Design and Implementation of

The JAviator Quadrotor

an Aerial Software Testbed

DISSERTATION

zur Erlangung des akademischen Grades
Doktor der technischen Wissenschaften

Angefertigt am
Fachbereich für Computerwissenschaften
der Naturwissenschaftlichen Fakultät
der Paris-Lodron-Universität Salzburg

Eingereicht von
Dipl.-Ing. Rainer Karl Ludwig Trummer

Eingereicht bei
Univ.-Prof. Dr.-Ing. Christoph M. Kirsch

Salzburg, 2010

Copyright © 2010 by Rainer K. L. Trummer

To Irene, Ella, and Josef.

Abstract

Helicopters don't look very elegant either ... but they fly reasonably well.

Heinrich Dorfmann, *The Flight of the Phoenix*, 1965

In recent years, the research community has shown an increasing interest in autonomous aerial vehicles. Compared to fixed-wing aircraft that do not have the potential to hold some constant position in space, rotorcraft can be deployed in a much broader variety of missions, like search & rescue, traffic surveillance, area mapping, and wild-life monitoring, which are just a few examples of civil applications. Autonomous operation with respect to scout, observation, and also combat missions are further examples that give reasons for the particular interest indicated by military institutions. Besides the demand for high maneuverability, rotorcraft involved in such scenarios also need to provide sufficient flight durations, payload capacities, and robustness against disturbances.

The broad availability of various low-cost small-scale quadrotors that appeared over the last decade has established this type of rotorcraft as the de-facto standard platform chosen for research applications. However, the majority of research projects involving quadrotor vehicles is concerned with pure control engineering problems, paying only little or no attention to the rotorcraft design and software-specific aspects. Particularly, the platforms commonly consist of rather filigree airframes that suffer from low robustness and the propulsion systems provide insufficient payload capacities. Moreover, the control systems mostly represent classical event-triggered approaches that are typically tuned to some specific execution environment, and consequently, do not support time portability. In other words, the temporal behavior of the control-specific algorithm may vary or even be distorted largely upon changing the hardware platform.

To address these drawbacks and contribute some elaborate solutions, a high-precision quadrotor aircraft, called the JAviator, is designed and built completely from scratch. The JAviator quadrotor not only stands out from other vehicles of this kind with its high mechanical integrity, it also incorporates a very efficient propulsion system that provides both either high payload capacities or relatively long flight durations. The predominant method for providing positional feedback for autonomous indoor navigation is to employ vision-based sensing systems. As a low-cost alternative, radio-frequency-based localization

is used to enable position flight control. To cope with the quite low sensing accuracy offered by this system, a sensor-fusion-based state estimation scheme is developed. It is demonstrated that the achieved autonomous-hover accuracy represents a reasonably-well performance in relation to the error in the positional information. In order to facilitate changing the execution environment, a time-portable software architecture is presented. It is shown that the temporal behavior of the control system is preserved across various hardware platforms, even in the presence of high additional workload.

Contents

Abstract **i**

List of Figures **vii**

List of Tables **xi**

Acknowledgements **xiii**

1 Introduction **1**

1.1 JAviator Aircraft 2
1.2 Quadrotor Dynamics 3
1.3 Related Research 5
1.4 Main Contributions 10
1.5 Thesis Outline 14

2 Quadrotor History **17**

2.1 Bréguet-Richet Gyroplane No. 1 18
2.2 De-Bothezat-Jerome Helicopter 19
2.3 Œhmichen's No. 2 Helicopter 20
2.4 Convertawings, Model A 21
2.5 Curtiss-Wright VZ-7AP 23
2.6 Naito's Yuri I Helicopter 24
2.7 Spectrolutions, RC Flyers 26

3 Platform Development **29**

3.1 Airframe Construction 29
3.1.1 Design Elaboration 32
3.1.2 Composites and Joints 33
3.1.3 Parts and Assembly 35
3.2 Propulsion System 38
3.2.1 Power Transformation 39
3.2.2 Thrust Generation 41
3.3 Energy Distribution 42
3.3.1 Power Board Development 43

3.3.2 Brushless-Motor Controllers . . . 47
3.4 Sensor Equipment . . . 47
3.4.1 Ultrasonic Range Finder . . . 48
3.4.2 Barometric Measurement Unit . . . 48
3.4.3 Inertial Measurement Unit . . . 50
3.4.4 Stationary Localization System . . . 51
3.4.5 Laser Distance Sensor . . . 52
3.5 Computer System . . . 53
3.5.1 JAviator Avionics . . . 53
3.5.2 Ground Station . . . 54
3.5.3 Interconnectivity . . . 54

4 Control System Design **57**
4.1 Preliminaries . . . 57
4.1.1 Aircraft Orientation . . . 58
4.1.2 Coordinate Transformations . . . 59
4.1.3 Applied Control Principle . . . 60
4.2 Quadrotor Plant . . . 61
4.2.1 Sensor Measurements . . . 62
4.2.2 Sampling Characteristics . . . 62
4.3 Digital Filters . . . 63
4.3.1 DT Outlier Filter . . . 63
4.3.2 FIR Low-Pass Filter . . . 65
4.3.3 IIR Low-Pass Filter . . . 65
4.4 State Estimator . . . 66
4.4.1 Estimation Principle . . . 66
4.4.2 EKF Formalization . . . 68
4.4.3 EKF Implementation . . . 72
4.5 Motion Control . . . 78
4.5.1 Manual Flight Control . . . 80
4.5.2 Position Flight Control . . . 81
4.5.3 Spin Limit Extension . . . 82

5 Software Architecture **85**
5.1 Design Approach . . . 85
5.1.1 Logical Timing . . . 86
5.1.2 Communication . . . 88
5.1.3 System Layers . . . 88
5.2 JAviator Plant . . . 90
5.2.1 RoboMaster . . . 90
5.2.2 RoboSlave . . . 90
5.2.3 MockJAviator . . . 90

5.2.4 JAP-FCS Interface . . . 91
5.3 Flight Control System . . . 91
5.3.1 JControl . . . 92
5.3.2 EControl . . . 92
5.3.3 CControl . . . 93
5.3.4 TControl . . . 94
5.4 Ground Control System . . . 94
5.4.1 Control Terminal . . . 95
5.4.2 TuningFork . . . 97
5.4.3 Location Engine . . . 98
5.4.4 GCS-FCS Interface . . . 98

6 System Performance **99**
6.1 Platform Capability . . . 99
6.1.1 Lifting Capacity . . . 100
6.1.2 Flight Endurance . . . 101
6.2 Vehicle Controllability . . . 102
6.2.1 4-DOF Characteristics . . . 103
6.2.2 6-DOF Characteristics . . . 106
6.3 Time Portability . . . 110
6.3.1 Low I/O Load . . . 111
6.3.2 High I/O Load . . . 112

7 Conclusion **115**
7.1 Thesis Review . . . 115
7.2 Future Work . . . 116

A JAviator V1 Drawings **119**

B JAviator V2 Drawings **137**

Team Contributions **159**

Abbreviations **161**

Bibliography **165**

List of Figures

1 Introduction **1**

1.1 The JAviator quadrotor "aerial software testbed" 2
1.2 Fundamental flight maneuvers of a quadrotor helicopter 4
1.3 Outdoor flights performed in the courtyard of the Department of Computer Sciences at the University of Salzburg 11

2 Quadrotor History **17**

2.1 Bréguet-Richet Gyroplane No. 1 helicopter of 1907 18
2.2 De-Bothezat-Jerome helicopter of 1922 19
2.3 Œhmichen's No. 2 helicopter of 1922 21
2.4 Convertawings, Model A quadrotor of 1956 22
2.5 Curtiss-Wright VZ-7AP "Flying Jeep" of 1958 23
2.6 Naito's Yuri I human-powered helicopter of 1993 25
2.7 Dammar's 3rd prototype of 1991 and HMX-4 flyer of 1999 26
2.8 Spectrolutions, X-Pro flyer of 2002 and V-Ti flyer of 2004 27

3 Platform Development **29**

3.1 Prototype design of the JAviator V1 30
3.2 Initial design of the JAviator V2 31
3.3 JAviator V2 initial design versus JAviator V2 final design 33
3.4 Components in the wireframe graphics of a JAviator V2 girder 34
3.5 JAviator V2 girder connectors, girder flange, and fuselage connectors . . . 35
3.6 Demonstration of assembling a JAviator V2 fuselage 37
3.7 Demonstration of assembling a JAviator V2 girder 38
3.8 Top and bottom view of a rotor head and close-up view of a motor 39
3.9 Rotor equipped with custom-built blades and with X-Pro blades 42
3.10 PCB design of the JAviator Power Board 44
3.11 Fully populated JAviator Power Board 45
3.12 PCB design of the JAviator Sender Board 46
3.13 Devantech SRF10 digital URF and SensComp Mini AE analog URF 48

3.14 PCB design of the JAviator BMU 49
3.15 MicroStrain 3DM-GX1 IMU and mounted IMU-BMU stack 50
3.16 Ubisense Series 7000 SLS and Dimetix LSM2-15 LDS modules 52
3.17 Gumstix components and JAviator onboard computer system 53
3.18 Interconnectivity of JAviator Avionics and Ground Station 55

4 Control System Design **57**
4.1 Orientation in the Cartesian and aircraft coordinate system 58
4.2 Model of the JAviator control system 61
4.3 Performance of DT outlier filter and IIR low-pass filter 64
4.4 Model of the state estimation process 67
4.5 Performance of EKF provided with *synchronous* attitude observations and provided with *asynchronous* position observations 77
4.6 Performance of EKF provided with *asynchronous* velocity feedback and provided with *synchronous* velocity feedback 78
4.7 Model of the motion control process 81

5 Software Architecture **85**
5.1 Principle of logical timing according to the LET model 86
5.2 Timing source, task separation, and data flow in an event-triggered and in a time-triggered distributed control system 87
5.3 Packet format used for communication between JAP, FCS, and GCS . . . 88
5.4 JAviator software architecture separated into basic components 89
5.5 Screenshot of the Control Terminal with optional Position Monitor 95
5.6 Screenshot of the IBM TuningFork real-time logging system 97

6 System Performance **99**
6.1 Determination of maximum lifting capacity and battery service time 100
6.2 Indoor flights performed inside a classroom at the University of Salzburg and inside a corridor at the Charles University of Prague 103
6.3 Aggressive-commanding sequence issued during a manually piloted flight . 104
6.4 Fast-left-right-yawing sequence issued during a manually piloted flight . . . 105
6.5 RFID-based localization of a non-moving tag, estimates referring to the tag located at the center point, and 2D view of a position-hold flight . . . 107
6.6 3D visualization of the position-hold flight depicted in Figure 6.5 108
6.7 Hover sequence of the position-hold flight depicted in Figure 6.5 109
6.8 Low-I/O-load performance of the flight control software 111
6.9 High-I/O-load performance of the flight control software 112

A JAviator V1 Drawings **119**
A.1 Fuselage ring used as top and bottom central connecting junction 120
A.2 Fuselage plate used to carry the battery and mount the inertial sensor . . . 121
A.3 Airframe tubes used to assemble the fuselage and the four girders 122
A.4 Fuselage connector used to connect the top with the bottom fuselage ring . 123
A.5 Outer girder connector used to connect the girders with the rotor shafts . . 124
A.6 Inner girder connector used to connect the girders with the fuselage rings . 125
A.7 Rotor gear used to carry the rotor pylons and drive the rotor head 126
A.8 Rotor pylons used to mount the rotor blades and the connecting triangle . 127
A.9 Rotor shaft used to mount the rotor head and connect the girder ends . . . 128
A.10 Rotor bearings used to connect the rotor head with the rotor shaft 129
A.11 Rotor triangle used to connect the rotor pylons for more integrity 130
A.12 Motor carrier used to mount a motor and tighten the cogged belt 131
A.13 Rotor blade used to assemble the clockwise spinning rotors 132
A.14 Rotor blade used to assemble the counter-clockwise spinning rotors 133
A.15 Plan view of the complete airframe with all propulsion groups 134
A.16 Side view of the complete airframe with all propulsion groups 135

B JAviator V2 Drawings **137**
B.1 Fuselage ring used as top and bottom central connecting junction 138
B.2 Fuselage plate used to carry the battery and mount the inertial sensor . . . 139
B.3 Straight-airframe tubes used to assemble the fuselage and the four girders . 140
B.4 Inclined-airframe tubes used to assemble the fuselage and the four girders . 141
B.5 Fuselage connector used to connect the top with the bottom fuselage ring . 142
B.6 Girder connector used to connect to the fuselage rings and rotor shafts . . 143
B.7 Straight-airframe flange used to connect the girders to the fuselage rings . 144
B.8 Inclined-airframe flange used to connect the girders to the fuselage rings . . 145
B.9 Rotor gear used to carry the rotor pylons and drive the rotor head 146
B.10 Rotor pylons used to mount the rotor blades and the connecting triangle . 147
B.11 Rotor shaft used to mount the rotor head and connect the girder ends . . . 148
B.12 Rotor bearings used to connect the rotor head with the rotor shaft 149
B.13 Rotor triangle used to connect the rotor pylons for more integrity 150
B.14 Motor carrier used to mount a motor and tighten the cogged belt 151
B.15 Rotor blade used to assemble the clockwise spinning rotors 152
B.16 Rotor blade used to assemble the counter-clockwise spinning rotors 153
B.17 Plan view of the complete airframe with all propulsion groups 154
B.18 Cutaway plan view of the airframe with all propulsion groups 155
B.19 Straight-airframe side view including all propulsion groups 156
B.20 Inclined-airframe side view including all propulsion groups 157

List of Tables

3 Platform Development **29**
3.1 JAviator V1-versus-V2 general dimensions 32
3.2 JAviator V2 basic airframe components 36
3.3 JAviator V2 propulsion system components 41

6 System Performance **99**
6.1 Lifting capacities resulting from different gearings and thrusts 101
6.2 Battery service times resulting from different gearings and thrusts 102

Acknowledgements

Try not to become a man of success but rather to become a man of value.

Albert Einstein (1879−1955)

The road to this dissertation was paved with inspiration, creativeness, and fruitfulness, which is attributable in large part to the support, encouragement, and friendship of many people on my way, all of whom I owe a debt of gratitude.

In particular, I thank my supervisor, Prof. Christoph Kirsch, whom I had the honor to work with, a journey that was rewarding and fulfilling. Christoph's incredible academic enthusiasm, motivation, and guidance have greatly influenced me and the uncountably many discussions we went through often helped to improve my work. The fleet of JAviator aircraft, which I was granted to realize with unrestricted freedom of choice and action, would not exist without the supportive environment he provided. I thank my secondary advisor, Prof. Jochen Pfalzgraf, for his invaluable feedback on my progress and the many interesting conversations we had. He has been a supporting person for me from the beginning of my academic career and I deeply regret that he passed away before the completion of my degree. I thank the head of the department, Prof. Peter Zinterhof, who has been fond of this project and always willing to help when we encountered financial difficulties. I also would like to thank the chancellor of the university, Prof. Heinrich Schmidinger, who made it possible to rent the large basement room that serves as our JAviator lab and indoor testing site.

In the scope of the JAviator project, I had the great pleasure to collaborate with people from other research institutions. Particularly, I would like to thank David Bacon, Joshua Auerbach, and Vadakkedathu Rajan from the IBM T. J. Watson Research Center for their effort and work on the Java side of this project, leading to the first all-Java helicopter flight in history. I especially thank Daniel Iercan from the University of Timisoara for the initial control system version and his many contributions in various aspects over the years. I also would like to thank Gabe Hoffmann from the Stanford University for sharing his experiences regarding quadrotor control and providing invaluable feedback that helped to considerably improve the JAviator's control performance.

The Department of Computer Sciences has always been a nice and comfortable place to work, thanks to the support of many people around me. In particular, I thank my colleagues of the Computational Systems Group, who have made this group become an encouraging and fruitful environment for all of us. I especially thank Harald Röck, whom I had worked with from the beginning of the JAviator project, as well as Silviu Craciunas, who joint the project at some later point. I thank Ana Sokolova, Robert Staudinger, Hannes Payer, Andreas Haas, Andreas Löcker, Wolfgang Kreil, Clemens Krainer, Horst Stadler, and Bernhard Kast for the many scientific discussions, conversations far beyond science, and all the funny times we had at the department and our favorite place to go, the Müllner Bräu. I also would like to thank two special colleagues from other research groups, particularly, Thomas Soboll and Peter Hintenaus, who often helped with their specific knowledge to inspire solutions to problems that I encountered.

The present dissertation would probably not exist without the support, encouragement, and love of my parents, Ella and Josef, who have always shown understanding in my work, taken away certain duties from me, been helpful wherever they could, and never stopped believing in me. Therefor I owe them deep gratitude. I also very much thank my sisters, Gabriele and Roswitha, my brothers in law, and all of my friends for the relaxing times we spent together and encouraging me whenever I struggled with my studies.

My greatest thanks go to my girlfriend, Irene, who has gone with me through all the ups and downs that have come along with my work, provided strength and patience whenever I found myself lost, and with her loving support often helped me more than extensive trials. She is the source of the love and joy that I feel every day of my life.

Chapter 1

Introduction

We are at the very beginning of time for the human race. It is not unreasonable that we grapple with problems. But there are tens of thousands of years in the future. Our responsibility is to do what we can, learn what we can, improve the solutions, and pass them on.

Richard P. Feynman (1918−1988)

In Computer Science one is normally not faced with the development of physical devices and problems that arise when creating some piece of hardware. However, there are many physical phenomena that cannot be simulated sufficiently to solve complex engineering problems. Consequently, in some cases it is unavoidable to incorporate real hardware, especially if the underlying software highly depends on the physical environment, which, usually can only be simulated to some extent. This often leads to interdisciplinary projects that require fundamental knowledge in different fields of engineering. In this context, the engineer is faced with the rapidly arising problem of going beyond the area of personal expertise, entering unknown terrain, not being equipped with the required knowledge and tools, but equipped with the opportunity of not being influenced by field-specific, prevalent knowledge and common solutions. This can be seen as the most challenging, inspiring, and exciting task for any practically oriented engineer, because, in addition to the knowledge already gained, it requires bridging between different engineering fields, combining common principles, and creating individual solutions.

Regarding embedded software, aimed to execute on some embedded device, it is of vast importance to verify any solution on the specific target system, especially if developing "hard" real-time software that must adhere to strict deadlines. Despite this necessity, it is of course very advantageous to have some device for demonstration issues that allows for more tangible presentations than is possible with software alone. Inspired by these reasons, someday the idea was born to create our own testbed, which had to be mobile, preferably aerial, and something that could be realized by ourselves. Our considerations finally led to the development of the JAviator quadrotor [1], a high-precision electric model helicopter designed and built entirely from scratch.

Figure 1.1: The JAviator quadrotor, a high-precision "aerial software testbed".

The name JAviator is a combination of the two words *Java* and *Aviator* and refers to the early phase of the JAviator project [2]. At that time, the primary goal was to develop all software components exclusively in the programming language Java. To this end, except for the low-level drivers written in the programming language C, the entire flight and ground control software was originally developed in Java. In addition to standard Java, the flight control software was also implemented with Exotasks [3, 4], an extension to Java that enables real-time programming without changing the underlying Java semantics. Today, most of the JAviator's software is initially written in C, because the generated binaries do not require a virtual machine for execution and accessing hardware devices is somewhat more comfortable and easier to handle. In the scope of our work on Java programming abstractions with respect to time portability, the C-based solutions serve as performance baselines for the implementations in Java.

1.1 JAviator Aircraft

The fundamental question is, what kind of vehicle is the JAviator? The JAviator is, as the term "quadrotor" suggests, a four-rotor Vertical Take-Off and Landing (VTOL) vehicle, presented in Figure 1.1. Strictly speaking, because it is an unmanned aircraft, also known as *drone*, it is a so-called quadrotor Unmanned Aerial Vehicle (UAV). Regarding manned aircraft, quadrotor helicopters do not belong to state-of-the-art helicopters anymore. This is primarily due to the fact that, except for some historical models that can be found in designated museums, they are not produced in full scale anymore. Despite the reasons responsible for this specific evolution in flight history, quadrotors are on their way back into publicity again, at least in the scope of small-scale models.

The advantage of a four-rotor aircraft is that it can be built without a means for cyclic pitch control, as required in conventional single-main-rotor designs. A quadrotor is therefore mechanically simpler and less expensive in regard to production costs. However, small-scale quadrotors are − mainly due to their small sizes and weights − quite agile and by nature inherently instable, which demands at any rate a means of computer-facilitated attitude control. Albeit computerized attitude control basically suffices for most Remote Control (RC) quadrotors that are maneuvered manually, for quadrotor aircraft that are

intended to fly autonomously, automatic altitude control is absolutely indispensable and largely depends on sufficiently precise altitude measurements. In order to enable fully autonomous navigation, complete 3D position feedback is required, hence raising the need for some sophisticated localization system.

Compared to most other quadrotor aircraft that are in wide use today, one major goal of the JAviator project was to go beyond the prototypical stage of usually ungainly looking contraptions and try to provide some professional look right from the beginning. For this purpose, a quadrotor aircraft was developed that not only stands out with its high robustness and payload capacity, but also achieves a high degree of utilization and demonstrative impact. Two special features of the JAviator that make it unique are, first, its fully symmetrical airframe, resulting from an identical top and bottom frame, and second, the incorporation of custom-built brushless motors, which are significantly stronger than conventional motors.

Remark 1. In addition to the JAviator prototype, referred to as Version 1 (V1), the final and more advanced model, referred to as Version 2 (V2), was already produced in a small series consisting of ten identical aircraft. At the time of this thesis, five of them are fully operational and in use by ourselves as well as our research collaborators at the IBM T. J. Watson Research Center in Hawthorne, New York, USA.

1.2 Quadrotor Dynamics

The rotors of a quadrotor helicopter are always arranged in a cross-like shape, but not necessarily forming an exact square. Although this is the most prevalent arrangement that can be found today, some of the last full-scale quadrotors built between 1950 and 1960 introduced an "H"-like rotor configuration (see Section 2.4 and 2.5). In either case, all quadrotor aircraft have in common that one pair of opposite rotors spins clockwise and the other one counter-clockwise to cancel the rotors' torque reaction. Compared to conventional single-main-rotor helicopter designs that require a tail rotor for compensating the main rotor's torque reaction, a quadrotor can be considered representing an "optimal" solution with respect to the thrust-to-power ratio.

Another great advantage of a quadrotor is that there is no need for pitching the rotor blades. More precisely, in a rotor system with cyclic control, which can be seen as the de facto standard in today's helicopter designs, the rotor blades are pitched cyclically as they rotate around the rotor shaft. In other words, the "bite" into the air is regulated individually for each blade while rotating. In order to vary the helicopter's altitude, all blades are pitched collectively by the same amount, hence varying the blades' angle of attack by a certain degree over the complete cycle. Due to this technique of cyclically and collectively pitching the rotor blades, the helicopter can accomplish any directional changes that are physically possible. However, the mechanics involved in such a rotor system is usually quite complex, rather expensive regarding production costs, and needs to be maintained frequently (for a detailed explanation see [5]).

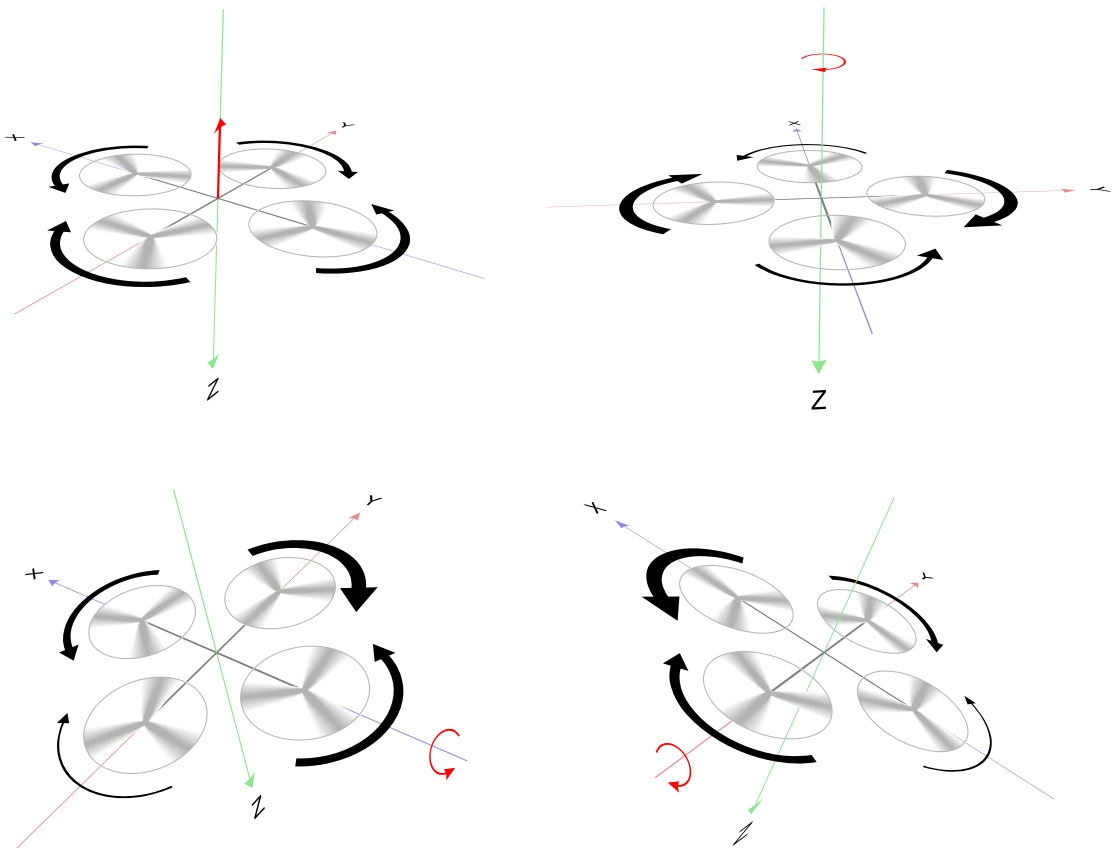


Figure 1.2: Fundamental flight maneuvers of a quadrotor helicopter: *climb* (left top), *yaw* (right top), *roll* (left bottom), and *pitch* (right bottom).

Referring to the basic quadrotor design, the blades are mounted in some fixed position that does not allow for any change concerning the blades' angle of attack. Differences in the amount of thrust generated by such a fixed-pitch rotor are obtained by merely regulating its rotation speed. In fact, any physically possible movement of a fixed-pitch quadrotor can be accomplished by individually regulating the speed of its four rotors. Although this sounds fairly simple, in reality it is quite difficult to successfully control quadrotors due to their strong tendency to instability. The reason is that the complexity of a helicopter's thrust dynamics – and that counts for any type of rotorcraft – grows with the number of rotors involved. Specifically, changing the speed of just one rotor in a four-rotor system inevitably influences each of the other three rotors.

In order to deal with a quadrotor's complex dynamics, the amount of possible motions can be separated into the four fundamental VTOL maneuvers *climb*, *yaw*, *roll*, and *pitch*. Figure 1.2 illustrates how these basic flight maneuvers are performed with a quadrotor. In particular, climbing is achieved by collectively varying the speed of all four rotors. In other words, each rotor's speed is increased, or decreased in case of descending, by the same magnitude. Yawing to the left or to the right is accomplished by varying the speed ratio between the two pairs of opposite rotors. In this sense, yawing, which is avoided in

normal operation by regulating the rotor speeds such that the torque reaction of clockwise and counter-clockwise spinning rotors cancels, is achieved by simply utilizing this effect of unbalanced torques. For rolling to the left or to the right, the speed ratio between the left- and the right-hand-side rotor is varied, and similarly, for pitching up or down, the speed ratio between the front and the rear rotor is varied.

1.3 Related Research

Regarding custom-built quadrotors, which range from micro-scale and ultralight vehicles to large and heavy aircraft, a huge variety of individual platforms exists today. Besides these mostly hand-made vehicles, many research groups, especially those concerned with four-rotor control, involve commercially available RC quadrotors. Sometimes the original shape is kept and they are merely equipped with markers for motion tracking, sometimes the RC electronics is replaced with an avionics system comprising of several sensors, and sometimes only the motors and gear boxes are replaced with more efficient propulsion groups. In either case, the most prevalent platforms exploited for this purpose are the quadrotors of the Draganflyer series, which rapidly became very popular platforms since the first commercial release in 1999. In particular, the Draganflyer quadrotors comprise of the HMX-4, also known as the Roswell Flyer, the Draganflyer I through Draganflyer V-Ti Pro, all distributed by the Draganfly Innovations Inc. [6], as well as the larger Dragenflyer X-Pro, meanwhile distributed under the name X-Pro Flyer by the VeraTech Aero Corp. [7]. Further commercially available RC quadrotors that can be found in academic projects are the Hummingbird from the Ascending Technologies GmbH [8], the Mikrokopter from an open-source project [9], and the X-UFO from the Silverlit Inc. [10]. Potential platforms of this kind that might appear in academic projects someday are the MD4-200 and MD4-1000 from the Microdrones GmbH [11] as well as the AR100-B from the AirRobot GmbH & Co. KG [12].

Albeit incomplete, because of the great many of publications that could only be covered to some extent, the following survey provides a brief, chronological overview (authors in alphabetical order) about related research work published in recent years.

2004. Bouabdallah *et al.* [13] present the design, modelling, and control scheme of the Omnidirectional Stationary Flying OUtstretched Robot (OS4) indoor micro-quadrotor and provide experimental results obtained from tethered flights performed on a test bench. In a subsequent paper [14], they present an evaluation of two different control techniques, particularly, Proportional Integral Derivative (PID) and Linear Quadratic Regulator (LQR) control. In contrast to the modern LQR controller, which failed in the experiments, the classical PID controller enabled the first free-flights of their OS4 craft.

Hoffmann *et al.* [15] introduce the Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC), which incorporates a Draganflyer III equipped with an onboard computer system. Based on Sliding Mode control for altitude stabilization

and LQR control for attitude stabilization, the free-flight performance of the STARMAC I quadrotor with respect to manual piloting is demonstrated.

Pounds *et al.* [16] report on realizing their custom-built Mark II quadrotor, which differs largely from other four-rotor vehicles with its unconventional airframe design and relatively high weight. Besides control engineering aspects, they study the effects of blade flapping and introduce a sprung-teetering rotor hub in combination with aero-elastic rotor blades. Flight experiments are not conducted at this project stage.

2005. Altuğ *et al.* [17] introduce a dual-camera pose estimation method combined with nonlinear control techniques aimed to enable autonomous flight of an HMX-4 quadrotor. They incorporate feedback linearization and backstepping controllers for stabilization and tracking control. Simulation results are discussed, results referring to the tethered-flight experiments are not available.

Castillo *et al.* [18] present a vision-based control system that implements the Nested Saturation Control Strategy to achieve autonomous hover with a Draganflyer III. The ground station, which is connected to a Polhemus 3D motion capture system that provides the craft's position and attitude, computes the control signals and sends them to the craft through a radio link. Despite reducing most of the oscillations in the system, the obtained performance does not suffice to accomplish autonomous hover.

Hanford *et al.* [19] present a custom-built, spruce-based quadrotor that comprises of solely inexpensive off-the-shelf components. The onboard electronics is developed with the intention to provide enhanced stability for manually controlled flights. Initial testing is performed on a Whitman training stand, but the insufficient integrity of the wooden airframe prevents successful flights.

Waslander *et al.* [20] propose Integral Sliding Mode and Reinforcement Learning as two control methods that lead to improved performance over classical control techniques. They show that their STARMAC I quadrotor is able to maintain autonomous hover for a duration of about 2 min within a circle of 3 m by employing a Trimble Lassen LP differential Global Positioning System (GPS) receiver.

2006. Hoffmann *et al.* [21] present their custom-built STARMAC II craft and introduce a distributed control algorithm for computing optimal control inputs for a multi-vehicle, coordinated localization of a stationary target. The algorithm is based on information theoretic methods with particle filters and, for safety, incorporates collision avoidance and control authority constraints. They provide results of simulating a bearings-only sensor model with multiple vehicles. Initial test flights with their STARMAC II quadrotor show the feasibility of this algorithm in real world situations.

Pounds *et al.* [22] introduce a PID-based control algorithm for stabilizing the attitude of their Mark II quadrotor and conduct tethered flights on a test bench. They observe that the system shows a chaotic behavior and becomes highly instable as the rotor speed increases, which renders untethered flights impossible at this point.

Tayebi and McGilvray [23] propose a quaternion-based feedback control scheme for exponential attitude stabilization, which comprises of a compensator for the Coriolis and gyroscopic torques. They perform tests with a Draganflyer III tethered to a ball joint base and find that compensating for the Coriolis and gyroscopic torques does not make a significant difference concerning their particular case.

Tournier *et al.* [24] present a pose estimation scheme based on monocular vision and Moiré patterns, which is implemented on a Draganflyer V-Ti Pro equipped with a camera that is mounted with the target in the field of view. The relevant attitude and position information describing the craft's pose relative to the target is extracted from the images by utilizing four single-point discrete Fourier transformations on the Moiré patterns. They demonstrate autonomous hover maintained for a period of around 9 s, results from longer flight durations are not available.

2007. Adigbli *et al.* [25] compare the stability and performance resulting from feedback, backstepping, and sliding-mode control based on simulating autonomous navigation. They decide to implement the backstepping approach on their custom-built, low-cost quadrotor by utilizing open-source software. Results are presented for the simulations and conducted experiments with the craft mounted on a tripod.

Bouabdallah *et al.* [26] propose an obstacle avoidance controller based on ultrasonic ranging, intended to be implemented on their OS4 quadrotor. In particular, four range finders, facing North, South, West, and East, are used for obstacle recognition and a fifth one facing downwards is exploited for altitude control. They provide results obtained from simulating this approach, but are not able to undertake free-flights because of the insufficient control performance achieved.

Fowers *et al.* [27] present a vision-aided drift stabilization scheme that deploys an onboard Field-Programmable Gate Array (FPGA) for performing the computationally intensive vision processing tasks of their custom-built HelioCopter micro-quadrotor. The drift stabilization algorithm, which executes in real time on the FPGA, makes use of Harris feature detection and template matching methods. They demonstrate the detection of special features in the sampled images, however, performance results that allow for reasoning about the achieved drift compensation are not available.

Hably and Marchand [28] propose a global asymptotic stabilization control law with bounded inputs developed for a Draganflyer III, aimed to provide a simple and more suitable solution for small embedded devices. The control scheme is based on saturation functions and utilizes the global stabilization of multiple integrators with bounded inputs. Because the implementation on the real platform is still in progress, only simulation results are provided at this project stage.

Haukrogh *et al.* [29] report on their approach of enabling autonomous navigation of a Dragenflyer X-Pro augmented with additional sensors for providing inertial and positional information as well as an onboard computer system. They design a Steady-State Kalman Filter (SSKF) and an Unscented Kalman Filter (UKF), which are tested in simulations with data obtained from the sensor instrumentation during a flight controlled with the

original system. They find that the UKF yields less promising performance results and conclude with the decision to implement the SSKF on the craft.

Hoffmann *et al.* [30] investigate three different aerodynamic effects that influence the quadrotor dynamics, specifically, vehicular velocity, angle of attack, and airframe design, which cause moments that affect attitude and altitude control. Based on the knowledge gained from these studies, an improved controller design is presented. They demonstrate autonomous hover maintained for about 45 s within a circle of 0.8 m, performed indoors with their STARMAC II quadrotor by utilizing blob tracking software in conjunction with an overhead-mounted camera.

Roberts *et al.* [31] present their custom-built, Printed Circuit Board (PCB)-based quadrotor that utilizes four infrared distance sensors, facing North, South, West, and East, for position control, in addition to a downward-facing ultrasonic range finder exploited for altitude control. They demonstrate that their craft is capable of maintaining autonomous hover within a circle of approximately 2 m.

2008. DeNardi and Holland [32] propose the development of position controllers based on minimal physical assumptions and the use of evolutionary programming in an efficient co-evolutionary framework. The system parameters are determined via sampling real data with a Hummingbird quadrotor that carries five infrared tracking markers. These markers are deployed in combination with a Vicon MX 3D motion capture system that provides updates of the craft's attitude and position with a sampling frequency of 100 Hz and an accuracy in the sub-millimeter range. The general feasibility of developing a control algorithm with genetic programming methods is demonstrated, practical results of flight experiments are not available at this project stage.

Grzonka *et al.* [33] present the setup and algorithms used for estimating the position of a Mikrokopter equipped with a laser range scanner for localizing the craft's position within a closed environment. The range scanner is also utilized for altitude control by deflecting several laser beams towards the ground with a mirror. Based on Monte-Carlo Localization, the craft's position in a known map of the environment is estimated with a particle filter. They demonstrate the general feasibility of their approach by testing the localization algorithm during manually piloted flights. In a subsequent paper [34], the proposed localization scheme is explained in more detail and further experiments including altitude and yaw control are discussed. They provide a link to a video-recorded flight that demonstrates autonomous navigation in a known environment based on a map created a priori by a mobile ground robot.

He *et al.* [35] describe a trajectory planning algorithm for estimating the position of a Hummingbird quadrotor. Similar to Grzonka *et al.* (2008), the craft is equipped with a laser range scanner for localizing the craft's position within a known environment. The proposed planning scheme is based on a generalization of the Belief Road Map algorithm, which represents an extension of the Probabilistic Road Map algorithm, such that a UKF can be applied in combination with Sensor Uncertainty Sampling methods. The path-tracking accuracy is determined in the experiments by comparing the position estimates

of the algorithm against the position information provided by a stationary motion capture system. Based on an existing map, they demonstrate autonomous navigation in a known environment with a positional accuracy of 0.17 m.

Hoffmann *et al.* [36] present a trajectory tracking algorithm to follow a desired path and an algorithm for the generation of dynamically feasible trajectories, where the tracking algorithm separates the plan generation for obstacle avoidance from the computation of travel speeds and control inputs. They demonstrate that their STARMAC II quadrotor is able to track a path indoors with an improved accuracy of 0.1 m by utilizing the vision-based system described in Hoffmann *et al.* (2007) and outdoors with an accuracy of 0.5 m by employing a NovAtel Superstar II carrier-phase differential GPS receiver.

Hürzeler *et al.* [37] present the control scheme and design details of a Hummingbird quadrotor equipped with a camera and a laser range scanner exploited for position control. Based on the fusion of visual and dimensional information provided by the camera and, respectively, the range scanner, they demonstrate autonomous hover with an average positional accuracy of 0.3 m.

2009. Bachrach *et al.* [38] present a position control scheme for autonomously navigating a quadrotor in unknown indoor environments. Similar to He *et al.* (2008), they employ a Hummingbird quadrotor equipped with a laser range scanner for localizing the craft's position. However, their approach allows for exploring and mapping an unstructured and unknown indoor environment without the need of an a priori created map. The proposed control scheme is based on a multilevel sensing and control hierarchy, a fast laser scan-matching algorithm, an Extended Kalman Filter (EKF) for data fusion, an advanced Simultaneous Localization And Mapping (SLAM) implementation, and an exploration planner. The algorithm's ability to enable autonomous operation in a variety of unknown indoor environments is demonstrated through several experiments. They also provide a link to video-recorded flights discussed in the paper.

Goel *et al.* [39] present the modelling, simulation, and control design aimed to enable autonomous navigation of an X-UFO quadrotor. The proposed PID-based control scheme consists of an inner and outer loop comprising of solely attitude, respectively position controllers, which, in this configuration, represents a classical control approach without state estimation. The craft carries a custom-built autopilot that provides inertial and GPS-based positional information. Autonomous position control is simulated, but not demonstrated in the free-flight experiments.

Grzonka *et al.* [40] extend their recent work on a localization scheme for autonomous indoor navigation (2008) with a SLAM implementation based on an online variant of a stochastic gradient optimization algorithm, heading to eliminate the need of an a priori created map. They demonstrate the general feasibility of their approach by creating a map of the office environment in real time during a manually piloted flight.

Zhang *et al.* [41] introduce a vision-based position estimation scheme that enables a Hummingbird quadrotor to track the movements of a wheeled mini-robot. They utilize an external PC for image processing and EKF-based data fusion of the measurements

received from the craft. The ground robot is equipped with two illuminated markers that are tracked by the craft's onboard camera. Because there are two markers, the hovering craft may also follow the robot's yaw movements. The true tracking error is determined by comparing the position estimates with the positional information provided by a stationary Visualeyez VZ-4000 3D motion sensing system. Despite hovering with oscillations, the quadrotor is able to maintain autonomous hover above the non-moving ground robot within a circle of around 0.5 m and a yaw deviation of ±5 deg. They further demonstrate the craft's ability to track the moving and rotating ground robot with a maximum position error of 0.4 m and a maximum yaw error of 25 deg.

1.4 Main Contributions

In the effort to achieve a result that is competitive to existing solutions, the aerial platform as well as the entire software system was developed completely from scratch with the focus placed on new ideas apart from commonly applied principles. As a consequence, the contributions presented in the scope of this thesis fall into three different engineering disciplines and are therefore separated into three leading topics, particularly, platform development, control system design, and software architecture.

Platform Development. Referring to the related research addressed in the previous section, it points out that the majority of academic projects is concerned with pure control engineering aspects. Many groups make use of commercially available RC quadrotors or build their own low-cost vehicles. Except for a few outstanding designs, most airframes represent rather filigree constructions that do not offer much integrity and the propulsion systems rarely offer surpassing payload capabilities. In other words, there is commonly little attention payed to the design of the underlying aerial platform, even though it plays the most important role with regard to accomplishing the aero-physical tasks. In order to provide a more sophisticated solution with the JAviator, first, a robust and lightweight airframe was constructed, and second, an efficient and powerful propulsion system was developed, with the focus in both cases on reproducibility.

The first and probably most challenging task when creating a quadrotor vehicle is, similar to any type of aircraft, to construct an airframe that is as light as possible and, at the same time, as robust as possible. The second and not minor challenging task is to design a reliable propulsion system that captivates with an excellent thrust-to-power ratio, especially if based on transforming electric power and the flight endurance is a major issue. All of the quadrotors mentioned in the previous section have in common that the rotors are mounted in the conventional classical way. In particular, the rotor head is mounted on the top end of some shaft, which can be either the motor axle itself or a separate rotor shaft. Due to this, the rotors can be subject to undesired twisting, especially if the design incorporates large rotors or propellers and the frame construction does not provide sufficient integrity.

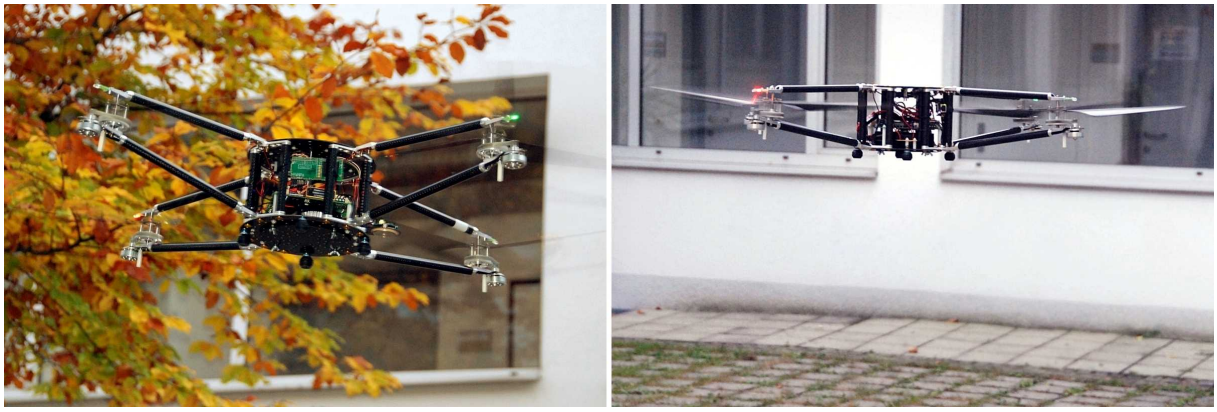


Figure 1.3: Outdoor flights performed in the courtyard of the Department of Computer Sciences at the University of Salzburg, showing the JAviator flying at a height of about 3 m (left) and hovering at a height of about 1.5 m (right).

A new approach that differs largely from existing designs was realized with the JAviator airframe. It consists of an identical top and bottom frame structure connected at the inside via vertical braces forming a cage-like fuselage and at the outside via vertical rods serving as the rotor shafts. The resulting airframe, which can be imagined as a horizontal bicycle wheel consisting of a central hub with four "top" and four "bottom" spokes, provides very high mechanical integrity at relatively low weight. The hub of this construction, or, in aviation jargon, the fuselage, acts like a protection frame for the onboard electronics, hence offering extremely high robustness in case of collisions. The vertical rods at the far ends are screwed to the top and bottom booms with the rotors spinning around these *fixed* shafts. This construction has two significant advantages over conventional designs: first, besides preventing undesired rotor twisting, the rotors can be adjusted very precisely regarding plane parallelism, and second, a two-end-mounted rotor shaft allows for much larger rotors and more complex rotor heads than possible with a one-end-mounted rotor shaft. Figure 1.3 shows the JAviator during outdoor flights from two different perspectives that convey the bigger picture of the unique airframe design.

The second problem concerning the implementation of a surpassing propulsion system was also solved quite differently compared to state-of-the-art designs. In recent years, brushless Alternate Current (AC) motors[1] have started widely to displace brushed Direct Current (DC) motors, especially in the domain of model sport. The reason for this emergence is that brushless motors offer a higher power-to-weight ratio, allow for higher rotation speeds, and also provide a higher moment of torque. Referring to quadrotor

[1]Brushless motors are sometimes incorrectly denoted as *brushless DC motors*. Particularly, brushless motors employed in model sport are almost always 3-phase AC permanent-magnet synchronous motors, and therefore, belong to the category of so-called *3-phase AC synchronous machines*. In cases where no 3-phase AC supply is available, as is usually the case in model sport, a so-called *brushless-motor controller* is used to generate the required 120-deg-phase-shifted 3-phase AC signal, which is often called *electronic commutation*. Despite the fact that the brushless-motor controller is supplied with DC, the connected motor is driven by electronically commutated 3-phase AC, and thus, no DC motor.

aircraft, most models that can be found nowadays make use of these advantages and incorporate small rotors or propellers that are mounted directly on the axles of brushless motors. Accordingly, as a side effect, gearing boxes are not required as long as the motors are sufficiently powerful to drive the rotors without gear reduction. However, the downside of this method is that the rotors are fairly restricted in their size and larger, more complex rotor heads can only be employed to some extent.

To overcome this burden, the propulsion system of the JAviator is constructed in a way that combines the advantages of the symmetrical airframe design with the advantages of the more powerful brushless motors. As mentioned previously, the two-end-mounted rotor shafts of the present airframe design allow for much larger and heavier rotors. Hence to achieve a high payload capacity, the JAviator is equipped with rather huge rotors that are driven by extremely strong, custom-built brushless motors. They are quite small in their dimensions and contain a winding of very low impedance, altogether resulting in an excellent power-to-weight ratio. The transmission between the motors and the rotors is performed via industry-standard cogged-belt drives, which are more robust and far less noisy than toothed-gear transmissions. So far, neither the JAviator's unconventional but effective airframe design could be found in the present or some related form, nor the noteworthy thrust-to-power ratio accomplished with its propulsion system could be found being exceeded by any other quadrotor.

Control System Design. Two difficulties pointed out right from the beginning that it would become a very challenging task to achieve a control performance competitive to existing solutions: first, the large three-bladed rotors of the JAviator, and second, the relatively low accuracy of the available Radio Frequency Identification (RFID)-based localization system. Referring to the large rotors, micro-quadrotors, which are almost always equipped with propellers, largely profit from the rather high rotational momentum of their propellers. They usually run at rotation speeds that are an order of magnitude higher compared to paddle-shaped rotors, and consequently, incur a significant gyroscopic effect that causes increased stability in a natural way. We observed this circumstance while experimenting with different blade sizes, but nevertheless decided for the large rotors in favor of extended payload capacities. Concerning the employed localization system, it is apparent from the previous section that the predominant method of providing positional feedback for autonomous indoor navigation is to use vision-based sensing. Methods that belong to this category not only enable accuracies in the millimeter range, they also mostly offer high update rates. However, vision-based localization environments, like the Vicon MX 3D motion capture system, come at extraordinary expense and can therefore hardly be afforded by research groups that must get along with a low budget.

To cope with these problems, a control system was developed that represents a state space approach in conjunction with advanced PID-based controllers. More precisely, the problem of the diminished gyroscopic effect was solved by increasing the bandwidth of the attitude controllers with enhanced feedback. Besides providing these controllers with acceleration feedback in addition to velocity estimates, they also incorporate computed

velocities that refer to changes in the reference commands. Due to this extension, the control response of the JAviator could be improved by an order of magnitude, resulting in precise controllability that allows for extreme flight maneuvers even in limited space. The problem with the low localization accuracy was solved by designing an EKF-based state estimator that fuses positional information with inertial measurements for generating more accurate position and velocity estimates.

Compared to high-precision motion capture systems, RFID-based localization can be seen as a low-cost alternative that we applied in the scope of a concurrent research project concerned with autonomously navigating a small co-axial RC helicopter [42]. Albeit not the optimal choice in terms of position control, we decided to reutilize this localization system for the JAviator project due to financial restrictions. Despite the rather imprecise position feedback, reasonably-well positional controllability could be achieved. Moreover, the autonomous-hover accuracy demonstrated by the JAviator has proven to be optimal in relation to the accuracy offered by the sensing system. To the best knowledge of the author, this was also the very first time that an RFID-based localization system was involved to autonomously navigate a quadrotor aircraft.

Software Architecture. The control software of a quadrotor usually executes on some embedded device with limited computational resources. Such devices not uncommonly comprise of pure integer-based processors, which demands to deal with the burden of fixed-point programming. Control software is mostly implemented in form of event-triggered mechanisms and typically tuned in accordance with the comprised sensor and actuator equipment. As a consequence, an implementation of this kind is by default susceptible to modifications applied to the execution environment, which might cause the whole system to behave completely differently, demanding sometimes cumbersome adapting or even re-engineering the entire system architecture.

Surprisingly, especially real-time control systems that must adhere to strict deadlines are still almost always implemented by applying traditional concepts, and consequently, rarely offer sufficient flexibility with respect to changes of the hardware and/or software platform. In analogy to the paradigm "write once − run anywhere" associated with the programming language Java, the necessity to extend this notion to the temporal domain has become a major issue in recent years. However, this fact seems to be still widely ignored by the control engineering community. Despite the reasons responsible for this circumstance, it can be assumed that control systems not providing time portability will probably have no relevant chance to market in the future.

The approach realized in the JAviator control system differs from the classical concepts in many ways. Most importantly, it is a fully time-triggered system that is driven by a software-based high-resolution timer. This approach not only enables precise timing of the control cycle, it also allows for decoupling all tasks related to sensing and actuating from the computational control aspects. In this sense, it represents a distributed control system encompassing several lower-level and higher-level devices that are controlled by the same common timing domain. The lower-level devices are responsible for data acquisition

and actuator signaling, whereas the higher-level devices are responsible for computing the control signals and handling wireless communication. Accordingly, the time-critical tasks related to sensing, actuating, and computing are performed by different devices and the essential information is communicated among them via designated channels. These channels adhere to the same communication policy, which is implemented in form of a simple packet-oriented, byte-based protocol. All lower-level and higher-level devices are therefore able to exchange data with each other arbitrarily.

The modular architecture of the JAviator control system allows to experiment with different module implementations without the need to recompile any of the other modules, even if using a different programming language for each module. Due to facilitating time portability, the temporal behavior of the control system is preserved with very low jitter in the microsecond range across different hardware and software platforms.

Summary. The contributions discussed can be summarized as follows:

- A high-precision quadrotor that stands out from others with its unique, symmetrical airframe design, enormous robustness, and surpassing payload capacity.
- A flight control system that achieves a remarkable response behavior and a high degree of autonomous-hover accuracy with RFID-based positional feedback.
- A fully time-portable software architecture that preserves the temporal behavior of the control algorithm in regard to changes of the execution environment.

1.5 Thesis Outline

Chapter 1 – Introduction. The first chapter is aimed to make the reader familiar with the term "quadrotor", shade some light on the complex dynamics of such aircraft, discuss related research associated with quadrotor UAVs, convey the main contributions, and outline the topics covered in this thesis.

Chapter 2 – Quadrotor History. This chapter is aimed to give a brief overview about the most remarkable and successful quadrotor aircraft that appeared in the last century. It is shown that early quadrotor-based designs have played an important role in the development of today's modern helicopters. The chapter further introduces some of the first commercially available RC model quadrotors that paved the way for many research projects involving quadrotor UAVs.

Chapter 3 – Platform Development. In this first major chapter, the JAviator platform and its design is described in much detail. It starts with an explanation of the airframe construction realized in both the prototype JAviator V1 and its successor the JAviator V2, followed by a description of the employed propulsion system. The chapter

continues with an explanation of the avionics equipment comprising of the power supply, the various sensors involved, and the underlying computer system.

Chapter 4 – Control System Design. In this second major chapter, the process that led to the JAviator control system is described. After imparting some preliminary knowledge, the applied control principle and resulting system model is introduced. This model is separated into four specific system components that are examined consecutively as follows: (1), the quadrotor plant providing the sensor measurements, (2), the digital filters improving the data quality, (3), the state estimator generating the state estimates, and (4), the motion control computing the motor signals.

Chapter 5 – Software Architecture. In this third major chapter, the JAviator software system and its architecture is introduced. Beginning with a discussion of the essential differences between the present approach and the most commonly chosen design, the chapter continues with a thorough description of the software architecture divided into three fundamental layers: (1), the plant layer performing all sensing and actuating tasks, (2), the flight control layer providing the motor signals to the plant, and (3), the ground control layer providing the reference commands to the flight control.

Chapter 6 – System Performance. This chapter describes the experiments that have been conducted to prove the presented solutions. Particularly, the JAviator platform is evaluated from three distinct perspectives: (1), its capability with respect to lifting capacity and flight endurance, (2), its controllability with respect to manually piloted and position-controlled flights, and (3), its control system's time portability with respect to different execution environments.

Chapter 7 – Conclusion. The last chapter reviews the topics addressed in the thesis and summarizes the main contributions in regard to platform development, control system design, and software architecture. It concludes the present work with a brief outline of possible future directions.

Appendix A – JAviator V1 Drawings. In the first appendix chapter, all of the drawings used to build the prototype version are presented.

Appendix B – JAviator V2 Drawings. In the second appendix chapter, all of the drawings referring to the advanced version are presented.

Remark 2. The complete set of drawings included in the Appendix is also available online and can be downloaded from the official JAviator project web site [2].

Chapter 2

Quadrotor History

But the fact that some geniuses were laughed at does not imply that all who are laughed at are geniuses. They laughed at Columbus, they laughed at Fulton, they laughed at the Wright Brothers. But they also laughed at Bozo the Clown.

Carl Sagan (1934−1996)

The dream of flying exists probably as long as the human being. The imagination to climb vertically to a certain height, hover for some time, fly in a specific direction for a certain distance, come back into hover, and safely return to the ground − what is expected, per definition, from any modern helicopter − was perhaps there long before any ambitions concerned with the development of aircraft. Mainly due to a gap in understanding the complex aerodynamics of rotorcraft and the lack of sufficiently powerful engines at the time, it was many decades later − after already accomplished with airplanes − until the first fully controlled, reliable, and comfortable free flights were achieved with helicopters. The very earliest known ambitions of vertical flight can be back-annotated to 400 BC, represented by the Chinese Top, basically a shaft equipped with feathers as rotor blades, a toy still very popular today (see [5]). Many centuries later, at the end of 1700, the first experiments and successful flights with small-scale models very conducted. It was to take more than another century, until the beginning of 1900, before the first full-scale human-carrying helicopters appeared. At that time, the attempts to vertically lift piloted vehicles were primarily influenced by the enthusiasm and inspiration of their inventors. In other words, to scientifically approach the encountered problems, the necessary knowledge of helicopter aerodynamics and accompanied aeromechanics has not been given, and is still not entirely explored and understood today.

As one would probably not expect, much of pioneer work in the effort to overcome the technical barriers of vertical-rising locomotion was performed with quadrotor-based helicopter designs. Two likely reasons for this circumstance might have been the missing technology, first, to produce a large enough single rotor capable of meeting the thrust and integrity requirements, and second, to efficiently generate a counteracting torque for compensating the torque induced by a single main rotor. However, among a vast amount

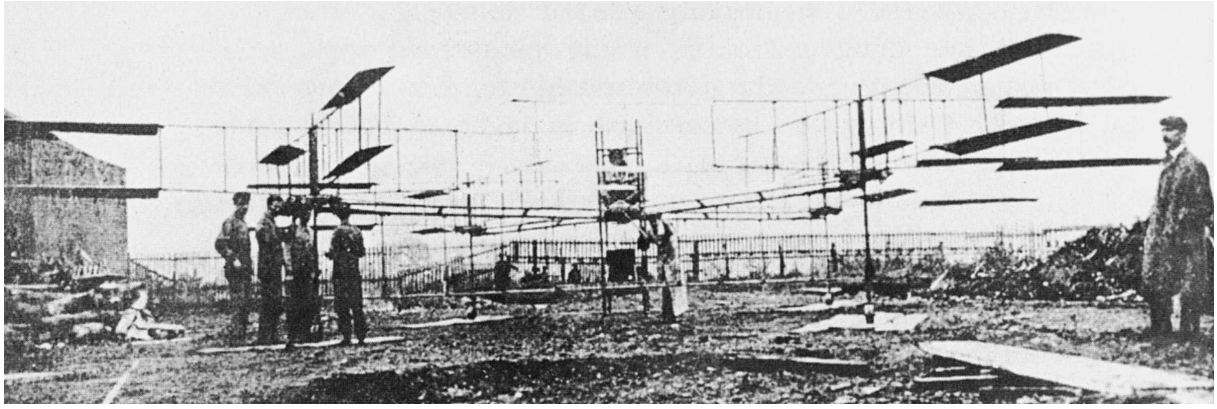


Figure 2.1: The Bréguet-Richet Gyroplane No. 1 helicopter of 1907 at Douai in France. Taken from Gablehouse [43].

of different helicopter designs, some more, some less efficient, quadrotor-based approaches have played an important role to pave the way for the final success of rotorcraft.

2.1 Bréguet-Richet Gyroplane No. 1

In 1907, the brothers Louis and Jacques Bréguet, in association with Professor Charles Richet, three enthusiastic Frenchmen with a strong interest in vertical flight, built the Bréguet-Richet Gyroplane No. 1 helicopter. It consisted of a tubular steel frame with the shape of cross-like arranged ladders. A fabric-covered four-bladed biplane rotor was mounted at each of the far ends, giving the craft a total of thirty-two lifting surfaces. One pair of diagonally opposite rotors was spinning clockwise, the other pair spinning counter-clockwise. The four rotors were driven via belts by an Antoinette engine of approximately 45 hp, installed in the center of the airframe right above the pilot's seat. The craft had a diameter of roughly 8.1 m and an empty weight of around 510 kg. Because the Bréguets' initial intention was to demonstrate vertical rising of a piloted vehicle under its own power, the aircraft neither had a means for stability nor directional control, merely a throttle for regulating the engine's rotation speed.

Historical notes about the first successful lift-off performed at Douai in France differ, denoting August 24, September 19, as well as September 29, 1907. Data about the reached height also vary from 0.6 to 1.5 m, whereas the hover endurance is reported to last for about 1 min. It is further handed down that the Bréguets were able to find a willing pilot for their craft, a Monsieur Volumard, who was chosen essentially on account of his modest weight of 68 kg. However, due to the lack of any means for control, a supporting person was required at each of the four girder ends. Whether these people contributed to the lift-off in some facilitating way has been disputed largely at that time and is still discussed today (see [44]). Except for some more tests not exceeding a height of 1.5 m, no further development was applied to that aircraft in favor of their second helicopter Gyroplane No. 2, which was equipped with two forward-tilted rotors. Nevertheless, the

Figure 2.2: The de-Bothezat-Jerome helicopter of 1922 descending after a test flight on February 21, 1923 at McCook Field near Dayton, Ohio. Taken from Edison [50].

Bréguet-Richet Gyroplane No. 1 quadrotor was the first helicopter to successfully lift itself and a pilot under its own power.

Present description and data are based on Apostolo [45], Gablehouse [43], King [46], Leishman [5], Munson [47], and Smith *et al.* [48].

2.2 De-Bothezat-Jerome Helicopter

More than a decade later, at the beginning of 1920, the French-named Russian professor Dr. George de Bothezat was invited by the Army Air Service (AAS), which was later to become the US Air Force, to work with them. Professor de Bothezat, who was already well-known for his theories about vertical flight, received a contract to build a full-scale helicopter within seven months, including all preliminary design and final construction work. As a consequence of this challenge, de Bothezat emigrated to the United States and, together with his co-designer Ivan Jerome, developed the de-Bothezat-Jerome helicopter, which was a quadrotor. It consisted of four large girders constructed of intersecting steel tubes, each carrying a windmill-shaped six-bladed rotor. The girders together with the rotors were slightly inclined inwards at an angle of approximately 3 degrees, intended to increase lateral stability. The conical shape of the rotor blades, with a surface small at the root and extended outwards, seems to be closely related to his work about Blade Screws (see [49]). However, the rotors were designed to support collective pitch (all blades are applied the same angle simultaneously to increase the lift) and could also be regulated individually (differential collective pitch between the rotors) to enable directional control. To further enhance control, four additional small rotors were installed. The pilot's seat was placed directly behind a 180-hp Le Rhône rotary engine, which served as the craft's central source of power, later replaced by a 220-hp British Bentley rotary. The total diameter was over 18 m and its empty weight around 1600 kg.

After receiving an extension of the deadline, the AAS' first helicopter was completed in the fall of 1922. When Professor de Bothezat and his assistances prepared the craft for its

first test flight at McCook Field (later to become the Wright-Patterson Air Force Base) near Dayton, Ohio, on December 18 of that year, the curious crowd soon began to laugh at the weird looking assemblage of tubes and wires. Immediately the name "The Flying Octopus" was unluckily gained, a denotation that can often be found in conjunction with their helicopter. Notes about the pilot of the first test flight differ, some state it was de Bothezat himself at the controls, some state it was Thurman H. Bane (by then Colonel), who is known today as the US Army's first helicopter pilot. Nevertheless, the craft rose to a height of roughly 1.8 m and hovered there for 1 min and 42 s, while drifting away for about 150 m. More than one hundred test flights were conducted in the following year with both improved endurance and payload capacity, reaching heights of up to 6 m. It set an endurance record of 2 min and 45 s in February 1923 and carried four additional men two months later. However, it turned out that the craft was far less maneuverable than expected and directional changes were mainly a result of the blowing wind. Moreover, technical modifications demanded by the AAS could not improve the overall performance. Due to this and the AAS' increasing interest in autogyros, the project was soon abandoned and de Bothezat was obliged to discontinue his work. Although the specifications of the AAS could not be met, the de-Bothezat-Jerome quadrotor was the first helicopter that succeeded in performing several, at least marginally, controlled free flights.

Present description and data are based on Francis [51], Gablehouse [43], Gerhardt [52], Glines [53], Leishman [5], Spooner [54], and Teale [55].

2.3 Œhmichen's No. 2 Helicopter

At about the same time, the French designer Étienne Œhmichen, a contemporary of de Bothezat, was to write helicopter history. Œhmichen, who invented a total of eight different vertical take-off machines, developed his first twin-rotor No. 1 helicopter in 1920. It was equipped with a 25-hp engine, which was too weak for lifting the craft. To overcome the lack of power, he simply added a balloon filled with hydrogen, which was connected to the craft's top frame. It is handed down − certainly due to the ingenious but ungainly looking aircraft − that Œhmichen was asked by brash spectators "if he was sure that he had invented a helicopter" (see [43]). Nevertheless, in 1921, he started to build his most remarkable and competitive No. 2 helicopter, which, similar to de Bothezat's helicopter, was a quadrotor. The airframe was constructed from steel tubes in a cross-like style with a paddle-shaped two-bladed rotor at each of the four ends. The rotor blades could be warped to change their "bite", hence providing collective pitch. Vertical thrust was regulated by applying collective pitch to all four rotors simultaneously, whereas to attain directional control and propulsion, eight additional small rotors were integrated in the design. All of the twelve rotors were driven by a single 120-hp Le Rhône rotary installed in the center of the airframe. The engine, which was later replaced by a 180-hp Gnôme rotary, was coupled with a large flywheel aimed to increase stability. The total diameter of the craft was approximately 16 m and its empty weight around 800 kg.

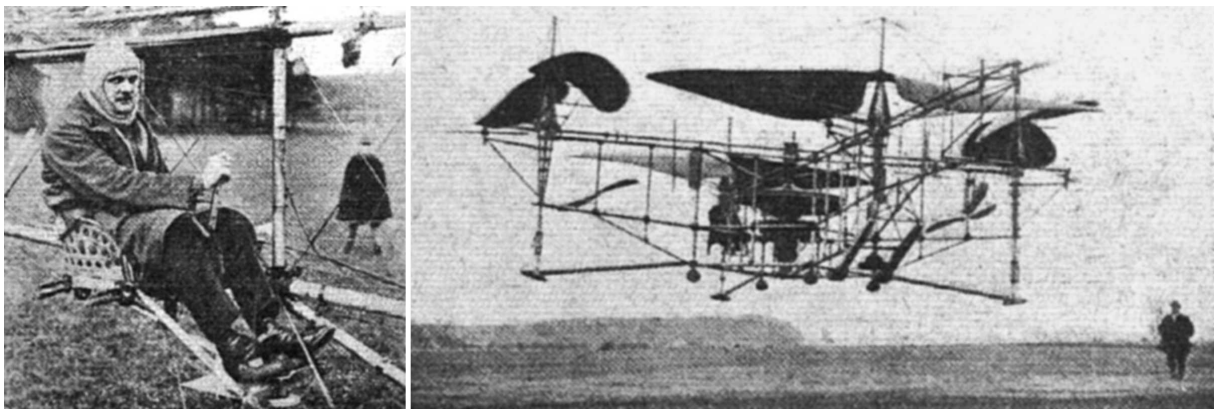


Figure 2.3: Étienne Œhmichen at the controls of his No. 2 helicopter of 1922 (left) and airborne at Les Breuils in Verdun, France (right). Taken from Spooner [56].

The aircraft flew successfully for the very first time on November 11, 1922, already one month before the first flight of the de-Bothezat-Jerome helicopter. Compared to many other competing vertical take-off machines at that time, Œhmichen's No. 2 helicopter achieved a noteworthy degree of stability and controllability. It performed more than one thousand test flights in the following years and on April 14, 1924, with a flown distance of 360 m, established the first-ever distance record for helicopters recognized by the Fédération Aéronautique Internationale (FAI). Less than a month later, on May 4, 1924, at Les Breuils in Verdun, France, the craft was airborne for 14 min and flew more than 1.6 km, while carrying a payload of 200 kg. In the scope of that flight, it rose to a height of 15 m and completed the first 1-km closed-circuit flight in 7 min and 40 s, honored with a large cash award received from the French Air Ministry. Albeit successful, Œhmichen was not satisfied with the rather low climbing capability of his No. 2 machine and embarked working on single-main-rotor designs. However, despite that his craft was considered sheer impractical for any meaningful usage, Œhmichen's quadrotor-based No. 2 helicopter proved to be exceedingly stable and maneuverable, and it represents a milestone in the history of helicopter flight.

Present description and data are based on Gablehouse [43], Gerhardt [52], Harris [57], Leishman [5], Munson [47], Spooner [56], and Teale [55].

2.4 Convertawings, Model A

In 1956, by a time than various helicopter designs had already established their place in military, industrial, as well as commercial usage (for an overview of the great many helicopters already in use at that time see [45, 58]), the Convertawings Inc. came up with a completely different approach. Until then, the predominant helicopter design consisted of a single main rotor and a small tail rotor for compensating the main-rotor torque reaction. The advantage of a single main rotor is that directional changes can be achieved by applying a cyclical pitch to the blades. That is, in addition to collective pitch, the

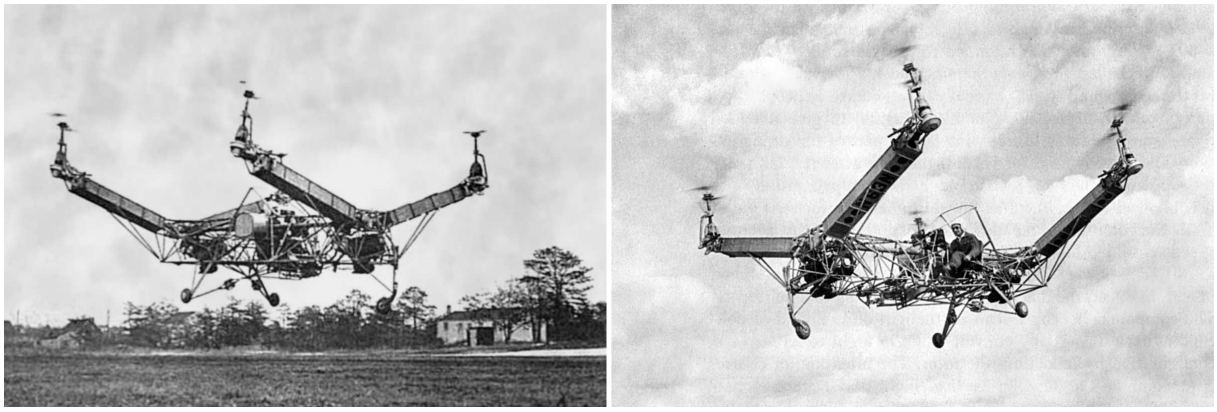


Figure 2.4: The Convertawings Model A quadrotor of 1956 during its very first lift-off (left) and airborne with its designer Kaplan at the controls (right) at Zahn's Airport in Amityville, New York. Taken from Stoff [59] (left) and Gablehouse [43] (right).

blades are pitched individually as they rotate around the rotor shaft, usually implemented by a means of swash-plate mechanism (for a detailed technical description see [5]). The disadvantage is the rather high complexity of such a mechanism that counts as the most critical part of the rotor head, and consequently, demanding frequent maintenance. The American helicopter designer David H. Kaplan, who worked with the Convertawings Inc., once described the problem of reducing the complexities in a rotor system with cyclic control as follows: "In a cyclic-controlled rotor, every time the designer tries to deny the blade a freedom, it demands compensation somewhere else in the rotor mechanism. The history of helicopter is filled with attempts to reduce complication ... invariably this turns into a game of Chinese checkers as the designer feverishly moves the complicated problem from one part to another, never getting rid of it." (see [43]).

In his endeavor to overcome the burden of cyclic pitch control, Kaplan designed the Convertawings Model A quadrotor, which comprised a simplified rotor mechanism with strap-mounted blades. Directional control and propulsion were accomplished solely via differential collective pitch between the four rotors and the potentiality to tilt each rotor individually. In contrast to the quadrotors described previously, where the rotors were positioned in cruciform, the Model A quadrotor introduced an "H" form with two pairs of rotors arranged side by side, in regard to helicopters, generally called *twin-tandem* arrangement. The craft's fuselage was shaped like a large triangular-trussed girder with the pilot's cab positioned on the left-hand side near the center. Two long booms were connected to the front and rear of the airframe, each carrying a two-bladed rotor. The booms were built of aluminum alloy and inclined inwards at an angle of approximately 25 degrees, with the rotors in upright position when not tilted at the whim of the pilot. The rotors were driven via multiple V-belts by two 90-hp Continental C-90 engines, one installed in the front of the fuselage, the other one in the rear. Additional interconnection between the shafting and transmission cases enabled either of the two engines to drive the whole rotor system, hence offering a considerable degree of safety. The craft's rotor

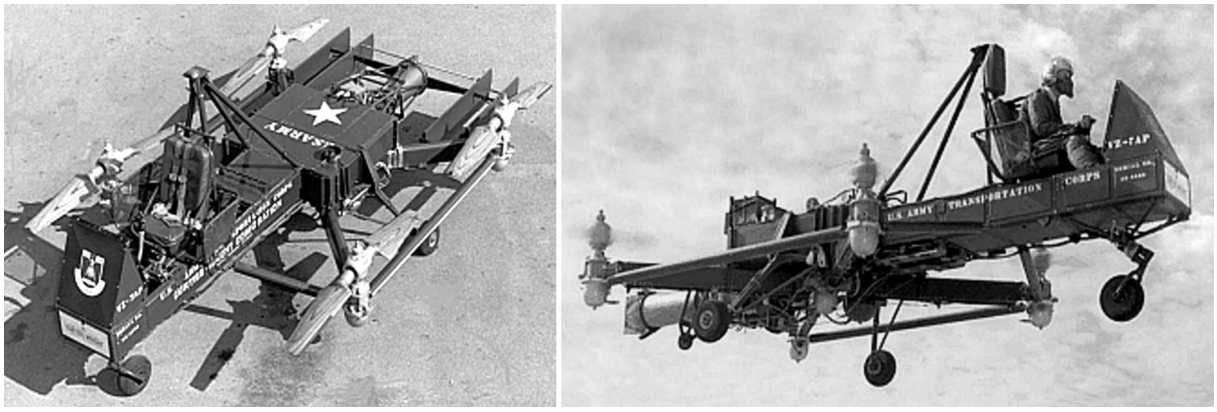


Figure 2.5: The Curtiss-Wright VZ-7AP "Flying Jeep" of 1958 as delivered to the US Army Transportation Corps (left) and during one of the numerous test flights conducted by the US Army between 1958 and 1960 (right). Taken from Hannant [64].

diameter was 5.91 m, the overall length 7.93 m, and the gross weight 998 kg.

The first free flight was performed on March 30, 1956, at Zahn's Airport in Amityville, New York, with D. H. Kaplan at the controls. The unique concept of differential thrust control combined with a versatile rotor inclination mechanism was realized in the Model A design with great success, resulting in a remarkably maneuverable helicopter. According to Convertawings, the Model A machine was just a "flying testbed" to prove their novel rotor concepts and a precursor for the Model E project suggesting a much larger quadrotor, which was announced to have a gross weight of around 19000 kg and a payload capacity of about 5000 kg. Unfortunately, due to cutbacks in defense spending, the US Army discontinued funding quadrotors and the Convertings Inc. gave up the Model E project in favor of a large transport convertiplane. However, the Convertawings Model A quadrotor was the World's first helicopter that exhibited perfect maneuverability about all axes under complete elimination of cyclic pitch control and rotor hinge mechanisms, and it was also the first quadrotor to demonstrate successful forward flight.

Present description and data are based on Allaway *et al.* [60, 61], Gablehouse [43], Smith *et al.* [62, 63], and Stoff [59].

2.5 Curtiss-Wright VZ-7AP

One of the last full-scale quadrotor aircraft that appeared in the last century was the Curtiss-Wright VZ-7AP, by then dubbed "Flying Jeep", a research aircraft developed in 1958 by the Santa Barbara Division (formerly the Aerophysics Development Corp.) of the Curtiss-Wright Corp. in Wood-Ridge, New Jersey. The development of the VZ-7AP happened under contract with the US Army Transportation Corps and was intended to fill the gap between the Army Jeep and a reconnaissance helicopter. In particular, the contract stipulated the development, initial testing, and shipment of two prototype VTOL utility vehicles that had to be light, robust, and easy to operate and maintain.

The prototypes, which the US Army received in mid-1958, were twin-tandem-style quadrotors and of extraordinarily simple design. The VZ-7AP essentially consisted of a central fuselage positioned over a rectangular airframe with a propeller mounted at each corner. The fuselage had a length of 5.2 m and carried the pilot's cab on top of the front end. The remaining surface behind the pilot's seat was intended to be used as utility space. The empty weight was 771 kg, the gross weight 952 kg. Although originally planned and equipped with ducted fans, both prototypes were delivered with unshrouded propellers, which were driven by a single 425-shp Turbomeca Artouste IIB turboshaft engine. The propellers had a diameter of 2 m and did not provide any means for changing the pitch setting. Accordingly, controllability over the craft was obtained solely by individually varying the speed of each propeller, and hence the generated thrust, similar in its effect to differential collective pitch. Advanced yaw control was attained with moveable vanes installed over the engine exhaust.

The VZ-7AP prototypes proved to be perfectly stable and maneuverable, unfortunately, they consistently failed in meeting the specification regarding service ceiling and forward speed. Consequently, the US Army returned the vehicles to the manufacturer after two years of testing in mid-1960 and did not issue any follow-up order. Despite the project's eventual termination, the Curtiss-Wright VZ-7AP quadrotor was the first helicopter that demonstrated the potentiality of a four-rotor control system based exclusively on varying the speed of the rotors. Today, this is the prevalent form of control exploited in electrically powered small-scale quadrotor vehicles.

Present description and data are based on Harding [65] and Smith *et al.* [66, 67].

2.6 Naito's Yuri I Helicopter

The last full-scale quadrotor that became known before the Millennium was the Yuri I Human-Powered Helicopter (HPH) designed by Dr. Akira Naito, a former professor at the Nihon University in Kamakura, Japan.[1] Professor Naito retired in 1991, but continued working on HPH designs, and between 1992 and 1993, in collaboration with the Nihon Aero Student Group (NASG) from the College of Science and Technology at the Nihon University, developed the World's first human-powered quadrotor. Yuri I was inspired by the Sikorsky HPH Competition, whose primary specifications − hovering for at least 1 min at an altitude of at least 3 m while drifting away for not more than 10 m − have by far not been reached since its announcement by the American Helicopter Society in 1980 (for a detailed description of the competition see [68]).

After inventing four different HPH craft, all not been able to lift off and hover for a while, Professor Naito conducted various experiments with small-scale models of specific HPH designs to investigate the aerodynamics of rotors quite close to the ground. He discovered that, when hovering in ground effect, HPHs with four rotors achieve the highest thrust-to-power ratio. This finding led to the fruitful design of an elaborate four-rotor

[1] Yuri is a Japanese name meaning Lily.

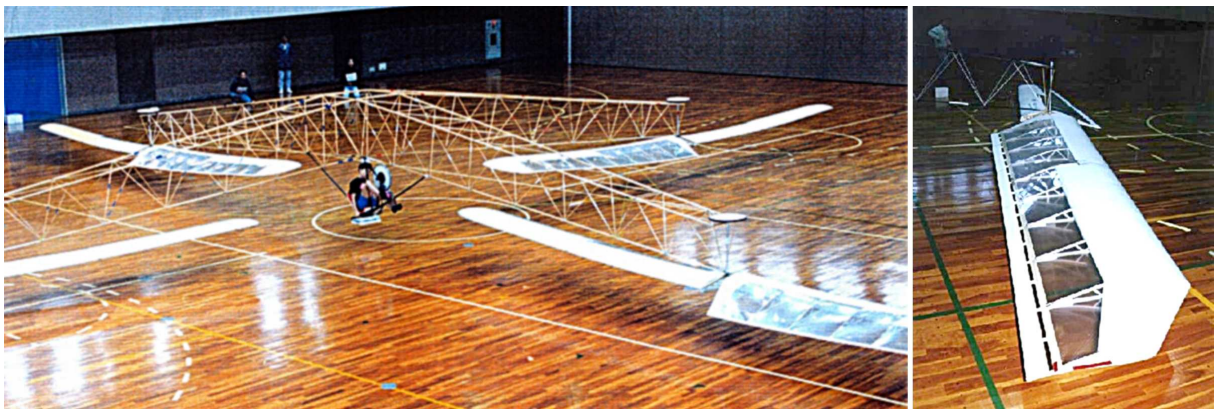


Figure 2.6: Naito's Yuri I human-powered helicopter of 1993 during its world-record flight on March 7, 1994 at the Nihon University in Kamakura, Japan (left) and one of its transparent rotors revealing the airfoil construction (right). Taken from Lehoux [69].

machine, the Yuri I quadrotor. It consisted of four trussed girders that formed a pyramid-like structure with the pilot's cab positioned in its center. Each girder featured a huge, downwards-pointing two-bladed rotor with the blades almost touching the ground. Human power was transmitted via an oval sprocket connected to a flywheel, which transferred the smoothed pedaling energy to the rotors via light cables guided over winches.[2] The rotors had a diameter of 10 m, each spanned a blade area of 35.2 m^2, and they achieved a rotation speed of approximately 20 rpm. The craft's empty weight was 38 kg, the weight of its initial pilot, the triathlete Norikatsu Ikeuchi, was 50 kg, summing up to 88 kg that had to be lifted under human power.

The first HPH that managed to lift off and hover was the Da Vinci III machine built by students of the California State Polytechnic University, supervised by Professor William B. Patterson. In December 1989, Da Vinci III set the official record with an achieved height of 0.2 m and a hover endurance of 7.1 s. This record held for about five years until March 7, 1994, when Yuri I succeeded in breaking it with an equivalent height but an endurance of 19.46 s, while drifting away for 9.95 m. With that flight, witnessed by the Japanese Aeronautic Association, the Yuri I quadrotor established a new FAI duration record for HPHs, since then the official world record. In August 1994, at a symposium on human-powered flight held by the American Institute of Aeronautics and Astronautics (AIAA) in Seattle, Washington, Yuri I unofficially achieved a height of 0.7 m and flew for 24 s. Thereafter, Professor Naito started to work on a novel three-rotor HPH design with the aim to break the 1-min barrier.

Present description and data are based on Drees [70], Lehoux [69], Naito [71], Roper [72], and Sopher [73].

[2]The flywheel was permitted in that case, because it was used to smooth out the pedaling motion but it did not store energy, which would have violated the rules of the Sikorsky HPH Competition.

Figure 2.7: Dammar's 3rd prototype of 1991 (left) and the Area 51 Technologies HMX-4 flyer of 1999 (right), later sold under the name Draganflyer by Draganfly Innovations. Taken from Spectrolutions [74] (left) and Cedric [75] (right).

2.7 Spectrolutions, RC Flyers

Even though full-scale quadrotor aircraft never found their way into permanent usage, neither industrial nor military, in some smaller form they have meanwhile established their presence with increasing interest. The speech is about UAVs, which span a wide range of applications, for instance, from industrial and military drones over university research platforms to commercial RC models. Especially in research, small-scale quadrotor vehicles are the most commonly employed aerial platforms and core of numerous scientific projects. The reason why quadrotors dominate over conventional single-main-rotor helicopters in this field is that they can be built without a means for cyclic pitch control and are therefore easier to produce, and consequently, less expensive.

One of the first companies that placed the focus on the development of RC quadrotor models was the Spectrolutions Inc., located in Blaine, Minnesota. Against the widespread belief that the well-known Draganflyer quadrotor vehicles were produced by the Draganfly Innovations Inc., actually, all Draganflyer quadrotors were developed and manufactured at the Spectrolutions Laboratory in Brooklyn Park, Minnesota, but sold over the years by various distributers. Specifically, they were designed by Michael A. Dammar, a prolific inventor and helicopter enthusiast.

After two preceding designs, first successful flights with an endurance of approximately 1 min were conducted in 1991 with the third prototype model. It consisted of a rectangular airframe constructed from perforated sheet aluminum with four propellers mounted at the corners and each propeller driven by a separate DC motor. In order to provide lateral stability − indispensable to release the operator from this exhaustive task − the control system was augmented with mechanical gyroscopes. Solid-state gyroscopes were already available at the time, but only at exceedingly high costs, which would have rendered the flyer sheer unaffordable for a large community.

Passing a brief gestation period, comprising of a belt-driven model in 1994, Dammar's great break-through came in the late 90s when camera manufactures started to exploit

Figure 2.8: The Spectrolutions X-Pro flyer of 2002 (left) and the Spectrolutions V-Ti flyer of 2004 (right), sold under the names Draganflyer X-Pro and Draganflyer V-Ti, respectively, by Draganfly Innovations. Taken from Draganfly [6].

ceramic gyroscopes for image stabilization. The advance of this technique caused the prices for solid-state gyroscopes to drop dramatically, making them attractive for many other applications as well and also affordable to be incorporated in small-scale RC flyers. As a result, Dammar formed the Area 51 Technologies Inc., which announced its first commercial release in 1999 with the HMX-4 flyer. The craft, which is more commonly known as the Roswell Flyer, had a quite futuristic look, involved solid-state gyroscopes for improved stability, and achieved a flight endurance of around 5 min.

At about the same time, Dammar formed the Spectrolutions Inc. as a separate design company and started to work on his own rotor blades. In order to obtain a marketable product with all the required properties, he went through a series of almost fifty different blade designs. Around 2000, the Draganfly Innovations Inc. became the sole distributer of the HMX-4 flyer and sold it under the name Draganflyer. About one and a half year later, Area 51 Technologies dissolved and Spectrolutions continued to sell Original Equipment Manufacturer (OEM) kits to Draganfly Innovations. In particular, the first large flyer and predecessor of the later Draganflyer X-Pro appeared in 2000, followed by the commercial release of the Draganflyer II in 2001, which represented an improved version of the Draganflyer I prototype in regard to easier customer assembly. In 2002, the Draganflyer III was released, which contained an integrated dual-conversion Frequency Modulation (FM) radio receiver and, for that time, already achieved a reasonably-well flight endurance of around 15 min.

The largest model of Spectrolutions, the Draganflyer X-Pro, was released in 2002, which is now sold under the name X-Pro Flyer (available at [7]) by the VeraTech Aero Corp., another company formed by Dammar. The X-Pro incorporates numerous Carbon-Fiber Composite (CFC) components in its design and is equipped with four rather strong, collapsible girders, intended to save space when transporting the helicopter (patented by M. A. Dammar [76]). The propulsion system involves belt-drive gearing between its relatively large DC motors and two-bladed rotors, which can be upgraded to three-

bladed rotors if desired. The blades are self-manufactured by Spectrolutions and consisted originally of CFC, but are now also available in molded plastic, aimed to provide the same efficiency at reduced costs for the customer. The X-Pro has a total diameter of 1.2 m, an empty weight of 2.3 kg, is capable of carrying payloads of up to 0.5 kg, and achieves a flight endurance of around 20 min.

The so far last Draganflyer model designed and manufactured by Spectrolutions, the Draganflyer V-Ti (available at [77]), was released in 2004. The airframe consists of four small CFC tubes, which are connected at their far ends by four additional, integrity-increasing tubes, giving the craft a closed rectangular shape. Each of the corner connectors mounts a small DC motor that drives a rotor equipped with rigid-foam blades. The V-Ti contains a thermal-based self-leveling system (patented by J. A. Gwozdecki [78] and licensed to Spectrolutions, Inc.), which represents an advanced form of stabilization control. It has a total diameter of 0.76 m, an empty weight of 0.482 kg, can carry payloads of up to 0.2 kg, and achieves a flight endurance of around 15 min.

Besides possible other contemporary competitors, it appears that the first commercial release of an RC model quadrotor is attributable to the Japanese company Keyance. They sold a small four-rotor vehicle capable of flying a little more than a minute, but frequently became instable after a few seconds, essentially due to vibrations coupled into the mechanical gyroscopes. Accordingly, Spectrolutions was not the first company to sell RC quadrotors, but they were the first with their HMX-4 flyer to bring out a truly viable production version. In contrast to the large X-Pro flyer, which can rarely be found in scientific usage, the much smaller models of the Draganflyer series, primarily version III and V-Ti, have rapidly become very popular aerial platforms since their appearance, which serve as platforms in many ongoing research projects.

Present description and data are based on Draganfly [79], Sacco [80], correspondence with Mike Dammar, President of Spectrolutions, Inc. (now also Director of Engineering at VeraTech Aero, Corp.), and Spectrolutions [81, 74].

Remark 3. In order to be consistent with the International System of Units (Système International d'Unités), the Imperial units *foot* (ft), *mile* (mi), and *pound* (lb) have been converted to the SI units *meter* (m), *kilometer* (km), and *kilogram* (kg), respectively, whereas the obsolete units *horsepower* (hp) and *shaft horsepower* (shp) have been kept for more tangibility than is given with the SI unit *kilowatt* (kW).

Remark 4. All of the pictures presented in this chapter, especially the faded historical black-and-white photographs, have been digitally remastered by the author with the aim to improve their quality and appearance.

Chapter 3

Platform Development

The engineer is the key figure in the material progress of the world. It is his engineering that makes a reality of the potential value of science by translating scientific knowledge into tools, resources, energy and labor to bring them into the service of man ... To make contributions of this kind the engineer requires the imagination to visualize the needs of society and to appreciate what is possible as well as the technological and broad social age understanding to bring his vision to reality.

Sir Eric Ashby (1904−1992)

The first major chapter imparts the essentials of the constructive process that led to the prototype JAviator V1 (see Figure 3.1) and its successor the JAviator V2 (see Figure 3.2). Beginning with a detailed description of the airframe construction in Section 3.1, the propulsion system design is explained thoroughly in Section 3.2. Thereafter, the focus is shifted to the onboard electronics, which can be seen as the second major part concerning the platform development. The description of the avionics begins with the power supply covered in Section 3.3, continues with the sensor equipment explained in Section 3.4, and concludes with the computer system addressed in Section 3.5.

3.1 Airframe Construction

When it was decided to develop an entire quadrotor platform completely from scratch, three essential properties were considered. First, the aircraft should be powerful enough to carry payloads of up to 1.5 kg without significant loss in controllability. Second, it should be highly robust to withstand small crashes without serious damage. Third, and most important, the construction should facilitate simple reproducibility, and therefore, not contain any molded or casted parts. In this sense, the envision was to design a quadrotor aircraft that can be built from a bunch of widely available off-the-shelf accessories, based

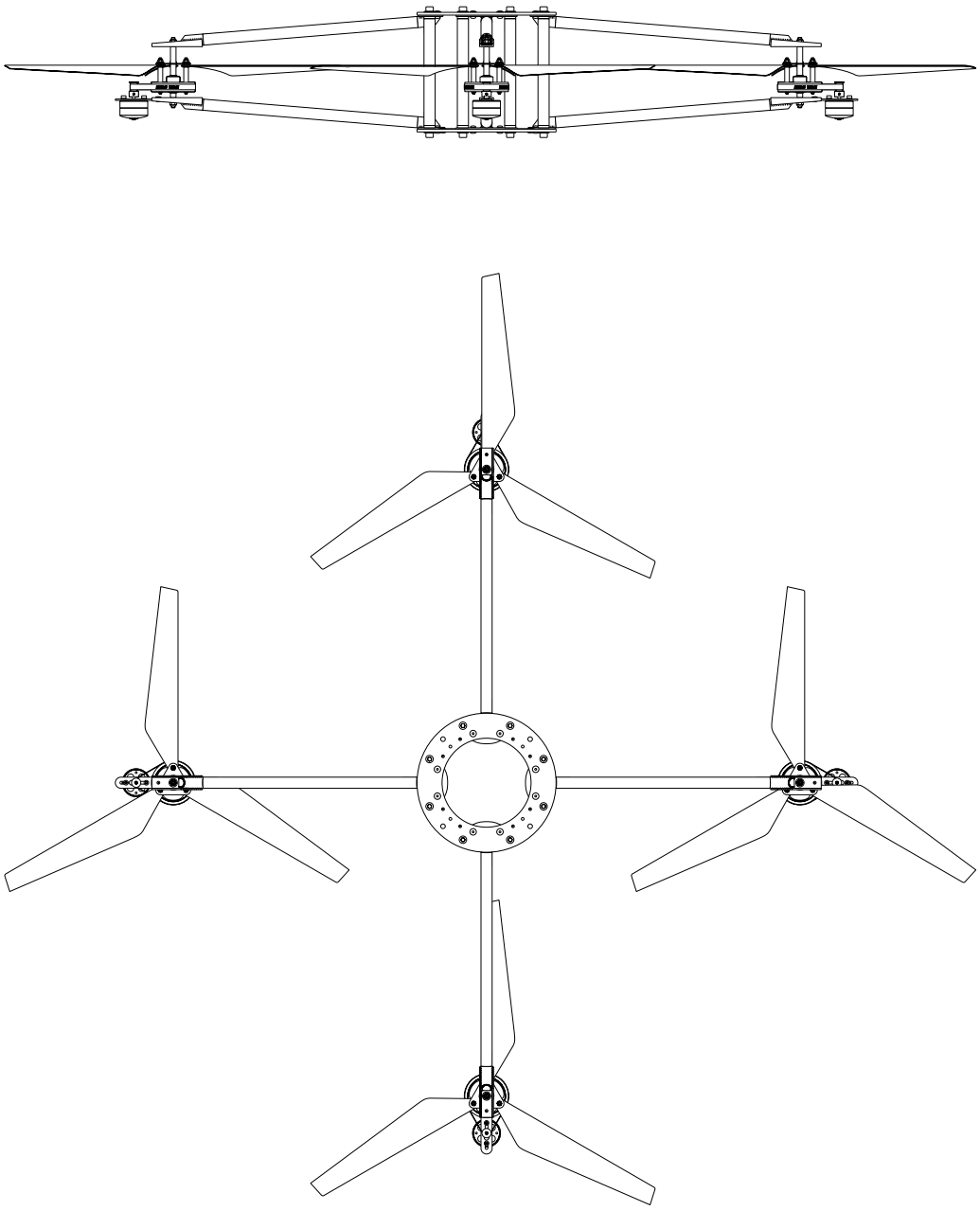


Figure 3.1: Prototype design of the JAviator V1.

on a small set of easy-to-grasp construction drawings, by almost everyone provided with the required tools and fairly good handicraft skills.

In searching for an appropriate design that satisfies all of the previously addressed properties, the ideal shape was found in form of a bicycle wheel with intersecting spokes. This kind of wheel is extremely robust due to its ingenious design and nevertheless quite simple in its construction. Imagine a horizontally positioned bicycle wheel comprising of merely four spokes originating from the upside of the wheel's hub and another four

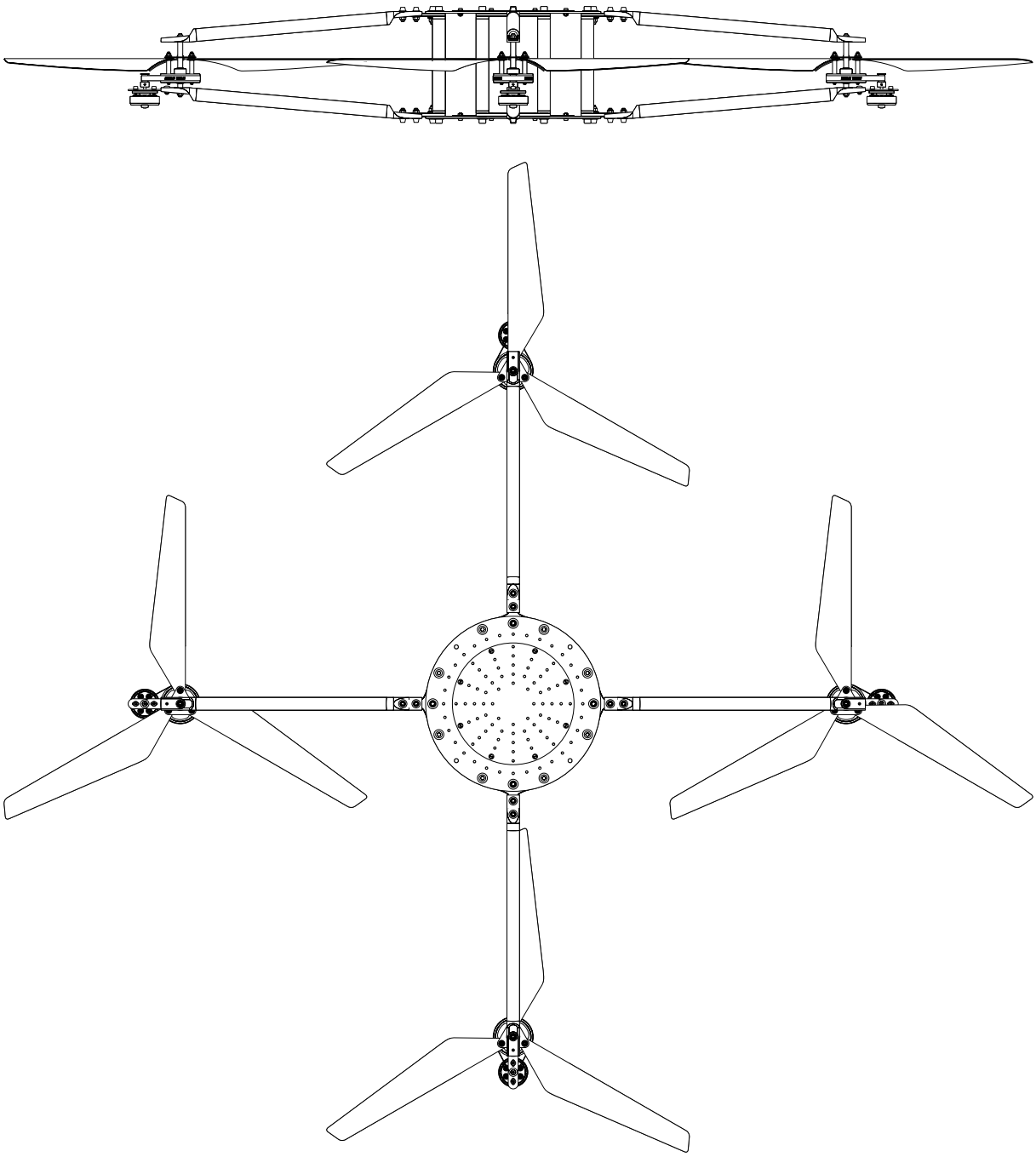


Figure 3.2: Initial design of the JAviator V2.

originating from the hub's underside. Now imagine removing the surrounding rim and connecting each pair of a top and opposing bottom spoke with some vertical rod. The resulting creation represents the underlying structure of the JAviator airframe and actual origin of its fully symmetrical design.

Based on this concept, the airframe is composed of a central fuselage (hub) with four girders (spokes), each carrying a three-bladed rotor. It is of cruciform when viewed from the top and shaped like a horizontal bicycle wheel containing four "top" and four "bottom"

spokes when viewed from the side. The fuselage is designed like a cage and consists of two flat rings connected via eight vertical tubes. These fuselage rings are closed in normal operation with thin plates, where the bottom fuselage plate carries the battery and the top one the inertial sensor. The girders comprise of a top and bottom tube of the same diameter as the fuselage tubes. They are connected at their outer ends by a vertical rod that serves as the rotor shaft. At their inner ends, they are connected to the top and bottom fuselage ring. Due to the symmetry of the top and bottom frame structure, this design not only allows to incorporate light and thin materials, it also provides high mechanical integrity.

3.1.1 Design Elaboration

The development of the JAviator design started with the prototype JAviator V1, depicted in Figure 3.1, aimed to proof our design considerations and gather experiences with this type of aerial platform. This aircraft served as our primary testbed for more than two years before it was withdrawn from service. During this time, it turned out that it would be advantageous to have more space for additional avionics equipment, and eventually, a larger or even second battery. As a consequence, the maximum payload capacity of approximately 1.5 kg by then would have had to be increased in favor of more payload. Though it would have been possible to employ stronger motors, the V1's maximum lift capacity was restricted by the diameter of its rotors, which, with a payload exceeding 1.5 kg, would have had to be driven far above their technical limit.

In order to provide more fuselage space for additional payload, the whole V1 design had to be scaled up appropriately. Furthermore, to ensure sufficient thrust reserves, the propulsion system had to be modified to attain a maximum payload capacity of around 2.5 kg. Based on the V1 prototype and the experiences gained through numerous tests, the initial design of the JAviator V2, depicted in Figure 3.2, was created. The propulsion system could be taken over without significant changes, merely equipped with slightly stronger motors and larger rotor blades, whereas almost all airframe dimensions had to be enlarged to meet the aforementioned requirements. The general dimensions of both the V1 and the V2 design are summarized in Table 3.1.

Table 3.1: JAviator V1-versus-V2 general dimensions.

Dimension	JAviator V1	JAviator V2	Unit
Fuselage Height	129	129	mm
Fuselage Diameter	150	210	mm
Rotor Diameter	424	501	mm
Total Diameter	1110	1300	mm
Empty Weight[1]	1830	2190	g

[1] *Empty Weight* refers to the ready-to-flight weight including the battery and all onboard electronics.

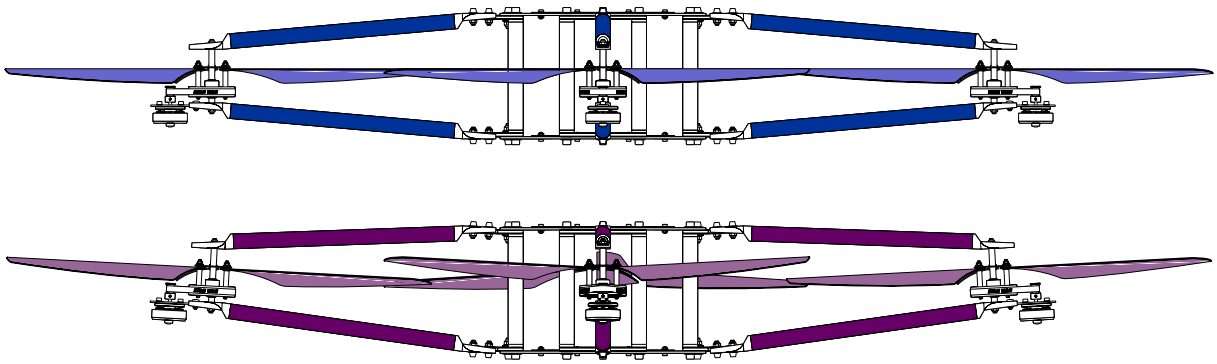


Figure 3.3: JAviator V2 initial design with fully symmetrical top and bottom frame (top) versus JAviator V2 final design with 3-deg inwards-inclined girders (bottom).

Analogous to the JAviator V1, the initial design of the JAviator V2 consisted of a fully symmetrical top and bottom frame structure. However, this version of the V2 never left the drawing board. The reason is that sometime during test flights conducted with the V1 prototype, the question arose, what if applying some inwards-directed inclination to the rotors? In other words, would such an inclination positively affect the craft's lateral stability? Investigations regarding early quadrotor designs revealed that some historic aircraft, like the de-Bothezat-Jerome helicopter of 1922 (see Section 2.2), had already incorporated this idea. Accordingly, in the endeavor to enhance the V2's lateral stability, the girders were designed to support an inwards-directed inclination at angles of up to 6 degrees. This essential difference between the V2's initial and final design is illustrated in Figure 3.3, where the girders in the final design are inclined inwards at an angle of exactly 3 degrees. Basically, there is no need for inclining the rotors of a quadrotor aircraft. Nevertheless, we observed that the slightly conical air cushion generated by the inclined rotors as well as the lowered barycenter positively affect the JAviator's aerodynamical behavior. We did not verify this observation formally, but noticed a more stable behavior during lift-off and when flying in ground effect.

3.1.2 Composites and Joints

Referring to the aforementioned envision of designing a quadrotor aircraft that can be built entirely from off-the-shelf accessories, and hence does not contain any molded or casted parts, appropriate connectors for joining the composite-based airframe components were needed. Due to the JAviator's symmetrical frame design, the amount of different connectors could be reduced to a lower number in the following way. The V2 airframe contains eight vertical fuselage tubes as well as eight horizontal girder tubes. Each of the four girders is connected with its top and bottom tube to the top and bottom fuselage ring, respectively, between a pair of fuselage tubes. Figure 3.4 illustrates this construction in wireframe representation and highlighting the girder connectors in orange, the girder flanges in green, and the fuselage connectors in red. In case of the vertical fuselage tubes,

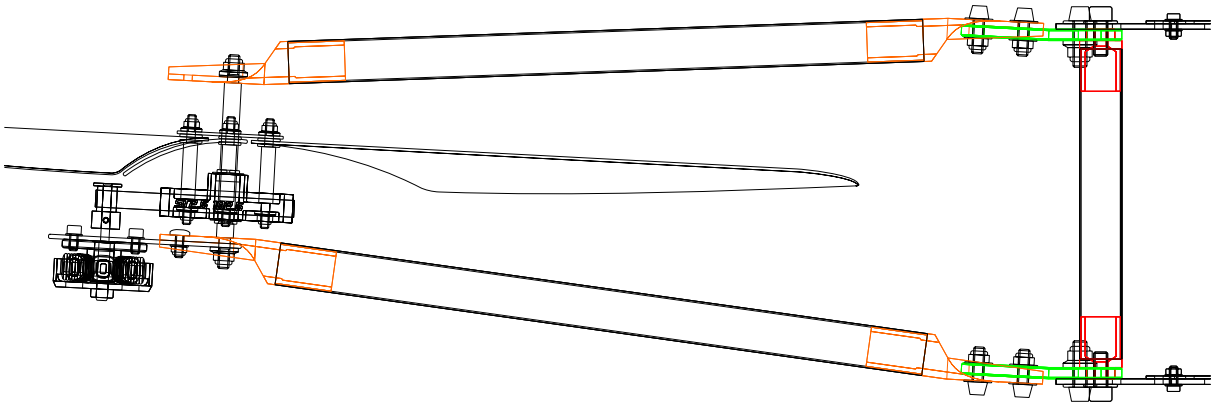


Figure 3.4: Girder connectors (orange), girder flanges (green), and fuselage connectors (red) in the wireframe graphics of a JAviator V2 girder.

the same connector (red) can be used on both ends for connecting the eight fuselage tubes with the top and bottom fuselage ring. The design of the girder connectors (orange), however, turned out to be a challenge. A girder consists of a top and a bottom tube. Thus, there are four tube ends to be equipped with appropriate connectors that need to satisfy five elementary requirements as specified below.

1. The top and bottom connector on the fuselage side need to connect to the top and bottom fuselage ring, respectively.
2. The fuselage-side connectors should allow to inwards-incline the girder at arbitrary angles of up to 6 degrees.
3. The top and bottom connector on the rotor side need to mount the rotor shaft.
4. The bottom rotor-side connector must provide a means for mounting a motor.
5. The top rotor-side connector should allow for mounting a signal light.

For the prototype version of the JAviator, all connectors were hand-made and built from conventional, off-the-shelf aluminum tubes and profiles. Because the V1 prototype was neither designed for inclining the girders nor installing signal lights, the top girder tube connectors could be made identical to the bottom ones. Nonetheless, this solution was based on two different connectors. Furthermore, the fuselage-side connectors did not facilitate inclining the girders.

In the effort to create a single connector that fulfills all of the five requirements, the connector presented in Figure 3.5 (left) was designed. This connector serves as the end tap for both sides of both the top and bottom girder tube. The problem with enabling an inclination was solved by designing an appropriate girder flange, green-highlighted in the wireframe graphics and depicted in Figure 3.5 (middle), which contains a connecting strap that fits in the milled-out portion of the girder connector. The large holes at the flange's curved ends have the purpose that each girder flange gets fastened down by two fuselage connectors, depicted in Figure 3.5 (right), which fit with their small ends into

Figure 3.5: JAviator V2 CNC-turned and -milled girder connectors (left), laser-cut girder flange (middle), and CNC-turned fuselage connectors (right).

these holes when screwed together with the fuselage ring. Due to this technique, it is ensured that the girder flanges, and thus the girders, keep in correct, orthogonal position relative to the fuselage rings. The angle between the turned and the milled part of the girder connector was chosen to enable inclinations of up to 6 deg without stressing the comprised components. Any desired inwards-directed inclination in the admissible range can be achieved by adapting the connecting straps to the desired angle and appropriately shortening the top girder tubes. In case of the girder flange shown in the figure, the connecting strap is given an inclination of 3 deg, which demands the top girder tube to be exactly 7 mm shorter than the bottom one.

Referring to the rest of the previous connector specification, which addresses means for mounting a motor and a signal light, these problems could be solved easily. In particular, the milled-out portion of the girder connector was chosen such that it allows to assemble a thin strap of titanium sheet for carrying a motor as well as a small PCB for attaching a signal light in the final configuration.

3.1.3 Parts and Assembly

The most obvious choice for constructing an ultralight airframe was to incorporate CFC as the main material in combination with aluminum for the connecting parts. For those components that must provide very high integrity, such as rotor shafts, rotor triangles, and motor carriers, titanium was chosen. In contrast to the V1 prototype, which was completely hand-manufactured, for the more advanced JAviator V2 the focus was placed on Computerized Numerical Control (CNC) fabrication to attain a maximum degree of precision and reproducibility. In particular, the following fabrication methods and specific materials were used for the custom-built airframe and rotor components.

- The fuselage rings and plates are flow-jet-cut from conventional CFC plates.
- The connecting tubes are diamond-disk-cut from ultralight CFC tubes.
- The connectors are CNC-turned/milled from high-cohesion aluminum rounds.

- The connecting flanges are laser-cut from high-cohesion aluminum plates.
- The motor carriers and rotor triangles are laser-cut from titanium plates.
- The rotor gears are CNC-turned/milled from aluminum belt-drive blanks.
- The rotor pylons are CNC-turned from high-cohesion aluminum rounds.
- The rotor shafts are CNC-turned from high-grade titanium rounds.

In order to fully exploit CNC fabrication, and hence reduce the production costs, the V2 airframe was designed to comprise of a minimum number of different components. This was accomplished, first, by incorporating as much frame symmetry as possible, and second, by designing a girder tube connector to be used on both ends of both the top and bottom girder tube. The fuselage and girder connectors are designed for bonding with the fuselage and girder tubes, respectively. We tested many different adhesives and found that the best results are achieved with slow-curing, epoxy-based two-pack adhesive. The basic components required to build the airframe of a JAviator V2, as well as their quantities of occurrence and corresponding masses, are listed in Table 3.2.

Table 3.2: JAviator V2 basic airframe components.

Component	Material	Quantity	Mass (g)	Total (g)
Fuselage Ring	CFC 2.0 mm	2	53.30	106.60
Fuselage Plate	CFC 1.0 mm	2	25.60	51.20
Fuselage Tube	CFC 0.5 mm	8	4.00	32.00
Fuselage Connector	AlZn5.5MgCu	16	4.50	72.00
M5 Fuselage Screw[2]	Al Alloy	24	1.18	28.32
M5 Fuselage Nut[3]	Al Alloy	8	0.62	4.96
Girder Tube	CFC 0.5 mm	8	8.00	64.00
Girder Connector	AlZn5.5MgCu	16	8.90	142.40
Girder Flange	Al Alloy	8	12.70	101.60
M4 Girder Screw[4]	Al Alloy	16	0.74	11.84
M4 Girder Nut[5]	Al Alloy	16	0.44	7.04
M3 Girder Screw[6]	NiCr Alloy	8	0.78	6.24
M3 Girder Nut[7]	NiCr Alloy	8	0.42	3.36
Sum				631.56

[2]Differing from V2 drawings: M5×20 raised-head Allen screw without washer.
[3]Conform with V2 drawings: M5 self-locking collar nut without washer.
[4]Differing from V2 drawings: M4×20 raised-head Allen screw without washer.
[5]Differing from V2 drawings: M4 self-locking collar nut without washer.
[6]Conform with V2 drawings: M3×8 socked-head Allen screw with M3 washer.
[7]Conform with V2 drawings: M3 self-locking non-collar nut with M3 washer.

Figure 3.6: Bottom fuselage ring with girder flanges installed (left), with fuselage plate and tubes installed (middle), and complete JAviator V2 fuselage (right).

Once all components are at someone's disposal, the assembly of a V2 airframe is straightforward. Starting with the fuselage assembly, Figure 3.6 (left) shows a bottom fuselage ring with four girder flanges installed, which are screwed to the fuselage ring through their center holes. They are aligned to be in correct position on the fuselage ring when installing the fuselage tubes, which fit with their connectors into the holes at the flanges' curved ends. Strictly speaking, this central screw is not desperately needed. However, it is quite useful in case of repairs, because it allows to remove the complete top and bottom frame together with the girder tubes without the need to reassemble the whole airframe. The top fuselage ring is prepared in the same manner as the bottom one, except for one important difference: the small connecting straps of the flanges, which are pointing upwards on the bottom ring, need to point in opposite direction on the top ring. The next step is to install the eight fuselage tubes, shown in Figure 3.6 (middle), which are screwed together with the girder flanges and the fuselage ring. Note that in the figure, the fuselage plate, which serves as battery carrier, is already fastened to the upside of the fuselage ring. Though this conforms to the construction drawings, it turned out to be more beneficial fastening this plate to the downside of the ring using either wing nuts or some other form of quick coupling device. This allows for conveniently exchanging the battery and charging it outside the helicopter under secure conditions. The installation of the top fuselage ring assembled with girder flanges, shown in Figure 3.6 (right), completes the fuselage. It merely misses the top fuselage plate, which carries the inertial sensor, and therefore, is installed together with the onboard electronics. The present JAviator V2 fuselage is simple in its design, features fast and easy (re)assembly, and also ensures high integrity at relatively low weight. Compared to most other quadrotor airframes, this cage design acts like a protection frame for the onboard electronics and has proven in crashes to withstand hardest collisions without serious damage.

The girder assembly begins with preparing the rotor-side connectors of the girder tubes. The only difference between the girder connectors used for the fuselage side and the ones used for the rotor side is the diameter of the drilled holes. The fuselage-side connectors

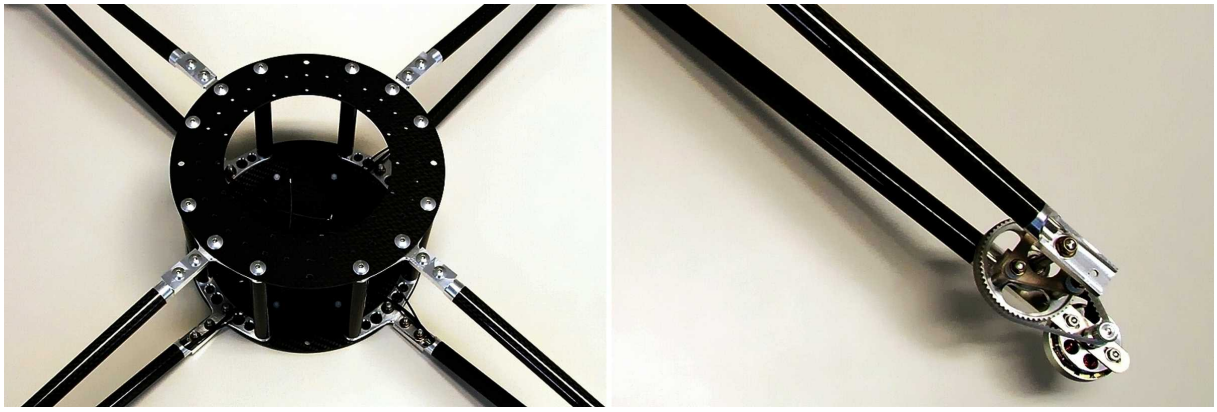


Figure 3.7: Fuselage with installed girders completing the JAviator V2 airframe (left) and girder rotor-side end with propulsion group installed (right).

contain two bores with a diameter of 4 mm, whereas the rotor-side connectors contain one 4-mm bore and one 3-mm bore. Before their installation to the fuselage, the four rotor-side connectors of the slightly longer bottom girder tubes are equipped with the motor carriers, which are screwed to the connectors using the 3-mm bore. In similar fashion, the four rotor-side connectors of the top girder tubes are equipped with the PCBs for the signal lights. On the fuselage side, the top and bottom girder tubes are screwed to the top and bottom connecting straps, respectively, as shown in Figure 3.7 (left). On the rotor side, the pre-assembled rotors together with the cogged belts are installed between the open girder ends. A girder's rotor-side end equipped with complete propulsion group is shown Figure 3.7 (right). Note that it is strongly recommended not to fully tighten the fuselage screws before finishing the installation of all rotors. This procedure ensures proper plane parallelism of the rotor shafts and also avoids possible tensions in the frame structure that might occur otherwise.

3.2 Propulsion System

The propulsion system of the JAviator consists of four identical propulsion groups. Each rotor head mounts three rotor blades, where one pair of diagonally opposing rotors is equipped with clockwise and the other pair with counter-clockwise spinning blades. Rather than employing toothed gears for the transmission between motors and rotors, it is performed via cogged belts, which tend to be less noisy and are more robust for this purpose. The cogged-belt-drive gears and belts are industry standard DIN-norm[8] T2.5 components (available at [82]), which are produced in huge variety. They can be found basically in tools of any description, for instance, small planing machines.

The central part of the rotor head is a 60-tooth cogged-belt-drive gear fabricated from a die-casting blank. The unfinished blank already contains the 60-tooth structure milled into its jacket as well as a center bore, but no other special shape. During fabrication, this

[8]DIN stands for "Deutsches Institut für Normung" meaning German Institute of Normalization.

Figure 3.8: Top view (left) and bottom view (middle) of completely assembled rotor head and close-up view of 4-pole-pair 3-phase AC motor (right).

blank is reduced to a much lower weight by removing most of the redundant material. The machined rotor gear contains a large fitting bore for the ball-bearings and three equilaterally arranged small bores for the rotor pylons, which are used for mounting the rotor blades. The pylons are conjuncted by a thin triangle on their top ends, shown in Figure 3.8 (left), for compensating centrifugal forces that might cause pylon deformations at high rotation speeds. The rotor head is kept in position on the rotor shaft with two circlips, one at the top and one at the bottom, as shown in Figure 3.8 (middle). These circlips not only secure the rotor head against vertical movements, the top circlip actually transfers the entire generated lifting force via the rotor shaft into the airframe. Note that the rotor shafts, besides carrying the rotors, serve as primary airframe components that connect the top and bottom girder tubes. Compared to conventional quadrotor designs, where merely one end of the rotor shaft is connected to the airframe, the present double-end-mounted rotor design has two significant advantages: first, it enables precise adjusting of the four rotor shafts regarding plane parallelism, and second, it prevents the rotors from undesired twisting that can occur with one-end-mounted rotors.

3.2.1 Power Transformation

In order to generate maximum thrust at minimum weight, the propulsion groups feature 3-phase AC permanent-magnet synchronous motors, also known as *brushless motors* or *outrunners*. Due to the permanent excitation through magnets, they do not require an excitation coil, and therefore, do not contain any slip rings. As a consequence, this type of 3-phase AC motor rotates synchronously to the induced 3-phase AC field independently of the transformed power and, in general, achieves a much higher coefficient of efficiency that lies typically around 0.98. In case of DC power supply through a battery, the required 3-phase AC is usually generated with a Brushless-Motor Controller (BMC), which transforms the DC input into a 3-phase AC output based on Transistor-Transistor Logic (TTL)-controlled Field Effect Transistor (FET) bridges.

All of the motors ever used for the JAviator are custom-built versions comprising a

great deal of skilled handiwork. The prototype JAviator V1 was equipped with 3-pole-pair motors, which, with a weight of merely 35 g, had a remarkable power output of around 200 W. However, to attain a higher payload capacity, for the JAviator V2 we switched to a more advanced 4-pole-pair version, shown in Figure 3.8 (right), with a weight of 43 g and a power output of around 250 W. It is composed of CNC-fabricated aluminum parts, equipped with a titanium axle and an outrunner containing fourteen gold-plated neodymium magnets. The armature is hand-coiled and contains only ten windings of 0.6-mm wire per pole. This motor is therefore significantly stronger and achieves a much higher power-to-weight ratio than conventional motors of identical weight. The primary purpose of this motor, namely, to generate high power output should also be apparent to some extent by taking a look at the rather coarse, low-impedance coil of the armature (see Figure 3.8). This type of motor (available at [83]) was originally designed for model planes, but re-shaped for us according to our specifications. In particular, the following modifications were applied.

- The armature side was re-shaped to fit the JAviator-specific motor carrier and all redundant material removed to reduce the weight and improve cooling.
- The axle was reinforced by extending the diameter from 3 to 4 mm with a short 3-mm stub for the pinion and placed upside-down to have the axle's stub located on the mounting side rather than the outrunner side.
- The armature poles were reconfigured to contain only ten windings of 0.6-mm wire and the power input wires replaced by wires with a higher load rating.

On the JAviator V1 equipped with the 3-pole-pair motors, the rotors' 60-tooth gears were combined with 10-tooth pinions resulting in a 1:6 transmission. Sticking to this transmission was one possible option for the JAviator V2. However, due to the stronger 4-pole-pair motors, the opportunity of upgrading to 12-tooth pinions, thus forming a 1:5 transmission, cropped up. In order to find the most beneficial configuration with regard to flight endurance and payload capacity, both gearing options were tested exhaustively under various payload conditions. Surprisingly, although the V2's rotor diameter is 77 mm larger than the V1's, the 12-tooth pinions did not seem to stress the motors seriously. Hence we decided to employ the more effective 1:5 transmission for the JAviator V2, because the flight endurance without payload is similar to the 1:6 transmission, but it provides more thrust reserves in favor of higher payloads.

Referring to the so far described construction of the rotor head, brushless motor, and transmission used, a listing of all essential components that are part of the JAviator V2 propulsion system is given in Table 3.3.

[9] *Mass* refers to the average mass of a rotor blade with a fabrication-dependent deviation of ±0.5 g.
[10] Conform with V2 drawings: M3 self-locking collar nut without washer.
[11] Conform with V2 drawings: M4 self-locking collar nut without washer.
[12] *Material* refers to the motor's stator without armature and external rotor without magnets.
[13] Conform with V2 drawings: M3×6 socked-head Allen screw with M3 washer.

Table 3.3: JAviator V2 propulsion system components.

Component	Material	Quantity	Mass (g)	Total (g)
Rotor Gear	AlCuMgPb	4	14.60	58.40
Rotor Pylon	AlZn5.5MgCu	12	1.44	17.28
Rotor Shaft	TiAl6V4	4	7.77	31.08
Rotor Bearing	NiCr Alloy	8	0.76	6.08
Rotor Triangle	Ti Alloy	4	2.16	8.64
Rotor Blade[9]	CFC 1.2 mm	12	12.00	144.00
M3 Pylon Nut[10]	Al Alloy	24	0.24	5.76
M4 Shaft Nut[11]	Al Alloy	8	0.44	3.52
Motor Carrier	Ti Alloy	4	5.96	23.84
4-Pole-Pair Motor[12]	Al Alloy	4	42.60	170.40
M3 Motor Screw[13]	NiCr Alloy	8	0.68	5.44
12-Tooth Pinion	AlCuMgPb	4	3.23	12.92
ST2.5-182.5 Belt	Polyurethane	4	1.28	5.12
Sum				492.48

3.2.2 Thrust Generation

Besides an efficient transformation of supplied power into rotational energy, for the final transformation into lifting force it is of vast importance to have rotors of high efficiency. In case of small-scale quadrotor vehicles, an acceptable alternative is to use propellers instead of rotors. The advantage is the elimination of a gearing box, since they are usually mounted directly on brushless motors. However, propellers are designed to primarily provide forward propulsion rather than vertical lift. They are therefore less efficient for large-scale quadrotors aimed to carry high payloads. In contrast to conventional helicopters with a single main rotor, where vertical and directional propulsion is attained by collectively and cyclically pitching the blades, a quadrotor aircraft can be designed with fixed-pitch blades to be entirely controlled by differentially changing the rotation speeds of the four rotors. This form of purely rotor-speed-based control was realized successfully for the first time in the Curtiss-Wright VZ-7AP quadrotor of 1958, which was equipped with fixed-pitch propellers (see Section 2.5). Accordingly, regarding quadrotors the primary purpose of the rotor or propeller blades is to provide vertical lift without the need for varying their "bite" into the air. In case of rotors, paddle-shaped blades seem to be most efficient for moving large amounts of air. This unconventional blade design can also be found in the Œhmichen No.-2 helicopter of 1922 (see Section 2.3). In searching for an advanced paddle-shaped blade design suitable for the JAviator, the blades of the X-Pro Flyer (see Section 2.7) appeared to be a promising approach. The reasonably-well performance of the X-Pro blades (available at [84]) was also verified recently in the scope of an evaluation that addresses seven different blade designs (see [85]).

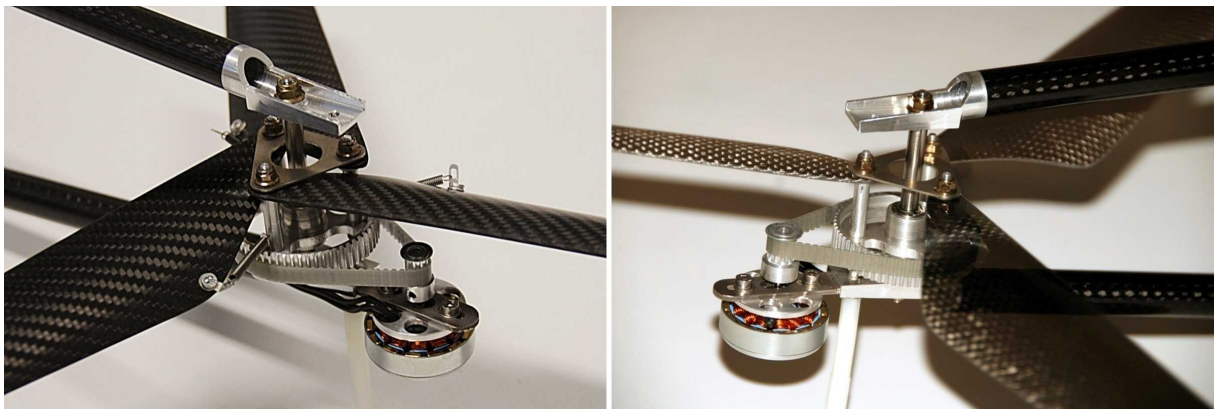


Figure 3.9: 1:6-geared (10-tooth pinion) rotor equipped with custom-built blades (left) and 1:5-geared (12-tooth pinion) rotor equipped with X-Pro blades (right).

Based on the X-Pro's blade design, the twelve blades for the prototype JAviator V1 were produced entirely by hand. Because the V1 was much smaller than the X-Pro Flyer, they were scaled down by approximately a third and cut out from a motorcycle muffler jacket consisting of metal-grid-reinforced CFC. These blades were extremely robust, but relatively heavy due to the enclosed metal grid. Consequently, for the JAviator V2 we considered employing ultrathin CFC blades. On account of the V2's larger airframe, the X-Pro blade dimensions could be taken over without significant change resulting in the blades depicted in Figure 3.9 (left). It can be seen that the three blades are in hinged position when not spinning. Strictly speaking, the blades are pretensioned via springs to achieve a self-centering behavior during rotation. There are two reasons for this: first, pretensioning the blades prevents them from fluttering at higher rotation speeds, and second, it eliminates the need for positioning the blades correctly before each flight. When rotating, the blades fully expand at approximately half of the maximum rotation speed. The blades shown in the figure are custom-built according to our specifications and similar in shape and size to the X-Pro blades, but only half as thick, and therefore, less in weight by more than a third. We achieved reasonable aerodynamical performance with these ultrathin blades and conducted experiments with even thinner blades. However, the pretensioning mechanism turned out to be rather cumbersome in practice and often caused long (dis)assembling times. To get rid of this problem, we decided to use stronger and more reliable blades for the JAviator V2 series production and therefore equipped all propulsion groups with off-the-shelf X-Pro blades, as depicted in Figure 3.9 (right). Note the transparency of the blades in this figure, which reveals that this shot was taken while the rotor was spinning at relatively high speed.

3.3 Energy Distribution

The energy consumed by the four motors and the avionics equipment is supplied by a single Thunder Power 4S3P 14.8-V 6300-mAh lithium-polymer battery (available at [86]). This

battery, which weighs exactly 458 g, is actually the heaviest component of the JAviator. Nevertheless, because the JAviator V2 provides sufficient payload capabilities and its fuselage offers enough remaining space, a second battery can be added to extend the flight time if desired. In the prototype JAviator V1, the BMCs were connected directly to the battery and the motors. In other words, in case of any hardware, software, and/or remote-connectivity failure, there was no way to shut down the motors independently of the onboard computer system. On the one hand, performing merely a software-triggered shutdown can be considered risky and unsafe, especially if the remote connectivity breaks down. On the other hand, cutting the Pulse Width Modulation (PWM) signals to the BMCs via some remotely controlled mechanism would not be very effective, because the BMCs wait for 3 s upon signal loss before they shut down.

3.3.1 Power Board Development

Primarily due to the aforementioned lack of safety, we decided to develop a PCB, which we call the Power Board, aimed to satisfy the following five requirements.

1. The PCB needs to provide a reliable means for cutting the power to the motors independently of the onboard computer system.
2. The shutdown mechanism should also allow to arm the helicopter. That is, rather than just providing a means for shutting down the motors, arbitrary (dis)arming should be possible.
3. For safety reasons, the remote connection used for (dis)arming must be independent of the remote connection between the helicopter and the ground station.
4. A mechanism for driving signal lights at the upper girder ends should be provided for indicating the helicopter's arming status.
5. To reduce the amount of wires between the battery and the BMCs, and to facilitate a quick replacement of the entire high-power electronics, the four BMCs should be integrated on the PCB rather than being placed separately.

Figure 3.10 depicts the resulting PCB layout developed for the Power Board. There are four identical units comprising of BMC, relay, and motor connector, located symmetrically in the four corners of the layout. According to the prevalent coloring conventions for PCB layers, color red represents the top layer and color blue the bottom one. It points out that the conductor paths between BMCs, relays, and motor connectors appear in dark violet, which is due to the overlay of an identical top and bottom routing. In case of the BMCs' AC output, this "double routing" prevents the conductor paths from overheating. However, in case of the BMCs' DC input, supporting wire bridges had to be incorporated to ensure a reliable distribution of the current, which can raise to 50 A under high-payload conditions. These bridges that consist of rather thick wires located around the board's center can be seen in Figure 3.11, which shows the fully populated Power Board.

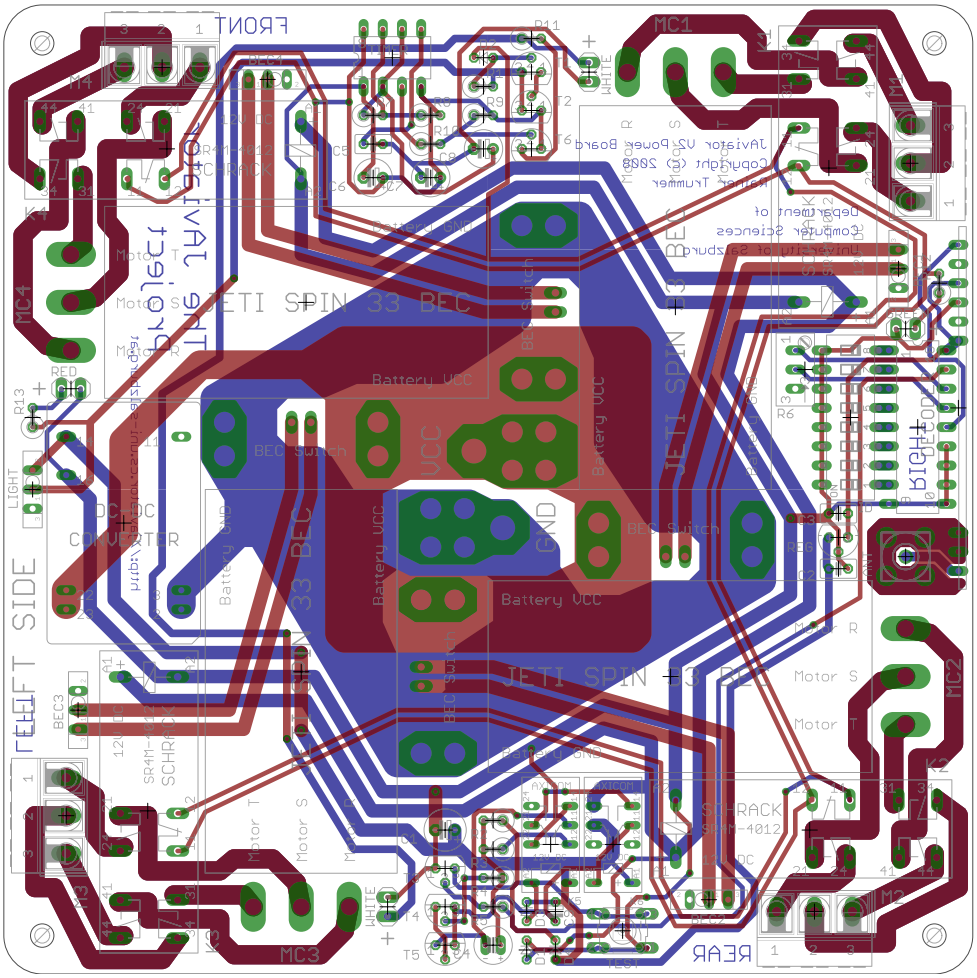


Figure 3.10: PCB design of the JAviator Power Board (real size).

Requirement (1) of cutting the power to the motors was solved by interrupting the 3-phase connection between each pair of BMC and motor with a 4-contact relay. This decision was based on the following consideration: when demanding full thrust, each motor consumes about 12 A, which sums up to a total of 48 A at 14.8 V supplied battery voltage. Cutting this high DC is not easy and would at least require some huge relay. Even if employing four separate, say, 15-A relays, the dimension and weight of such a relay would still exceed the admissible limit for this purpose. One final remark: cutting the power at the BMCs' input side is not advantageous in case of the JAviator, because it would also cut the power to the onboard computer system, which is supplied by the so-called Battery Eliminator Circuit (BEC) of at least one BMC or by interconnecting several (up to four) BECs in case more power is needed. Consequently, 4-contact relays were chosen for cutting the power directly between the BMCs and the motors. In order to obtain a high degree of safety and sufficient switching capability, we decided for the Schrack SR4M4012 security relay, which contains a separate chamber for each contact and is capable of switching a maximum of 8 A at 250 V AC per contact. This relay suffices in our case, because the BMC's DC input of roughly 200 W transforms to a 3-phase AC

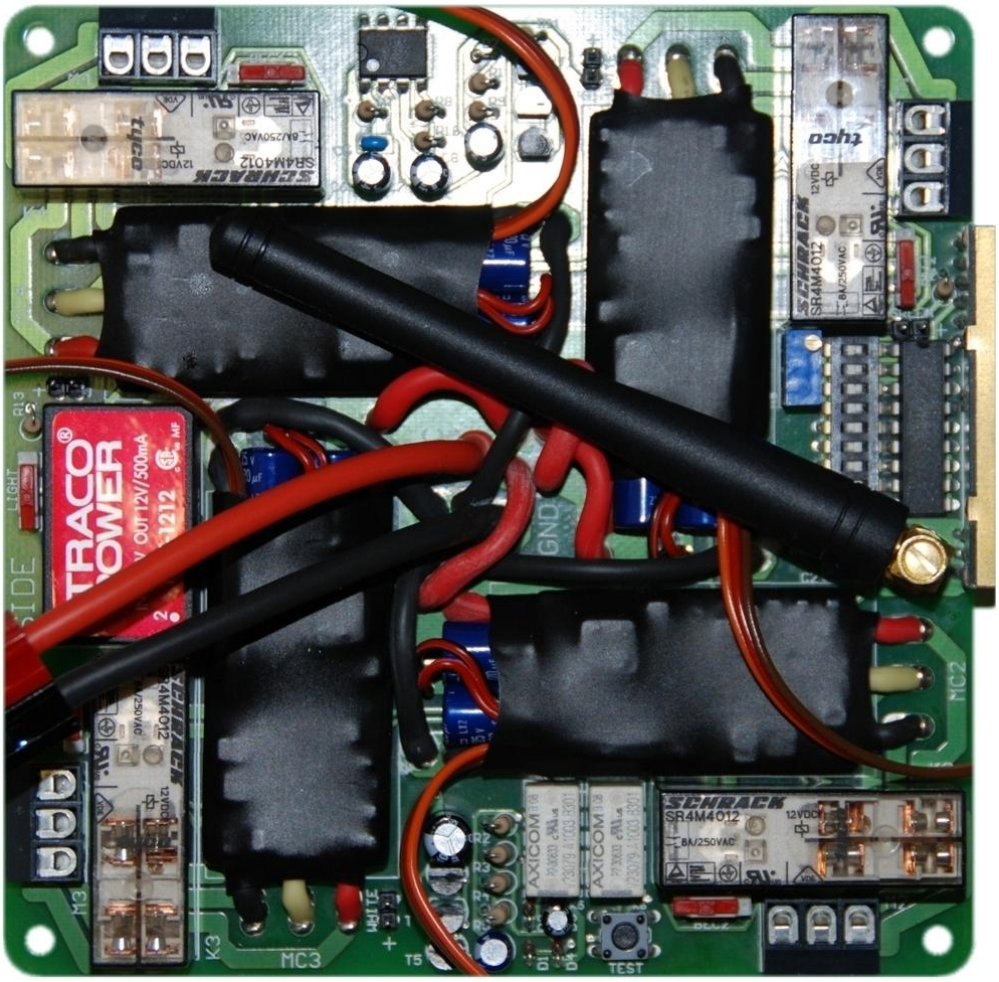


Figure 3.11: Fully populated JAviator Power Board (real size).

output of less than 120 W per phase. More precisely, a 120-deg phase shift implies that $P_{phase} = P_{total}/\sqrt{3} = 200\ \mathrm{W}/\sqrt{3} = 115.47\ \mathrm{W}$.

Requirement (2), which states that arbitrary (dis)arming should be possible, was solved by a mechanism that reacts to a transmitted signal in a time-dependent manner for both arming and disarming. More precisely, whenever a signal is transmitted from the sender, the mechanism disarms the helicopter, but at the same time, starts arming it by loading a specific capacitor of the relay control circuit. In order to fully load this capacitor, and hence to arm the helicopter, the signal must be sent for at least 3 s without interruption, because any interruption immediately resets the relay control circuit. As a consequence, the proper operation of the emergency shutdown mechanism is checked automatically with each arming procedure.

Requirement (3), addressing the software-independent remote connection, was solved by designing the emergency shutdown mechanism to operate on an entirely hardware-based, analog 868-MHz FM radio channel that bypasses the helicopter's software-based flight control system. Particularly, the radio channel integration consists of a Radiometrix RX3A 868-MHz FM receiver that matches a Radiometrix TX3A 868-MHz FM sender,

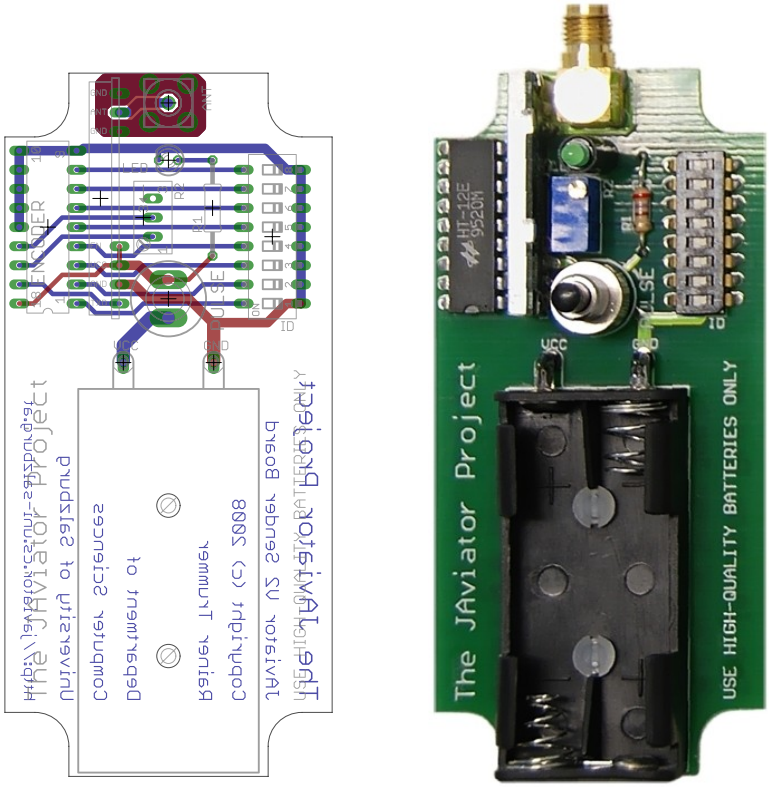


Figure 3.12: PCB design of the JAviator Sender Board (real size).

which is used to transmit the (dis)arming signal. The 868-MHz FM band was chosen due to a European restriction that prohibits the usage of frequencies in this band for longer than 30 s without interruption. In this sense, the entire 868-MHz FM band is reserved for transmitting short signals and may not be utilized for long-term data transmission, which accounts for reduced interferences.

In fulfilling requirement (4), demanding a mechanism for driving signal lights, the upper girder ends were equipped with navigation lights according to the international rules for aircraft: red light on the left wing, green light on the right wing, and flashing beacons at the front and rear end. Besides the purpose to give the JAviator V2 some fancy note, these lights are coupled individually with the relay control circuit for visualizing the helicopter's arming status.

Concerning requirement (5), the integration of the four BMCs on the Power Board, the challenge was to place the BMCs in a way that leads to shortest possible conductor paths in each unit encompassing a BMC, a security relay, and a motor connector. This problem was solved with a fully symmetrical arrangement of the four units by placing them in the four corners of the board.

The PCB layout belonging to the sender used for (dis)arming the high-power circuit is depicted in the left-hand part of Figure 3.12, the corresponding Sender Board is shown in the right-hand part of the figure. For the purpose of pairing multiple senders with multiple receivers, both the Sender Board's signal encoder and the Power Board's signal decoder support 8-bit channel encoding, hence offering a total of 256 possible combinations. The

associated 8-pole switch for code selection is located in the right-hand-side middle area on the Power Board and in the right-hand-side top area on the Sender Board.

3.3.2 Brushless-Motor Controllers

The four black, shrink-wrapped boxes in Figure 3.11 are the BMCs, which are soldered in with their flat label side facing down. In the present case, they are Jeti Spin-33 BMCs (available at [87]). This type of controller represents a lightweight version of the standard Jeti 33-A controller and accepts batteries up to 5S, meaning up to five lithium-polymer cells connected in series, which, with a cell voltage of 3.7 V, results in a maximum of 18.5 V. The controller contains a remarkably powerful BEC, normally used to drive the RC receiver and the servos, which is capable of providing up to 2 A. Due to the opportunity of consuming up to 8 A in total, there is no need to incorporate additional low-voltage regulators for the avionics equipment. However, the drawback of the Spin series − and that counts for the majority of commercially available BMCs − is the low internal resolution of merely 8 bit. In other words, the PWM input signal, independently of the resolution used, is mapped to a maximum of 256 speed steps. This might be sufficient for most applications in the domain of RC models, but turned out to be too coarse for performing advanced quadrotor control. To this end, we have started to replace the Spin controllers with Castle Creations Phoenix-35 BMCs (available at [88]), which utilize two extra bits to provide an extended 10-bit resolution. Accordingly, the PWM input signal is mapped to 1024 speed steps adhering to 4-time finer changes in the control signal. Similar to the Spin-33, the Phoenix-35 is also a lightweight controller that allows battery configurations up to 5S. However, nothing comes for free. The downside of this controller is a sheer unusable BEC that can only be used in conjunction with batteries not exceeding 3S. Because the JAviator V2 battery is a 4S type, the BECs of these controllers cannot be exploited for the avionics equipment and separate low-voltage regulators need to be employed. Nevertheless, the Phoenix-35 controllers encompassed a great improvement with respect to controllability.

3.4 Sensor Equipment

The essential, permanently involved instrumentation for flying the JAviator manually, but supported by automatic altitude and attitude control, consists of a SensComp Mini AE Ultrasonic Range Finder (URF), a self-fabricated Barometric Measurement Unit (BMU) featuring a high-precision Integrated Pressure Sensor (IPS), and a MicroStrain 3DM-GX1 Inertial Measurement Unit (IMU). The application-dependent instrumentation comprises of a Ubisense Ubitag 7022 RFID tag, which is part of a Ubisense Series 7000 Stationary Localization System (SLS), and two Dimetix LSM2-15 Laser Distance Sensors (LDSs). The auxiliary devices are exploited for position control, and due to their small sizes and weights, can be operated simultaneously on the JAviator, hence offering the opportunity to experiment with two different localization systems at the same time.

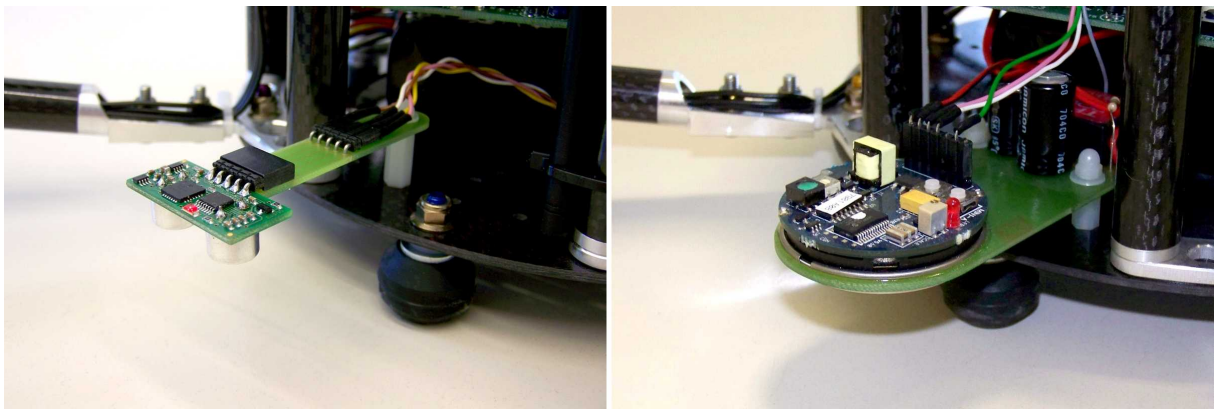


Figure 3.13: Devantech SRF10 digital URF (left) of the early project phase and SensComp Mini AE analog URF (right) employed currently.

3.4.1 Ultrasonic Range Finder

The primary source for determining the JAviator's altitude is a URF that measures the distance to the ground. The first device exploited for this purpose was a Devantech SRF10 URF (available at [89]), shown in Figure 3.13 (left), which can measure distances in the range of 0.03 to 11 m with a resolution of 0.01 m. The SRF10 provides a convenient way of connecting several such sensors over an Inter-Integrated Circuit (I^2C) bus, but the rather coarse 20-mm average resolution renders it almost impractical for performing satisfactory automatic altitude control. To get rid of this low-resolution problem, the SRF10 was replaced finally by a SensComp Mini AE URF (available at [90]), shown in Figure 3.13 (right). This sensor is a pure analog device, hence the resolution obtained is merely limited by the Analog-to-Digital Converter (ADC) used for converting the output signal. In addition to this 5-V analog result, the sensor provides a PWM-compatible clock signal that indicates the return of the echo. This signal is very useful for scheduling the conversion cycle, because the time required for an echo to return varies continuously with the measured distance. The Mini AE can be triggered externally with a TTL-compatible clock signal, which allows to vary the timed echoing arbitrarily in favor of higher sampling frequencies. The sensor is adjusted currently to provide distance data in the range of 0.1 to 3 m with a resolution of 3 mm. Longer distances of up to 6 m are possible, but at the expense of decreased precision and reliability.

3.4.2 Barometric Measurement Unit

Due to the rather modest altitudes that can be reached with ultrasonic sensing, a BMU, also known as *altimeter*, was developed entirely from scratch, aimed to co-operate with, or even eliminate, the URF device. The BMU, shown in Figure 3.14, is designed to match the size and drilling holes of the IMU, so that these two units can be mounted on each other (see Figure 3.15). Referring to the circuit, the BMU basically comprises of a Freescale Semiconductor MPXA6115A6U IPS (available at [91]), whose analog output signal is

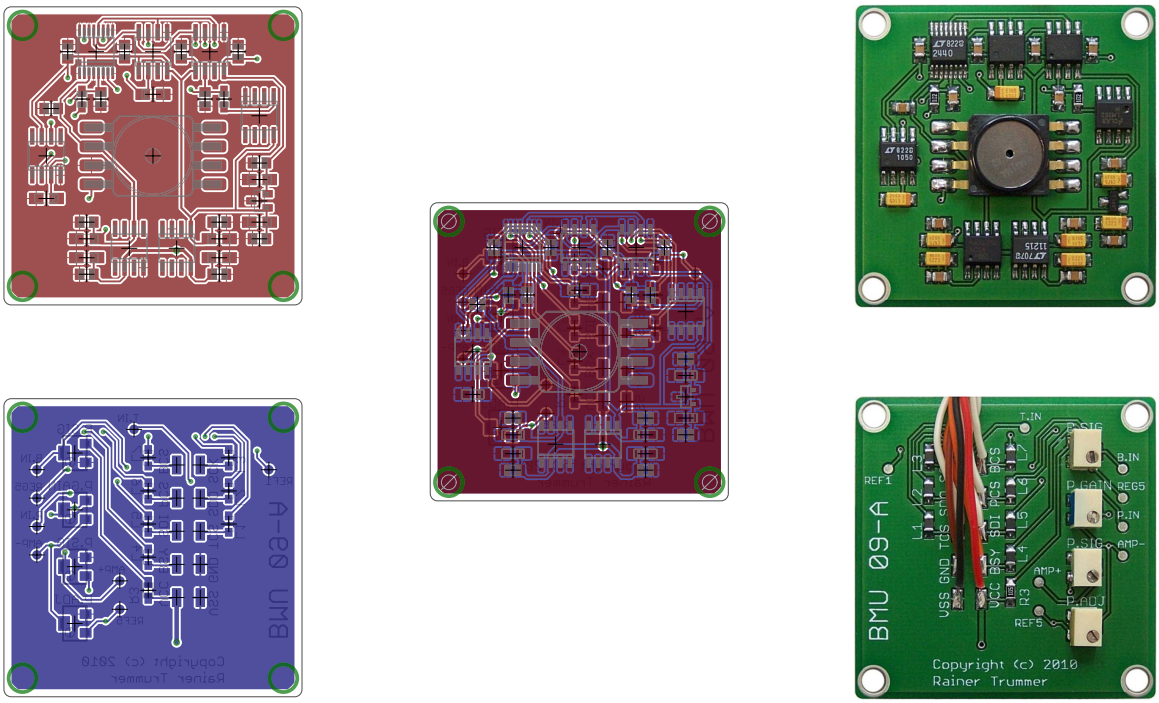


Figure 3.14: BMU Board PCB design (real size): top-layer layout (left top), bottom-layer layout (left bottom), complete two-layer layout (center), populated top side (right top), and populated bottom side (right bottom).

slightly filtered and then passed to a Linear Technology LTC1050CS8 precision zero-drift operational amplifier that is connected to a Linear Technology LTC2440CGN 24-bit high-speed delta-sigma ADC. This combination enables altitude measurements up to several thousand meters with a digital resolution down to millimeters, depending on the chosen sampling frequency. The conversion results are digitally temperature-compensated via measurements performed with a National Semiconductor LM35DM precision centigrade temperature sensor connected to a Maxim MAX1241BCSA 12-bit low-power ADC.

When exploiting pressure sensors for determining the altitude with a resolution of a few millimeters, this actually means to deal with signal changes in the range of nanovolts down to picovolts. In this context, extracting the desired information can be viewed as balancing between isolating the essential signal changes and amplifying the surrounding circuit's noise. The three "golden rules" for designing such sensitive circuits are as follows: (1) employ only high-quality components, (2) reduce the number of involved components to a minimum, and (3) keep the critical signal paths as short as possible. In accordance with rule (1), the 5-V reference signal supplied to the air-pressure-cognizant circuitry is provided by a Maxim MAX6250ACSA precision voltage reference with an accuracy of 2 ppm/V ($4.999 \leq 5.000 \leq 5.001$ V) for an input voltage of $10 \leq V_{IN} \leq 36$ V (note that the onboard battery has a nominal voltage of 14.8 V) and the 1.024-V reference signal supplied to the temperature-cognizant circuitry is provided by an Intersil ISL60002-10 precision voltage reference with an accuracy of 2 ppm/V ($1.023 \leq 1.024 \leq 1.025$ V) for

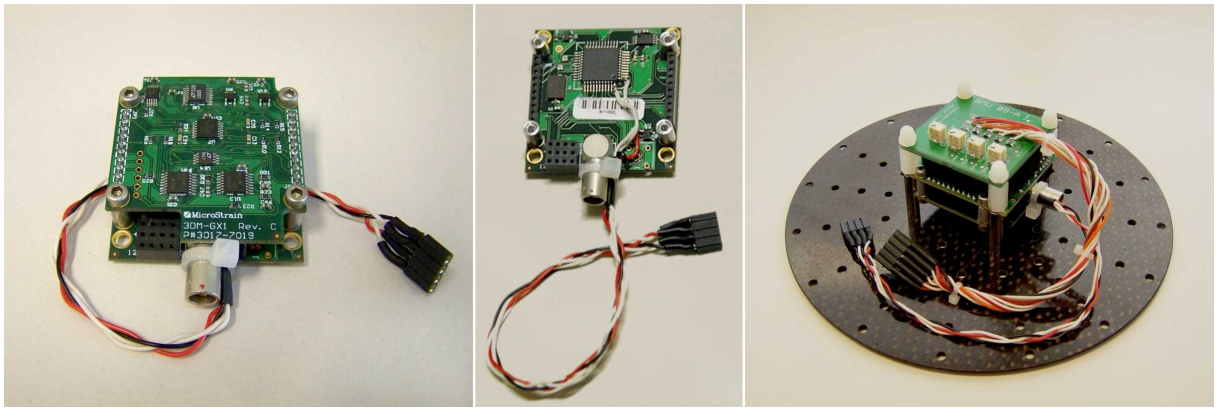


Figure 3.15: MicroStrain 3DM-GX1 IMU (left), modified 3DM-GX1 processor board (middle), and IMU-BMU stack mounted on top fuselage plate (right).

an input voltage of $2.7 \leq V_{IN} \leq 5.5$ V. Regarding rule (2), the reduction of comprising components, an LTC1050CS8 was chosen for amplification, because it contains internal sampling capacitors and therefore does not require an external sample-and-hold circuitry. In conforming to rule (3), addressing the short signal paths, it was attempted to keep the layout as compact as possible with the focus placed on avoiding any through connections in critical paths. (All components not cited are available at [82]).

The BMU provides a bi-directional Serial Peripheral Interface (SPI) that allows to configure the output rate of the 24-bit ADC used for the air pressure conversion. There are ten settings for the sampling rate, ranging from 6.9 Hz with an ultra-low noise value of 200 nV to 3.5 kHz with 25 μV noise. However, in the present case of converting an extremely amplified signal, sampling rates not exceeding the 110-Hz setting must be chosen for the centimeter range in order to obtain useful data, whereas the 27.5-Hz setting should be considered as the maximum for a resolution in the millimeter range. In addition to measuring the air pressure and temperature, the BMU contains a circuitry for generating 10-bit conversions of the battery voltage. This feature enables monitoring the voltage drop during flights, which is absolutely indispensable to prevent exhaustively discharging the lithium-polymer cells.

3.4.3 Inertial Measurement Unit

The JAviator's attitude with respect to the fixed Earth is determined with a MicroStrain 3DM-GX1 IMU (available at [92]), shown in Figure 3.15 (left). This orientation sensor contains angular-rate gyroscopes, accelerometers, and magnetometers for all three axes and provides triaxial orientation data with an accuracy of ± 2 % in several formats like Euler angles or quaternions, either gyro-stabilized or instantaneous. In particular, the data format exploited for the JAviator comprises of gyro-stabilized Euler angles, angular velocities, and linear accelerations. The Euler angles and angular velocities serve for attitude control, whereas the linear accelerations are fused by the state estimator with the altitude and location measurements for improved position control.

The 3DM-GX1 IMU provides a Radio Sector 232 (RS232) serial interface that is not supported one-to-one by the 5-V Universal Asynchronous Receiver Transmitters (UARTs) of the onboard computer system. In order to avoid an additional space- and energy-consuming RS232-to-TTL voltage level converter, the 3DM-GX1 was modified to provide direct access to its internal 5-V TTL serial interface. More precisely, the IMU's original port for RS232-based data transfer and power supply was bypassed via soldering wires directly to the microprocessor's RX and TX pin, to the voltage regulator's input pin, and to an unused ground pad, respectively, as shown in Figure 3.15 (middle). Another modification was to change the IMU's configuration settings referring to the internal clock to reduce the clock tick interval, and consequently, the time required for a complete calculation cycle. Due to this, the maximum output rate of 76.29 Hz (13.12 ms), which was achieved for calculating the gyro-stabilized Euler matrix in continuous mode, could be increased to 100 Hz, hence providing a new sample every 10 ms.

Onboard the JAviator, the IMU is mounted in stack-like fashion together with the BMU on the top fuselage plate, as shown in Figure 3.15 (right), which is installed with the sensor stack pointing downwards to be located protected inside the fuselage.

3.4.4 Stationary Localization System

Conventional GPS-based localization usually does not work within buildings, at least not sufficiently as required for performing automatic position control. Due to disagreeable weather dependencies and experimental equipment that increases over time, it is often desirable to have some indoor localization system that offers a characteristic similar to GPS technology. For this reason, a Ubisense Series 7000 SLS, shown in Figure 3.16 (left), is utilized for position control when experimenting indoors.

The standard setup consists of one or several active, battery-powered RFID tags, which send out pulses for determining their positions, leastwise four stationary antennas that analyze signals received from the tags, and a server that collects the location data from the antennas. More precisely, the tags transmit Ultra-Wide Band (UWB) impulses to the antennas, which use a number of different measurement methods to determine where the tags are located. The antennas are grouped into cells, which are typically of rectangular shape. In each cell, a master antenna manages the activities of the other antennas, communicates with all detected tags, and forwards the tags' locations to the server that runs a software called Ubisense Location Engine. This software evaluates, visualizes, and communicates the positional information to users and other systems. Our programs use the Application Programming Interface (API) of this software for receiving the tags' 3D position data. The Series 7000 system operates in a UWB range of 6 to 8 GHz for location sensing and utilizes the 2.4-GHz Integral Sliding Mode (ISM) radio band for data communication with the tags.

The Ubisense Location Engine delivers new measurements with a maximum update rate of 10 Hz. We were able to reproduce the claimed 3D positional accuracy of 0.15 m in the center of a cell of $6.0 \times 3.5 \times 2.7$ m with four antennas and a non-moving tag,

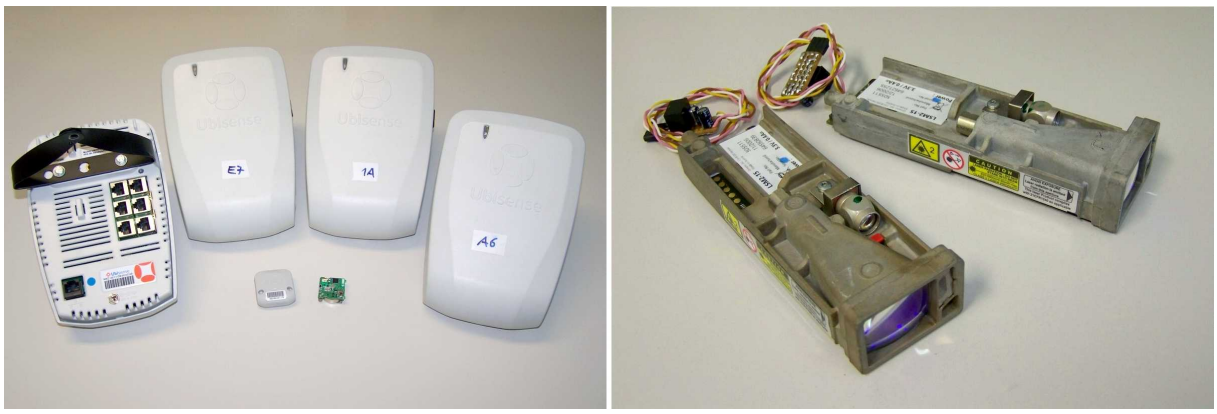


Figure 3.16: Ubisense Series 7000 SLS antennas (left), Ubisense Ubitag 7022 RFID tag with/without housing (left center), and Dimetix LSM2-15 LDS modules (right).

but accuracy lessens considerably outside the center and if the antennas' direct view to the tag is blurred. Nevertheless, we have successfully, autonomously flown a small toy helicopter, called JJ, with the Ubisense Series 7000 SLS (see [42]). In contrast to the JJ helicopter, which makes use of the full 3D position information, for the much larger and heavier JAviator, which demands more accurate vertical position data to accomplish automatic altitude control, only the 2D position information is exploited.

3.4.5 Laser Distance Sensor

The Dimetix LSM2-15 LDS (discontinued, follow-up model DLS-C15 available at [93]) is a high-precision industry laser sensor module. It is normally not available in the present form shown in Figure 3.16 (right), since it is commercially sold encapsulated in a quite huge and heavy aluminum housing that is sealed due to warranty issues (a contract with special warranty agreements had to be stipulated with Dimetix in order to get the bare laser modules). The LSM2-15 allows for measuring distances of up to 500 m with an accuracy of 1.5 mm when using a target plate. On natural surfaces, like the walls of a building, the maximum range is limited by the strength of the returned signal. However, in most cases measurements of up to 150 m can be achieved without loss of precision. It provides a serial interface, is powered with a supply voltage of 3.3 V, and consumes a maximum of 250 mA when operated in the fast tracking mode.

Compared to the relatively short ranges offered by small URFs, the LSM2-15 LDS enables high-precision surveying within a radius of 150 m. The downside of this sensor is that even in the fast tracking mode a sampling frequency of 7 Hz on the average is rarely exceeded. Despite this drawback, one would not expect that permanent perturbations during flight, such as vibrations and fast movements, do not cause the sampling rate to drop significantly. It was planned originally to mount two LSM2-15 sensors on top of the JAviator facing in opposite directions with the aim to facilitate surveying where conventional surveying instruments cannot be applied. However, due to the LSM2-15's unexpected impassivity with respect to perturbations, the two sensors are meanwhile

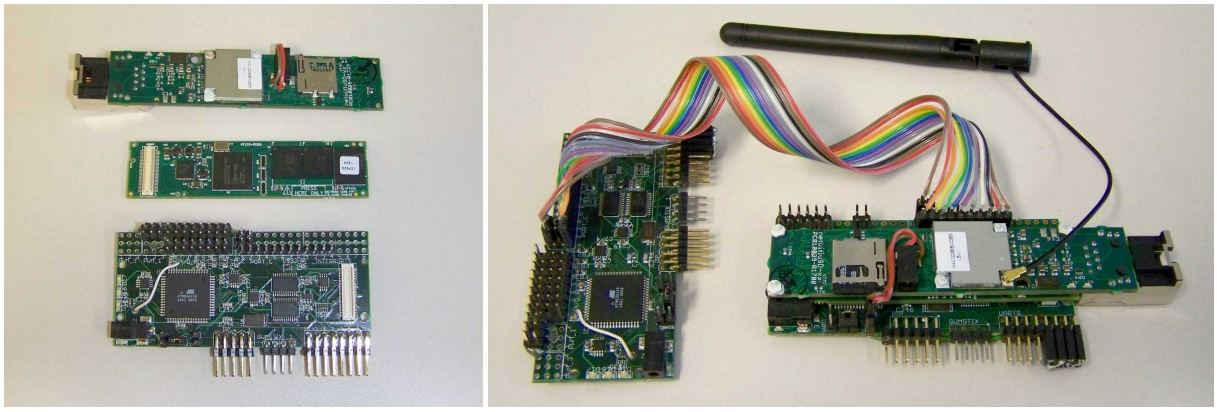


Figure 3.17: Gumstix Netpro-VX (left top), Gumstix Verdex Pro XL6P (left center) Gumstix Robostix (left bottom), and JAviator onboard computer system (right).

mounted in an orthogonal arrangement, because it turned out that they can be utilized for position control in indoor environments to some extent. When using the introduced RFID-based SLS for position control, the two LDSs may serve for obtaining precise reference data, hence allowing to determine the true error in the SLS measurements.

3.5 Computer System

The computer system applied for flying the JAviator can be divided into two subsystems: first, a distributed system of embedded devices mounted inside the JAviator's fuselage, which we call JAviator Avionics, and second, a distributed system encompassing various stationary devices, which we call Ground Station, for communicating with the onboard computers. The computational components of these two systems and their interconnection are described in more detail in the following subsections.

3.5.1 JAviator Avionics

The JAviator is equipped with a set of embedded computational devices that serve as the onboard computer system. In particular, the first device is a Gumstix Verdex Pro XL6P computer-on-module, shown in Figure 3.17 (left center), which features an Intel XScale PXA270 Central Processing Unit (CPU) clocked at 600 MHz, a 128-MB Random Access Memory (RAM), and a 32-MB flash memory, but no Floating-Point Unit (FPU). The operating system on this module is a Linux system running a kernel version with real-time extensions and support of high-resolution timers, which we modified to work with the Gumstix Verdex. The second device is a Gumstix Robostix expansion board, shown in Figure 3.17 (left bottom), which contains an Atmel ATmega128 processor clocked at 16 MHz. This board provides the necessary communication interfaces for connecting to the sensors and the required PWM units for driving the four motors. Because there are more sensors involved in the JAviator control system than can be handled by a single

Robostix, there are two such boards incorporated, which are connected via an emulated 8-bit parallel interface (described in Subsection 3.5.3). The third device is a Gumstix Netpro-VX expansion board, shown in Figure 3.17 (left top), which provides the required wireless Local Area Network (LAN) connectivity to the Ground Station. This module also provides an Ethernet-based LAN connection that we use for debugging. (All embedded devices of this subsection are available at [94]).

The Netpro-VX can be stacked onto the Verdex to be interconnected by an 80-pin Hirose Input/Output (I/O) connector. This combination can further be stacked onto the Robostix to be interconnected by a 60-pin Hirose I/O connector. In this way, a very compact and powerful computer stack is obtained, which is shown in Figure 3.17 (right) together with the second Robostix connected via the emulated parallel interface.

3.5.2 Ground Station

The Ground Station comprises of an IBM ThinkPad T60p laptop computer that runs our Control Terminal software, a Lenovo IdeaPad S10 laptop computer that runs the Ubisense Location Engine, and in case of performing Java-based flights, a separate logging station that collects trace data (time-stamped values of program variables) received from the onboard computer system. Except for the IdeaPad S10 that needs to run Windows XP to execute the Location Engine, all other ground computers run some Linux distribution. All of the ground computers are interconnected by an encapsulated LAN and each of them is connected to the onboard computers by a separate wireless LAN channel. The ThinkPad T60p executing the Control Terminal application is also connected to a Logitech Freedom 2.4 4-axis joystick used for manually piloting the JAviator.

3.5.3 Interconnectivity

When it comes to embedded systems, in particular when dealing with microcontrollers, there are usually always too little ports available. Albeit most microcontrollers provide either or both an I^2C and an SPI interface, which allow for interconnecting several devices over a shared bus, there are many sensors and other devices that demand some RS232-compatible or TTL-compatible UART interface. The XScale PXA270 processor on the Verdex board and the ATmega128 processor on the Robostix board both provide two UART interfaces. In contrast to the Robostix, where most of the ATmega128 ports are accessible via pin connectors, the XScale PXA270 ports of the Verdex can only be accessed via the 60-pin Hirose I/O connector. In case of stacking the Verdex onto the Robostix, the two UARTs of the Verdex become accessible via pin connectors located on the Robostix. Accordingly, a Verdex-Robostix combination provides a total of four UART ports. However, as can be seen in Figure 3.18, which depicts the interconnectivity of the JAviator Avionics (blue) and the Ground Station (green), there are currently three sensors involved, particularly the IMU and two LDSs, which are connected via UARTs. Regarding the Verdex, one UART port is required for communicating with the Robostix, which also

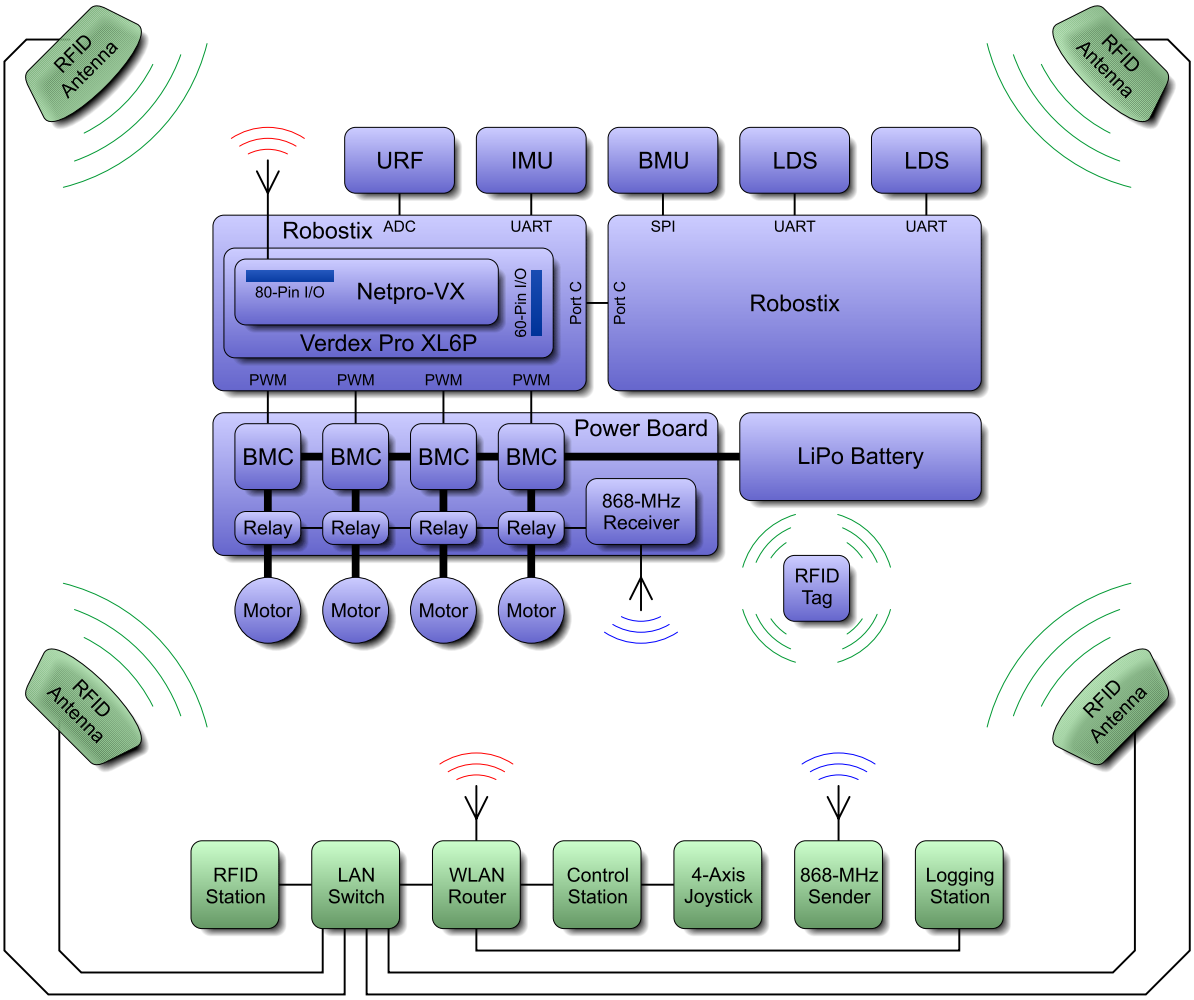


Figure 3.18: Interconnectivity of JAviator Avionics (blue) and Ground Station (green).

needs to spend one of its UARTs for this purpose. The second UART of the Verdex board is reserved for the console input, because this is the only way of communicating with the Verdex in case of any malfunctions that might occur in the operating system or the Ethernet-based connectivity.

Besides a UART channel connecting the Verdex to the Robostix, which needs to be enabled externally with a bridge connector, these devices can also make use of an SPI connection, which is already established physically by the 60-pin Hirose I/O connector. However, this approach raises three significant drawbacks. First, one of the three sensors requiring a UART port needs to be connected directly to the Verdex board, and thus, to the control software. In over words, data acquisition processes would not be completely decoupled as intended, because the control software would have to take care of driving this sensor. Second, compared to the *asynchronous* UART communication, the obligatory *synchronous* communication demanded by the SPI protocol is far more troublesome to handle and becomes less reliable as the size of transferred data increases. Third, in the preset hard-wired SPI configuration the Verdex always acts as the "master" and the

Robostix as the "slave". This means that all SPI-connected devices communicate through the Robostix board but cannot be controlled by the Robostix software.

To this end, the approach of integrating a second Robostix for extending the number of available UARTs was straightforward in order to cope with the problem of missing ports. Obviously, the two Robostix boards need to be connected by making use of any other port than their UARTs, since otherwise no additional UART port would be gained. Applying SPI for connecting these two boards turned out to be cumbersome for two reasons: first, SPI is already in use to communicate with three ADCs that are part of the BMU, which complicates the synchronization of data transfers, and second, there is also some unpredictable asynchronous communication of control commands, which can cause interceptions in the regular transfer. I^2C is an option, but can be considered troublesome and unreliable when it comes to transferring large amounts of data. In searching for an appropriate solution, finally the idea of implementing an 8-bit parallel port was born. Because there are many unused general-purpose I/O pins, this type of connection could actually be realized easily. Specifically, to transfer 8-bit data words between two Robostix boards in parallel, we use an 8-pin I/O port for the data and two configurable interrupt pins for the required control signals in order to implement an emulated 8-bit parallel interface. In this way, a fast and reliable connection is accomplished that is controlled solely via interrupt-triggered software.

Although there are only four of the six PWM ports of the first Robostix in use for driving the motors, the remaining two cannot be exploited for conventional RC servos. This is due to the configuration of the PWM units for the Fast-PWM mode (250 Hz in the present case), which is incompatible to standard RC devices requiring a phase- and frequency-correct 50-Hz PWM signal. However, an advantageous side effect that came with the second Robostix is the availability of six additional PWM ports that can be utilized arbitrarily for laser and/or camera positioning servos.

Chapter 4

Control System Design

Of course the word 'chaos' is used in a rather vague sense by a lot of writers, but in physics it means a particular phenomenon, namely that in a nonlinear system the outcome is often indefinitely, arbitrarily sensitive to tiny changes in the initial condition.

Murray Gell-Mann (1929–)

The second major chapter conveys extensive insight into the engineering process that cumulated in the present JAviator control system. In the preliminary Section 4.1, the two coordinate systems that must be dealt with are discussed, followed by a formalization of certain coordinate transformations that allow to arbitrarily change from one system to the other. Thereafter, the applied control principle and accompanying control system model is introduced. This model is aimed to serve as guideline for subsequent sections that are dedicated to specific control system components, beginning with Section 4.2 describing the quadrotor plant from the perspective of available sensor data. The different types of digital filters incorporated for improving the data quality are covered in Section 4.3. The fundamental principle of inertial navigation is discussed in Section 4.4, followed by a detailed formalization of the employed state estimator, which can be considered as the most essential component of the control system. This section also shades some light on computational issues that should be taken into account when it comes to implementation. Section 4.5 concludes with a thorough description of the motion control, which comprises of several controllers for providing inertial and positional stability.

4.1 Preliminaries

In contrast to vehicles that operate *on* land or water, like cars and ships, respectively, which – under normal conditions – are bound to the surface in regard to their attitude, vehicles designed to operate *in* space or water, like aircraft and submarines, respectively, usually have the ability to take on *any* desired attitude independently of the environment.

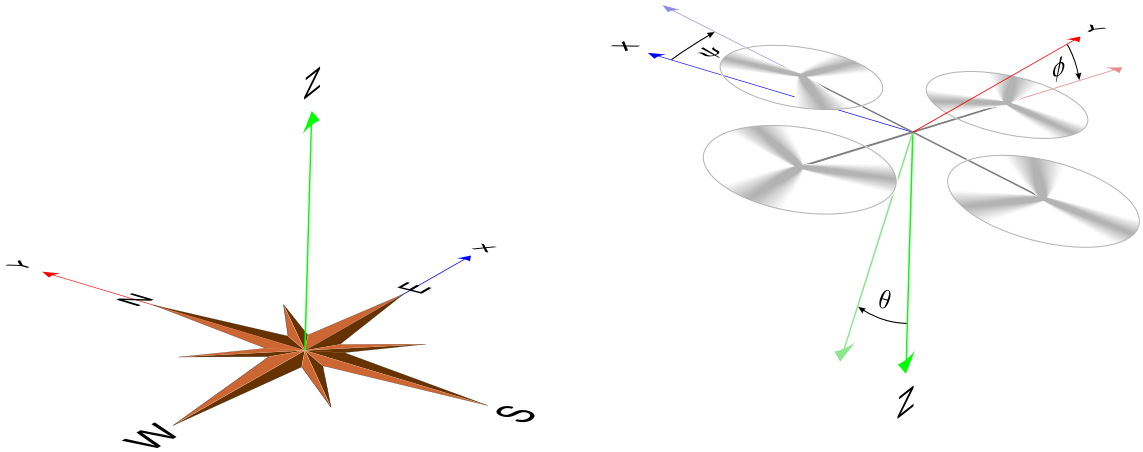


Figure 4.1: Orientation in the earth-fixed X-Y-Z Cartesian coordinate system (left) and orientation in the body-fixed Z-Y-X aircraft coordinate system (right).

Due to this, as soon as an aircraft lifts off, or similarly, a submarine descents, and by doing so starts to change its original attitude, the craft no longer conforms to the coordinate frame of the fixed Earth. As a consequence, any sensor data that are influenced by the craft's attitude, for instance, gyroscope and accelerometer measurements as well as laser and ultrasonic measurements, must be transformed from the craft's *body-fixed* frame to the *earth-fixed* frame before being passed to a system that expects earth-fixed data. (For a detailed description of inertial navigation principles see, e.g., [95]).

4.1.1 Aircraft Orientation

Regarding an aircraft's orientation in space, the two above mentioned "frames", or strictly speaking, coordinate systems, are depicted for comparison in Figure 4.1. The left-hand part of the figure illustrates the orientation in the earth-fixed, or *global*, X-Y-Z Cartesian coordinate system, whereas the right-hand part of the figure illustrates the orientation in the body-fixed, or *local*, Z-Y-X aircraft coordinate system.

According to Euler's rotation theorem, any possible rotation of a rigid body in the 3D Euclidian space can be described by a set of three angles. Referring to the most prevalent convention for representing an aircraft's attitude with respect to the fixed Earth, these angles are defined as three Euler angles. More precisely, given body-fixed coordinate axes X, Y, Z describing the craft's orientation in space, the three Euler angles, their common denotations, and ranges are specified as follows:

$$\begin{array}{lll} \text{Rotation by } \phi \text{ around } X: & \boldsymbol{Roll} & (-\pi \leq \phi \leq \pi) \\ \text{Rotation by } \theta \text{ around } Y: & \boldsymbol{Pitch} & (-\frac{\pi}{2} < \theta < \frac{\pi}{2}) \\ \text{Rotation by } \psi \text{ around } Z: & \boldsymbol{Yaw} & (-\pi \leq \psi \leq \pi) \end{array}$$

Note that the pitch angle (θ) is restricted to the range $-\pi/2 < \theta < \pi/2$, because in the case of $\theta = \pm\pi/2$ both the roll angle (ϕ) and the yaw angle (ψ) become magnitudes of angles about parallel axes, and consequently, one degree of rotational freedom is lost. This

effect is generally referred to as the *gimbal lock*, which arises, for example, in a mechanical system with three gimbals when the inner gimbal is rotated through $\pm\pi/2$.

4.1.2 Coordinate Transformations

Most of the data measured by the onboard sensor suite cannot be used directly as input for the controllers since they represent body-fixed magnitudes, and therefore, need to be transformed to earth-fixed magnitudes first. For this purpose, we make again use of Euler's rotation theorem, which further states that for a rigid body in the 3D Euclidian space any body-fixed coordinates can be transformed to earth-fixed coordinates via three consecutive rotations by the three angles describing the body's attitude. Accordingly, given an attitude vector $\boldsymbol{\varphi} = [\,\phi\,\theta\,\psi\,]^T$ and three rotation matrices $\boldsymbol{R}_X(\phi), \boldsymbol{R}_Y(\theta), \boldsymbol{R}_Z(\psi)$ defining rotations by ϕ, θ, ψ around X, Y, Z, respectively, a rotation matrix, called the Direct Cosine Matrix (DCM), for transforming body-fixed vehicle coordinates to earth-fixed vehicle coordinates can be obtained as follows:

$$
\begin{aligned}
\boldsymbol{R}_{b\to e}(\boldsymbol{\varphi}) &= \boldsymbol{R}_Z(\psi)\cdot\boldsymbol{R}_Y(\theta)\cdot\boldsymbol{R}_X(\phi) \\
&= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \\
&= \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}
\end{aligned}
$$

Note that the three consecutive rotations $\boldsymbol{R}_X(\phi), \boldsymbol{R}_Y(\theta), \boldsymbol{R}_Z(\psi)$ are not commutative. That is, the sequence of rotations is solely determined by the order of the underlying body-fixed coordinate system. Consequently, in case of Z-Y-X aircraft coordinates the rotations must be accomplished in ψ-θ-ϕ order, thus implying the Z-Y-X order.

Unfortunately, the DCM cannot be used for transforming angular magnitudes, because they are directly related to the angles as they represent angular changes with respect to time. In particular, the angular velocities $(\omega_x, \omega_y, \omega_z)$ provided by the IMU are physically measured with gyroscopes and do not conform to the Euler angle derivatives $(\dot{\phi}, \dot{\theta}, \dot{\psi})$, which are discontinuous in regard to the time variation of the Euler angles. Therefore, a different rotation matrix must be constructed for transforming angular velocities from body-fixed coordinates to earth-fixed coordinates. Besides many ways to accomplish this, the probably most comprehensible one, but not necessarily simplest one, is outlined here (for a completely different approach see, e.g., [95]). In this sense, given an attitude vector $\boldsymbol{\varphi} = [\,\phi\,\theta\,\psi\,]^T$, three rotation matrices $\boldsymbol{R}_X(\phi), \boldsymbol{R}_Y(\theta), \boldsymbol{R}_Z(\psi)$ defining rotations by ϕ, θ, ψ around X, Y, Z, respectively, and $\boldsymbol{J}$ denoting an unknown Jacobian matrix, a rotation matrix for transforming body-fixed angular velocities to earth-fixed angular velocities can

be obtained as follows:

$$\begin{aligned}
\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}_X(\phi)^T \cdot \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}_X(\phi)^T \cdot \boldsymbol{R}_Y(\theta)^T \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \\
&= \begin{bmatrix} \dot{\phi} - \dot{\psi}\sin\theta \\ \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta \\ -\dot{\theta}\sin\phi + \dot{\psi}\cos\phi\cos\theta \end{bmatrix} \equiv \boldsymbol{J}^{-1} \cdot \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}
\end{aligned}$$

Solving for the inverse Jacobian matrix $\boldsymbol{J}^{-1}$ yields:

$$\begin{aligned}
\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} &= \boldsymbol{J} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \\
&= \begin{bmatrix} 1 & \sin\phi\sin\theta/\cos\theta & \cos\phi\sin\theta/\cos\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}
\end{aligned}$$

Hence, the desired rotation matrix is given by:

$$\boldsymbol{R}_{\omega\to\dot{\varphi}}(\boldsymbol{\varphi}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}$$

Remark 5. For transforming earth-fixed to body-fixed vehicle coordinates, one can apply the transposed DCM $\boldsymbol{R}_{b\to e}(\boldsymbol{\varphi})^T$. Similarly, for transforming earth-fixed to body-fixed angular velocities, one can apply the transposed rotation matrix $\boldsymbol{R}_{\omega\to\dot{\varphi}}(\boldsymbol{\varphi})^T$.

4.1.3 Applied Control Principle

The control principle inherent in the JAviator control system characterizes a so-called *closed-loop state space design*, which, in the present case, is based on state estimation, a feature that can be seen as the most essential property of the system. Figure 4.2 depicts the resulting control system model that separates the system into four major components: (1), the quadrotor plant providing inertial and positional measurements, (2), digital filters improving the data quality, (3), the state estimator generating estimates for data that are not available periodically, like low-frequency position measurements, as well as data that are not available at all, like linear velocities, and (4), the motion control generating the control signals assigned to the PWM units. (Each of these major components is explained thoroughly in one of the next four sections). As apparent from the figure, the reference input ($\boldsymbol{r}$) is directly passed to the motion control and entirely bypasses the state estimator, an approach that can be found in most of today's modern state space designs. Contrariwise, there exist also control systems where the reference input is

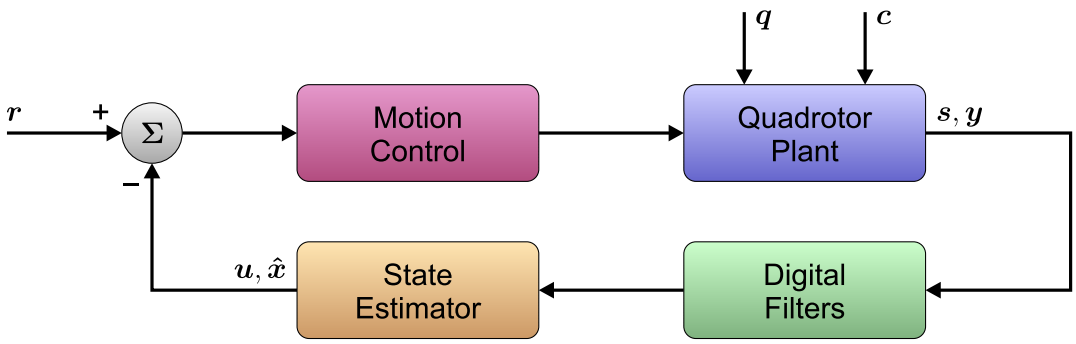


Figure 4.2: Model of the JAviator control system representing a closed-loop state space design with the reference input entirely bypassing the state estimator.

supplied *additionally* and sometimes *only* to the state estimator (for an overview of the many different approaches see, e.g., [96, 97]).

The control cycle starts with the quadrotor plant, which provides the essential sensor data, particularly, gyroscope and accelerometer measurements ($\boldsymbol{s}$) that are influenced by process noise ($\boldsymbol{q}$) and the control-output, or *observation*, vector ($\boldsymbol{y}$) that is influenced by observation noise ($\boldsymbol{c}$). Thereafter, some of the data pass the digital filters before being supplied to the state estimator, which is sometimes called *state observer*, since it utilizes the observations provided by the control-output vector ($\boldsymbol{y}$), which represents the craft's true pose, to correct the estimated values. In the scope of this process, the gyroscope and accelerometer measurements ($\boldsymbol{s}$) are transformed to the control-input vector ($\boldsymbol{u}$), which is then exploited to generate the state vector ($\hat{\boldsymbol{x}}$) that contains the estimates. Lastly, in addition to the reference-input vector ($\boldsymbol{r}$), both the control-input vector ($\boldsymbol{u}$) and the state vector ($\hat{\boldsymbol{x}}$) are supplied to the motion control.

Remark 6. For the purpose to distinguish an *estimated* value from a *true* value, we follow the common statistics convention and denote an estimate with a "ˆ" symbol.

4.2 Quadrotor Plant

The quadrotor plant is the main source for providing sensor data and, according to the control system model introduced in the previous section, also treated as the *only* source of sensor data. Strictly speaking, this is not true to some extent, because there are also sensors involved that are located externally to the plant. In case of the JAviator, the SLS uses RFID antennas that are placed at fixed environmental points, and due to this, requires the plant merely to carry an active RFID tag, but no real position sensor. Consequently, the positional information is also delivered by an external system and forwarded to the control system, but not sensed onboard the plant. Needless to say that this is of course the case whenever an external sensing system, for instance, vision-based motion capturing, is employed.

4.2.1 Sensor Measurements

The source of inertial and positional measurements can be divided basically into two distinct sensor suites: first, the essential sensor instrumentation required for flying the JAviator manually, and second, the auxiliary sensor instrumentation deployed for position control and/or other control-independent applications (for a detailed description of each sensor see Section 3.4). When incorporating the complete sensor equipment, the following sequences of sampled data can be obtained:

$$\begin{aligned} IMU &: (\phi, \theta, \psi,\ \omega_x, \omega_y, \omega_z,\ a_x, a_y, a_z) \\ SLS &: (x, y) \\ LDS &: (x, y) \\ URF &: (z) \\ BMU &: (z) \end{aligned}$$

where IMU contains 3D Euler angles (ϕ, θ, ψ), *body*-fixed angular velocities $(\omega_x, \omega_y, \omega_z)$, and *body*-fixed linear accelerations (a_x, a_y, a_z); SLS contains *earth*-fixed 2D position data (x, y), whereas LDS contains *body*-fixed 2D position data (x, y); URF contains *body*-fixed 1D position data (z), whereas BMU contains *earth*-fixed 1D position data (z).

4.2.2 Sampling Characteristics

All of the involved sensors have either a *constant* or *variable* sampling time, and thus, can be driven to deliver their measurements either *synchronously* or *asynchronously* to the control system's sampling period,[1] which is core to the entire control process. Formally, given a *continuous-time* control system and $t \in \mathbb{R}^+$ denoting a time instant, the sampling period, denoted by Δt, is defined as the interval between two consecutive iteration cycles, such that $\Delta t = t_k - t_{k-1}$ $\{k \in \mathbb{N} \,|\, 0 < k < \infty\}$. In contrast to this definition, when given a *discrete-time* control system, the sampling period is usually defined as a constant value and some timing mechanism is incorporated to ensure that for any k satisfying $0 < k < \infty$ the relation $t_k - t_{k-1} \equiv \Delta t$ holds.

In addition to the general sensor properties and accuracy characteristics described in Section 3.4, the specific sampling characteristics, which are of vast importance in regard to timing issues, can be summarized as given below.

Inertial Measurement Unit. The IMU is the fastest of all introduced sensors and also facilitates a constant sampling time of $t_{IMU} = 10$ ms. Due to this, the control system's sampling period is always chosen to satisfy $\Delta t > t_{IMU}$, which allows for generating a new sequence of inertial measurements concurrently to each control cycle. Note that this sensor comes with a factory setting of 13.12 ms and must be re-programmed in order to enable a sampling time of 10 ms (see Subsection 3.4.3).

[1] The *sampling period* is sometimes also called *control loop period* or simply *control period*. For the remainder of this thesis, the more commonly used term *sampling period* will be used.

Stationary Localization System. The SLS is a relatively slow sensing system but facilitates constant sampling times. They can be chosen via selecting an integer number n representing a power of 2 in the range $4, ..., 8192$, which indicates an update is desired every nth time slot. The slots have a constant period of 27.19 ms, which translates to sampling times of $t_{SLS} = 108.76, ..., 222740.48$ ms.

Laser Distance Sensors. The LDSs provide much more accurate data than the SLS, but are the slowest devices and, depending on the quality of the reflected laser beams, need variable amounts of time to perform measurements. This results in sampling times varying in the range of $t_{SLS} \sim 140, ..., 200$ ms, can approach $t_{SLS} \to \infty$ with decaying signal strength, and reach $t_{SLS} = \infty$ in case of entire signal loss.

Ultrasonic Range Finder. The URF also needs a variable amount of time for every measurement, since it must wait for the ultrasonic echo to return. This leads to sampling times varying from $t_{URF} \sim 0.8$ ms (for the minimum measurable distance of 0.13 m) to $t_{URF} \sim 35$ ms (for the maximum measurable distance of 6 m), and similar to the LDSs, can approach $t_{URF} \to \infty$ with decaying signal strength and reach $t_{URF} = \infty$ in case of entire signal loss.

Barometric Measurement Unit. The BMU is on the average slower than the URF, but facilitates constant sampling times, which can be selected via software, spanning from $t_{BMU} = 9.09$ ms (for a minimum resolution in the centimeter range) to $t_{BMU} = 145.45$ ms (for a maximum resolution in the millimeter range).

As apparent from these characteristics, in contrast to the inertial data that are available with each sampling period, all of the positional data are received asynchronously and most of them at much lower frequencies.

4.3 Digital Filters

Some of the data sequences received from the sensors contain serious outliers, some of them resemble a series of stepwise changes, and others are much too noisy to be useful at all. Concerning these inconveniences, a specific set of appropriate filters may sometimes significantly improve the signal quality. In this sense, digital filters play an important role in the JAviator control system and they are one of the keys that led to success.

4.3.1 DT Outlier Filter

It turned out that all of the positional data contain outliers frequently, a circumstance that seems to come along with position measurements independently of the sensors used. Particularly, concerning the SLS and the BMU, large outliers are primarily due to signal interferences and resulting measuring errors, whereas concerning the LDSs and the URF,

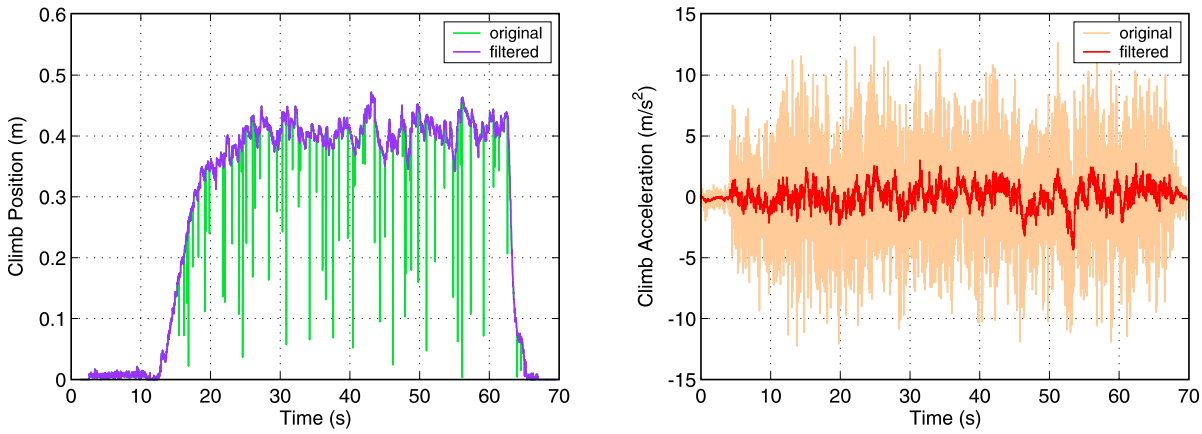


Figure 4.3: Performance of DT outlier filter (left) applied to z position measurements and performance of IIR low-pass filter (right) applied to a_z acceleration measurements.

large outliers are usually caused by gaps and bumps in the surrounding surfaces that are unintentionally tracked by the laser beams and ultrasonic pulses. In order to cope with this problem, a rather simple but very effective Differential Threshold (DT) outlier filter was implemented. This filter detects outliers by comparing the difference between two consecutive measurements against a constant threshold. For the case that some measurement identified as an outlier actually represents a valid value, the discard rate is bounded by an upper limit, which defines the maximum number of values that may be discarded. Without this "discard limit", one can easily imagine situations where no more valid measurement would pass the filter, for example, if a sudden change in altitude happens concurrently to URF perturbations caused by unexpected gaps on the tracked ground. Clearly, it is possible to deal with such scenarios by supplying the filter with acceleration feedback, which would allow to distinguish between "true" vehicle movements and "false" ones caused by perturbed measurements. However, it turned out that most of the outliers represent values that are ways larger than the average of differential changes, and due to this, a constant differential threshold, which was determined experimentally, works reasonably well. Given a series of measurements $x_0, ..., x_n$ $\{x_k \in \mathbb{R} \mid 1 \leq k \leq n\}$, the implemented DT outlier filter can be formalized as follows:

$$f_{DTO}(x_{k-1}, c_{k-1}, x_k) \quad = \quad \begin{bmatrix} x_{k-1} \ \text{ if } |x_k - x_{k-1}| > a \ \wedge \ c_{k-1} < b, \text{ else } x_k \\ c_{k-1} + 1 \ \text{ if } |x_k - x_{k-1}| > a \ \wedge \ c_{k-1} < b, \text{ else } 0 \end{bmatrix}$$

where a $\{a \in \mathbb{R} \mid 0 < a < \infty\}$ and b $\{b \in \mathbb{N} \mid 0 < b < n\}$ are constants denoting the differential threshold and the discard limit, respectively, and c $\{c \in \mathbb{N} \mid 0 \leq c \leq b\}$ is a variable denoting the number of consecutive outliers.

Figure 4.3 (left) shows a trace of position (z) measurements taken during an outdoor flight conducted with the aim to test the DT outlier filter under "extreme" conditions for the URF. In particular, the trace depicts the original ultrasonic measurements and demonstrates the filter's behavior when flying over a lawn scattered with flowers.

4.3.2 FIR Low-Pass Filter

The sampled air pressure data received from the BMU are provided as raw data that come with considerable noise, and consequently, need to be smoothed before they can be converted to meaningful altitude data. To this end, a first-order low-pass filter could be used and a median filter would also be an option. However, in contrast to the IMU measurements that are available with each sampling period of the control system, the air pressure measurements require longer sampling times, especially if the BMU is set to the highest possible resolution. In other words, the altitude data are already extremely delayed relative to the sampling period and the incorporation of a first-order low-pass filter or a median filter would induce an additional delay, which, depending on the filter gain, could be quite large. Due to this, smoothing the air pressure data is accomplished with a Finite Impulse Response (FIR) low-pass filter, also known as *second-order* low-pass filter. This filter involves only a *finite* measurement history and therefore, if adapted appropriately, causes smaller delays without significant performance degradation. Given a series of measurements $x_0, ..., x_n$ $\{x_k \in \mathbb{R} \mid 2 \leq k \leq n\}$, the implemented FIR low-pass filter can be formalized as follows:

$$f_{FIR}(x_{k-2}, x_{k-1}, x_k) \quad = \quad \tfrac{1-a}{2} x_{k-2} + a x_{k-1} + \tfrac{1-a}{2} x_k$$

where a $\{a \in \mathbb{R} \mid 0 < a < 1\}$ is a constant denoting the filter gain.

4.3.3 IIR Low-Pass Filter

The linear accelerations, which are by nature extremely noisy, are provided by the IMU as raw sensor data that need to be smoothed in order to attain useful signals. Contrariwise, the reference commands, if generated with the joystick or the keyboard, tend to form a series of stepwise changes, which are primarily a reflection of the human reaction time. Both the noisy acceleration measurements and the discontinuous commands are updated with each sampling period, which allows for applying an Infinite Impulse Response (IIR) low-pass filter, also known as *first-order* low-pass filter. This filter is based on involving the complete, *infinite* measurement history, and as a consequence, causes larger delays but dominates over the FIR low-pass filter when seen from the perspective of performance. Given a series of measurements $x_0, ..., x_n$ $\{x_k \in \mathbb{R} \mid 1 \leq k \leq n\}$, the implemented IIR low-pass filter can be formalized as follows:

$$f_{IIR}(x_{k-1}, x_k) \quad = \quad x_{k-1} + a(x_k - x_{k-1})$$

where a $\{a \in \mathbb{R} \mid 0 < a < 1\}$ is a constant denoting the filter gain.

Figure 4.3 (right) shows a trace of acceleration (a_z) measurements taken during the aforementioned flight over the flower field and demonstrates the behavior of the IIR low-pass filter when applied to a series of noisy data. Note that the original acceleration data do not contain any noticeable, outstanding peaks that would indicate a sudden vertical movement of the craft. This is due to the fact that all outliers contained in the original position (z) data have been removed successfully by the DT outlier filter.

4.4 State Estimator

The state estimator can be seen as the most essential component of the JAviator control system, because it fuses inertial and positional measurements from different sensors in order to estimate the craft's complete pose. This process also includes the generation of estimates for the missing linear velocities, which are not available directly from any of the sensors. New estimates are generated with every sampling period and are incorporated recursively in further estimation cycles to provide more accurate state information with each measurement update available.

The fundamental vectors spanning the craft's 6-Degree of Freedom (DOF) state space, which have been slightly ripped in Subsection 4.1.3, are specified as follows:

$$\boldsymbol{\varphi} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix},$$

where $\boldsymbol{\varphi}$ represents the craft's attitude, $\boldsymbol{\omega}$ its angular velocity, $\boldsymbol{\alpha}$ its angular acceleration, $\boldsymbol{p}$ its position, $\boldsymbol{v}$ its linear velocity, and $\boldsymbol{a}$ its linear acceleration. Given the craft's 6-DOF state space, the extended estimator state space is specified as follows:

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\alpha} \\ \boldsymbol{a} \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{\varphi} \\ \boldsymbol{p} \\ \boldsymbol{v} \end{bmatrix}, \quad \boldsymbol{q} = \begin{bmatrix} \sigma_\omega \\ \sigma_a \\ \sigma_a \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} \sigma_\varphi \\ \sigma_p \\ \sigma_v \end{bmatrix},$$

where $\boldsymbol{u}$ is the control-input vector referring to data that serve as input for the estimator and the motion control, $\boldsymbol{x}$ is the state vector describing the craft's pose, $\boldsymbol{y}$ is the control-output vector referring to observations that serve as correctors for the estimates, $\boldsymbol{q}$ and $\boldsymbol{c}$ are Gaussian, zero-mean, uncorrelated, white process and observation noise, respectively, and $\sigma_{\{\omega,a,\varphi,p,v\}} \sim \mathcal{N}\big([\,0\,0\,0\,]^T, \Sigma_{\{\omega,a,\varphi,p,v\}}\big)$ refers to the standard data variances.

Remark 7. σ_ω and σ_a that represent the gyroscope and accelerometer noise, respectively, were determined experimentally by computing the average deviations over a period of 30 min. σ_φ and σ_p that represent the attitude and position noise, respectively, were determined empirically based on the discovered sensor inaccuracies. σ_v that represents the velocity noise was added to extend the bandwidth of filter tuning.

4.4.1 Estimation Principle

Referring to the JAviator control system, the process of state estimation, based on fusing inertial and positional measurements, can be separated into two main objectives: first, "improving the existing" information, and second, "estimating the missing" information. More precisely, in the first case, the intention is to improve the existing attitude and position data by generating estimates that more closely reflect the actual state of the craft. In the second case, the intention is to enhance the state information by generating estimates for the missing velocity data. This process is illustrated in Figure 4.4, which

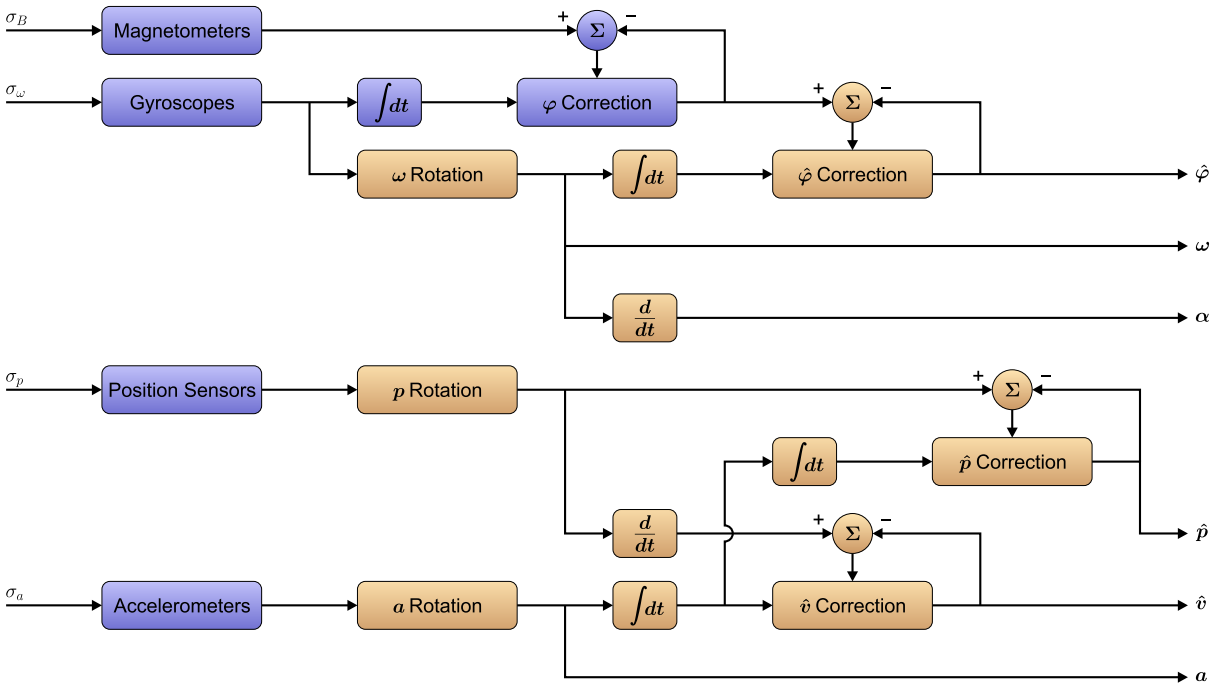


Figure 4.4: Model of the state estimation process comprising of inertial and positional sensing components (blue) referring to the quadrotor plant and computational components (yellow) referring to the control system's state estimator.

depicts the state estimation model comprising of various inertial and positional sensing components (blue) referring to the quadrotor plant as well as computational components (yellow) referring to the state estimator. As can be seen from the blue items in the figure, there is also an IMU-internal estimator involved that fuses the magnetometer and gyroscope measurements. The purpose of this estimator, which is implemented by a Kalman Filter (KF), is, first, to transform the estimated *relative* angles to *absolute* angles with respect to the fixed Earth, and second, to compensate for the angular drift of the gyroscopes. Compensating for the angular drift without compass-based feedback given by the magnetometers would be much more difficult and the resulting angles would also be much less accurate. Unfortunately, this method also raises a significant problem, namely, the estimated Euler angles are highly sensitive to interferences of the Earth's magnetic field. Such interferences can be caused, in general, by any ferromagnetic substances in the surrounding environment. For instance, like experienced by the author, if the lab used for test flights is located in the building's basement, whose masonry consists of ferroconcrete and hence acts as a Faraday cage. To cope with this problem, the Euler angles provided by the IMU are "re-estimated" by the control system's state estimator with the purpose to diminish the influence of the magnetometers. This is accomplished by tuning the attitude estimator gains such that less uncertainty is assigned to the gyroscope measurements, which are transformed to earth-fixed angular velocities that are used for generating new Euler angle estimates, whereas more uncertainty is assigned to the original estimates that

are now exploited for correction. Besides transforming the gyroscope data, the derivatives of the earth-fixed angular velocities are computed to obtain the corresponding earth-fixed angular accelerations, which are forwarded to the motion control.

Analogous to the gyroscope data, the accelerometer and position measurements are transformed from body-fixed coordinates to earth-fixed coordinates in the first stage of the state estimator. Note that some of these data are processed by digital filters (see Section 4.3) before being passed to the estimator. The transformed accelerometer data are then used for generating the velocity estimates, and, in a subsequent step, for generating the position estimates. In case of the position estimates, the earth-fixed position data are exploited for correction, whereas in case of the velocity estimates, the derivatives of the transformed position data need to be computed to obtain earth-fixed velocities that can be utilized for correction. Note that the position measurements are received at much lower frequencies, and due to this, the correction step for both the position and velocity estimates is performed asynchronously. In addition to the state estimator, the earth-fixed linear accelerations are also passed to the motion control.

4.4.2 EKF Formalization

There exist many sophisticated methods for implementing the described state estimation principle, which, in the field of avionics, is generally referred to as the *inertial navigation principle* (see, e.g., [95]). However, the probably most prevalent method is to employ an EKF for this purpose. Compared to the original KF, introduced by Rudolf E. Kálmán in 1960, which is limited to *linear* systems, or, strictly speaking, to linear assumptions, the EKF, introduced by Stanley Schmidt in 1967, represents an "extended" version that was elaborated for the *nonlinear* case. Due to this extension, the EKF became the most widely used estimation algorithm applied to nonlinear navigation and control problems and many other problems as well. Both the KF and the EKF are recursive filters developed for dynamic systems with the aim to fuse various measurements in order to generate a better estimate than the estimate that would be obtained by using anyone measurement alone. Accordingly, the KF and the EKF, as well as many existing refinements, can be seen as efficient *sensor fusion* algorithms. They all have in common, first, a so-called *predictor-corrector* structure, and second, they involve the complete prediction history based on Taylor series expansion.

Remark 8. The aim of this subsection is to convey a brief understanding of the basic steps that lead to an EKF formalization. In this context, only the most essential background is covered, where many of the explanations and mathematical formalisms, sometimes adapted to match the present control system problem, are mainly based on the work of D. Simon [98]. Thus, for a complete description of the fundamentals, like linear systems theory and probability theory, as well as different KF algorithms and their mathematical derivations, the reader is referred to detailed literature, e.g., [98, 99].

Linear Dynamic Systems

Consider a continuous-time, deterministic, *linear* dynamic system, which can be expressed in the form

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \end{aligned}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{u}$ is the control-input vector, $\boldsymbol{y}$ is the control-output vector, and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ are appropriately dimensioned matrices. In accordance with the vectors, $\boldsymbol{A}$ is often called the system matrix, $\boldsymbol{B}$ the input matrix, and $\boldsymbol{C}$ the output matrix. If $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are constant, then the solution to this system is given by

$$\begin{aligned} \boldsymbol{x}(t) &= e^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}(t_0) + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)\, d\tau \\ \boldsymbol{y}(t) &= \boldsymbol{C}(t)\boldsymbol{x}(t) \end{aligned}$$

Note that, even if the matrices are not time-invariant, the system will still be linear.

Suppose the system is implemented as computer program to be executed on some digital device, which is usually the case regarding state estimation algorithms. Accordingly, we can only compute solutions at discrete instants of time, and therefore, need to transform the given system from continuous-time to discrete-time dynamics. Referring to the above solution, defining $t_k = t$ and $t_{k-1} = t_0$ yields

$$\boldsymbol{x}(t_k) = e^{\boldsymbol{A}(t_k - t_{k-1})}\boldsymbol{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{\boldsymbol{A}(t_k-\tau)}\, d\tau\, \boldsymbol{B}(t_{k-1})\boldsymbol{u}(t_{k-1})$$

where $\boldsymbol{A}(\tau)$, $\boldsymbol{B}(\tau)$, and $\boldsymbol{u}(\tau)$ are assumed to be approximately constant in the interval of integration. Defining $\Delta t = t_k - t_{k-1}$ and $\alpha = \tau - t_{k-1}$ as well as substituting for τ in the previous equation then gives

$$\begin{aligned} \boldsymbol{x}(t_k) &= e^{\boldsymbol{A}\Delta t}\boldsymbol{x}(t_{k-1}) + \int_0^{\Delta t} e^{\boldsymbol{A}(\Delta t-\alpha)}\, d\alpha\, \boldsymbol{B}(t_{k-1})\boldsymbol{u}(t_{k-1}) \\ &= e^{\boldsymbol{A}\Delta t}\boldsymbol{x}(t_{k-1}) + e^{\boldsymbol{A}\Delta t}\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha}\, d\alpha\, \boldsymbol{B}(t_{k-1})\boldsymbol{u}(t_{k-1}) \end{aligned}$$

and by defining substitution matrices

$$\begin{aligned} \boldsymbol{F} &= e^{\boldsymbol{A}\Delta t} \\ \boldsymbol{G} &= \boldsymbol{F}\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha}\, d\alpha\, \boldsymbol{B} \end{aligned}$$

we obtain the linear discrete-time approximation

$$\boldsymbol{x}_k = \boldsymbol{F}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{u}_{k-1}$$

with a sampling period of Δt.

Assume there is given a discrete-time, deterministic, linear dynamic system with control input $\boldsymbol{u}$, control output $\boldsymbol{y}$, and Gaussian, zero-mean, uncorrelated, white process noise and observation noise[2] $\boldsymbol{q}$ and $\boldsymbol{c}$, respectively, which can be written in the form

$$\begin{aligned}
\boldsymbol{x}_k &= \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{G}_k \boldsymbol{u}_k + \boldsymbol{q}_k \\
\boldsymbol{y}_k &= \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{c}_k \\
\boldsymbol{q}_k &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k) \\
\boldsymbol{c}_k &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_k)
\end{aligned}$$

where $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$ define the state transition, control-input, and control-output process, respectively, and $\boldsymbol{Q}$ as well as $\boldsymbol{C}$ are known covariance matrices.[3] Then a discrete-time KF can be applied to obtain periodic estimates describing the state of the system.

Nonlinear Dynamic Systems

Now consider a continuous-time, deterministic, *nonlinear* dynamic system, which can be expressed in the form

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{q}) \\
\boldsymbol{u} &= \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{s}) \\
\boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{c})
\end{aligned}$$

where $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ are arbitrary vector-valued functions that model the nonlinear state transition, control-input, and control-output dynamics, respectively, $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{y}$, $\boldsymbol{q}$, and $\boldsymbol{c}$ are the same vectors as before, and $\boldsymbol{s}$ is a vector consisting of new control-input measurements.

Suppose the model functions $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ represent an appropriate discretization of the system's continuous-time dynamics, such that it behaves approximately linear in the interval of one (sufficiently short) sampling period. Then the system and measurement equations can be written in the form

$$\begin{aligned}
\boldsymbol{x}_k &= \boldsymbol{f}_k(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{q}_k) \\
\boldsymbol{u}_k &= \boldsymbol{g}_k(\boldsymbol{x}_{k-1}, \boldsymbol{s}_k) \\
\boldsymbol{y}_k &= \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{c}_k) \\
\boldsymbol{q}_k &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k) \\
\boldsymbol{c}_k &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_k)
\end{aligned}$$

which can be used as input to a discrete-time EKF algorithm.

[2]The *process noise* and *observation noise* are most commonly denoted by $\boldsymbol{w}$ and $\boldsymbol{v}$, respectively. In order to avoid confusion with the *angular velocity* and *linear velocity* denoted by $\boldsymbol{\omega}$ and $\boldsymbol{v}$, respectively, the denotations $\boldsymbol{q}$ and $\boldsymbol{c}$ are used instead of $\boldsymbol{w}$ and $\boldsymbol{v}$.

[3]The *covariance matrices* referring to the process noise and observation noise are most commonly denoted by $\boldsymbol{Q}$ and $\boldsymbol{R}$, respectively. In order to avoid confusion with *rotation matrices* denoted by $\boldsymbol{R}$, the denotation $\boldsymbol{C}$ is used instead of $\boldsymbol{R}$.

Discrete-Time EKF Algorithm

The EKF algorithm for the discrete-time case comprises of three fundamental steps: first, the *initialization* step, second, the *prediction* step, also known as the "time update," and third, the *correction* step, also known as the "measurement update," where the latter two steps define the iterative estimation process. Recall from the previously discussed formulations of a linear system that the system dynamics, even if time-varying, is inherent in the state-transition, control-input, and control-output matrix. In the nonlinear case, these matrices, in general, need to be established periodically by computing the partial derivatives of the corresponding model functions.

Remark 9. In order to avoid double indices for vectors and matrices that are updated more than once within an iteration cycle, we follow the widespread convention of indicating an *a priori* update with a "−" superscript and an *a posteriori* update with a "+" superscript. Due to this convention, as an example, the projection sequence

$$\cdots \quad \hat{x}_{k-1|k-1} \quad \longrightarrow \quad \hat{x}_{k-1|k} \quad \longrightarrow \quad \hat{x}_{k|k-1} \quad \longrightarrow \quad \hat{x}_{k|k} \quad \cdots$$

becomes

$$\cdots \quad \hat{x}_{k-1}^{-} \quad \longrightarrow \quad \hat{x}_{k-1}^{+} \quad \longrightarrow \quad \hat{x}_{k}^{-} \quad \longrightarrow \quad \hat{x}_{k}^{+} \quad \cdots$$

Initialization. The correct initialization of an EKF is essential, because inappropriate values can cause the filter to diverge rapidly, usually within a few iterations. Provided that the initial state $\boldsymbol{x}_0$ is known, the initialization step is performed as follows:

$$\begin{aligned} \hat{\boldsymbol{x}}_0^+ &= \mathcal{E}(\boldsymbol{x}_0) \\ \boldsymbol{P}_0^+ &= \mathcal{E}\big[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+)(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+)^T\big] \end{aligned}$$

where $\boldsymbol{P}$ represents the covariance of the estimation error.

Prediction. The prediction step begins with computing the required Jacobian matrices

$$\boldsymbol{F}_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1}^+}, \qquad \boldsymbol{L}_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}}\right|_{\hat{\boldsymbol{x}}_{k-1}^+}$$

where $\boldsymbol{F}$ and $\boldsymbol{L}$ represent the state-estimate and process-noise transition, respectively. Given these matrices, the time update of the state estimate and the estimation-error covariance is performed as follows:

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- &= \boldsymbol{f}_k(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_k, \boldsymbol{0}) \\ \boldsymbol{P}_k^- &= \boldsymbol{F}_k \boldsymbol{P}_{k-1}^+ \boldsymbol{F}_k^T + \boldsymbol{L}_k \boldsymbol{Q}_k \boldsymbol{L}_k^T \end{aligned}$$

Correction. Similar to the prediction step, the correction step begins with computing the required Jacobian matrices

$$\boldsymbol{H}_k = \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_k^-}, \qquad \boldsymbol{N}_k = \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{c}}\right|_{\hat{\boldsymbol{x}}_k^-}$$

where $\boldsymbol{H}$ and $\boldsymbol{N}$ represent the control-output and observation-noise transition, respectively. Given these matrices, the measurement update of the state estimate and the estimation-error covariance is performed as follows:

$$\begin{aligned}
\boldsymbol{K}_k &= \boldsymbol{P}_k^- \boldsymbol{H}_k^T \big(\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{N}_k \boldsymbol{C}_k \boldsymbol{N}_k^T\big)^{-1} \\
\hat{\boldsymbol{x}}_k^+ &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \big(\boldsymbol{y}_k - \boldsymbol{h}_k(\hat{\boldsymbol{x}}_k^-, \boldsymbol{0})\big) \\
\boldsymbol{P}_k^+ &= (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^-
\end{aligned}$$

where $\boldsymbol{K}$ represents the "optimal" Kalman gain and $\boldsymbol{I}$ the *identity* matrix.

Remark 10. The above expressions for calculating $\boldsymbol{K}_k$ and $\boldsymbol{P}_k^+$ are the most practically used ones. However, there exist some other expressions that are more stable and robust, but also computationally more intensive. They are therefore rarely implemented if the computational resources are considerably limited.

4.4.3 EKF Implementation

The key to successful estimation is to sufficiently model the system dynamics. Referring to the aforementioned presentation form of a nonlinear dynamic system, there are three functions, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$, which can be used to model the state-transition, control-input, and control-output process, respectively. In this context, the three process models that are implemented in the JAviator control system will now be introduced.

Control-Input Model

The control input is usually modelled as part of the state transition model, especially if the provided measurements are not needed outside the estimator for any other purpose. In the present case, it is advantageous to model this process separately, because there are measurements that are also needed by the motion control as well as computed angular accelerations that are not required by the state estimator. Given new angular-velocity and linear-acceleration measurements $\boldsymbol{s}_k$, the control input $\boldsymbol{u}_k$ is defined by

$$\boldsymbol{g}_k(\boldsymbol{x}_{k-1}, \boldsymbol{s}_k) = \begin{bmatrix} \boldsymbol{R}_{\omega\to\dot{\varphi}}(\boldsymbol{\varphi}_{k-1}) \cdot \boldsymbol{\omega}_k \\ (\boldsymbol{R}_{\omega\to\dot{\varphi}}(\boldsymbol{\varphi}_{k-1}) \cdot \boldsymbol{\omega}_k - \boldsymbol{\omega}_{k-1})\Delta t^{-1} \\ \boldsymbol{R}_{b\to e}(\boldsymbol{\varphi}_{k-1}) \cdot \boldsymbol{a}_k + \boldsymbol{g} \end{bmatrix}$$

where $\boldsymbol{\omega}_{k-1}$ refers to the previous value that is kept in memory and $\boldsymbol{g} = [\,0\,0\,g\,]^T$ denotes the gravity vector representing the Earth's gravitational force.

State Transition Model

The state transition model describes how the state estimate is projected ahead with respect to the sampling period, or, in other words, how the *prediction* of the future state is performed. Given the control input $\boldsymbol{u}_k$ and the process noise $\boldsymbol{q}_k$, the transition of the state estimate $\hat{\boldsymbol{x}}_k$ is defined by

$$\boldsymbol{f}_k(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{q}_k) \quad = \quad \begin{bmatrix} \boldsymbol{\varphi}_{k-1} + (\boldsymbol{\omega}_k + \sigma_\omega)\Delta t \\ \boldsymbol{p}_{k-1} + \boldsymbol{v}_{k-1}\Delta t + \frac{1}{2}(\boldsymbol{a}_k + \sigma_a)\Delta t^2 \\ \boldsymbol{v}_{k-1} + (\boldsymbol{a}_k + \sigma_a)\Delta t \end{bmatrix}$$

As apparent from the model, each expression consists of a deterministic component that involves the control input and a stochastic component that involves the process noise. The EKF algorithm separates these two components and linearizes the stochastic part about the *current* state estimate, which can be considered as one of the fundamental steps performed by every KF.

Control-Output Model

This model describes how the attitude and position measurements, or, in other words, the *correction* feedback is prepared for fusion with the state estimate. Analogous to the state transition model, the control-output model can be divided into a deterministic component, the observations and computed velocities, and a stochastic component, the observation noise, which are separated by the EKF algorithm to linearize the stochastic part about the *updated* state estimate. Given new attitude and (possibly new) position measurements $\boldsymbol{x}_k$, the control output $\boldsymbol{y}_k$ is defined by

$$\boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{c}_k) \quad = \quad \begin{bmatrix} \boldsymbol{\varphi}_k + \sigma_\varphi \\ \boldsymbol{p}_k + \sigma_p \ \text{ if } \boldsymbol{p}_k \text{ updated, else } \boldsymbol{p}_{k-n} \\ (\boldsymbol{p}_k - \boldsymbol{p}_{k-n})(n\Delta t)^{-1} + \sigma_v \ \text{ if } \boldsymbol{p}_k \text{ updated, else } \boldsymbol{v}_{k-n} \end{bmatrix}$$

where $\boldsymbol{p}_{k-n}$ and $\boldsymbol{v}_{k-n}$ refer to the latest update of $\boldsymbol{p}$ and $\boldsymbol{v}$, respectively, which is kept in memory for n sampling periods, and

$$\boldsymbol{p}_k \quad = \quad \begin{bmatrix} \boldsymbol{R}_{b\to e}(\boldsymbol{\varphi}_{k-1}) \cdot [\,(k_{x,x} + x_k)\ k_{x,y}\ k_{x,z}\,]^T \ \text{ if the LDS is used, else } x_k \\ \boldsymbol{R}_{b\to e}(\boldsymbol{\varphi}_{k-1}) \cdot [\,k_{y,x}\ (k_{y,y} + y_k)\ k_{y,z}\,]^T \ \text{ if the LDS is used, else } y_k \\ \boldsymbol{R}_{b\to e}(\boldsymbol{\varphi}_{k-1}) \cdot [\,k_{z,x}\ k_{z,y}\ (k_{z,z} + z_k)\,]^T \ \text{ if the URF is used, else } z_k \end{bmatrix}$$

where $k_{\{x,y,z\},\{x,y,z\}}$ are constants representing the laser-beam and ultrasonic-pulse origin of the LDSs and the URF, respectively, relative to the craft's center point. Note that, similar to the control-input model, all rotations are performed depending on the previous attitude estimate $\boldsymbol{\varphi}_{k-1}$, even though new attitude observations would be available at this point. This is due to the fact that the rotation matrices are updated always at the end of an EKF iteration cycle.

Computation of EKF Matrices

Provided the state vector $\boldsymbol{x}$, the state transition function $\boldsymbol{f}$, the process noise vector $\boldsymbol{q}$, the control-output function $\boldsymbol{h}$, and the observation noise vector $\boldsymbol{c}$, solving for the required system matrices yields:

$$\boldsymbol{F}_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^+_{k-1}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & & \boldsymbol{0} \\ & \boldsymbol{I}_{3\times3} & \boldsymbol{S}_{3\times3}(\Delta t) \\ \boldsymbol{0} & & \boldsymbol{I}_{3\times3} \end{bmatrix}$$

$$\boldsymbol{L}_k = \left.\frac{\partial \boldsymbol{f}_k}{\partial \boldsymbol{q}}\right|_{\hat{\boldsymbol{x}}^+_{k-1}} = \begin{bmatrix} \boldsymbol{S}_{3\times3}(\Delta t) & & \boldsymbol{0} \\ & \boldsymbol{0} & \boldsymbol{S}_{3\times3}(\frac{1}{2}\Delta t^2) \\ \boldsymbol{0} & & \boldsymbol{S}_{3\times3}(\Delta t) \end{bmatrix}$$

$$\boldsymbol{Q}_k = \boldsymbol{D}_{9\times9}(\boldsymbol{q}_k)^2 = \begin{bmatrix} \boldsymbol{S}_{3\times3}(\sigma^2_\omega) & & \boldsymbol{0} \\ & \boldsymbol{S}_{3\times3}(\sigma^2_a) & \\ \boldsymbol{0} & & \boldsymbol{S}_{3\times3}(\sigma^2_a) \end{bmatrix}$$

$$\boldsymbol{H}_k = \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^-_k} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & & \boldsymbol{0} \\ & \boldsymbol{I}_{3\times3} & \\ \boldsymbol{0} & & \boldsymbol{I}_{3\times3} \end{bmatrix}$$

$$\boldsymbol{N}_k = \left.\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{c}}\right|_{\hat{\boldsymbol{x}}^-_k} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & & \boldsymbol{0} \\ & \boldsymbol{I}_{3\times3} & \\ \boldsymbol{0} & & \boldsymbol{I}_{3\times3} \end{bmatrix}$$

$$\boldsymbol{C}_k = \boldsymbol{D}_{9\times9}(\boldsymbol{c}_k)^2 = \begin{bmatrix} \boldsymbol{S}_{3\times3}(\sigma^2_\varphi) & & \boldsymbol{0} \\ & \boldsymbol{S}_{3\times3}(\sigma^2_p) & \\ \boldsymbol{0} & & \boldsymbol{S}_{3\times3}(\sigma^2_v) \end{bmatrix}$$

where $\boldsymbol{D}$ represents a *diagonal* matrix and $\boldsymbol{S}$ a *scalar* matrix. Since the sampling period Δt is constant, for any $k \in \mathbb{N}$ satisfying $0 < k < \infty$ we obtain:

$$\begin{aligned} \boldsymbol{F}_{k-1} &= \boldsymbol{F}_k = \boldsymbol{F}_{k+1} \\ \boldsymbol{L}_{k-1} &= \boldsymbol{L}_k = \boldsymbol{L}_{k+1} \\ \boldsymbol{Q}_{k-1} &= \boldsymbol{Q}_k = \boldsymbol{Q}_{k+1} \\ \boldsymbol{H}_{k-1} &= \boldsymbol{H}_k = \boldsymbol{H}_{k+1} \\ \boldsymbol{N}_{k-1} &= \boldsymbol{N}_k = \boldsymbol{N}_{k+1} \\ \boldsymbol{C}_{k-1} &= \boldsymbol{C}_k = \boldsymbol{C}_{k+1} \end{aligned}$$

As a consequence, all of the system matrices can be established in advance and need not to be re-computed with each iteration cycle.

Simplification of EKF Terms

The matrix product computations concerning the transition of the process and observation noise covariance can be eliminated by the following two substitutions:

$$\boldsymbol{W} = \boldsymbol{L}\boldsymbol{Q}\boldsymbol{L}^T = \begin{bmatrix} \boldsymbol{S}_{3\times3}(\sigma_\omega^2\Delta t^2) & & \boldsymbol{0} \\ & \boldsymbol{S}_{3\times3}(\frac{1}{4}\sigma_a^2\Delta t^4) & \boldsymbol{S}_{3\times3}(\frac{1}{2}\sigma_a^2\Delta t^3) \\ \boldsymbol{0} & \boldsymbol{S}_{3\times3}(\frac{1}{2}\sigma_a^2\Delta t^3) & \boldsymbol{S}_{3\times3}(\sigma_a^2\Delta t^2) \end{bmatrix}$$

$$\boldsymbol{V} = \boldsymbol{N}\boldsymbol{C}\boldsymbol{N}^T = \begin{bmatrix} \boldsymbol{S}_{3\times3}(\sigma_\varphi^2) & & \boldsymbol{0} \\ & \boldsymbol{S}_{3\times3}(\sigma_p^2) & \\ \boldsymbol{0} & & \boldsymbol{S}_{3\times3}(\sigma_v^2) \end{bmatrix}$$

where $\boldsymbol{V} = \boldsymbol{C}$, because $\boldsymbol{N}$ resulted in an identity matrix.

Besides these substitutions, the computation of the optimal Kalman gain as well as the correction of the estimation-error covariance can be further simplified by removing $\boldsymbol{H}$, which also resulted in an identity matrix. Accordingly, the original system of EKF equations takes on the following form:

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- &= \boldsymbol{f}_k(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_k, \boldsymbol{0}) \\ \boldsymbol{P}_k^- &= \boldsymbol{F}\boldsymbol{P}_{k-1}^+\boldsymbol{F}^T + \boldsymbol{W} \\ \boldsymbol{K}_k &= \boldsymbol{P}_k^-\big(\boldsymbol{P}_k^- + \boldsymbol{V}\big)^{-1} \\ \hat{\boldsymbol{x}}_k^+ &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\big(\boldsymbol{y}_k - \boldsymbol{h}_k(\hat{\boldsymbol{x}}_k^-, \boldsymbol{0})\big) \\ \boldsymbol{P}_k^+ &= (\boldsymbol{I} - \boldsymbol{K}_k)\boldsymbol{P}_k^- \end{aligned}$$

Finally, the initialization of the filter with respect to the JAviator control system is straightforward. More precisely, except for the roll and pitch angle, which are around zero provided that the ground is sufficiently even, the yaw angle, the position, and the velocity are permanently adjusted to zero when the craft is settled. Due to this, the initial state is well-known. In order to keep an EKF's transient time short and also to prevent unexpected divergence, it is advisable to have the estimation-error covariance indicating high uncertainty in the initial measurements. Hence, the filter is initialized as follows:

$$\begin{aligned} \hat{\boldsymbol{x}}_0^+ &= \boldsymbol{0} \\ \boldsymbol{P}_0^+ &= \boldsymbol{W} \end{aligned}$$

EKF Decomposition

The EKF is implemented in the present matrix-valued form as MATLAB script,[4] which, from the perspective of verification, serves as reference algorithm for the implementation

[4] MATLAB, produced by the MathWorks Inc., is a widespread tool for technical computing, which, for example, allows for implementing mathematical formalisms in a fast and convenient way.

in the JAviator control system, and in the light of operation simplicity, is exploited for tuning the filter.[5] Although implementing the matrix-valued expressions in their original form facilitates a convenient and comprehensible coding style, in the present case, it also largely increases the computational load. Particularly, all of the system matrices have a dimension of 9×9, which leads to 81 computations in case of matrix addition, subtraction, and multiplication with a scalar, and a total of 1458 computations in case of building the matrix product. This can be seen as an extremely high computational effort due to the rather large redundancy in the computations, because all of the system matrices contain many zero entries. An even more challenging problem is solving for the *inverse* matrix of the estimation-error covariance, which is required for computing the optimal Kalman gain. Closed formulas for building the inverse of a matrix, which are of reasonable computational effort, exist only for matrices not exceeding a dimension of 3×3. For matrices of higher dimension, the inverse matrix can be determined via the *adjunct* matrix and the *determinant* according to the relation $\boldsymbol{A}^{-1} = \operatorname{adj}(\boldsymbol{A}) / \det(\boldsymbol{A})$. However, this requires to implement an efficient algorithm for computing the adjunct and determinant of a matrix, for instance, via Gaussian elimination or Lower-Upper (LU) factorization (for a detailed description of different methods see, e.g., [100]).

Nevertheless, there is some light at the end of the tunnel. In contrast to the physical dependencies that exist between attitude, position, and velocity, the only computational dependencies that exist in the approximated system dynamics refer to the coordinate transformations, which depend on all three axes. Actually, they are already separated from the EKF by the control-input and control-output model, which define the preparation of measurements, and hence all required transformations, as external processes. This circumstance immediately suggests the implementation of a 1D EKF that can be used as template for creating three instances. Following this suggestion, the dimension of the system matrices can be reduced to 3×3, hence allowing to apply a closed formula for computing the inverse matrix. However, there is even one more opportunity left that should be taken into account, namely, the attitude estimate is generated independently of the position and velocity estimate. This means that the proposed 1D EKF can be further reduced with respect to the system's dimension by dividing into a 1D attitude EKF and a 1D position-velocity EKF. In case of the attitude filter, since the system's dimension is 1, all system equations degenerate into scalar expressions. In case of the position-velocity filter with a system dimension of 2, the closed formula $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]^{-1} = \frac{1}{ad-bc}\left[\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right]$ can be used to break down the inverse-matrix computation into scalar expressions, hence allowing for a completely scalar filter implementation.

The computational savings gained by this decomposition reach an order of magnitude, such that both EKFs can be implemented in the proposed scalar form even on smallest microcontrollers. As a side effect, asynchronous updates can be handled easily. That is, as long as no new observations are available to perform a correction of the state estimate,

[5] Tuning an EKF, or any other type of KF, in general, means to adjust the *filter gains*, which, a bit misleadingly, refers to adjusting the *noise values* until the filter shows the desired behavior.

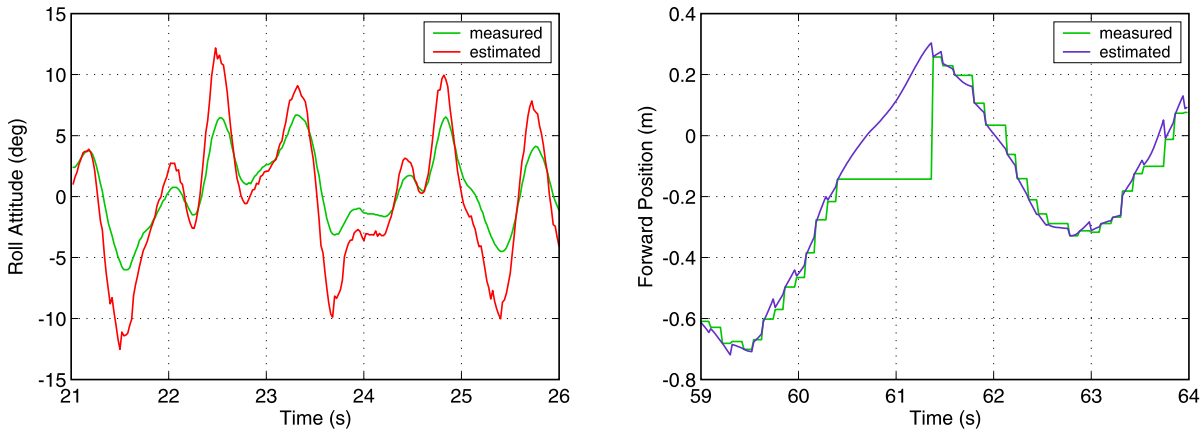


Figure 4.5: Performance of attitude EKF provided with periodic ω_x measurements and *synchronous* ϕ observations (left) and performance of position EKF provided with periodic a_x measurements and *asynchronous* x observations (right).

only the prediction step is executed, otherwise the estimate would not be projected ahead properly. In case of not separating the attitude estimation from the position and velocity estimation, this means that one has to deal in a matrix-valued system with the problem of executing the correction step differently for each component of the state vector. For the matrix-valued implementation in MATLAB, this problem was solved by simply setting the corresponding entries in the Kalman gain matrix to zero when no update is available. Lastly, it is mentionable that with decomposition there also comes along a considerable drawback. In particular, once the filter equations have been decomposed, all redundant expressions removed, and the remaining algorithm implemented in scalar form, it is indeed difficult and error-prone to apply any changes if necessary. Therefore, it is advantageous to use MATLAB, or any other comparable tool for technical computing, in order to verify all modifications applied to the stripped implementation.

EKF Performance

The performance achieved with the introduced state estimation technique, including the preprocessing of some measurements with digital filters, is demonstrated in Figure 4.5 and Figure 4.6, which depict the general behavior of the EKF for different control inputs and correction feedbacks. Specifically, Figure 4.5 (left) shows the effect of re-estimating the attitude data, which, in the present case, are obtained by assigning less uncertainty to the gyroscope measurements and more uncertainty to the already KF-estimated attitude observations. As apparent from the curve referring to the EKF-estimated attitude, the influence of the increased trust in the gyroscope measurements and decreased trust in the original magnetometer-based, gyro-stabilized attitude data, which are now exploited as correction feedback, leads to an equally shaped but remarkably more intense attitude prediction. Concerning the generation of position estimates, an exceptionally extreme condition is depicted in Figure 4.5 (right), where, for whatever reason, no position data

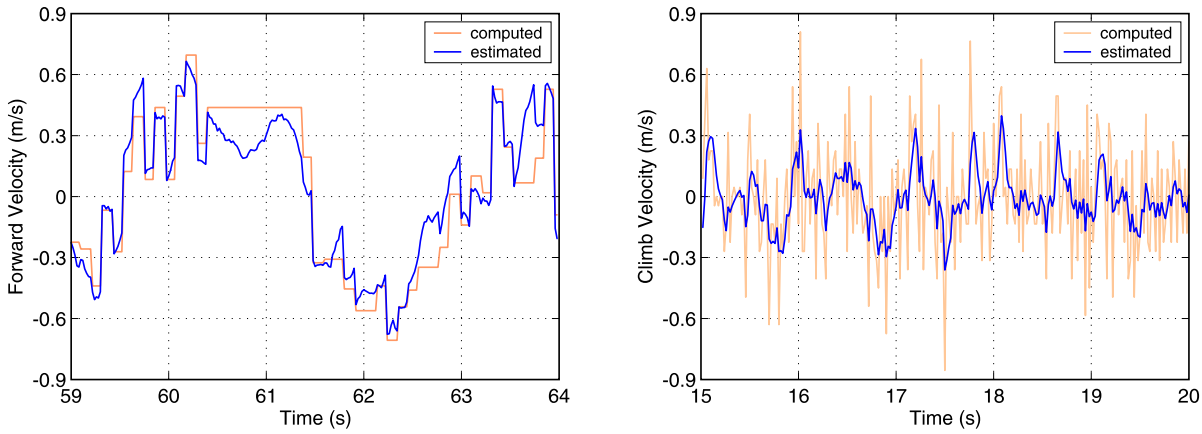


Figure 4.6: Performance of velocity EKF provided with periodic a_x measurements and *asynchronous* v_x feedback (left) and performance of velocity EKF provided with periodic a_z measurements and *synchronous* v_z feedback (right).

were delivered by the SLS for almost one second. It is quite impressive and seems like magic how well and accurate the filter keeps predicting the craft's position in the absence of any correction feedback. The associated velocity that was estimated concurrently based on the same measurements, and hence without computed velocity feedback during this one-second lag of position observations, is depicted in Figure 4.6 (left). It is indeed interesting to see that the curve referring to the velocity estimate does not show any divergent or otherwise unexpected behavior in this exceptional situation. Note that, even if the position feedback is provided without significant lags, the position and velocity estimate is corrected at a much lower frequency. Especially in case of estimating the velocity, this circumstance generally leads to curves that contain many stepwise changes. Contrariwise, Figure 4.6 (right) shows a typical velocity curve resulting from more frequent, almost periodic position observations, which, in the present case, were delivered by the URF. One can imagine now that it becomes unavoidable to incorporate some form of state estimation when being faced with erratically generated control input.

4.5 Motion Control

In the last stage of the control cycle, the motion control is invoked, which comprises of several controllers that contribute with their produced output signals, or *control efforts*, to the generation of the PWM signals used for driving the four motors. Referring to the dynamics of a quadrotor helicopter (see Section 1.2), all possible motions in space can be accomplished via certain combinations of the four fundamental maneuvers "roll", "pitch", "yaw", and "climb", which, per definition of a VTOL vehicle, are similar for any type of helicopter. In conformance with this dynamics, it suffice to implement an individual controller for each of these four maneuvers in order to control a quadrotor. Particularly, the *roll* (ϕ) controller and the *pitch* (θ) controller stabilize the craft about the x-axis

and the y-axis, respectively, by regulating the *ratio* of generated thrust between a pair of opposite rotors, specifically, the ratio between the front and rear rotor and the ratio between the left and right rotor. In other words, the additional amount of thrust generated by the one rotor is subtracted in the same amount from the other rotor, such that the total of thrust, and hence the rotational moment, keeps unchanged. The *yaw* (ψ) controller stabilizes the craft about the z-axis by regulating the *ratio* of generated thrust between the aforementioned pairs of opposite rotors. Analogous to the ϕ and θ controller, the amount of additional thrust generated by the one pair of rotors is deducted from the other pair, such that the total of thrust, and hence the entire lifting force, keeps unchanged. Lastly, the *climb* (z) controller stabilizes the craft at the desired altitude by regulating the amount of thrust that is required to compensate for the Earth's gravitational force.

Quadrotor Control Law

According to the above specifications, given the control efforts u_ϕ, u_θ, u_ψ, and u_z produced by the ϕ, θ, ψ, and z controller, respectively, and the PWM signals u_1, u_2, u_3, and u_4 referring to the *front*, *right*, *rear*, and *left* motor, respectively, the following control law for a quadrotor helicopter can be derived:

$$\begin{aligned} u_1 &= u_z/4 + u_\psi/2 + u_\theta \\ u_2 &= u_z/4 - u_\psi/2 - u_\phi \\ u_3 &= u_z/4 + u_\psi/2 - u_\theta \\ u_4 &= u_z/4 - u_\psi/2 + u_\phi \end{aligned}$$

where a *counter-clockwise*-spinning front/rear rotor and a *clockwise*-spinning left/right rotor is assumed. Note that in case of a *clockwise*-spinning front/rear rotor and a *counter-clockwise*-spinning left/right rotor, the sign of the u_ψ term needs to be changed in each of the four control law equations.

Remark 11. The control effort constants 1/4 and 1/2 are usually omitted when it comes to implementation, because they can be combined with the scaling constants that determine a controller's individual control contributions.

PIDD Controller Model

The controller model exploited as baseline for all controller implementations represents a so-called Proportional Integral Derivative 2nd Derivative (PIDD) controller, also known as PID-D2 controller, which can be expressed in the following form:

$$u = k_p(\xi_{ref} - \xi) + k_i \int_0^t (\xi_{ref} - \xi)\, dt + k_d(\dot{\xi}_{ref} - \dot{\xi}) + k_{d2}\, \ddot{\xi}$$

where ξ_{ref} and $\dot{\xi}_{ref}$ denote the reference command and its derivative, respectively, and $k_{\{p,i,d,d2\}}$ denote the constants referring to the specific control effort contributions. This PIDD model is based on a controller design introduced by G. M. Hoffmann [101], which,

in addition to computing the velocity error $(\dot{\xi}_{ref} - \dot{\xi})$, also comprises the computation of the acceleration error $(\ddot{\xi}_{ref} - \ddot{\xi})$. In the scope of experimenting with the controller in the proposed form, we observed a somewhat more aggressive behavior when incorporating the second derivative of the reference command. Consequently, the acceleration term was changed to the standard form of involving the plain acceleration feedback instead of the computed acceleration error. Nonetheless, compared to conventional PIDD controllers that only compute the proportional error, this approach stands out with a significantly increased response time in regard to changes of the reference command. Due to this, the craft's response to fast changes of the attitude commands, generated either by the joystick in case of manually piloted flights or by the position controllers in case of autonomous flights, could be improved by an order of magnitude.

4.5.1 Manual Flight Control

Similar to flying conventional RC aerial vehicles, the JAviator can be piloted manually by invoking only the ϕ, θ, ψ, and z controller. In case of RC aerial vehicles, the user-issued control commands are sampled via some radio steering device and transmitted in form of PWM signals that can be assigned on the receiver side directly to the servos. In case of the JAviator, the Control Terminal software samples the control commands that are issued by the user via a 4-axis joystick or the keyboard and sends them periodically to the control system as a sequence of reference commands $\boldsymbol{r}_{4DOF} = (\, r_\phi, r_\theta, r_\psi, r_z \,)$.

Roll and Pitch Controller

The ϕ and θ controller need to respond as fast as possible and therefore do not involve the integral effort, which otherwise would slow down the settling process. The key to fast response is a relatively high derivative feedback in combination with a modest acceleration feedback to compensate for the overshoot. That being said, the ϕ and θ controller take on the following form:

$$u_\phi = k_{\phi,p}(r_\phi - \hat{\phi}) + k_{\phi,d}(\dot{r}_\phi - \omega_x) + k_{\phi,d2}\,\alpha_x$$

$$u_\theta = k_{\theta,p}(r_\theta - \hat{\theta}) + k_{\theta,d}(\dot{r}_\theta - \omega_y) + k_{\theta,d2}\,\alpha_y$$

Yaw Controller

The ψ controller neither requires an integral effort, since there are no strong moments that hamper reaching the set point, nor does it require a high derivative feedback that must be compensated by acceleration feedback. Accordingly, the ψ controller is the simplest of all controllers involved, which takes on the following form:

$$u_\psi = k_{\psi,p}(r_\psi - \hat{\psi}) + k_{\psi,d}(\dot{r}_\psi - \omega_z)$$

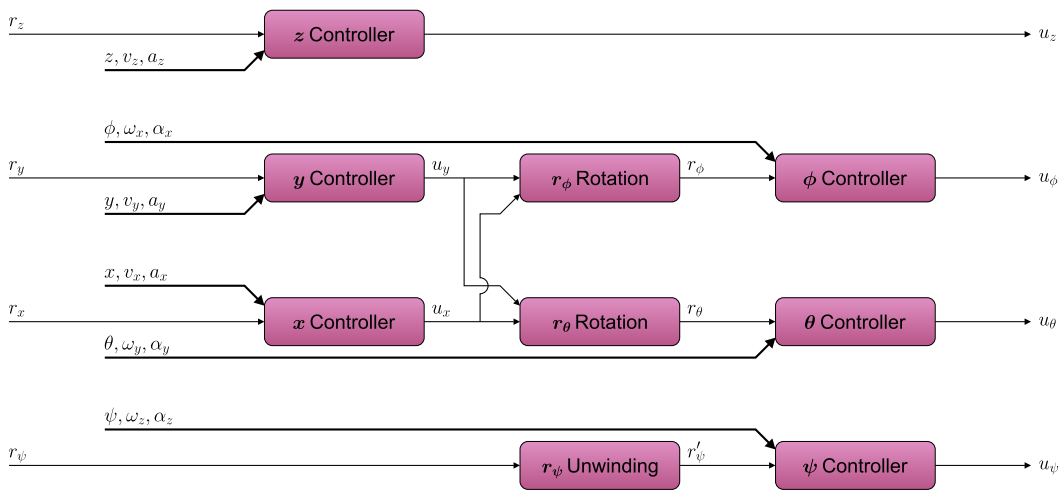


Figure 4.7: Model of the motion control process comprising of position controllers (left), computational components (middle), and attitude controllers (right).

Climb Controller

The z controller is actually the most essential and also the most extensive one. The duty of this controller is to regulate the amount of thrust required to stabilize the craft at some altitude, which demands to permanently compensate for thrust perturbations that are incurred by the other controllers. Since the proportional error as well as the velocity and acceleration converge to zero once the desired altitude is reached, the integral effort remains as the only contribution to the overall control effort. Consequently, the integral effort is core to vertically stable hovering and flying. Similar to the ϕ and θ controller, the z controller uses a relatively high velocity feedback with acceleration-based compensation to provide sufficiently fast response. In order to keep the control range small, the control effort is offset with a *base signal*, denoted by u_b, which corresponds to a thrust generation close to lift-off. In addition to this, the vertical thrust that is lost in favor of directional propulsion is compensated with a trigonometric correction term. Hence, the z controller takes on the following form:

$$u_z = u_b + \left(k_{z,p}(r_z - \hat{z}) + k_{z,i} \int_0^t (r_z - \hat{z})\, dt + k_{z,d}(\dot{r}_z - \hat{v}_z) + k_{z,d2}\, a_z \right) (\cos\phi \cos\theta)^{-1}$$

4.5.2 Position Flight Control

Raising the so far discussed 4-DOF control system to a full 6-DOF control system that allows for autonomous position control is accomplished with two additional controllers, specifically, the *forward* (x) controller and the *lateral* (y) controller. As apparent from the motion control model depicted in Figure 4.7, the integration of these two controllers results in a cascaded design. This is due to the fact that directional changes of a quadrotor helicopter using fixed-pitch rotors can only be achieved by changing the craft's attitude.

More precisely, in contrast to positional changes along the z-axis performed directly by the z controller, any positional changes along the x-axis and the y-axis must be accomplished by appropriately varying the ϕ and θ angle. This means that the corresponding ϕ and θ reference commands, which are generated by the user concerning the 4-DOF control system, are now generated by the y and x controller, respectively. Note that moving along the x-axis, which points *ahead* in the aircraft coordinate system (see Subsection 4.1.1), requires to "pitch" the craft by invoking the θ controller, whereas moving along the y-axis requires to "roll" the craft by invoking the ϕ controller. Compared to the 4-DOF control system, the 6-DOF control system accepts *absolute* position commands, which are received periodically from the JAviator Control Terminal as a sequence of reference commands $\boldsymbol{r}_{6DOF} = (\, r_x, r_y, r_z, r_\psi \,)$.

Forward and Lateral Controller

The x and y controller do not use aggressive velocity feedback and therefore also do not require acceleration feedback for compensation. However, similar to the z controller, they make use of the integral effort to horizontally stabilize the craft around the set point. Due to this, the x and y controller take on the following form:

$$\begin{aligned} u_x &= k_{x,p}(r_x - \hat{x}) + k_{x,i} \int_0^t (r_x - \hat{x})\, dt + k_{x,d}\, \hat{v}_x \\ u_y &= k_{y,p}(r_y - \hat{y}) + k_{y,i} \int_0^t (r_y - \hat{y})\, dt + k_{y,d}\, \hat{v}_y \end{aligned}$$

4.5.3 Spin Limit Extension

The ϕ and θ angle are defined in the ranges $-\pi \leq \phi \leq \pi$ and $-\pi/2 < \theta < \pi/2$, respectively, which can be considered as limits that are never exceeded or even reached under normal flight conditions. However, in case of the ψ angle, which, analogous to the ϕ angle, is defined in the range $-\pi \leq \psi \leq \pi$, these limits obviously prevent the craft from continuously spinning about the z-axis. Actually, it is not these limits that prevent spinning for more than one full turn, it is the fact that neither the EKF algorithm nor the PIDD controller is capable of handling noncontinuous control inputs. For this reason, both the ψ measurement and the ψ reference command are mapped to continuous ranges by multiplying the corresponding angle by the number of counted full-circle windings, which is commonly referred to as the *winding number*. Accordingly, the winding number, denoted by γ, and the unwinded angle, denoted by ψ', are computed as follows:

$$\begin{aligned} \gamma_k &= \begin{cases} \gamma_{k-1} - 1 & \text{if } \psi_k - \psi_{k-1} > \pi, \\ \gamma_{k-1} + 1 & \text{if } \psi_k - \psi_{k-1} < -\pi, \\ 0 & \text{if craft settled} \end{cases} \\ \psi'_k &= \psi_k + 2\gamma_k \pi \end{aligned}$$

In order to enable arbitrary, unlimited spinning about the z-axis without influencing the directional navigation, the ϕ and θ reference commands are rotated by the current ψ angle before being passed to the ϕ and θ controller, respectively. This rotation, which merely depends on the ψ angle, is performed in the following way:

$$\begin{aligned} r_\phi &= u_x \sin\psi + u_y \cos\psi \\ r_\theta &= u_x \cos\psi - u_y \sin\psi \end{aligned}$$

Note that the present approach of fully decoupling the ψ alignment from the directional navigation demands that the ψ angle is adjusted to zero when the craft is settled.

Chapter 5

Software Architecture

An apprentice carpenter may want only a hammer and saw, but a master craftsman employs many precision tools. Computer programming likewise requires sophisticated tools to cope with the complexity of real applications, and only practice with these tools will build skill in their use.

Robert L. Kruse, *Data Structures and Program Design*, 1994

The third major chapter describes the JAviator software architecture, beginning with an introductory discussion of the main differences compared to the most commonly chosen design applied to control software. Appended to this discussion given in Section 5.1, the underlying communication protocol is introduced, followed by an overview of the basic software architecture and comprised components. The subsequent, more detailed description is divided into three sections referring to the three fundamental layers of the software system. Particularly, Section 5.2 describes all components related to sensing and actuating, Section 5.3 continues with explaining the components of the flight control software, and Section 5.4 concludes with a description of the ground control software.

5.1 Design Approach

When implementing control software on some small embedded device, one is usually faced with very limited computational resources and, not uncommon, with pure integer-based processors that bring along the burden of fixed-point programming. Control software is mostly implemented in form of an event-triggered system that executes interrupt routines in case of interrupt-throwing sensors or otherwise polls for new data. Tasks running at different frequencies often lead to a cascaded software design that is typically tuned depending on the employed sensor and actuator equipment. Such sensor-frequency-based designs are by nature of this approach highly susceptible to changes in the encompassed environment. Even smallest changes in the flow of execution often cause the whole system to behave completely differently, which sometimes demands cumbersome adapting or even

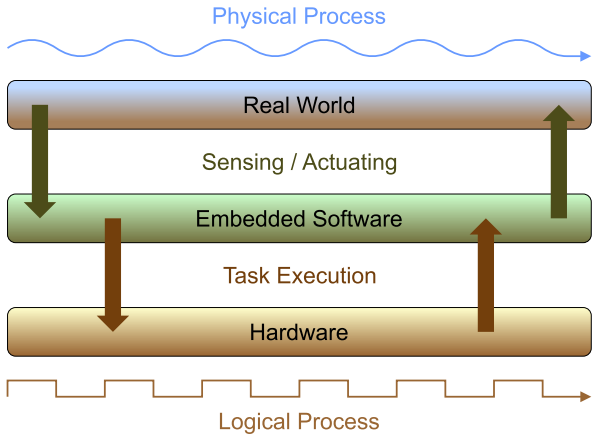


Figure 5.1: Principle of logical timing according to the LET model: *sensing* and *actuating* happens at exactly defined time instants independently of the actual *task execution*.

re-engineering the entire software architecture. One way to prevent these problems is the integration of a mechanism that offers more flexibility with respect to timing, an approach that was realized in the JAviator software system.

5.1.1 Logical Timing

In order to provide a control software architecture that enables temporal independency in regard to the sensor frequencies and also allows for convenient high-level programming, the design of the JAviator software system is based on the principle of Logical Execution Time (LET) [102]. More precisely, the LET model isolates the hardware-dependent *real* execution time required by some task and encapsulates it into a virtual time frame that represents the software-dependent *logical* execution time assigned to this task. This form of separation into "logically assigned" and "actually consumed" execution time is depicted in Figure 5.1, where the embedded-software layer implements the LET model. According to the LET principle, *sensing* and *actuating* is performed at exactly defined time instants, whereas the time interval referring to the actual execution of the task associated with the sensed data depends on two conditions: first, the task's *release for execution*, which might not be granted due to other sensor data and/or computational resources not available currently, and second, the task's *scheduled start time*, which might vary from the planned schedule due to a possibly postponed release of the task. In any case, after the task's execution has completed, the results are withhold until the next actuating cycle. This mechanism, which is commonly referred to as *temporal isolation*, raises two significant advantages compared to traditional approaches: first, a LET-based and consequently fully time-triggered control system can be implemented to be completely *time-invariant*, and second, decoupling the sampling period from the true sensor frequencies allows for a *reliable distribution* of the sensing and actuating processes over several computational devices, hence facilitating largely distributed architectures.

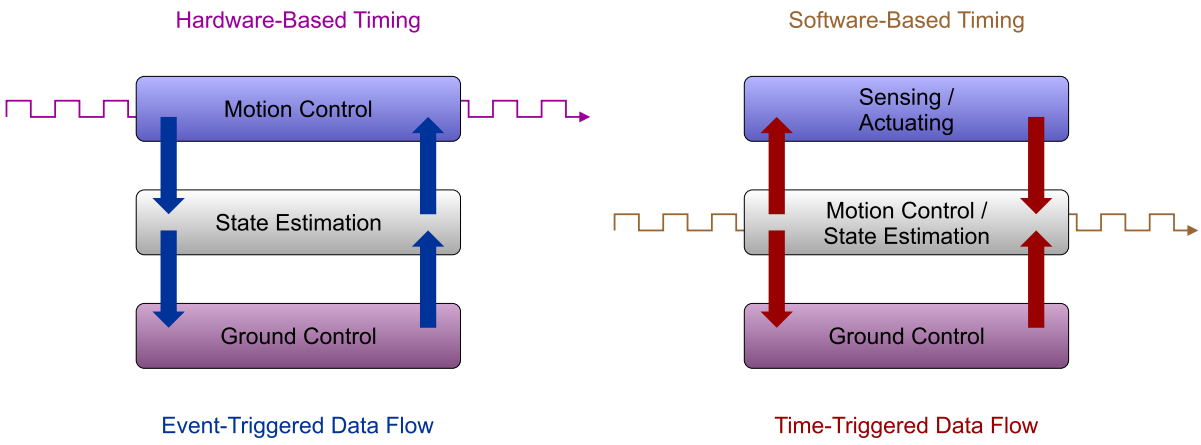


Figure 5.2: Timing source, task separation, and data flow in an event-triggered distributed control system (left) and in a time-triggered distributed control system (right).

Consider an embedded control system that must be implemented on some small device with low clock frequency and very limited computational resources. Assume the control software contains a state estimator that demands considerable computational effort. To cope with the computational limitations, it is not unusual that another, more powerful processor is involved to handle the additional workload. However, the supplementary processor leads to a distributed system, and consequently, increased communication that must be synchronized to some extent. The predominant approach for timing a control system is to use a hardware-based clock signal, for instance, provided by some continuously running sensor that delivers its measurements at a constant output rate. Such a sensor can be utilized to time the entire control cycle, in other words, to define the system's sampling period. If the system represents a cascaded control design consisting of several inner and outer loops, the sampling period is usually the fastest period and assigned to the most inner loop. The timing of the slower loops is often derived directly from the fastest loop to avoid loop synchronization that would otherwise be necessary. In any case, what we obtain is a typical event-triggered control system, depicted in Figure 5.2 (left), where all functional tasks, whether computing signals or communicating data, adhere to the period of the hardware-generated clock signal. When viewed from the perspective of hardware utilization, an event-triggered system can often be tuned to achieve nearly optimal efficiency. Nevertheless, nothing comes for free, because the higher the achieved efficiency the lower the flexibility in regard to hardware and software changes.

Now consider the system with the same initial conditions, but this time the software is written based on logical execution times. That is, by applying the LET model the timing source becomes a central part of the software, as depicted in Figure 5.2 (right). Thus, as a consequence of defining logical execution times that adhere to the software-based clock, the hardware-associated sensing and actuating tasks can be decoupled from the control cycle and established as independent processes. Due to this, the functional part of the control cycle can be executed on any other device – ideally on the same device that runs

the estimation process − and the time invariance of the system will be preserved as long as the underlying hardware provides the required computational resources. Hence, what we obtain by this approach is a time-portable control software that allows for changes across the hardware and software platform without loss of its temporal behavior.

5.1.2 Communication

Sophisticated communication mechanisms belong to the key elements of every distributed system. Besides affecting the overall throughput and resulting efficiency of a system, they play an important role in the light of compatibility to other software modules. For the purpose to arbitrarily exchange the layers of the JAviator software system, all layers communicate via a simple, custom-designed message-passing protocol of the following format: a message is a packet with a header, a payload of variable size, and a checksum to ensure data integrity. As illustrated in Figure 5.3, each packet starts with a packet delimiter (`0xFF`) followed by two header bytes. The first one of these two bytes contains

Figure 5.3: Packet format used for communication between JAP, FCS, and GCS.

the type of the packet, for example, sensor data. The second byte encodes the payload size, followed by the payload, two checksum bytes, and another, final packet delimiter. Due to this protocol-based layer separation, we are able to experiment with different control algorithms, alternative implementations and runtime environments, and even different programming languages like C and Java.

5.1.3 System Layers

The JAviator software system consists of three layers called the JAviator Plant (JAP), the Flight Control System (FCS), and the Ground Control System (GCS), as shown in Figure 5.4 (left). Each of these layers was developed independently with well-defined interfaces to the other layers. There are multiple, alternative implementations of all layers, which can be switched at compile time without changing any of the other layers.

The lowest layer, in terms of software hierarchy, is the JAP that provides an abstraction of sensors and actuators to the FCS. It either communicates sensor and actuator data with the real JAviator, referred to as the Physical Plant, or implements a simulation of the JAviator's flight dynamics, referred to as the Simulated Plant. The Physical Plant is divided into two programs written in C that implement device drivers for the sensors and actuators. Specifically, the RoboMaster manages the permanently involved sensor suite that samples inertial and altitude data, generates the PWM signals assigned to the motor controllers, and communicates with the FCS, whereas the RoboSlave takes care of the auxiliary sensor suite that samples positional data. These low-level programs run

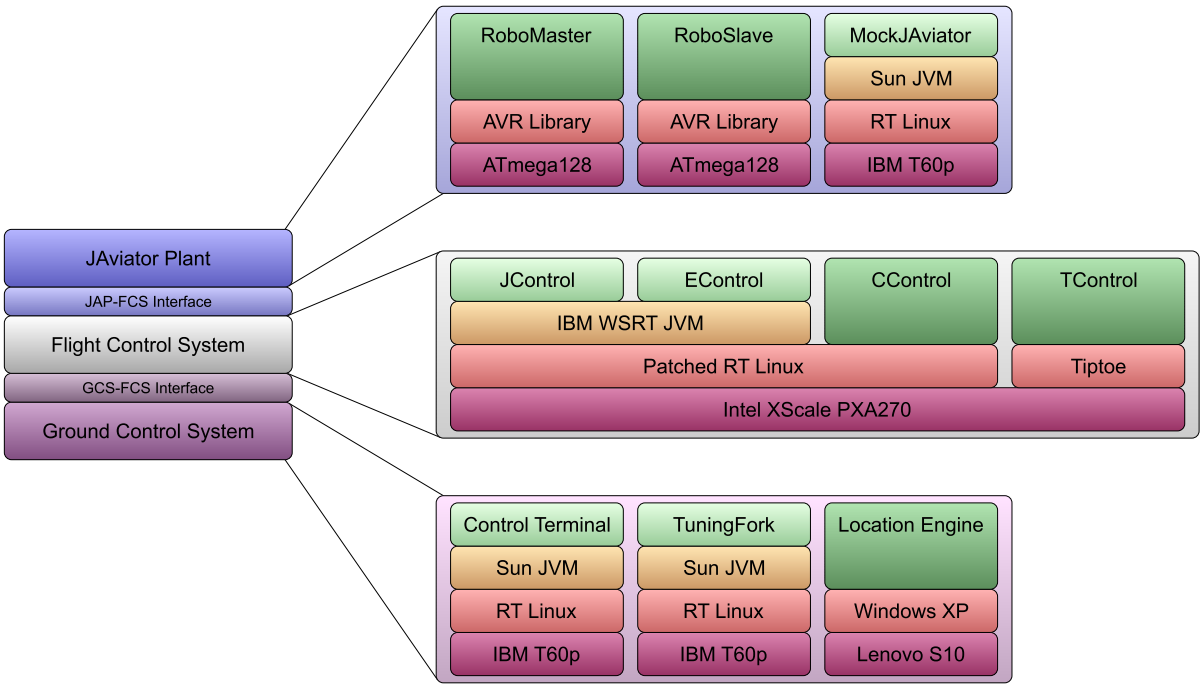


Figure 5.4: Model of the JAviator software architecture separated into basic components of the JAviator Plant, the Flight Control System, and the Ground Control System.

on the two Robostix boards and are therefore tailored to be executed on an ATmega128 processor. The Simulated Plant, in contrast, is a program written in Java, called the MockJAviator, which runs on any Java Virtual Machine (JVM).

The FCS layer is the most complex component of the JAviator software system. It implements the flight control algorithms and communicates with both the JAP and the GCS. As shown in Figure 5.4 (right), there are four alternative FCS implementations: JControl, a Java-thread-based controller written in Java, EControl, an Exotask-based controller written in Java, CControl, a Linux-process-based controller written in C, and TControl, a Tiptoe-based controller written in C. JControl and EControl run on top of a JVM, CControl runs as a multi-threaded Linux process, and TControl is compiled directly into the Tiptoe kernel. Except for TControl, which runs bare-metal on Verdex boards, all other controllers are executed on a Linux operating system with special real-time support for CPUs of the XScale processor family.

Finally, the GCS comprises of the Control Terminal, a Java-based application created for piloting and monitoring the JAviator, the IBM TuningFork logging system [103], a Java-based application used for real-time data analysis, and the Ubisense Location Engine configuration tool [104], a C-based application required for calibrating and operating the localization system that is utilized for indoor flights.

More detailed information about the introduced control system layers and their specific components will be provided in the scope of the next three sections.

5.2 JAviator Plant

The JAviator Plant can be represented by either operating the real JAviator, in this case denoted Physical Plant, or running a simulation of its flight dynamics, in that case denoted Simulated Plant. As mentioned previously, the Physical Plant refers to a separation into two programs, the RoboMaster and the RoboSlave, which run on the Robostix boards, whereas the Simulated Plant, realized in form of the MockJAviator, can be run on any JVM-providing machine that is connected to the control system.

5.2.1 RoboMaster

The RoboMaster periodically polls the IMU and the URF for new values and buffers them locally for remote requests by the FCS, which are remitted in form of packets containing new motor signals. The rotation speed of the four motors is controlled by generating individual PWM signals according to the received motor signals. Each packet with motor signals also contains a sequence number issued by the FCS, which is incremented by the RoboMaster and returned to the FCS with the next packet of new sensor data. Due to this mechanism, the FCS is able to identify a lost packet, and if so, displays a corresponding error message in the associated console window.

The physical connection between the RoboMaster and the FCS can be set up to use either the RS232 port or the SPI port, whereas the physical connection to the RoboSlave is established via an emulated 8-bit parallel port (see Subsection 3.5.3). The default connection type between the RoboMaster and the FCS is an RS232 port, because the corresponding UART interface is operated in asynchronous mode and thus provides more reliability in regard to the limited computational resources of the ATmega128 processor. However, the supplementary implementation of the SPI interface as an optional connection type offers the freedom to make use of one more available RS232 port if required.

5.2.2 RoboSlave

The RoboSlave manages the auxiliary sensor suite, employed primarily for experimental reasons, which currently consists of the BMU and the two LDSs. In this context, the RoboSlave can be utilized for handling any type of sensor, provided that compatibility to one of the available Robostix ports is given. Analogous to the RoboMaster, the RoboSlave periodically polls the attached sensors for new values and buffers them locally for remote requests by the FCS. Since the LDSs are the most power-consuming devices of all sensors connected to the JAviator Plant, the RoboSlave implements a mechanism for disabling the LDSs in case of a shutdown message or upon connection loss.

5.2.3 MockJAviator

The MockJAviator is a simulation of the JAviator platform, which receives the motor signals from the FCS and, in response, computes realistic sensor values according to an

estimation of the JAviator's flight dynamics. For the purpose to ensure compatibility with the FCS, the device-independent aspects of the JAP-FCS Interface implemented in the MockJAviator are the same as in the RoboMaster program. However, instead of an RS232 serial interface, the MockJAviator uses either a User Datagram Protocol (UDP) or Transmission Control Protocol/Internet Protocol (TCP/IP) connection to communicate with the FCS. For this reason, the device-dependent aspects of the JAP-FCS Interface are implemented in the FCS not only on RS232 but also on Ethernet. An advantage of this design is that the MockJAviator can run either locally on the same machine as the FCS or remotely on any other machine.

5.2.4 JAP-FCS Interface

The JAP-FCS Interface defines the communication protocol between the JAP and the FCS, and according to the underlying policy, only the FCS can initiate the communication among them. The FCS requests new sensor data by sending a packet to the JAP, which contains new motor signals, and in response to this packet, the JAP generates a packet containing the latest sensor data and sends it to the FCS. Besides immediately stopping the motors in response to a shutdown message, the JAP keeps track of the time when packets are received and slowly revs down the motors for safety reasons if packets have not been received within a period of 100 ms.

5.3 Flight Control System

All FCS versions implement the LET model for timing the control cycle and the same control algorithms for piloting the JAviator. The control cycle inherent in each FCS version can be summarized as follows: within a sampling period of 15 ms (66.67 Hz), the FCS receives sensor values from the JAP and the SLS as well as reference commands from the GCS, and in response to these data, computes new motor signals that are sent back to the JAP. Note that the only sensor values not delivered by the JAP are the position data provided by the SLS, which are received asynchronously via a separate UDP channel that connects the FCS directly with the Location Engine. When connected to the GCS, the FCS sends a complete report to the GCS at the end of each control cycle, which contains the current sensor data, motor signals, and specific trace data.

As already mentioned in Subsection 5.1.3, there are four alternative implementations of the FCS: JControl, EControl, CControl, and TControl (see Figure 5.4). CControl can be seen as the baseline controller that is used for any kind of experiments concerning different control algorithms, timing policies, and practical applications. It therefore represents a template for all other controller implementations in regard to the system-specific aspects. Currently, only CControl implements the most recent version of position flight control. Upon completion of the final tests and verification, this part of the control system will then be ported to the other controllers.

5.3.1 JControl

JControl implements the FCS in Java threads, which run on any JVM that supports Java version 1.4 or higher. The control algorithms, the timing, and the communication logic are all implemented in standard Java. Only five procedure calls to the RS232 serial-port driver require implementations as native methods in C. JControl is designed to enable code reuse by other components, for example, JControl's communication subsystem is reused in the EControl and GCS implementation. Moreover, EControl reuses JControl's code that implements the functional aspects of the control algorithms.

At runtime, JControl invokes three separate threads of execution, particularly, the controller thread that executes the control algorithm as well as two receiver threads in the communication subsystem. Both receiver threads are identical and handle incoming packets, except that one thread polls the JAP interface and the other one polls the GCS interface. If an incoming packet contains information needed by the control algorithm, a new Java object is generated and a reference to this object is stored in the communication subsystem. On the one hand, the controller thread retrieves the latest object at the start of each sampling period. On the other hand, packets that are not intended for the FCS are forwarded to the opposite communication interface.

Although JControl runs on any JVM with support of Java version 1.4, even more promising real-time results, in terms of latency and pause times, are achieved with the IBM WebSphere Real Time (WSRT) JVM [105, 106]. This effect is primarily due to the circumstance that the WSRT JVM invokes the Metronome garbage collector [107], which shortens pause times and thus enables predictable execution of user applications written in Java. The WSRT JVM has been selected, for instance, by the Raytheon Company as computing platform for the DD(X), the US Navy's future surface combatant ship program.

In this context, all of our test flights with JControl and EControl were conducted with the WSRT JVM modified for support of Exotasks.

5.3.2 EControl

EControl implements the FCS with Exotasks, which facilitate real-time programming in Java. Exotasks are isolated communicating tasks that execute in real time. Determinism is achieved by enforcing spatial isolation of each task and applying the LET model [102] for communication among the tasks. Spatial isolation prohibits different tasks from sharing any state, and therefore, allows to individually garbage-collect the tasks. Communication between tasks is handled by sending messages via dedicated channels, which are managed by the Exotask runtime system. Such messages are delivered at predefined logical instants of time in accordance with the LET model and controlled by a scheduler that is part of the Exotask runtime system.

Currently, the Exotask runtime system supports a Giotto-like scheduler [102] and a Hierarchical Timing Language (HTL) scheduler [108]. Giotto is a coordination language for distributed control systems based on the LET model, whereas HTL can be seen as

the most recent successor of Giotto. With the Giotto-like or HTL scheduler, the Exotask system provides deterministic timing across changes of both the hardware and software platform, even in the presence of other Java threads. Note that the Exotask environment also provides a scheduler framework that allows for arbitrary scheduler implementations that may not incorporate the LET model.

The tasks of a program in combination with their communication links result in a graph, where the nodes represent the tasks and the edges represent the communication connections. The graph of an Exotask program, called the Exotask Graph, is implemented either in Java source classes that follow a certain design pattern or via a graphical editor that is shipped with the Exotask development system as an extension to Eclipse.[1] While creating an Exotask Graph, the graphical editor performs additional error checks and, after successfully validating the graph, generates Java source code.

For a detailed description and performance analysis of Exotasks, the reader is referred to our collaborative work with IBM Research presented in [3, 4].

5.3.3 CControl

CControl implements the control algorithm in a multi-threaded Linux process. In contrast to JControl, which also uses a communication subsystem for performing all I/O-related tasks, CControl invokes four separate threads of execution. Particularly, the controller thread executing the control algorithm and three receiver threads in the communication subsystem. This is due to the fact that CControl is currently the only FCS version that implements position flight control, which demands an additional receiver thread for the positional information delivered by the SLS.

The control loop is started after the communication channels have been setup and initiated successfully. CControl implements a high-resolution timer that is involved in processing all timed operations, such as triggering the timely start of each sampling period. The control cycle begins with sending the computed motor signals to the JAP, followed by reading the latest sensor values and reference commands from the communication subsystem. The next step consists of rotating all values that need to be transformed from body-fixed to earth-fixed magnitudes as well as computing the derivatives of specific values. After applying digital filters to some measurements, the state estimator is invoked for generating the attitude, position, and velocity estimates. The process continues with passing all values and estimates to the motion control that computes new motor signals, which are sent to the JAP via the communication subsystem at the beginning of the next control cycle. After completion of the control algorithm, checking for a requested mode transition, and sending a report of all values to the GCS, the control loop calls a sleep procedure with the start of the next sampling period as argument.

Concurrently, the communication subsystem, which is permanently active, sends all outgoing messages through the correct connections and polls the incoming channels for new packets. If any packets have arrived, they are parsed and data intended for the control

[1] Eclipse, produced by the IBM Corp., is a popular integrated development environment for Java.

algorithm are stored in dedicated buffers. Packets received from the GCS and intended for the JAP are forwarded without modifications, and similarly, packets received from the JAP and intended for the GCS are also forwarded without modifications.

CControl implements a special framework to abstract the system-dependent aspects of the controller. This design supports the execution on different platforms by changing only the low-level implementation of the system-dependent modules, for instance, the I/O operations of the communication subsystem and the timing operations.

5.3.4 TControl

TControl is a port of CControl to our own real-time operating system called Tiptoe [109, 110]. The current version of TControl is hard-wired into the Tiptoe kernel for lack of support in regard to user processes. Nevertheless, we were able to implement the FCS with Tiptoe, run it natively on the Verdex, and fly the JAviator.

Analogous to CControl, TControl connects to the JAP physically via an RS232 serial port. However, the Tiptoe kernel does not support any type of Ethernet connection, which is required to communicate with the GCS. Instead, Tiptoe multiplexes all network traffic in real time in custom-designed Ethernet frames onto a dedicated Ethernet connection to another Verdex running our real-time-enhanced version of Linux. This host machine demultiplexes all incoming traffic and relays it to a regular-wired or wireless Ethernet connected to the GCS [111]. The other direction works similarly.

We currently work on a Tiptoe version that will offer full support of user processes. TControl will then be implemented in the so-called Workload-Oriented Programming (WOP) model [112] of Tiptoe, which abstracts concurrency-dependent process execution times into concurrency-independent process response times. More precisely, WOP is a design methodology for specifying throughput and latency of real-time software processes on the level of individual process actions. The key programming abstraction is that the workload involved in executing a process action, such as a system or procedure call, fully determines the action's response time, independently of any prior or concurrent actions. The WOP model thus enables sequential and concurrent real-time process composition while maintaining each action's workload-determined real-time behavior.

We plan to implement TControl in the WOP model by specifying timing constraints on individual actions of the controller. Actions such as reading sensor values and updating motor signals have latency-oriented timing requirements while log and trace data usually need throughput guarantees. In the WOP model, we are able to specify these different constraints within the control cycle itself and guarantee the correct timely execution of each controller action on top of Tiptoe.

5.4 Ground Control System

The GCS consists of the Control Terminal application used for piloting the JAviator and displaying its status, the TuningFork real-time logging system, which collects trace data

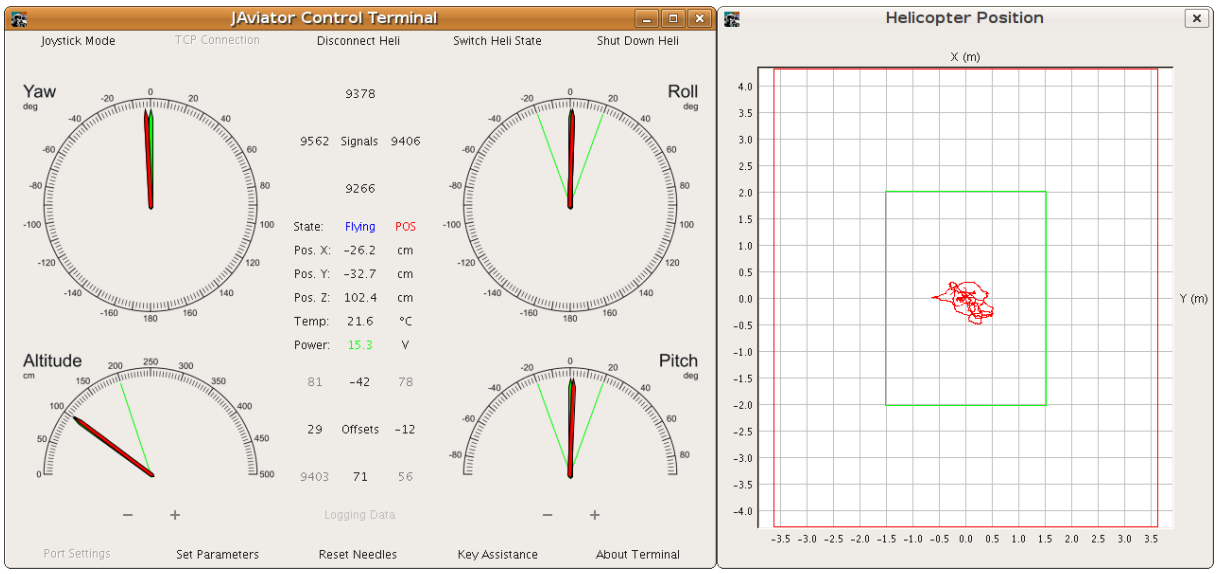


Figure 5.5: Screenshot of the Control Terminal with optional Position Monitor.

received from the onboard computer system, and the Location Engine configuration tool, which localizes the JAviator's position and sends it to the FCS.

5.4.1 Control Terminal

The Control Terminal, shown in Figure 5.5 (left), displays the sensor data received from the FCS graphically and provides a means for piloting the connected JAP. Note that it does not make a difference if the Physical Plant or the Simulated Plant is connected to the Control Terminal. The physical connection between the Control Terminal and the FCS is established by a 2.4-GHz wireless LAN router, which is connected to the GCS via regular Ethernet and to the onboard computer system through wireless Ethernet. The protocol can be chosen to be either UDP, which corresponds to the default setting, or, optionally, TCP/IP, which turned out to be less reliable for periodic data transmissions due to its packet-oriented transfer mechanism.

The main functionality lies in monitoring and modifying the JAP's attitude, position, and velocity. Specifically, "modifying" refers to changing either the JAviator's attitude and altitude in case of 4-DOF manual flight control or its position and azimuth in case of 6-DOF position flight control. This is accomplished with four analog meters that indicate the "desired" and "current" roll, pitch, yaw, and altitude values via green and red needles, respectively, which facilitate manually piloted flights. Particularly, the user controls the connected JAP by changing the green needles via a 4-axis joystick. Since the red needles refer to the JAP, they cannot be influenced by the user. Except for the yaw meter, each meter contains a safety option that allows to set a limit for the green needle. The Control Terminal also displays the following FCS information: *Signals* represent the four computed values assigned currently to the motors; *Offsets* are composed of displaying both the relative motor offsets (differences between the previous and current motor signals) and

the partial motor offsets (control effort of the roll, pitch, yaw, and altitude controller); *State* indicates the state of the FCS (Ground, Flying, or Halt); *Pos. X*, *Pos. Y*, and *Pos. Z* represent the JAP's current position numerically for the purpose to provide some position feedback even if no optional position visualization is used; *Temp* and *Power* refer to the air temperature and battery voltage, respectively.

The sensor values and motor signals received from the JAP can also be viewed in form of sliding curves by activating the Signals Dialog. This dialog contains several diagram views referring to specific sets of data, which can be selected via clicking the tab of the view of interest. In case of selecting the attitude-altitude view, the values are displayed together with the generated reference commands for facilitating in-flight studies of the controllers' individual tracking behavior. For visual support of position-controlled flights, the user may optionally involve the Position Monitor, shown in Figure 5.5 (right), which displays the desired and current position of the craft by drawing a green and red trace, respectively. If the Position Monitor is active, depending on whether the flight mode is set to position control, the user generates horizontal position commands by moving the green point either with the joystick or, preferably, with the arrow keys of the keyboard. Referring to the Position Monitor shown in the figure, note that the green point is covered by the red trace. This is due to a test flight that was conducted with the real JAviator without changing the initial position command after lift-off, which caused the craft to hover over the starting point. Similar to the analog meters, the Position Monitor contains a safety option that allows to restrict the area of operation. Specifically, the green rectangle indicates the limits with respect to position commands, whereas the red rectangle represents the effective area captured by the SLS antennas.

In order to provide a more advanced visualization and interaction with the connected JAP, the Control Terminal has been extended with a 3D environment. This feature enables visual comparisons of the real JAviator against the computed flight behavior. Along with the 3D environment comes the opportunity to view a flight from different perspectives in the 3D scene, freely adjustable via the rotation, scaling, and translation settings. In this sense, the 3D representation extends the navigation environment by offering full 3D motion tracking and close-to-reality visualization effects (for screenshots of the 3D environment see [1]). For the purpose of studying different controller settings, the Control Terminal has also been enhanced to support recording and replaying of conducted flights. As a consequential benefit from the incorporation of the 3D environment, any recorded flight can be visualized almost authentically while replaying.

Another beneficial feature is the Settings Dialog, a means that allows to dynamically adjust the controller parameters. In other words, each parameter of each controller can be changed arbitrarily during flight without the need for landing and re-programming. Lastly, the probably most important feature of the Control Terminal is the built-in logging mechanism, which can be considered as an alternative to the TuningFork logging system. If logging is enabled, all sensed measurements, computed signals, and issued reference commands are saved to a text file, which can then be interpreted by every application that supports the comma-separated file format.

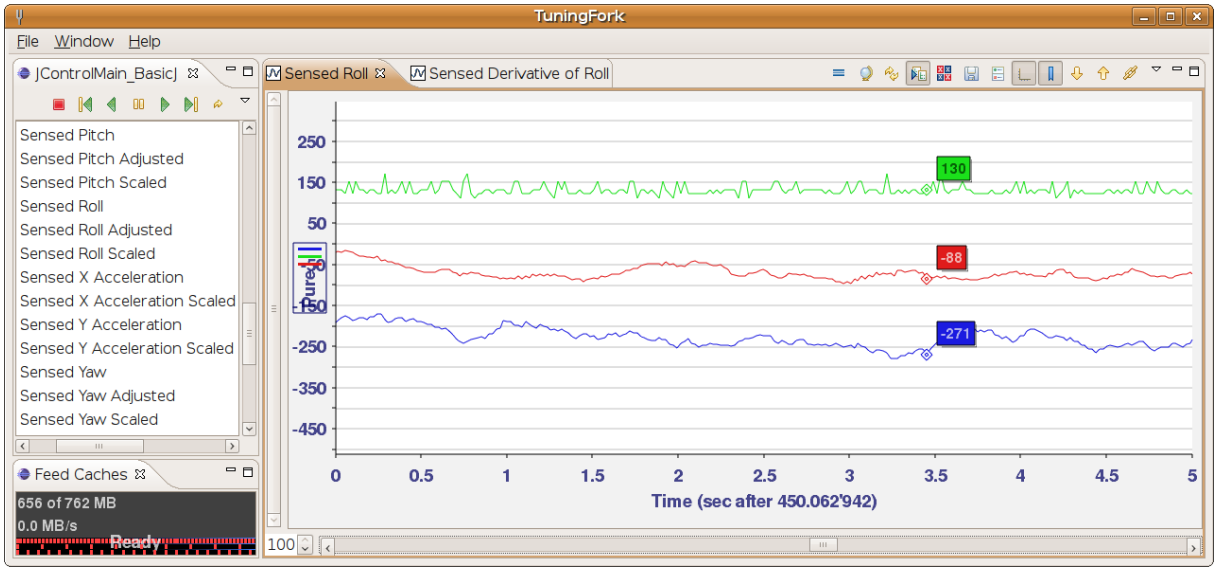


Figure 5.6: Screenshot of the IBM TuningFork real-time logging system.

Besides the hardware-based shutdown mechanism (see Subsection 3.3.1), the Control Terminal provides three forms of triggering a software-based shutdown: first, via clicking the "Shut Down Heli" button, second, via pressing a predefined hot key, and third, in case of manually piloted flights, via releasing the joystick's "fire" trigger, which must be pulled permanently during flight for suppressing the Halt mode.

5.4.2 TuningFork

In order to study and improve the flight behavior and system performance, the GCS has been instrumented with a logging system that enables real-time tracing during flights. This logging system makes use of IBM's TuningFork [103], an Eclipse-based performance analysis and visualization tool for real-time applications with support for Java, JVM, C/C++, and Linux. TuningFork was originally designed for diagnosing IBM's real-time JVM and real-time Linux, and therefore, features a powerful data processing mechanism and graphical user interface. Nonetheless, because of its scripting capabilities, it can also be applied for tracing user-specified system data. This is basically accomplished by incorporating a couple of class files that refer to the trace mechanism and augmenting the source code with trace-dependent TuningFork instructions.

Referring to the JAviator software system, connectivity within the GCS environment is established via sockets. All logged data (time-stamped program variables) and events are saved in a single trace file together with the current controller parameters. Due to the additional parameter saving, it is possible to restore the exact controller setup for any logged flight. We are interested in both the system-specific values, such as memory mutator utilization and garbage collection activity, as well as the sensor and actuator data transmitted from and to the JAP. The TuningFork logging system thus allows us to analyze and interpret various data streams in both real-time and hindsight.

5.4.3 Location Engine

The Ubisense Series 7000 SLS uses a cellular architecture to cover even large areas by tiling them into groups of cells, where each cell consists of a small number of antennas working together. The activities of all antennas within a group are managed by the master antenna, which forwards the individual measurements to the Location Engine. Besides the ability to collect the location data of several tags from multiple cells, the Location Engine serves as a tool to configure and calibrate the sensing environment.

The Series 7000 antennas support two-way control and telemetry communication with the tags over a conventional bi-directional 2.4-GHz radio channel. Due to this full-duplex capability, the Location Engine can be applied to dynamically change the sampling time of a tag, query status information like a tag's battery level, and remotely re-program a tag to enable specific features. The sampled position information is delivered to the receiving systems via registered UDP channels that are managed by the Location Engine. In this context, Ubisense provides an API that allows to incorporate all required functionalities to define and communicate information in the Ubisense-specific data format.

5.4.4 GCS-FCS Interface

The GCS-FCS Interface defines the communication protocol between the GCS and the FCS, and similar to the JAP-FCS Interface, only the FCS can initiate the communication among them. The FCS requests new reference commands by sending a packet to the GCS, which contains the current sensor data, and in response to this packet, the GCS generates a packet containing the latest reference commands and sends it to the FCS. Note that the sensor data received from the FCS do not refer to measurements delivered by the JAP that were forwarded to the GCS, instead they refer to the rotated, filtered, and estimated FCS values that are passed to the motion control.

Because the wireless Ethernet connection between the GCS and the FCS is much more error-prone than the hard-wired RS232 or SPI connection between the JAP and the FCS, the GCS-FCS Interface implements a mechanism for re-connecting. In case of connection loss, the FCS, which is the only layer that can initiate communication, tries to re-establish the connection until a certain threshold is reached. If the re-connect attempt was successful, merely an error message is displayed in the FCS's console window. If the connection could not be re-establised, the GCS signals a loss of connectivity and the FCS enters the Halt mode, which triggers a shutdown message sent to the JAP.

Chapter 6

System Performance

The idea is to try to give all the information to help others to judge the value of your contribution; not just the information that leads to judgment in one particular direction or another.

Richard P. Feynman (1918−1988)

This chapter presents some of the many experiments that have been conducted to verify and substantiate the introduced solutions. Specifically, Section 6.1 is dedicated to the JAviator's electromechanical capability in regard to lifting capacity and flight endurance. The achieved control response behavior is discussed in Section 6.2, accompanied by several diagrams for illustrating the control system's 4-DOF and 6-DOF characteristics. Finally, the control software's portability with respect to timing issues is addressed in Section 6.3. In this scope, it is shown that the system preserves its temporal behavior when executed on different hardware platforms.

6.1 Platform Capability

One of the major goals concerning platform development was to make the JAviator as robust as possible and, at the same time, achieve a competitive weight. The so-emerged extraordinary airframe design outperforms most of the existing ones with an incredibly high mechanical integrity and enormous robustness in case of crashes. We did not conduct explicit crash tests, because of limited financial resources, but unintentionally experienced them uncountable times over the past years in the scope of exhaustively tuning and testing the control system and software. Besides several video records reporting on this, many eye-witnesses saw the JAviator crashing, sometimes on grass, sometimes on concrete, and sometimes straight into a wall, and after a short pause for a quick inspection, saw it lifting off again without any noticeable degrade in flight performance.

If the JAviator falls, for example, due to a triggered shutdown, from a height of about 2 m and hits the ground at an angle exceeding 20 deg, than usually one or more of its

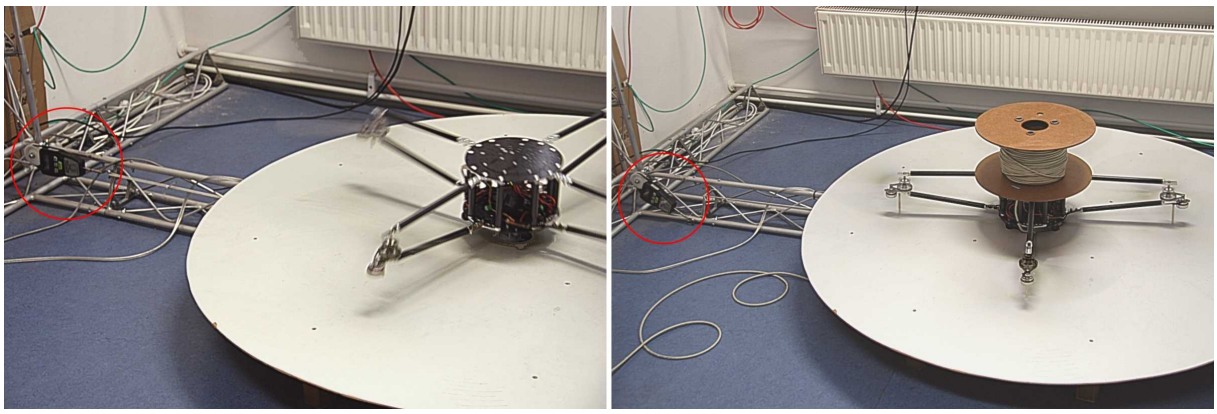


Figure 6.1: Determination of maximum lifting capacity (left) by connecting the craft to a high-precision spring scale (red-encircled) and determination of maximum battery service time (right) by running the rotors at full speed until the battery is fully drained.

rotor blades will need to be replaced, because in such case, they will mostly bend or even break at the blade root. Occasionally, it happens when crashing very hard that a girder's top tube splinters under the incurred pressure. This is due to the fact that the top airframe actually needs to absorb most of the shock wave, which is normally induced over one of the four nylon motor protection rods and then transferred via the rotor shaft into the top girder tube. Nevertheless, the JAviator endured uncountably many collisions without serious damage, and in this sense, has proven its robust nature.

6.1.1 Lifting Capacity

In the endeavor to verify our estimations regarding thrust generation and flight times, we conducted several experiments with a fully assembled JAviator. The motors and the entire avionics equipment were powered during all tests via a single, onboard Thunder Power 4S3P 14.8-V 6300-mAh lithium-polymer battery (see Section 3.3). The JAviator weighs a little less than 2.2 kg in its ready-to-flight configuration. We tested two different gearings, particulary, 1:6 and 1:5, in order to reveal the propulsion system's maximum capability. To do so, we placed the JAviator on a round launch pad with a small hole in the middle, which we originally used for tethered flights. Thrust was then measured with one end of a nylon rope attached to the center of the JAviator's bottom fuselage plate, put through the hole in the launch pad, guided over a pulley, and the other end attached to a high-precision spring scale. The rope had a play of approximately 10 cm, hence allowing the JAviator to lift off and hover freely, as shown in Figure 6.1 (left). Due to this setup, we were able to measure the thrust generated in addition to lifting the craft's empty weight. The maximum thrust for both the 1:6 and the 1:5 gearing was determined by letting the JAviator pull the rope with maximum power, but, for safety reasons, for not longer than 10 s. During this time span, the spring scale was oscillating around some mean value that can be found in Table 6.1, which summarizes the conducted propulsion tests. Note that it makes a significant difference if a helicopter's vertical thrust

is determined *in* or *above* ground effect. According to Leishman [5], the ground effect may account for up to 25 % of a helicopter's lifting force under ideal conditions. Due to this, the results provided in the tables of this section are presented in form of both measured values (indicated in brown), which were obtained in ground effect, as well as estimated values (indicated in blue), which refer to flying above ground effect.

Compared to many other small-scale quadrotor UAVs that rarely provide remarkable lifting capacities, it points out that the JAviator is capable of lifting more than its empty weight, independently of the gearing used, when flying in ground effect. Though originally not intended, it is amazing to see the JAviator hovering over the ground while carrying another fully equipped JAviator mounted beneath its bottom fuselage plate, which was demonstrated in the scope of performing test flights with high payload.

Table 6.1: Lifting capacities resulting from different gearings and thrusts separated into measured values (brown) that were obtained in ground effect and estimated values (blue) that refer to flying above ground effect.

Experiment	Gearing	Thrust (%)	Force (kg)
Craft without payload	1:6	43/54	2.2
Craft plus lifting 2.4/1.3 kg	1:6	100	4.6/3.5
Craft without payload	1:5	38/48	2.2
Craft plus lifting 3.2/1.9 kg	1:5	100	5.4/4.1

6.1.2 Flight Endurance

After determining the maximum lifting capacities, some heavy weight was placed on top of the JAviator to prevent the craft from lifting off during the flight time tests, as shown in Figure 6.1 (right). The flight endurance of any VTOL is proportional to its available energy resources, and in case of an electrically powered helicopter, directly related to the service time of the employed battery. In order to determine the battery service times for both gearings and for both the JAviator's empty weight with and without maximum payload, we conducted four different endurance tests. Moreover, for each of the four tests the battery was fully drained until one of the motors dropped off, in other words, was shut down by its motor controller.

The results obtained from these experiments and the corresponding estimates that refer to flying above ground effect can be found in Table 6.2. As apparent from the table, even though the service times drop significantly when demanding full thrust, the JAviator can provide both either high payload capacities or relatively long flight times. Regarding the measurements without payload it is noticeable that, compared to the 37-min service time achieved with the weaker 1:6 gearing, the same measurement performed with the stronger 1:5 gearing resulted in a service time of 39 min. We assume that this result is due to certain tolerances associated with the charger, equalizer, and/or battery. Note that each

full drain of a lithium-polymer battery considerably reduces its life cycle. We therefore did not repeat these experiments for the purpose to reproduce the results. However, we could prove average flight times around 30 min through frequent flights with moderate payload in the past years and even achieved an endurance of nearly 40 min while hovering in ground effect at a height of approximately 1 m.

Table 6.2: Battery service times resulting from different gearings and thrusts separated into measured values (brown) that were obtained in ground effect and estimated values (blue) that refer to flying above ground effect.

Experiment	Gearing	Thrust (%)	Time (min)
Craft without payload	1:6	43/54	37/28
Craft plus lifting 2.4 kg	1:6	100	11
Craft without payload	1:5	38/48	39/29
Craft plus lifting 3.2 kg	1:5	100	8

6.2 Vehicle Controllability

From the perspective of control system design, the goal was to achieve the best possible controllability to be able to indisputably identify any deteriorations in the flight behavior that are caused by the software system.

Platform Configuration. All performance measurements presented throughout the rest of the chapter were obtained with the same JAviator platform, which was equally configured for all experiments. Specifically, the craft's attitude was determined with a MicroStrain 3DM-GX1 IMU, its vertical position with a SensComp Mini AE URF, and its horizontal position with a Ubisense Series 7000 SLS, as described in Section 3.4 (hardware details) and Section 4.2.1 (sampling details). Since the JAviator lab allows for a maximum flight altitude of about 2.5 m, the URF was chosen for determining the vertical position. This decision was due to the fact that the URF provides measurements at much higher frequencies than the BMU for short distances not exceeding 2 m, and consequently, yields more accurate altitude tracking when flying at modest heights. The SLS was chosen for determining the horizontal position, because the LDSs turned out to be a rather impracticable option when it comes to position-controlled flight. Although their measurement precision clearly dominates over the RFID-based SLS, their relatively high average latency allows for holding a certain position, but renders them useless as soon as the speed of horizontal movements increases. Regarding control software, the CControl C-based FCS version (described in Section 5.3) was applied, which is currently the only one that also supports position flight control. Though 12 ms is the fastest sampling period that works properly, the default setting of 15 ms was used for all flights. The IMU requires

Figure 6.2: Indoor flights performed within an area of about 4×4 m inside a classroom at the University of Salzburg (left) and within an area of about 5×3 m inside a corridor at the Charles University of Prague (right).

10 ms for computing the Euler angles and another 2 ms are needed for communicating the generated data from the IMU to the RoboMaster and then transmitting it to the FCS. Accordingly, 12 ms suffice to receive new sensor values from the JAP periodically without data loss. However, we observed that a sampling period of 15 ms does not significantly degrade the flight behavior but offers additional time reserves for intercepting messages, like mode switches, which otherwise occasionally lead to losing one or more data packets containing sensor values.

Environmental Setup. The JAviator lab is located in the basement of a building next to the Department of Computer Sciences at the University of Salzburg. The flight area that is captured by the Ubisense Series 7000 SLS has a dimension of around $8 \times 7 \times 3$ m and is surrounded by windowless walls to all sides. The masonry of these walls consists of ferroconcrete that acts as a Faraday cage interfering with the Earth's magnetic field, which is measured by the IMU for determining the craft's absolute attitude with respect to the fixed Earth. The magnetic field within the flight area changes for a total of 180 deg when moving from either side straight to the opposite one. These strong magnetic interferences posed a totally unexpected challenge for flying in the JAviator lab.

6.2.1 4-DOF Characteristics

Besides sufficiently accurate sensors, the 4-DOF flight performance primarily depends on the controllers' characteristics. Due to the sophisticated controller design, in general, the JAviator demonstrates excellent control response and resulting maneuverability. This allows to precisely navigate the JAviator when manually piloted, even in very limited space. For example, Figure 6.2 (left) shows the JAviator during a flight inside a partially cleared-out classroom at the University of Salzburg, Austria, and Figure 6.2 (right) shows the JAviator during a flight inside a small corridor at the Charles University of Prague, Czech Republic, which was performed in the scope of an invited talk given at the JTRES

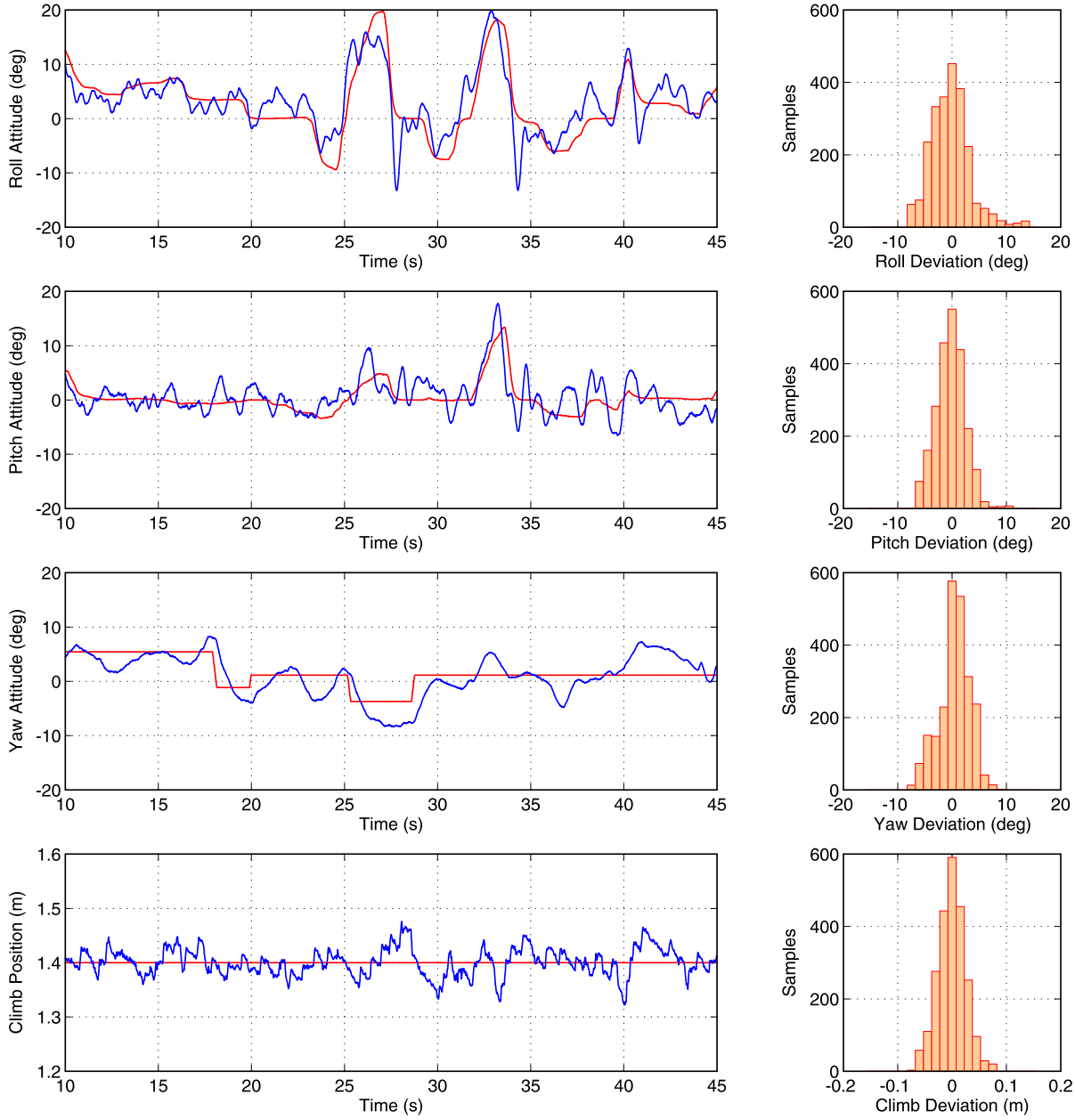


Figure 6.3: Aggressive-commanding sequence issued during a manually piloted flight and corresponding histograms depicting the specific error distributions.

2010 workshop [113]. Nevertheless, the more interesting question is, how well performs the JAviator when issuing rather extreme control commands?

Figure 6.3 depicts a flight sequence resulting from aggressive control commands. As apparent from the figure, the climb reference was kept constant at 1.4 m, whereas the attitude reference was largely varied in the range of ±20 deg by simultaneously moving the joystick about all three axes, hence not letting the craft settle at some specific attitude. The record shows that the JAviator perfectly tracks the reference commands without serious overshoots or any oscillating behavior. Despite these rapid changes in the craft's attitude, the desired climb position is kept within ±5 cm on the average.

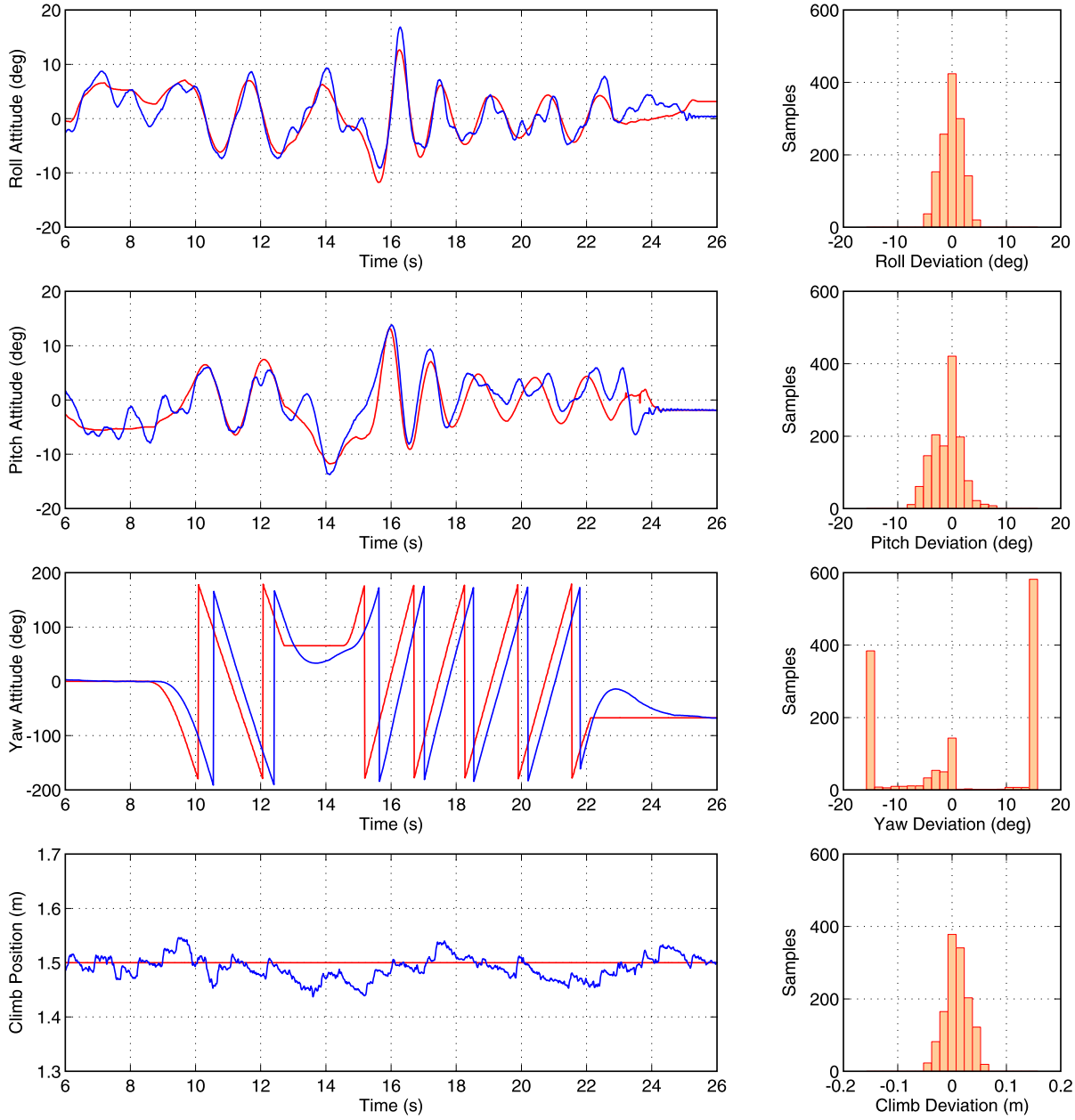


Figure 6.4: Fast-left-right-yawing sequence issued during a manually piloted flight and corresponding histograms depicting the specific error distributions.

Due to the applied spin limit extension, the JAviator is able to continuously spin about its vertical axis and, at the same time, preserve full maneuverability. Note that for human-piloted aircraft this is solely possible at relatively low rotation speeds and practically only becomes necessary for a single-main-rotor helicopter in case of a tail rotor malfunction. However, the flight record depicted in Figure 6.4 reveals that the JAviator is able to spin during flight with up to 40 rpm while tracking the yaw command with a deviation around 15 deg. In our effort to reach a rotation speed of one full rotation per second, we tested various controller settings, but realized that the maneuverability lessens rapidly when spinning with more than 50 rpm.

In any case, the achieved ability to preserve maneuverability while spinning about the vertical axis undoubtedly proves two essential properties of the JAviator: first, the control system's noteworthy responsiveness to extreme control commands, and second, the platform's excellent electromechanical capability of tracking such commands.

Remark 12. In order to avoid the same legend appearing in all figures that refer to flight records, we follow the commonly used notation of indicating the reference commands in red and the measured values in blue.

6.2.2 6-DOF Characteristics

In contrast to attitude stabilization that largely depends on the quality of the controllers, satisfactory position control primarily depends on the quality of the positional feedback. In case of the climb position, the available sensors, especially the URF, enable the JAviator to track the desired altitude with a deviation of merely a few centimeters. In case of the horizontal position, however, the available sensor suite does not allow for an analogous performance. This is due to the fact that both the LDSs and the SLS fulfill only one of the desired position-sensing properties. Specifically, the LDSs offers high accuracy in the millimeter range, but provide the measurements non-frequently with quite long latencies. Contrariwise, the SLS provides the measurements with a nearly constant and sufficiently high frequency, but offers an accuracy of merely 0.15 m in the very best case. As already mentioned at the beginning of this section, the LDSs turned out to be sheer impractical for performing reasonable position flight control. Though it is possible to hold a certain position within a circle of around 1.2 m, moving to a different position usually results in a flight behavior that appears to happen more randomly than controlled. Obviously, this behavior is due to the long measurement latencies, because for an extremely agile quadrotor helicopter it does not suffice to provide position feedback just every now and then to achieve positional stabilization. Consequently, since the only choice that was left suggested the SLS, the question arose, what is the best that we can expect by utilizing a localization system with an average accuracy around 0.3 m?

In order to get a basic understanding about the relation between positional accuracy and required signal quality, it is helpful to study the results of related projects, especially those achieved with the STARMAC II quadrotor. There are two reasons for this: first, this craft demonstrated considerable position control and path tracking, and second, it gives a coarse impression of what might be achievable with the JAviator's controller design, which is based on the controller model used for this craft. In 2007, Hoffmann *et al.* [30] demonstrated autonomous hover within a circle of 0.8 m performed indoors with the STARMAC II quadrotor. This 0.8-m accuracy was achieved with an overhead-mounted camera in conjunction with colored-blob tracking software, which provides updates at 10 Hz with an accuracy of 0.01 to 0.02 m. One year later, Hoffmann *et al.* [36] showed that STARMAC II is able to track a path with an accuracy of 0.1 m indoors and 0.5 m outdoors. The 0.1-m indoor accuracy was achieved with the aforementioned vision-based system at an increased update rate of 15 Hz, whereas the 0.5-m outdoor accuracy was achieved with

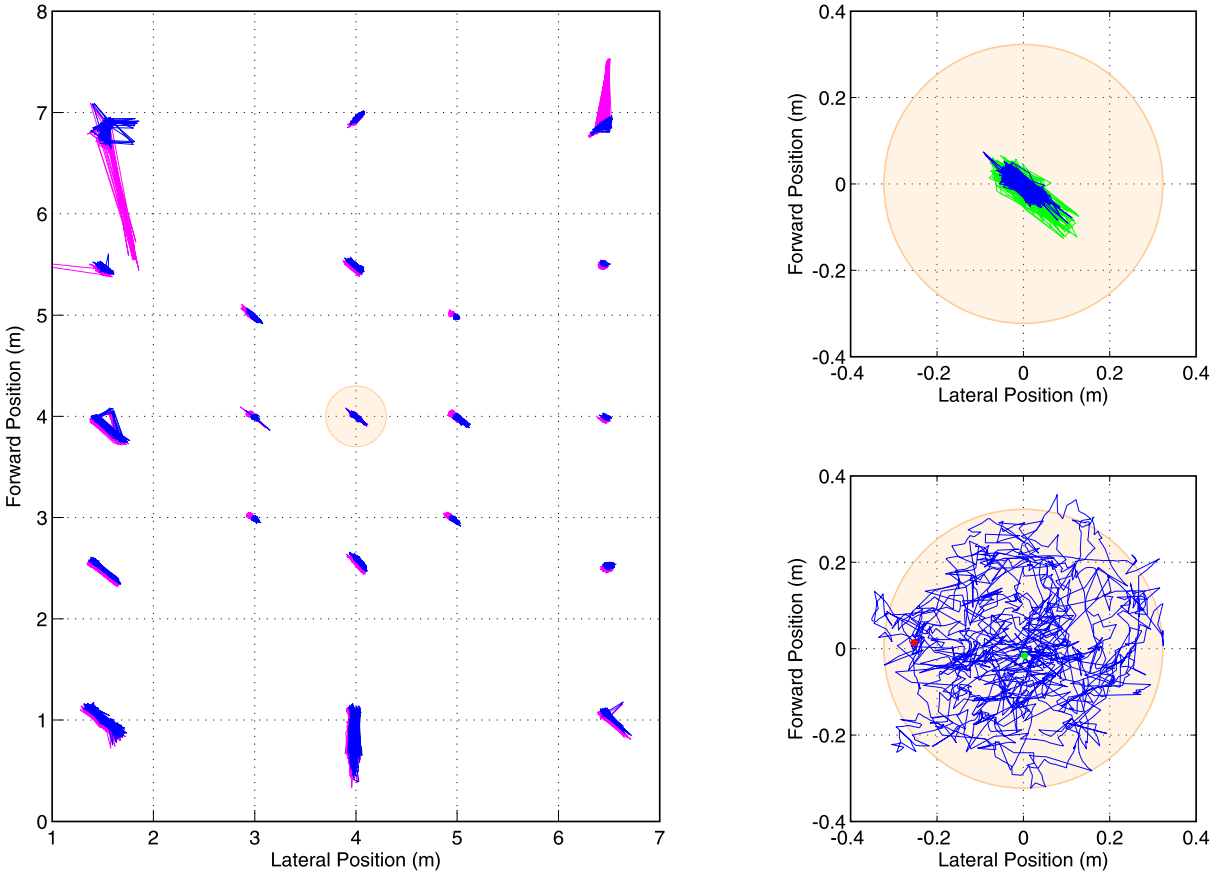


Figure 6.5: RFID-based localization of a non-moving tag placed at twenty-one positions (left) depicting original measurements (magenta) received from the SLS and estimated values (blue) generated by the EKF; position estimates referring to the tag located at the center point (right top) without incurred vibrations (blue) and with the craft's rotors spinning at nominal speed (green); 2D view of a position-hold flight (right bottom) with highlighted lift-off (green) and landing (red) position. The yellow-highlighted area in the diagrams indicates an error circle of 0.6 m.

a NovAtel Superstar II carrier-phase differential GPS receiver, which provides updates at 10 Hz with an accuracy of 0.02 to 0.05 m. According to these relations, given an update rate around 10 Hz, we can assume to achieve hovering within a circle that has a diameter of ten times the accuracy of the position measurements. Referring to the Ubisense Series 7000 SLS, which provides updates at 9 to 10 Hz with a best-case accuracy of 0.15 m, this assumption forecasts a circle with a diameter of 1.5 m. This forecast actually corresponds to the JAviator's hover behavior that was observed during the initial experiments without the EKF-based state estimator.

Figure 6.5 (left) gives an overview of the attainable signal quality in the JAviator lab within the area captured by the SLS. More precisely, the figure depicts the measurements obtained by placing a tag at twenty-one positions and gathering 1000 samples per location. The magenta-colored traces show the original values as received from the SLS, whereas the blue-colored values refer to the estimates generated by the EKF. The helicopter carrying

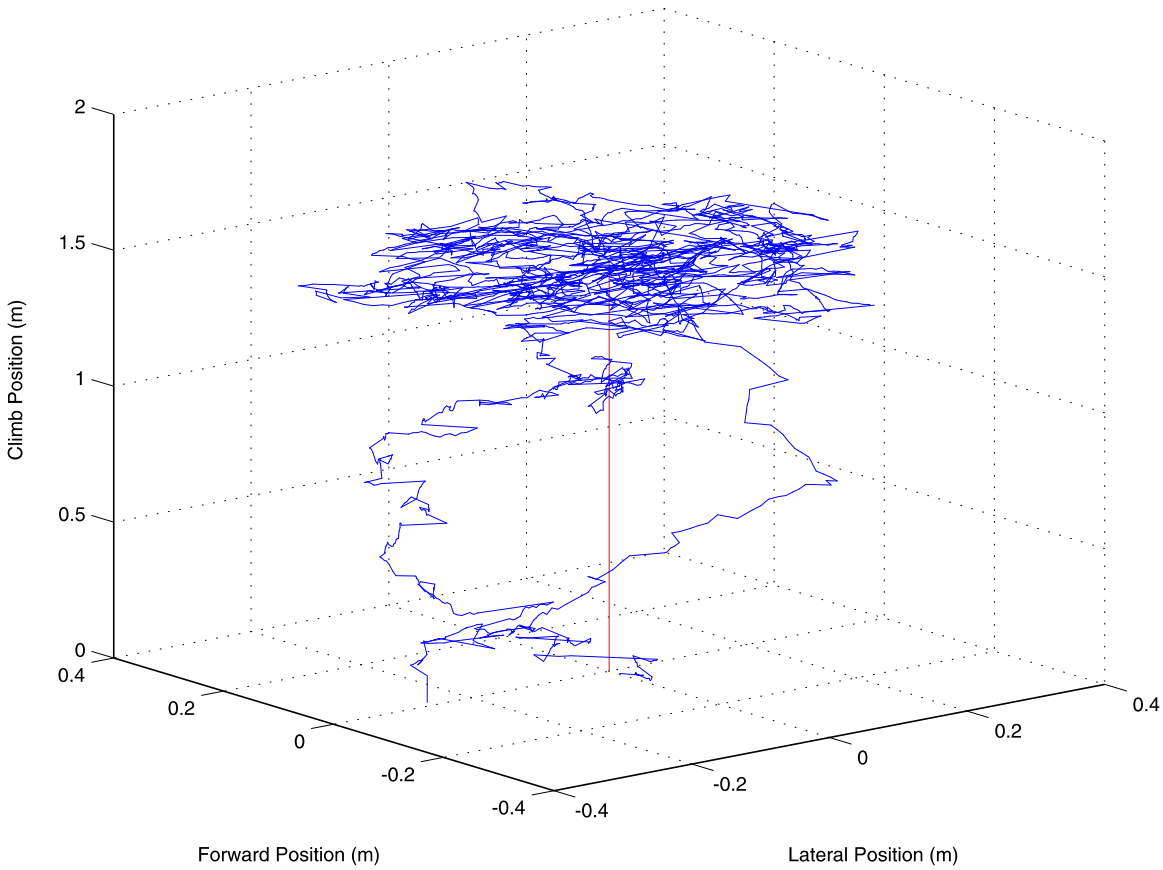


Figure 6.6: 3D visualization of the position-hold flight depicted in Figure 6.5.

the tag was not moved while sampling the positional information, and consequently, the linear-acceleration measurements received from the IMU were oscillating around zero and did not significantly influence the sensor-fusion-based estimates. As apparent from the figure, the signal quality attained is neither evenly distributed over the captured area nor highest in the center. There are locations to the right that stand out with accuracies similar or even better than in the center and there are locations to the left that significantly lessen in signal quality. Altogether, the EKF-based estimates yield an average accuracy of around 0.2 m over all twenty-one positions for a non-moving tag. (A thorough evaluation of the Ubisense Series 7000 SLS can be found in [114]).

The Ubisense Ubitag 7022 RFID tags are quite sensitive to vibrations, as illustrated in Figure 6.5 (top right), which depicts the position estimates referring to the tag located at the center point, without incurred vibrations (indicated in blue) and with the craft's rotors spinning at nominal speed (indicated in green). This effect causes the error in the position measurements to increase to about 0.3 m on the average. Surprisingly, despite this relatively low accuracy, it was possible without difficulty to achieve hover within a circle of less than 1 m upon integrating the state estimator. After many days of experimenting with different estimator gains and controller settings, finally, the position-hold accuracy could be improved significantly. In particular, it became possible to accomplish autonomous hover maintained within a circle of 0.6 m in the center of the area captured by the SLS.

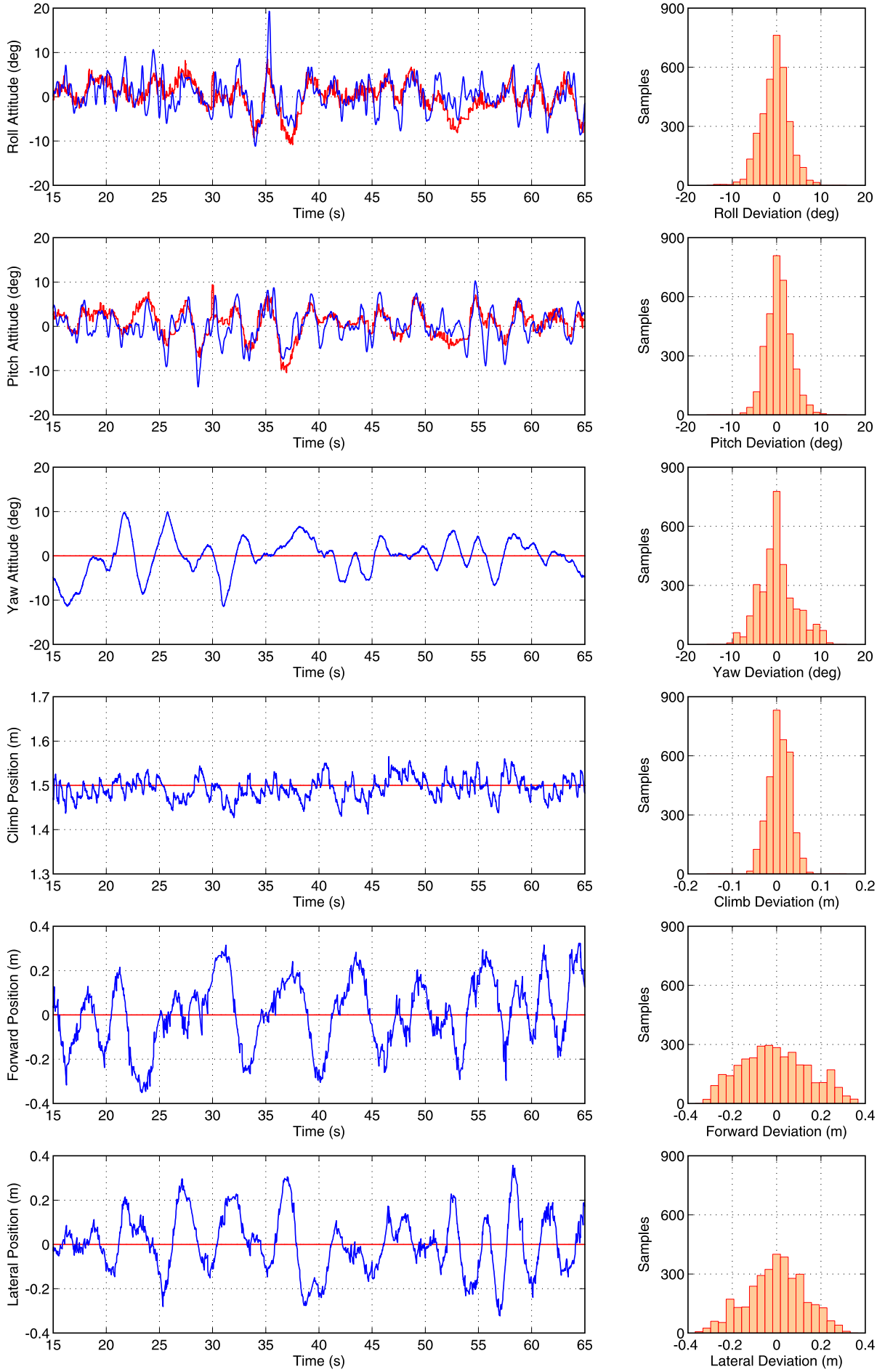


Figure 6.7: Hover sequence of the position-hold flight depicted in Figure 6.5.

This unexpected success is illustrated in Figure 6.5 (bottom right) depicting the position trace of a 1-min flight including lift-off and landing, as well as Figure 6.6, which presents a more tangible view of this flight in 3D visualization. The complete hover sequence of the corresponding flight record is given in Figure 6.7. Compared to the previous sequences referring to manually piloted flights, this sequence differs strongly concerning the roll and pitch reference commands. It is plain to see that these commands much more frequently change if generated by the position controllers.

We also conducted path-tracking flights in the JAviator lab, but so far could not achieve a satisfactory performance yielding a tracking error of less than 1 m on the average. There are mainly two reasons for this: first, the relatively instable RFID accuracy outside the center area captured by the SLS, and second, the large yaw drift caused by the ferroconcrete that interferes with the Earth's magnetic field. Although the yaw drift could be compensated to some extent by the attitude EKF, it could not be eliminated adequately to allow for reasonable waypoint navigation.

6.3 Time Portability

In the light of software architecture, the major goal was to provide a control system that is fully time-triggered and also time-portable in regard to changes of the computational hardware. Referring to the previous chapter, time portability was achieved by applying the LET model [102] to the control cycle to enable precise timing of the sensing and actuating processes independently of the underlying hardware platform. More precisely, new motor signals are transmitted from the FCS to the JAP at the beginning of each control cycle, followed by a transmission of new sensor values from the JAP to the FCS. In this sense, computed motor signals and sampled sensor values are not exploited as soon as they are available, rather they are accessed in LET-based fashion with constant frequency, which, in the present case, is defined by the sampling period.

In order to demonstrate that the JAviator's control software provides time portability across various hardware platforms, we executed the FCS on two quite different devices, operated the more powerful device with two specific kernel versions, and conducted all experiments under the constraints of both low and high I/O load. In particular, the software was executed on a Gumstix Verdex Pro XL6P running ARM-Linux with kernel version 2.6.24.7-rt21, on an IBM ThinkPad T60p laptop running Ubuntu 8.10 in real-time mode with kernel version 2.6.31-9-rt, and on the same laptop as before running Ubuntu 8.10 in generic mode with kernel version 2.6.27-17-generic.

For the purpose of proving time potability, there are two measurements of interest: first, the Signals Computation Time (SCT), which represents the complete interval required for executing a single control cycle including all computational tasks as well as all I/O-related tasks, and second, the Interarrival Time Deviation (ITD), which refers to the temporal deviation between the periodic execution of two consecutive control cycles. Independently of the SCT, which varies with the computational hardware, the variation with respect to

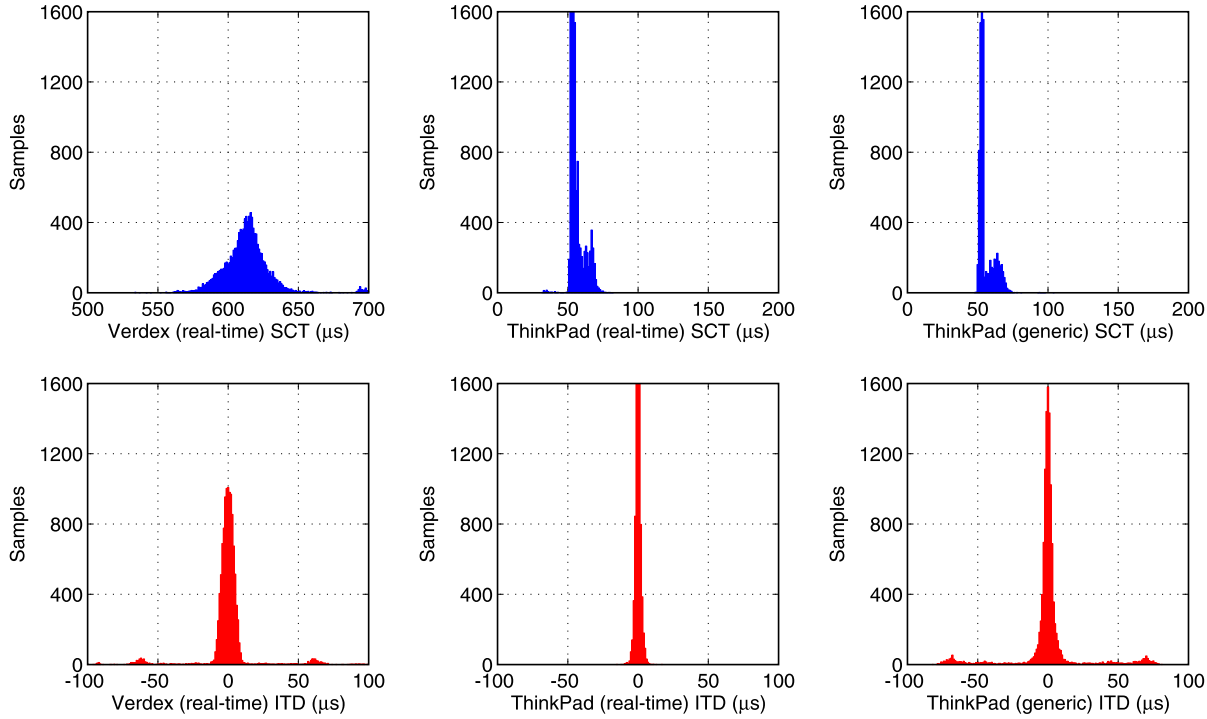


Figure 6.8: Low-I/O-load performance depicting SCTs (top row) and ITDs (bottom row) of the FCS executed on a Gumstix Verdex (left column), an IBM ThinkPad in real-time mode (middle column), and the same device in generic mode (right column).

the ITD should be as small as possible to preserve the temporal behavior of the system. Clearly, the bounds for the ITD depend on the timing constraints associated with the specific system. Referring to the JAviator control software, which runs with a default sampling period of 15 ms, we observed that even an ITD of ± 1 ms does not conspicuously affect the flight behavior. Consequently, an ITD at least two times smaller can safely be considered as being sufficiently small to guarantee unchanged flight behavior in regard to variations of the computational hardware.

6.3.1 Low I/O Load

The first series of experiments was conducted with low I/O load, which refers to the I/O load caused by the control system due to communication between the JAP, the FCS, and the GCS. This load comprises of transmitting motor signals from the FCS to the JAP, sensor values from the JAP to the FCS, all signals, values, and trace data from the FCS to the GCS, and reference commands from the GCS to the FCS.

The three histograms in the top row of Figure 6.8 depict the SCTs determined for the execution on the Verdex, the ThinkPad running a real-time kernel, and the same ThinkPad running a generic kernel. As can be seen from these histograms, the SCT yields a mean value of approximately 615 μs on the Verdex and 55 μs on the ThinkPad for both kernel versions. Even though the Verdex contains merely a single-core CPU clocked at 600 MHz

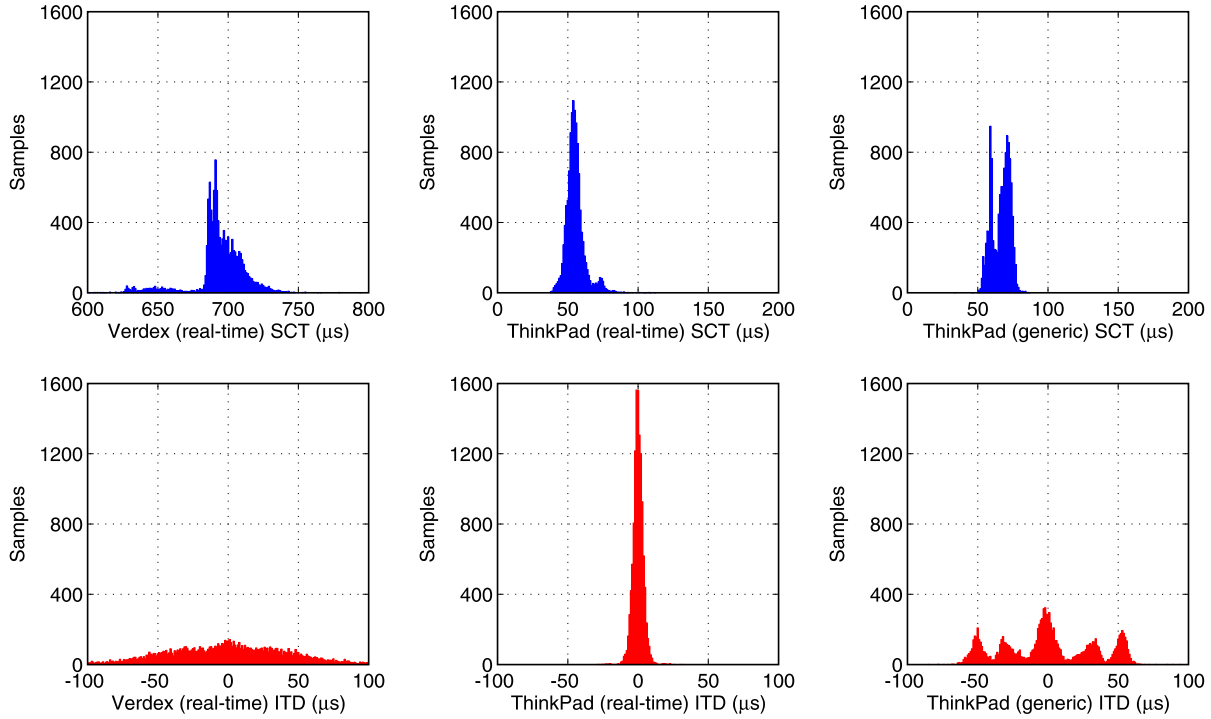


Figure 6.9: High-I/O-load performance depicting SCTs (top row) and ITDs (bottom row) of the FCS executed on a Gumstix Verdex (left column), an IBM ThinkPad in real-time mode (middle column), and the same device in generic mode (right column).

and no FPU, it performs unexpectedly well compared to the ThinkPad, which contains a dual-core CPU clocked at 2.16 GHz. Since the SCT is relatively short on the ThinkPad, it does not surprise that no significant difference occurred between the two kernel versions. However, a certain difference is plain to see when looking at the corresponding ITD histograms presented in the bottom row of this figure. Compared to the ThinkPad in real-time mode, which forms a sharp needle in the range of ± 5 μs, the ThinkPad in generic mode shows already some outliers that span the ITD over a range of ± 80 μs with a mean value around ± 10 μs. Due to the real-time extensions applied to the Verdex, its ITD performance, though taking on a similar shape than the ThinkPad in generic mode, spans a range of ± 70 μs with a mean ITD around ± 10 μs. This result can be considered as being extremely accurate in regard to the aforementioned computational limitations of the Verdex. In conformance with the obtained results, time portability is achieved with an average jitter of less than ± 30 μs for three different execution environments in the presence of low I/O load.

6.3.2 High I/O Load

Despite the fact that the experiments referring to low I/O load produced excellent results, the more interesting and important question is, how does the system perform under the constraint of high I/O load? To answer this question, the second series of experiments was

conducted with additionally generating considerable I/O load concurrently to executing the control software. More precisely, the Verdex was stressed by copying about 130 files, ranging from a few kilobytes to several megabytes, from the Ground Station to the local file system, whereas the ThinkPad was stressed, in both real-time and generic mode, by playing a Digital Versatile Disc (DVD) movie in the background.

As before, Figure 6.9 presents three histograms in the top row referring to the SCTs and three histograms in the bottom row referring to the ITDs, determined for the execution on the Verdex, the ThinkPad running a real-time kernel, and the same ThinkPad running a generic kernel. Against our expectations, the SCTs do not increase significantly despite the presence of relatively high I/O load. In case of the Verdex, the SCT takes on a mean value around 690 μs, whereas for the ThinkPad in real-time mode the mean SCT does not change, and for the ThinkPad in generic mode an increase to about 65 μs can be noticed. However, concerning the ITDs, the results look quite different now. It points out that the ITD associated with the Verdex flattens out evenly over a range of ± 100 μs, leading to a mean value around ± 50 μs. As expected, the ThinkPad in real-time mode yields the lowest ITD change with an increase to a mean value around ± 10 μs, whereas the ITD of the ThinkPad in generic mode is now distributed almost evenly over a range of ± 60 μs, leading to a mean value around ± 30 μs.

Nevertheless, the obtained results give evidence that, even under the constraint of high I/O load, the temporal behavior of the JAviator control system is preserved with an average jitter below ± 50 μs across different hardware platforms.

Chapter 7

Conclusion

When I examine myself and my methods of thought, I come to the conclusion that the gift of fantasy has meant more to me than any talent for abstract, positive thinking.

Albert Einstein (1879−1955)

This thesis was written with the aim not just to report on the scientific work carried out in the scope of the JAviator project, but also to provide some insight into the development process associated with custom-built quadrotor UAVs, for research groups and all others who are interested and plan to develop their own individual quadrotor platform.

7.1 Thesis Review

Quadrotor model helicopters are ideal experimental platforms for various applications in many different fields of both research and commercial usage. Most of the platforms that are commercially available come with very limited payload capabilities and are thus mostly impractical for missions that require the incorporation of additional avionics equipment. Low robustness and difficulties in modifying off-the-shelf vehicles are further reasons for relying on custom-built platforms. In this context, high-precision quadrotor helicopters were developed from scratch, particularly, the prototype JAviator V1 and its successor the more advanced JAviator V2. To provide maximum integrity at minimum weight, a new concept comprising of a fully symmetrical airframe was realized. To offer sufficient thrust capacities for carrying high payloads, a new propulsion concept consisting of belt-driven high-cohesion rotor heads combined with custom-built brushless motors was realized. Both the airframe and propulsion system design have been explained thoroughly and analyzed in regard to applied materials and corresponding weights. It has been shown that the JAviator quadrotor offers either extraordinary payload capacities, not yet found by any other quadrotor platform, or relatively long flight durations.

Due to their inherent instability and immense agility, quadrotor helicopters require some sophisticated control system to successfully stabilize them about all axes. For this purpose, a control system was developed that incorporates an EKF-based state estimator in combination with a set of advanced PIDD controllers. Specifically, the attitude and position measurements received from various sensors are fused to improve the accuracy of the estimated values. The derived estimation principle and all formulae developed for the application in conjunction with an EKF algorithm have been introduced. The control system's excellent responsiveness and, in that scope, the JAviator's remarkable maneuverability, even under extreme conditions, was demonstrated. Successful position flight control was demonstrated indoors with an RFID-based localization system and also for the very first time with a quadrotor helicopter. Furthermore, it has been shown that the achieved position-hold accuracy represents a reasonably-well result in relation to the accuracy of the available positional information.

Control systems employed in quadrotors, as well as in many other applications, are mostly developed in the traditional way. That is, they usually represent an event-triggered approach, are tuned to some specific hardware platform, and hence do not facilitate portability with respect to the temporal behavior. This circumstance was eliminated in the introduced control software by implementing a fully time-triggered system based on the concept of logical timing. It has been shown that the temporal behavior of the JAviator control system is preserved across different hardware platforms with considerable accuracy, even in the presence of relatively high I/O load. Accordingly, it has been demonstrated that time-triggered control systems come along with many advantages and therefore may dominate over event-triggered approaches in the future.

7.2 Future Work

It has been shown that the JAviator, strictly speaking, its control system, is capable of performing position-controlled flights. However, this capability could not be exploited to a great extent due to the limited accuracy of the current localization system. Thus, to explore the full potential of the control system in regard to position flight control, it would be interesting to experiment with more sophisticated localization systems. In particular, vision-based motion tracking for indoor flights as well as carrier-phase differential GPS for outdoor flights appear to be very promising options.

In this context, a challenge that has been left for future work is to improve the BMU, which, in the current configuration, cannot compete with the URF regarding heights not exceeding a few meters. This is due to the circumstance that, compared to the URF, the BMU demands much longer sampling times if adjusted for high accuracy. Running the air-pressure-based BMU with both high frequency and high amplification ratio leads to increased noise, and consequently, lower signal quality. To cope with this problem, a more advanced filter stage on the analog level in conjunction with a higher-order EKF algorithm might yield improved performance.

Finally, another major challenge is the development of algorithms that allow for fully autonomous root planning and trajectory control. In this scope, it is indeed advantageous to incorporate an additional sensor suite dedicated to obstacle recognition and implement algorithms that enable collision avoidance.

Appendix A

JAviator V1 Drawings

The creation of something new is not accomplished by the intellect but by the play instinct acting from inner necessity. The creative mind plays with the objects it loves.

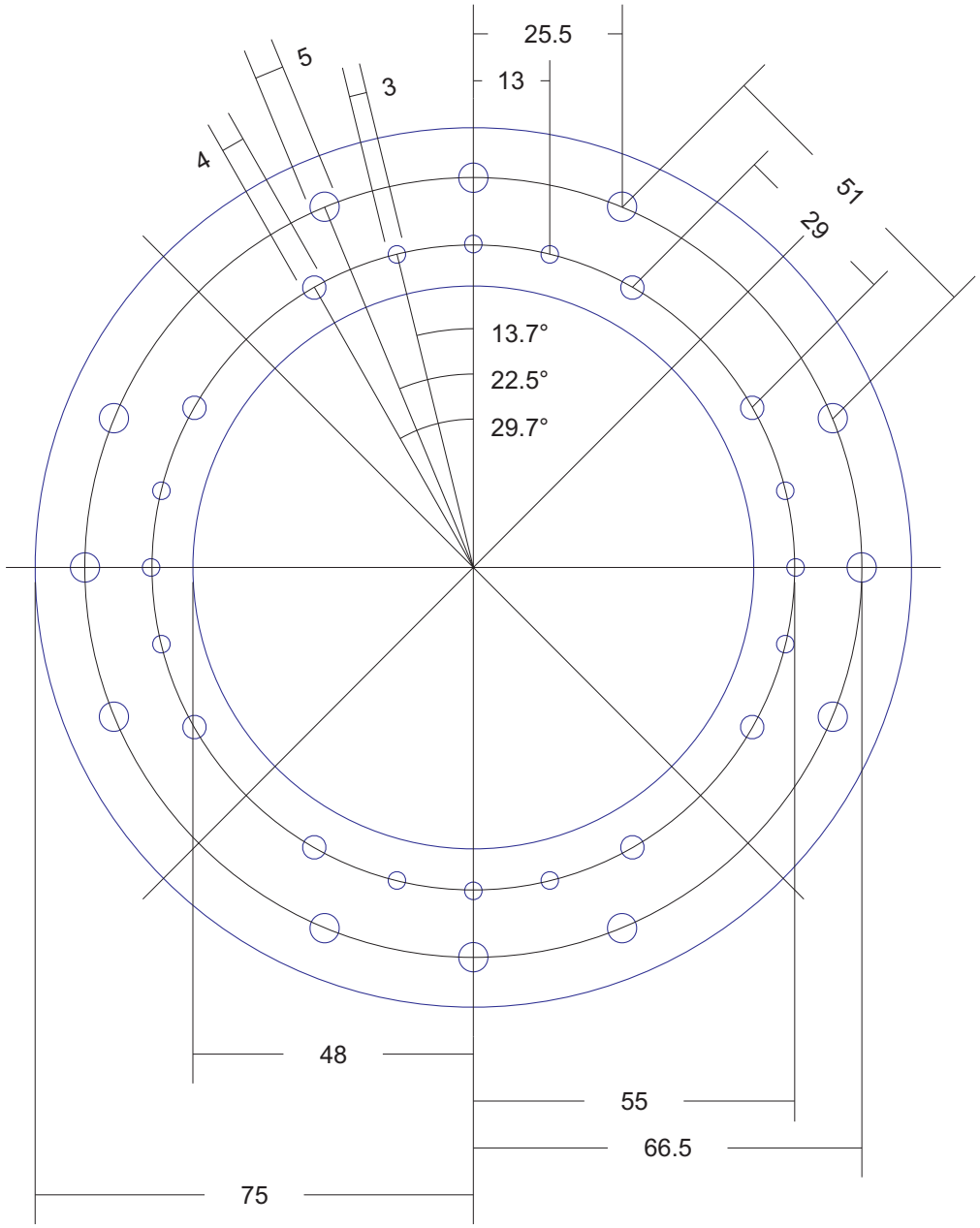
Carl Jung (1875−1961)

Disclaimer

Copyright © 2006−2010 by Rainer K. L. Trummer. All rights reserved. Permission is hereby granted, without written agreement and without license or royalty fees, to use, copy, modify, and distribute this design for any purpose, provided that this copyright notice and the following two paragraphs, or the original drawing header containing a pointer to this disclaimer (`http://javiator.cs.uni-salzburg.at/disclaimer.html`), appear in all copies of this design.

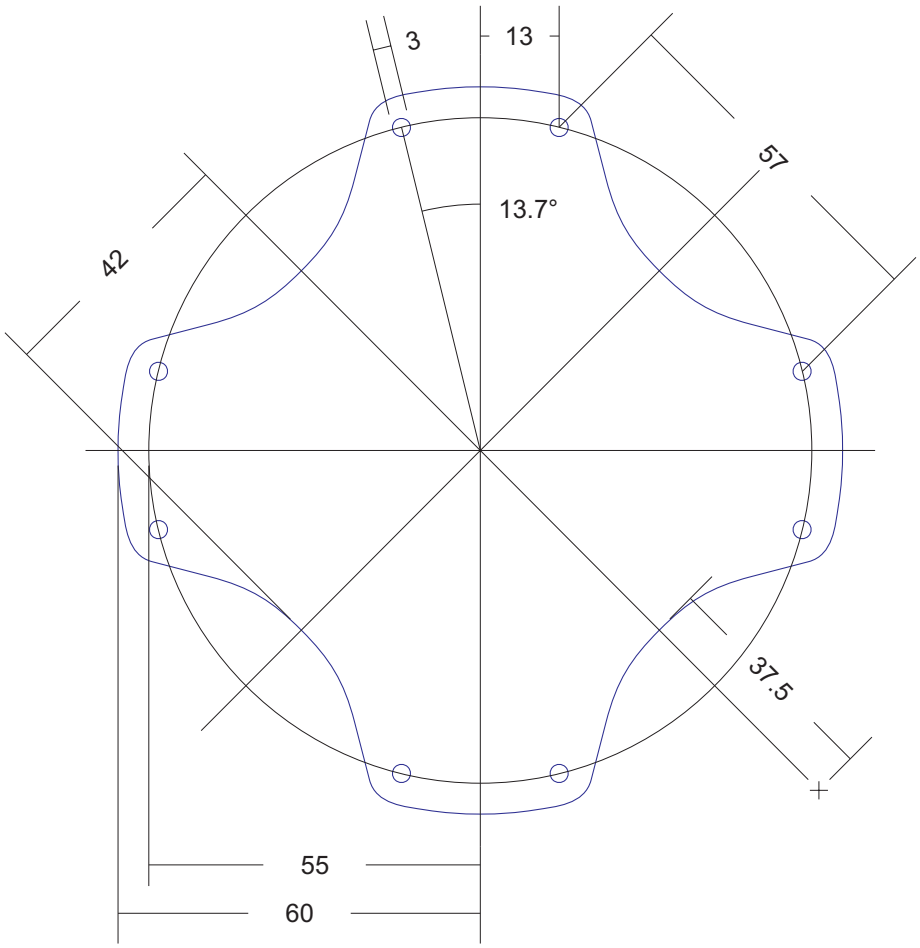
IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS DESIGN AND ITS DOCUMENTATION, EVEN IF THE COPYRIGHT HOLDER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE COPYRIGHT HOLDER SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE DESIGN PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE COPYRIGHT HOLDER HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.



Drawing		Fuselage Ring		Engineer	Rainer Trummer
Material		Carbon Fiber 3.0 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/24/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		03/25/2006		of Liability	disclaimer.html

Figure A.1: Fuselage ring used as top and bottom central connecting junction.



Drawing		Fuselage Plate		Engineer	Rainer Trummer
Material		Carbon Fiber 0.5 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/24/2006		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.2: Fuselage plate used to carry the battery and mount the inertial sensor.

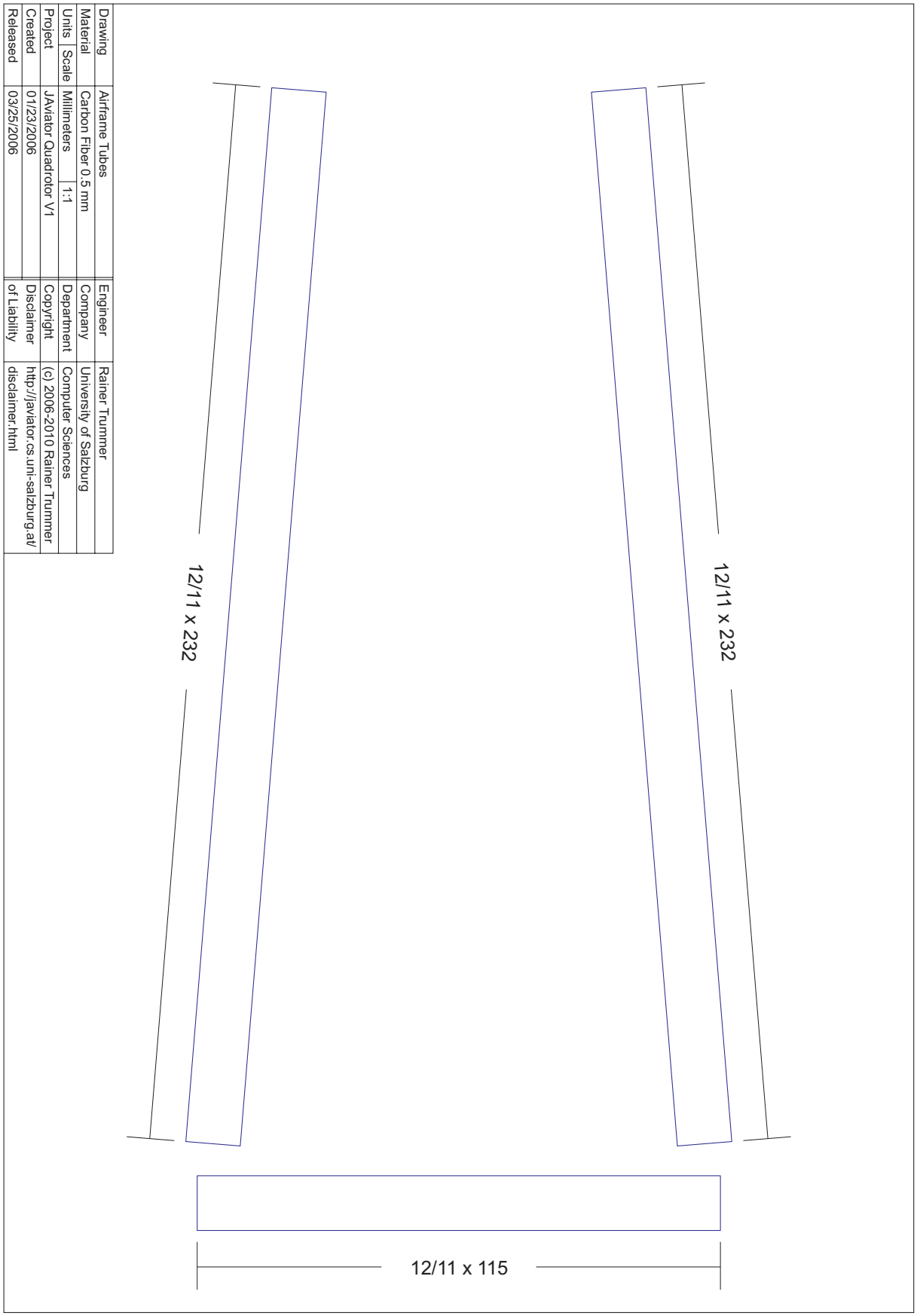
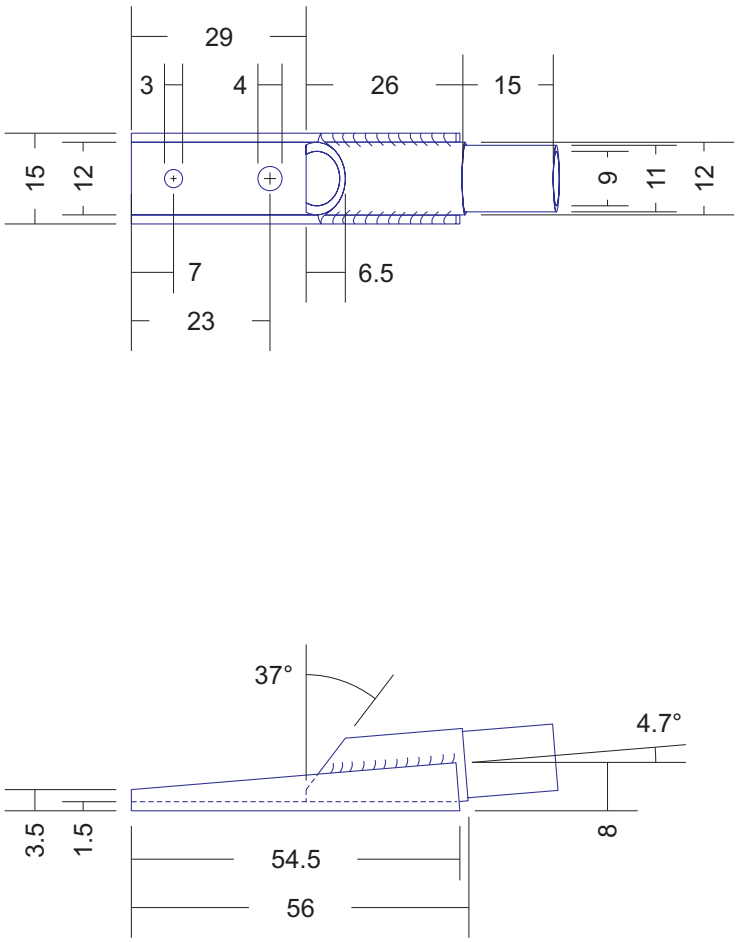


Figure A.3: Airframe tubes used to assemble the fuselage and the four girders.

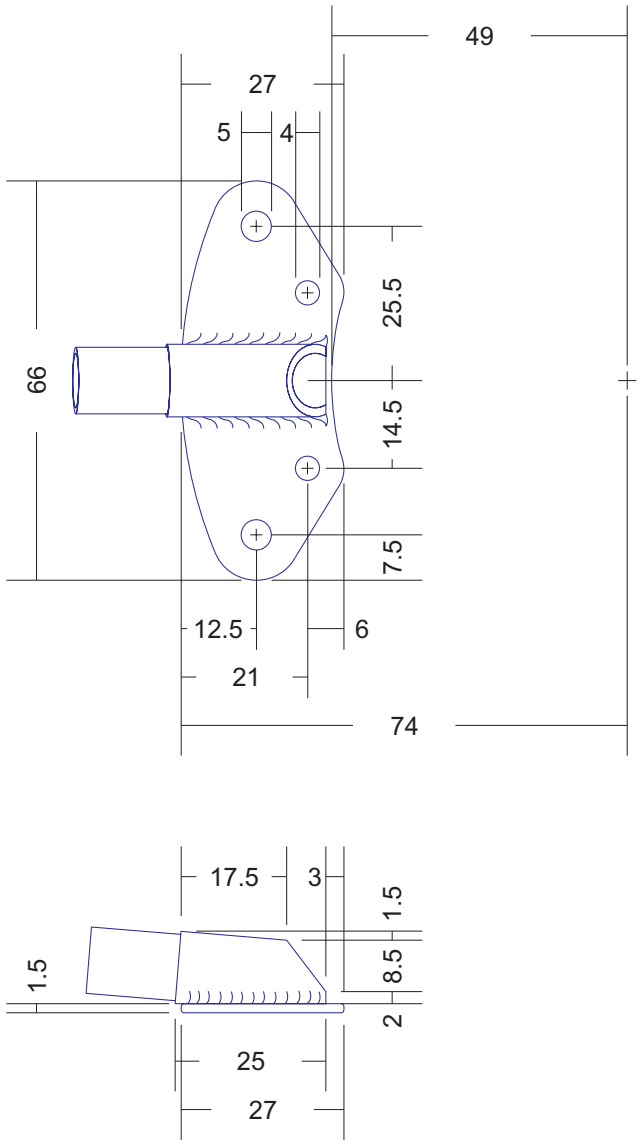
Drawing		Fuselage Connector		Engineer	Rainer Trummer
Material		Aluminum Alloy 6060AlMgSi		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/27/2006		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.4: Fuselage connector used to connect the top with the bottom fuselage ring.



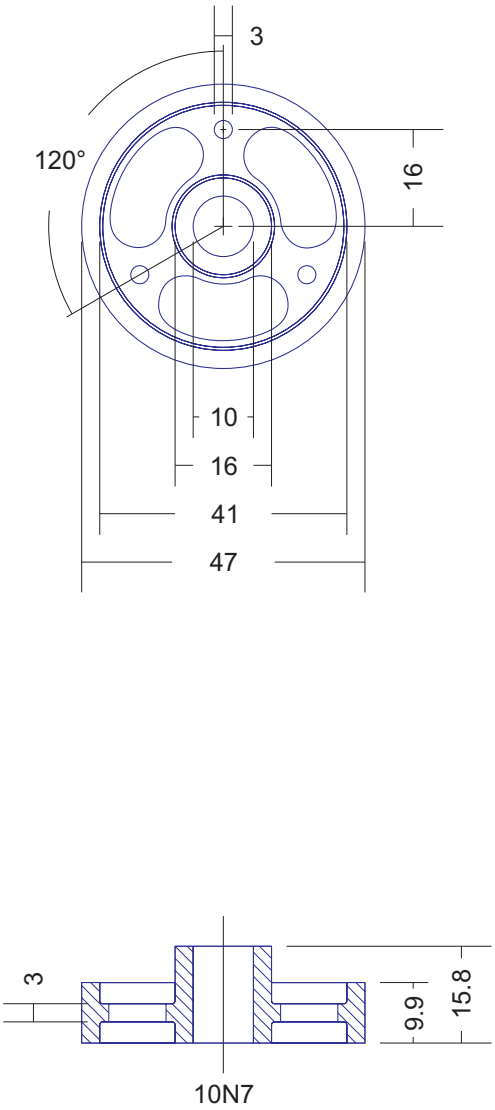
Drawing		Outer Girder Connector		Engineer	Rainer Trummer
Material		Aluminum Alloy 6060AlMgSi		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/25/2006		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.5: Outer girder connector used to connect the girders with the rotor shafts.



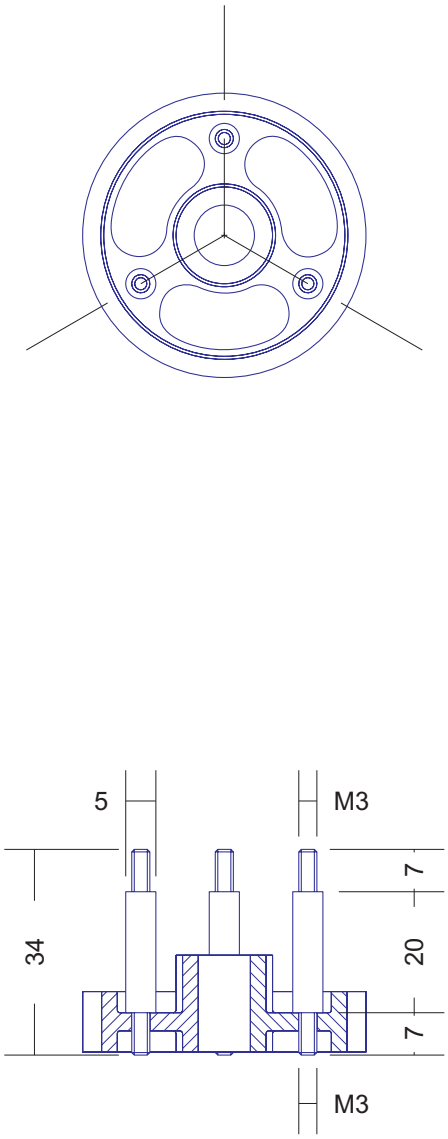
Drawing		Inner Girder Connector		Engineer	Rainer Trummer
Material		Aluminum Alloy 6060AlMgSi		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/26/2006		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.6: Inner girder connector used to connect the girders with the fuselage rings.



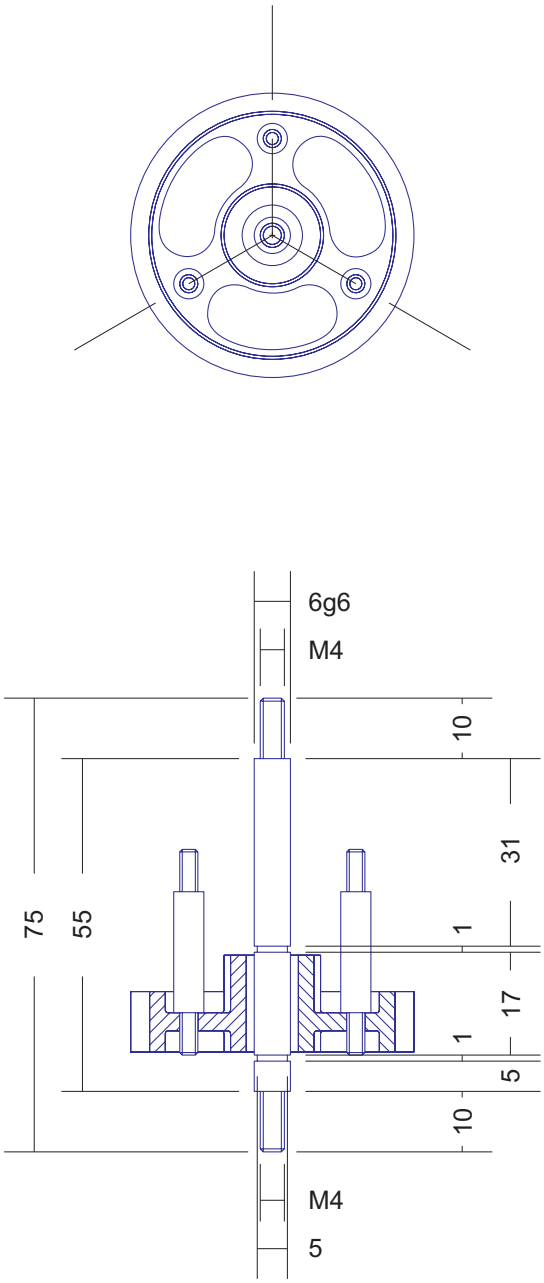
Drawing		Rotor Gear 60Z-T2.5		Engineer	Rainer Trummer
Material		Aluminum Alloy AlCuMgPb		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		12/30/2005		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.7: Rotor gear used to carry the rotor pylons and drive the rotor head.



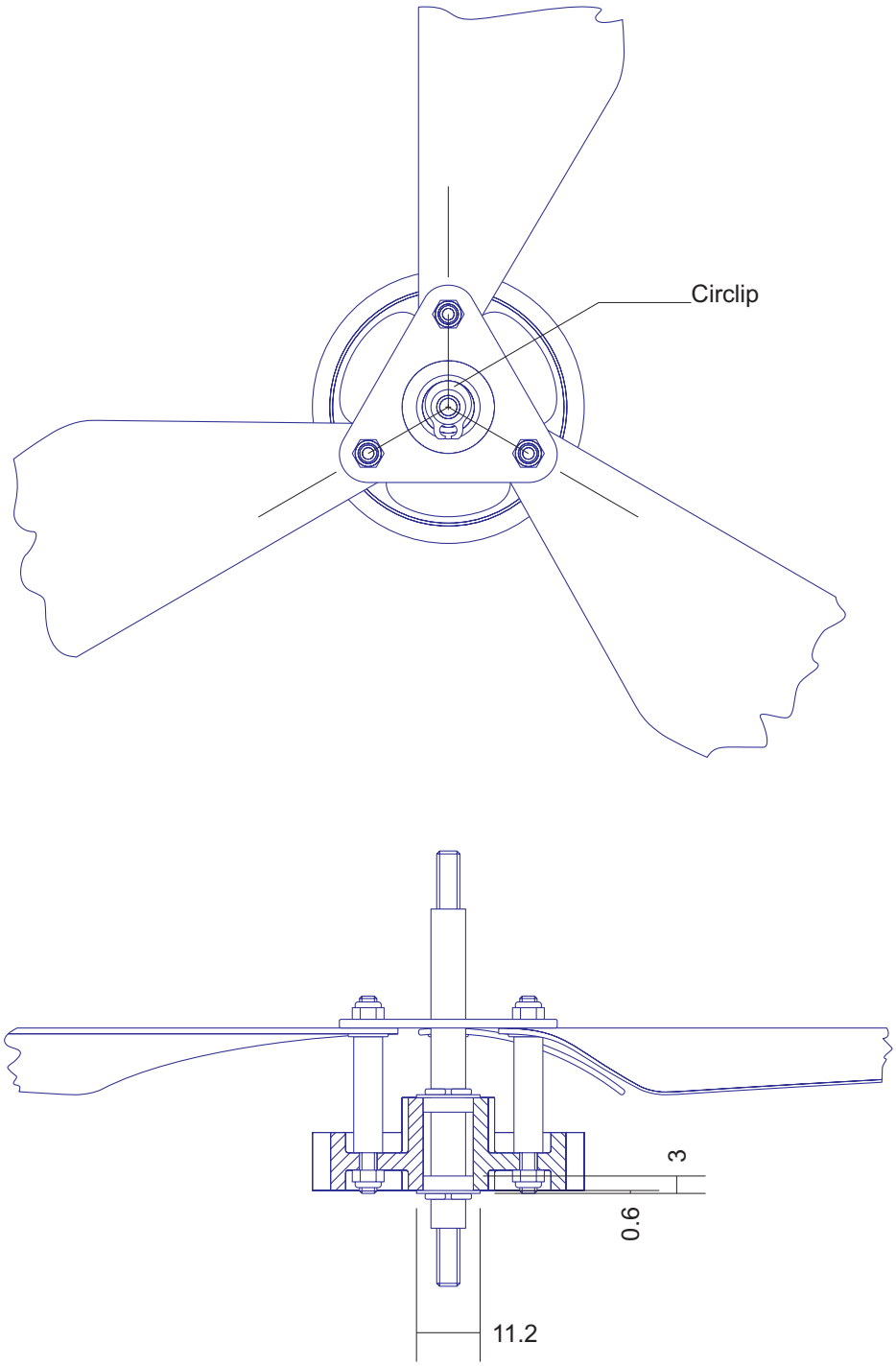
Drawing		Rotor Pylons		Engineer	Rainer Trummer
Material		Titan Alloy TiAl6V4		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/07/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		03/25/2006		of Liability	disclaimer.html

Figure A.8: Rotor pylons used to mount the rotor blades and the connecting triangle.



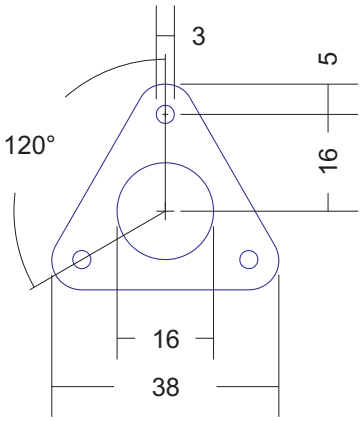
Drawing	Rotor Shaft	Engineer	Rainer Trummer
Material	Titan Alloy TiAl6V4	Company	University of Salzburg
Units / Scale	Millimeters / 1:1	Department	Computer Sciences
Project	JAviator Quadrotor V1	Copyright	(c) 2006-2010 Rainer Trummer
Created	01/07/2006	Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released	03/25/2006		

Figure A.9: Rotor shaft used to mount the rotor head and connect the girder ends.



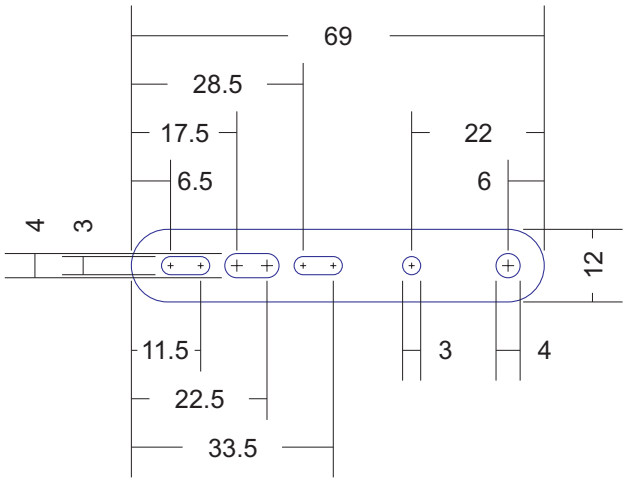
Drawing		Rotor Bearings DDLF-1060		Engineer	Rainer Trummer
Material		Stainless-Steel Alloy		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/07/2006		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.10: Rotor bearings used to connect the rotor head with the rotor shaft.



Drawing		Rotor Triangle		Engineer	Rainer Trummer
Material		Carbon Fiber 1.5 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		12/30/2005		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		03/25/2006			

Figure A.11: Rotor triangle used to connect the rotor pylons for more integrity.



Drawing		Motor Carrier		Engineer	Rainer Trummer
Material		Carbon Fiber 1.5 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/28/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		03/25/2006		of Liability	disclaimer.html

Figure A.12: Motor carrier used to mount a motor and tighten the cogged belt.

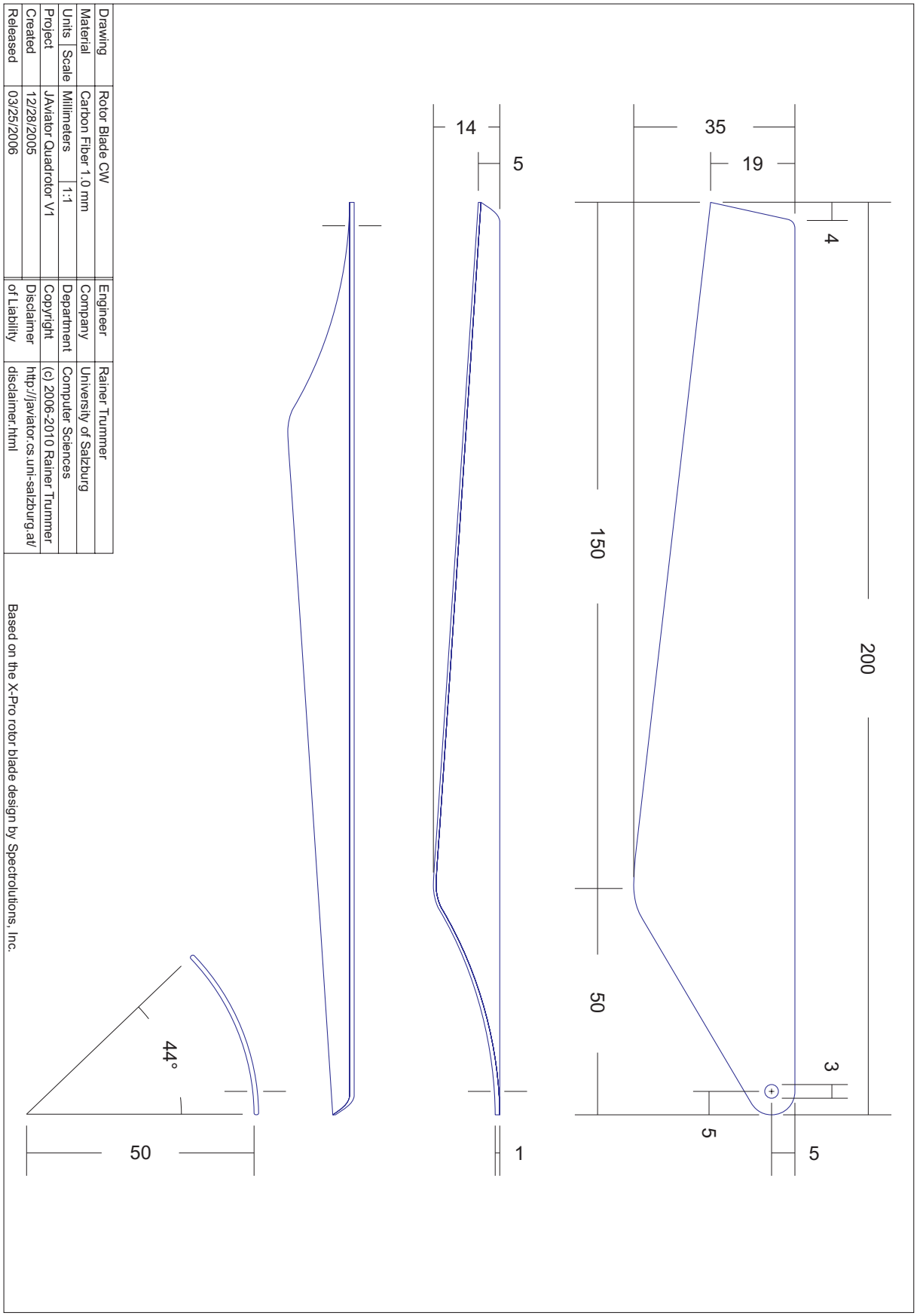


Figure A.13: Rotor blade used to assemble the clockwise spinning rotors.

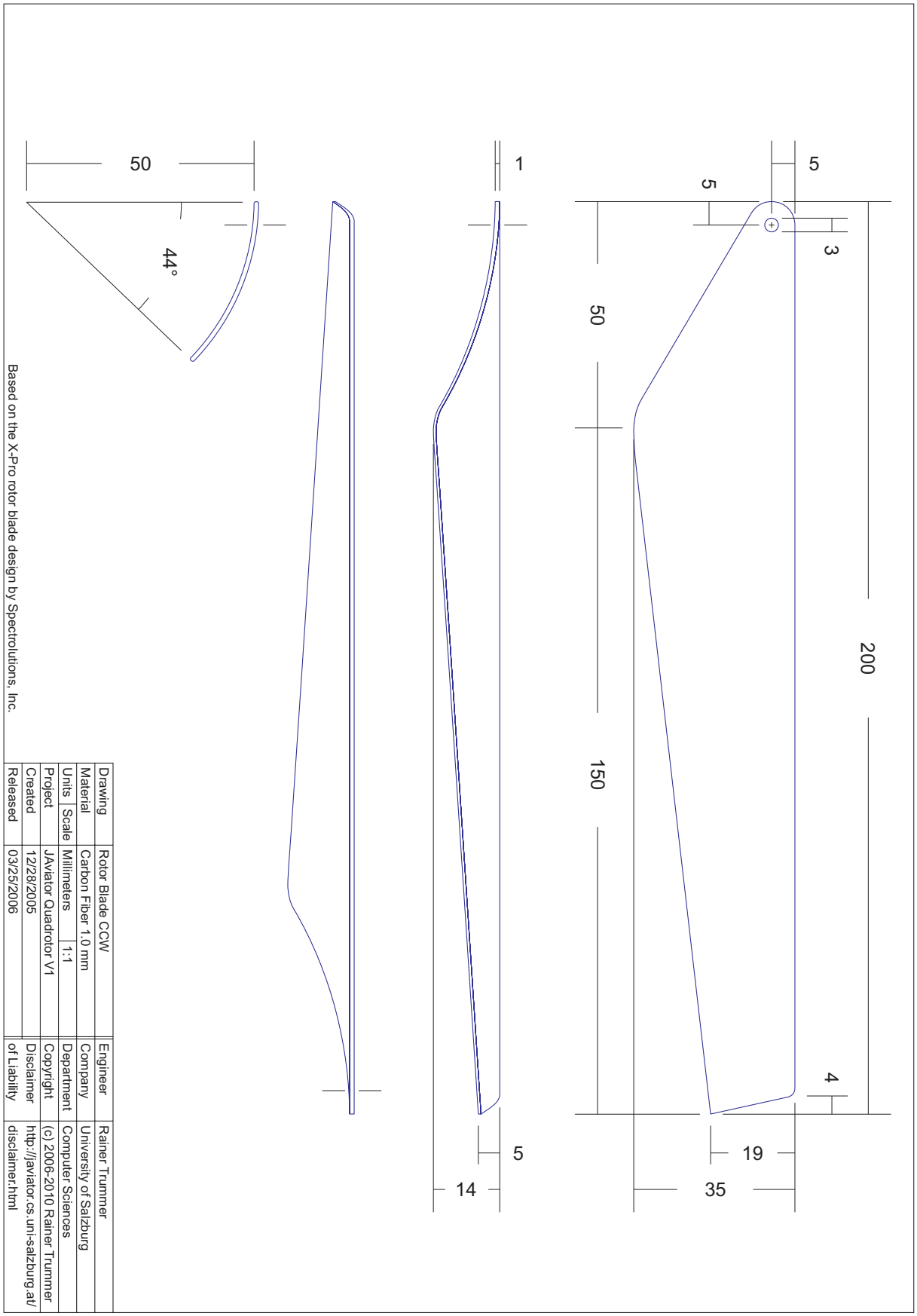


Figure A.14: Rotor blade used to assemble the counter-clockwise spinning rotors.

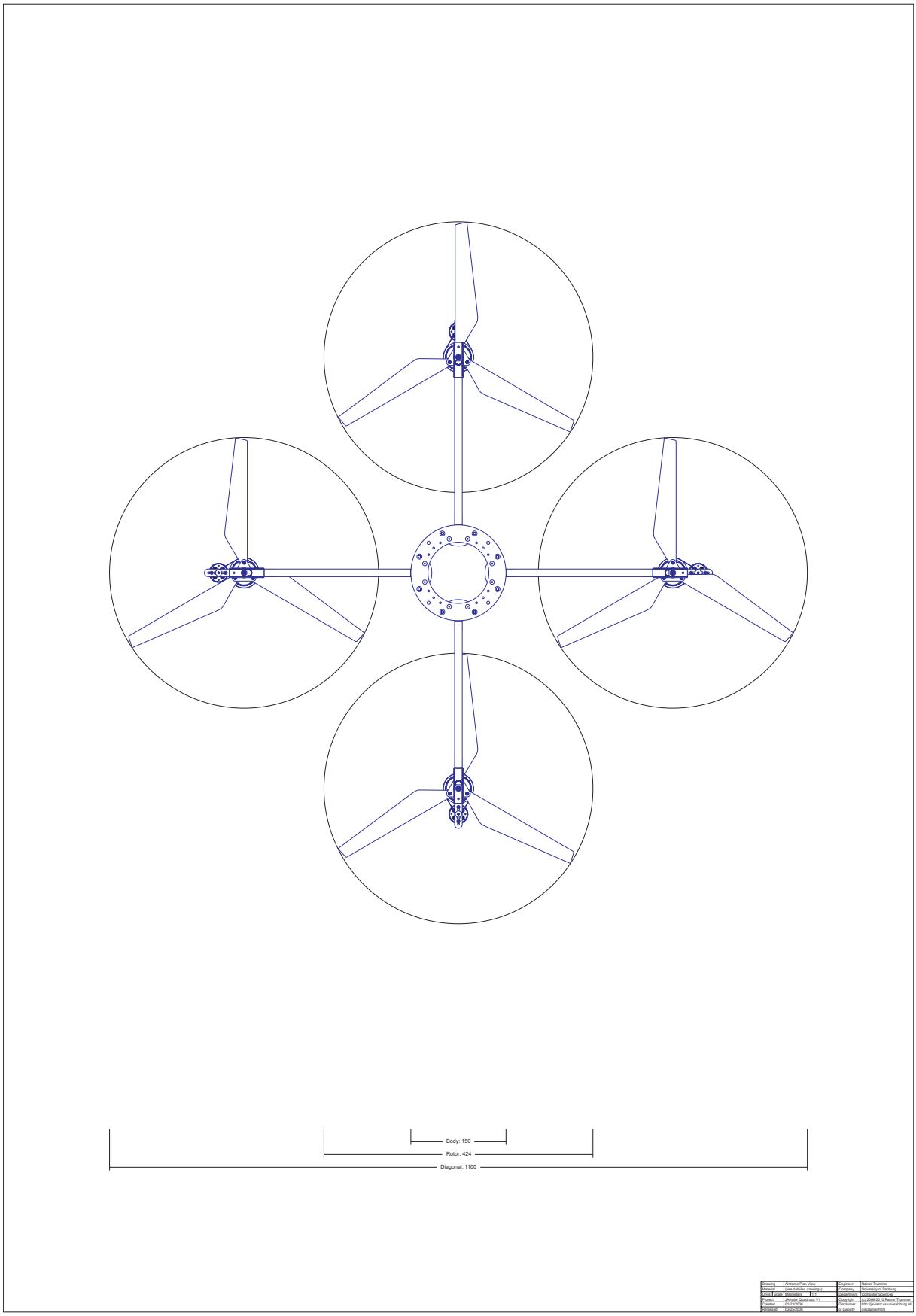


Figure A.15: Plan view of the complete airframe with all propulsion groups.

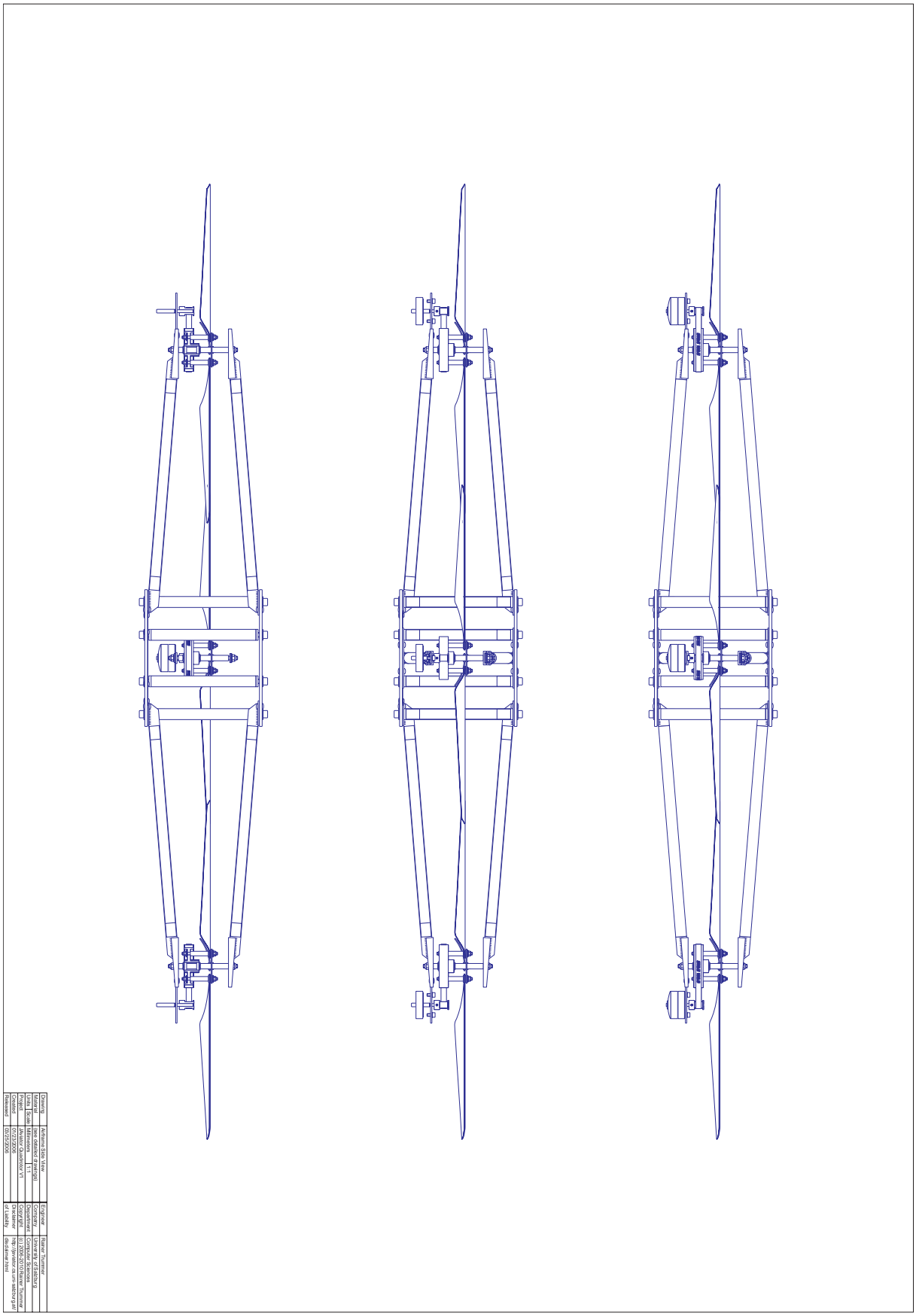


Figure A.16: Side view of the complete airframe with all propulsion groups.

Appendix B

JAviator V2 Drawings

The second system is the most dangerous system a man ever designs ... The general tendency is to over-design the second system, using all the ideas and frills that were cautiously sidetracked on the first one.

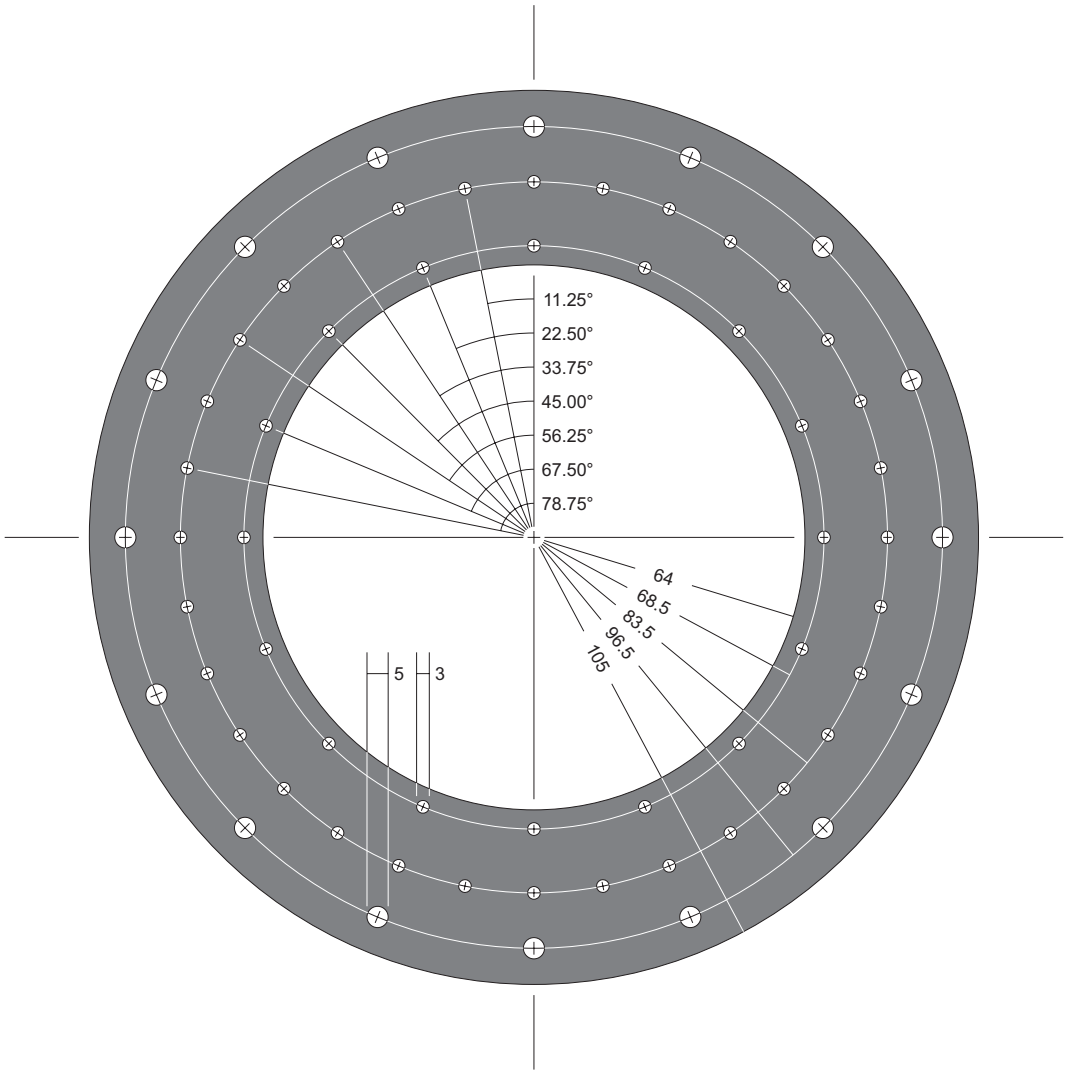
F. P. Brooks, Jr., *The Mythical Man-Mouth*, 1975

Disclaimer

Copyright © 2006−2010 by Rainer K. L. Trummer. All rights reserved. Permission is hereby granted, without written agreement and without license or royalty fees, to use, copy, modify, and distribute this design for any purpose, provided that this copyright notice and the following two paragraphs, or the original drawing header containing a pointer to this disclaimer (`http://javiator.cs.uni-salzburg.at/disclaimer.html`), appear in all copies of this design.

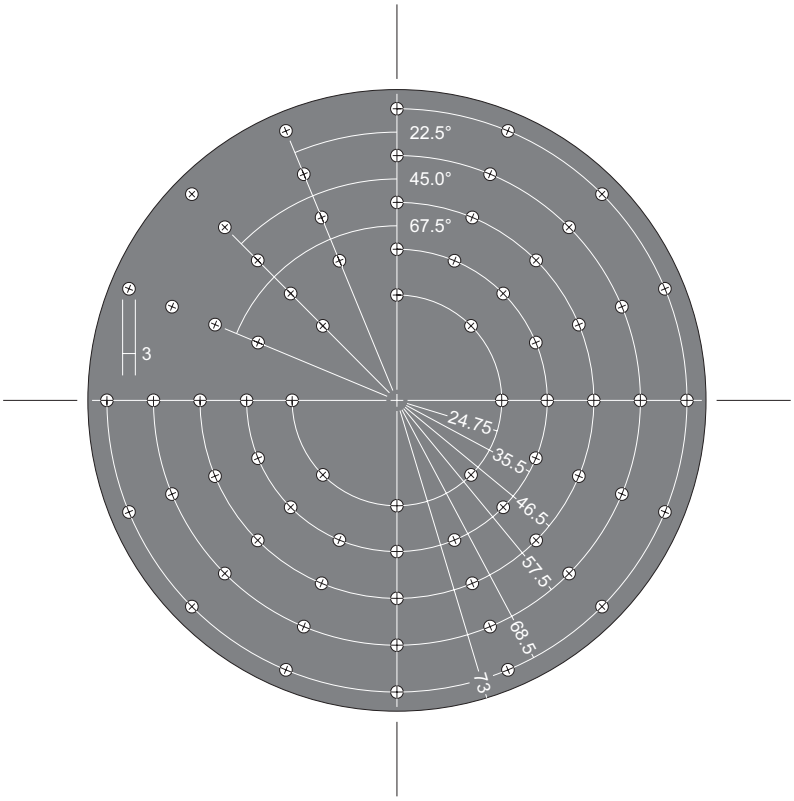
IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS DESIGN AND ITS DOCUMENTATION, EVEN IF THE COPYRIGHT HOLDER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE COPYRIGHT HOLDER SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE DESIGN PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE COPYRIGHT HOLDER HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.



Drawing		Fuselage Ring		Engineer	Rainer Trummer
Material		Carbon Fiber 2.0 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V2		Copyright	(c) 2007-2010 Rainer Trummer
Created		03/26/2007		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		04/01/2008			

Figure B.1: Fuselage ring used as top and bottom central connecting junction.



Drawing		Fuselage Plate		Engineer	Rainer Trummer
Material		Carbon Fiber 1.0 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V2		Copyright	(c) 2007-2010 Rainer Trummer
Created		03/27/2007		Disclaimer of Liability	http://javiator.cs.uni-salzburg.at/disclaimer.html
Released		04/01/2008			

Figure B.2: Fuselage plate used to carry the battery and mount the inertial sensor.

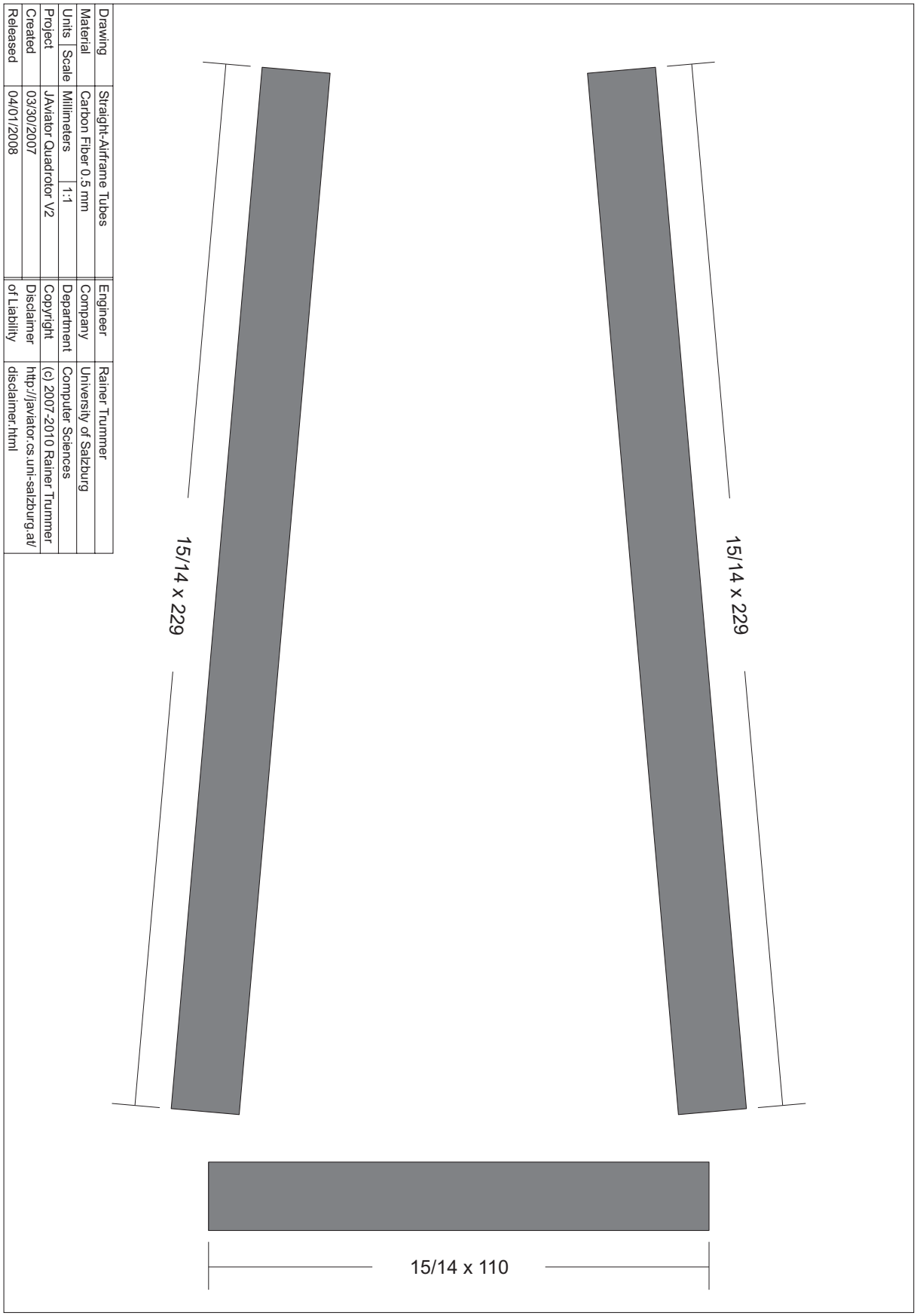


Figure B.3: Straight-airframe tubes used to assemble the fuselage and the four girders.

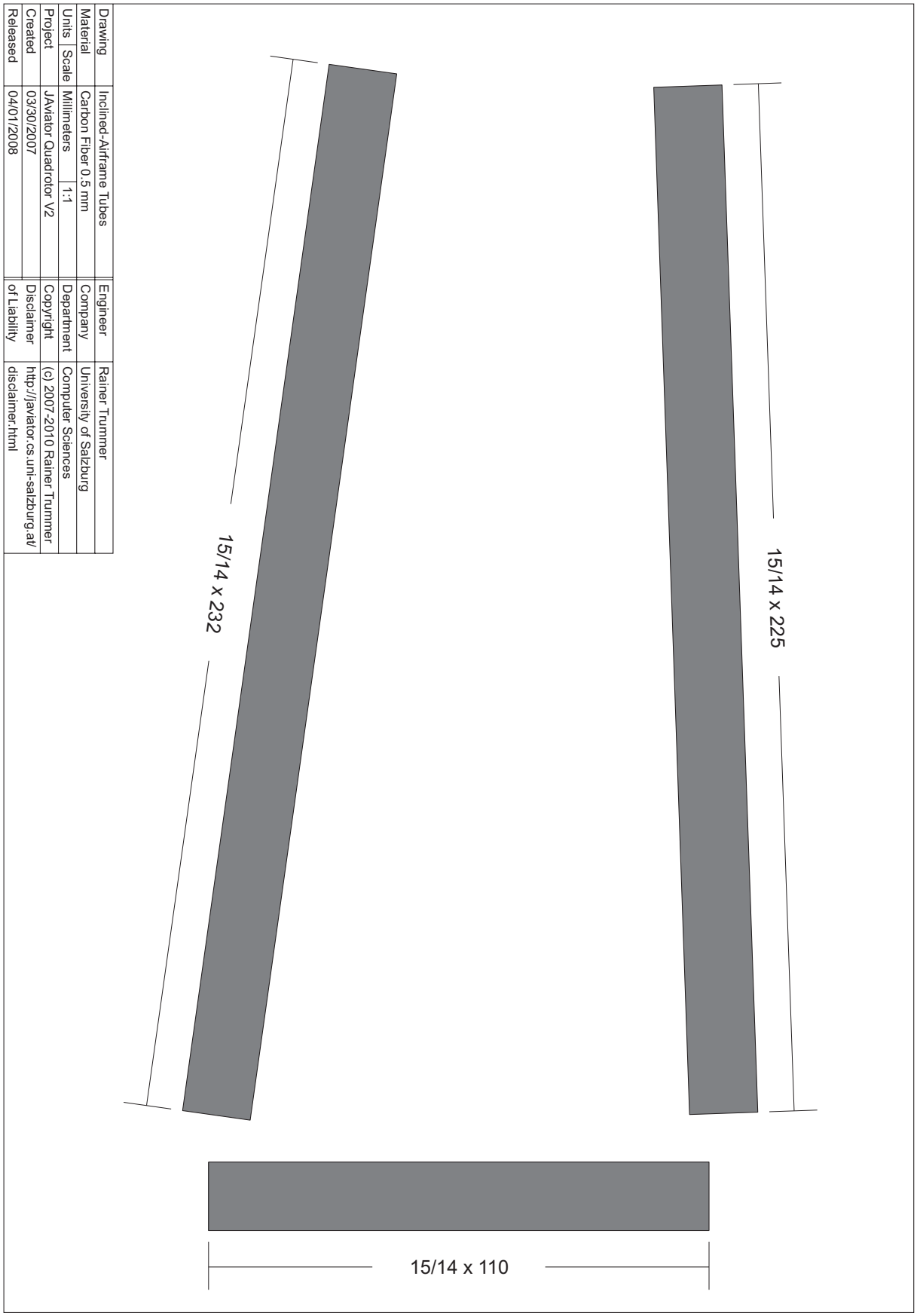
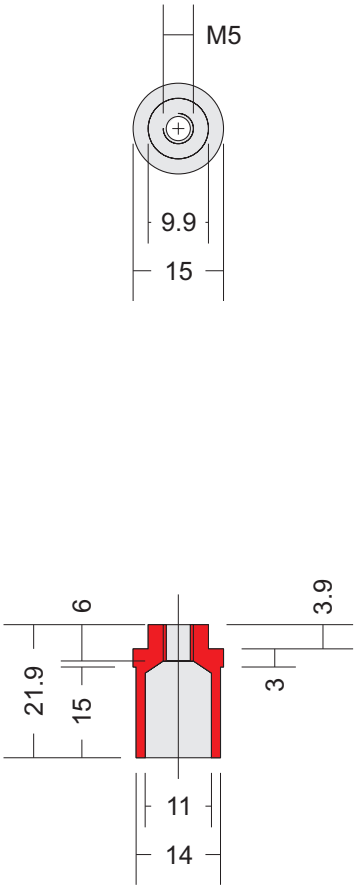
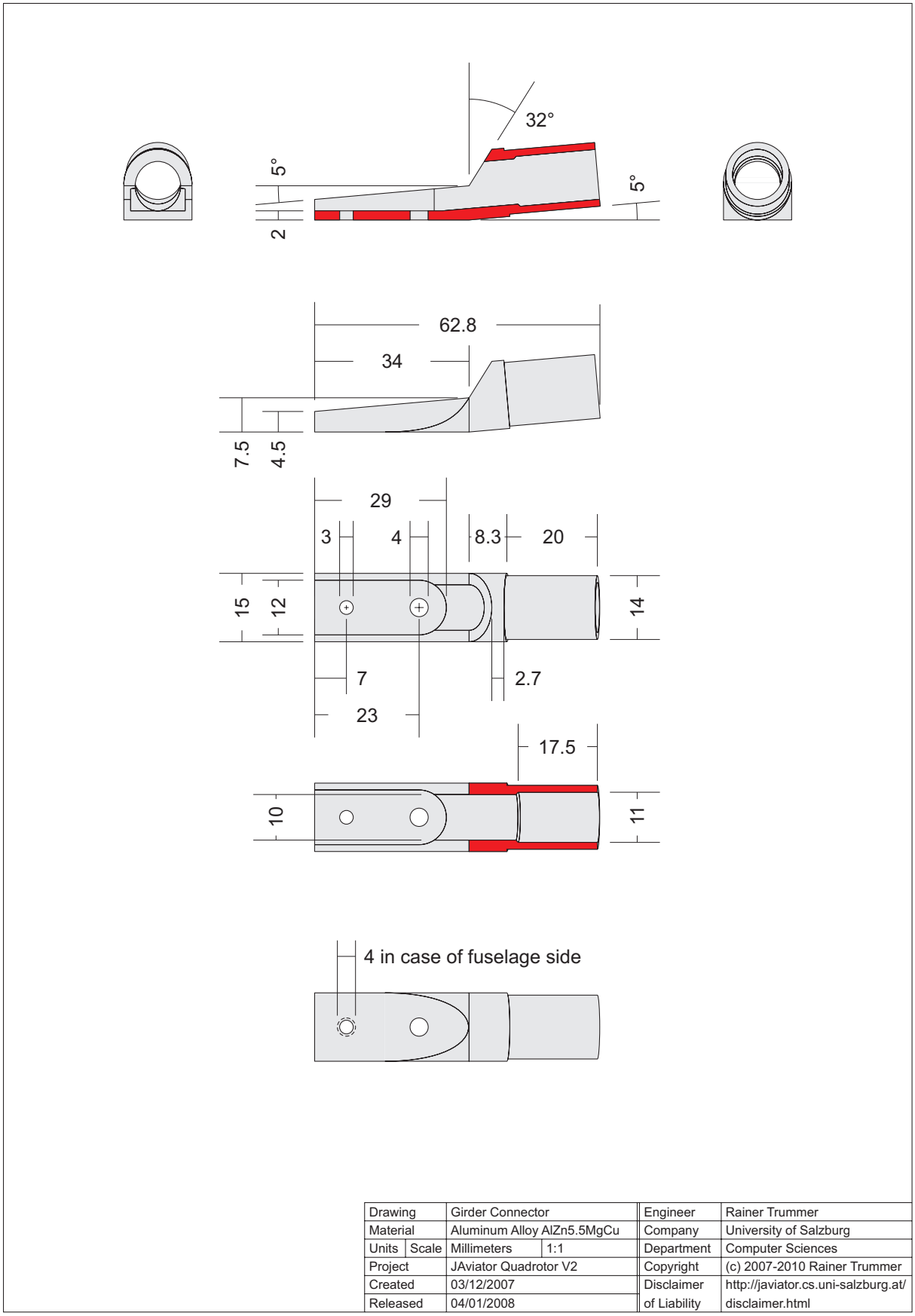


Figure B.4: Inclined-airframe tubes used to assemble the fuselage and the four girders.



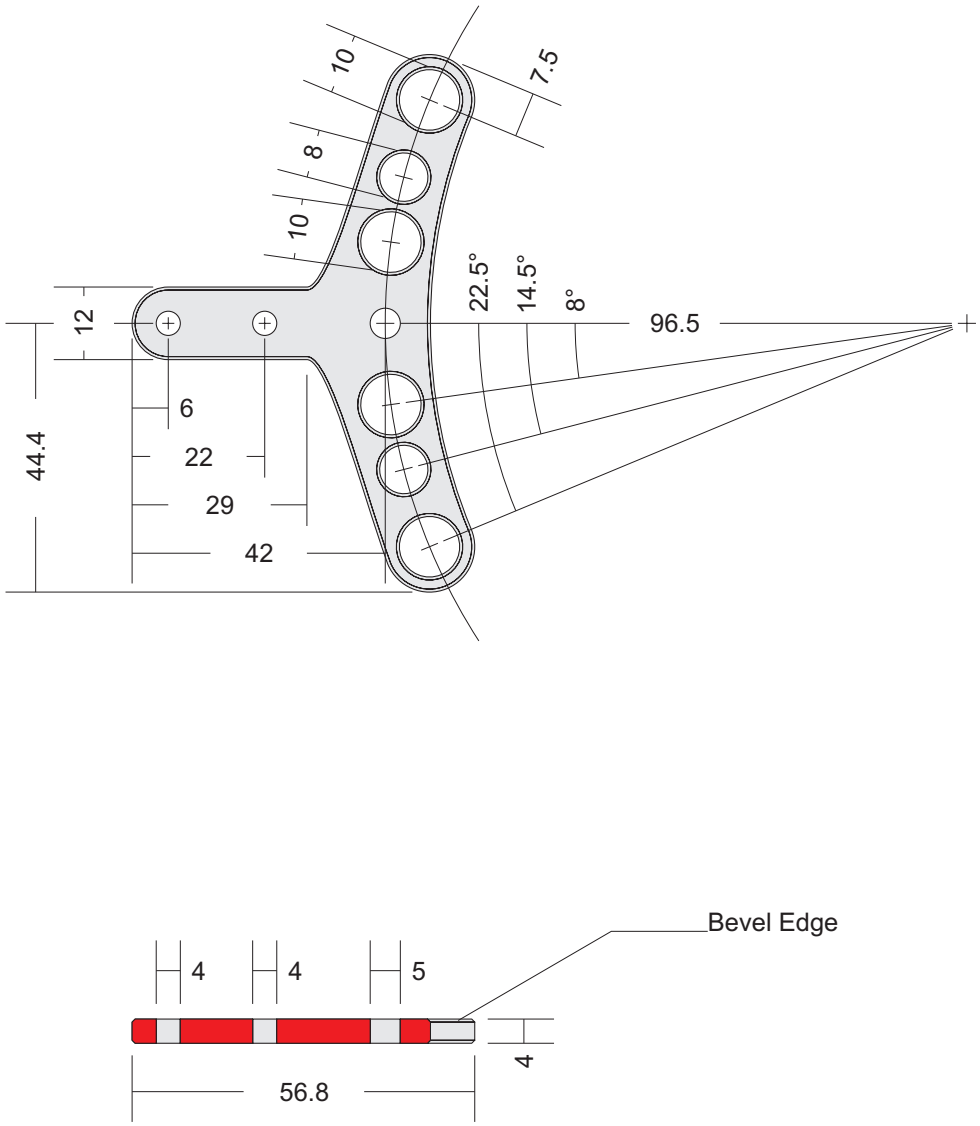
Drawing		Fuselage Connector		Engineer	Rainer Trummer
Material		Aluminum Alloy AlZn5.5MgCu		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V2		Copyright	(c) 2007-2010 Rainer Trummer
Created		03/12/2007		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		04/01/2008		of Liability	disclaimer.html

Figure B.5: Fuselage connector used to connect the top with the bottom fuselage ring.



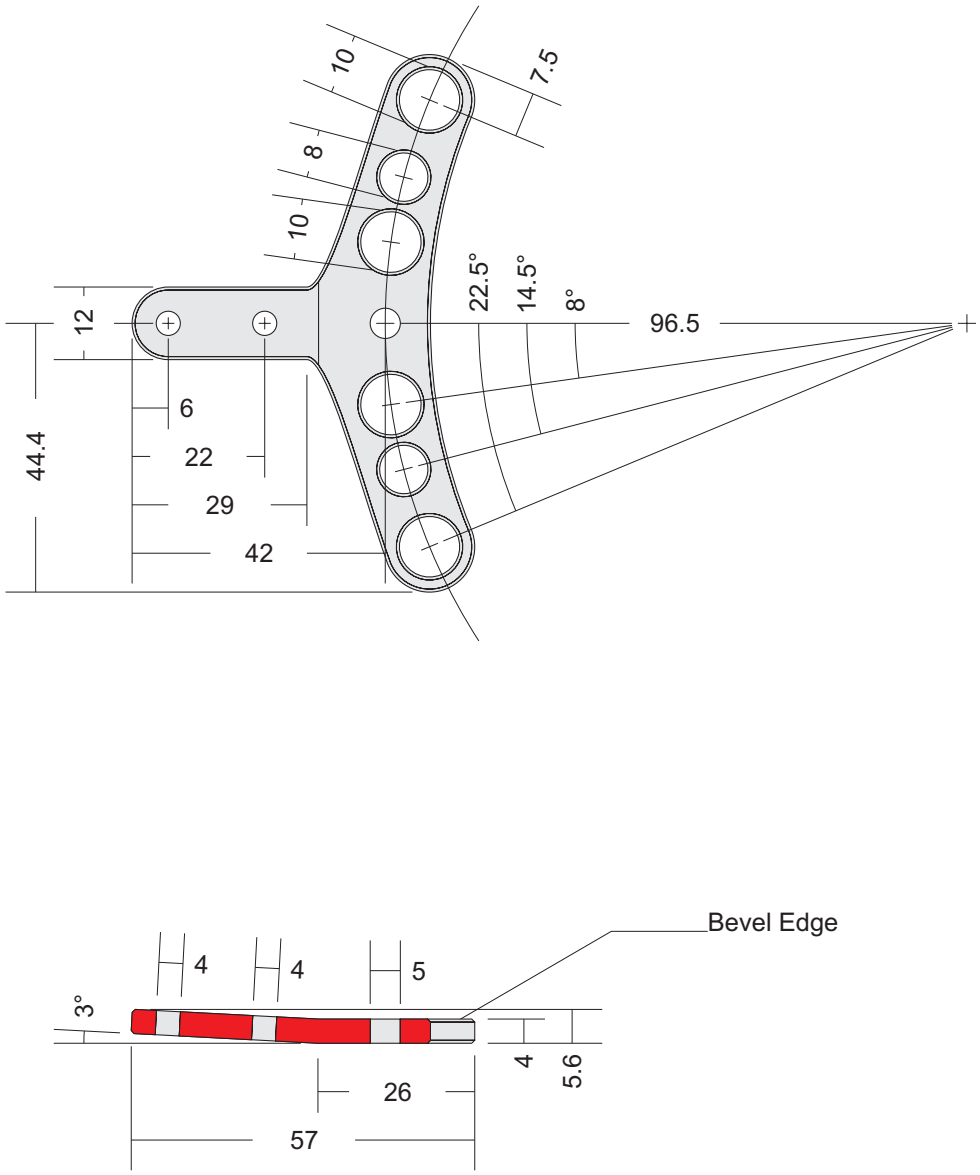
Drawing		Girder Connector		Engineer	Rainer Trummer
Material		Aluminum Alloy AlZn5.5MgCu		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V2		Copyright	(c) 2007-2010 Rainer Trummer
Created		03/12/2007		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		04/01/2008		of Liability	disclaimer.html

Figure B.6: Girder connector used to connect to the fuselage rings and rotor shafts.



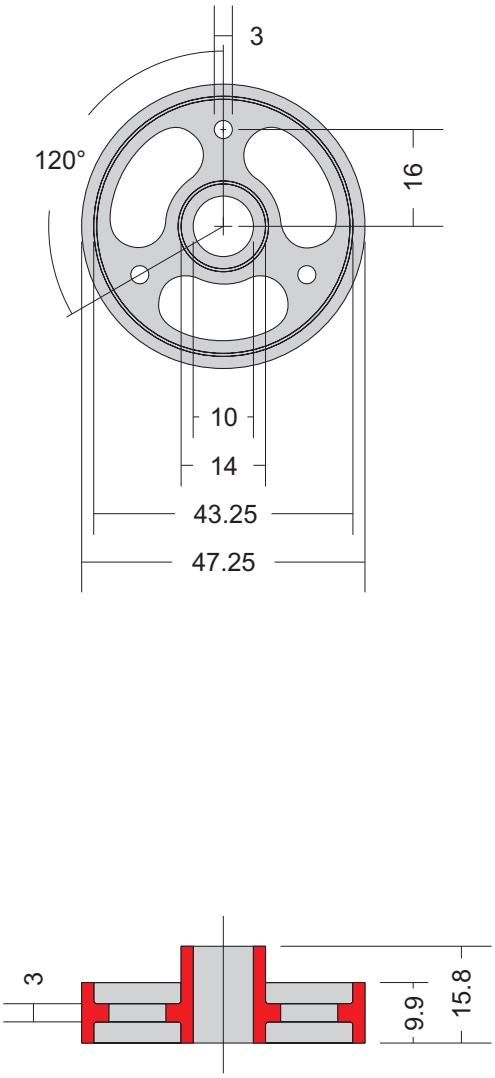
Drawing		Straight-Airframe Flange		Engineer	Rainer Trummer
Material		Aluminum Alloy		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V2		Copyright	(c) 2007-2010 Rainer Trummer
Created		03/24/2007		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		04/01/2008		of Liability	disclaimer.html

Figure B.7: Straight-airframe flange used to connect the girders to the fuselage rings.



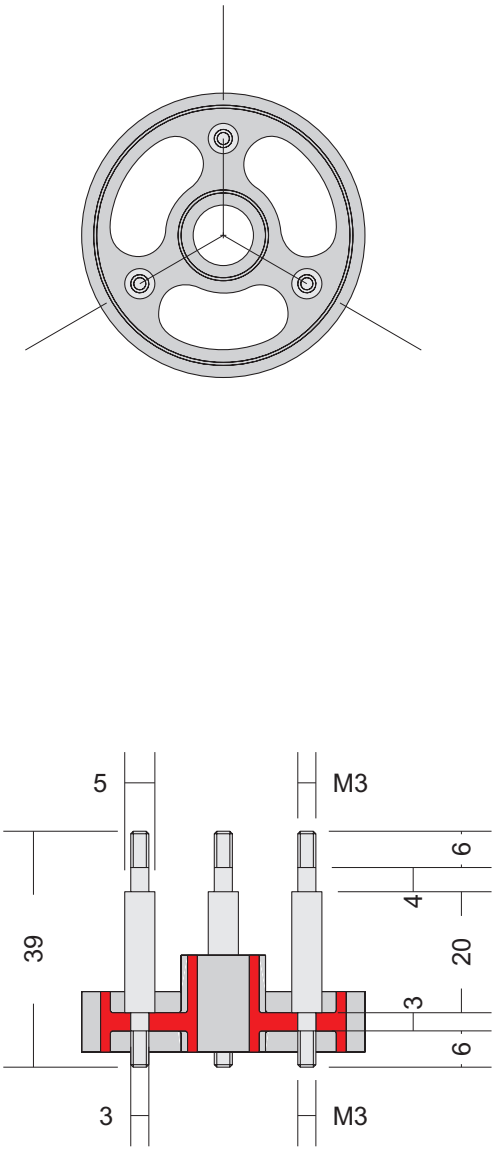
Drawing		Inclined-Airframe Flange		Engineer	Rainer Trummer
Material		Aluminum Alloy		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V2		Copyright	(c) 2007-2010 Rainer Trummer
Created		03/24/2007		Disclaimer	http://javiator.cs.uni-salzburg.at/
Released		04/01/2008		of Liability	disclaimer.html

Figure B.8: Inclined-airframe flange used to connect the girders to the fuselage rings.



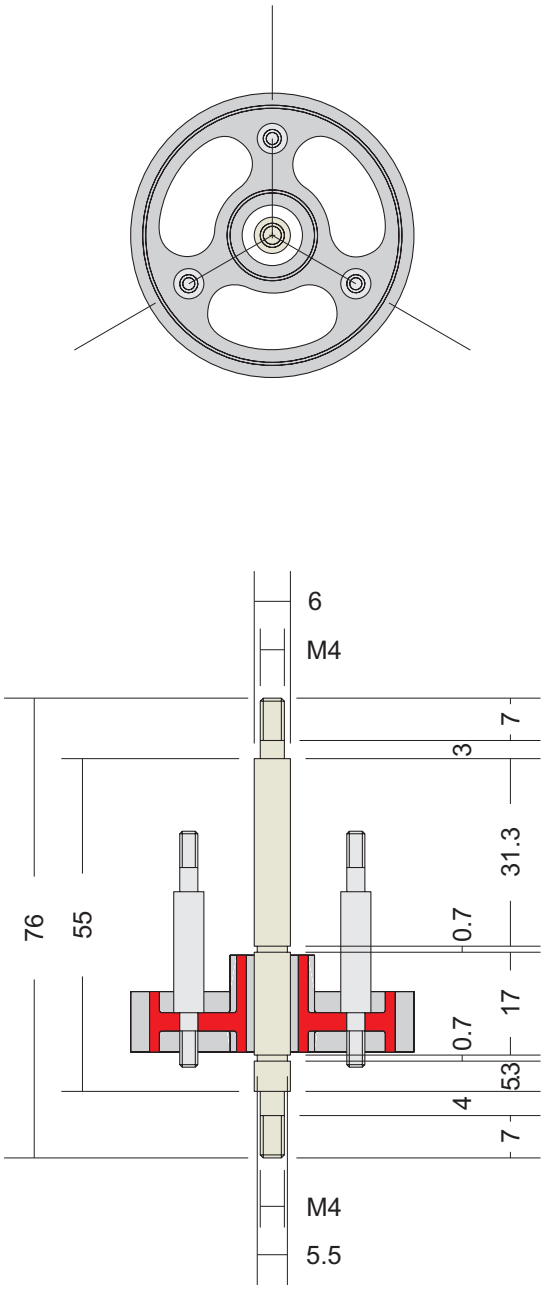
Drawing		Rotor Gear 60Z-T2.5		Engineer	Rainer Trummer
Material		Aluminum Alloy AlCuMgPb		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1, V2		Copyright	(c) 2006-2010 Rainer Trummer
Created		12/30/2005		Disclaimer	http://javiator.cs.uni-salzburg.at/
Updated		03/30/2006, 02/17/2007		of Liability	disclaimer.html

Figure B.9: Rotor gear used to carry the rotor pylons and drive the rotor head.



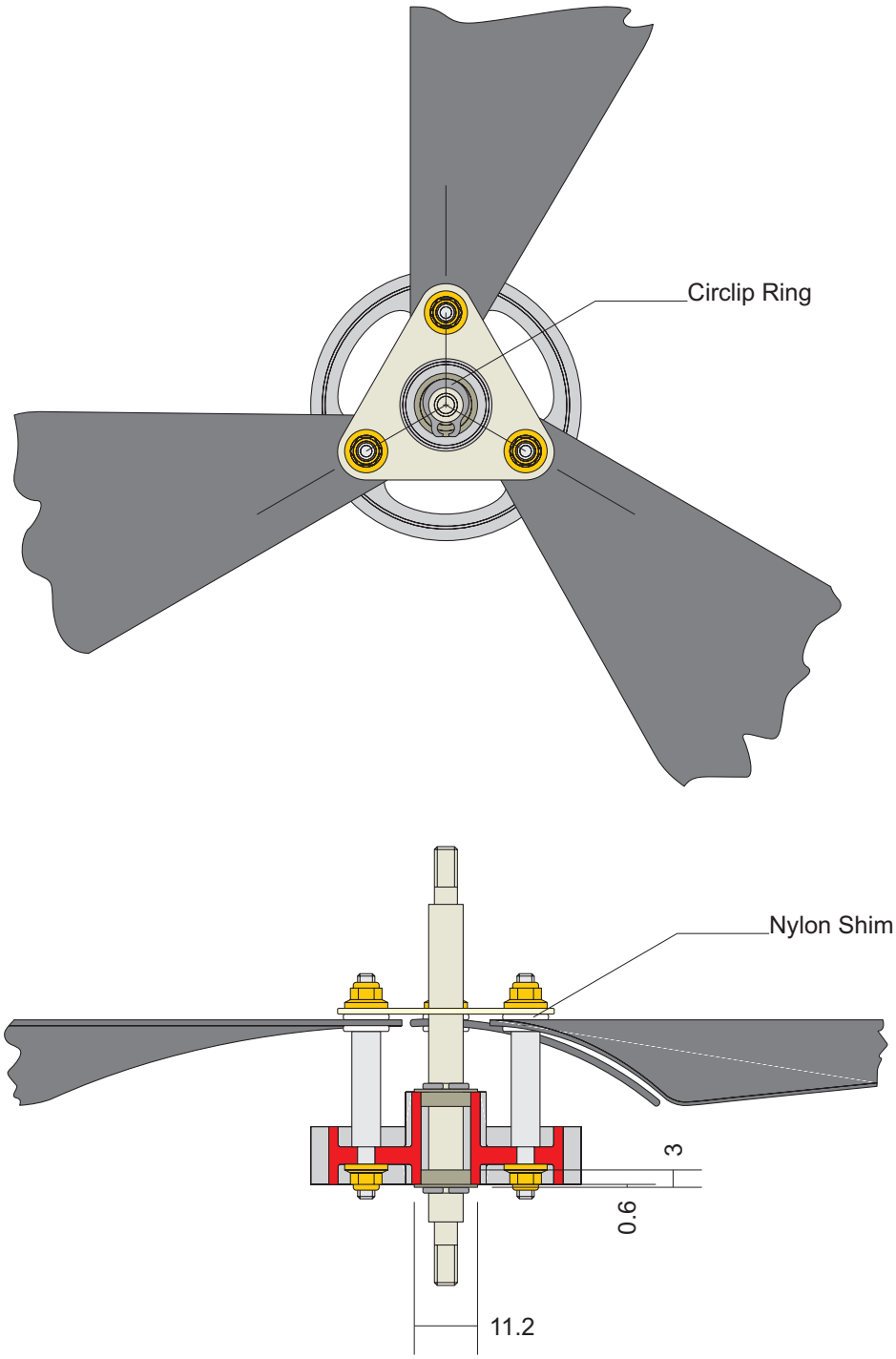
Drawing		Rotor Pylons		Engineer	Rainer Trummer
Material		Aluminum Alloy AlZn5.5MgCu		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1, V2		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/07/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Updated		03/31/2006, 02/17/2007		of Liability	disclaimer.html

Figure B.10: Rotor pylons used to mount the rotor blades and the connecting triangle.



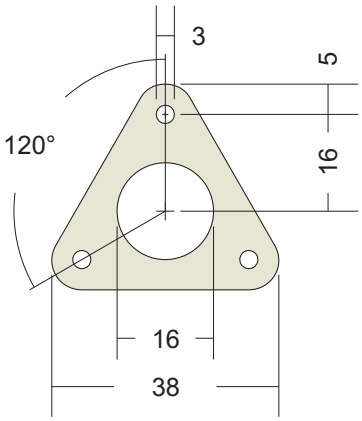
Drawing		Rotor Shaft		Engineer	Rainer Trummer
Material		Titanium Alloy TiAl6V4		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1, V2		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/07/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Updated		03/31/2006, 02/17/2007		of Liability	disclaimer.html

Figure B.11: Rotor shaft used to mount the rotor head and connect the girder ends.



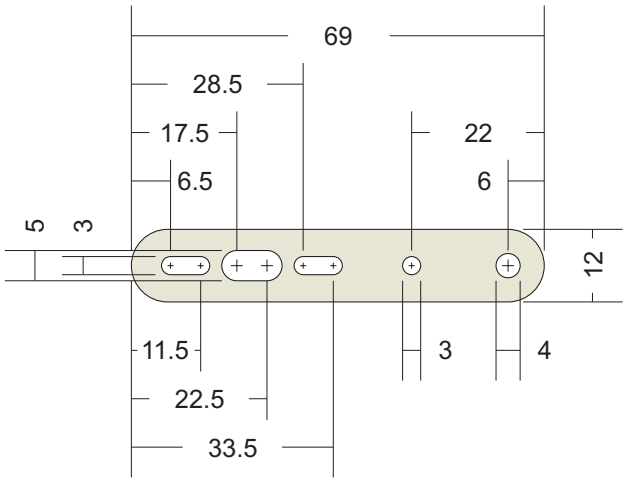
Drawing		Rotor Bearings MF-106-ZZ		Engineer	Rainer Trummer
Material		Stainless-Steel Alloy		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1, V2		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/07/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Updated		03/31/2006, 02/18/2007		of Liability	disclaimer.html

Figure B.12: Rotor bearings used to connect the rotor head with the rotor shaft.



Drawing		Rotor Triangle		Engineer	Rainer Trummer
Material		Titanium Alloy 0.8 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1, V2		Copyright	(c) 2006-2010 Rainer Trummer
Created		12/30/2005		Disclaimer	http://javiator.cs.uni-salzburg.at/
Updated		03/30/2006, 02/17/2007		of Liability	disclaimer.html

Figure B.13: Rotor triangle used to connect the rotor pylons for more integrity.



Drawing		Motor Carrier		Engineer	Rainer Trummer
Material		Titanium Alloy 1.5 mm		Company	University of Salzburg
Units	Scale	Millimeters	1:1	Department	Computer Sciences
Project		JAviator Quadrotor V1, V2		Copyright	(c) 2006-2010 Rainer Trummer
Created		01/28/2006		Disclaimer	http://javiator.cs.uni-salzburg.at/
Updated		04/01/2006, 02/17/2007		of Liability	disclaimer.html

Figure B.14: Motor carrier used to mount a motor and tighten the cogged belt.

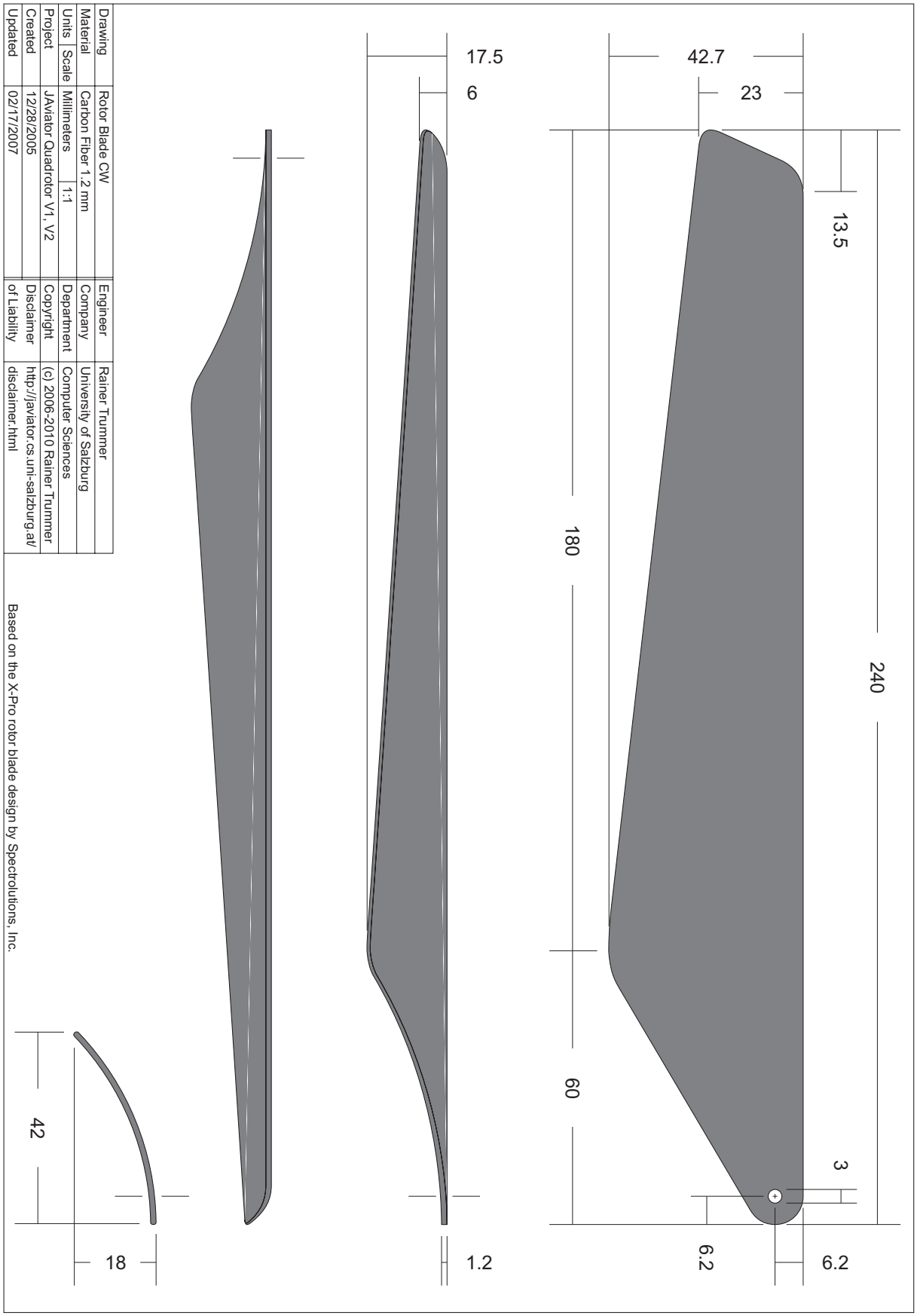


Figure B.15: Rotor blade used to assemble the clockwise spinning rotors.

Figure B.16: Rotor blade used to assemble the counter-clockwise spinning rotors.

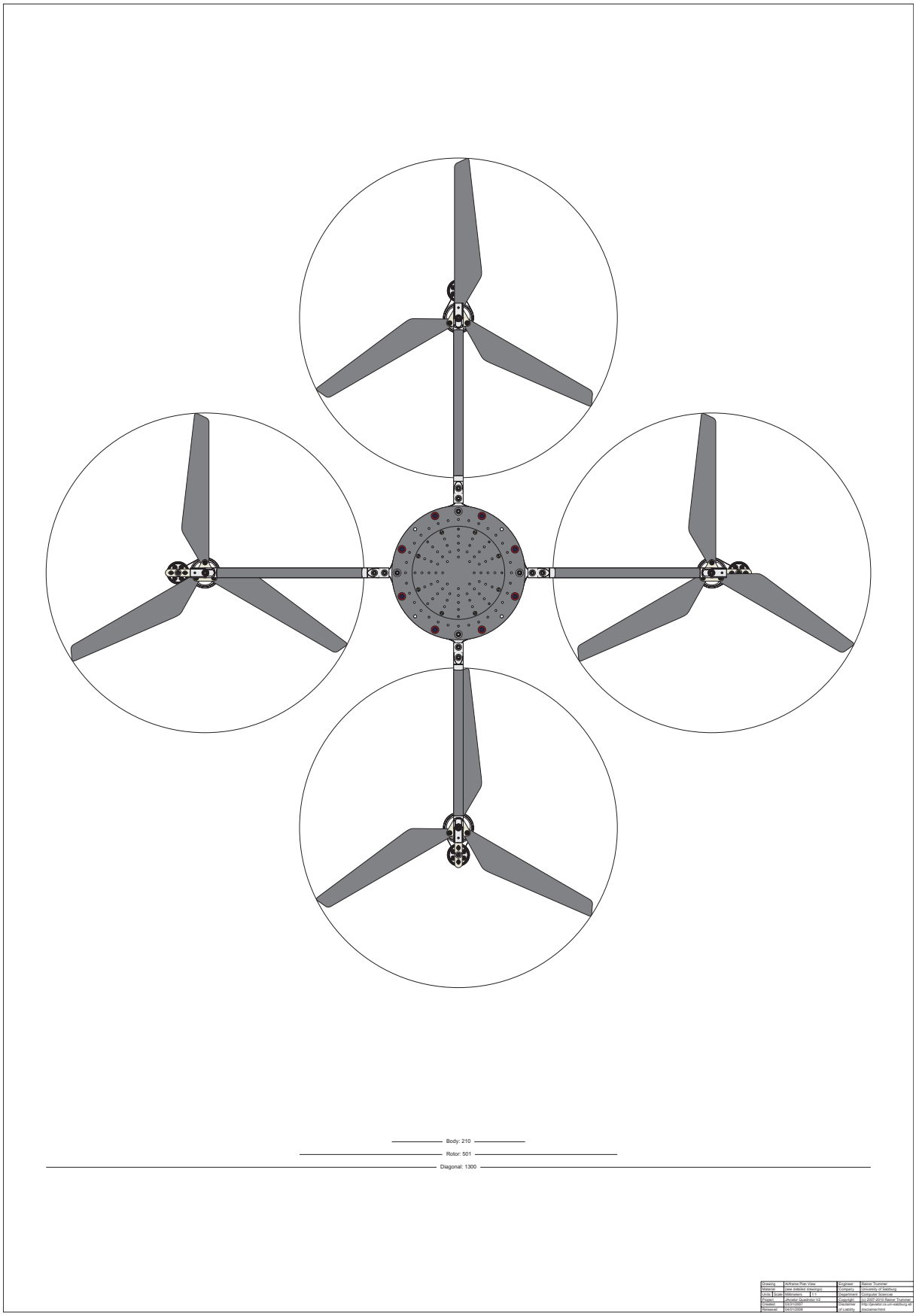


Figure B.17: Plan view of the complete airframe with all propulsion groups.

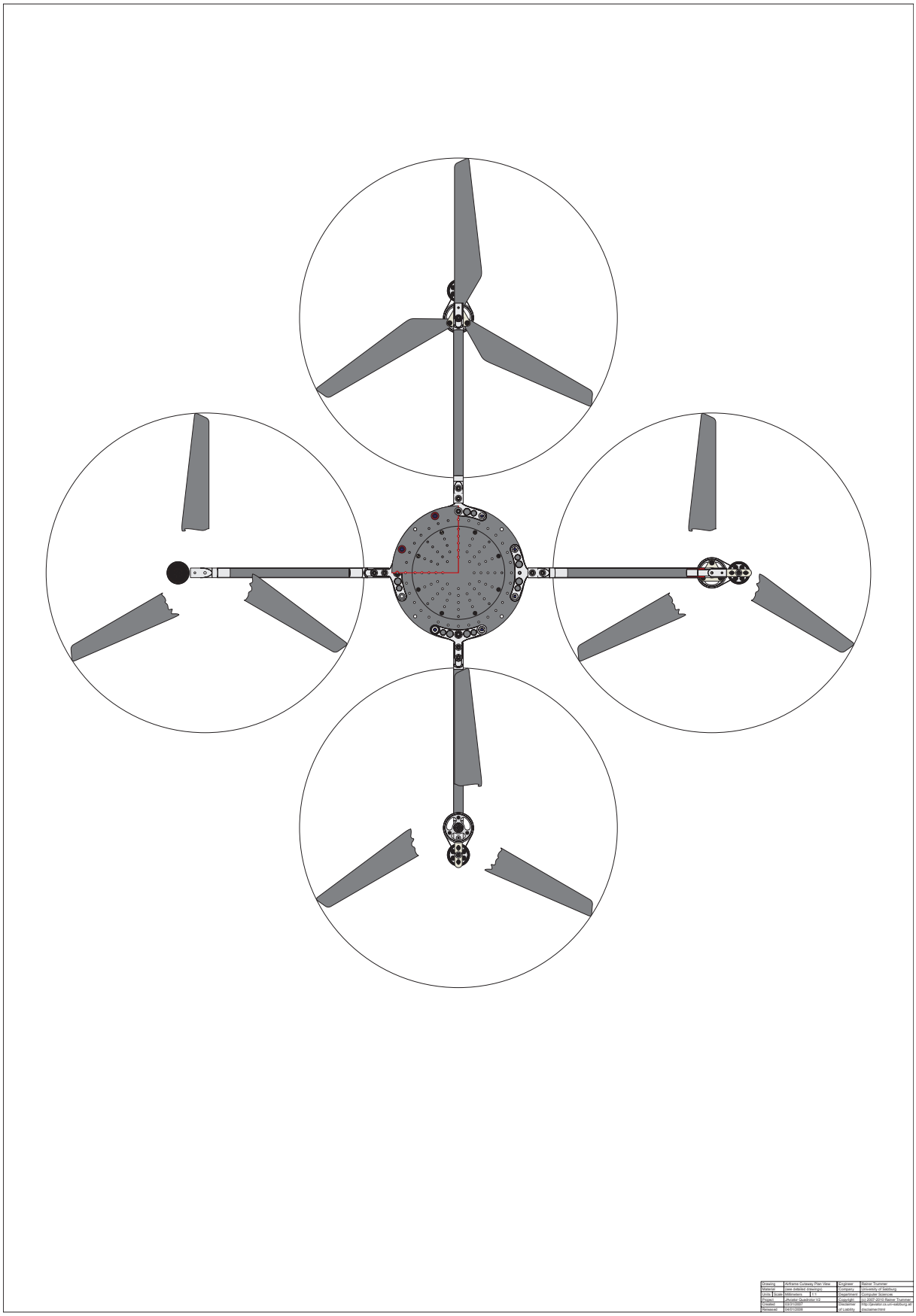


Figure B.18: Cutaway plan view of the airframe with all propulsion groups.

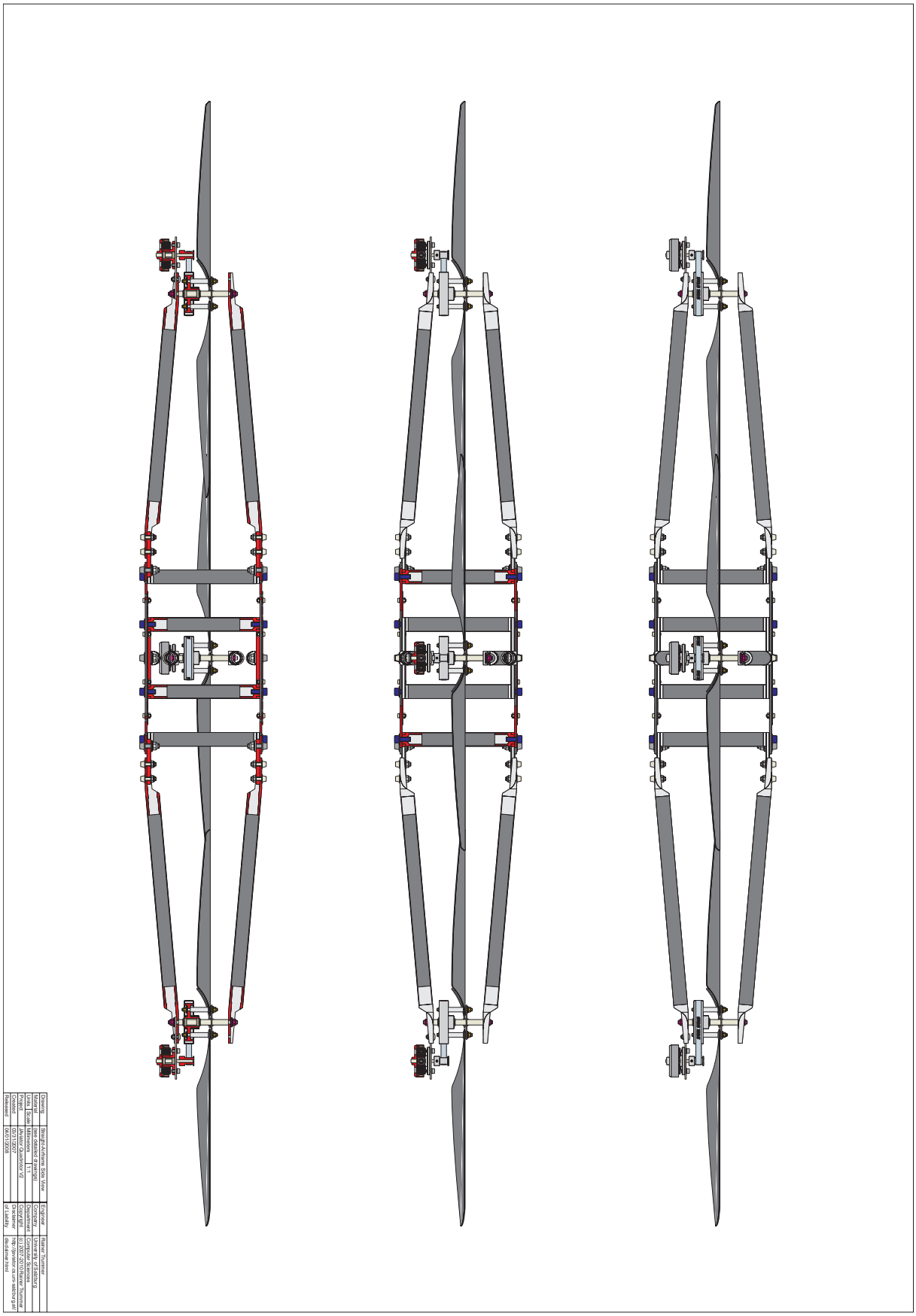


Figure B.19: Straight-airframe side view including all propulsion groups.

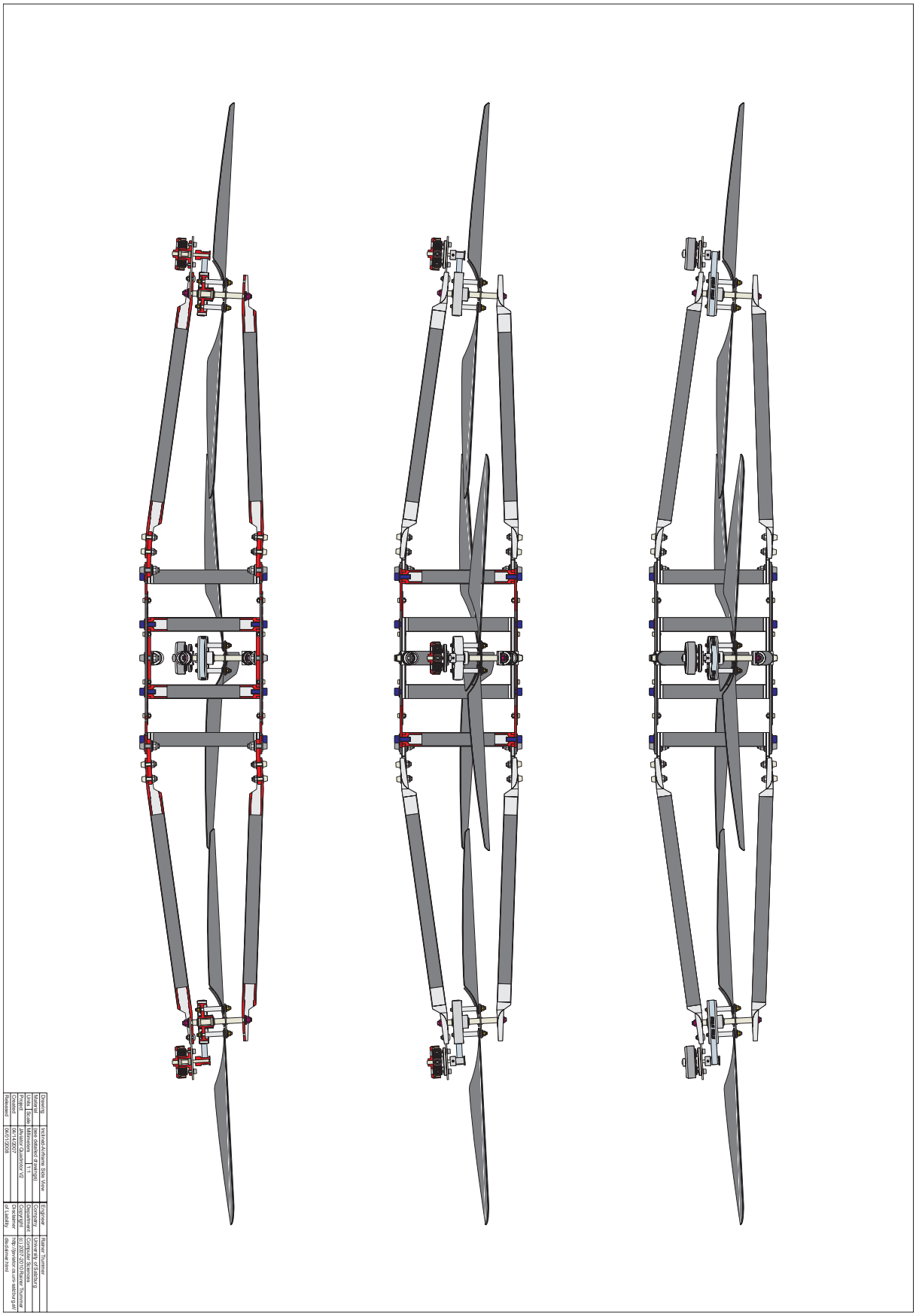


Figure B.20: Inclined-airframe side view including all propulsion groups.

Team Contributions

We need men who can dream of things that never were.

John F. Kennedy (1917−1963), speech in Dublin, Ireland, June 28, 1963

The JAviator software system would definitely not exist in the present form without the achievement of collaborative work with and among the members of the JAviator project team (in alphabetical order):

Joshua Auerbach built the Exotask library [3, 4] and integrated its custom support in the IBM WSRT JVM [105, 106]. He further implemented EControl by porting the JControl algorithms to an Exotask framework and, over the years, contributed to many components of the Java-based software system.

David F. Bacon developed the Metronome [107] garbage collector that is core to IBM's WSRT JVM, and in that scope, worked on various dodgy details, like real-time garbage collection, to ensure reliable operation within the Java environment. He also worked on the TuningFork [103] real-time logging system and integrated the required instrumentation in JControl and EControl.

Silviu S. Craciunas extended the Control Terminal with a 3D visualization environment to offer close-to-reality 3D motion tracking of the JAviator. As a necessary consequence, he extended the MockJAviator from the original 4-DOF version to a full 6-DOF simulator.

Daniel T. Iercan implemented the initial C-based controller that served for performing the very first steps in the JAviator project. During that time, he augmented the Control Terminal with the Signals Dialog to provide real-time diagram views of specific control system data. In the scope of his work on HTL [108], he also developed an HTL-based scheduler for the Exotask runtime system.

Christoph M. Kirsch, as the head of the JAviator project team, did not implement any JAviator-related software by his own. However, he acted as the overall vision proponent and is therefore the initiator behind many concepts and ideas, like the LET model [102] and Exotasks, which were realized and incorporated to some extent.

Vadakkedathu T. Rajan designed the physical algorithms realized in the MockJAviator and further implemented a more advanced attitude and altitude controller in JControl, which were then integrated in EControl and exploited throughout all Java-based flights.

Harald Röck worked on almost all facets of the JAviator software system right from the beginning. Particularly, he implemented the JControl infrastructure and, by porting it to C, established the basis for today's CControl version. He also integrated CControl in the Tiptoe kernel [109, 110], which led to the first version of TControl. Besides this incredible effort, Harald was always my primary source of knowledge whenever I struggled with software-specific problems.

I would like to thank all of these persons for their invaluable endeavor that significantly contributed to the great success of the JAviator project.

Abbreviations

AAS	Army Air Service	19
AC	Alternate Current	11
ADC	Analog-to-Digital Converter	48
AIAA	American Institute of Aeronautics and Astronautics	25
API	Application Programming Interface	51
BEC	Battery Eliminator Circuit	44
BMC	Brushless-Motor Controller	39
BMU	Barometric Measurement Unit	47
CFC	Carbon-Fiber Composite	27
CNC	Computerized Numerical Control	35
CPU	Central Processing Unit	53
DC	Direct Current	11
DCM	Direct Cosine Matrix	59
DOF	Degree of Freedom	66
DT	Differential Threshold	64
DVD	Digital Versatile Disc	113
EKF	Extended Kalman Filter	9
FAI	Fédération Aéronautique Internationale	21
FCS	Flight Control System	88
FET	Field Effect Transistor	39
FIR	Finite Impulse Response	65
FPGA	Field-Programmable Gate Array	7
FPU	Floating-Point Unit	53
FM	Frequency Modulation	27

GCS Ground Control System 88
GPS Global Positioning System 6
HPH Human-Powered Helicopter 24
HTL Hierarchical Timing Language 92
I^{2}C Inter-Integrated Circuit 48
IIR Infinite Impulse Response 65
IMU Inertial Measurement Unit 47
IPS Integrated Pressure Sensor 47
ISM Integral Sliding Mode 51
ITD Interarrival Time Deviation 110
I/O Input/Output 54
JAP JAviator Plant 88
JVM Java Virtual Machine 89
KF Kalman Filter 67
LAN Local Area Network 54
LDS Laser Distance Sensor 47
LET Logical Execution Time 86
LQR Linear Quadratic Regulator 5
LU Lower-Upper 76
OEM Original Equipment Manufacturer 27
PCB Printed Circuit Board 8
PID Proportional Integral Derivative 5
PIDD Proportional Integral Derivative 2nd Derivative 79
PWM Pulse Width Modulation 43
RAM Random Access Memory 53
RC Remote Control 2
RFID Radio Frequency Identification 12
RS232 Radio Sector 232 51
SCT Signals Computation Time 110
SLS Stationary Localization System 47
SPI Serial Peripheral Interface 50
SSKF Steady-State Kalman Filter 7

TCP/IP Transmission Control Protocol/Internet Protocol 91
TTL Transistor-Transistor Logic 39
UART Universal Asynchronous Receiver Transmitter 51
UAV Unmanned Aerial Vehicle 2
UDP User Datagram Protocol 91
UKF Unscented Kalman Filter 7
URF Ultrasonic Range Finder 47
UWB Ultra-Wide Band 51
VTOL Vertical Take-Off and Landing 2
WOP Workload-Oriented Programming 94
WSRT WebSphere Real Time 92

Bibliography

[1] S. S. Craciunas, C. M. Kirsch, H. Röck, and R. Trummer. The JAviator: A high-payload quadrotor UAV with high-level programming capabilities. In *Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC)*, 2008. 1, 5.4.1

[2] J. Auerbach, D. F. Bacon, S. S. Craciunas, D. T. Iercan, C. M. Kirsch, V. T. Rajan, H. Röck, and R. Trummer. The JAviator Project. `http://javiator.cs.uni-salzburg.at`, 2010. 1, 2

[3] J. Auerbach, D. F. Bacon, D. T. Iercan, C. M. Kirsch, V. T. Rajan, H. Röck, and R. Trummer. Java takes flight: Time-portable real-time programming with Exotasks. In *Proc. of the ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, 2007. 1, 5.3.2, B

[4] J. Auerbach, D. F. Bacon, D. T. Iercan, C. M. Kirsch, V. T. Rajan, H. Röck, and R. Trummer. Low-latency time-portable real-time programming with Exotasks. *ACM Transactions on Embedded Computing Systems (TECS)*, 8(2):1–48, January 2009. 1, 5.3.2, B

[5] J. G. Leishman. *Principles of Helicopter Aerodynamics*. Cambrige University Press, New York, NY, second edition, 2006. 1.2, 2, 2.1, 2.2, 2.3, 2.4, 6.1.1

[6] Draganfly Innovations Inc. RC Planes, Helicopters, and Blimps, 2010. `http://www.rctoys.com`. 1.3, 2.8

[7] VeraTech Aero Corp. X-Pro Flyer, 2010. `http://www.veratech.aero/rotorx.html`. 1.3, 2.7

[8] Ascending Technologies GmbH. Multi-Rotor Flying Platforms, 2010. `http://www.asctec.de`. 1.3

[9] MikroKopter Open-Source Project. MikroKopter – HexaKopter, 2010. `http://www.mikrokopter.de`. 1.3

[10] Silverlit Toys Manufactory Ltd. X-UFO, 2010. `http://www.silverlit.com`. 1.3

[11] Microdrones GmbH. MD-200 − MD-1000, 2010. `http://www.microdrones.com/en_home.php`. 1.3

[12] AirRobot GmbH & Co. KG. AR100-B, 2010. `http://www.airrobot.de/englisch/index.php`. 1.3

[13] S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and control of an indoor micro-quadrotor. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004. 1.3

[14] S. Bouabdallah, A. Noth, and R. Siegwart. PID vs. LQ control techniques applied to an indoor micro-quadrotor. In *Proc. of the IEEE/RSJ International Confernece on Intelligent Robots and Systems (IROS)*, 2004. 1.3

[15] G. M. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin. The Stanford testbed of autonomous rotorcraft for multi-agent control (STARMAC). In *Proc. of the Digital Avionics Systems Conference (DASC)*, 2004. 1.3

[16] P. Pounds, R. Mahony, J. Gresham, P. Corke, and J. Roberts. Towards dynamically-favourable quad-rotor aerial robots. In *Proc. of the Australasian Conference on Robotics and Automation (ARAA)*, 2004. 1.3

[17] E. Altuğ, J. P. Ostrowski, and C. J. Taylor. Control of a quadrotor helicopter using dual-camera visual feedback. *The International Journal of Robotics Research*, 24(5):329–341, May 2005. 1.3

[18] P. Castillo, R. Lozano, and A. Dzul. Stabilization of a mini-rotorcraft with four rotors. *IEEE Control Systems Magazine*, 33(1066):45–55, December 2005. 1.3

[19] S. D. Hanford, L. N. Long, and J. F. Horn. A small semi-autonomous rotary-wing unmanned air vehicle (UAV). In *Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC)*, 2005. 1.3

[20] S. L. Waslander, G. M. Hoffmann, J. S. Jang, and C. J. Tomlin. Multi-agent X4-flyer testbed control design: Integral sliding mode vs. reinforcement learning. In *Proc. of the IEEE/RSJ International Confernece on Intelligent Robots and Systems (IROS)*, 2005. 1.3

[21] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin. Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications. In *Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC)*, 2006. 1.3

[22] P. Pounds, R. Mahony, and P. Corke. Modelling and control of a quad-rotor robot. In *Proc. of the Australasian Conference on Robotics and Automation (ARAA)*, 2006. 1.3

[23] A. Tayebi and S. McGilvray. Attitude stabilization of a VTOL quadrotor aircraft. *IEEE Transactions on Control Systems Technology (TCST)*, 14(3):562–571, March 2006. 1.3

[24] G. P. Tournier, M. Valentiy, J. P. How, and E. Feron. Estimation and control of a quadrotor vehicle using monocular vision and Moiré patterns. In *Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC)*, 2006. 1.3

[25] P. Adigbli, C. Grand, J.-B. Mouret, and S. Doncieux. Nonlinear attitude and position control of a micro-quadrotor using sliding-mode and backstepping techniques. In *Proc. of the 3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV) & European Micro Air Vehicle Conference and Flight Competition (EMAV)*, 2007. 1.3

[26] S. Bouabdallah, M. Becker, V. de Perrot, and R. Siegwart. Toward obstacle avoidance on quadrotors. In *Proc. of the International Symposium on Dynamic Problems of Mechanics (DINAME)*, 2007. 1.3

[27] S. G. Fowers, D.-J. Lee, B. J. Tippetts, K. D. Lillywhite, A. W. Dennis, and J. K. Archibald. Vision-aided stabilization and the development of a quad-rotor micro UAV. In *Proc. of the International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2007. 1.3

[28] A. Hably and N. Marchand. Global stabilization of a four-rotor helicopter with bounded inputs. In *Proc. of the IEEE/RSJ International Confernece on Intelligent Robots and Systems (IROS)*, 2007. 1.3

[29] J. Haukrogh, O. Binderup, and S. Gislason. Platform development, sensor modelling, and estimation of a Draganflyer X-Pro. Technical report, Aalborg University, Department of Electronic Systems, Section for Automation and Control, Aalborg, Danmark, 2007. 1.3

[30] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC)*, 2007. 1.3, 6.2.2

[31] J. F. Roberts, T. S. Stirling, J.-C. Zufferey, and D. Floreano. Quadrotor using minimal sensing for autonomous indoor flight. In *Proc. of the 3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV) & European Micro Air Vehicle Conference and Flight Competition (EMAV)*, 2007. 1.3

[32] R. DeNardi and O. E. Holland. Coevolutionary modelling of a miniature rotorcraft. In *Proc. of the 10th International Conference on Intelligent Autonomous Systems (IAS)*, 2008. 1.3

[33] S. Grzonka, S. Bouabdallah, G. Grisetti, W. Burgard, and R. Siegwart. Towards a fully autonomous indoor helicopter. In *Proc. of the IEEE/RSJ International Confernece on Intelligent Robots and Systems (IROS)*, 2008. 1.3

[34] S. Grzonka, G. Grisetti, and W. Burgard. Autonomous indoors navigation using a small-size quadrotor. In *Proc. of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2008. 1.3

[35] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008. 1.3

[36] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin. Quadrotor helicopter trajectory tracking control. In *Proc. of the AIAA Guidance, Navigation, and Control Conference (GNC)*, 2008. 1.3, 6.2.2

[37] C. Hürzeler, J.-C. Metzger, A. Nussberger, F. Hänni, A. Murbach, C. Bermes, S. Bouabdallah, D. Schafroth, and R. Siegwart. Teleoperation assistance for an indoor quadrotor helicopter. In *Proc. of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2008. 1.3

[38] A. Bachrach, R. He, and N. Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217–228, December 2009. 1.3

[39] R. Goel, S. M. Shah, N. K. Gupta, and N. Ananthkrishnan. Modeling, simulation and flight testing of an autonomous quadrotor. In *Proc. of the International Conference and Exhibition on Aerospace Engineering (ICEAE)*, 2009. 1.3

[40] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009. 1.3

[41] T. Zhang, W. Li, M. Achtelik, K. Kühnlenz, and M. Buss. Multi-sensory motion estimation and control of a mini-quadrotor in an air-ground multi-robot system. In *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2009. 1.3

[42] W. Kreil. Cubic UWB-Based Soft Walls for a Micro-UAV. Master's thesis, University of Salzburg, Salzburg, Austria, 2009. 1.4, 3.4.4

[43] C. Gablehouse. *Helicopters and Autogiros.* J. B. Lippincott Company, Philadelphia and New York, revised edition, 1969. 2.1, 2.2, 2.3, 2.3, 2.4, 2.4

[44] J. G. Leishman. The Bréguet-Richet quadrotor helicopter of 1907. *Vertiflite*, 47(3):30–32, 2001. 2.1

[45] G. Apostolo. *The Illustrated Encyclopedia of Helicopters*. Bonanza Books, New York, NY, 1984. 2.1, 2.4

[46] H. F. King. The first fifty years. *Flight Magazine*, 64(2342):753–754, December 1953. 2.1

[47] K. Munson. *Helicopters and other Rotorcraft since 1907*. The Pocket Encyclopedia of World Aircraft in Color. The Macmillian Company, New York, NY, 1969. 2.1, 2.3

[48] M. A. Smith, H. F. King, and J. Yoxall. Background to the helicopter. *Flight Magazine*, 63(2296):92–93, January 1953. 2.1

[49] G. de Bothezat. The genral theory of blade screws. Technical report, NACA TR 29, Washington, DC, 1919. 2.2

[50] Edison National Historic Site. Helicopter designed by George de Bothezat, 2010. `http://www.nps.gov/archive/edis/edisonia/04000000.htm`. 2.2

[51] D. Francis. No helicopter for you tomorrow. *Popular Science*, 144(6):57–61, June 1944. 2.2

[52] W. F. Gerhardt. Straight up! The dream of fliers. *Popular Science*, 105(2):38–39, August 1924. 2.2, 2.3

[53] C. V. Glines. The Flying Octopus. *Air Force Magazine*, 73(10):1–4, October 1990. 2.2

[54] S. Spooner. The de Bothezat helicopter. *Flight Magazine*, 15(9):125, March 1923. 2.2

[55] E. Teale. Planes that go straight up. *Popular Science*, 126(3):13–15, 116, March 1935. 2.2, 2.3

[56] S. Spooner. Notices to airmen − A successful French helicopter. *Flight Magazine*, 16(4):47, January 1924. 2.3

[57] D. Harris. Whole world in race for supremacy in the air. *Popular Science*, 104(2):34–35, February 1924. 2.3

[58] D. Mondey. *The New Illustrated Encyclopedia of Aircraft*. Chartwell Books Inc., Secaucus, NJ, revised edition, 2000. 2.4

[59] J. Stoff. *Historic Aircraft and Spacecraft in the Cradle of Aviation Museum*, chapter Convertawings Model A Quadrotor, page 48. Dover Publications Inc., Mineola, NY, November 2001. 2.4

[60] H. Allaway, F. Rowsome Jr., R. P. Stevenson, and A. C. Jensen. New in aviation. *Popular Science*, 168(2):121, February 1956. 2.4

[61] H. Allaway, F. Rowsome Jr., R. P. Stevenson, and A. C. Jensen. New in aviation. *Popular Science*, 169(3):176, September 1956. 2.4

[62] M. A. Smith, H. F. King, W. T. Gunston, and R. Casey. Helicopters of the World – Convertawings, Inc. *Flight Magazine*, 70(2493):722–723, November 1956. 2.4

[63] M. A. Smith, H. F. King, W. T. Gunston, and R. Casey. Helicopters of the World – Convertawings, Inc. *Flight Magazine*, 73(2565):394, March 1958. 2.4

[64] H. G. Hannant Ltd. Item ANIG7217 from Catalogue, 2010. `http://www.hannants.co.uk/search/?FULL=ANIG7217`. 2.5

[65] S. Harding. *U.S. Army Aircraft since 1947*, chapter Curtiss-Wright VZ-7, pages 93–94. Airlife Publishing Ltd., Shrewsbury, England, 1990. 2.5

[66] M. A. Smith, H. F. King, W. T. Gunston, and R. Casey. From all quaters – Saucers and platforms. *Flight Magazine*, 76(2643):484, November 1959. 2.5

[67] M. A. Smith, H. F. King, W. T. Gunston, and R. Casey. The American industry – Curtiss-Wright Corp. *Flight Magazine*, 78(2684):247, August 1960. 2.5

[68] AHS International. The American Helicopter Society Igor I. Sikorsky Human-Powered Helicopter Competition, 1980. `http://vtol.org/awards/hph.html`. 2.6

[69] G. Lehoux. Human-Powered Helicopters – The history, the technology, the people. `http://www.humanpoweredhelicopters.org`, 2010. 2.6

[70] J. M. Drees. Human-powered helicopter competition heats up! *Vertiflite*, 41(2):32–34, March 1995. 2.6

[71] A. Naito. Unknown problems in human-powered helicopter. In *Proc. of the International Human-Powered Flight Symposium*, 1994. 2.6

[72] C. Roper. The human-powered-flight international symposium, August 1994, Seattle. *Human Power*, 11(4):18–19, 1994. 2.6

[73] R. Sopher. The AHS Igor Sikorsky human-powered helicopter competition. *Vertiflite*, 43(3):32–34, May 1997. 2.6

[74] Spectrolutions Inc. Design History of the Flyer, 2005. `http://www.spectrolutions.com/pdf/flyer_history.pdf`. 2.7, 2.7

[75] Walter Cedric. Roswell Flyer, 2010. `http://www.waltercedric.com/rc-helicopter-mainmenu-65/135-draganflyer-roswell-flyer/171-roswell-flyer-hmx-4-draganflyer.html`. 2.7

[76] M. A. Dammar. Four-Propeller Helicopter, November 2002. http://www.freepatentsonline.com/D465196.html. 2.7

[77] Draganfly Innovations Inc. Draganflyer V-Ti, 2010. http://www.rctoys.com/rc-products-catalog/rc-helicopters-draganflyer-vti.html. 2.7

[78] J. A. Gwozdecki. Aircraft Attitude Sensor and Feedback Control System, January 2001. http://www.freepatentsonline.com/6181989.html. 2.7

[79] Draganfly Innovations Inc., Saskatoon, Saskatchewan. *Draganflyer V-Ti*, 2005. http://www.rctoys.com/pdf/df5ti-manual.pdf. 2.7

[80] N. Sacco. How the Draganflyer flies. *Rotory Modeler Magazine*, online:1–5, 2002. http://www.rctoys.com/pdf/draganflyer3_rotorymagazine.pdf. 2.7

[81] Spectrolutions Inc., Brooklyn Park, Minnesota. *Draganfly X-Pro Flyer*, 2003. http://www.spectrolutions.com/pdf/xproman1.pdf. 2.7

[82] RS Components Handelsges.m.b.H. Electronic, Electromechanical, and Industrial Components, 2010. http://www.rs-components.com. 3.2, 3.4.2

[83] Armin Hassberg Modellbau. Brushless Motors and Accessories, 2010. http://www.ahm-brushless.de. 3.2.1

[84] Spectrolutions Inc. Custom Prototyping and Electronics Design, 2010. http://www.spectrolutions.com. 3.2.2

[85] S. G. Roche. Investigation of Performance Improvements for a Quadrotor UAV. Master's thesis, Cranfield University, Bedfordshire, England, 2007. 3.2.2

[86] Advance Energy Inc. Thunder Power RC Batteries, 2010. http://www.thunderpowerrc.com. 3.3

[87] Modellsport Schweighofer GmbH. RC Models and Accessories, 2010. http://www.der-schweighofer.at. 3.3.2

[88] MHM-Modellbau GmbH. RC Models and Accessories, 2010. http://www.mhm-modellbau.de. 3.3.2

[89] Acroname Inc. Robotics Accessories, 2010. http://www.acroname.com. 3.4.1

[90] SensComp Inc. Global Components, 2010. http://www.senscomp.com. 3.4.1

[91] Elpro Inc. Electronic Components, 2010. http://www.elpro.org. 3.4.2

[92] MicroStrain Inc. Microminiature Sensors, 2010. http://www.microstrain.com. 3.4.3

[93] Dimetix AG. Laser Distance Measurement, 2010. http://www.dimetix.com. 3.4.5

[94] Gumstix Inc. Embedded Development Platforms, 2010. `http://www.gumstix.com`. 3.5.1

[95] D. H. Titterton and J. L. Weston. *Strapdown Inertial Navigation Technology*. Peter Peregrinus Ltd., Stevenage, UK, second edition, 1997. 4.1, 4.1.2, 4.4.2

[96] G. F. Franklin, J. D. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Ellis-Kagle Press, Half Moon Bay, CA, third edition, 1998. 4.1.3

[97] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, Hoboken, NJ, 2001. 4.1.3

[98] D. Simon. *Optimal State Estimation*. Wiley-Interscience, Hoboken, NJ, 2006. 8

[99] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2006. 8

[100] M. T. Heath. *Scientific Computing – An Introductory Survey*. McGraw-Hill, New York, NY, second edition, 2002. 4.4.3

[101] G. M. Hoffmann. *Autonomy for Sensor-Rich Vehicles: Interaction Between Sensing and Control Actions*. PhD thesis, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, 2008. 4.5

[102] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A time-triggered language for embedded programming. *Proceedings of the IEEE*, 91(1):84–99, January 2003. 5.1.1, 5.3.2, 6.3, B

[103] IBM Corporation. *TuningFork Visualization Tool for Real-Time Systems*, 2009. `http://www.alphaworks.ibm.com/tech/tuningfork`. 5.1.3, 5.4.2, B

[104] Ubisense Inc. Precise Real-Time Location, 2010. `http://www.ubisense.net`. 5.1.3

[105] IBM Corporation. *WebSphere Real Time User's Guide*, first edition, 2006. 5.3.1, B

[106] J. Auerbach, D. F. Bacon, B. Blainey, P. Cheng, M. Dawson, M. Fulton, D. Grove, D. Hart, and M. Stoodley. Design and implementation of a comprehensive real-time Java virtual machine. In *Proc. of the ACM International Conference on Embedded Software (EMSOFT)*, 2007. 5.3.1, B

[107] D. F. Bacon, P. Cheng, and V. T. Rajan. A real-time garbage collector with low overhead and consistent utilization. In *Proc. of the ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages (POPL)*, 2003. 5.3.1, B

[108] A. Ghosal, T. A. Henzinger, D. T. Iercan, C. M. Kirsch, and A. L. Sangiovanni-Vincentelli. A hierarchical coordination language for interacting real-time tasks. In *Proc. of the ACM International Conference on Embedded Software (EMSOFT)*, 2006. 5.3.2, B

[109] S. S. Craciunas, C. M. Kirsch, H. Payer, H. Röck, and A. Sokolova. Programmable temporal isolation in real-time and embedded execution environments. In *Proc. of the Workshop on Isolation and Integration in Embedded Systems (IIES)*, 2009. 5.3.4, B

[110] S. S. Craciunas, C. M. Kirsch, H. Payer, H. Röck, A. Sokolova, H. Stadler, and R. Staudinger. The Tiptoe Project. `http://tiptoe.cs.uni-salzburg.at`, 2010. 5.3.4, B

[111] H. Stadler. A Virtualized Real-Time I/O Subsystem. Master's thesis, University of Salzburg, Salzburg, Austria, 2008. 5.3.4

[112] S. S. Craciunas, C. M. Kirsch, and A. Sokolova. A workload-oriented programming model for temporal isolation with VBS. In *Online Proc. of the Workshop on Reconciling Performance with Predictability (RePP)*, 2009. 5.3.4

[113] JTRES 2010. The 8th International Workshop on Java Technologies for Real-time and Embedded Systems, 2010. `http://d3s.mff.cuni.cz/conferences/jtres2010`. 6.2.1

[114] K. Muthukrishnan and M. Hazas. Position estimation from UWB pseudorange and angle-of-arrival: A comparison of non-linear regression and Kalman filtering. In *Proc. of the Fourth International Symposium on Location and Context Awareness (LoCA)*, 2009. 6.2.2