microcontroller equipped with a 16 MHz crystal so $f_{clK\_I/O} = 16{,}000{,}000\ Hz$ and the sampling frequency of IMU sensor is 100 Hz (Table 3.2), so the frequency of the control loop is also 100 Hz, then the PWM frequency $f_{OCnxPWM}$ should be higher than 100 Hz so that each pulse can be fully send inside one control loop. To fulfill this requirement the value of prescale N has been chosen equal to 256. Now the PWM frequency is calculated using (3.2) :

$$f_{OCnxPWM} = \frac{16{,}000{,}000\ Hz}{256 \cdot (1 + 255)} = 244\ Hz \tag{3.3}$$

The period of PWM signal is:

$$T_{PWM} = \frac{1}{f_{OCnxPWM}} = 4.096\ ms \tag{3.4}$$

→Please refer to the **ATmega2560 datasheet**[6] for more information on this topic.

The code can be found in "motor_init()" and "run_motor(motor,speed)".

The function "run_motor(char motor,char speed)" send the PWM signals speed from 0 to 255 to the corresponding motor, the numbers of motor are: 1 to front rotor, 2 to back rotor, 3 to left rotor and 4 to back rotor; speed=0 means motor stops and speed=255 means the maximal rotational speed.

For example:

```
runmotor(1,255);        //the front rotor rotates at maximal speed
```

3.3 USART Interrupt Routine

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The ATmega2560 has four USART's, USART0, USART1, USART2, USART3. On the controller board RN-MEGA 2560 Model V1.0 (see www.robotikhardware.de) the USART3 is changed into the normal USB interface to the PC [14]. By using this interface we can send the data from microcontroller to the software "terminal" installed on the PC and capture the data into TXT file, and then these data can be plot by Matlab in order to compare the experiment results with the simulation results.

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDREn) and Transmit Complete (TXCn). Both flags can be used for generating interrupts. Here the UDRE3 is used for generating the data transmit flag. The UDRE3 Flag indicates whether the transmit buffer is ready to receive new data. When the transmit buffer is full, the UDR3 interrupt is enabled and start to transmit the data, after all the data in transmit buffer are sent, the transmit buffer is empty and the UDR3 interrupt is disabled.

The advantage of using the interrupt routine is that the main program and the data transmitting through serial are running in parallel, the data transmitting will not stop the main program. Otherwise, without interrupt routine, while the transmitting of the data, the main program will wait there until all the data has been transmitted, which will slow down the whole program.

According to the Table 3.4, double the USART Transmission Speed disabled (U2X=0), the baud rate of 76800 has been chosen which has small error value between an actual baud rate and target baud rate. With higher error the Receiver will have less noise resistance, especially for large serial frames.

Baud Rate (bps)	f_{osc} = 16.0000 MHz			
	U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%
76.8k	12	0.2%	25	0.2%
115.2k	8	-3.5%	16	2.1%
230.4k	3	8.5%	8	-3.5%
250k	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. (1)	1 Mbps		2 Mbps	

Table 3.4 Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

→Please refer to the **ATmega2560 datasheet**[6] for more information on this topic.

The Figure 3.3 show the data transmitting time measured by oscilloscope. From the figures we can see the sampling time of the IMU sensor and the control loop time are 10 ms. The data transmitting time is about 6 ms, that means every IMU data packet is possible to be sent to PC. The code can be found in “usart3.c”.

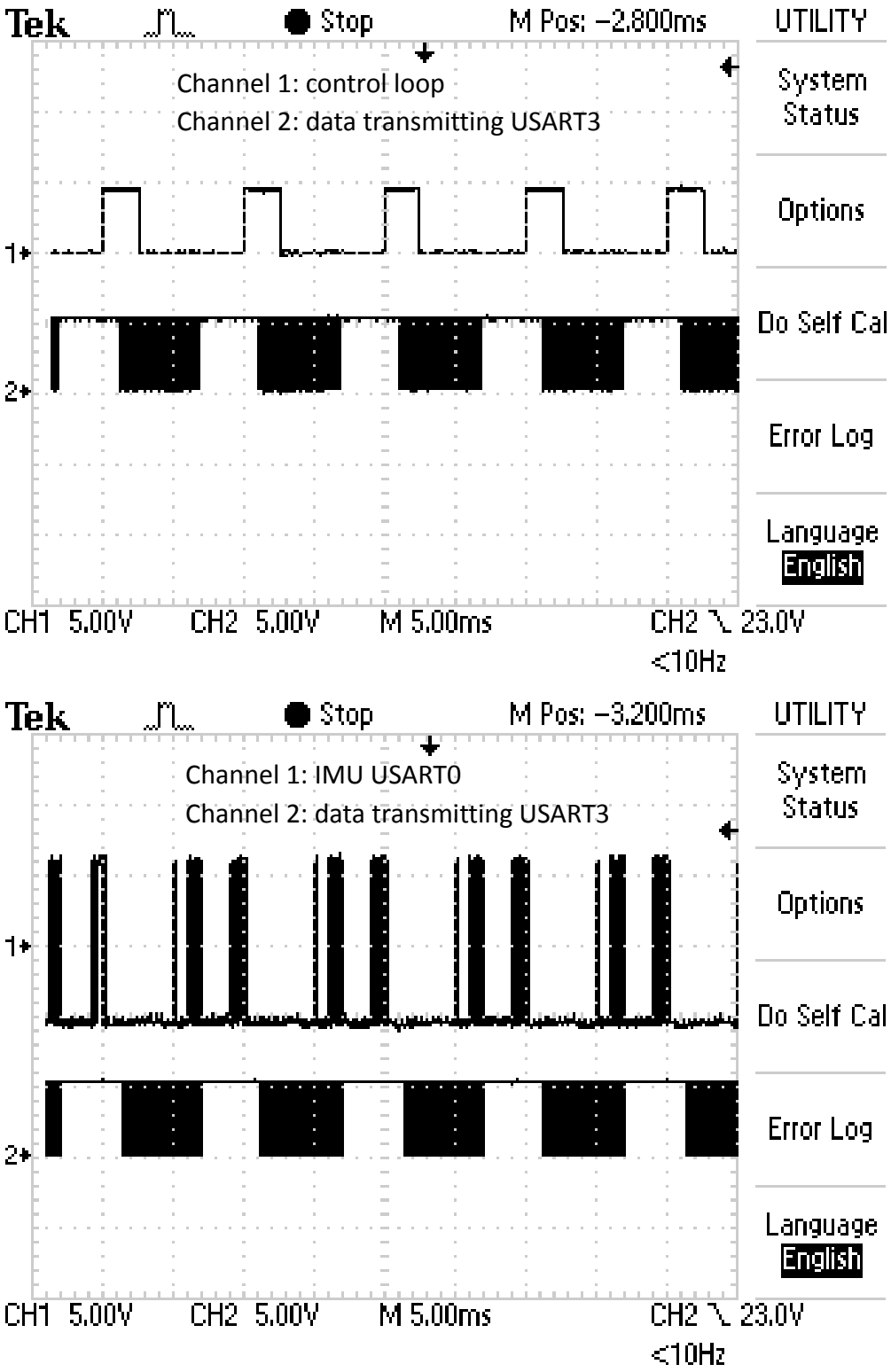


Figure 3.3 Data Transmitting Measurement

Chapter 4

Quadrotor Kinematics and Dynamics

There are lots of studies and papers have described how to model the quadrotor and since the modeling of the quadrotor is emphasis of this paper, so this chapter will give only give the overall model of the quadrotor dynamics used for simulation in [1]. More detailed information about modeling can be found in [5],[11],[15],[16].

4.1 Quadrotor Kinematics

Generally, the quadrotor can be modeled with a four rotors in cross configuration. The whole model can be considered as a rigid body. Figure 4.1 illustrates the basic motion control of the quadrotor.

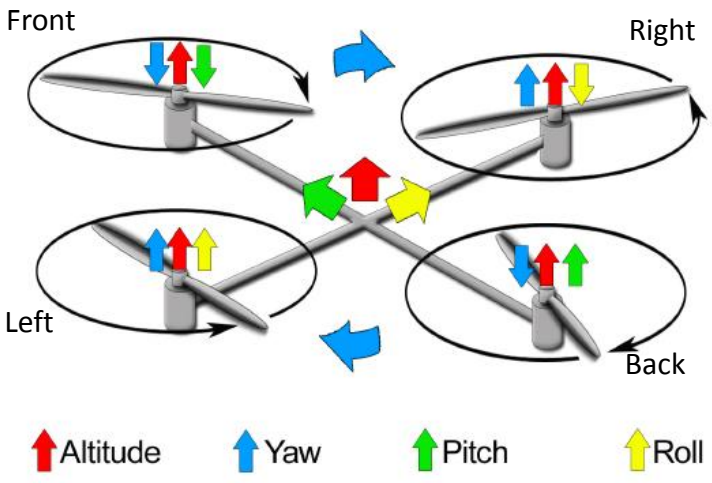


Figure 4.1 Quadrotor control mechanism [17]

- Throttle ($u_1\ [N]$)

  The throttle movement is provided by increasing (or decreasing) all the rotor speeds by the same amount. It leads a vertical force $u_1\ [N]$ with respect to body-fixed frame which raises or lowers the quadrotor.

- Roll ($u_2\ [N \cdot m]$)

  The roll movement is provided by increasing (or decreasing) the left rotors' speed and at the same time decreasing (or increasing) the right rotors' speed. It leads to a torque with respect to the green axis showed in Figure 4.1 which makes the quadrotor roll. The overall vertical thrust is the same as in hovering.

- Pitch ($u_3\ [N \cdot m]$)

  The pitch movement is provided by increasing (or decreasing) the front rotors' speed and at the same time decreasing (or increasing) the back rotors' speed. It leads to a torque with respect to the yellow axis showed in Figure 4.1 which makes the quadrotor go forward or backward. The overall vertical thrust is the same as in hovering.

- Yaw ($u_3\ [N \cdot m]$)

  The yaw movement is provided by increasing (or decreasing) the front-rear rotors' speed and at the same time decreasing (or increasing) the left-right couple. It leads to a torque with respect to the red axis showed in Figure 4.1 which makes the quadrotor turn in horizon level. The overall vertical thrust is the same as in hovering.

4.2 Quadrotor Dynamics

In order to model the quadrotor dynamics, two frames have to be defined, as showed in Figure 4.2:

- The earth inertial frame (e_I frame)
- The body-fixed frame (e_B frame)

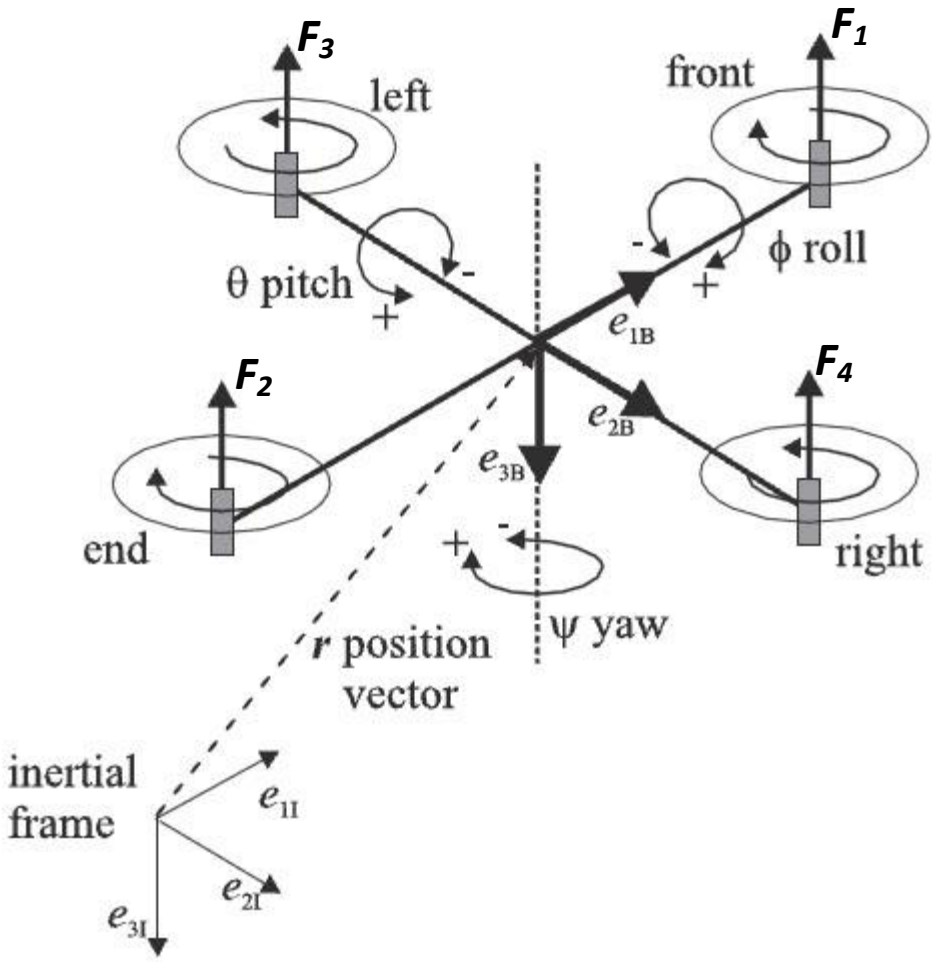


Figure 4.2 Configuration, inertial and body fixed frame of the quadrotor[1]

There are different ways to describe the dynamics of the quadrotor, such as Euler angle, quaternion and rotational matrix. In this work the orientation of the quadrotor is given by the three Euler angles, as mentioned in 14.1 Quadrotor kinematics, who are roll angle ϕ, pitch angle θ and yaw angle ψ, the units are $[rad]$. These three Euler angles form the vector $\boldsymbol{\Omega^T} = (\phi, \theta, \psi)$. The position of the vehicle in the inertial frame is given by the vector $\boldsymbol{r^T} = (x, y, z)$. The transformation of vectors from the body fixed frame to the inertial frame is given by the rotation matrix $\boldsymbol{R}$ where c_θ for example denotes $\cos\theta$ and s_θ denotes $\sin\theta$:

$$\boldsymbol{R} = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \tag{4.1}$$

The thrust force generated by rotor i, i=1, 2, 3, 4 is:

$$F_i = b \cdot \omega_i^2 \tag{4.2}$$

where b is the thrust factor and $\omega_i\ [rad/s]$ is the rotational speed of rotor i. Then the thrust force applied to the airframe from the four rotors is given by

$$T = \sum_{i=1}^{4} |F_i| = b \sum_{i=1}^{4} \omega_i^2 \tag{4.3}$$

Now the first set of differential equation that describes the acceleration of the quadrotor can be written as:

$$\ddot{\boldsymbol{r}} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = g \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} - \boldsymbol{R} \cdot \frac{T}{m} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.4}$$

With the inertia matrix $\boldsymbol{I}$ (which is a diagonal matrix with the inertias $\boldsymbol{I_x}$, $\boldsymbol{I_y}$ and $\boldsymbol{I_z}$ on the main diagonal), the rotor inertia $\boldsymbol{I_R}$, the vector $\boldsymbol{M}$ that describes the torque applied to the vehicle's body and the vector $\boldsymbol{M_G}$ of the gyroscopic torques we obtain a second set of differential equations:

$$\boldsymbol{I} \cdot \ddot{\boldsymbol{\Omega}} = -\left(\dot{\boldsymbol{\Omega}} \times \boldsymbol{I} \cdot \dot{\boldsymbol{\Omega}}\right) - \boldsymbol{M_G} + \boldsymbol{M} \tag{4.5}$$

The vector $\boldsymbol{M}$ is defined as (see Figure 4.2):

$$\boldsymbol{M} = \begin{pmatrix} Lb(\omega_3^2 - \omega_4^2) \\ Lb(\omega_1^2 - \omega_2^2) \\ d(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \end{pmatrix} \tag{4.6}$$

with the drag factor d and the length L of the lever. The gyroscopic torques caused by rotations of the vehicle with rotating rotors are:

$$\boldsymbol{M_G} = \boldsymbol{I_R} \cdot \left(\dot{\boldsymbol{\Omega}} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right) \cdot (\omega_1 + \omega_2 - \omega_3 - \omega_4) \tag{4.7}$$

The four rotational velocities ω_i of the rotors are the input variables of the real vehicle, but with regard to the obtained model a transformation of the inputs is suitable. Therefore, the artificial input variables can be defined as follows:

$$\begin{aligned} u_1 &= b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ u_2 &= b(\omega_3^2 - \omega_4^2) \\ u_3 &= b(\omega_1^2 - \omega_2^2) \\ u_4 &= d(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \end{aligned} \tag{4.8}$$

where $u_1 = T$ (4.3) denotes the thrust force applied to the quadrotor airframe; u_2 denotes the force which leads to the roll torque; u_3 denotes the force which leads to the pitch torque and u_4 denotes the force which leads to the yaw torque.

However, also the gyroscopic torques depend on the rotational velocities of the rotors and hence on the vector $u^T = (u_1, u_2, u_3, u_4)$ of the transformed input variables. We assume that:

$$g(u) = \omega_1 + \omega_2 - \omega_3 - \omega_4 \tag{4.9}$$

and then evaluation of (4.4) and (4.5) yields the overall dynamic model in the following form:

$$\ddot{x} = -(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \cdot \frac{u_1}{m}$$

$$\ddot{y} = -(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi) \cdot \frac{u_1}{m}$$

$$\ddot{z} = g - (\cos\phi \cos\theta) \cdot \frac{u_1}{m}$$

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{I_R}{I_x}\dot{\theta}g(u) + \frac{L}{I_x}u_2$$

$$\ddot{\theta} = \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) + \frac{I_R}{I_y}\dot{\phi}g(u) + \frac{L}{I_y}u_3$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}\left(\frac{I_x - I_y}{I_z}\right) + \frac{1}{I_z}u_4 \tag{4.10}$$

The entire dynamical model can be rewritten in state variable form $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{x} \in \mathbb{R}^{\mathbf{9}}$ is the vector of state variables.

$$\boldsymbol{x}^T = (\dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}) \tag{4.11}$$

Using (4.9) and (4.10) we obtain

$$\dot{\boldsymbol{x}} = \begin{pmatrix} -(\cos x_4 \sin x_5 \cos x_6 + \sin x_4 \sin x_6) \cdot u_1/m \\ -(\cos x_4 \sin x_5 \sin x_6 - \sin x_4 \cos x_6) \cdot u_1/m \\ g - (\cos x_4 \cos x_5) \cdot u_1/m \\ x_7 \\ x_8 \\ x_9 \\ x_8 x_9 I_1 - \frac{I_R}{I_x} x_8 g(u) + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{I_R}{I_y} x_7 g(u) + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{pmatrix} \tag{4.12}$$

with $I_1 = (I_y - I_z)/I_x$, $I_2 = (I_z - I_x)/I_y$ and $I_3 = (I_x - I_y)/I_z$. Here it has also been taken into account that reference variable of the quadrotor is a desired velocity vector and not a position vector. The system (4.12) will be simplified and widely used in the control algorithms design.

Chapter 5

Attitude Control Algorithm Design

The attitude control loop is responsible for the generation and stabilization of a currently required movement of the quadrotor, so that the quadrotor can also hover in the air. Later on, other control task might be developed for the quadrotor, such as altitude control, velocity of the quadrotor control, or the target tracking control and landing. All those control tasks are based on the attitude stabilization of the quadrotor. So we can consider the attitude control loop as a inner control loop which is more faster than the others. From a control engineering point of view, this means, the controller used for attitude control should be very fast to achieve the steady state and also has the ability to compensate against any disturbances.

With regard to control engineering, there are many different control concepts has been developed and studied for autonomous control of a quadrotor UAV, such as:

- Nonlinear control using feedback-linearization[1]
- Nonlinear State-Dependent Riccati Equation Control[18]
- Nonlinear and Neural Network-based Control[19]
- PD/PD^2 Controller Design[11]
- Enhanced PID Controller Design[15]

and so on. In this chapter, the “Nonlinear control using feedback-linearization” control theory has been chosen and implemented on the quadrotor to test the results. Furthermore, the new controllers have been designed using PID control.

The kinematics and dynamics of the quadrotor are well described in the previous chapter. However the most important concepts can be summarized in the set of equations (4.8) and (4.12). The state space model of the quadrotor (4.12) can be decomposed into one subset of differential equations that describes the dynamics of the attitude (i.e. the angles), which will be used later in this chapter for attitude control design, and one subset that describes the translation of the UAV[1]. From (4.12) we obtain the one subset of differential equations that describes the quadrotor's angular rate as:

$$\begin{pmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} x_8 x_9 I_1 - \frac{I_R}{I_x} x_8 g(u) + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{I_R}{I_y} x_7 g(u) + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{pmatrix} \tag{5.1}$$

where we have the state variables $\boldsymbol{x}^T = (\dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi})$.

The artificial input variables which related the basic movement to the propellers' squared speed are:

$$\begin{aligned} u_1 &= b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ u_2 &= b(\omega_3^2 - \omega_4^2) \\ u_3 &= b(\omega_1^2 - \omega_2^2) \\ u_4 &= d(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \end{aligned} \tag{5.2}$$

With the help of (5.1) and (5.2), we are able to design the attitude control for the quadrotor. In the following section, different approaches of the quadrotor attitude control design are explained and the corresponding controllers are translated from the form in simulation software Matlab™ into implementation form in C language. All control effect results of different controllers will be discussed in Chapter 6.

5.1 Nonlinear Control using Feedback-Linearization[1]

5.1.1 Control Algorithm Design

For design the attitude controller G_C the influence of the gyroscopic terms is neglected, which are comparatively small because of the small rotor inertias of the quadrotor. Then the simplified sub model can be obtained as:

$$\begin{pmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} x_8 x_9 I_1 + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{pmatrix} \tag{5.3}$$

Now we apply a feedback linearization in order to obtain a linear system:

$$\begin{aligned} u_2 &= f_2(x_7,\, x_8,\, x_9) + u_2^* \\ u_3 &= f_3(x_7,\, x_8,\, x_9) + u_3^* \\ u_4 &= f_4(x_7,\, x_8,\, x_9) + u_4^* \end{aligned} \tag{5.4}$$

with the new input variables $u_2^*,\ u_3^*,\ u_4^*$. In order to obtain a linear system, the following conditions must be fulfilled:

$$\begin{aligned} x_8 x_9 I_1 + \frac{L}{I_x} f_2(x_7,\, x_8,\, x_9) &= K_2 \cdot x_7 \\ x_7 x_9 I_2 + \frac{L}{I_y} f_3(x_7,\, x_8,\, x_9) &= K_3 \cdot x_8 \\ x_7 x_8 I_3 + \frac{1}{I_z} f_4(x_7,\, x_8,\, x_9) &= K_4 \cdot x_9 \end{aligned} \tag{5.5}$$

with the so far undetermined constant parameters $K_2,\ K_3,\ K_4$. Evaluation of (5.5) yields the nonlinear feedback for linearization:

$$f_2(x_7,\ x_8,\ x_9) = \frac{I_x}{L}(K_2 \cdot x_7 - x_8 x_9 I_1)$$

$$f_3(x_7,\ x_8,\ x_9) = \frac{I_y}{L}(K_3 \cdot x_8 - x_7 x_9 I_2) \tag{5.6}$$

$$f_4(x_7,\ x_8,\ x_9) = I_z(K_4 \cdot x_9 - x_7 x_8 I_3)$$

Using this feedback (5.3) turns into the linear and decoupled system:

$$\begin{pmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} K_2 x_7 + \frac{L}{I_x} u_2^* \\ K_3 x_8 + \frac{L}{I_y} u_3^* \\ K_4 x_9 + \frac{1}{I_z} u_4^* \end{pmatrix} \tag{5.7}$$

It can be shown that the resulting linearized closed-loop system is stable even if we consider the gyroscopic terms in (5.1). For that purpose we consider $u_2^* = u_3^* = u_4^* = 0$ and the operating point $x_7 = x_8 = x_9 = 0$. We define the Lyapunov function $V(x_7,\ x_8,\ x_9)$ which is attitude controller G_C and positive defined around the operating point:

$$V(x_7, x_8, x_9) = 0.5 \cdot (x_7^2 + x_8^2 + x_9^2) \tag{5.8}$$

Now we calculate the first derivative of V using the model (5.1) also including gyroscopic terms and the derived feedback (5.4), (5.6). In addition we assume a perfect cross configuration of the quadrotor with $I_x = I_y$ which results in $I_1 = -I_2$ and $I_3 = 0$. The derivative of the Lyapunov function then finally can be calculated as:

$$\begin{aligned} \dot{V} &= x_7 \dot{x}_7 + x_8 \dot{x}_8 + x_9 \dot{x}_9 \\ &= K_2 \cdot x_7^2 + K_3 \cdot x_8^2 + K_4 \cdot x_9^2 \end{aligned} \tag{5.9}$$

which is also independent from the gyroscopic terms. This derivative is negative defined if $K_2, K_3, K_4 < 0$ and this guarantees that the operating point of the feedback linearized system is asymptotically stable.

Taking into account that $\dot{x}_4 = x_7, \dot{x}_5 = x_8, \dot{x}_6 = x_9$ (see (4.12)) it becomes obvious that the dynamics of the angles using the linearized dynamics and neglecting the gyroscopic terms are described by linear decoupled differential equations of second-order, respectively.

Submit $\dot{x}_4 = x_7, \dot{x}_5 = x_8, \dot{x}_6 = x_9$ into (5.7) we obtain:

$$\begin{pmatrix} \ddot{x}_4 \\ \ddot{x}_5 \\ \ddot{x}_6 \end{pmatrix} = \begin{pmatrix} K_2\dot{x}_4 + \frac{L}{I_x}u_2^* \\ K_3\dot{x}_5 + \frac{L}{I_y}u_3^* \\ K_4\dot{x}_6 + \frac{1}{I_z}u_4^* \end{pmatrix} \tag{5.10}$$

If x_{4d}, x_{5d}, x_{6d} are the desired angles, application of the linear controller:

$$\begin{aligned} u_2^* &= w_2 \cdot (x_{4d} - x_4) \\ u_3^* &= w_3 \cdot (x_{5d} - x_5) \\ u_4^* &= w_4 \cdot (x_{6d} - x_6) \end{aligned} \tag{5.11}$$

with constant parameters w_2, w_3, w_4. Submit (5.11) into (5.10) and we apply Laplace transformation on (5.10) in order to transform the system (5.10) from time domain to s domain. This will lead to a closed-loop system of second order with the transfer functions:

$$\begin{aligned} F_4(s) &= \frac{X_4(s)}{X_{4d}(s)} = \frac{w_2}{I_x/L \cdot s^2 - K_2 I_x/L \cdot s + \omega_2} \\ F_5(s) &= \frac{X_5(s)}{X_{5d}(s)} = \frac{w_3}{I_y/L \cdot s^2 - K_3 I_y/L \cdot s + \omega_3} \\ F_6(s) &= \frac{X_6(s)}{X_{6d}(s)} = \frac{w_4}{I_z \cdot s^2 - K_4 I_z \cdot s + \omega_4} \end{aligned} \tag{5.12}$$

The dynamics of these closed-loop systems (5.12) can now be easily defined by adjustment of the pairs of parameters $(K_2, w_2), (K_3, w_3), (K_4, w_4)$, respectively, with the limitation that the parameter K_2, K_3, K_4 must be negative. The limitation from the hardware aspects (see 5.4 Motor Torque Design) should also be considered.

5.1.2 Control Algorithm Implementation

Submit (5.6) and (5.11) into (5.4) we can finally obtain the controller equations (5.13) which can be easily implemented in C language.

$$u_2 = \frac{I_x}{L}(K_2 \cdot x_7 - x_8 x_9 I_1) + w_2 \cdot (x_{4d} - x_4)$$

$$u_3 = \frac{I_y}{L}(K_3 \cdot x_8 - x_7 x_9 I_2) + w_3 \cdot (x_{5d} - x_5)$$

$$u_4 = I_z(K_4 \cdot x_9 - x_7 x_8 I_3) + w_4 \cdot (x_{6d} - x_6) \quad (5.13)$$

Where $x_7 = \dot{\phi}, x_8 = \dot{\theta}, x_9 = \dot{\psi}$, and x_{4d}, x_{5d}, x_{6d} are the desired angles. For hovering $x_{4d} = x_{5d} = x_{6d} = 0$.

The modeling of the quadrotor system (4.12) is done by using the S-Function of Matlab. In a first simulation, we assume an initial deviation of the angles $\Omega^T = (\phi = 20^\circ, \theta = -10^\circ, \psi = 10^\circ)$ where the control goal is to stabilize a hovering position. Figure 5.1 shows the control block diagram, the inverted movement matrix computes the PWM signals sent to rotors from the four artificial input variables (see 5.4).

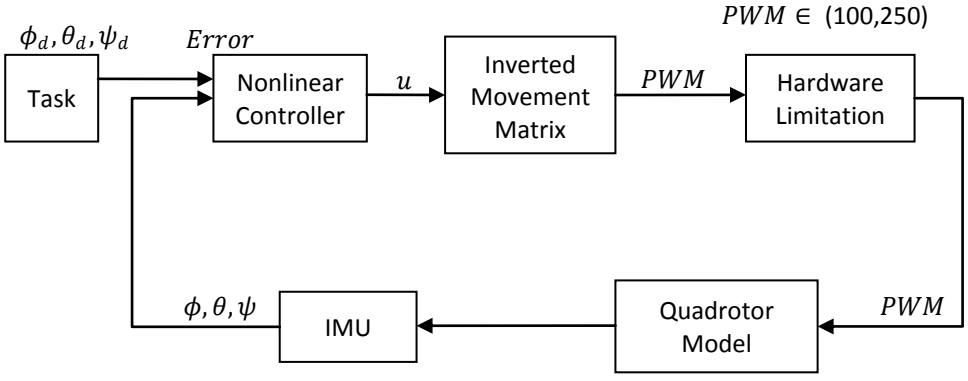


Figure 5.1 Nonlinear Control Block Diagram

5.2 Simple PD Controller

5.2.1 Controller Algorithm Design

In order to provide an easy inverse model which can be implemented in the control algorithms, the quadrotor dynamics must be simplified a lot. Equation (5.1) can be rearranged according following considerations:

- The gyroscopic terms can be neglected because the rotor inertias are small.
- The Coriolis-centripetal terms can also be neglected since the motion of the quadrotor can be assumed close to the hovering condition; the angular changes which come from cross coupling of angular speeds are smaller than the main.

Now the simplified linear model is showed in (5.14).

$$\begin{pmatrix} \ddot{x}_4 \\ \ddot{x}_5 \\ \ddot{x}_6 \end{pmatrix} = \begin{pmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} \frac{L}{I_x} u_2 \\ \frac{L}{I_y} u_3 \\ \frac{1}{I_z} u_4 \end{pmatrix} \tag{5.14}$$

Apply Laplace transform on (5.14) we obtain the set of transfer function for roll, pitch and yaw separately.

$$\begin{aligned} \frac{x_4(s)}{u_2(s)} &= \frac{L}{I_x s^2} \\ \frac{x_5(s)}{u_3(s)} &= \frac{L}{I_y s^2} \\ \frac{x_6(s)}{u_4(s)} &= \frac{1}{I_z s^2} \end{aligned} \tag{5.15}$$

It is obvious that each transfer function in (5.15) has two poles at the origin. Using the Root Locus Method to design the controller, we can place a zero using a PD-controller to improve the dynamic behaviour of the closed loop. The is used to obtain vertical lines as asymptotes for that branches of the root locus that converge to infinity and to obtain two dominate stable poles. With at least one pole in the origin the closed control loop of the system has no steady-state error. [20] The root locus graph is shown in following figure.

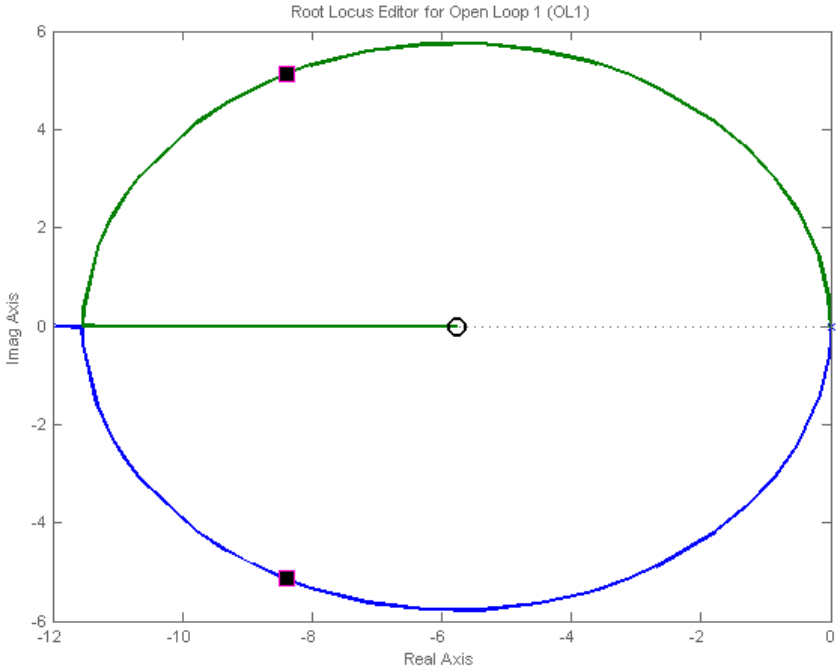


Figure 5.2 Root Locus using simple PD Controller

Now we have the simple PD controller for roll, pitch and yaw control with the form:

$$u(s) = \left(K_p + K_D \cdot s\right)e(s) \tag{5.16}$$

with K_p is the parameter of the proportional part and K_D is the parameter of the differential part. Since the real control loop is not a time continuous system but with the sampling time T of $0.01\,s$. Thus in Matlab simulation we use discrete PID control for continuous plant [21].

5.2.2 Controller Algorithm Implementation

According to the real control loop implemented on the microcontroller, we take a series of sampling time points kT_s instead of continuous time t, take numerical integration using rectangular method instead of integration approximately and take difference quotient with respect to time instead of differential, which are:

$$\begin{cases} t \approx kT_s \quad (k = 0, 1, 2, \cdots) \\ \int_0^t error(t)dt \approx T_s \sum_{j=0}^{k} error(j) \\ \frac{derror(t)}{dt} \approx \frac{error(k) - error(k-1)}{T_s} \end{cases} \tag{5.17}$$

then we can obtain the controller (5.16) in time discrete expression as:

$$u(k) = K_p error(k) + K_D \frac{error(k) - error(k-1)}{T_s} \tag{5.18}$$

The expression (5.18) can be directly implemented in C language.

Instead of the nonlinear controller in Figure 5.1 the controller block diagram is illuminated in Figure 5.3.

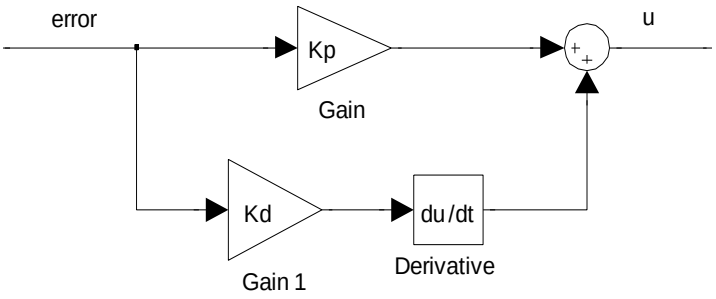


Figure 5.3 PD Controller Block Diagram

5.3 PD controller Design with Partial Differential

Instead of the simple PD controller, the PD controller with Partial differential is more widely used in industry area because of its better performance. Hence here we also design a new PD controller with partial differential based on the simplified system (5.15).

5.3.1 Controller Algorithm Design

In PID control theory, the introduce of the differential part can improve the dynamic characteristic of the system, but also make the system easily be disturbed by the high frequency signal, the drawback of the differential part is especially obvious when the error signal has a big disturbance. However, the system performance can be improved by adding a low pass filter to the control algorithm.

To overcome the drawback described above, one way is that adding a first order inertial element (low pass filter) $G_f(s) = 1/(1 + T_d s)$, which can improve the system performance. The structure of the PD controller with partial differential is shown in Figure 5.4, the low pass filter is added after the PD controller.[21]

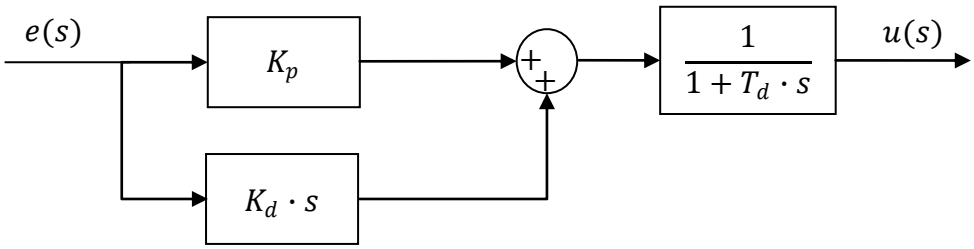


Figure 5.4 Block Diagram of PD Controller with Partial Differential

If we rearrange the PD controller shown in Figure 5.4 we can obtain the PD controller in the form:

$$G_C(s) = \frac{u(s)}{e(s)} = K_P \frac{1 + T_N s}{1 + T_d s} \tag{5.19}$$

with $T_N = K_D/K_P$ and $T_d \ll T_N$. The controller (5.19) is so called real PD-controller[20] and using the Root Locus method this controller is simply placing one real zero near to the origin as a dominate pole and placing one real pole far away from the added zero on the negative real axis of the Root Locus graph. Now we obtain the Root Locus diagram as shown in Figure 5.5.

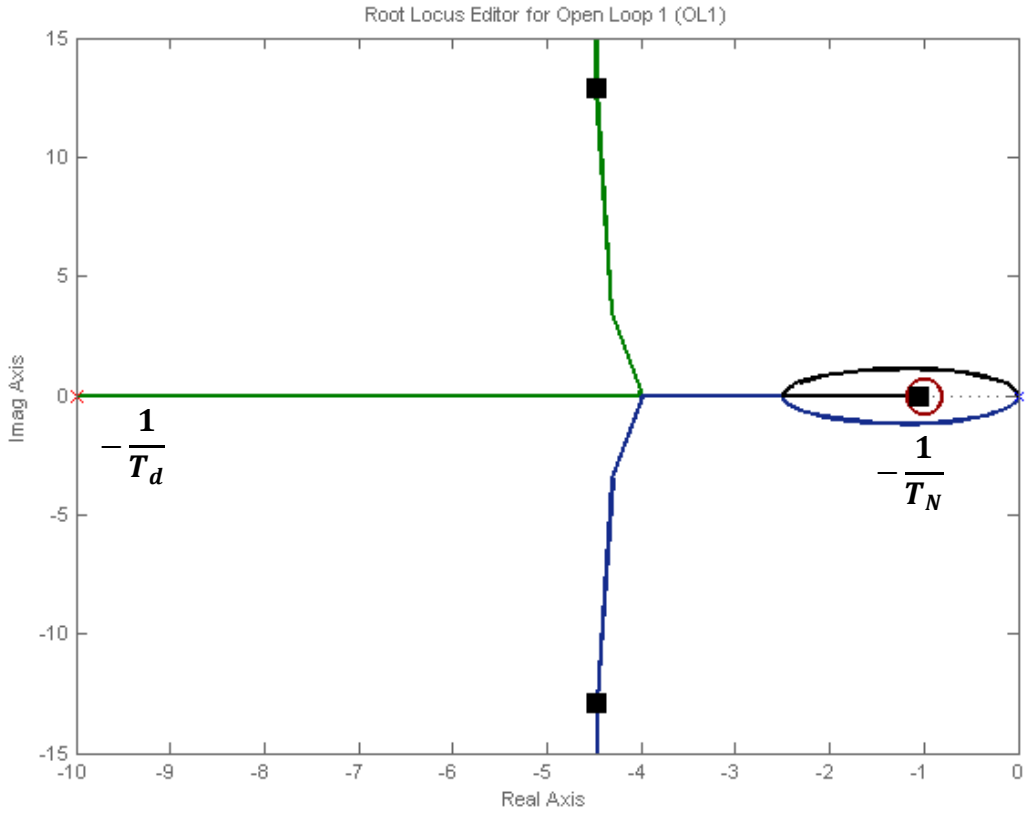


Figure 5.5 Root Locus using PD Controller with Partial Differential

5.3.2 Controller Algorithm Implementation

To implement the controller (5.19) on the microcontroller with the sampling time $T_s = 0.01\ s$, we need to transfer the controller into time discrete expression. The controller equation (5.19) can be rewritten as:

$$u(s) + T_d s \cdot u(s) = K_P\big(e(s) + T_N s \cdot e(s)\big) \tag{5.20}$$

by applying reverse Laplace transform on (5.19) we can transform the controller from s domain to t domain, which results:

$$u(t) + T_d \frac{du(t)}{dt} = K_P\left(e(t) + T_N \frac{de(t)}{dt}\right) \tag{5.21}$$

the differential parts in (5.21) can be approximately replaced by difference quotient with respect to time (5.17), which results:

$$u(k) + T_d \frac{u(k) - u(k-1)}{T_s} = K_P\left(e(k) + T_N \frac{u(k) - u(k-1)}{T_s}\right) \tag{5.22}$$

After the rearrangement of (5.22) it becomes:

$$u(k) = C_1 u(k-1) + C_2 e(k) - C_3 e(k-1) \tag{5.23}$$

with $C_1 = \frac{T_d}{T_s + T_d}, C_2 = K_P \frac{T_s + T_N}{T_s + T_d}, C_3 = K_P \frac{T_N}{T_s + T_d}$.

Now the controller can be easily implemented in C language.

5.4 Motor Torque Design

The task of motor torque design is to relate the controller output variables u_i to the desired rotor velocity $\omega_{d,i}$, but in this work the PWM signal is used to control the motor rotational speed. Therefore, with the help of motor force experiment done in section 2.7 and the linearized motor force characteristic with regard to the PWM signals, the controller output variables u_i are directly related to the PWM signals.

The output variables of the above mentioned controller are artificial input variables $u^T = (u_1, u_2, u_3, u_4)$ which are defined in (4.8), with (4.2) we can rewrite the u^T as:

$$\begin{aligned} u_1 &= F_1 + F_2 + F_3 + F_4 \\ u_2 &= F_3 - F_4 \\ u_3 &= F_1 - F_2 \\ u_4 &= \frac{d}{b}(F_1 + F_2 - F_3 - F_4) \end{aligned} \tag{5.24}$$

The (5.24) can be represented with the matrix format:

$$u = C \cdot F \tag{5.25}$$

where $F^T = (F_1, F_2, F_3, F_4)$ and

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \\ \frac{d}{b} & \frac{d}{b} & -\frac{d}{b} & -\frac{d}{b} \end{bmatrix}$$

According to (2.2) and Table 2.2 quadrotor constants we have

$$F = A + B \cdot PWM \tag{5.26}$$

with $PWM^T = (PWM_1, PWM_2, PWM_3, PWM_4)$ which are the PWM signals sending to the corresponding motors and

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \quad B = \begin{bmatrix} b_1 & 0 & 0 & 0 \\ 0 & b_2 & 0 & 0 \\ 0 & 0 & b_3 & 0 \\ 0 & 0 & 0 & b_4 \end{bmatrix}$$

From (5.25) and (5.26) we can get

$$PWM = B^{-1} \cdot [C^{-1} \cdot u - A] \tag{5.27}$$

The (5.27) can be rewritten as:

$$\begin{aligned} PWM_1 &= \frac{1}{b_1}\left(\frac{1}{4}u_1 + \frac{1}{2}u_3 + \frac{b}{4d}u_4 - a_1\right) \\ PWM_2 &= \frac{1}{b_2}\left(\frac{1}{4}u_1 - \frac{1}{2}u_3 + \frac{b}{4d}u_4 - a_2\right) \\ PWM_3 &= \frac{1}{b_3}\left(\frac{1}{4}u_1 + \frac{1}{2}u_2 - \frac{b}{4d}u_4 - a_3\right) \\ PWM_4 &= \frac{1}{b_4}\left(\frac{1}{4}u_1 - \frac{1}{2}u_2 - \frac{b}{4d}u_4 - a_4\right) \end{aligned} \tag{5.28}$$

The set of equations (5.28) can be easily implemented in C language to calculate the desired PWM signals. Take into account that the operating range of the motor speed $PWM \in (100, 250)$ for attitude control, it's necessary to limit the value of controller output variable with $u_2 \in (-0.8, 0.8)[N], u_3 \in (-0.8, 0.8)[N]$ and $u_4 \in (-0.3, 0.3)[N]$. This hardware limitation is always added to the controller in the implementation test. The operating range of PWM signals for each are not the same because there are big difference between the performances of the motors, see Table 2.1 Motor Lift Force Measurement.

Chapter 6

Simulation and Implementation Results

In this chapter, the plot diagrams of the simulation results using Matlab™ and the results of controller implementation on the quadrotor will be shown. According to the comparison between the simulation results and test results on real system, the effect of the controller should be described and discussed. After the analysis the test results, the existing problem and the corresponding reasons and the solutions will be explained in the conclusion. In order to understand the relationship between the PWM signals sent to the motor and the resulted quadrotor movement, the configuration of the quadrotor Euler angles will be restated in Figure 6.1.

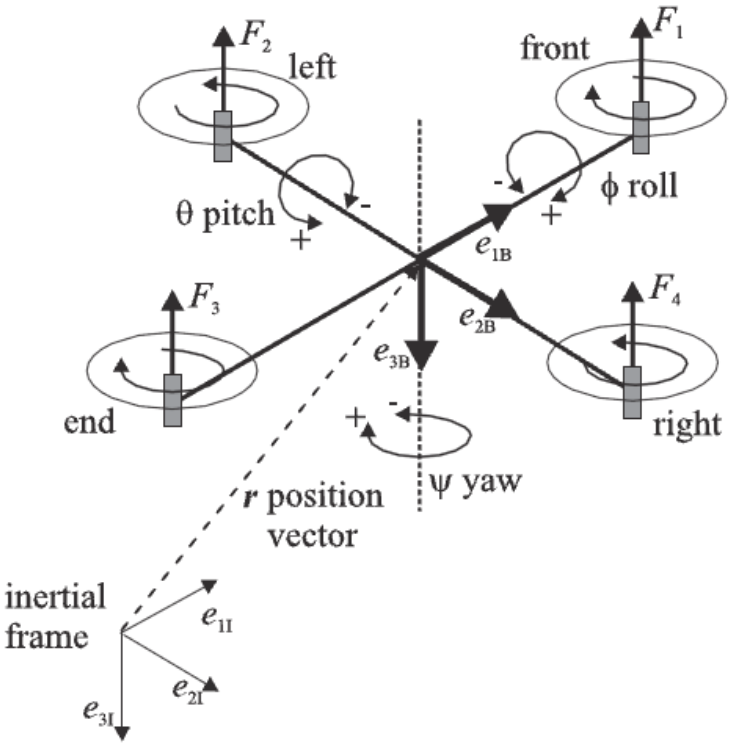


Figure 6.1 Configuration, inertial and body fixed frame of the quadrotor

6.1 Nonlinear Feedback Control

The nonlinear feedback controller has the form shown in (5.13), considering the hardware limitation with $\left(PWM_1 \in (100,200),\ PWM_2 \in (130,220),\ PWM_3 \in (110,210), PWM_4 \in (150,250)\right)$ the controller parameters has been chosen as:

$$K_2 = -30, K_3 = -30, K_4 = -15$$

$$w_2 = (0.7K_2)^2 I_x / L$$

$$w_3 = (0.7K_3)^2 I_y / L$$

$$w_4 = (0.5K_4)^2 I_z$$

The simulation results are shown in Diagram 6.1 and Diagram 6.2. The test results are shown in Diagram 6.3 and Diagram 6.4 with only the roll and pitch controlled simultaneity, Diagram 6.5 and Diagram 6.6 with roll, pitch and yaw controlled simultaneity. The PWM signals are shown in the way of differences between each pair of the rotors which provide the corresponding roll, pitch and yaw movement. Because of the limitation of the test platform, the freedom of roll and pitch angle are restricted between -20 and 20 degree, the freedom of the yaw angle is only restricted by the power supply cable which is between -360 and 360 degree.

From the test results we can see that the nonlinear feedback controller can stabilize the roll and pitch angles at the same time with only small error and also can compensate the disturbances quickly, see Diagram 6.3. However when the controller is applied to roll, pitch and yaw control at the same time, there is a big error with roll control which can't make the roll angle be stabilized to zero, see Diagram 6.5.

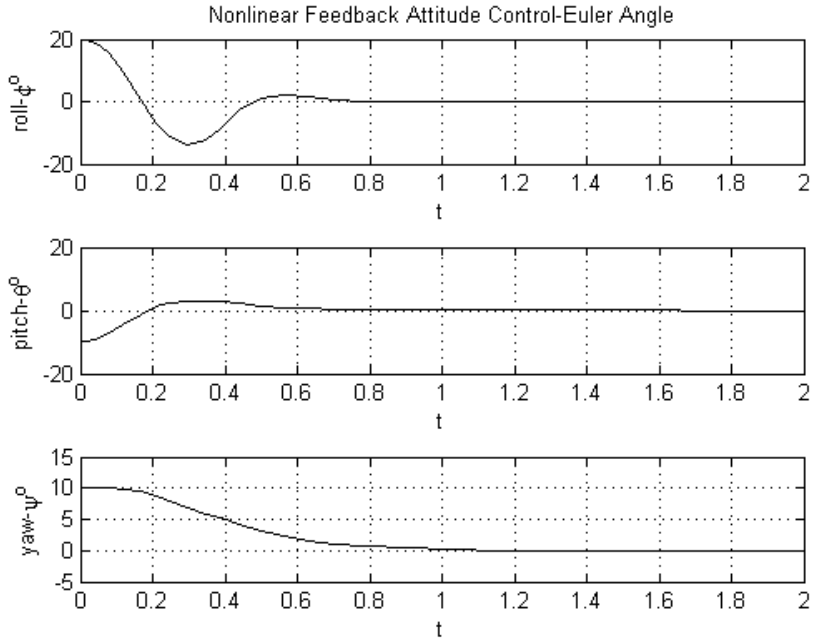


Diagram 6.1 Simulation Results, Nonlinear Feedback, Euler Angles

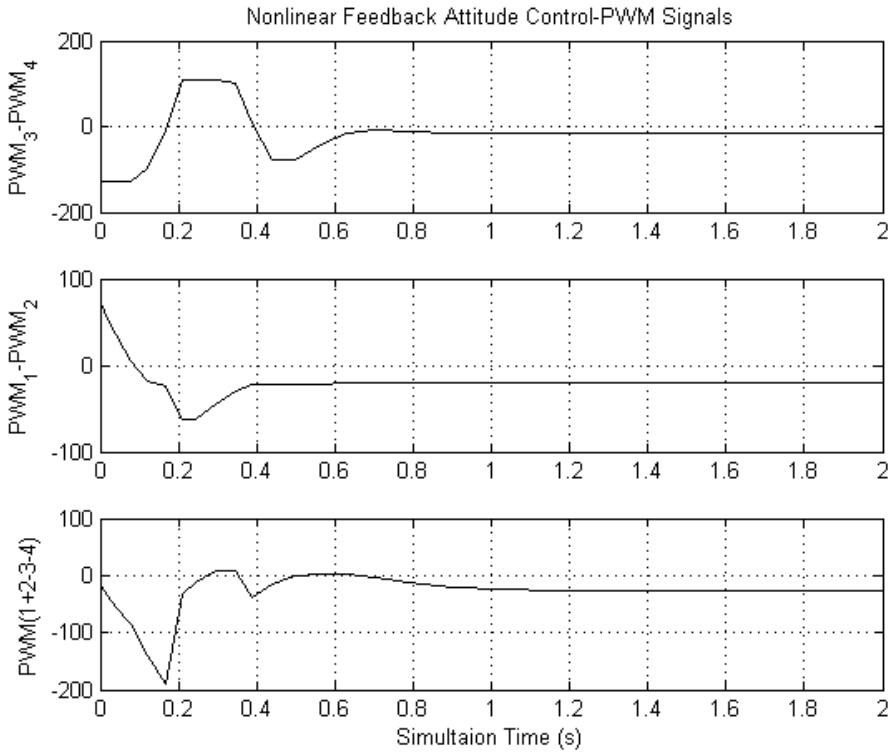


Diagram 6.2 Simulation Results, Nonlinear Feedback, PWM Signals

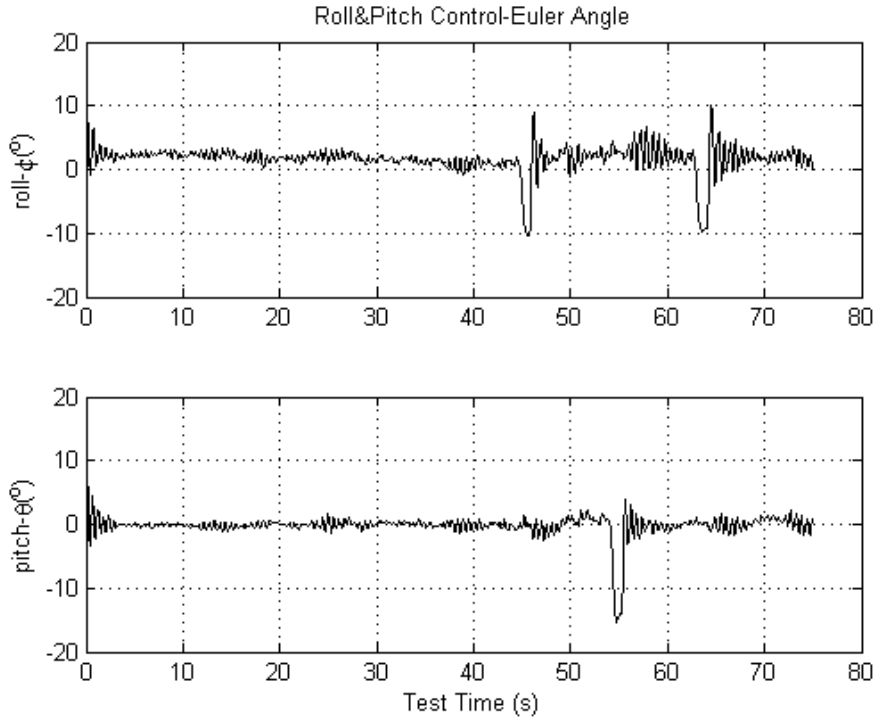


Diagram 6.3 Test Results, Roll & Pitch control, Euler Angles

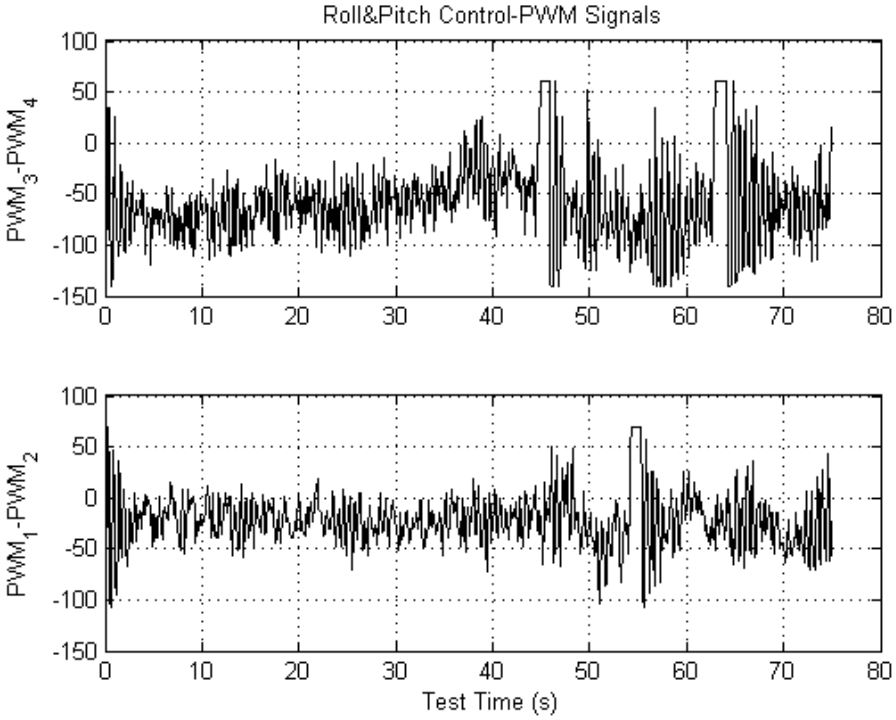


Diagram 6.4 Test Results, Roll & Pitch control, PWM Difference

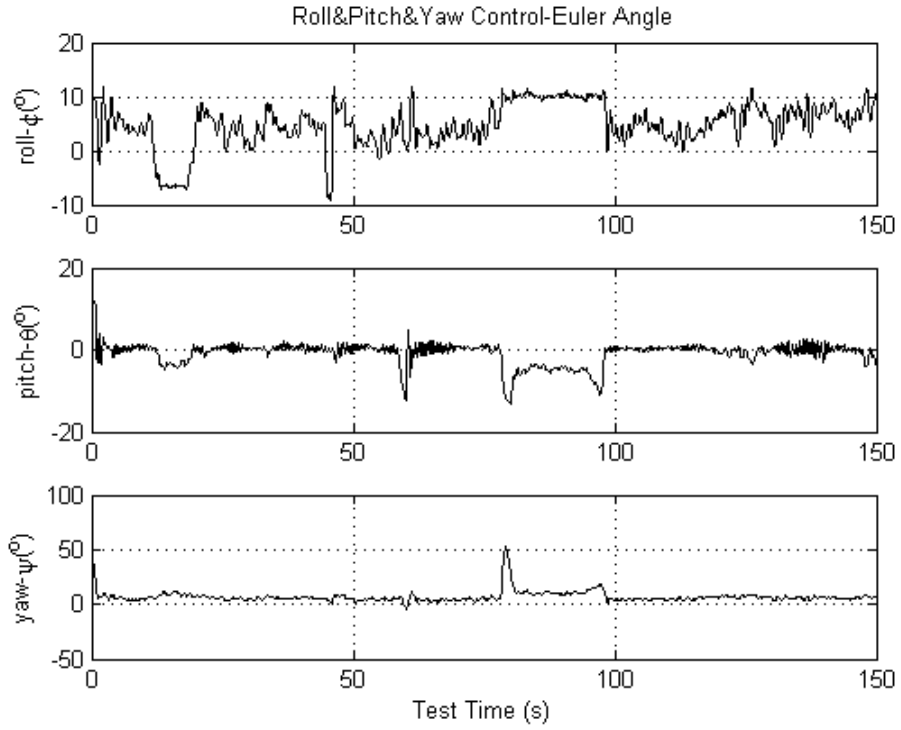


Diagram 6.5 Test Results, Roll, Pitch & Yaw control, Euler Angles

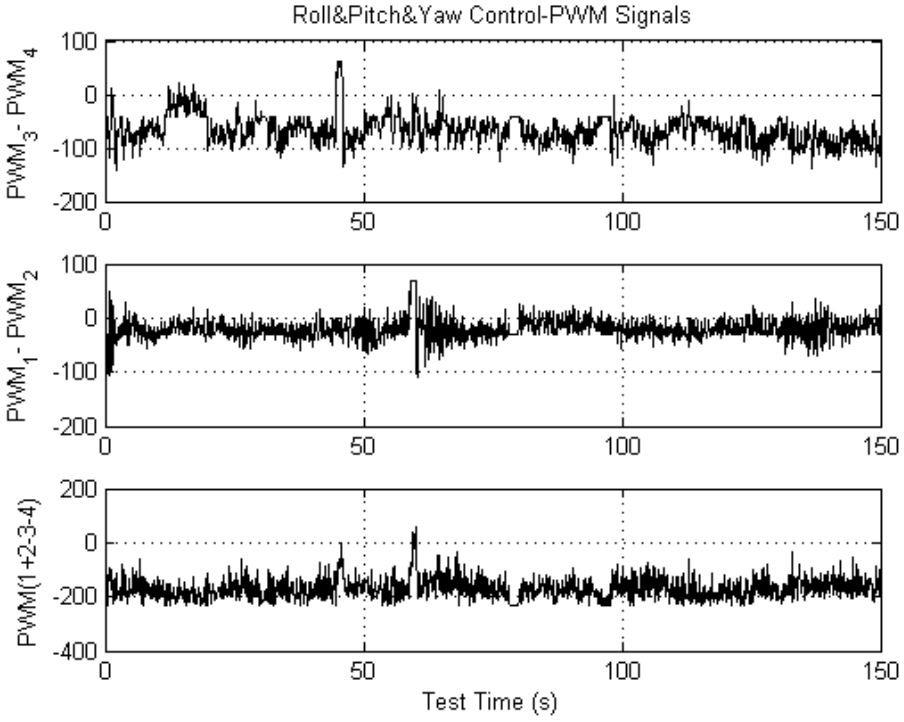


Diagram 6.6 Test Results, Roll, Pitch & Yaw control, PWM Difference

6.2 Simple PD Controller

According to the simulation and experiment, the parameter of simple PD controller (see (5.18)) has been chosen as:

Kp_roll=20; Kd_roll=0.85;

Kp_pitch=20; Kd_pitch=0.85;

Kp_yaw=10; Kd_yaw=0.65;

The simulation results are shown in Diagram 6.7. Diagram 6.8 shows the test results with that each time only one Euler angle is controlled. Diagram 6.9 shows the test results that roll and pitch angle were controlled simultaneity. Finally, Diagram 6.10 shows the test results that roll, pitch and yaw angle were controlled simultaneity with simple PD Controller.

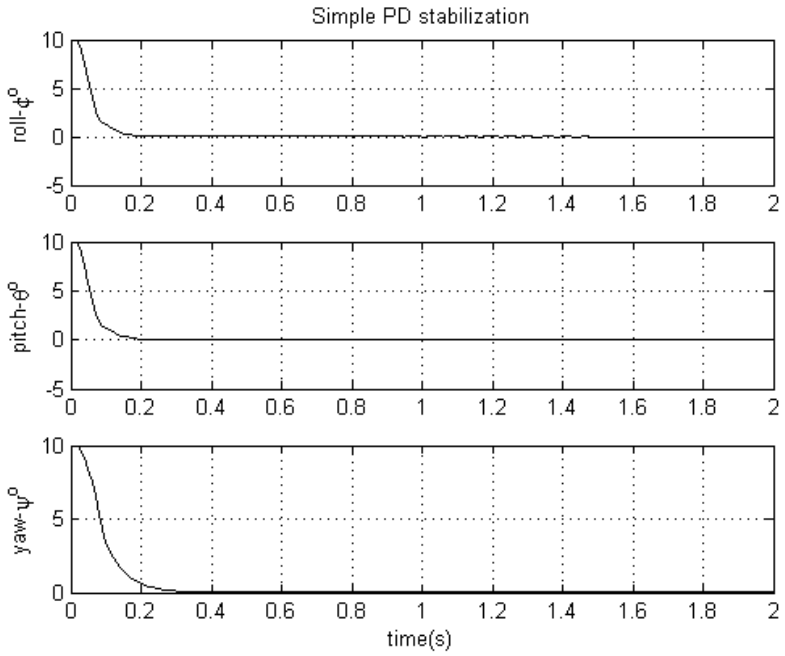


Diagram 6.7 Simulation results of Simple PD controller

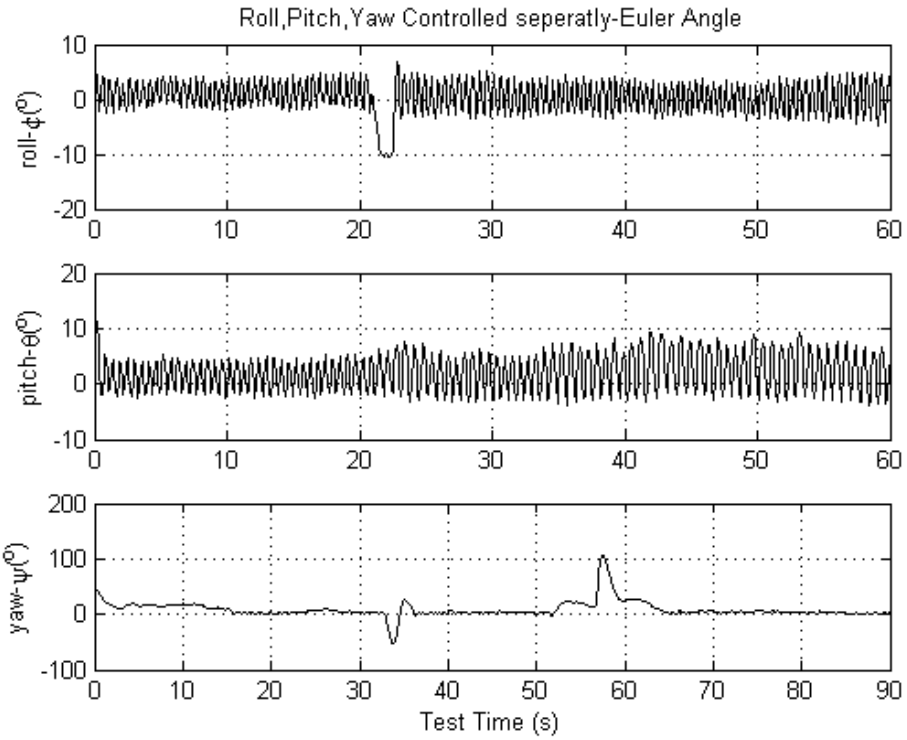


Diagram 6.8 Test results, simple PD, ϕ, θ, ψ are controlled separately

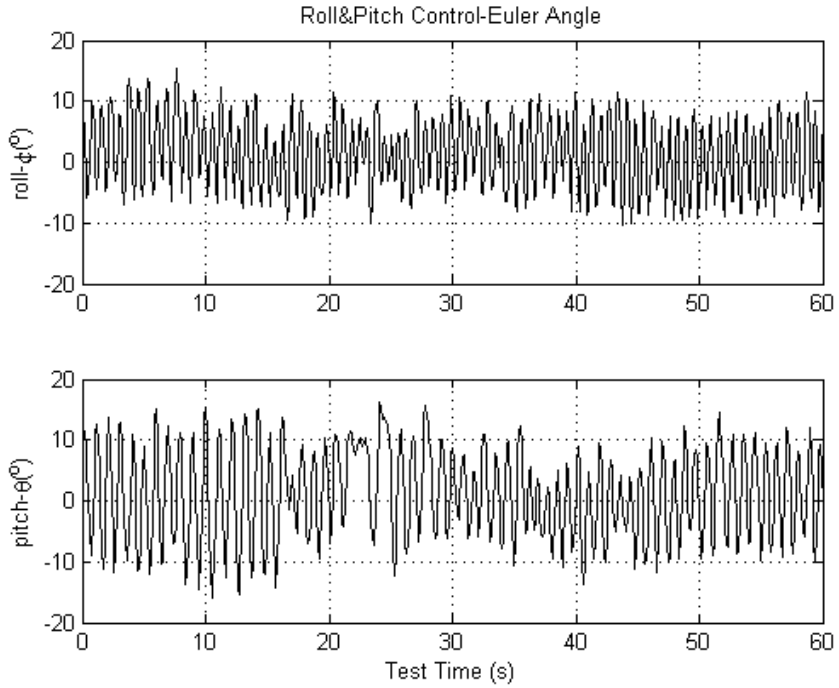


Diagram 6.9 Test results, simple PD, ϕ, θ are controlled synchronously

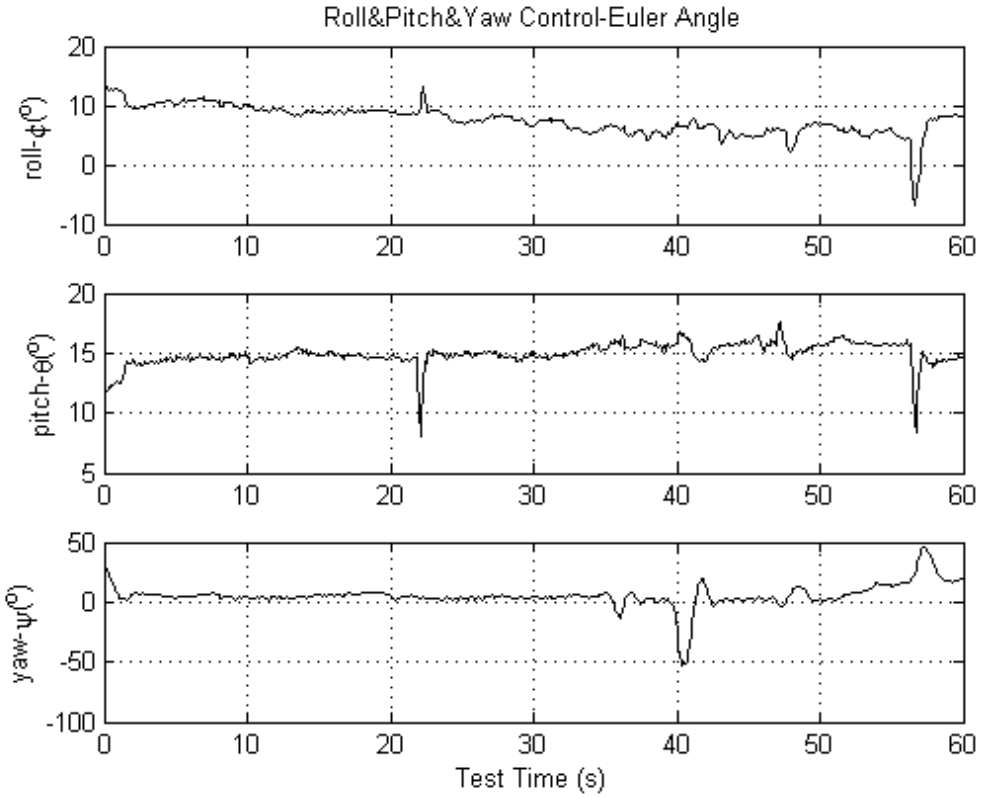


Diagram 6.10 Test results, simple PD, ϕ, θ, ψ are controlled synchronously

Form the test results it is obvious that the simple PD controller doesn't behave like what we got from the simulation results and is not sufficient to stabilize the quadrotor attitude in the hovering experiment. When we controlled the roll and pitch angle at the same time, there is a big and high frequency vibration, see Diagram 6.9. Later in Diagram 6.10 we can see the control effect by using simple PD controller to control roll, pitch and yaw angle at the same time, but the yaw controller is much faster and stronger than the other two, this leads to a very strong control effect on yaw angle while the other two have totally lost the controllability.

6.3 PD Controller with partial differential

Through the simulation results and experimental test of the control effect with different controller parameters, the final values for the real PD controller (see (5.23)) parameters were determined as:

K_p_Roll= 0.44;Tn_Roll=1; Td_Roll=0.1;

K_p_pitch= 0.48; Tn_Pitch=1; Td_Pitch=0.1;

K_p_Yaw= 0.48; Tn_ Yaw =1; Td_ Yaw =0.1;

Because of the differences between the motor performance (see Diagram 2.1), in order to control the roll and pitch angle more fast and effectively, the PWM_2 factor and PWM_4 factor was defined during the roll stabilize test. These factors are applied with following structure in the control algorithm code in C which will increase the corresponding PWM value with regard to the controller output variables u.

```
/**********PD controller with partial differential*****************************/
        if (u2>=0)    PWM4_faktor=0.5;
        else          PWM4_faktor=2.6;
        if (u3>=0)    PWM2_faktor=0.5;
        else          PWM2_faktor=0.9;
        PWM_1=(1/b1)*(0.25*u1+PWM2_faktor*u3+0.25*b*u4/d-a1);
        PWM_2=(1/b2)*(0.25*u1-PWM2_faktor*u3+0.25*b*u4/d-a2);
        PWM_3=(1/b3)*(0.25*u1+PWM4_faktor*u2-0.25*b*u4/d-a3);
        PWM_4=(1/b4)*(0.25*u1-PWM4_faktor*u2-0.25*b*u4/d-a4);
/*****************************************************************************/
```

The simulation results are shown in Diagram 6.11 by step response, the test results are shown in Diagram 6.13, Diagram 6.14, Diagram 6.15 and Diagram 6.12.

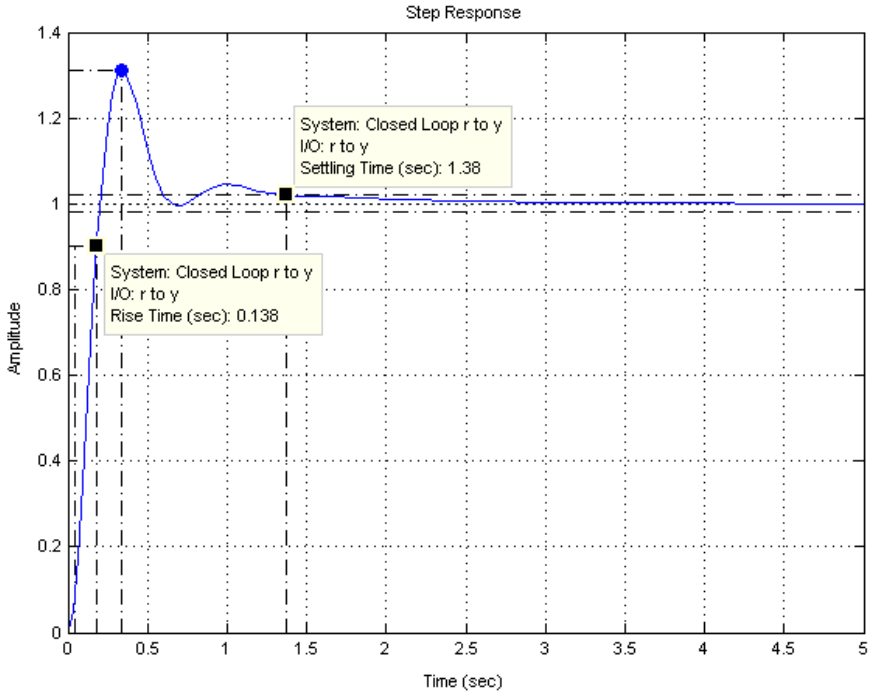


Diagram 6.11 Simulation results, Real PD controller, step response

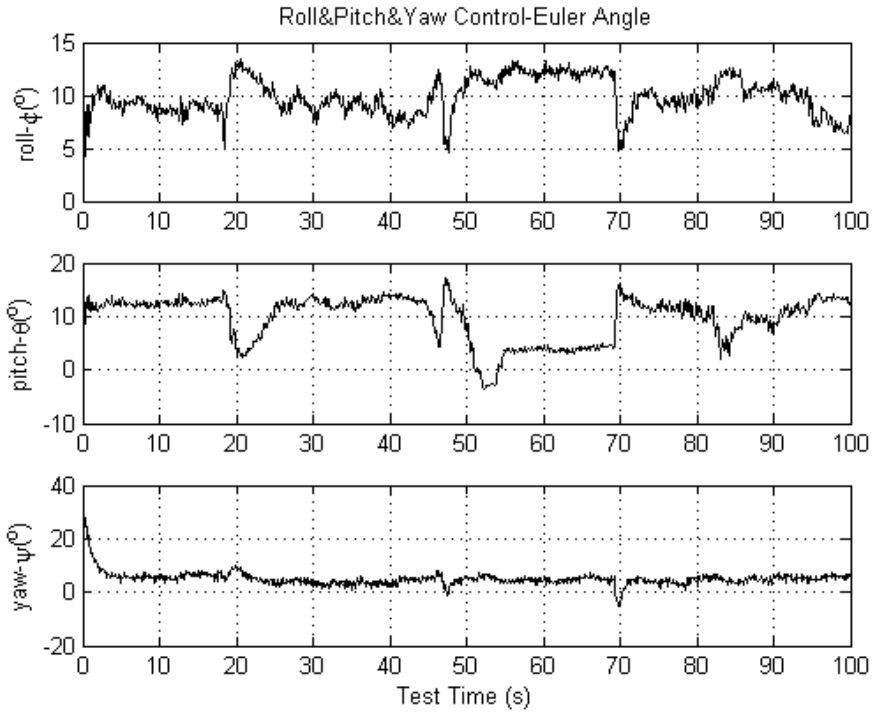


Diagram 6.12 Test results, ϕ, θ, ψ are controlled synchronously

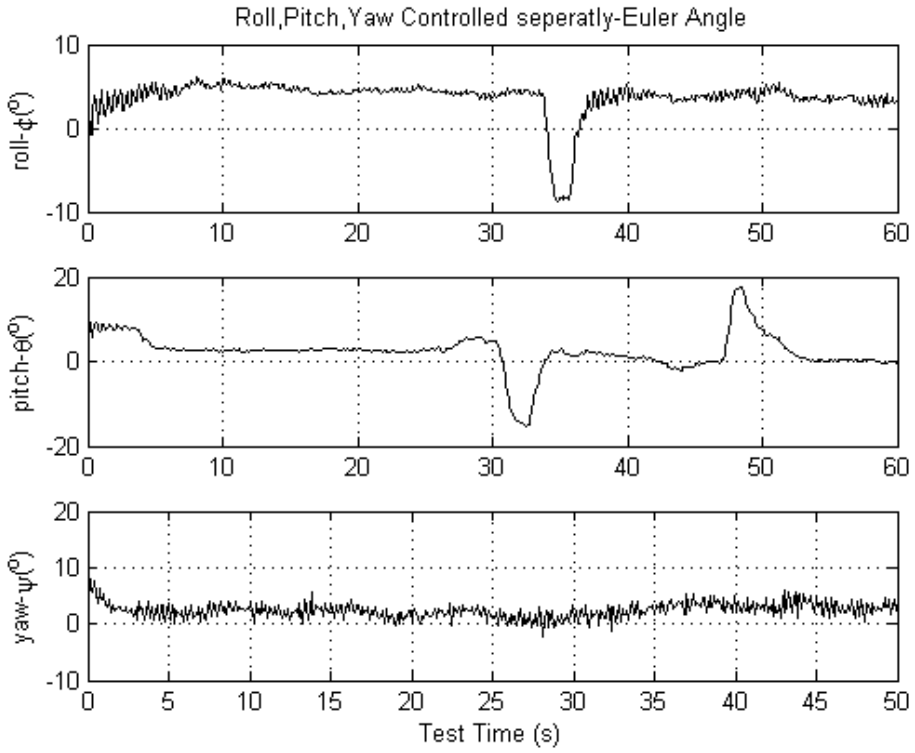


Diagram 6.13 Test results, Real PD, ϕ, θ, ψ are controlled separately

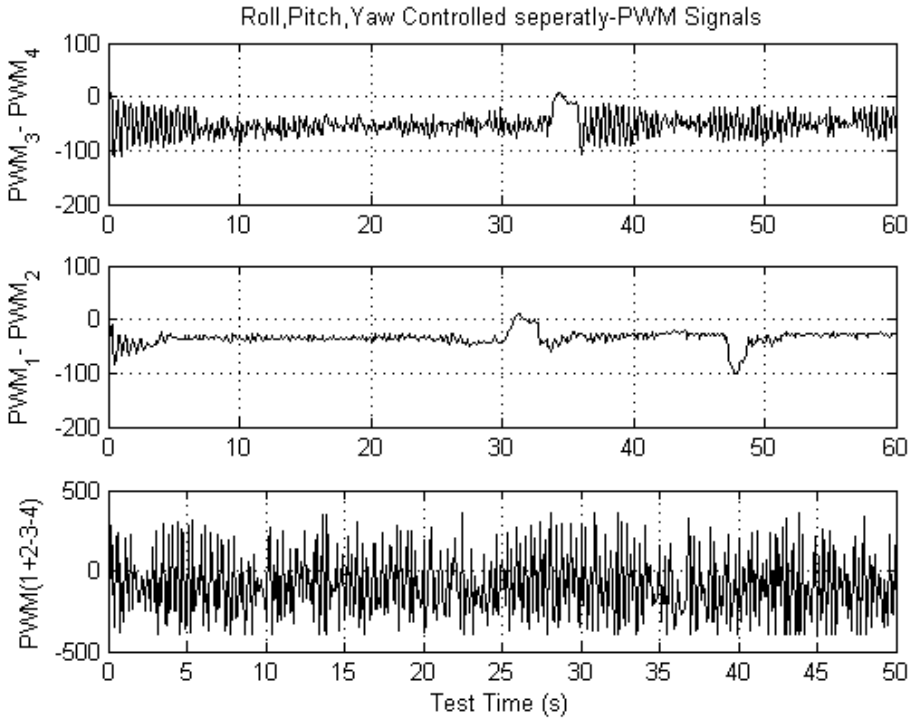


Diagram 6.14 Test results, Real PD, ϕ, θ, ψ are controlled separately, PWM

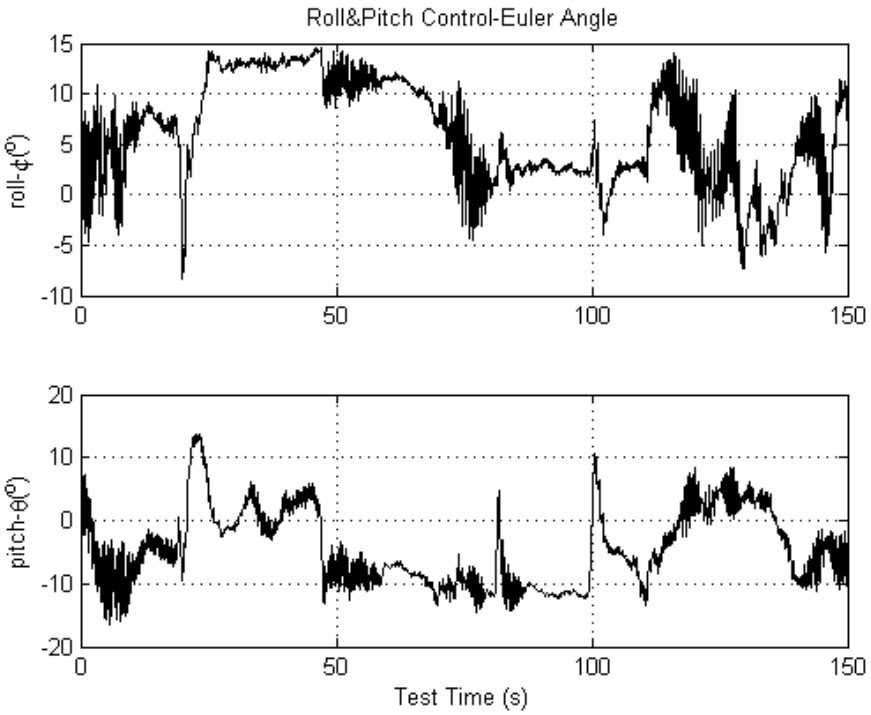


Diagram 6.15 Test results, ϕ, θ are controlled synchronously

From the test results we can see that the improved real PD controller has better control effect than the simple PD controller on controllability of roll, pitch and yaw angle separately. Comparing Diagram 6.13 and Diagram 6.8, it is obvious that the real PD controller has smaller vibration than the simple PD controller on the single roll and pitch control, however for yaw control, the controllability of the simple PD controller seems better.

If we look into the Diagram 6.15 and Diagram 6.9, either the simple PD or the real PD controller is not able to control the roll and pitch at the same time. When the three Euler angles are controlled synchronously, only yaw angle is under control, roll and pitch have lost their controllability, see Diagram 6.12.

Chapter 7

Conclusion

7.1 Comparison of the control algorithms

In Chapter 5 three control algorithms are developed over the control theory of nonlinear control using feedback-linearization (theory 1) and conventional PID technique (theory 2). Here we simply name the nonlinear controller as controller 1, simple PD controller as controller 2 and real PD controller as controller 3. All the simulation and experimental test results are given in Chapter 6. Based on these results we can compare these three controllers and find out the advantages and drawbacks of each controller. Since controller 2 and 3 are both PD controller but controller 3 is more advanced and widely used in industrial area than controller 2 and also the better controllability, in the following section we compare only controller 1 and controller 3 which will show the differences between the nonlinear control theory 1 and PID control theory 2.

From a control engineering point of view, theory 1 is more complicated and advanced than theory 2, since theory 1 directly takes the nonlinear dynamics of the vehicle into account, while theory 2 is the most used liner regulators in the industrial area with simple structure. From a hardware point of view, theory 1 is model dependent which means that the control effect is depend on the accuracy of the model parameters, while theory 2 is tunable even without a specific model of the controlled system.

From the experimental test results it is proven that the controller 1 has better control effect than the controller 3. By using PID technique the controller 3 can control the attitude only separately alone with roll, pitch or yaw, but if these three Euler angles are controlled at the same time, the interaction between the four motors causes the nonlinear dynamics. However the controller 3 using theory 2 is based on the simplified and linearized model. This is one of the reasons that results the failure to stabilize the quadrotor attitude by using controller 3. Now we have a look of control effect of controller 1 developed by using theory 1, at least it can stabilize the roll and pitch at the same time with small steady state error and rapid compensation against disturbances which basically allows the quadrotor to hover. Even when the roll, pitch and yaw angle are controlled together at the same time, the control effect of pitch and yaw are also good except roll control, see Diagram 6.3, Diagram 6.5. The reason caused roll control failure will be explained later in 0.

From the implementation of controller algorithm point of view, controller 1 is much easier to be implemented in C language than controller 3 since with the help of feedback-linearization the controller 1 has linear expression and there is no integration or differential part comparing with controller 3. From the experimental test point of view, because there is big difference between the model for simulation and the real quadrotor mode, so the controller parameters derived from simulation are actually not working fine in the real test. The optimized controller parameters need to be found during a number of tests which takes a lot of time.

Finally in summarize the control theory of nonlinear control using feedback-linearization is recommend in this paper to be used to develop control algorithm for the quadrotor attitude stabilization.

7.2 Project Contributions

This project makes the following contributions:

- Quadrotor Test Platform

  The test platform we ordered allows the flight test of quadrotor in-door which protects the mechanical structure of the quadrotor from the damages caused by unexpected flight failures. It also protects the people from getting hurt of the rotational propellers when the quadrotor is flying.

- IMU configuration

  The IMU sensor is configured to get the necessary data from the sensor which is ready to be used for attitude control.

- USART Interrupt Routine

  The USART Interrupt Routine is developed which allows attitude information and PWM information to be sent to PC in real time without slow down the control loop during the hovering test. This information can be stored and later on plotted by Matlab which helps to analyze and compare the control effect of different controller in parallel.

- Experimental Results

  The experimental results prove the advantages of using nonlinear control with feedback-linearization against PID technique which basically achieves final goal of the thesis, attitude stabilization of a quadrotor.

7.3 Future Work

The requirement of attitude stabilization of the quadrotor is still not perfectly fulfilled. There are several reasons which are responsible for that. Some works can be done in the future to eliminate or decrease the influences of these reasons.

- Replace the DC motor by brushless motor

  The motors play a very important role in quadrotor dynamics. In this project the brushed DC motors we used are coming with the original DraganFlyer and there is a big difference between the motors, see Table 2.1 Motor Lift Force Measurement, which is mainly responsible for the unexpected roll control failure shown in Diagram 6.5 and big steady state error of roll angle control shown in Diagram 6.13. Furthermore, after the running for a long time, the DC motors became hot and run with a low efficiency; however the brushless DC motor can provide a longer lifetime and higher efficiency.

- Replace the propellers

  The propellers we used are made of nylon which is a soft material. the purpose of using this kind of soft propellers is to prevent from hurting people when they are rotating with a high speed. However by high rotational speed of the propeller, because of the form changing the nylon propeller also lose the force. This will limit the payload of the quadrotor. If we use carbon propellers or some hard plastic propellers the quadrotor could have a better performance.

- Redesign the mechanical structure of the quadrotor

  The mechanical structure I designed in this project is only for the first step test, it's a little heavy and the connection between the test platform and the quadrotor is not with same length in X and Y direction which actually results the inertia value of $I_x \neq I_y$. This also leads to the different control effects which are unexpected for roll and pitch control.

- Identify the quadrotor constants by experiments

  The nonlinear controller depends a lot on the accuracy of the quadrotor constants. If the quadrotor constants are more accurate and close to the real model, it will help the quadrotor to have better attitude control performance by using nonlinear controller. The experiments to determine the quadrotor constants are described in paper [11].

Besides above mentioned works related to improvement of attitude control, the quadrotor provide interesting control challenges and impressive application possibilities. Based on this attitude control, the other more advanced control task can be implemented such as velocity control by using Inertial Navigation System (INS)[1] , height sensor to control the altitude, GPS sensor to control the position, wireless video sensor used for target tracking and so on. Although there has not been any fully autonomous quadrotor flight possible to date, there are still other groups who will continue to work on the quadrotor project to finally realize the autonomous flight.

List of Figures

List of Diagrams

List of Tables

Appendix

CD-ROM

Bibliography

[1]. H. Voos, “nonlinear Control of a Quadrotor Micro-UAV using Feedback-Linearization”, University of Applied Science Ravensburg-Weingarten, Germany

[2]. Katsuhiko Ogata, Modern control engineering, 4. edition, IBSN 0-13-043245-8, 2002

[3]. Schmitt Günter, Mikroconputertechnik mit Controllern der Atmel AVR-RISC-Familie, ISBN 978-3-486-58400-4, 2007

[4]. T. Goclel, R. Dillmann, Embedded Robotics, ISBN 3-89576-155-9, 2005

[5]. J. Bjørn, M. Kjærgaard, M. Sørensen, “Autonomous Hover Flight for a Quad Rotor Helicopter”, Automation and Control Department of electronic systems, AALBORG University, Denmark, 2007

[6]. MT Software Development Kit Documentation, Xsens Technologies B.V. , Revision F, Feb. 21, 2007, www.xsens.com

[7]. MTi and MTx Low-level communication Documentation, Xsens Technologies B.V., Revision F, Sep. 20, 2007, www.xsens.com

[8]. Mti and MTx User Manual, Xsens Technologies B.V., Revision I, Jan. 30, 2007, www.xsens.com

[9]. Df5ti-manual, draganFly Innovations Inc, www.rctoys.com

[10]. „BTS 621 L1“ motor controller datasheet, SIEMENS Semiconductor Group

[11]. A. Tayebi, S. McGilvray, Attitude stabilization of a four-rotor aerial robot, in Proc. of 43rd IEEE Conf. on Decision and Control, Atlantis, Paradise Island, Bahamas, 2004

[12]. M. Binswanger, Autonomous Airship, Development of a Navigation- and Control-unit for a Airship Model, University of Applied Science Ravensburg-Weingarten, March 2008

[13]. ATmega2560 datasheet, ATMEL, 2549l-AVR, July 2006, www.atmel.com

[14]. RN-MEGA 2560 Modul V1.0 datasheet, robotikhardware.de, Feb. 05, 2007

[15]. T. Bresciani, “Modeling, Identification and Control of a Quadrotor Helicopter”, Department of Automatic Control, Lund University, October 2008

[16]. S. Bouabdallah, P.Murrieri, R. Siegwart, “Design and Control of an Indoor Micro Quadrotor”, in Proc. Of the Int. Conf. on Robotics and Automation ICRA’ 2004, New Orleans, USA, 2004

[17]. Beau J. Tippetts, “Real-time implementation of vision algorithms for control, stabilization, and target tracking, for a hovering micro-UAV”, Brigham Young University, August 2008

[18]. H. Voos, “Nonlinear State-Dependent Riccati Equation Control of a Quadrotor UAV”, *In Proc. of the IEEE Trans. on Control Systems Technology*, VOL.12, No.4, July 2004

[19]. H. Voos, Nonlinear and Neural Network-based Control of a small Four-Rotor, *in Proc. of the IEEE/ASME Int. Conference on Advanced Intelligent Mechatronics*, Zurich, CH, 2007

[20]. H. Voos, Advanced Control, University of Applied Science Ravensburg-Weingarten, 2008

[21]. J. Liu, Advanced PID Control Matlab Simulation (second edition), Publishing House of Electronics Industry, Beijing, China, Mai 2007

[22]. Jie Chen, Matlab User Guide, Publishing House of Electronics Industry, Beijing, 2008