Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

RAMA - a Low-Cost Modular Control System for Unmanned Aerial Vehicles

Doctoral Thesis

Ondřej Špinka

Prague, December 2009

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics

Supervisor: ***Doc. Dr. Ing. Zdeněk Hanzálek***

Any sufficiently advanced technology is indistinguishable from magic.

Arthur Charles Clarke

Acknowledgements

In the first place, I would like to thank my advisor Zdeněk Hanzálek for his passionate support throughout all those years he had to wait for this thesis to materialize (and for funding me). His patience surely had to endure some rather exacting tests in the meantime. My big thanks and hughs also belong to my family and friends, for supporting me in various ways. Namely, I would like to thank Linda Hroníková and Martin Mádlík, for being my thesis-mates and much, much more.

There are many folks that significantly contributed to the RAMA project, and whose help has been invaluable. Namely (in alphabetical order), my big thanks belong to: Ota Herm, for designing the Servo Control Unit, which proved so reliable over the years; Ondřej Holub, for his gratuitous help with the control algorithms and CNC milling; Štěpán Kroupa, for his passionate support and relentless urge to explore all the blind paths; Jiří Novotný, for his great diligence and for laying the foundations of the Main Control Computer; Marek Peca, for the SpejblARM module, sensor data fusion algorithms and other support; Pavel Píša, for his inspiring ingenuity, invaluable advice and deep knowledge, and for being my mentor; Michal Sojka, for his great support, patience and wisdom, not only in the Linux-relates causes; Petr Svoboda, a true genius I had the privilege to supervise during his graduate projects, for the Ground Station and help with the magnetometer; and Miloslav Žižka, for his help with embedded Linux and the motherboard for the Main Control Computer. May the Lord Rama bless them.

I would also like to thank sir Arthur Charles Clarke, a visioneer, philosopher and writer, and to sir Isambard Kingdom Brunell, an ingenious victorian constructor, for inspiring me to become an engineer. And last but not least, my big thanks belong to Jorge Cham and his Ph.D. comics [1], for the great art of procrastination.

This work was supported by the Ministry of Education of the Czech Republic under project 1M0567.

Czech Technical University in Prague *Ondřej Špinka*
December 2009

Abbreviations

ACB	Actuator Battery
ACL	Attitude Control Layer
ACM	Automatic Control Mode
AP	Airborne Part
ARCL	Angular Rate Control Layer
AVB	Avionics Battery
CFM	Critical Failure Mode
CRC	Cyclic Redundancy Check
DAM	Data Acquisition Module
DU	Decoupling Unit
EC	Electronic Container
EMC	Electromagnetic Compatibility
FOD	Foreign Object Debris
GPS	Global Positioning System
GS	Ground Station
IMU	Inertial Measurement Unit
IPS	Internal Power Supply
MCC	Main Control Computer
MCM	Manual Control Mode
MCU	Micro Controller Unit
NU	Navigation Unit
PCB	Printed Circuit Board
PCL	Position Control Layer
PIO	Parallel Input-Output
RC	Radio Control
RPM	Revolutions Per Minute
SACM	Semi-Automatic Control Mode
SCU	Servo Control Unit
TAM	Three-Axis Magnetometer
TCAS	Traffic Collision Avoidance System
TDCE	Telemetry Data Composition Error
TMM	Telemetry Measurement Message
TSM	Time Synchronization Message
TTL	Trajectory Tracking Layer

VB	Vehicle Bus
VCL	Velocity Control Layer
VIC	Vector Interrupt Controller
WCU	Wireless Control Unit
WDCU	Wireless Data Communication Unit
YPR	Yaw, Pitch and Roll

RAMA - a Low-Cost Modular Control System for Unmanned Aerial Vehicles

Ing. Ondřej Špinka
Czech Technical University in Prague, 2009

Thesis Advisor: Doc. Dr. Ing. Zdeněk Hanzálek

This thesis presents a modular control system for Unmanned Aerial Vehicles (UAVs), the RAMA control system. RAMA is a universal, lightweight and compact autopilot for small UAVs, designed and built at the Department of Control Engineering, Faculty of Electrical Engineering of the Czech Technical University in Prague. RAMA is designed as a modular, distributed, hierarchical system, designed with emphasis on reliability and safety. It is fault-tolerant, implementing a novel architecture, utilizing the graceful degradation and a simple reconfiguration techniques to maintain the most critical functions. RAMA serves as a relatively cheap, open and modular research platform, allowing to test various hardware and software solutions and data acquisition and control algorithms. The data and technical expertise is publicly shared, allowing other researchers, interested in the field, to participate on the project, or to use our results to support their own work. RAMA is an excellent platform for student work, mainly in the form of semestral or graduation projects. It is also an outstanding tool for the PR (Public Relations) and advertising purposes, attracting attention of future students. The design and construction, both hardware and software wise, along with the control algorithms and testing results, are shown in this document.

Goals and Objectives

The goals of this work were set as follows:

1. To design and develop a rotorcraft-based UAV (Unmanned Aerial Vehicle) with necessary on-board avionics (both hardware and software).
2. To develop the basic control scheme and the basic control laws.
3. Perform flight experiments and acquire flight data, in order to prove the feasibility of the proposed solutions.
4. Thoroughly document the whole project and make the technology and results publicly available at the Internet.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) are used in many different fields of human business nowadays. Although being developed mainly for the military in the past, a lot of other applications emerged recently. UAVs are becoming popular with the emergency services, aerial photography, movie making industry, agriculture or traffic observations. Their operation and maintenance costs are very favorable compared to manned vehicles and they might also be used in a rather hazardous environment, where the deployment of a manned vehicle would prove too dangerous. The UAVs are usually much smaller and lighter than their full-scale counterparts, being able to fly in much more restricted space.

Rotorcrafts are often being used as the vehicle of choice for airborne unmanned systems, being popular mainly for their outstanding maneuverability and the ability to hover. Small helicopter is capable of very aggressive maneuvering as well as stable hovering and usually has an excellent self-to-payload weight ratio.

In some areas, such as military or traffic or agriculture observations, full autonomy, high reliability and long flight duration are required, leading to complex and potent vehicle designs. The avionics of these drones is very sophisticated, robust and often redundant, designed in a very similar manner as for the manned aircraft. Obviously, such vehicles are capable of carrying out complex mission profiles. On the other hand, they also require relatively complex maintenance and ground support systems, and are extremely expensive. Let us note the well-known U.S. Air Force Predator drone or the Yamaha RMAX R-UAV [2] as an example.

In other areas, like in the movie making industry, aerial photography or emergency services, long-duration missions and full autonomy of the vehicle are often not required. On the contrary, small, light, simple and cheap drones are wanted, providing a semi-automatic, stable flying platform for light payloads, mainly cameras. The term “semi-automatic” means that the vehicle is not fully autonomous - it is still op-

Figure 1.1: A rotorcraft UAV equipped with the RAMA system

erated by a human pilot, however, the autopilot provides some sort of stabilization. Cheap and lightweight avionic is required for those vehicles, yet providing a reasonable safety margin for mission-critical functions. Such vehicles are also popular in academia for research case studies.

Strict weight, power consumption and price limitations put on the avionics of these vehicles does not allow to double hardware components. A necessary redundancy for the most critical functions can be achieved due to the *system reconfiguration* and *graceful degradation* in case of a failure. The purpose of this is to "jury rig" the system, sacrificing the less important functionality to allow the most critical tasks to be carried out. This usually leads to a mission loss, but can prevent the vehicle loss and other possible consequences, due to a potentially dangerous craft coming completely out of control.

1.2 Outline and Contribution

This thesis presents a case-study of low-cost, universal and modular design, the RAMA UAV control system (Figure 1.1). RAMA stands for Remotely operated Aerial Model Autopilot.

The intension was to build a relatively cheap, light, open and modular research platform, allowing to test various hardware and software solutions and data acquisition and control algorithms. The data and technical expertise is publicly shared, as was mentioned above, allowing other researchers interested in the field to participate on the project, or to use our results to support their own work. RAMA is an excellent platform for student work, mainly in the form of semestral or graduation projects. In fact, important parts of the RAMA project were developed by students (as will be

mentioned in the text) and overall, about eight students carried out their graduation projects on it. It is also an outstanding tool for the PR (Public Relations) and advertising purposes. RAMA is not designed as a final product; it is rather a test bed, but can serve as a good basis for possible future ventures in the UAV field.

Although designed as an universal autopilot, RAMA is currently focused on a rotorcraft control and is tested using a small helicopter. Therefore, the bulk of this thesis is focused on rotorcraft control.

Original contributions of this thesis are as follows:

- The modular, distributed and hierarchical architecture of the RAMA system (see chapter 3) is original and relatively novel in the field of small UAV autopilots; no comparable design is known to the author currently. This architecture allows to implement the graceful degradation and reconfiguration safety principles, as well as it simplifies software and hardware upgrades and testing procedures (as described in chapters 3, 6 and 7). Its innovative value was also recognized by the editors and reviewers of a prestigious engineering journal [Špinka et al.(2009a)]. The whole hardware and software design and implementation are original.

- RAMA is an open-source project, meaning that in-depth technical information and data are publicly shared at the project website [Špinka(2007)]. This makes RAMA especially appealing, not only for the academia community. There are not many projects of this type around; the only other the author is aware of is the Paparazzi project [3], but this particular project is focused somewhat differently (see section 1.3).

- Further contributions of this work include original control scheme and algorithms (chapter 4) and the testing results (chapter 7), which are backed by a vast pool of real flight data.

The thesis organization is as follows: The first chapter defines goals and assets of the work, introduces the project and discusses the related work. The second chapter is only informative and its purpose is to define basic terms and principles of the helicopter flight, needed for proper understanding of the following text. It may be skipped by a reader who is familiar with this topic. The third chapter presents the hardware architecture of RAMA control system, followed by description of the control scheme and algorithms in chapter 4. Chapter 5 explains the software of the system, while chapter 6 describes the fault-tolerance and safety features of the system. Chapter 7 presents the experiments and testing results. The last chapter concludes the thesis and discusses the results and future work.

1.3 Related Work

A lot of work had already been done in the field of UAV control systems. There is almost inexhaustible amount of various control designs that were published among the years. Let us note at least those that directly influenced the RAMA project, or have much in common.

A very informative outlook on small rotorcraft mathematical modeling is given in [4], [5] and [6]. These works present a complex non-linear model (and derive a linearized state-space version), along with identification methods and a case-study, showing parameters and modeling results of a specific hobby-helicopter.

The works [7], [8], [9], [10] [11], [12] and [13] deal with control of small rotorcraft. In [7], a novel SDRE (State-Dependent Riccati Equation) low-level stabilizing controller is presented. This option is very interesting from the theoretical point of view because of its optimality and was considered for the RAMA system, but was later abandoned in favor of a more conventional cascaded PID control, which, although less optimal, is a lot easier to implement, set-up and test. A cascaded PID approach is also more convenient from the point of fault-tolerance, as it simplifies the graceful degradation scheme implementation. The cascaded-PID approach is demonstrated in [8], and following work in [9] and [10]. In [11] and [12], an interesting artificial intelligence approach to hover control, using a neural network, is presented. It was not considered for implementation on RAMA for the same reasons that eventually led to the rejection of the SDRE, but represents another original approach to the low-level rotorcraft stabilization problem. In [13], a hierarchical flight control scheme for UAVs is given, which eventually inspired our own control scheme, described in chapter 4.

A good general methodology for platform-based design and systematic integration of control systems, dealing with implementation constraints and reuse of previously designed components, is given in [14]. A framework for dynamic reconfigurability and graceful degradation in embedded systems is presented in [15]. Another approach, called the Simplex architecture, is shown in [16]. The dependability and reconfiguration of the distributed embedded systems is also dealt with in [17] and [18]. Those are theoretical works, which inspired some of the solutions implemented in the RAMA system, namely the graceful degradation scheme. The specific requirements and properties of the real-time [19][20][21] and embedded systems [22] have been considered in the software design.

UAV navigation, sensor data fusion and vehicle guidance are handled in [23], [24] and [25]. Those works inspired our own approach to sensor data fusion, described in section 4.2.

An interesting and complex rotorcraft UAV project, called MARVIN, can be found in [26]. The goals of this project are similar to RAMA, however, the design philosophy is somewhat different. Both systems were designed as low-cost and mainly implement off-the-shelf components. The MARVIN system is more complex than

RAMA and offers a more sophisticated functionality. On the contrary to RAMA, it can operate fully autonomously and is able to take-off and land automatically, which RAMA cannot do in its present state. On the other hand, RAMA was designed with much more emphasis put on the modularity, extensibility and fault-tolerance. Another important asset of the RAMA system is its open design, as was explained in the previous section.

Another interesting UAV autopilot project is the Paparazzi [3]. It is an open-source project and provided inspiration for RAMA. It presents a simple solution, suitable for many non-demanding applications. Its scope is somewhat different to RAMA; Paparazzi is intended to provide a simple and cheap solution to the end-users, while RAMA,on the other hand, is a good research platform. The main difference is that RAMA provides a broadband telemetry, modular structure and powerful computing platform, all being very important assets for research and development. On top of that, RAMA's project website also offers recorded flight data for theoretical researchers.

Chapter 2

Flight Mechanics and Mathematical Modeling of a Rotorcraft

Before describing the functional principles, control algorithms and testing results of the RAMA project, it is important to have at least some very basic understanding of the helicopter flight mechanics, its characteristics and properties. This chapter is only informative, but should provide the necessary insight, needed for proper understanding of the rest of this thesis. It was compiled using sources [27], [28], [6], [4] and [5], and the personal experience and knowledge of the author.

2.1 Definition of the Coordinate System

At the beginning, it is necessary to define the coordinate systems (Figure 2.1 - this figure was taken from the Wikipedia, the free encyclopedia), which will be used throughout this thesis. Standard aerospace system was adopted for the RAMA project. Two reference frames are defined; The *earth frame* and the *body frame*. Both are orthogonal Cartesian frames. The earth frame xyz has the x axis pointing to the north, the y axis pointing to the east and the z axis pointing downwards. The body frame XYZ is oriented to have the X axis pointing forward along the fuselage, the Y axis pointing to the right and the Z axis pointing down, to form a right-hand Cartesian frame. The X, Y and Z axes are sometimes called the roll, pitch and yaw axes of the vehicle.

The attitude of the vehicle in space is expressed by the three Euler angles [29]. zyx rotation is used to obtain the transformation from the earth frame to the body frame - meaning that the first rotation, by angle ψ, takes place around the z axis, the second rotation (by angle ϕ) around the newly-defined y' axis and the last rotation (by angle θ) around the new x'' axis (the x'' axis already coincides with the X axis of the body frame XYZ). This set of rotations yields the XYZ frame from the xyz.

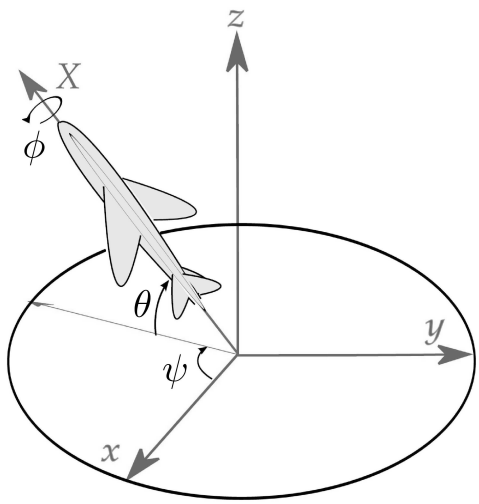


Figure 2.1: Definition of the yaw, pitch and roll angles. The axes Y and Z are not shown in the picture for the sake of clarity. (courtesy Wikipedia)

The attitude angles ψ, ϕ and θ are called the yaw, pitch and roll angles of the vehicle (sometimes denoted YPR angles for short). The conversion from one reference frame to the other might be expressed by the rotation matrix R [29]:

$$R = R_1.R_2.R_3 \tag{2.1}$$

where the matrices R_1, R_2 and R_3 represent the respective rotations.

As is widely known, the Euler angles suffer from singularities, occurring for the 90° tilt angles. This singularities cause the presence of the *gimbal lock* phenomenon [30] in the attitude representation. The gimbal lock is a condition when a degree of freedom is lost and two attitude angles become dependent. For example, for 90° pitch angle the roll and yaw angles become dependent - the resulting attitude of the vehicle is be the same if the values of roll and yaw are swapped. Consider following sequence of rotations: yaw by 45°, pitch by 90° and roll by -30°. The resulting attitude is the same as for the sequence: yaw -30°, pitch 90° and roll 45°. The presence of a gimbal lock is indicated by one of the rotation matrices R_1, R_2 or R_3 in equation 2.1 becoming an identity matrix.

The gimbal lock is not a problem for the attitude representation though; the attitude could still be unambiguously determined, given the attitude angles. It only becomes a problem if Euler angles are used for the trajectory reconstruction (it is very informative to study the history of the NASA's Apollo project and the case when the gimbal lock nearly claimed the lives of the Apollo 13 crew [31][32]).

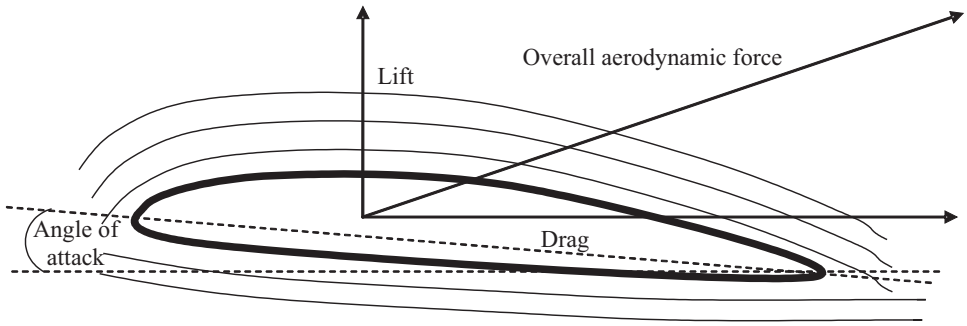


Figure 2.2: Angle of attack definition

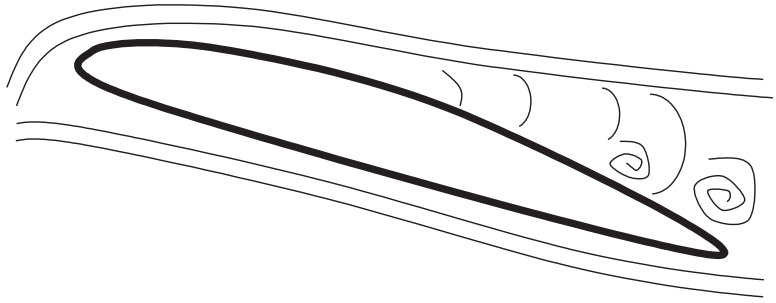


Figure 2.3: Stall condition - forming of turbulence

2.2 Flight Mechanics of a Rotorcraft

2.2.1 Basic Aerodynamics

It is a well-known fact [27] that an airfoil profiled body (or *any* body for that matter), subjected to the air flow, generates the *drag* and the *lift*. Both drag and lift are non-linearly dependent on the relative speed of the flow and the *angle of attack* (Figure 2.2), which is a relative angle between the airflow vector and the airfoil axis. The higher the angle of attack is, the higher is the generated lifting force (and the drag force alike) [27], up until the point of *stall* - this is a point where the laminar flow around the lifting body becomes turbulent (Figure 2.3). The angle of attack, corresponding to the situation when stall just begins to occur (for a given airfoil), is called the *critical angle of attack*. When the stall condition happens, the lifting force drops suddenly and very sharply - there is almost a complete loss of lift in this situation. Similar rule applies also to the relative speed of the airflow - the generated lift and drag increase non-linearly with the increasing speed, up to the *critical Mach number* of the body. Weird things begin to happen at critical Mach numbers and for the sake of simplicity, we shall assume that the airflow speed is always well below that limit for the rest of this text. This condition is met for most of the UAVs (although the speed of the tips

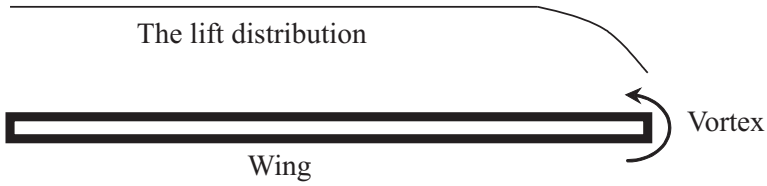


Figure 2.4: Lift distribution along a wing

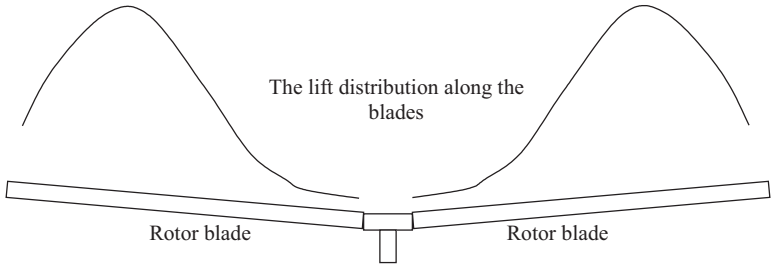


Figure 2.5: Lift distribution along the rotor blades

of the rotor blades or propellers might be actually closer to the critical Mach number than one might intuitively think).

An infinitely long wing, subjected to a constant speed airflow along its whole length, would generate uniform lift per length unit. For the real-life wings the situation is somewhat more complicated, because they actually *have* to end somewhere. Naturally, there is a pressure difference between the top and the bottom side of a wing; that is how the lift is generated. At the wing tip, this difference produces a parasite airflow from the bottom side to the top (as can be seen at Figure 2.4), creating a *vortex* at the wing tip. This vortex creates turbulece and decreases the lift generated by the outward portion of the wing, so the lift force generated along the length of a wing decreases towards the wing tip, as depicted in Figure 2.4.

In case of a rotorcraft, the situation is a little more complicated. The lifting force of the vehicle is generated by the main rotor. We may consider the rotor blades as "wings", but those wings are not subjected to the uniform airflow. The angular velocity of the rotor might be considered constant, but the circumferential velocity is obviously not. It increases along the blade from the root to the tip, where it is at maximum. The lifting force distribution along the rotor blades is depicted in Figure 2.5 - it increases (non-linearly, of course) along with the circumferential speed, up to the point where the tip vortex comes into play; then it starts to fall again to zero, reached at the blade tip.

The situation becomes even more complicated when the vehicle is not hovering, but flying at a constant velocity. Consider the case where a helicopter is flying forward, as shown in Figure 2.6. Assume that the rotor is turning clockwise. The air flow

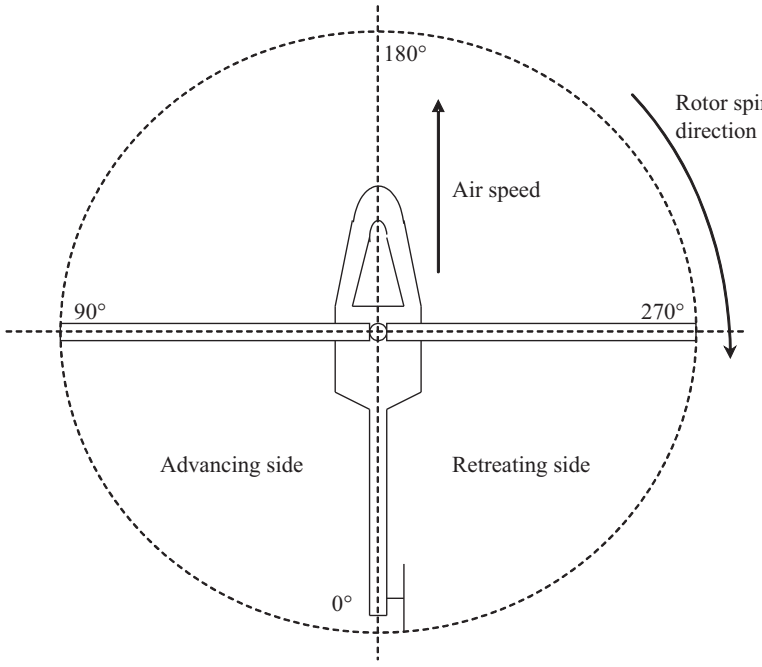


Figure 2.6: Definition of the azimuth angle

speed along a part of the blade is now given by the vector addition of the current circumferential velocity vector at the given point and the negative air speed of the vehicle (the vector of the airflow is logically directly opposite to the vehicle airspeed). It is obvious that the air flow speed (and consequently the generated lift) is not distributed uniformly along the rotor disc, but depends on the current angle of the blades relative to the velocity vector (uniform to the vehicle axis) - called the *azimuth angle*.

The azimuth of a blade is the angle of the blade relative to the main axis of the vehicle, as shown in Figure 2.6. At $90°$, a blade produces maximum lift, because the airflow generated by the movement of the vehicle blows directly opposite to the blade; at $270°$ azimuth the situation is reversed and the blade generates the lowest lift. The lift distribution in this case (one blade at $90°$ azimuth and the other at $270°$) is depicted in Figure 2.7. It is obvious that a blade to the left of the fuselage (azimuth in the interval of $(0°, 180°)$) is traveling against the "wind" caused by the vehicle movement, and is consequently generating more lift than the opposite blade to the right (in our case). The left side is therefore called the *advancing side* and the right side the *retreating side*. Advancing and retreating sides are always defined relative to the vehicle velocity vector. And in case the wind is blowing... Well, the point is surely clear enough by now.

The disbalance in the generated lift naturally produces a torque, acting at the vehicle. It might seem intuitive that in the case described above, this torque would cause

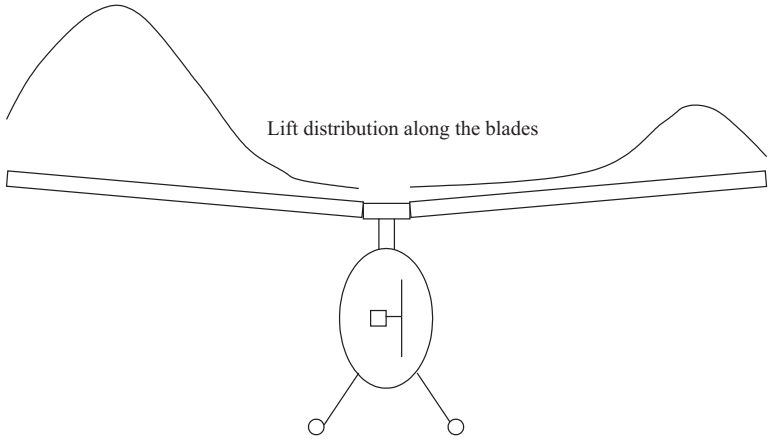


Figure 2.7: Lift distribution along the rotor blades in forward flight

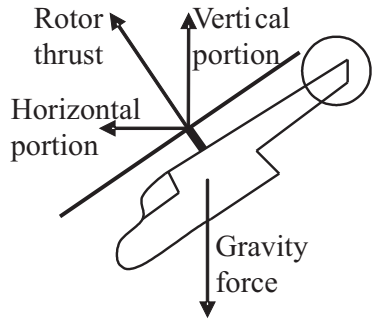


Figure 2.8: A helicopter accelerating forwards

the vehicle to roll to the right - but here comes the surprise. In fact, the torque in this case causes the vehicle to *pitch backwards*. This is because the gyroscopic effect of the rotor spin comes into play. This effect is causing a phase shift (a phase delay) between the lift force and the corresponding torque; this shift is typically by $90°$ (but could be less or more, depending on many factors, creating not only longitudinal, but also a lateral torque). Therefore, the maximum torque is acting with one blade at $180°$ azimuth and the other at $0°$, causing the vehicle to pitch backwards. This effect works as a kind of natural negative feedback in the velocity control of the vehicle. In order to fly forward, the vehicle naturally has to pitch forward (to cause the portion of the rotor lifting force to accelerate the vehicle forwards as shown in Figure 2.8). With increasing speed, the vehicle tends to straighten again, decreasing the forward acting force in effect.

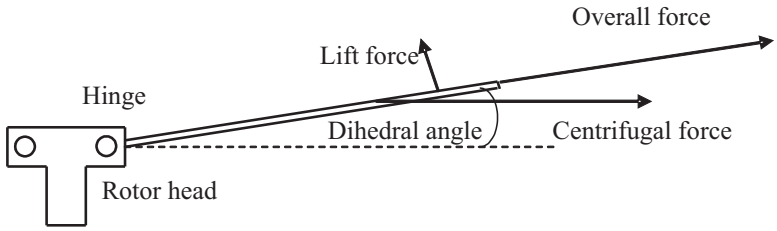


Figure 2.9: Main forces acting upon a rotor blade

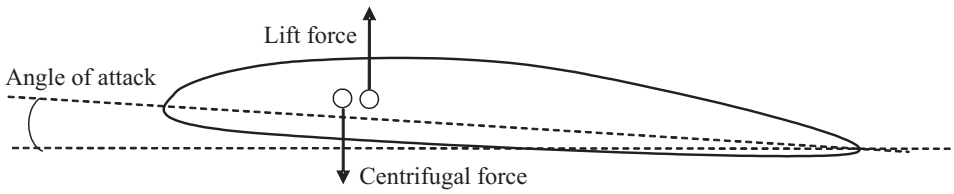


Figure 2.10: Forces acting upon a blade in side cut view - stable situation

2.2.2 Rotor Blade Dynamics

Assume a helicopter in hover. There are two main forces acting on a rotor blade; the aerodynamic lift force, which was discussed in previous section, and the *centrifugal force*. Both forces do not share a common point of action; The acting point of the centrifugal force is situated in approx 58% of the length of the blade, while the acting point of the lift force is located a little more outwards [28] (Figure 2.9).

The vector sum of those two vectors determines the *dihedral angle* of the blade, measured between the plane perpendicular to the rotor axis and the blade. The rotor blades are usually hinged to allow for that movement, but the movement is restricted and damped by very stiff rubber dampers. So, in reality, the dihedral angle would be much less than the “natural” angle would be, because of the dampers restricting the movement. It is worth mentioning that the centrifugal force is normally about an order of magnitude greater than the lifting force, giving a relatively shallow dihedral angle even without the dampers. In practice (with the dampers involved), the dihedral angle of the blades is typically around $1 - 4^{\circ}$.

Let us now examine the effects of those two forces on the angle of attack of a blade, assuming that the blade is flexible and therefore can bend. The carbon fibre composite rotor blades, used on most UAVs, are usually very stiff and do not bend as much as the blades of their full-scale counterparts, so this aspect can be safely neglected on most of the UAVs; but it is still worth mentioning and plays a significant role on bigger rotorcrafts.

It is very important to have the acting point of the centrifugal force in front of the acting point of the lifting force, as depicted at Figure 2.10. In this case, the torque,

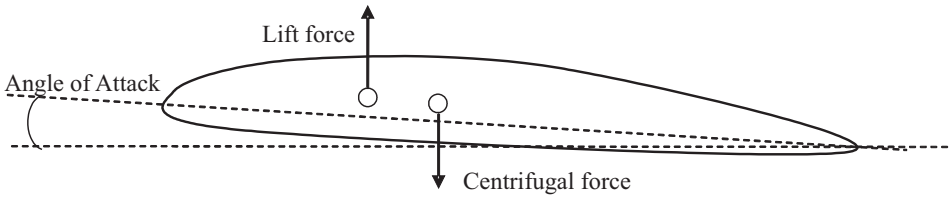


Figure 2.11: Forces acting upon a blade in side cut view - unstable situation

which is bending the blade, would tend to reduce the angle of attack, because the leading edge of the blade is bent downwards. This would in turn reduce the lifting force a little and an equilibrium is therefore quickly established, so the blade is stable (a negative feedback loop is present).

On the contrary, if the acting points of the forces would be swapped (as shown in Figure 2.11, the blade would be unstable. The torque would bend the leading edge of the blade upwards, increasing the lift, and therefore further increasing the torque (a positive feedback loop is established). This case might be actually stable too (given the structural parameters of the blade), but often leads to the effect called *flutter*. The angle of attack would increase, increasing the lift in turn, up to the point of stall (remember the previous section), when the critical angle of attack is reached. At this moment, the lifting force would rapidly decrease and the blade would snap back down because of its flexibility. This would suddenly reduce the angle of attack again and the cycle would repeat. This is obviously a very dangerous behavior, which often leads to the structural failure of the blade, suffering from this phenomenon. The mass of a blade must therefore be distributed so that the points of action of both forces are in stable configuration.

Let us now consider the blade dynamics in the forward flight. Recall the disbalance of lift, generated by the rotor blades in that case, as described in previous section. The hinges, which allow for the camber movement of the blades to accommodate the dihedral angle, also serve as a partial mitigation of this problem.

Because the lifting force is increasing in the azimuth interval $(270°, 90°)$, the blade would travel upwards (gradually increasing its dihedral angle) in the interval of $(0°, 180°)$, reaching peak at $180°$. Again, this is because the gyroscopic effect comes into play; the maxim dihedral angle is reached $90°$ *after* the peak force. Because the peak force is acting at $90°$, peak dihedral angle is achieved at $180°$. Analogically, the blade would travel down between the azimuth of $(180°, 0°)$, with the lowest point occurring at $0°$. This periodic vertical movement is known as the *blade flapping*, and it is one of the most important phenomenons of the helicopter flight. Let us investigate its consequences.

The blade flapping reduces the *effective angle of attack* of the blade traveling upwards and increases the effective angle of attack of the blade going downwards. This is because the blades no longer move directly against the airflow generated by

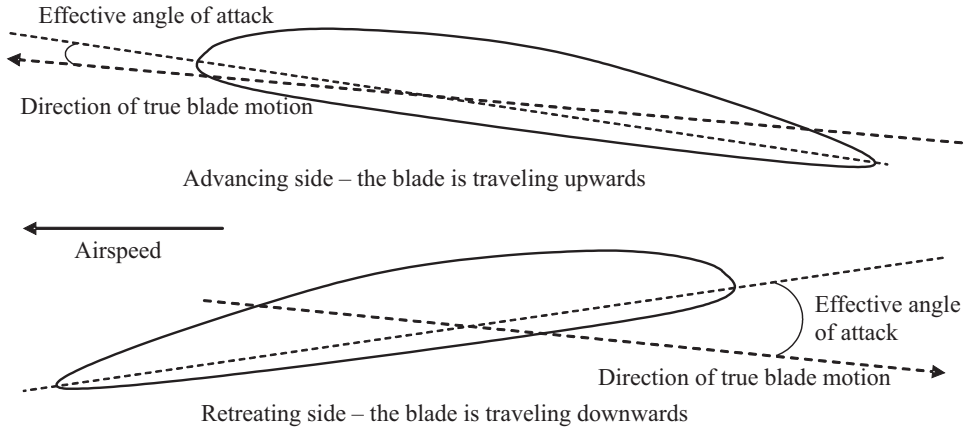


Figure 2.12: Definition of the effective angle of attack

the movement of the helicopter; their effective speed is now given by the vector sum of the velocity airflow vector, circumferential speed vector and the current flapping speed vector. This effect can be clearly seen in Figure 2.12. Because of this, in ideal case, the disbalance of lift, induced by the movement of the vehicle, would be negated by the effect of the blade flapping. In reality, it is not quite so, because the dampers in the camber hinges prevent the blades from flapping freely, so some of the disbalance still persists - but it is somewhat smaller than what would be the case without the hinges. Why are the camber dampers necessary will be explained in the next section.

In reality, there are many other effects affecting the blade flapping, but those are deliberately neglected in this text, for the sake of simplicity.

2.2.3 Helicopter Control Fundamentals

In this section, the actuators and basic control principles of a helicopter shall be described. Basically, a helicopter has five inputs - the engine control (controlling the torque, which drives the main rotor), the collective control of the main rotor blades (controls the magnitude of the thrust the main rotor produces), the longitudinal and lateral cyclic control of the main rotor blades - controlling the “tilting” of the thrust vector, i. e. producing the rolling and pitching moments - and the tail rotor pitch (please note that the word “pitch” has a double meaning - it could mean the pitch of the helicopter body, or it could be used to describe the angle of attack of the blades; it is somewhat unfortunate, but it is a tradition). The tail rotor pitch controls the amount of thrust produced by the tail rotor, which is in turn used to control the yaw of the helicopter.

2.2.3.1 Collective and Cyclic Controls

The thrust of both the main and tail rotors is controlled by tilting (feathering) of their blades along their longest axis (changing their angle of attack). The blades are attached to the rotor head using the *feathering shaft*, to make the tilting possible. The speed of the main rotor is kept constant throughout the flight, as well as the speed of the tail rotor (which is usually driven by a drive shaft or a drive belt from the main rotor gear at a fixed gear ratio, so its speed is directly proportional to the main rotor speed). The engine is kept at constant speed all the time; it only has to react to the variable demand of torque (with is proportionally dependent on the produced thrust). A simple, independent P or PD controller could be used to control the engine speed; even a blunt feed-forward gain, derived from the collective control, is viable and often used in practice. It works well enough, because there is a lot of inertia in the whole system. The *collective control* tilts both blades of the main rotor together by the same angle, therefore controlling the amount of thrust produced by the rotor. The tail rotor control works in exactly the same manner. It is important to note that both rotors are able to produce thrust in both directions; i. e. the main rotor is able to thrust both upwards and downwards and the tail rotor clockwise as well as counterclockwise. For acrobatic flying, the amount of thrust produced by the main rotor is set to be symmetrical, meaning that the same thrust is available in both directions, allowing for inverted flight. The tail rotor is usually set so it can produce a little more thrust in the direction it has to compensate for the counter torque of the engine; i. e. counterclockwise for a clockwise spinning main rotor. This setting allows for the same available yawing speed in both directions.

In order to tilt the helicopter (i. e. to make it roll or pitch), the *cyclic control* is used. There is the longitudinal cyclic for the pitch and the lateral cyclic for the roll tilting. The cyclic control is based on periodic changes of the angle of attack of the rotor blades with respect to their azimuth. It is best to explain its function using an example.

Imagine we want to induce a left roll. This is achieved by increasing the angle of attack of a blade in the azimuth interval of $(0°, 180°)$ (with peak coming at $180°$) and decreasing that angle in the opposite interval (which is $(180°, 0°)$), as shown in Figure 2.13. Now recall the section 2.2.2 and the blade flapping phenomenon. This action would cause the blades to reach the maximum dihedral angle at the azimuth of $270°$ and the minimum at $90°$, tilting the imaginary *rotor plane* to the left in effect (Figure 2.14). Also, this action produces a torque, tilting the helicopter body in the same direction.

Now, remember the camber dampers, mentioned in previous section. As was already said, those dampers are in place to prevent unrestricted blade flapping. This is necessary in order to have some control authority over the vehicle. Imagine that there would be no dampers at all and the blades would be left to flap freely. In this case, the cyclic control would cause the blades to flap, but the flapping would have absolutely

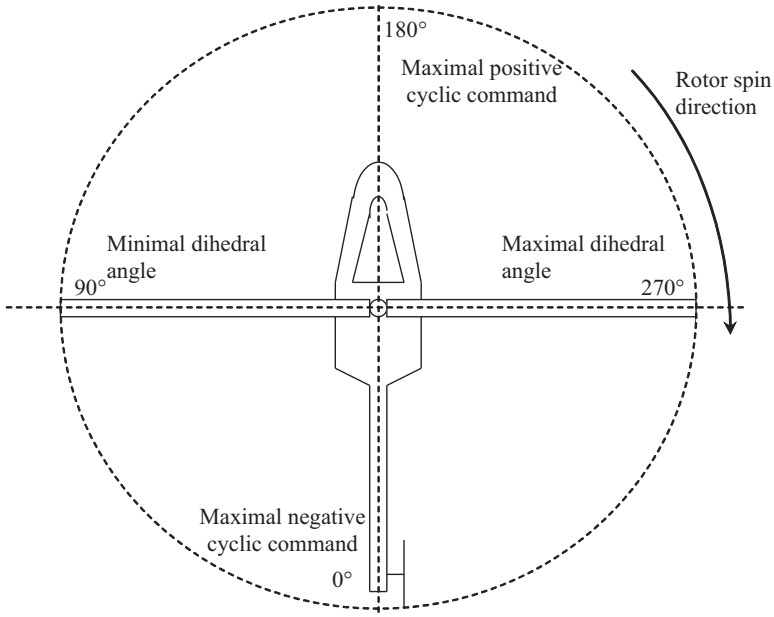


Figure 2.13: Application of the lateral cyclic control - tilting the helicopter to the left

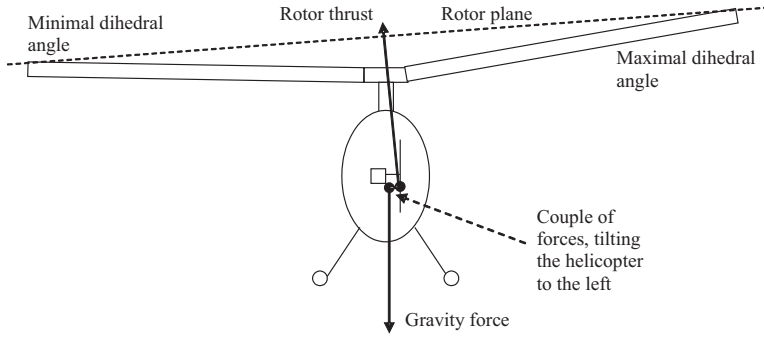


Figure 2.14: Rotor plane tilting due to the blade flapping

no effect on the helicopter body, because there would be no way how to translate the produced tilting torque from the blades to the body. So the dampers are necessary to transfer the torque from the blades to the fuselage. The stiffness of these dampers determines how much control authority the vertical movement of the rotor blades has over the helicopter body; the stiffer the dampers are, the more control authority is there. In turn, the more *adverse* effect would have the lift disbalance in the flight, as explained in section 2.2.2.

The longitudinal cyclic control works analogically. Let us now summarize the effects of the collective and the cyclic control of the main rotor blades. The collective

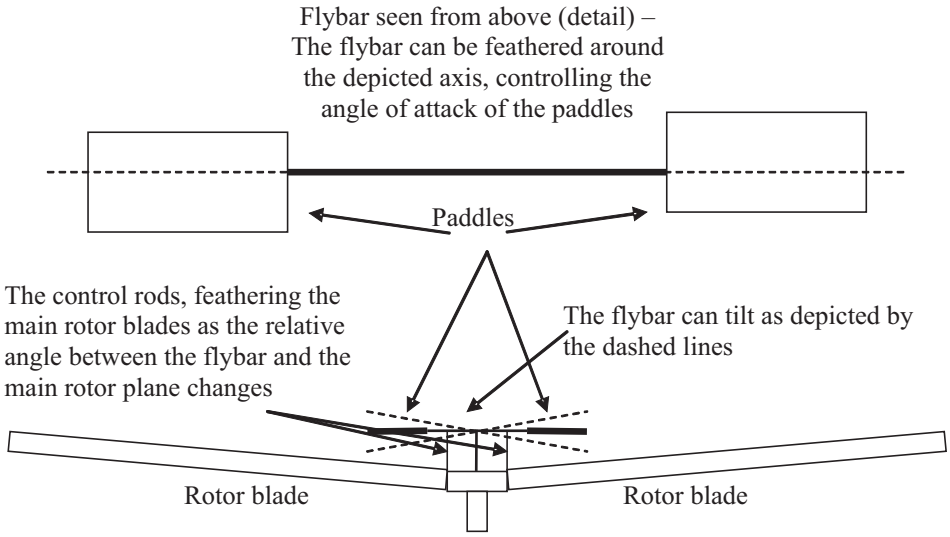


Figure 2.15: The flybar and the Bell stabilizer

control determines the *basal* angle of attack of both blades, controlling the overall amount of produced thrust; the cyclic control periodic blade tilting is *superposed* to that basal angle, inducing lift variations in the course of one revolution of the rotor, in turn producing the *torque*. The longitudinal and lateral cyclic could be naturally combined, so it is possible to induce both roll and pitch commands in one moment, causing the vehicle to tilt, for example, left forwards.

The problem with the cyclic control is that it usually has *too much* authority over the vehicle, causing too rapid response for a human pilot to handle; moreover, the reaction is strongly non-linear. As the bank angle of the vehicle increases, so is increasing the torque; it is obvious from the Figure 2.14. The gravity vector direction does not change, but the thrust vector acting point is shifting further away from it. This causes the spin of the vehicle to *accelerate*. At the bank angle of 90°, the situation reverses and the spin begins to decelerate again. So, when making a 360° roll (a very basic aerobatic figure) for example, the natural response of the vehicle is somewhat "herky-jerky". This is very unpleasant behavior from the pilot's perspective.

Therefore, a mechanical rate stabilizing system was devised right at the advent of the helicopter flight in the 1940s. This was the Bell's system, later improved by Hiller, hence commonly known as the Bell-Hiller system. This system is obsolete nowadays, because it can be easily replaced by electronic gyro-stabilizers, but it is still very common in the UAV field. The Freya helicopter used in the RAMA project does have the Bell-Hiller system too, and therefore it in necessary to explain its function.

The original Bell's system consisted of a *flybar* (Figure 2.15), which is a shaft, connecting two paddles, mounted over (or under) the main rotor. The flybar is usually mounted perpendicular to the rotor blades (seen from above). It can tilt freely, so the

paddles can move up and down, and it can also be feathered, meaning that the angle of attack of the paddles is under control. The flybar is connected with the main blades feathering system by control rods, so that the tilt of the flybar causes tilting of the main rotor blades in the corresponding direction.

The flybar behaves like a small rotor; it would tilt (meaning the plane of its rotation would tilt) if the paddles are feathered. It also behaves like a gyroscope; it has the tendency to maintain its plane of rotation, so if the helicopter (along with the main rotor) would increase its bank angle (in any direction), the flybar would maintain its previous position (there are no camber dampers which would prevent it to do so). So, the plane of the flybar and the main rotor would no longer be parallel; and the flybar would therefore tilt the main rotor blades (via the connecting rods) and induce a countering cyclic control, up until the point when the planes of rotation are parallel again.

Similarly, if a command is applied onto the flybar, causing its plane of rotation to tilt, the flybar would induce a cyclic command to the main rotor, proportional to the relative angle of the flybar and main rotor disc; the grater the angle is, the greater is the command. As the helicopter begins to accelerate its rotation, this angle would decrease, reducing in turn the applied cyclic command.

So, the Bell's system behaves like a purely proportional controller; The flybar tilting determines the set point and the relative angle between the flybar and the main rotor discs is the control error. The cyclic actuation is applied solely to the flybar and only the flybar controls the cyclic feathering of the main blades. The p constant of this controller is given by the weight and size of the paddles. The more the paddles weight, the more gyroscopic - and stabilizing - effect they have, so making them heavy would stabilize the system, but increase the reaction time to the setpoint change; on the contrary, making them lighter makes the response of the system faster, for partial loss of stability.

This system somewhat linearized the angular rate response of the helicopter, but induced a lag in the cyclic control response. The reaction to a setpoint change was somewhat lazy and the pilots were still not completely happy. Therefore, the system was improved by Mr. Hiller to make the response faster. Hiller introduced a feed-forward branch into the controller; he added control rods, which feed a portion of the setpoint directly to the main blades (so, a little portion of the actuator movement is applied directly to the blades and a greater portion to the flybar). This system retains the stability of the Bell's system, but makes the response to the setpoint variations faster.

It was determined that the Bell-Hiller stabilizer consumes a significant portion of the engine power; typically about 15-20%. It is therefore much better to get rid of this system completely and introduce the electronic rate stabilizers, as the RAMA system does.

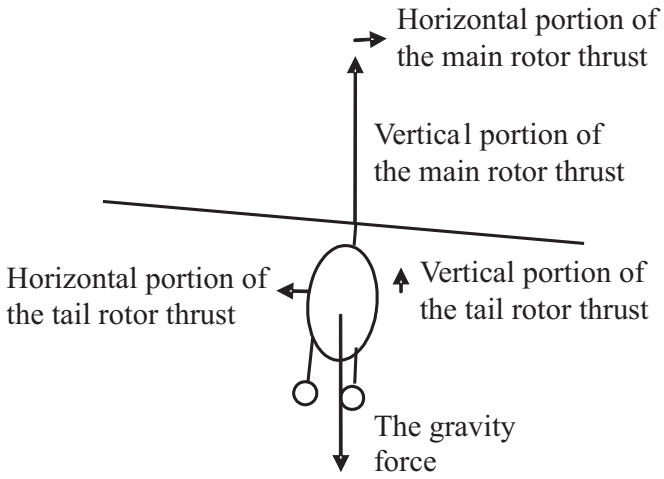


Figure 2.16: Forces acting upon a helicopter in hover

2.2.3.2 Tail Rotor Control

The tail rotor serves to control the yaw of the helicopter and is actuated in exactly the same manner as the collective control of the mail rotor (as was mentioned earlier). In a steady state, the tail rotor must compensate for the torque reaction, induced by the engine. This torque reaction acts in opposite direction than the engine torque; therefore, if the rotor revs clockwise, the fuselage tends to spin counterclockwise. Hence, in this case, the tail rotor must thrust to the left, In order to produce a clockwise counter-torque. This thrust naturally induces not only the torque (with respect to the main rotor axis, where the tail boom serves as a lever), but also draws the helicopter to that direction (i. e. to the left). To compensate this draw, the helicopter must bank to the right slightly, so that the main rotor could produce a counter-thrust (otherwise, the helicopter would accelerate leftwards). So, in the hover, the helicopter is never completely straight; it has to bank slightly, in order to maintain the steady state. The bank angle is typically around $3 - 5^\circ$ (Figure 2.16).

This bank also tilts the tail rotor slightly upwards, so now it tends to pitch the helicopter slightly forwards. However negligible this portion of the tail rotor thrust might be, it would still bank the helicopter slightly forwards in the long term, causing it in turn to accelerate slightly forwards. So, this thrust must be compensated by a tiny backward pitch of the vehicle. However, the bank angle in this direction is almost negligible in reality.

2.2.4 Effects of the Rotor Speed

As was said earlier, the rotor speed is kept constant throughout the flight. This is because the rotor speed has great influence on the control authority of the helicopter.

With increasing rotor speed, the controls are becoming more responsive, because the control surfaces (i. e. the blades) produce greater amount of thrust at the same angle of attack. The relation between the rotor speed and the generated lift is strongly non-linear, so the speed variations would largely disrupt the control loops. Either the control gains would have to be set over-conservatively, reducing the effectiveness of the control loops, or some sort of adaptivity (usually the gain scheduling) would have to be incorporated. Either way, the controllability of the helicopter would be compromised nevertheless. Hence, it is very desirable to minimize the rotor speed variations. Fortunately, this is not so hard, as was mentioned in the previous section, because of the great amount of inertia accumulated in the rotor.

The variations of the air speed of the helicopter in flight naturally have the same effect as rotor speed variations, but in this case, the influence is somewhat lower, because the flight speed is typically seven or eight times lower than the circumferential speed of the rotor tips; hence, the rotor speed variations have much greater effect than air speed variations. Usually, the PID control loops (the lowest-layer, i. e. the angular rate control loops) are robust enough to cope with these variations under reasonable flight conditions (slow flight); however, it might beneficial to incorporate some sort of adaptivity (gain scheduling), if it is desirable to exploit the full flight envelope of the vehicle.

2.2.5 Centre of Gravity

The *Centre of Gravity* (CG) position is naturally of vital importance for any flying vehicle, because it greatly influences the stability and controllability of the vehicle. For helicopters, the CG should be positioned right at the main rotor axis, as low as possible. The lower the CG is, the greater is the natural stability of the vehicle, because the lever between the acting point of the lift force and the CG would be greater (think of the pendulum effect).

If the CG position would be shifted away from the main rotor axis, the main rotor thrust force and the gravity force would form a permanent torque, which would tend to turn the vehicle (either in the roll or the pitch), which would have to be permanently compensated for by the cyclic control. This is naturally undesirable, so the helicopter should be always balanced so that the CG is located on the main rotor axis.

2.3 Mathematical Model of a Rotorcraft

A lot of mathematical models of helicopters have been published in the past. However, it is not so easy to find an appropriate model, suitable for a miniature helicopter, as one might think. This is caused by considerable differences between miniature helicopters and their full-scale counterparts. The first major difference is caused just by the size itself, although it is being overlooked surprisingly often. There is no “scale

preservation law" in the physics and small machines may behave quite differently compared to their big equivalents.

Talking about helicopters, the moments of inertia decreases with the fifth power of the scale factor, while the rotor thrust decreases proportionally to the body mass, that is with the third power. Therefore, it can be easily seen that the small rotorcrafts feature a lot more favorable weight-to-thrust ratio, compared to the big ones. Given this difference, it is hardly surprising that small helicopters are lot more agile and maneuverable and can bear heavier payloads, compared to their weight, than normal choppers. Thrust of a hobby helicopter can easily exceed its weight two or three times, a measure never reached by the full-scale rotorcrafts. Hence, the response to the actuators (the collective and cyclic controls, as well as the tail rotor pitch) is much more aggressive, giving the little machine the ability to spin around an arbitrary axis at a ratio up to 360° per second and to produce astonishingly fast ascent or flight speed. The hobby helicopters are also able to produce negative thrust, allowing for rapid descents or inverted flight. So clearly the scale difference has a considerable impact to all vehicle characteristics.

Another differences are given by the construction of the rotor head and characteristics of the rotor blades. Vast majority of small rotorcrafts utilize the two-blade rotors, often equipped with the Bell-Hiller stabilizer. This system was abandoned at full-scale helicopters in the 1950s. Moreover, the hobby helicopters usually use very stiff blades, commonly made of carbon-fiber composites. On the other hand, full-scale choppers usually use multi-blade rotor heads with very flexible blades. That results in significantly more complex rotor and blade flapping dynamics compared to the small helicopters. In fact, a lot of non-linearity in the full-scale helicopter dynamics is caused by the flexibility of rotor blades itself.

It is therefore clear that mathematic models of the full-scale helicopters are not directly applicable to the small ones, and so an adopted model is needed, better describing their unique features. Fortunately, mathematical description of a small-scale rotorcraft is considerably simpler compared to the big one. This is thanks to their simpler rotor dynamics given by the simple rotor head construction and relatively stiff blades. These features effectively eliminate a lot of non-linearity incorporated in the common full-scale rotorcraft models. Also the number of the state variables is considerably lower. While complex helicopter models (for example [33]) incorporate nearly 100 state variables, the well-known small helicopter model [6] has only 17. So, although being non-linear and inherently unstable, small-scale helicopters can be mathematically described by somewhat simpler models then the big ones.

Although not very numerous, some sophisticated small helicopter models exist. An interesting study on this topic is presented in [4]. In the RAMA project we plan to utilize the model presented by Gavrilets *et al* in [6], although the Freya helicopter parameters have not been identified yet. This model is therefore briefly described in the following sections.

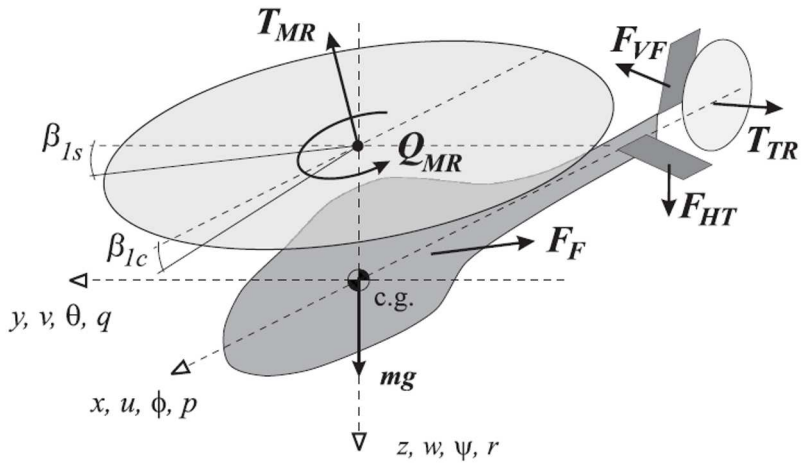


Figure 2.17: Forces and moments acting on a helicopter (courtesy V. Gavrilets)

2.3.1 Helicopter Parameters

The state variables and parameters of the mathematical model are arranged in the table 2.1. Meaning of the most important state variables is shown in the Figure 2.17 (this Figure was taken from [6]).

Table 2.1: Helicopter parameters

Parameter	Description
m	helicopter mass
I_{xx}	rolling moment of inertia
I_{yy}	pitching moment of inertia
I_{zz}	yawing moment of inertia
K_β	hub torsional stiffness
γ_{fb}	stabilizer bar lock number
$B^{nom}_{\delta_{lat}}$	lateral cyclic to flap gain
$A^{nom}_{\delta_{lon}}$	longitudinal cyclic to flap gain
K_μ	scaling of the flap response to the speed variation
Ω_{nom}	nominal main rotor speed
R_{mr}	main rotor radius
c_{mr}	main rotor chord
a_{mr}	main rotor blade lift curve slope
$C^{mr}_{D_0}$	main rotor blade zero lift drag coefficient
continued on the next page	

Table 2.1 – continued from previous page

Parameter	Description
$C_{T_{max}}^{mr}$	main rotor maximal thrust coefficient
$I_{\beta_{mr}}$	main rotor blade flapping inertia
R_{tr}	tail rotor radius
c_{tr}	tail rotor chord
a_{tr}	tail rotor blade lift curve slope
$C_{D_0}^{tr}$	tail rotor blade zero lift drag coefficient
$C_{T_{max}}^{tr}$	tail rotor maximal thrust coefficient
n_{tr}	main-to-tail rotor gear ratio
n_{es}	engine-to-shaft gear ratio
δ_r^{trim}	tail rotor pitch trim offset
S_{vf}	effective vertical fin area
$C_{L_\alpha}^{vf}$	vertical fin lift curve slope
ϵ_{vf}^{tr}	fraction of the vertical fin area exposed to the main rotor wash
S_{hf}	horizontal fin area
$C_{L_\alpha}^{hf}$	horizontal fin lift curve slope
P_{eng}^{idle}	engine idle power
P_{eng}^{max}	engine maximal power
f_p^s	rolling resonance frquency of the suspension system
f_q^s	pitching resonance frquency of the suspension system
f_r^s	yawing resonance frquency of the suspension system
ξ^s	damping ratio of the suspension system material
S_x^{fus}	frontal fuselage drag area
S_y^{fus}	side fuselage drag area
S_z^{fus}	vertical fuselage drag area
h_{mr}	main rotor hub height above the centre of gravity
l_{tr}	tail rotor hub location behind the centre of gravity
h_{tr}	tail rotor height above the centre of gravity
l_{hf}	stabilizer location behind the centre of gravity
x	longitudinal body frame axis
y	lateral body frame axis
z	vertical body frame axis
u	longitudinal body frame velocity
v	lateral body frame velocity
w	vertical body frame velocity
ϕ	roll angle
θ	pitch angle
ψ	yaw angle
p	rolling angular velocity

continued on the next page

Table 2.1 – continued from previous page	
Parameter	**Description**
q	pitching angular velocity
r	yawing angular velocity
X_{mr}	in-plane x axis rotor force
Y_{mr}	in-plane y axis rotor force
Z_{mr}	in-plane z axis rotor force
X_{fus}	fuselage drag along the x axis
Y_{fus}	fuselage drag along the y axis
Z_{fus}	fuselage drag along the z axis
Y_{tr}	tail rotor thrust along the y axis
Y_{vf}	vertical fin drag along the y axis
Z_{hf}	horizontal fin drag along the z axis
L_{mr}	main rotor rolling moment
M_{mr}	main rotor pitching moment
Q_e	engine yawing moment
M_{hf}	horizontal fin pitching moment
L_{vf}	vertical fin rolling moment
N_{vf}	vertical fin yawing moment
L_{tr}	tail rotor rolling moment
N_{tr}	tail rotor yawing moment

2.3.2 Rigid Body Dynamics

Basically, the helicopter dynamics is modeled as a *rigid body dynamics*, with all the forces and moments, generated by the rotor, tail rotor, engine, aerodynamic drag etc. acting on it. The most common way to describe the rigid body dynamics is to use the well-known *Newton-Euler equations of motion*. In this case, the cross products of inertia were neglected:

$$\begin{aligned}
\dot{u} &= vr - wq - gsin\,\theta + (X_{mr} + X_{fus})/m \\
\dot{v} &= wp - ur - gsin\,\phi\,cos\,\theta + (Y_{mr} + Y_{fus} + Y_{tr} + Y_{vf})/m \\
\dot{w} &= uq - vp - gcos\,\phi\,cos\,\theta + (Z_{mr} + Z_{fus} + Z_{hf})/m \\
\dot{p} &= qr\,(I_{yy} - I_{zz})/I_{xx} + (L_{mr} + L_{vf} + L_{tr})/I_{xx} \\
\dot{q} &= pr\,(I_{zz} - I_{xx})/I_{yy} + (M_{mr} + M_{hf})/I_{yy} \\
\dot{r} &= pq\,(I_{xx} - I_{yy})/I_{zz} + (-Q_e + N_{vf} + N_{tr})/I_{zz}
\end{aligned} \tag{2.2}$$

Meaning of all the variables is explained in table 2.1 in the previous section. The inertial velocities may be obtained from the equations above using a coordinate transformation, either by Euler angles [29] or Quaternions [34]. As was already said, the Euler angles are simpler, but due to the goniometric functions properties they have

singular points for the angles of $\pm 90°$.

2.3.3 Main Rotor Forces and Moments

2.3.3.1 Thrust

The *main rotor thrust* is dependent on a lot of conditions. Most important factors of these are the collective pitch angle θ_0, the lift curve slope of the rotor blades a_{mr} and the advance ratio μ. We assume the air inflow to be steady and uniform, an assumption that is well justified [6]. We may also neglect the blade flapping angles, as they are of no great importance here, because the cyclic control influence is of second order for small advance ratios $\mu < 0.15$. It may be also well assumed that the main rotor blades have no twist, an assumption well justified by the stiffness of the carbon fiber blades. Resulting set of equations is therefore considerably simpler compared to the one describing the full-scale rotor behavior [33], which has to take all those factors into account.

Following system of equations is based on the momentum theory and is to be solved iteratively. Maximum thrust T_{max} has to be introduced, since momentum theory does not take the aerodynamic blade stall effect into account (see section 2.2.1 for explanation). This boundary value must be determined empirically. For the sake of simplicity, the index mr is omitted in the following equations.

$$C_T = \frac{T}{\rho \left(\Omega R\right)^2 \pi R^2} \tag{2.3}$$

$$\lambda_0 = \frac{C_T}{2\eta_w \sqrt{\mu^2 + \left(\lambda_0 - \mu_z\right)^2}} \tag{2.4}$$

$$C_T^{ideal} = \frac{a\sigma}{2}\left(\theta_0\left(\frac{1}{3} + \frac{\mu^2}{2}\right) + \frac{\mu_z - \lambda_0}{2}\right) \tag{2.5}$$

$$C_T = \begin{cases} C_T^{ideal} & \text{if } -C_T^{max} \leq C_T^{ideal} \leq C_T^{max} \\ -C_T^{max} & \text{if } C_T^{ideal} < C_T^{max} \\ C_T^{max} & \text{if } C_T^{ideal} > C_T^{max} \end{cases} \tag{2.6}$$

$$C_T^{max} = \frac{T^{max}}{\rho \left(\Omega R\right)^2 \pi R^2} \tag{2.7}$$

Where

$$\mu = \frac{\sqrt{\left(u - u_{wind}\right)^2 + \left(v - v_{wind}\right)^2}}{\Omega R}$$

is the advance ratio,

$$\mu_z = \frac{w - w_{wind}}{\Omega R}$$

is the normal airflow component, and

$$\sigma = \frac{2c}{\pi R}$$

is the solidity ratio.

Coefficient η_w represents the power loss due to the non-uniform velocity of the rotor blades and non-ideal pressure wake contraction. It was determined empirically [6] to be approx. 0.9.

Note that in hover, when the vertical velocity is equal to the inflow velocity, the denominator of the term 2.4 becomes zero. This corresponds to the *vortex ring state* phenomenon, which cannot be described by the momentum theory. The vortex ring state is very dangerous, extremely rare flight condition that occurs only at hover, and is caused by the vortexes, forming at the rotor blade tips (see section 2.2.1). When the vertical velocity equals the inflow speed, those vortexes become extremely large and therefore the rotor downwash is forced outwards and upwards, and the same air is sucked through the rotor all over again. This forms a closed loop between the air intake and outlet, and the helicopter literally sucks itself to the ground. The more collective, the worse this effect becomes. There are only two ways out; either to pitch the vehicle using the cyclic control and therefore disturb the equality of the velocities, or to disengage the clutch between the engine and the rotor and enter autorotation. In both cases, it is extremely important that the pilot realizes what is going on in time and reacts immediately, for with decreasing height and increasing descent rate the recovery becomes more and more difficult.

2.3.3.2 Torque

The *main rotor torque* is given by the sum of the torque, induced due to the generated thrust, and the drag of the blades. Because the drag is not significantly dependent on the angle of attack, it could be treated constant. Induced yawing moment is then given by:

$$Q_{mr} = C_Q \rho \left(\Omega R\right)^2 \pi R^3 \tag{2.8}$$

Where

$$C_Q = \frac{Q}{\rho \left(\Omega R\right)^2 \pi R^3} \;=\; C_T \left(\lambda_0 - \mu_z\right) + \frac{C_{D_0}\sigma}{8}\left(1 + \frac{7}{3}\mu^2\right) \tag{2.9}$$

2.3.3.3 Blade Flapping Dynamics and the Main Rotor Moments and Forces

The *blade flapping* phenomenon is very complex and would not be comprehensively described here. Just a few of the most important facts shall be treated. More detailed information can be found in [6].

The blade flapping angle is a function of it's azimuth angle ψ (Figure 2.6):

$$\beta(\psi) = a_0 + a_1 cos\,\psi + b_1 sin\,\psi \tag{2.10}$$

Flapping of the flybar (Bell-Hiller stabilizer) is represented by:

$$\beta_s(\psi) = a_{1s} sin\,\psi + b_{1s} cos\,\psi \tag{2.11}$$

where the constant term is omitted, since no coning takes place. The flybar contributes to the main blades pitch angle due to the connecting rods:

$$\theta(\psi) = \theta_0 + \theta_{lon} sin\,\psi + \theta_{lat} cos\,\psi + k_s \beta_s \tag{2.12}$$

And the resulting lateral and longitudinal flapping is represented by the following set of equations (derivation can be found in [6]):

$$\dot{b_1} = -p - \frac{b_1}{\tau_e} - \frac{1}{\tau_e}\frac{\partial b_1}{\partial \mu_v}\frac{v - v_w}{\Omega R} + \frac{B_{\delta_{lat}}}{\tau_e}\delta_{lat} \tag{2.13}$$

$$\dot{a_1} = -q - \frac{a_1}{\tau_e} + \frac{1}{\tau_e}\left(\frac{\partial a_1}{\partial \mu}\frac{u - u_w}{\Omega R} + \frac{\partial a_1}{\partial \mu_z}\frac{w - w_w}{\Omega R}\right) + \frac{A_{\delta_{lon}}}{\tau_e}\delta_{lon} \tag{2.14}$$

Resulting rolling and pitching moments, which enter the rigid body equations of motion 2.2, can be computed as follows:

$$L_{mr} = (K_\beta + Th_{mr})\, b_1 \tag{2.15}$$

$$M_{mr} = (K_\beta + Th_{mr})\, a_1 \tag{2.16}$$

and the resulting main rotor forces are:

$$\begin{aligned} X_{mr} &= -T_{mr} a_1 \\ Y_{mr} &= T_{mr} b_1 \\ Z_{mr} &= -T_{mr} \end{aligned} \tag{2.17}$$

Meaning of all variables can be found in table 2.1.

2.3.4 Engine and Rotor Speed Model

The *rotor speed dynamics* can be expressed as:

$$\dot{\Omega} = \dot{r} + \frac{1}{I_{rot}}\left[Q_e - Q_{mr} - n_{tr} Q_{tr}\right] \tag{2.18}$$

We may well assume that the engine torque Q_e is proportional to the throttle setting δ_t:

$$P_e = P_e^{max}\delta_t \tag{2.19}$$

and the engine torque is:

$$Q_e = \frac{P_e}{\Omega} \tag{2.20}$$

We may consider the engine response to the throttle as instantaneous, for the time lags associated with the engine inertia, air and fuel intake and combustion can be neglected, compared to the vehicle dynamics.

In RAMA, no feedback engine governor is used; instead, a pre-programmed collective-to-throttle feed is established, so the throttle is controlled in the open loop fashion. Due to the fast engine response and high rotor inertia this solution works fine and is capable to keep the rotor revs within the range of approx. ± 100 RPM.

2.3.5 Fuselage Forces

The *fuselage drag* counteracts the forward or side movement of the helicopter and also affects the rotor downwash in hover. Fuselage drag forces can be approximated as:

$$\begin{aligned} V_\infty &= \sqrt{u_a^2 + v_a^2 + (w_a + V_{imr})^2} \\ X_{fus} &= -0.5\rho S_x^{fus} u_a V_\infty \\ Y_{fus} &= -0.5\rho S_y^{fus} v_a V_\infty \\ Z_{fus} &= -0.5\rho S_z^{fus} (w_a + V_{imr}) V_\infty \end{aligned} \tag{2.21}$$

where V_{imr} is the induced main rotor speed. We assume that the fuselage Centre of Pressure coincides with its Centre of Gravity. The fuselage projection areas are approximately $S_y^{fus} \approx 2.2 S_x^{fus}$, $S_z^{fus} \approx 1.5 S_x^{fus}$ (considering the proportions of the helicopter). Fuselage moments, induced by the drag forces, are small compared to the main rotor moments and can be neglected.

2.3.6 Tail Surfaces Forces and Moments

2.3.6.1 Vertical Fin

Main purpose of the *vertical fin* is to protect the tail rotor from hitting the ground. However, it also influences the dynamics, as it produces some aerodynamic drag,

caused by the tail rotor wash. The sideforce, produced by the fin, could be expressed as:

$$Y_{vf} = -0.5\rho S_{vf}\left(C_{L_\alpha}^{vf} V_\infty^{tr} + |v_{vf}|\right) v_{vf} \tag{2.22}$$

where S_{vf} is the vertical fin area, $C_{L_\alpha}^{vf}$ is its lift curve slope, $V_\infty^{tr} = \sqrt{u_a^2 + w_{tr}^2}$ is the axial velocity at the location of the tail rotor hub, v_{vf} is the side velocity relative to the air and w_{tr} is the vertical velocity, modeled as:

$$v_{vf} = v_a - \epsilon_{vf}^{tr} V_{itr} - l_{tr} r \tag{2.23}$$

$$w_{tr} = w_a + l_{tr} q - K_\lambda V_{imr} \tag{2.24}$$

Where V_{itr} is the tail rotor induced velocity, r is the yaw rate, ϵ_{vf}^{tr} is the area of the part of the vertical fin exposed to the tail rotor wash, l_{tr} is the vertical distance between Centre of Gravity and the tail rotor hub, V_{imr} is the main rotor induced velocity and finally K_λ is the wake intensity factor. Because of the stall of the vertical fin, the maximal side force has to be bounded:

$$|Y_{vf}| = 0.5\rho S_{vf}\left(\left(V_\infty^{tr}\right)^2 + v_{vf}^2\right) \tag{2.25}$$

Resulting side force induces a yawing and small rolling moments:

$$\begin{aligned} N_{vf} &= -Y_{vf} l_{tr} \\ L_{vf} &= Y_{vf} h_{tr} \end{aligned} \tag{2.26}$$

2.3.6.2 Horizontal Fin

The purpose of the *horizontal fin* is to help to stabilize the helicopter in forward flight, but its effect is somewhat disputable. The fin produces a pitching moment due to its aerodynamic drag. The fin may be completely or partially submerged in the main rotor downwash. Exact location of the fin is determined empirically during the trim flights.

Effective vertical speed at the horizontal fin location is determined by:

$$w_{hf} = w_a + l_{hf} q - K_\lambda V_{imr} \tag{2.27}$$

Then the force produced by the fin is computed:

$$Z_{hf} = 0.5\rho S_{hf}\left(C_{L_\alpha}^{hf} |u_a| w_{hf} + |w_{hf}| w_{hf}\right) \tag{2.28}$$

To take the stall effect into account, the result has to be bounded:

$$|Z_{hf}| \leq 0.5\rho S_{hf}\left(u_a^2 + w_{hf}^2\right) \tag{2.29}$$

Finally, the pitching moment produced by the horizontal fin is:

$$M_{hf} = Z_{hf} l_{hf} \tag{2.30}$$

2.3.7 Tail Rotor Forces and Moments

It is no easy task to accurately simulate the *tail rotor* behavior, because there is a wide range of hardly predictable factors, which can have significant influence. The tail rotor often operates in its own wake, especially in hover, when the tail swing from one side to the other and back again. Moreover, the tail rotor often finds itself partially or completely in the main rotor wake, which affects its efficiency in a very complex way. Airspeed of the helicopter also has considerable and very complex influence, since it affects the tail rotor directly and indirectly, due to the changes in the main rotor downwash.

Hence, the momentum theory, which was used so successfully to compute the main rotor thrust, fails under utterly different conditions, reigning at the tail area of the helicopter. Due to all these complications, there is nothing but approximate estimations to rely on.

Approximate equations, describing the tail rotor thrust, are based on linearization of the thrust-inflow iteration equations around various axial velocities and the tail rotor pitch.

$$\begin{aligned}
C_{T_{\mu_z^{tr}}}^{tr} &= \frac{\partial C_T^{tr}}{\partial \mu_z^{tr}}\left(|\mu_{tr}|, \mu_z^{tr} = 0, \delta_r^{trim}\right) \\
C_{T_{\delta_r}}^{tr} &= \frac{\partial C_T^{tr}}{\partial \delta_r}\left(|\mu_{tr}|, \mu_z^{tr} = 0, \delta_r^{trim}\right) \\
Y_v^{tr} &= -C_{T_{\mu_z^{tr}}}^{tr}\frac{f_t \rho \Omega_{tr} R_{tr} \pi R_{tr}^2}{m} \\
Y_{\delta_r}^{tr} &= -C_{T_{\delta_r}}^{tr}\frac{f_t \rho \left(\Omega_{tr} R_{tr}\right)^2 \pi R_{tr}^2}{m} \\
Y_{tr} &= m Y_{\delta_r}^{tr}\delta_r + m Y_v^{tr}\mu_z^{tr}\Omega_{tr} R_{tr}
\end{aligned} \tag{2.31}$$

where f_t is the fin blockage factor:

$$f_t = 1 - \frac{3}{4}\frac{S_{vf}}{\pi R_{tr}^2} \tag{2.32}$$

The tail rotor speed is proportionally dependent on the main rotor speed, since the tail rotor is driven via a driving pulley with constant gear ratios. Therefore

$$\Omega_{tr} = n_{tr}\Omega_{mr} \tag{2.33}$$

where n_{tr} is the gear ratio coefficient. To emulate the main rotor influence, we have to compute the so-called wake intensity factor K_λ first. This factor models the apparent in-plane velocity change, seen by the tail rotor. Following variables are needed to compute K_λ:

$$\begin{aligned} g_i &= \frac{l_{tr} - R_{mr} - R_{tr}}{h_{tr}} \\ g_f &= \frac{l_{tr} - R_{mr} + R_{tr}}{h_{tr}} \end{aligned} \tag{2.34}$$

In hover or low enough air speed, the tail rotor finds itself out of the main rotor wake. This can be represented by the condition $V_{imr} \leq w_a$:

$$\frac{u_a}{V_{imr} - w_a} \leq g_i \tag{2.35}$$

and $K_\lambda = 0$. The tail rotor is fully in the wake if:

$$\frac{u_a}{V_{imr} - w_a} \geq g_f \tag{2.36}$$

and $K_\lambda = 1.5$. In the remaining cases, linear growth of the wake intensity factor with air speed is assumed:

$$K_\lambda = 1.5\frac{\frac{u_a}{V_{imr} - w_a} - g_i}{g_f - g_i} \tag{2.37}$$

K_λ, computed using the expressions above, is used to compute the vertical component of the airspeed in the tail location, according the equation 2.24. Next, the advance ratio of the tail rotor must be determined:

$$\mu_{tr} = \frac{u_a^2 + w_{tr}^2}{\Omega_{tr} R_{tr}} \tag{2.38}$$

Due to the stall condition, resulting tail rotor thrust has to be limited:

$$\begin{aligned} Y_{tr}^{max} &= f_t C_{T_{max}}^{tr} \rho \left(\Omega_{tr} R_{tr}\right)^2 \pi R_{tr}^2 \\ |Y_{tr}| &\leq Y_{tr}^{max} \end{aligned} \tag{2.39}$$

And the yawing and rolling moments are enumerated as follows:

$$\begin{aligned} N_{tr} &= -Y_{tr} l_{tr} \\ L_{tr} &= Y_{tr} h_{tr} \end{aligned} \tag{2.40}$$

To determine the tail induced velocity, used in equation 2.23, the inflow ratio has to be computed:

$$\lambda_0^{tr} = \mu_{z_{tr}} - 2 \left[\frac{2C_T^{tr}}{a_{tr}\sigma_{tr}} - \delta_{tr} \left(\frac{1}{3} + \frac{\mu_{tr}^2}{2} \right) \right] \tag{2.41}$$

where C_T^{tr} is the tail rotor thrust coefficient, a_{tr} is the tail rotor blade lift curve slope and σ_{tr} is the tail solidity ratio:

$$\sigma_{tr} = \frac{2c_{tr}}{\pi R_{tr}} \tag{2.42}$$

Finally, the tail rotor torque Q_{tr} can be computed using equations 2.8 and 2.9, with the tail rotor parameters in place of the main rotor parameters.

Chapter 3

Control System Hardware Architecture

3.1 Requirements

The RAMA system is intended as a cheap (less than 3000 Euro components price), reasonably lightweight (approx. 1.5 kg) and compact (330x160x65 mm) universal control system, which could be fitted to any kind of UAV (either a rotorcraft or a fixed-wing aircraft). As an academic project, it is designed for research applications and it is not meant as an "out-of-the-shelf product for everybody". This implies many of its assets and limitations.

It was foreseen that RAMA will have to be easily modifiable, both hardware and software wise, and should be able to accommodate relatively sophisticated algorithms and broad telemetry bandwidth. Therefore, it is designed as a modular system with high degree of independence between the nodes, so that each of them could be modified without affecting the rest of the system very much. This also contributes to its safety (see chapter 6 for details), but on the other hand makes it more complicated and increases it's power consumption, size and mass.

RAMA's on-board computers are very much over-designed in the sense of computing power, for the reasons mentioned above. This allows for previously unforeseeable extensions of its control algorithms, but increases the power demand. This is not regarded as a problem for an experimental system with limited mission duration (it is still able to run up to 4 hours on internal power).

The wireless telemetry link is very simple and offers up to 54 Mbps bit rate, but is not reliable enough for critical tasks and works only on a short range and within direct line of sight. Hence, it is no problem to transmit very large data streams and to extend the telemetry as needed. The disadvantages are not limiting for an experimental system. Naturally, the telemetry could only be used for non-critical data transfers.

RAMA could possibly operate fully autonomously, however, it lacks the necessary

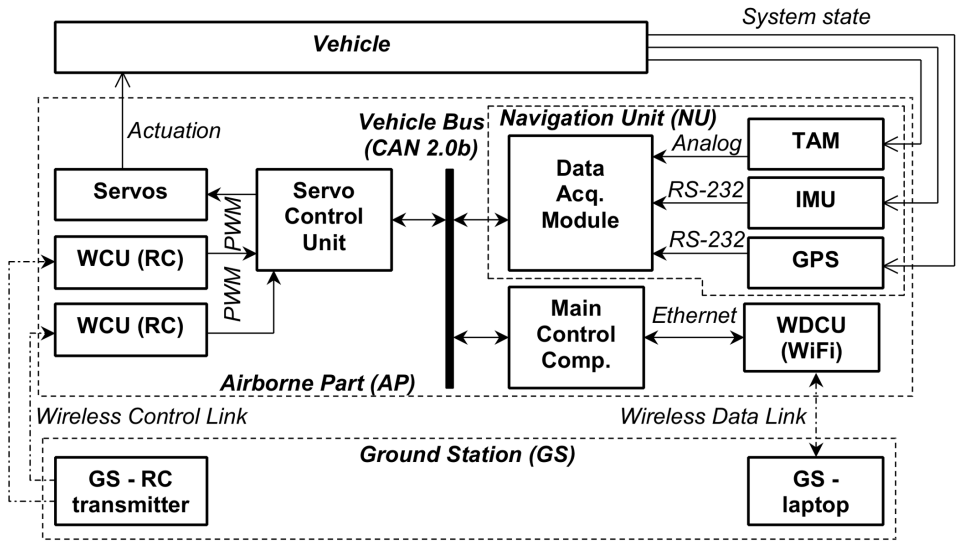


Figure 3.1: Control system block diagram

redundancy and some mandatory avionic equipment (such as a radar transponder and TCAS), required by UAV regulations for autonomous UAVs in most countries. It is not suitable nor intended for autonomous long-range missions. This is due to the strict budget, weight, size and power consumption design constraints. Nevertheless, it perfectly fulfills all the requirements for a semi-autonomous experimental small-UAV autopilot, as mentioned in the section 1 - it can be safely operated under the direct supervision of a human pilot.

Once fully functional, RAMA could provide a fully stabilized flying platform for any kind of payload. This has many advantages over the wholly manually controlled vehicle, as it was proven many times that a properly designed autopilot is superior to a trained human pilot when it comes to fine stabilization or precise trajectory tracking. An autopilot also significantly reduces the workload of the human pilot and reduces the possibility of human error [35][36]. Due to its run-time reconfiguration capability (see chapter 6), it is safe to operate the RAMA within the line of the pilot's sight, so that he/she can take over manual control in case of the system failure.

3.2 Overview

RAMA consists of the *Airborne Part* (AP) and the *Ground Station* (GS), comprised of a laptop computer and a RC (Radio Control) transmitter. The laptop is used only to visualize and record the on-line telemetry of the vehicle, while a model RC set is used to issue commands to the airborne control system, and also allows one to re-take full manual control of the drone in case of a control system failure. The airborne

part of the system is designed as a networked hierarchical distributed system. It consists of several basically independent functional blocks, interconnected via the *Vehicle Bus* (VB), as can be seen from the block diagram (Figure 3.1). Those blocks are the *Navigation Unit* (NU) (comprised of sensors and their management), *Main Control Computer* (MCC), *Wireless Data Communication Unit* (WDCU), two *Wireless Control Units* (WCU) (for redundancy) and the *Servo Control Unit* (SCU).

This structure allows a separate design and testing of each component, also simplifying any possible future extensions of the system. The nodes may be added or exchanged without affecting the rest of the system very much, as all nodes are interconnected only via a single interface and a single medium, the Vehicle Bus. Of course, integration work and testing must follow any modification, but it is simplified by a great deal due to the modular structure of the system.

The core member of the RAMA system is the Main Control Computer (MCC), where the control and communication algorithms run. The Main Control Computer is connected to the Ground Station via the Wireless Data Communication Unit (WDCU), which is currently an *IEEE 802.11g* (WiFi) module. The WiFi communication is non-critical, as it is used only for the telemetry. The commands for the autopilot are issued via the Wireless Control Units (WCU) (currently a model RC set), which is much more reliable and has a lot wider range than WiFi. RAMA is able to operate without the data communication with the Ground Station (GS), as it serves only for the on-line system monitoring and telemetry recording.

The sensor data are acquired and pre-processed by the *Navigation Unit* (NU). This unit includes the *Inertial Measurement Unit* (IMU), *Three-Axis Magnetometer* (TAM), *Global Positioning System* (GPS) receiver and the *Data Acquisition Module* (DAM). All sensors are connected to the DAM, the purpose of which is to synchronously sample all the data, pre-process it (filtering, unit conversion, etc.) and send it to other control system nodes. DAM also provides the *Time Synchronization Message* (TSM) for the rest of the system, the purpose of which is to synchronize all system nodes.

The actuators (the servomotors driving the control surfaces of the vehicle) are controlled by the *Servo Control Unit* (SCU). All actuators and both Wireless Control Units are connected to this node. Its purpose is to control the actuators, sample their positions and also to sample the control sticks' movement on the RC transmitter (provided by the signals from the Wireless Control Units).

The *Controller Area Network (CAN) 2.0b* [37], running at 1 Mbps, is used as the Vehicle Bus. Its utilization is approx. 10%, so there are no problems with extensive data packet delays. Proper real-time behavior is also ensured by the packet priorities and non-destructive arbitration. The overall performance of the bus and mean/extreme data latencies were evaluated statistically.

The Wireless Data Communication Unit (WDCU) is a standard embedded WiFi device, configured as an access point. The Ground Station laptop is the client - multiple Ground Stations are allowed, the RAMA system has an telemetry server and is

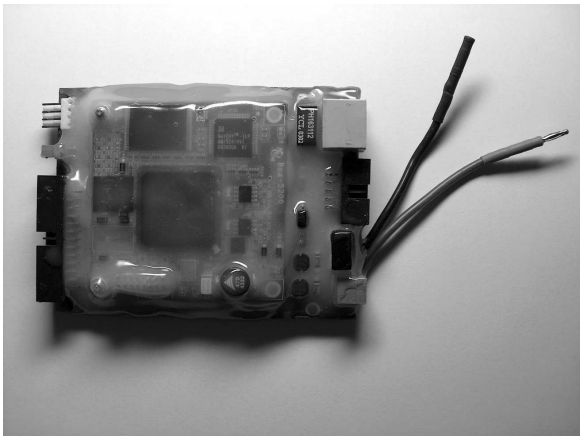


Figure 3.2: Main Control Computer - Boa 5200

able to broadcast data to multiple clients simultaneously. The communication data rate is 54 Mbps, which is more than enough for the telemetry, as the data packets are only some 500 bytes long and are sent 32 times per second. This results in approximately 2% of bus utilization, so there are no concerns with the real-time issues; the telemetry is a soft real-time task anyway. The communication is encrypted, using the standard 128-bit WEP algorithm.

Two separate batteries are used to power the system during flight. The first battery, called the *Avionics Battery* (AVB), serves as a power supply for on-board avionics. It is of Lithium-Polymer chemistry, capacity 4 Ah, configuration 3 cells serial (nominal voltage is therefore 11.1 V). The other battery is called the *Actuator Battery* (ACB) and serves as a power supply for the actuators. It is of Nickel-Cadmium chemistry, capacity 2.4 Ah, 4 cells serial (4.8 V nominal voltage). These two batteries form the *Internal Power Supply* (IPS) of the RAMA system. On the ground, RAMA can be supplied by a stabilized power source. Both batteries can be charged in-system using a service connector, without the need to disconnect and remove them.

The use of separate power sources for the actuators and the rest of the system is necessary because of electromagnetic interference, mainly affecting the sensors. This solution also allows one to incorporate a power supply redundancy for emergency scenarios in the future (see section 6.2.5 for details). Obvious setback of this solution is the additional mass it represents.

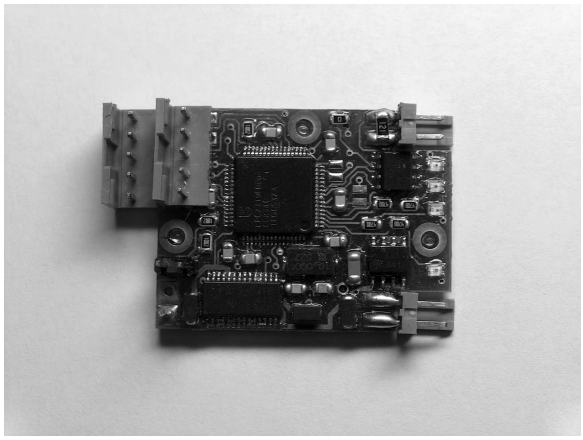


Figure 3.3: Data Acquisition Module

3.3 Components

3.3.1 Main Control Computer

BOA 5200 embedded computing platform [38] (Figure 3.2), from the Analogue & Micro company, is serving as the Main Control Computer (MCC). It utilizes the Linux Operating System with 2.6.xx core (it does not make much sense to mention exact kernel revision, because it is being updated as the time goes).

BOA 5200 is based on the MPC 5200B microprocessor, running at 300MHz. It is a decent computing platform, featuring 64 MB of SDRAM, up to 32 MB of internal NAND FLASH memory and a lot of peripherals (2x CAN controller, 10/100 Ethernet, USB, RS-232 and others). With a small PCB footprint (62x62 mm) and relatively low-power consumption (approx. 3 W), it forms an ideal computing platform for the RAMA system, able to execute complex control algorithms in real-time. A custom-made motherboard, hosting a stabilized power supply, bus drivers and connectors, was developed by our contractor Mr. Miloslav Žizka to support the BOA 5200 module.

3.3.2 Navigation Unit

The Navigation Unit (NU) consists of four parts. The Data Acquisition Module (DAM) is a simple embedded microcomputer, based on the Philips LPC2119 ARM7TDMI-core MCU (Figure 3.3). This item is a heritage hardware from Marek Peca's project on the walking robot Spejbl [39] [40] [41].

Main assets of the SPEJBL-ARM board are the simplicity, low power consumption (approx. 10mA@12V), compact size (37x28x6,5mm) and negligible weight. Aside from the MCU, there are only the CAN-driver and power regulator installed (and some LEDs). The MCU is driven at 60 MHz. The LE50CD power regulator

Figure 3.4: GPS receiver, mounted on the tail of the UAV

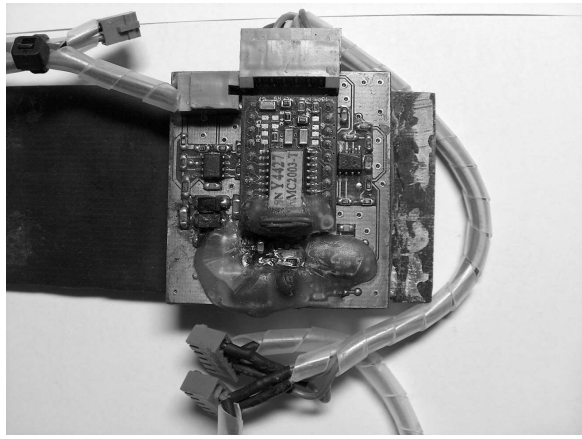


Figure 3.5: Three Axis Magnetometer with its motherboard

provides 5 V for the Philips PCA82C250T CAN driver, and the 3.3 V and 1.8 V voltages for the MCU are provided by the TPS73HD318 multi-purpose circuit. It also serves as a voltage watchdog for the RESET signal. Aside from that, only the obvious blocking capacitors and pull-up resistors are present. The IO pins of the MCU operate at 3.3 V levels, but are 5 V tolerant. The JTAG interface is not used; the software can be programmed either to the internal RAM or FLASH using the ISP (In-System Programming) via the UART line. In the RAMA system, both UARTs, A/D converter and CAN interface are being used, and four GPIOs to lit up some LEDs. No other interfaces of the MCU are employed.

The SPEJBL-ARM circuitry scheme can be found in appendix A.4.

Three sensing units are connected to the Data Acquisition Module:

Figure 3.6: Inertial Measurement Unit

MICRO-ISU BP-3010 [42] (Figure 3.6) serves as the Inertial Measurement Unit (IMU). It is a tiny device, measuring just 35x22x12 mm and weighting 30 g, with 0.5 W power consumption, providing accelerations and angular rates in all 3 vehicle axes. The measurement ranges are $\pm 10\,g$ and $\pm 300^\circ/s$ at a 64 Hz sampling rate. The data are provided via a serial link in the TTL levels, so the IMU can be connected directly to the DAM. The IMU is soldered to a simple PCB containing the 78M05 power regulator and a power filter (a set of capacitors with a self-induction coil). Besides, only some pull-ups, a power blocking capacitor and RESET protective circuitry (two anti-parallel diodes to dissipate voltage pulses and a small filtration capacitor) are employed. Circuitry scheme can be found in the appendix A.4. This device proved to be barely sufficient; the 64 Hz sampling rate is very much "at the edge", especially for the yaw control. Relatively low sampling rate caused many headaches with low-level stabilizers (see section 4.4) and moreover, the device also turned out to be very much vibration-sensitive, exhibiting some rather odd behavior under flight conditions (see section 3.5.1 for details). It will be replaced by Analog Devices' ADIS 16350 [43] in the future, which promises better results.

Garmin GPS 18-LVC receiver [44] (Figure 3.4) is an OEM GPS module with an integrated antenna and WAAS (Wide Area Augmentation System) capability. The data are sampled at a 1 Hz rate. This module is not ideal for this application, as it proved to be rather sensitive to the rotor blades, and consequently had to be positioned far at the tail boom, causing mechanical problems with the UAV. It should be also replaced by some better device in the future.

Honeywell HMC 2003 Three-Axis Magnetometer [45] (Figure 3.5) is measuring the magnetic field intensity components in each vehicle axis. The sampling rate is 512 Hz. A motherboard had to be designed for this sensor, accommodating the power filtration and the degaussing and amplifying circuitry (see appendix A.4). The de-

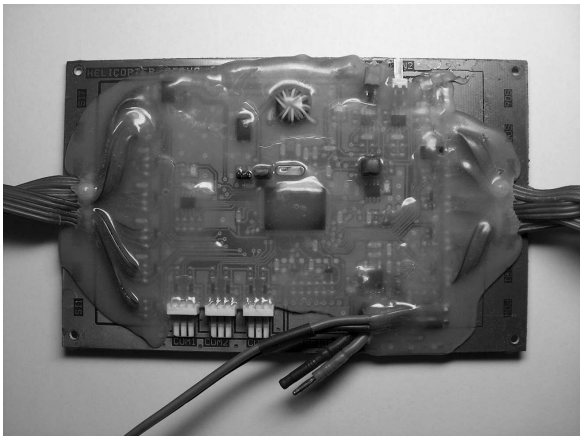


Figure 3.7: Servo Control Unit

gaussing circuitry is designed according manufacturer's reference solution, published in the datasheet. Delon voltage multiplier is used to obtain required 20 V from the 12.6-10 V, provided by the Avionics Battery (AVB). Linear operational amplifiers are used to provide additive offset and amplification of the output signals ($y = k.x + q$). First-order low-pass filters are accommodated as a simple anti-aliasing measure to suppress the high-frequency noise.

3.3.3 Servo Control Unit

The Servo Control Unit (SCU) (Figure 3.7) serves to control the actuators (i.e. servomotors) and, conveniently enough, it also measures both battery voltages. It is a simple, purpose-developed embedded computer, built around the Renesas H8S/2638F MCU. The SCU was designed by Ota Herm, a former graduate student, as his master project [46] (under the supervision of the author of this thesis). It is the most crucial element of the RAMA system, as its failure would result in one or more actuators being out of control, with obvious grave consequences.

The SCU is connected to the Wireless Control Unit (WCU), the servomotors, and the Vehicle Bus (VB). Also, both internal power sources (Avionics Battery - AVB, as well as Actuator Battery - ACB) are connected to the SCU. Basically, the SCU can operate in two modes; either in the Automatic or in the Manual Control Mode. In the *Automatic Control Mode* (ACM), the signals from the WCU are read and sent to the MCC (Main Control Computer), and the signals for the servomotors are generated by the SCU, according the commands from the MCC. In the *Manual Control Mode* (MCM), the signals from the RC receiver are read and sent to the MCC, and also sent directly to the actuators. Therefore, should the rest of the RAMA system fail and only the SCU would prevail, it is still possible to maintain manual control (see chapter 6 for details).

The mode switching is controlled by one of the input RC channels (channel 7), connected to the WCU. Therefore, it is completely independent on the rest of the RAMA. Hence, it is always possible to switch back to the MCM, even if the rest of the system would be dead. The mode switching cannot be affected by the MCC, or any other node, in any way. When the mode switch occurs, the MCC is informed by a message sent by the SCU.

The SCU was designed as simple as possible. It has 7 input channels, compatible with standard RC signals, and 6 servo outputs. The input channel no. 7 is reserved for mode switching, while the other channels may be associated arbitrarily. In our case, the channel 1 is used for the carburetor servo, channel 2 for the roll cyclic servo, channel 3 for the pitch cyclic servo, channel 4 for the tail rotor servo and channel 5 for the collective servo. Channel 6 is not being used. Except that, the SCU has 3 UART channels (channel 1 serves as a diagnostic terminal, channel 2 is used for the firmware downloading, and channel 3 is unused), 2 CAN channels (channel 1 is connected to the Vehicle Bus, channel 2 is unused), an external watchdog (voltage & time), (optional) EEPROM memory, and a power regulator & filter.

The actuators are operated and their respective positions are sampled at 50 Hz, which proved barely enough, especially for the yaw control (see section 4.4).

Let us now briefly describe the SCU circuitry. The H8S/2638F MCU is running at 20 MHz (10 MHz crystal is used, but an internal PLL multiplies that by a factor of 2). The RESET signal is controlled by the MAX1232ESA circuit, which serves as a combined time & voltage watchdog. Philips PCA82C250T CAN drivers are used to provide the CAN bus connection. Because the MCU is not equipped with internal EEPROM, an external 93C46 serial EEPROM is (optionally) connected to it. Currently, it is not being used by the software, and therefore may be omitted. All I/O pins of the SCU are protected against high-voltage spikes by a pair of diodes.

The 78M05 power regulator is used to provide the 5 V supply. The AVB input is filtered by a set of capacitors and a self-induction coil. The ACB is also filtered in a similar way, but a self-induction bifilar toroid coil is being used, despite its greater dimensions and weight, instead of a simple SMD self-induction coil. The AVB and ACB grounds are connected by the R43 (0Ω) resistor, which serves as a bridge. A self-induction coil might be used instead; in our case, it did not help very much to dissipate the noise induced into the power lines. Also, small (approx. 10Ω) resistor might be tried out. Both AVB and ACB positive lines are connected (via resistance voltage dividers) to the A/D converter of the MCU (to allow for battery voltage measurements).

The SCU circuitry scheme can be found in the appendix A.4.

3.3.4 Wireless Data Communication Unit

Wireless Data Communication Unit (WDCU) (Figure 3.8) provides the data connection between the Main Control Computer (MCC) and the Ground Station (GS). To

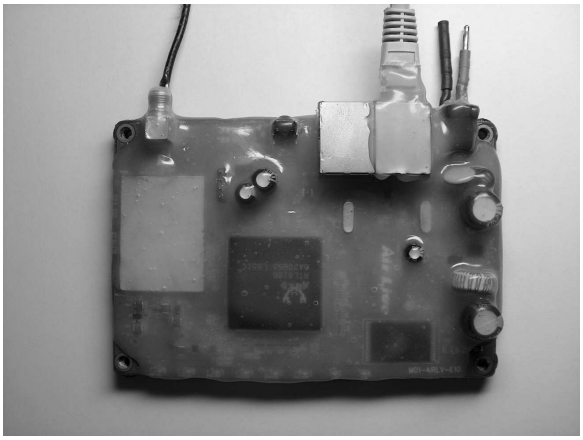


Figure 3.8: Wireless Data Communication Unit

reduce development time, off-the-shelf IEEE 802.11g (WiFi) device is used for this purpose. This solution proved to be feasible, as the WiFi link works very well within the ranges we use to fly, especially because there is a clear line of sight between the stations.

AirLive WL-5470AP Access Point is used as the WDCU, but any other IEEE 802.11g device could be used instead. It was only slightly adapted according our needs; Original housing was dismantled, the PCB was mechanically reinforced using protective lacquer coat and a polyethylene glue (see the section 3.4 for more details), and the antenna and the power supply connectors were changed. The reverse-SMA antenna connector is now attached to a 300mm long coaxial cable, to allow for external antenna mounting between the tail boom support rods. The power supply connector was replaced by RAMA's standard power connectors.

3.3.5 Wireless Control Unit

The Wireless Control Unit (WCU) is a standard model RC equipment. In the Manual Control Mode (MCM), it allows the human pilot to control the UAV in the same manner as an ordinary RC model; when Automatic Control Mode (ACM) is engaged, the WCU is used to transmit commands to the Airborne Part (AP) of the RAMA control system. In the current state of development, the positions of the control sticks of the RC transmitter set reference for the rate controllers in the ACM.

Spektrum DX-7 RC set is currently used, working in the 2.4 GHz range. A 35 MHz system was used earlier, but was abandoned and upgraded in order to increase reliability. The DX-7 system is redundant (two separate receivers are used, working on different frequency channels and set to orthogonal antenna polarization) and utilizes a digital encoding of the transmitted signal, secured by CRC (Cyclic Redundancy Check). It is much less prone to undesired electromagnetic interferences

Figure 3.9: Wireless Control Unit

than the previous “traditional” RC system was.

3.4 Mechanical Outfit and Housing

This section describes the mechanical construction of the Printed Circuit Boards (PCBs), wiring harnesses, housing and mating of the RAMA system to the carrier vehicle. The most important consideration, which has to be taken into account when designing the UAV avionics (from a mechanical point of view), are airframe vibrations. Another aspects include operating temperature ranges, moisture, possible FOD (Foreign Object Debris) hazards and crash resistance. The last one is especially important quality for experimental systems.

Mechanical vibrations could seriously compromise reliability of improperly designed or installed hardware. The weakest points, most prone to mechanical wear, are usually connectors and wire harnesses. The connectors must be small and light, yet robust enough to withstand the strain experienced under the flight conditions. It is important to use connectors with multiple points of contact and wide contact area, which are the most contributing factors to the connector’s longevity and reliability. It is necessary to safe the connectors against accidental disconnection; in our case, it is done by application of a polyethylene glue.

It is better to use connectors with crimped pins if possible; if the leads must be soldered to the connector pins, it is important to not allow any bends in the vicinity of the soldered joint, as the wire tends to loose its elasticity due to the soldering, and might develop a crack over the time. Any exposed joints must be treated properly, usually with a heat shrinking tube.

Wire harnesses must be fastened properly but not too tightly, in order to prevent excessive strain on the wires. It is important to use sleeves (as can be seen for example

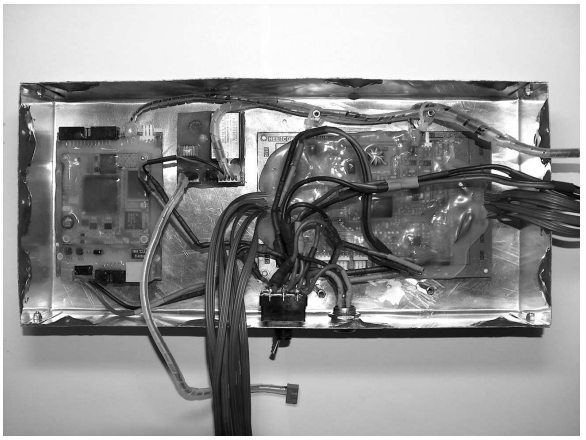


Figure 3.10: Electronic Container low-level, showing the Main Control Computer (left), the Decoupling Unit (center) and the Servo Control Unit (right). Note also the power switches and the service connector below.

in Figure 3.9), especially at attachment points or points of contact between the harness and the vehicle body. This prevents excessive wear of the wire insulation.

In the RAMA system, the backbone power wiring is made of special silicone-coated heavy duty cables and the power switches are redundant (two parallel switches are used). This renders a total power failure extremely improbable.

The PCBs must be mechanically robust, resistant to humidity and FODs, and also fulfill desired operating temperature range. It is wise to use at least industry-grade parts (temperature ranges from -40 to $+85°C$) to satisfy the last requirement. Generally speaking, semiconductors are more prone to higher temperatures and better tolerate the low ones. Resistors are usually sturdy and do not represent a problem, whereas capacitors (especially electrolytes) do not tolerate high temperatures for a prolonged time and too low temperatures either. It is wise not to use electrolytes at all if possible. Meaning of the "too high" and "too low" depends on the part's grade, but generally speaking (for common main-stream capacitors), "too low" can be interpreted as below $-10°C$ and "too high" as above $+70°C$.

In RAMA, all PCBs were protected with a lacquer coat initially. This coat serves as a mechanical and moisture protection and also protects the PCB from conductive FODs floating around, which could possibly cause a short circuit. Heavy parts (such as heat sinks or large coils) must be either screwed down or glued, using a polyethylene glue. It is important to secure bolted joints either by some appropriate sealant, or to use self-sealing screws, to prevent them from becoming loose due to the vibrations. The PCBs are mounted to its housing either using rubber shock-absorbers or secured using a thick double-adhesive foam tape, which also serves as a damper.

The PCB protection described above proved to work well enough under normal

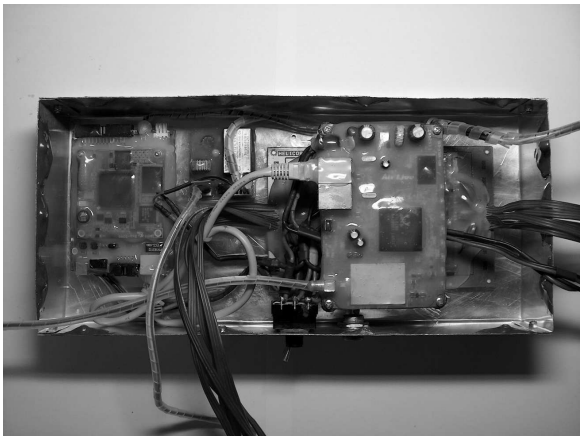


Figure 3.11: Here the Wireless Data Communication Unit (WDCU) is installed above the Servo Control Unit (SCU). The WDCU is deemed expendable and protects the SCU from the impact coming from above (which is the most likely cause in a crash - if the landing gear legs break upon impact, the helicopter body tends to crush the Electronic Container from above).

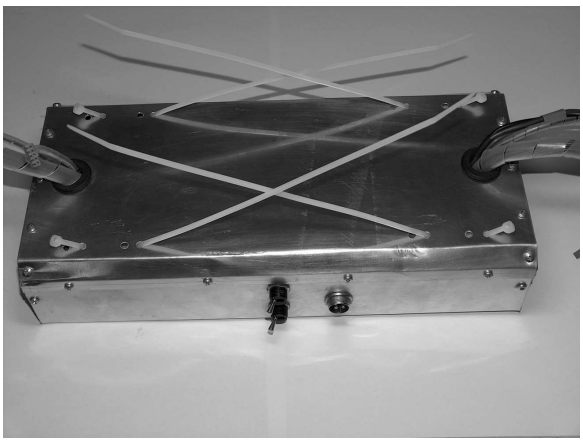


Figure 3.12: Complete Electronic Container, ready to be mated with the vehicle

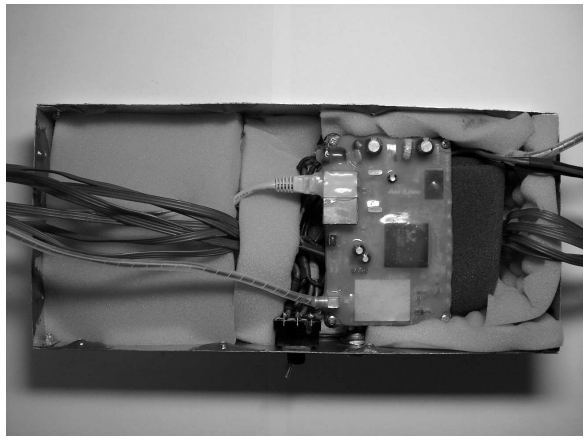


Figure 3.13: The low layer of the Electronic Container filled with foam rubber

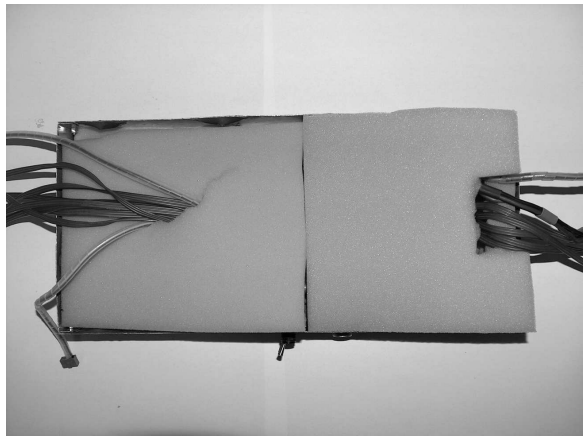


Figure 3.14: Upper part of the Electronic Container filled with foam rubber, ready to the cover mounting

flight conditions, but was rather insufficient in case of a crash. Since several electronic boards were lost due to the accidents, the crash protection was gradually improved and the final version evolved into the PCBs being completely coated with a thick polyethylene coating and the housing filled with multiple layers of foam rubber (Figures 3.13 and 3.14). The least expensive parts (such as the WDCU) are deemed “expendable” and form a sort of a “crash zone,” in order to improve protection of the more expensive parts. This “ultimate” solution proved to be particularly sturdy, and no piece of electronics was lost in a crash ever since its introduction. See Figures 3.10 and 3.11, showing the Electronic Container internals.

The housing and mating to the vehicle depends heavily on specific UAV type.

In our case, the bulk of the RAMA system is located in an aluminum case, called the *Electronic Container* (EC) (Figure 3.12). This container is located between the landing skids of the carrier helicopter and mated to the vehicle using cable ties, as can be seen in Figure 3.12. Rubber dampers are used between the EC and the airframe, in order to suppress vibration transfer and to de-tune possible resonance.

The GPS receiver is located far at the tailboom in order to clear the main rotor blades, which would otherwise obscure its signal reception. This is not ideal from the mechanical point of view, because the mass of the GPS receiver, located at the end of the tailboom, forms a mechanical resonator and could easily work as a resonance amplifier. It is important to mount the GPS receiver using some dampers and also to properly tune the tail boom support rods to mitigate this problem.

Both batteries are located at the very front of the helicopter, to compensate for the Centre of Gravity (CG) shift, caused by the mass of the GPS receiver so far back. The CG position is of vital importance (see section 2.2.5) and must be maintained. The sensors, both IMU and TAM, are located as close to the CG as practically possible, because any offset of the sensors from the CG position would cause discrepancy in the navigation algorithms and would have to be compensated. If the offset is small enough, it can be neglected in the navigation algorithms.

Both IMU and TAM are fastened using a double-adhesive foam tape, serving as a vibration damper. The shape and thickness of the tape is not arbitrary, but has to be determined experimentally, in order to provide the best possible damping. Improperly designed mounting of those parts could lead to a serious measurement deterioration. The TAM has to be statically compensated to account for magnetic parts of the airframe, affecting its measurements.

3.5 Problems and Solutions

Couple of long-term hardware-related problems, worth mentioning, were encountered among many others during the hardware design and development process. The first of them was caused by mechanical vibrations affecting the IMU measurements, the second was related to Electromagnetic Compatibility (EMC), and the third plagued the Three-Axis Magnetometer (TAM).

3.5.1 Inertial Measurement Issues

This problem was related to mechanical vibrations of the UAV airframe, strongly affecting the IMU measurements and leading to a rather odd IMU behavior. Let us at first briefly describe the origin and character of these vibrations. The level and frequency spectrum of airframe vibrations are dependent on many things; mainly on the UAV type (fixed-wing aircraft or rotorcraft) and the propulsion system (combustion or electric).

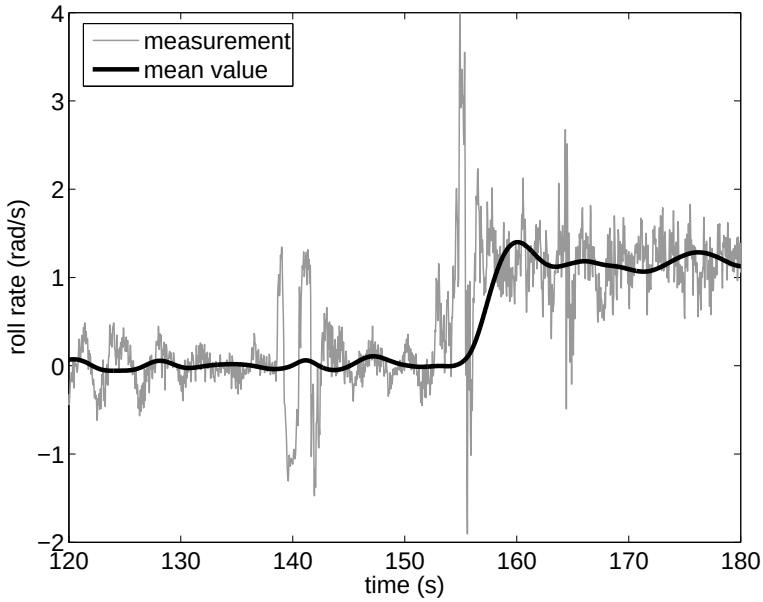


Figure 3.15: Roll rate gyro failure - 156 s into the flight, the mean value of the signal falsely rises to $\approx 1\,rad/s$

The rotorcraft tend to vibrate more and in much broader frequency spectrum than fixed-wing aircraft. They are also much more prone to catching resonance. The character of the vibrations is mainly related to the construction of the rotor head, tail rotor drive, tail boom construction and payload assembly. There are many rotating parts and any disbalance, however slight, could cause a lot of problems. It is therefore very important to precisely balance the main and tail rotor blades (both their weight and the Center of Gravity (CG) location) and the flybar paddles (see section 2.2 for details). This is not sufficient though - the rest of the airframe had to be mechanically de-tuned to avoid any unwanted resonance around the normal working point. This working point (and also the frequency spectrum of the vibrations) is determined by the main rotor speed, which is directly proportional to the engine rpm. This speed is kept constant during the flight, as was explained in section 2.2.

The highest frequency and also the most “hard” vibrations are induced by a combustion engine. The frequency of these vibrations can be higher than the Nyquist frequency of the sampling and have the most adverse effects, especially on the inertial sensors (gyroscopes and accelerometers). Their measurements must be filtered mechanically or electrically (still in the analog part of the sensor), to avoid their leakage into the lower frequency parts of the spectrum, due to the aliasing. These problems can be mostly avoided when using electrical propulsion.

In our case, the main rotor speed is around 1800 Revolutions Per Minute (RPM),

which is approx. 30 Hz. This is uncomfortably close to the Nyquist frequency of the sampling (the sampling rate is 64 Hz) and therefore forms an aliasing hazard. Moreover, the engine revs 18000 RPM, which is 300 Hz and therefore much higher than half of the sampling rate too. The datasheet of the BP3010 IMU device does not explicitly state whether the device has any internal anti-aliasing filters, but even in the case it does, they do not seem to work very well. The leakage of the high-frequency noise into the lower-frequency portions of the spectrum was experienced, but not only that. At particular vibration amplitudes, the device experienced a sudden and very significant change in the sensor offsets, meaning that - for example - a gyroscope reading would suddenly go off by one radian per second, or an accelerometer reading by one g (as shown in Figure 3.15). The affected sensor would still be operational, but its measurement would be significantly shifted. This problem affected the accelerometers and gyroscopes alike, and it was discovered that this is an amplitude-related, rather than frequency-related, issue.

This was a very serious problem, affecting mainly the roll gyroscope under the flight conditions, and lead to several in-flight emergencies when a roll gyroscope measurement suddenly shifted and caused the control system to compensate, falsely believing that the vehicle was rolling - and putting it itself into an undesired roll. To make the things even worse, this error condition was undetectable by the control system, because the offset would not change suddenly in one sample, but would rather grow over several samples. Hence, it could not be distinguished from a proper measurement in any way, as it was well within the measurement limits, and did not violate the gradient rule either. This condition is extremely unpleasant for the human pilot - the helicopter would happily fly around, and all of sudden it would start rolling violently, without any previous warning. The only way out was to disengage the ACM (Automatic Control Mode) immediately, enter the MCM (Manual Control Mode), and recover the vehicle flying manually (if there was enough time for that).

This condition was reproduced and thoroughly investigated at a vibration table and was extensively consulted with the BP3010 manufacturer. It was determined that an internal analog integrator, used in the device for analog averaging of the measurements between the sampling points, was most likely to blame. It was also found why the roll measurement was affected in flight; it was because the IMU housing is not symmetrical and have the smallest moment of inertia in the roll, making it prone to vibrate more in this particular direction. However, there was no way how to solve this issue “internally,” within the device itself. Because this was unacceptable, as well as the aliasing problem described above, a solution had to be found for both. The only way would be to mechanically isolate the device from excessive vibrations, both amplitude and frequency wise. A mass damper was developed as a solution to both problems. The IMU is mounted to a 200 g lead plate (the seismic mass), which significantly increases (and equalizes) its inertia in all directions, and this plate is mounted to the airframe using a double-adhesive foam tape, serving as the flexible part of the damper. It involved a lot of experiments to fine-tune the damper to work properly, but

a feasible solution was ultimately found.

3.5.2 EMC Issues

The second most notable hardware development problem was indirectly related to the GPS receiver, the root cause being the EMC (Electromagnetic Compatibility). In its early incarnation, RAMA had no GPS and the whole control system was contained inside the EC (Electronic Container). The then-used WCU (Wireless Control Unit), i. e. a 35 MHz RC receiver, was located outside and far from the EC, in the nose of the helicopter. As later experience showed, this was a very fortunate situation, for the EC shielded the lurking EMC horrors within. Once the GPS receiver was added, it was discovered that it has to be mounted on the very tip of the tailboom, in order to clear the main rotor blades, which were causing signal blurring. Naturally, a signal and power cable had to be routed to the receiver, and that (relatively long) cable was, as was discovered later, serving as a perfect antenna for the electromagnetic noise, emitted by the on-board computers through the power lines. The genie was released out of the bottle (or the EC, in our case) and caused a mayhem in the WCU, obstructing the 35 MHz communication badly. This was fortunately discovered in the course of the Flight Readiness Test (see section 7.2.1) prior to a flight test, but no simple solution was found immediately.

Somewhat semi-permanent cure was devised through a complete galvanic decoupling of the GPS receiver. This was not as easy as it might seem, for the receiver has to be supplied from the same power source as the rest of the system (a separate battery, serving solely for the GPS receiver, was rejected as impractical). After some research and a lot of experiments, the *Decoupling Unit* (DU) was designed. Its purpose is to completely decouple the GPS signal lines as well as the power lines.

The GPS UART data line is decoupled optically, using the PC817 photocoupler. To improve the form of the output signal coming from the photocoupler, the 74LS04 TTL negator is being used.

The power lines are decoupled using the FDD03-12S1 DC-DC converter. The output power lines are filtered, using a bifilar self-induction coil (purpose-made, 17 coils on the Amidon FT37-43 ferrite toroid), a capacitor set and a self-induction coil again. The circuitry scheme of the DU is available in the appendix A.4.

This solution helped, but proved to be only semi-permanent, just until the point where other system parts began to migrate from the overcrowded EC to other parts of the airframe. The DAM (Data Acquisition Module), along with the IMU, had to be moved out of the EC once the TAM (Three-Axis Magnetometer) was added, because it would not work inside the box and it was desirable to have the IMU and TAM as close as possible (because any non-negligible offset would make the navigation algorithms more complicated). It is also practical to have the DAM in the direct vicinity of both sensors, in order to keep the analog signal lines from the TAM as short as possible. This revived the same problem again, as the non-decoupled power

wiring popped again out of the EC. Because it was impractical to decouple the whole Navigation Unit (NU), and because the 2.4 GHz model RC systems were becoming available at the time (which are much more technologically advanced in other aspects too - they offer the redundancy the original Wireless Control System lacked), it was decided to replace the RC system and get rid of this problem for good. The Spektrum DX-7 system is not affected by the on-board computers any more, because it works well out of the frequency spectrum of the electromagnetic noise generated by them.

3.5.3 Three-Axis Magnetometer Issues

The Three-Axis Magnetometer also didn't work properly for a long time for unknown reasons; it was discovered later that two problems were actually present. The first and most severe issue was caused by an undocumented bug in the Philips LPC-2119 MCU, affecting the Sample-and-Hold circuit of its A/D converter under certain conditions, when the A/D channels were multiplexed. This bug causes interferences between the multiplexed A/D channels. The bug was fixed using a work-around and was later confirmed by the manufacturer in actualized errata document.

The other problem was caused by the sensor degaussing circuitry, which did not work properly. It was discovered later that there was a bug in the circuitry scheme, which was presented in the datasheet of the HMC2003 as a default solution.

Chapter 4

Control Algorithms

4.1 Overview

The vehicle control algorithm had been designed to control a rotorcraft UAV (a helicopter), with the ability to hover, but could be easily adopted for a fixed-wing aircraft (in fact, such control scheme would be much simplified, due to less degrees of freedom an aircraft has compared to a helicopter). The control scheme is hierarchically structured (see Figure 4.1), consisting of five separate layers. The first layer (*Angular Rate Control Layer*, ARCL) is responsible for the angular rates stabilization in the three vehicle axes. The second layer (*Attitude Control Layer*, ACL) serves for the attitude stabilization, while the third (*Velocity Control Layer*, VCL) controls the vertical and horizontal velocities of the vehicle. The fourth layer (*Position Control Layer*, PCL) is used to stabilize the vehicle position in space. The uppermost layer (*Trajectory Tracking Layer*, TTL) is responsible for the vehicle guidance along a pre-set trajectory.

This control scheme, as envisioned in Figure 4.1, is by far not completely implemented, not to mention validated and tested. In the current state of development, only the first and second control layers (ARCL and ACL) were implemented. The first layer has been thoroughly tested and is currently in working state, while the second layer passed laboratory testing and also the very first flight test (focused on proof of concept), when only the yaw angle was controlled (for the test results see chapter 7). The ACL will be undoubtedly subjected to further intense development (see section 4.4.2) and is not definite by any means. The other control layers were not implemented yet.

The same holds for the Kalman filter [47], depicted in Figure 4.1. It is meant for the sensor data fusion, in order to determine all necessary input data for the control algorithms. In order to do this, a complete "navigation" Kalman filter would be needed (such as [24] for example), for the higher-level algorithms require data that are not directly measurable (attitude, velocities and space coordinates). At this point of de-

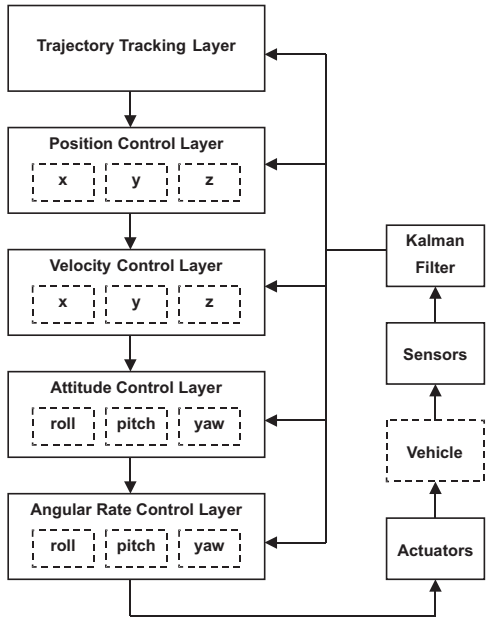


Figure 4.1: Control algorithm layered structure

velopment, no such Kalman filter is available in the RAMA project, although it will have to be developed ultimately. Only the data required for the first two control layers are available at the moment; the angular rates are measured directly and therefore represent no problem, and the attitude is determined analytically (see section 4.2).

In the next sections, the current state of development in the fields of measurements and sensor data fusion, as well as control algorithms, will be described.

4.2 Measurements and Sensor Data Fusion

For the two lowermost layers currently implemented, angular rates and attitude angles of the vehicle are necessary as inputs. The angular rates are measured directly by the gyroscopes, and therefore pose no problem. No Kalman filter or other algorithm for sensor fusion was necessary when only the ARCL was functional.

For the ACL, attitude angles are needed, and obviously those can be determined only indirectly, using some sort of sensor data fusion. The sensor data fusion and attitude determination is the work of Marek Peca, who is the author of algorithms mentioned in this section.

A Kalman filter would be the most obvious way to go, but for start, it was decided to develop a simple analytical computation as the very first method. The reasons behind this were the following: To obtain a better insight and to have some sort of workable solution, simple to debug; to have a benchmark for the Kalman filter devel-

opment; and to perform a feasibility study that would confirm that the data provided by the sensors are good enough and that the angles could be reliably determined. A Kalman filter is not so simple to debug and could also mask various systematic errors, that could come into play later. Also, a Kalman filter is very good in "masking" occasional corrupted samples, but on the other hand could "get lost" on some bad data and its recovery might be problematic, even once the data pick up again. In this case, it is very convenient to have some sort of simple solution, working as a reference, which has no memory (operates only with a single sample) and therefore recovers immediately once good data are received. This could be used to "reset" the Kalman filter and help it out of such a trap.

The Kalman filter for attitude angles determination is currently under intensive development by Marek Peca and close to completion, but it has not been finalized yet. It is similar to the solution presented in [23]. Currently, attitude angles for the ACL layer are computed only analytically. Let us now briefly explain the algorithm.

Euler angles [29] are used to express the rotation of the vehicle body in space. There is the possibility of the gimbal lock [30] occurance for the 90° tilt angles, but it does not matter for the attitude stabilization. The gimbal lock would be a problem for trajectory reconstruction and will have to be solved once the sensor fusion algorithm will be used for navigation. The obvious solution is to use the Quaternions [34] instead of the Euler angles. The Euler angles were chosen as an intermediate solution, because they are simple, intuitive and provide a direct insight into the algorithm, which Quaternions do not.

The attitude is computed using only the accelerometer and magnetometer data, i. e. the acceleration and magnetic intensity measurements in three vehicle axes. It is not possible to use any other data (i. e. the angular rates), because the resulting system of equations would be overspecified and therefore analytically unsolvable.

It is assumed that the vehicle is not accelerating much, so the acceleration measurements are given solely by the earth gravity vector. Both acceleration and magnetic intensity measurements are filtered, in order to get rid of the vehicle-induced accelerations and noises, affecting both accelerometers and magnetometers.

The first rotation (from the earth to the body frame) is chosen to be around the z axis, so it is independent on the acceleration measurements (the earth frame is chosen so that the gravity force vector coincides with the z axis and is zero in x and y axes, so the gravity measurement in the frame rotated around the z-axis is the same as in the original frame). So the first rotation angle is determined solely from the magnetic intensity measurements. The other rotations are chosen around the x' and the z'' axes and could be determined using solely the gravity measurements. Using this process, the rotation matrix R can be determined using following algorithm (the measurements are taken in the body frame, so the rotation matrix is computed in order from the body frame to the earth frame):

The first rotation from the body frame around the z'' axis can be computed using gravity measurements:

$$\Theta_1 = arctan\left(-\frac{g_x}{g_y}\right) \tag{4.1}$$

where Θ_1 is the angle of the first rotation and g_x, g_y are the gravity measurements in the x'' and y'' body frame axes. Now the second rotation could be computed as follows:

$$\Theta_2 = arctan\,\frac{g_x sin\,\Theta_1 - g_y cos\,\Theta_1}{g_z} \tag{4.2}$$

where Θ_2 is the angle of the second rotation around the x' axis and g_z is the gravity measurement in the z'' (and consequently also in the z') axis, because the first rotation did not affect the z'' axis measurement.

Now the rotation matrix R_{12}, representing the first two rotations, can be constructed:

$$R_{12} = \begin{pmatrix} c_1 & -s_1c_2 & s_1s_2 \\ s_1 & c_1c_2 & -c_1s_2 \\ 0 & s_2 & c_2 \end{pmatrix} \tag{4.3}$$

Where $s_1 = sin\,\Theta_1$, $c_1 = cos\,\Theta_1$, $s_2 = sin\,\Theta_2$ and $c_2 = cos\,\Theta_2$. The next step is to transform the magnetic intensity measurements into this newly constructed frame, using the R_{12} matrix:

$$v^T = h^T.R_{12} \tag{4.4}$$

Where h is the column vector of magnetic intensity measurements h_x, h_y and h_z in the body frame and v is the resulting column vector of the measurements, transformed into the doubly-rotated inter-frame. Now the last rotation is in order, computed using the transformed magnetic intensity measurements:

$$\Theta_3 = arctan\,\frac{v_y}{v_x} \tag{4.5}$$

where Θ_3 is the last Euler angle. Let us now construct the last rotation matrix R_z, representing the last rotation only:

$$R_z = \begin{pmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

where $s_3 = sin\ \Theta_3$ and $c_3 = cos\ \Theta_3$. The complete rotation matrix R can be obtained by:

$$R = R_{12}.R_z \tag{4.7}$$

The condition when the denominator of one of the $arctan$ function arguments in equations 4.1, 4.2 and 4.5 becomes zero corresponds to the gimbal lock condition in that attitude angle, meaning that a degree of freedom has been lost. This is a non-issue for this algorithm, because the C function *atan2* is used to compute the $arctan$ function, which can handle the zero-denominator condition; it returns 0 or π in that case, depending on the sign of the nominator. The corresponding attitude angle, to which the degree of freedom was lost, would then be computed accordingly. Therefore, this algorithm can safely handle the gimbal lock singularities, inherently present in the Euler angles attitude representation.

Unfortunately, the Euler angles Θ_1, Θ_2 and Θ_3 do not directly correspond to the yaw, pitch and roll (YPR) attitude angles of the vehicle. This is because the $z-x-z$ rotation was used for the sake of derivation simplicity, while the YPR attitude angles correspond to the $z-y-x$ rotation. It is therefore necessary to convert the angles Θ_1, Θ_2 and Θ_3 into the YPR angles ψ, ϕ and θ:

$$\begin{aligned}
\psi &= arctan\ \frac{R_{21}}{R_{11}} \\
\phi &= arctan\ \left(-\frac{R_{31}}{R_{11}cos\ \psi + R_{21}sin\ \psi}\right) \\
\theta &= arctan\ \frac{R_{13}sin\ \psi - R_{23}cos\ \psi}{R_{22}cos\ \psi - R_{12}sin\ \psi}
\end{aligned} \tag{4.8}$$

The conversion 4.8 can be found in the Robotic Toolbox for MATLAB [48].

4.3 SISO PID Controller Structure

RAMA currently utilizes SISO (Single Input Single Output) PID controllers at all levels of control. The controller scheme was proposed by Ondřej Holub and implemented by the author of this thesis, and is further developed in close cooperation of both authors. It is our original PID implementation (the controller internal structure is depicted in Figure 4.2), inspired by [49]. On top of standard PID control law it utilizes some additional techniques, like reference filtering and weighting, feed-forward branch and separate D-component filtering and restriction. It is also equipped with the bumpless control transition block, useful when taking over of a manually-controlled actuator.

Reference weighting in the P branch helps to suppress the non-linear gain of the controlled system, while in the D branch it serves to restrain the reference kick.

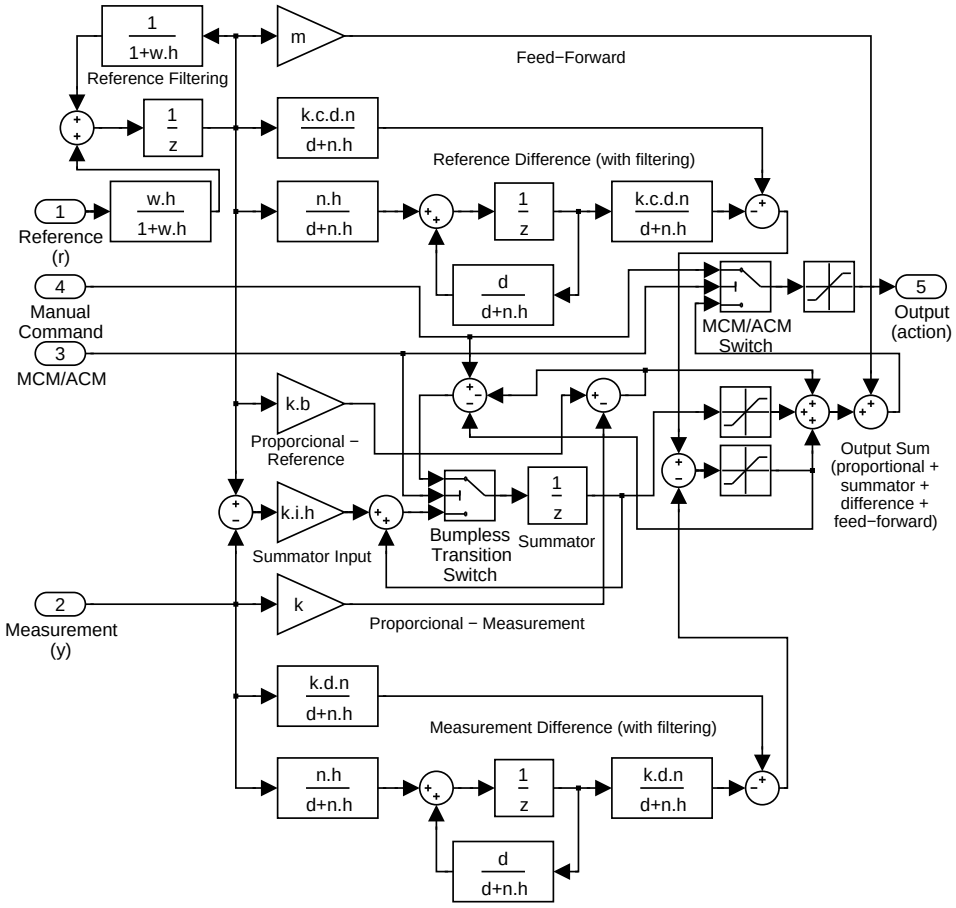


Figure 4.2: PID controller internal structure (Simulink scheme)

The controller parameters are as follows:

- k, i, d, m - Proportional, Integral, Derivative and Feed-Forward gains
- b, c - Reference weighting coefficients for proportional and derivative components
- w - Reference filter cut-off frequency
- n - D filter cut-off frequency
- h - Sampling period

The difference computation of the reference r and measurement y are divided into two separate branches, to allow for the reference weighting (as described in [49]).

Hence, the controller has separate inputs for r and y, instead of a single input for the control error e. A low-pass frequency filter is embedded in both D branches, to get rid of the high-frequency noise possibly superposed on the input signals. The D branch contribution to the output sum is bounded by a static restrictor.

The I branch is fairly conventional. Its contribution to the output is limited by a static anti-windup filter. In the Manual Control Mode (MCM), the bumpless transition branch sets the summator value of the I branch so that the controller output precisely matches the manual action. Therefore, when the mode switch occurs and the Automatic Control Mode (ACM) is engaged (the controller takes over), no sudden kick to the actuator can happen.

The P branch is also conventional, only the reference r is weighted by the parameter b before the control error e is computed.

The output of the controller is limited by a static restrictor as a safety measure, to prevent the action to exceed the saturation limits of the actuator (although this should not happen if the parameters are set properly).

4.4 Low-Level Control Loops

4.4.1 Angular Rate Control Layer

The *Angular Rate Control Layer* (ARCL) consists of three separate, independent SISO PID controllers, one for each axis (yaw, pitch and roll). The angular rates of a rotorcraft are completely independent and each can be controlled directly by a separate actuator, which makes the situation particularly simple from the architectural point of view. The yaw rate is controlled by the tail rotor thrust, the pitch rate via the longitudinal cyclic control and the roll rate through the lateral cyclic control.

Control parameters in respective control loops are very different, because the vehicle dynamics is very diverse in each axis. The yaw has the fastest dynamics, because the moment of inertia of the vehicle is the smallest in this axis. Small rotorcraft can typically spin at rates as fast as $350-450^\circ/s$ and the vehicle step response is very crisp - it can accelerate/decelerate significantly in that axis (see section 7.3.2.1 for details). The yaw rate dynamics is the determining parameter for the sampling frequency. At the beginning of the RAMA project, the sampling frequency of 32 Hz was deemed sufficient, but it turned out that the yaw rate is hardly controllable at this frequency. It was therefore increased to 64 Hz (the highest frequency the IMU can handle) and the situation got better, but it is still to slow. Another problem is that the SCU operates the actuators only at 50 Hz, and that frequency is fixed. Hardware and firmware revision would be needed in order to increase the sampling/actuating frequency any further. Such upgrade is planned for the future, because at least 100 Hz sampling/actuating frequency would be needed for a completely stable tail control. The summary of the yaw rate identification/control experiments and control loop settings can be found in section 7.3.2.1.

The pitch and roll rates are generally much slower than the yaw rate dynamics. This is caused by several factors; The most significant is the large gyroscopic effect of the main rotor, which has strong damping effect on the pitch and roll motions. The Bell-Hiller stabilizer (if present) is another important factor, especially if the flybar paddles are heavy. The pitch and roll rate actuators (the longitudinal and lateral cyclic controls) are the same, but the vehicle response in pitch is somewhat slower than in the roll, because the moment of inertia of the fuselage is much greater for the pitch movements compared to the roll. The maximal pitch and roll rates are approximately the same (in the range of $80 - 150°/s$ for a typical small UAV), but the angular acceleration in pitch is slower and the vehicle inertia is greater, compared to the roll. Experience shows that around 25 Hz sampling and actuating frequency is sufficient for the pitch and roll rate control. The RAMA system operates at the same frequency for each axis, so the pitch and roll control loops are running at the same 64 Hz sampling/50 Hz actuating frequency as the yaw loop, which is more than enough for the purpose. The summary of the pitch and roll rate identification and control experiments and settings of the control loops can be found in sections 7.3.3 and 7.3.4.

4.4.2 Attitude Control Layer

Unfortunately, the ARCL is the only layer where no dependences among the control loops can be found; The situation becomes more complicated for the Attitude Control Layer (ACL) and the following layers as well. There are inherent interdependencies among the attitude angle control between all three axes. For example, imagine a vehicle executing the following simple sequence of attitude maneuvers - first a $45°$ left roll, then $50°$ nose-up pitch and finally a $45°$ right roll. At the end of this sequence (assuming the original attitude was $0°$ yaw, $0°$ pitch and $0°$ roll), the vehicle will yaw $25°$ to the left, pitch $25°$ nose-up and roll by $0°$.

Obviously, the systematically correct solution would be some kind of MIMO (Multiple Input Multiple Output) controller, taking the interdependences into account. However, if the vehicle tilts by a reasonably small amount around a fixed working point, the interdependences can be neglected and separate SISO controllers could be used in each axis. For each controller, the attitude change induced by the other controllers into his realm would be just an additional external disturbance. If the current attitude is set as a new working point in each sample, this approach will work for the entire attitude envelope. This solution works well under the assumption the attitude maneuvers are reasonably slow, compared to the sampling frequency, which is a condition well fulfilled if the sampling frequency is sufficient for the angular rate control.

Setting the new working point in each sample also solves another particularly annoying problem as a by-product. This problem is the singularity point in the attitude measurement, when an attitude angle changes from $359.9°$ to $0°$. This change would cause a mayhem in the PID control loop if left unattended. If the working point for

each attitude angle is set to say $180°$ in each sample (the setpoint must naturally be re-calculated with respect to the newly set working point), this condition becomes a non-issue.

The ACL layer have been implemented in the RAMA system in the manner described above, i. e. as an independent PID controller for each axis. The attitude control loop output produces a set point for the corresponding rate control loop in each sample, so the controllers in the ACL and ARCL loops are connected in cascade.

Only the yaw angle control was tested in flight so far (see section 7.3.2.2), although all three control loops are implemented in the software (they are identical piece of code). The yaw angle hold test performed so far was only a proof of concept and the yaw angle controller was not tuned properly.

Chapter 5

Software Architecture

5.1 Overview

The RAMA system software is relatively complex and consists of the programs running on its three on-board computers - the Main Control Computer (MCC), the Data Acquisition Module (DAM) and the Servo Control Unit (SCU) - and on the Ground Station (GS). The DAM and SCU programs are running system-less (without any operating system), while the MCC and GS both use the Linux operating system. Any Linux distributions, utilizing kernel version 2.6.19 or above can be used. The MCC program does not have any special requirements, although FIFO scheduling policy must be supported, as the real-time thread priorities are used. The GS software requires the QT library 4.2.x or higher.

5.2 Basic Principles of Function

Basically, RAMA is a synchronous time-triggered system, working at the 64 Hz frequency, although some of the events can happen asynchronously (those related to various failure modes) and some of the data (the GPS provided data) are sampled at a different frequency. All three main nodes of the system (the MCC, the DAM and the SCU) are communicating solely via the Vehicle Bus (VB). The DAM provides the rest of the system with the *Time Synchronization Message* (TSM), sent periodically 64 times per second. This message determines the working cycle of the system - it marks the end of current working cycle and the beginning of the next. This is how all three nodes are commonly synchronized.

At the beginning of each working cycle (Figure 5.1), the data sampling and processing is taking place. Upon sending the TSM, the DAM prepares the latest sensor data (inertial measurements taken from the IMU and magnetic measurements from the TAM), and sends them to the MCC. The TSM also triggers sampling and sending of the control stick positions in the SCU.

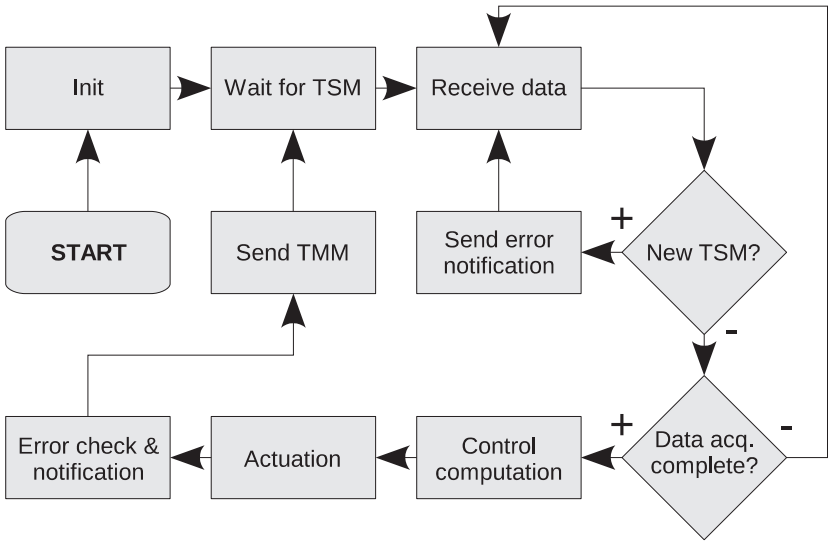


Figure 5.1: Working cycle of the RAMA system

The TSM prompts the MCC to force-finish the current working cycle (in case it did not end already) and wait for the new data acquisition. The force-finish should never occur under normal circumstances, as the regular working cycle should end well before the start of the new cycle. Occurrence of the force-finish indicates that something went wrong in the last cycle and is reported via telemetry as an error notification (see section 5.3.4).

Upon receiving the essential measurements for the control algorithms, the MCC executes the control loop, which computes the new control actions. Afterwards, newly computed actions are fed into the SCU (which applies them at the beginning of the next actuator control period, see [46] for details). When all measurements for the current cycle are completed and received, the synchronous *Telemetry Measurement Message* (TMM) is composed and sent to the Ground Station (GS) via the Wireless Data Communication Unit (WDCU).

The GPS data are sampled and processed independently, at 1 Hz sampling rate. This is not ideal, but there is no other way, because it is not possible to synchronize the IMU and the GPS (both are stand-alone units with their own internal timing). There is no control over their sampling points, nor any way to synchronize them. In fact, the TSM is derived directly from the IMU sampling point, so the RAMA system timing is built around the IMU timing. On the other hand, it is not a major problem either; the GPS data can be processed fairly independently in the control algorithms and does not necessarily need to be precisely synchronous with the rest of the data.

In its current state of development, RAMA can operate in two control modes - either in the Manual Control Mode (MCM), when the vehicle is fully controlled by

a human pilot, and RAMA serves only as a telemetry device; or in the Automatic Control Mode (ACM), when RAMA takes over some (or all) actuators. From the controller development point of view, it is sometimes convenient if automatic control could be applied only to some selected actuators, while the others are controlled manually; that's the reason why this option was implemented.

In both control modes, the operating cycle is almost exactly the same; there are two differences though. The first one resides in the control action computation. In the MCM, the action is computed, but the integrator state of the controller is set so that its output value equals the corresponding control stick position (see section 4.3) - this is done to provide the bumpless transition from the MCM to the ACM. The other difference is that the SCU does not apply received action values to the actuators; it applies the stick position signals directly to the actuator control signals (see [46] for details).

The communication protocol of the Vehicle Bus (VB) is relatively simple. Short 11-bit IDs are used and the priority of each message is determined by its ID (the lower the ID is the higher is the priority). The bit rate is 1 Mbps (the maximum for the CAN bus). For details on the CAN datagram format and CAN 2.0b physical layer, see for example [37]. The messages are broadcast and each node receives all the messages; however, it ignores all messages that do not affect his own operation. See the appendix A.3 for the complete list of VB messages and their meaning.

Let us now describe the telemetry communication protocol. There are several types of messages, distinguished by their identificator (ID). Messages are character-encoded (meaning that a message can only contain characters), so the numbers are sent as byte-strings in hexadecimal format (for example, a float number is sent in the form of its memory representation, i. e. as a four-byte number, where each byte is represented by two characters in the telemetry message). Each message starts with the starting character @, followed by the two-character message ID (the ID is a hexadecimal number in the range of 00-FF). Optionally, some data may follow. The messages do not have a termination character, nor are they secured by a CRC or any other consistency check; it is assumed that the TCP/IP protocol takes care of all this, so it would be redundant.

There are several asynchronous messages, which can be sent anytime; these are used to indicate asynchronous events, such as the error conditions on the control mode transition (ACM/MCM or reversed).

Two synchronous data messages are used, each sent in different time intervals. The first message contains all measurements except the GPS data and is sent 64 times per second (corresponding to the working cycle frequency of the RAMA system), while the other contains the GPS data and is sent once in a second (corresponding to the GPS sampling rate). Complete list of the telemetry messages can be found in appendix A.2.

The telemetry communication is one-way, no data are sent from the Ground Station (GS) to the Airborne Part (AP) of the system. Multiple Ground Stations are

allowed; The Airborne Part provides a data server and the telemetry can be broadcast among multiple clients.

5.3 Main Control Computer Software Framework

The MCC program is decomposed into several libraries and the main program. The libraries contain functions for CAN and TCP/IP communications, the PID control algorithm and several support functions, while the main program contains the rest. Let us now briefly describe the supporting libraries first, and the main program later.

5.3.1 Controller Library

The controller library consists of two functions. The function PID_init serves to initialize the structure, holding the internal parameters and state variables of the controller. The second function, called PID_control, contains the control algorithm itself. More details on that algorithm can be found in section 4.3.

5.3.2 Library for CAN Communication

This is a very simple support library for CAN communications. SocketCAN [50] is used, therefore the basic communication mechanism is exactly the same as for any other socket communication. The header file of this library contains definition of the CAN device identificators and the IDs of the CAN messages. This is not ideal though, as the message IDs are also defined in the DAM software, so there are actually two places where the definitions can be found; therefore, when editing an ID number (or adding one), two files have to be modified. This is because some legacy software was used, and shall be rectified in future software revisions.

The sole function contained in this library serves for the CAN device initialization, and is pretty self-explanatory (see the comments in the source code).

5.3.3 Library for TCP/IP Communication

The ipcomm library contains functions allowing one to set-up a simple server for the socket communication. This server is capable to handle multiple clients and broadcasts the telemetry messages. The server runs aboard the RAMA system, while the Ground Station (GS) is a client.

The header file of the library contains several constant definitions, such as various buffer lengths, the server port number and the bitwise data flags. Those flags are used to check for the data consistency in each sampling period; as each data part is written into the data holding structure for each data message, corresponding flag is set to indicate that the data have been properly updated.

The data holding structures for both synchronous messages are defined in the header file. Note that the GPS data structure consists only of a 64-byte long array and the data consistency variable - that is because the GPS data are currently not used for any on-board calculations, and therefore do not need to be parsed. Therefore, they are recorded into the array directly as they are received from the DAM and re-sent to the GS, and no decoding and processing takes place in the on-board software. Naturally, this will change with later software revisions, introducing navigation and guidance algorithms.

Basically, the server is running in two separate threads. One thread takes care of the client management, an the other for broadcasting data to all active clients.

The "client management thread" (function server) checks for activity on any of the socket filedescriptors, that is on the server socket and the active client sockets. The select function is used to handle multiple events in a single program thread. If a new client connects to the server socket, new socket is created for it and the file descriptor of this new socket is added into the active_fd_set list. When one of the active clients closes connection, it is removed from the list and the corresponding filedescriptor is closed. This list is used by the "broadcast thread" to send the data to all active clients.

The "Broadcast thread" (function send_packet_thread) checks the output data round buffer for any unsent data and broadcasts them to all active clients. The write function is blocking - it does not resume until all data are sent and their reception is acknowledged by the recipient. Therefore, should any communication problems occur, this thread would be suspended until the data are received properly by all the clients. In the meantime, all upcoming data are stored into the output round buffer by the sending threads. The size of this buffer is sufficient to handle up to 15 minute-long communication hangups. The buffer is statically allocated, so its size is constant - no dynamic allocations are used anywhere in the code for safety reasons, as is common in embedded system software. When the buffer is full, the oldest data are simply overwritten (as is normal in round buffers). The function send_packet is used by the other threads to submit data into the output round buffer - so it does not send the data directly. This function is non-blocking, so the submitting thread cannot be suspended because of possible communication problem.

5.3.4 Main Program

RAMA's main MCC program consists of four threads, each running in an infinite loop. Two threads are responsible for the telemetry broadcasting (see section 5.3.3), one for the Vehicle Bus (VB) communication and the last one for the control algorithm execution. The program is event-driven, so the control actions are triggered by data reception from the VB.

The working cycle of the program (Figure 5.2) is determined by the Time Synchronization Message (TSM), provided by the DAM. Upon receiving this message, the next working cycle begins. The VB messages are received and processed by the

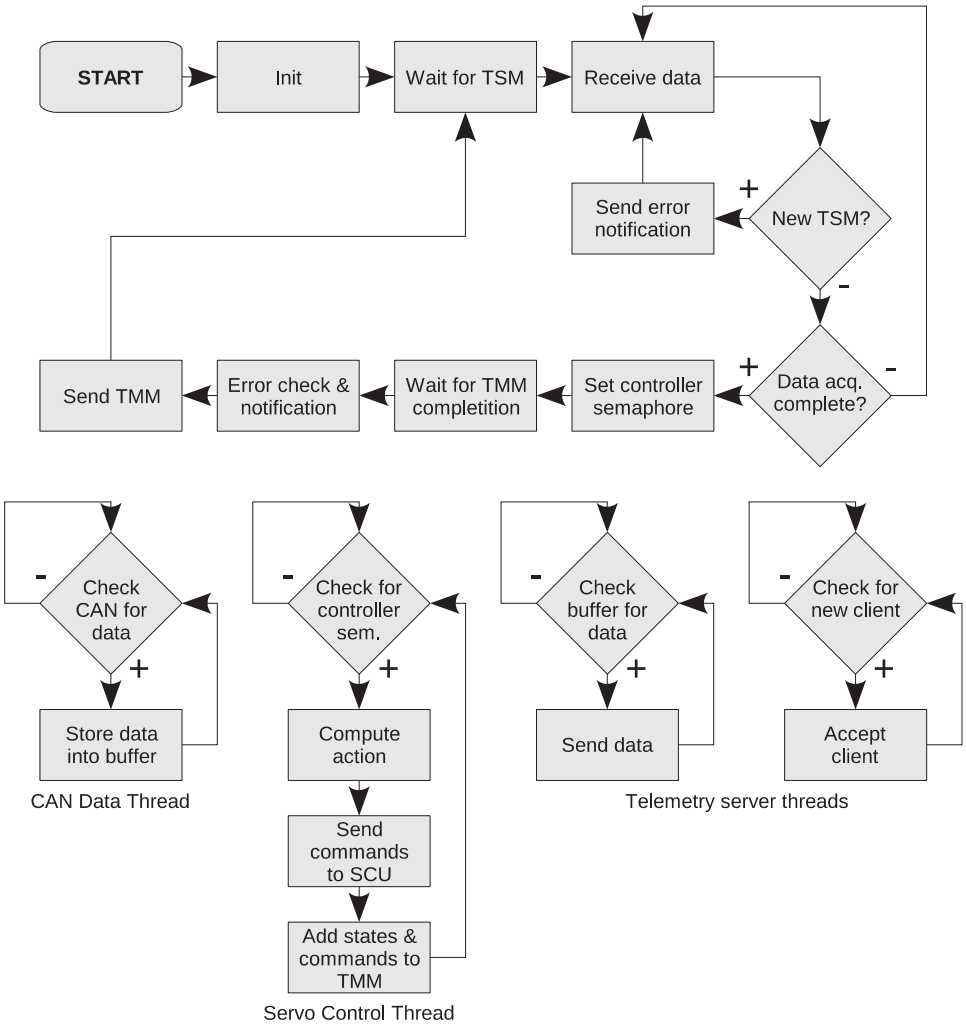


Figure 5.2: Working cycle of the Main Control Computer

read_from_can_thread, which reads the message and calls the processing function, somewhat obscurely called send_ip_msg.

This function is switched according the message ID and does the basic processing. After receiving all the data needed for the controller to run (determined by the reception flags), a semaphore is set, which in turn enables the servo_control_thread to run. When all data for the Telemetry Measurement Message (TMM) are gathered, the message is composed and submitted to the telemetry buffer, and the working cycle is finished. If next TSM is received prior to the end of the previous cycle, this cycle is force-finished - the missing data are not updated (in fact, the last values remain recorded in the data reading structure) and the Telemetry Data Composition Error

(TDCE) message is generated.

The servo_control_thread begins with the initialization of the controllers for the ARCL and ACL control layers, and then enters an infinite cycle. At the beginning of each cycle the thread waits for the synchronization semaphore (controller_enable) to be set. Afterwards, all the input values are copied into its local variables (to prevent a possible racing condition) and the normalization of the control stick positions is done. Control actions are computed afterwards - in the current version (7.1), only the Angular Rate Control Layer (ARCL) and Attitude Control Layer (ACL) are implemented. The ACL, if enabled, works as an attitude holder; it works only when the control sticks are in neutral positions. In that case, it provides the reference for the ARCL. If the ACL layer is disabled or the corresponding control stick is not in the neutral position, the ACL is inactive and the control stick position provides the reference for the ARCL (the desired angular rate).

When the computation is finished, the resulting control actions are de-normalized (servo positions are computed) and resulting commands are sent to the Servo Control Unit (SCU).

Before starting all the threads and entering the working cycle, the main program performs the initialization. A command is sent to the Data Acquisition Module (DAM) to reset itself. Then all the parameters (controller and filter parameters and other stuff) are read from the configuration file. Some of these parameters are sent over to the DAM, using the Vehicle Bus (VB). This is all done by the control_init function. The regular working cycle is not entered before the sensor calibration is performed. The length of the calibration process is determined by the cal_sample_count variable, which is read from the configuration file and determines the number of calibration samples. This procedure is used to determine the neutral positions of the control sticks and the offsets of the gyroscopes. It is assumed that the vehicle stands still during the calibration process and the control sticks are in their neutral positions. The angular rates, provided by the gyroscopes during the calibration process, are sampled and their mean value is computed for each gyroscope. This mean value is then subtracted from the corresponding gyroscope measurements in the run-time, to compensate for its offset. The same algorithm is used to determine the neutral positions of the control sticks.

5.4 Data Acquisition Module Software

The Data Acquisition Module (DAM) runs system-less. The program is relatively complicated though, and utilizes the resources of the Philips LPC 2119 MCU (Micro Controller Unit) to a relatively great extent. This is mainly because the numerical filtration of the measurements is taking place here, computed in the float numbers, and three separate devices must be monitored (The Inertial Measurement Unit or IMU, the Global Positioning System or GPS and the Three Axis Magnetometer or TAM).

To achieve greater precision, the TAM is internally sampled at 512 Hz (although the results are averaged and sent out along with the IMU data at 64 Hz). The IMU is sampled at 64 Hz, as was said many times before, and the GPS at 1 Hz. Both IMU and GPS are "intelligent" units and have serial interfaces, sending digital datapackets. The IMU is connected to the UART0 and the GPS to the UART1 ports of the MCU. The TAM is an analog device, giving three-channel voltage measurements proportional to the magnetic intensities, and is connected to the A/D converter (channels ADC0, ADC1 and ADC2).

The three ADC channels cannot be scanned successively, one after another, in a single time step; the error in the sample and hold circuitry of the LPC 2119 prevents this. Therefore, each channel is scanned separately in adjacent time steps. The timing is provided by the timer T0. This timer is running at 1536 Hz; This means that the measurements of all three axes are acquired 512 times per second (therefore the overall sampling frequency is 512 Hz).

PWM channel PWM2 is used to provide a 1 kHz signal for the Villard (Delon) step-up voltage converter (this voltage is being used to generate the periodic degauss pulses for the TAM).

Several simple libraries are utilized to access the peripherals, namely the CAN, UART and PWM. These libraries are heritage pieces of code and their detailed description is beyond the scope of this thesis. They are relatively simple and well documented in their source files.

The program starts with the initialization of the peripherals. The internal PLL of the MCU is set to multiply the external clock six times, therefore (with external 10 MHz crystal) provides the internal core clock at 60 MHz. The peripheral clock is set to the same frequency. The Vector Interrupt Controller (VIC) and the PIO (Parallel Input-Output ports) are initialized immediately after, followed by the PWM, CAN controller and watchdog. A loop is entered afterwards, awaiting the reception of the configuration parameters (the number of the calibration samples and the filter parameters). After receiving all required parameters, the rest of the peripherals are initialized (the ADC, both UARTs and the timer T0) and the main program loop is entered. In this loop, new data reception flags are checked for both UARTs and the CAN. The incoming data from each of these peripherals are read in their interrupt handlers and stored into separate round buffers.

When data are received from the UART0, the IMU_service function is called. This function contains a simple state-machine, which receives and decodes the datapackets provided by the IMU. The Time Synchronization Message (TSM) is also generated here. Any problems, detected during datapacket reception and decoding, are reported via error messages. When the process is completed and newly acquired data are filtered, they are broadcast on the Vehicle Bus (VB), both the raw samples and the filtered data.

UART1 data reception indicates the new GPS data arrival, and the GPS_service function is called to process them. Similarly to the previous case, there is a simple

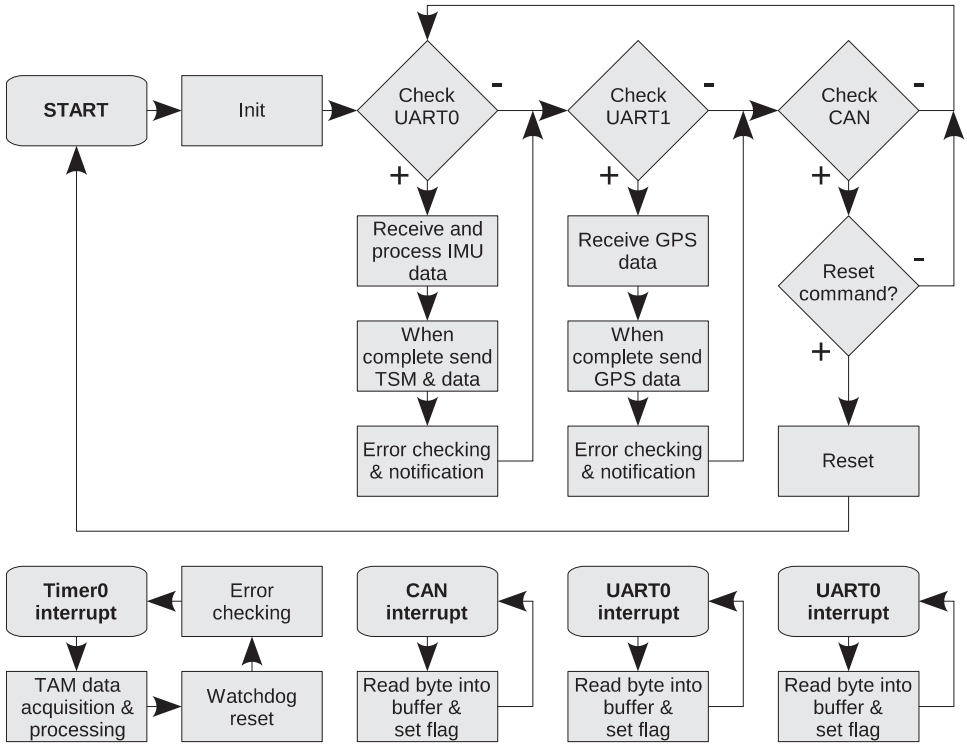


Figure 5.3: Working cycle of the Data Acquisition Module

state machine taking care of the data. When complete datapacket is received, it is broadcast on the VB. No data decoding or processing is applied.

In the case of the VB (CAN) data reception, the can_rx function is called. It mainly serves to receive the initial parameters before the main loop of the program is entered; In the run-time, all but one message are ignored. The only message not to be ignored is the command to reset. If this command is received, the watchdog resetting (which is normally taking place in the timer T0 handler) is disabled, and therefore the watchdog itself triggers the MCU reset. This is the most simple and reliable way to ensure a clean MCU restart.

The working cycle of the DAM is depicted in Figure 5.3.

5.5 Servo Control Unit Firmware

The Servo Control Unit firmware was designed and implemented by Ota Herm, a graduate student, working under the supervision of the author of this thesis. The documentation is currently available in the Czech language only and can be found in [46].

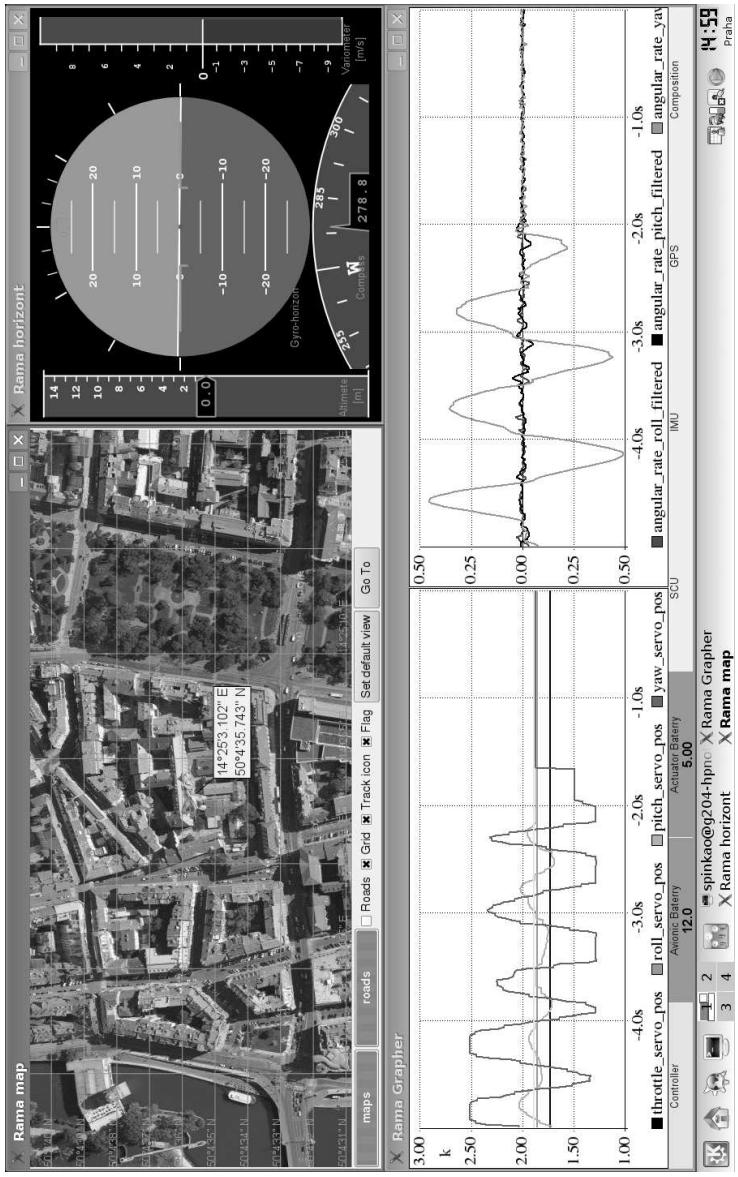


Figure 5.4: Ground Station (GS) of the RAMA system - display screenshot

5.6 Ground Station

The Ground Station (Figure 5.4) was designed and implemented by Petr Svoboda, a graduate student, working under the supervision of the author of this thesis. The documentation is currently available in the Czech language only and can be found in [51].

Chapter 6

Control System Safety

6.1 Overview

UAV control system is undoubtedly a critical system, and must have a reasonable safety margin to be of any practical use. An out-of-control UAV might cause property damage and/or personal injury (not to mention vehicle loss), even a small one, for which the RAMA control system is intended. However, other ways than the classical redundancy have to be sought, for the reasons noted in the introduction. Let us now describe various safety mechanisms incorporated in the RAMA UAV control system.

RAMA is designed so that for most of the failures, there is an appropriate *Failure Mode* the system can engage, in order to preserve the critical functions (i. e. steering) for the sake of some high-level functionality loss. Most of the failures would likely cause the mission to abort, but there is still a good chance of vehicle recovery by some sort of semi or fully manual control. If the experienced failure is so severe that it leaves no chance of saving the vehicle, RAMA should at least minimize the consequences - meaning that it prevents the injured craft from swinging around wildly and unpredictably, but would cut off the engine and put the control surfaces into some predefined positions, for orbiting-gliding in case of a fixed-wing aircraft and auto-rotation in case of a rotorcraft (this is called the *Critical Failure Mode* - CFM).

The principles of graceful degradation and system reconfiguration are used to achieve these goals. The only critical node of the system is the Servo Control Unit (SCU), whose complete failure would definitely lead to grave consequences. Partial or complete failure of any other system part (the Main Control Computer, the Data Acquisition Module, or some/all of the sensors) would lead to the loss of some functionality, but would definitely preserve the semi-automatic or at least fully manual control of the vehicle (providing that the wireless control communication is working).

It must be noted that there is no overall “failure control system”, which would put RAMA in an appropriate fail-safe mode automatically in case of a failure. It is always the responsibility of the pilot to perform the reconfiguration by flipping a

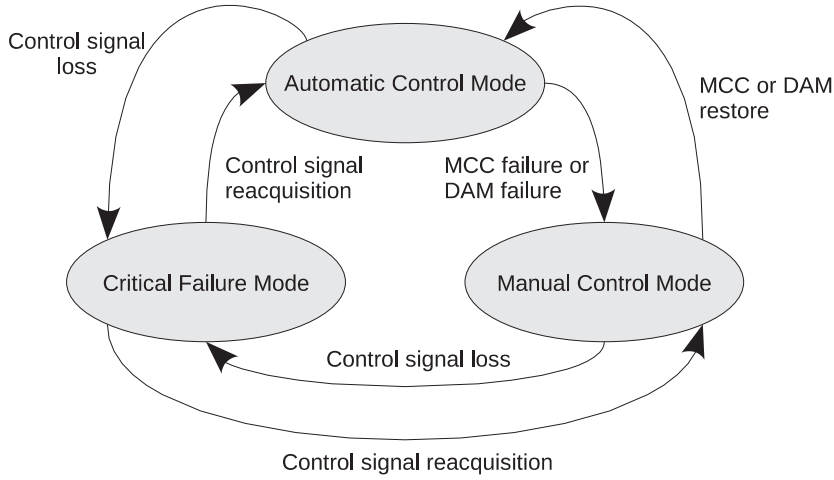


Figure 6.1: State diagram of RAMA's failure modes - current state

switch if he/she thinks it is necessary. The RAMA system only gives a warning that something has gone wrong, but does not perform any reconfiguration on its own. This is intentional, as pilots generally strongly dislike any autonomous system out of their direct control, which can dramatically change the behavior of the vehicle on its own decision.

A fully automatic "failure control system" would also add some failure modes itself and would increase the overall system complexity. RAMA generally does not need such a system, as it is not intended to operate autonomously. It is assumed that RAMA will always operate under a human pilot's direct supervision.

The only failure mode that RAMA enters automatically is when the wireless control signal from the ground is lost. Naturally, in this case, the vehicle is not under a human pilot's control anymore, and the control system must act on its own.

6.2 Control System Failure Modes

6.2.1 Main Control Computer Failure Modes

In case of the MCC failure (of any type) the only viable solution is to put the Servo Control Unit (SCU) into the Manual Control Mode (MCC). In this mode, the SCU takes over the direct control of the actuators and runs unaffected by the rest of the system. It ignores any commands possibly coming from the faulty MCC or other nodes and actuates the control surfaces directly according the control stick positions.

This solution is good enough in the current state of RAMA's development, because there is still the modeler Futaba GY-401 tail rate controller present (it is not used when RAMA takes over control, it serves only as a backup for the case of MCM

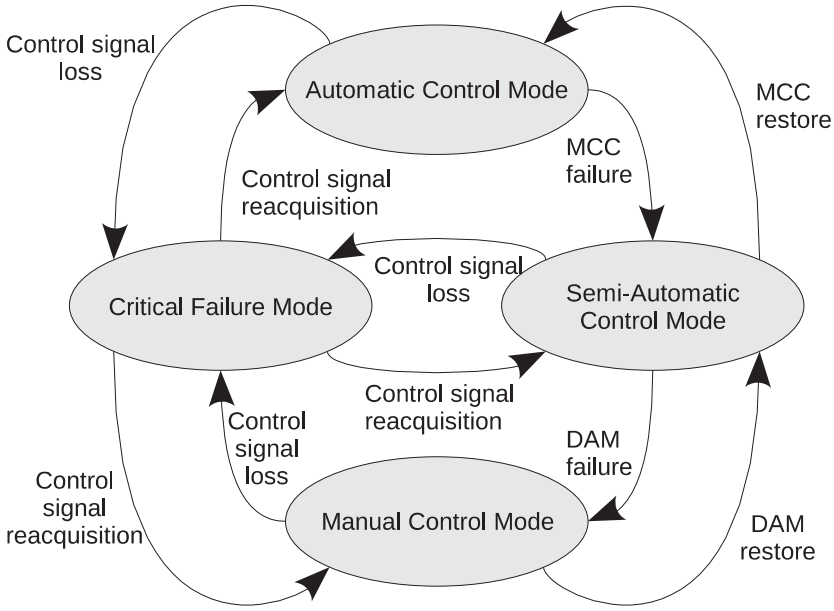


Figure 6.2: State diagram of RAMA's failure modes - proposed future solution

engagement) and also the Bell-Hiller roll and pitch rate stabilizer. So, when in the MCM, the helicopter could still be controlled as any other model helicopter with the help of the mechanical Bell-Hiller system and a backup electronic tail rate stabilizer (the GY-401). But the situation will change radically once those remnants of hobby helicoptering will be removed, as is ultimately planned. There will be no backup for the disabled electronic rate stabilizers anymore, which would be a very unfortunate situation. Therefore, there is a plan for future modification of the SCU software, so that it could execute the rate stabilizing algorithm directly in case of need. The SCU has more than enough computational power for this and the modification would be relatively simple and straightforward.

Once completed, there will be two failure modes available in the case of the MCC demise: The SCU could be reconfigured to take over the Angular Rates Control Layer (ARCL) instead of the MCC, providing the Data Acquisition Module (DAM) and Inertial Measurement Unit (IMU) are still running, entering the so-called *Semi-Automatic Control Mode* (SACM). Or, should the IMU and/or the DAM fail together with the MCC, the SCU could perform another reconfiguration, entering the Manual Control Mode (MCM), as it does now. This would make the human pilot really earn his money since his task would be significantly harder without the help of the rate stabilizers, but still gives him/her a good chance of recovering the injured vehicle somehow - for example by auto-rotation.

Overall, in all cases, the telemetry would be lost, together with higher control layers, but the semi-automatic or at least fully manual control of the vehicle should be

preserved in any case.

The MCC has no watchdog timer, as its possible restart would take too much time anyway, so it would bring no benefits.

The MCC failure is not detected automatically in any way, or at least not directly. The Ground Station (GS) monitors the MCC indirectly via the telemetry data, and in case it detects any erratic behavior (such as controller internal variables out of range, a datapacket loss or CRC errors), it informs the pilot that something is wrong with the telemetry (and possibly the MCC). However, it is up to him/her to decide whether to engage the Semi-Automatic Control Mode (SACM) or not.

Critical MCC failures are quite obvious to the pilot - the vehicle simply stops responding to his/her commands or behaves erratically. Again, it is up to him/her to engage the SACM. The reconfiguration is naturally completely independent on the MCC - the SCU receives the command and reconfigures itself into the MCM, so the reconfiguration works even in the case of a total MCC failure.

State diagram of the failure modes is shown in Figure 6.1 (current state of development), and proposed future solution, extended by the Semi-Automatic Control Mode, is depicted in Figure 6.2.

6.2.2 Navigation Unit Failure Modes

The Data Acquisition Module (DAM) failure makes the system enter the Manual Control Mode (upon the pilot's command). The DAM is equipped with a watchdog timer and brown-out detection (undervoltage reset) protective circuitry. A possible DAM restart after the watchdog reset would only take about 50 ms, which is fast enough compared to the vehicle dynamics.

In case of a failure of some of the sensors, the respective control layer which needs the missing data (along with all higher layers) would be turned down.

Failure of any of the sensors is detected by the DAM. The GPS and IMU (Inertial Measurement Unit) are both off-the-shelf products equipped with its own internal diagnosis, so they both issue an error message if anything is wrong with them. GPS reports the estimated position errors and the number of satellites detected, while the IMU reports the out-of range measurements and other internal errors. The communication loss with the GPS or IMU is detected by the DAM via timeouts, and is reported as a critical GPS or IMU failure. Also, communication with both units is secured via the CRC. The CRC errors are reported by the DAM to the MCC and eventually via telemetry to the Ground Station (GS). The affected data are not used in the control loops. The density of CRC errors is also monitored and reported to the GS.

The Three-Axis Magnetometer (TAM) is an analog unit. The out-of-range measurements and unnaturally rapid changes in the data (gradients) are registered and reported by the DAM as errors.

The DAM failures are detected by the MCC only by erratic or completely lost communication (via the CRC and timeouts). Also, an unexpected DAM reset (possi-

bly via the watchdog) is monitored and reported by the MCC (at start-up, the DAM sends the boot-up message to the MCC). If the DAM communicates, it is assumed it works well. DAM failures are reported by the MCC via the telemetry to the GS.

6.2.3 Servo Control Unit Failure Modes

The Servo Control Unit (SCU) failure would inevitably lead to the vehicle loss. If at least some of the actuators would be working (in the case of a partial SCU failure), the system would enter the Critical Failure Mode (CFM). The SCU is naturally the most rugged unit of the whole system, kept as simple as possible and thoroughly tested. It is equipped with a watchdog timer, ensuring a restart in the case of a microcontroller hangup. A full restart of the SCU is fast enough (it takes about 50 ms, much like the DAM restart). It also has the brown-out detection circuitry. However, the SCU should be redesigned in the future, to provide at least some redundancy (meaning that at least the Manual Control Mode should be made redundant somehow).

6.2.4 Communication Failure Modes

In case of a data communication loss (detected by the TCP/IP protocol via the loss of acknowledges), the telemetry datapackets are stored in a buffer and sent off-line as soon as the data link is re-established. Moreover, each datapacket is assigned a unique number and a time stamp, to allow one to check the received data consistency. Telemetry is not critical and can be also downloaded off-line, after landing.

Failure of the wireless control link (detected via the control signal loss) would currently make the system enter the Critical Failure Mode (meaning it sets the engine throttle to idle and engages neutral control surface positions). This is necessary because the higher control layers are not implemented yet, so the vehicle is not able to operate autonomously. In the future, when all control layers will be fully functional, this mode may be changed so that the vehicle would enter hover (in case of a rotorcraft) or orbiting (in case of fixed-wing) in case of this failure. The system has the ability to fully recuperate from the Critical Failure Mode immediately once the control link is re-established, so the vehicle can still be recovered if this failure occurs only momentarily.

The wireless control link is very reliable though. It is redundant - the RC transmitter transmits the control signal on two separate channels, with the ability to switch dynamically to another channel if an interference is detected. The system is equipped with two independent Wireless Control Units (receivers), with antennas positioned to mutually perpendicular polarizations. Only one channel is sufficient to maintain full control over the vehicle. Digital data encoding, secured with the CRC, is used. The transmitter is identified by a unique ID code, embedded into each datapacket, to prevent any other transmitter possibly taking over the vehicle control.

6.2.5 Power Failure Modes

Currently, there are no Battery Failure Modes - both batteries must be in good shape to keep the system running. Failure of any of them would have catastrophic consequences. A re-design of the SCU is considered, the purpose of which would be to make the SCU able to run from either battery, so it can drain power from the Actuator Battery (ACB), should the Avionics Battery (AVB) fail. Moreover, it should also be able to re-route power to the actuators from the AVB in case of ACB failure. This would bring a power supply redundancy into the system, without adding any mass or size.

The "backbone" power wiring and power switches (i. e. the lines whose failure would cause the whole system to fail due to power loss) are divided into two redundant parallel branches. This measure does not add much weight, while the safety benefits are obvious.

6.3 Fault-Tolerance of the RAMA System in Real Life

Let us now show three real-life cases, which accidentally occurred during flight tests and where the fail-safe ability of the RAMA system helped to prevent an accident. In the first case, the control signal was lost for approx. 600 ms (most likely because of radio interference). At the time, the vehicle was equipped with an older version of the Radio Control (RC) set, operating at 35 MHz with no redundancy. The control system entered the Critical Failure Mode (CFM) in response to the signal loss, and later recovered after the control signal reappeared.

Reaction of the RAMA control system to this failure can be seen in Figure 6.3, showing the actuator positions in time (on the y axis, 0 means actuator neutral position, while -0.5 means full throw to one side and 0.5 full throw to the other side). At 193.1 seconds into the flight, the control signal is lost. The condition is immediately detected and the control system reacts by engaging the CFM, which means re-positioning the actuators to preset positions - engine throttle to idle (fully negative actuator throw), collective pitch to slightly positive and roll, pitch and yaw actuators to their neutral positions. After 600 miliseconds, the control signal reception is renewed and the control system recovers from the CFM. The condition occurred when the vehicle was banking left at full-speed in forward flight, and although it was negatively perceived by the pilot, it did not cause a crash thanks to the rapid recovery. If the actuators were not set into neutrals and were subjected to some uncontrolled noise in the critical phase, the vehicle would likely crash.

In the second case, the roll gyroscope malfunctioned because of extensive vibrations of the helicopter body, caused by a mechanical resonance mode that showed up during the flight. The root cause was later traced to a loose bolt in the rotor head damper. In this case, the control system successfully engaged the Manual Control Mode (MCM). This action undisputedly saved the vehicle from crashing, as the roll

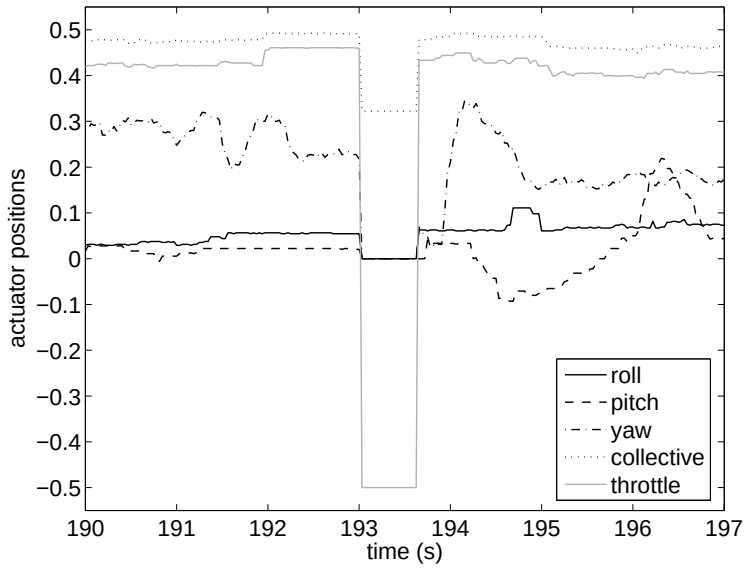


Figure 6.3: Control system response to the in-flight control signal loss

rate control loop did not work, leading to the loss of vehicle stability. In Figure 6.4, the roll rate measurements, along with the reference signal and actuator position, are shown. At 155 seconds into the flight (marked by the dashed line) resonance occurred, causing the roll gyroscope to fail (the mean value shift, explained in section 3.5.1, was experienced). The problem caused the roll rate controller to compensate, falsely believing that the vehicle was rolling, inducing itself an undesired roll. The pilot immediately commanded reconfiguration to the Manual Control Mode (MCM) and recovered the vehicle. Please note that from the dashed line on, the roll rate gyro readings and actuator position signals are no longer correlated. This is because the MCM was engaged, and in this mode the sensor readings are ignored and the rescaled reference signal (issued by a human pilot) is directly fed to the actuator (it can be seen that from the dash line on, the actuator position signal is equivalent to the rescaled reference signal).

The fail-safe ability and early warnings issued by the control system via the online telemetry were invaluable during flight tests, and prevented potential vehicle loss many times. In the third case, a failing actuator battery almost led to a crash, had it not been for the early warning issued by the control system. RAMA discovered an unusually high voltage drop on the Actuator Battery (ACB), which did not show up on earth, because the actuators were not subjected to any load (which is induced by aerodynamic forces acting on the control surfaces during the flight). This triggered a warning, issued to the human pilot. The vehicle was immediately landed and the suspected battery actually failed only two minutes after. This case led to the decision

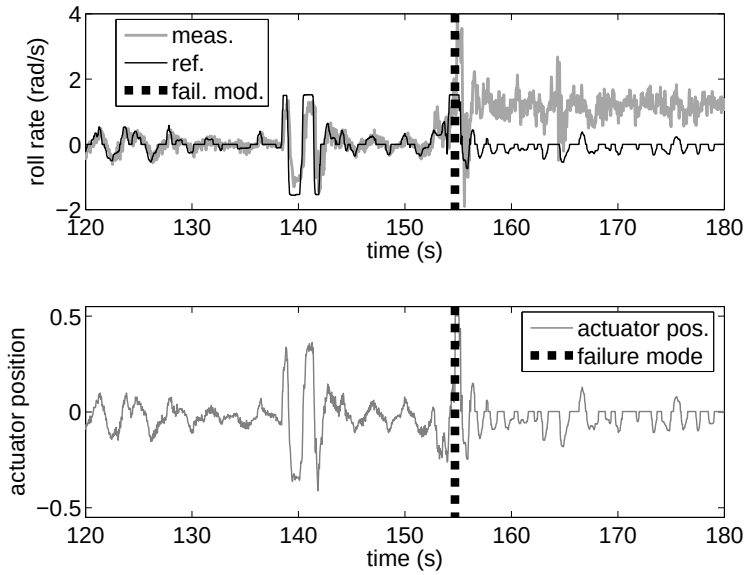


Figure 6.4: Control system response to roll rate gyroscope fault

to re-design the Servo Control Unit in the future, as was described in section 6.2.5.

Chapter 7

Experiments and Testing

Tests and experiments are inherent part of every research and development process. When developing a flying vehicle, we may basically distinguish between the *static tests* of the whole system (or its part), being conducted safely in the home comforts of a laboratory, and the *flight tests*, when newly designed and developed assets are put through their paces in real-life conditions. The former is of vital importance, never to be underestimated, and often determines the outcome of the latter. The flight tests are inherently expensive and dangerous, and it is essential to be well prepared, in order to maximize the chances of a positive result. This is why the static tests are conducted. Let us now describe the testing methodology, used when developing the RAMA system, and present the most important testing results.

7.1 Testing Vehicle

Hirobo Freya EVO 90 hobby-helicopter (Figures 7.1 and 7.2) is used as the RAMA carrier for the flight tests. It has the H1 (or Heim) type of swashplate, meaning that there is a separate actuator for each function (longitudinal cyclic, lateral cyclic and collective), so no software mixing is needed. This type has many advantages over the commonly used CCPM setup (for explanation of various swashplate types, see [52]), the only disadvantages being its comparably higher cost and mechanical complexity.

Yamada 91ST two-stroke combustion engine is used as the powerplant. Standard coreless Futaba S9202 servomotors drive the cyclic controls and the engine carburettor, while fast digital servos are used for the collective and the tail rotor control; Futaba S9350 is used for the collective and Futaba S9254 for the tail.

The rotor diameter is 1580-1620 mm (depending on the type of blades currently in use) and the fuselage length is 1350 mm. The vehicle, mated with the RAMA system, weights approx. 6.5 kg.

Figure 7.1: Hirobo Freya model helicopter, carrying the RAMA system

Figure 7.2: Hirobo Freya model helicopter, carrying the RAMA system

7.2 Static Tests

The goal of the static tests is to validate the system as much as could be done without flying and to prove its reliability before the flight tests. The latter is especially important, because every part of a UAV control system is more or less safety critical and any in-flight failure may have grave consequences. By *validation* we mean a verification that the newly developed part of the system really does what it is supposed to do and that it fulfills all the requirements. The goal of the *testing*, on the other hand, is to verify that the system is reliable and safe enough to fly.

There is no practical way how to generally describe the validation process, be-

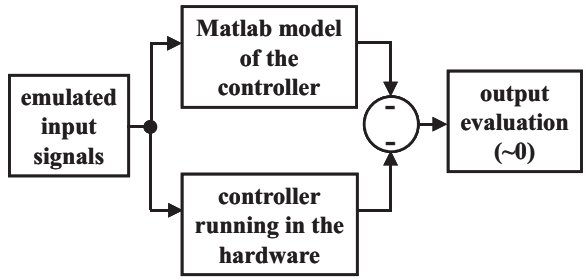


Figure 7.3: Controller performance evaluation setup

cause every feature is different and the validation tests must be tailored accordingly. On the contrary, the testing of the overall system *reliability* and *safety* is, and should be, a generally codified process, to ensure that no corners are cut when preparing to a flight test. The term “reliability” is used to somehow describe the probability of a failure; on the other hand, the term “safety” describes the consequences of a failure. When testing the reliability, the system must pass a set of test cases without any error; to test the safety of the system, errors or non-standard situations are deliberately induced into the system and the reaction is evaluated.

For the RAMA system, a general testing methodology has been developed and practiced over the years. It is called the *Flight Readiness Test* (FRT). It is a set of tests cases and focal points for both the software and hardware, avionics as well as the vehicle.

7.2.1 Flight Readiness Test

7.2.1.1 Avionics Tests

The new functions, implemented into the avionics, are validated and tested using usually some kind of the *processor-in-the-loop* tests. The testing setup for this kind of test is depicted in Figure 7.3. A Matlab/Simulink model of the newly tested feature runs on simulated data. The same data are fed into the control algorithm, running in the real hardware. The results of the simulation and the implementation must match. The evaluation is done offline, for the sake of convenience. This kind of test not only validates the algorithm, but is also able to discover systematic implementation errors.

Once the system is validated, the reliability tests take place. The avionics is left to run for a long enough time (an hour or so) and is stimulated with some kind of arbitrary signal, trying to emulate likely in-flight conditions. The control outputs, as well as the internal states of the algorithms, are monitored through the telemetry and checked for error codes or any other anomalies.

The hardware is also looked upon:

- The connectors, switches and wire harnesses are inspected for any signs of

wear.

- The mounting points are checked.
- The range of the Radio Control system is tested.
- Twice a year, both on-board batteries (ACB - Actuator Battery and AVB - Avionics Battery) are tested for capacity, along with the battery in the RC transmitter.

The safety is tested by intensionally inducing errors into the system. These can be induced by the software itself in some special cases, but the standard test involves disconnecting, reconnecting and resetting of various system parts in run-time. The standard testing procedure involves:

- DAM hardware reset.
- DAM loss - DAM power connector is disconnected.
- IMU and GPS disconnection/reconnection from the DAM.
- MCC loss - MCC power connector is disconnected.
- Control signal loss and reacquisition - the RC transmitter is switched down and up again.
- Loss of one power branch - one of the redundant power switches is switched off, then the other is tested in the same way.

The control system must respond correctly to all these states and appropriate fail-safe mode must be engaged. The manual control must be preserved all the time (the obvious exception being the control signal loss - in that case, the Critical Failure Mode must be engaged, and disengaged again after the signal reacquisition).

7.2.1.2 Vehicle Inspections

The Hirobo Freya helicopter is also inspected before each flight test:

- The fuel filter is cleaned.
- The fuel and tank pressurizing manifolds are inspected.
- The mechanics is checked for any signs of wear, loose parts and FODs (Foreign Object Debris).
- The blade grips are checked for any excessive play and the dampers are inspected and lubricated if necessary.

- The main and auxiliary rotor shafts are lubricated, along with the transmission, if needed.
- The bearings are tested for unobstructed movement.
- The control linkages are inspected for any excessive play.
- The one-way bearing within the main wheel is tested.
- The clutch is inspected.

7.3 Flight Tests

The flight tests are the ultimate proving ground for each system part. They play an unsubstitutable role in the development process and provide the invaluable real-life experience. However, each flight test is a substantial logistical undertaking and also pose an inherent risk of vehicle damage or even a complete loss. There is no second chance if anything goes fatally wrong in flight; that is why the static tests are so important. Each flight test has to be well prepared, the testing setup outlined and proven and testing priorities must be set, to avoid any confusion in the field. A flight test usually consists of 1-3 days and only a limited number of flights is available (a maximum of 10-14 flights a day), due to several constraints (weather, battery capacity, pilot fatigue and others). It is not possible to introduce any major last-minute changes to the testing setup during the process, emphasizing the need to statically test the hardware and software thoroughly beforehand and clearly define the testing means, goals and priorities. It would be technically possible to update the flight software in field, but it is never done for principal reasons, for such behavior would strongly enhance the possibility of inducing a human error.

A flight test is also relatively expensive in both money and labor, because of the logistical challenge and hardware attrition and maintenance, so it should be carefully considered what is worth flight-testing and what is not, in order to not to waste precious flight time. It is not possible to flight-test every bright idea that passes by and testing objectives must be carefully selected.

After each flight day, a post-flight review takes place. The data are inspected from several viewpoints. The performance of the tested feature is naturally evaluated, but not only that. Several standard procedures also take place:

- The error codes are checked.
- Internal states of the algorithms are checked.
- The battery performance is evaluated - the voltage graphs of both ACB and AVB are checked.

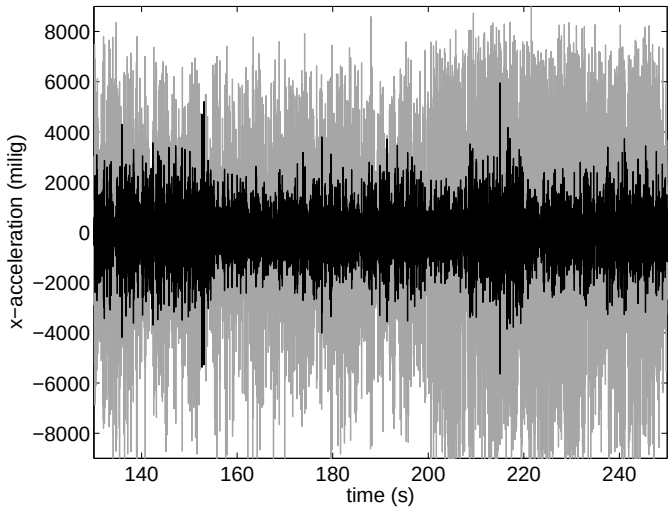


Figure 7.4: x-axis accelerometer measurements in hover, difference between two flights. The gray signal corresponds to an early type of IMU fixture, the black signal corresponds to the IMU mounted on the mass damper.

- The data are checked for consistency and the timestamps of the samples are evaluated to look for jitter.

7.3.1 Control System and Sensor Development

Many flight days were dedicated to the instrumentation development, because a lot of problems were encountered with nearly all the sensors under flight conditions (recall section 3.5). The Inertial Measurement Unit proved to be the most problematic, and countless flights had to be dedicated to remedy the situation. See Figure 7.4, where the difference between the data provided by the same sensor within the IMU are shown, depending on the mounting of the unit. The mass damper, described in section 3.5.1, had to be developed and tuned properly before any serious control experiments could begin.

The Garmin 18-LVC GPS receiver also proved to be problematic in various ways (see section 3.5.2). Its mass, located at the very tip of the tailboom, worked as a perfect vibration resonator and induced severe shivering into the airframe. It was very hard to de-tune the system to rectify the issue, because there was an interconnection between the horizontal fin and and the receiver itself, which was very hard to discover. The fin had to be removed eventually and the receiver mounted relatively loosely on some foam dampers. The relocation of the receiver was impossible, because it didn't work under the main rotor blades, which were obscuring the signal. Figure 7.5 shows an example. The GPS position fix express the mode the GPS receiver works in - 1

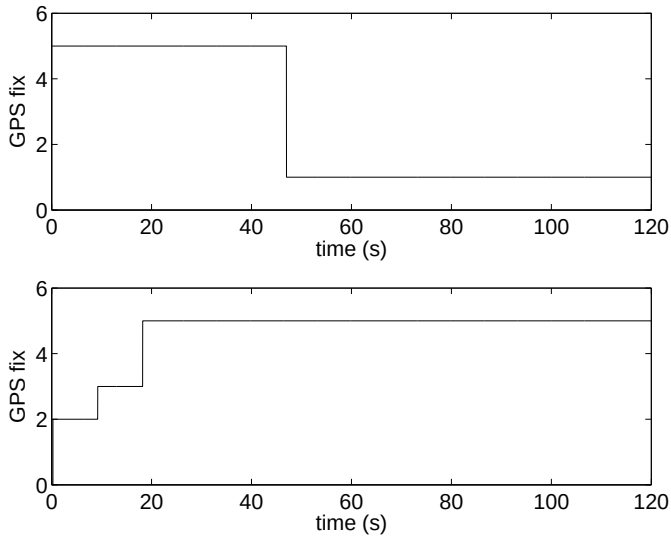


Figure 7.5: GPS position fix examples; above - signal loss, below - normal signal acquisition

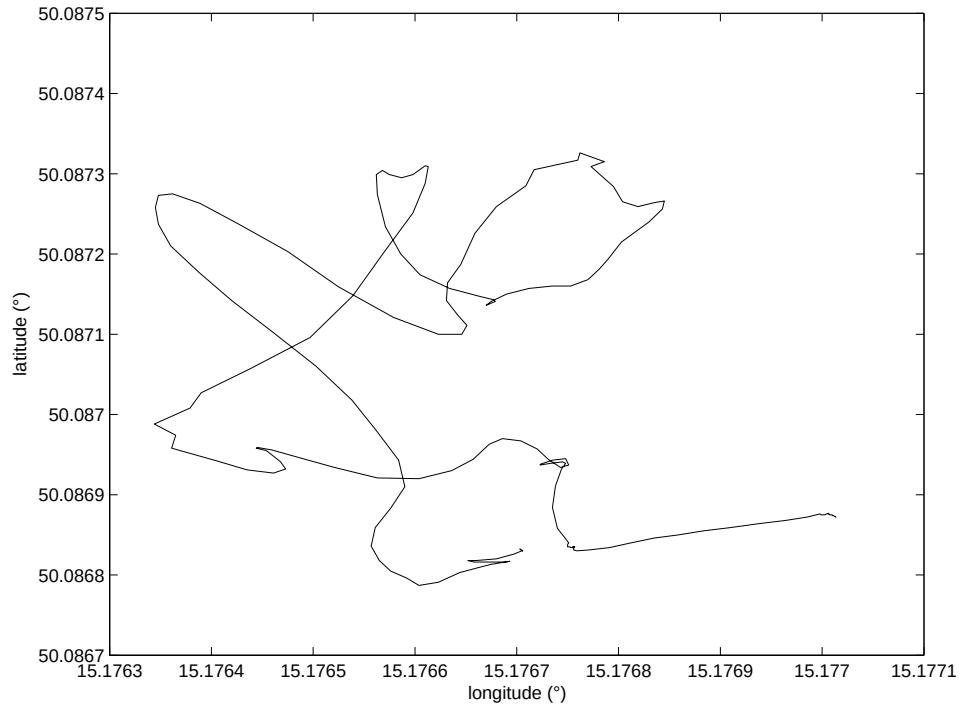


Figure 7.6: GPS recording of a flight path

corresponds to no fix (no signal received), 2 to the 2D mode (no altitude data), 3 to the 3D mode, 4 to the 2D differential mode and 5 to the 3D differential mode. The first case in Figure 7.5 shows the loss of the GPS signal in 43 s time, corresponding to the time when the main rotor started spinning. The second case shows the normal signal acquisition after startup - initial 2D fix is obtained, then the 3D mode is entered and after difference signal acquisition the 3D difference mode is engaged and never lost again during the flight. Figure 7.6 shows the GPS recording of the flight path.

Other system development flights focused on the hardware of the control system itself; the Electronic Container was repositioned several times and its mounting points altered, the connectors and wiring were changed and the mounting of the PCBs inside the EC was also altered many times to obtain the best solution.

7.3.1.1 Hardware-in-the-Loop Testing

There were also some software development flights, verifying the reliability and timing of the system and performance of the control loops under flight conditions. The Automatic Control Mode (ACM) was emulated, in order to verify the feasibility of the whole solution. The testing setup was following: The control system operated in the ACM during the whole flight, the only difference was that the manual commands were actually applied to the actuators at the end of each working cycle, instead of the computed automatic actions. The Servo Control Unit (SCU) operated in the ACM, measuring the positions of the control sticks and sending them to the Main Control Computer (MCC), waiting for commands. The MCC waited for the complete data acquisition, executed the control loops, but at the very end of its working cycle actually sent the measured control stick positions back to the SCU, instead of the computed commands.

The vehicle was therefore controlled manually, but the manual commands passed through the whole control loop (so they were subjected to the same delays as the automatic commands would) and the control system was fully running. After these flights, the performance of the control system was thoroughly evaluated; The timing was analyzed, looking for possible jitter, the internal states of all system parts were verified and the commands, computed by the control loops, were verified using a Matlab model of the control algorithms. The Matlab implementation was fed by recorded telemetry data and its output was compared to the output of the MCC control loops, also recorded in the telemetry. The results had to be the same (see Figure 7.3). This was the ultimate “dress rehearsal” of the control system as a whole, before the first attempts for semi-automatically controlled flights. In fact, it was the complete “hardware-in-the-loop” simulation. By this term, a most realistic testing setup is meant, where all control algorithms are running real-time in the hardware as they would in the real case, and input signals are not emulated - the sensors are actually subjected to the real physical conditions they should measure. This is the major difference compared to the “processor-in-the-loop” testing, mentioned in section 7.2.1.1.

7.3.2 Yaw Control

7.3.2.1 Yaw Rate Control

As was said in section 4.4.1, the yaw rate control is the fastest of all Angular Rate Control Layer (ARCL) loops. It is also the only one which cannot be properly controlled by a P controller in principle, because the vehicle dynamics in the yaw axis contains a first-order astatism; because of the reaction torque of the fuselage, which the tail rotor has to compensate for, the actuator is actually required to be out of the neutral position in order to maintain the zero control error. This can be achieved only when the I component is present, so a I, PI or PID controller is required for this axis.

Initial experiments with the PI controller didn't quite work out, because the controller proved to be rather unstable under variable angular rates. It usually worked fine for near-hover conditions, but as the angular rates went higher, it tended to oscillate excessively (see Figure 7.7). This was partly caused by the rather low sampling rate at the time; 32 Hz was clearly not enough to capture the fast yaw movements. A PID controller was introduced in an attempt to suppress these oscillations, but initially, the D component didn't work very well because of the noise, superposed to the measured signal. Additional signal filtering was introduced, improving the situation, but the PID controller proved rather hard to tune "by hand". It was impossible to find the correct setup intuitively.

So, some identification experiments were performed in order to determine the natural frequency and the critical gain of the vehicle in yaw. A P controller was used for this purpose, which gain was gradually increased up to the point when the tail started to oscillate. The natural frequency and the critical gain were determined and PID constants computed according the Ziegler-Nichols algorithm [53]. The PID controller, set this way, worked reasonably well from the beginning and required only little additional tuning. The reference weighting was also introduced into the P branch of the controller, in order to help to suppress the tail oscillations at higher angular rates. The final setup of the controller is denoted in table 7.1. Please note that instead of the proportional, integration and derivative gains p, i and d, the overall gain k and the circular frequencies ω_i and ω_d are given.

The sampling rate was later increased to 64 Hz, further improving things. The step response (measured in flight) of a properly set yaw rate PID controller is shown in Figure 7.8. The control quality is considerably improved over the PI controller (compare Figures 7.8 and 7.7).

7.3.2.2 Yaw Angle Control

Initial tests of the Attitude Control Layer (ACL) were performed only in the yaw axis so far. Because the ARCL works very well and the attitude excursions of the vehicle are very slow in time, it could be safely assumed that only conservatively set P or PI controllers will most likely be required for the ACL layer.

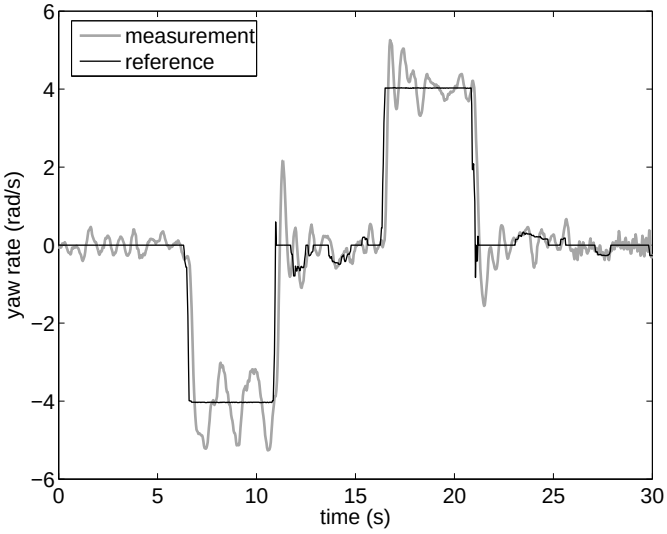


Figure 7.7: Yaw rate PI controller step response

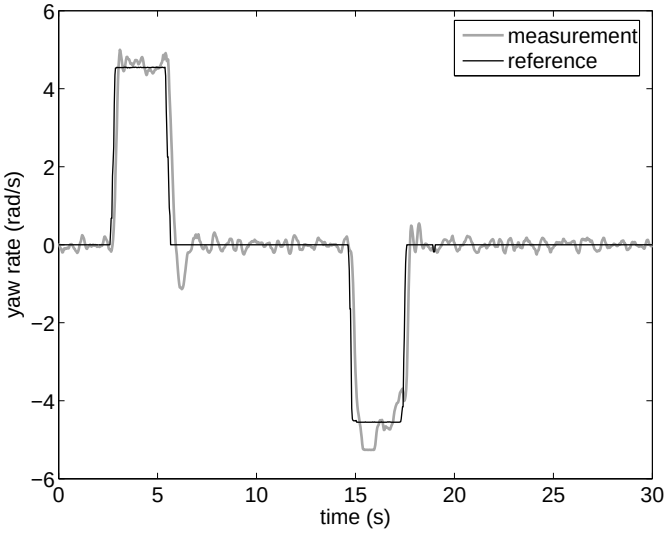


Figure 7.8: Yaw rate PID controller step response. Please note the unnaturally "flattened" spike in the rate measurement around 16 s time. This is caused by the gyroscope going briefly into saturation.

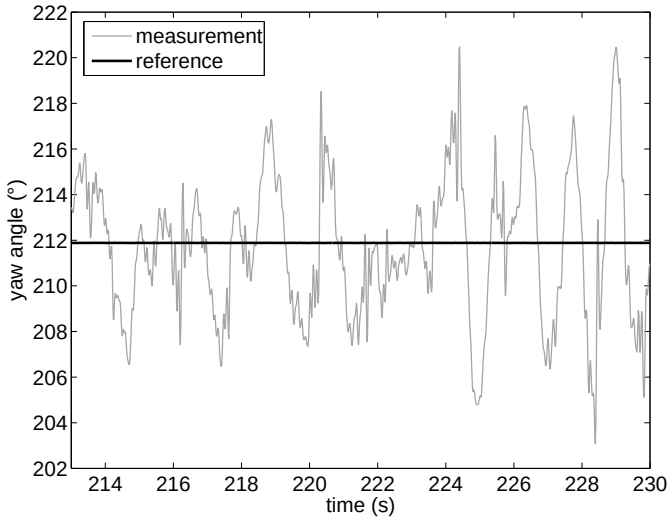


Figure 7.9: Yaw angle P controller position holding

The initial test of the ACL in yaw was hampered by the fact that the attitude computation algorithm was not finalized in time for the test, so only an intermediate solution was used; the yaw angle was computed using simple triangulation of the x and y magnetic intensity measurements. The yaw measurements were therefore working properly only when the vehicle was nearly level; even small bank angles in pitch and roll would affect the yaw angle measurements, because the magnetic field variations induced by the banking of the vehicle were not taken into account.

Even under those adverse conditions the ACL worked, although the control quality was not up to the required level. This is assumed to improve with better attitude measurement. The in-flight performance of the yaw angle control is shown in Figure 7.9. Simple P controller was used and only attitude holding test was performed (the reference was constant). The attitude excursions are caused by the measurement inaccuracy rather then the controller performance. Obviously, the attitude was hold reasonably well in global, although the overall control quality was rather unimpressive.

7.3.3 Pitch Control

The pitch rate control proved to be somewhat tricky. The dynamics of the vehicle in the pitch axis is the slowest and most vehicle inertia is present. The experiments with the P, PI and PID controllers were only partially successful; the same problem was plaguing all of them. This problem was that either the controller suffered a very large phase delay - the response was vastly delayed after the reference (see Figure 7.11) - or the control loop was unstable, experiencing severe oscillations (or both), as shown

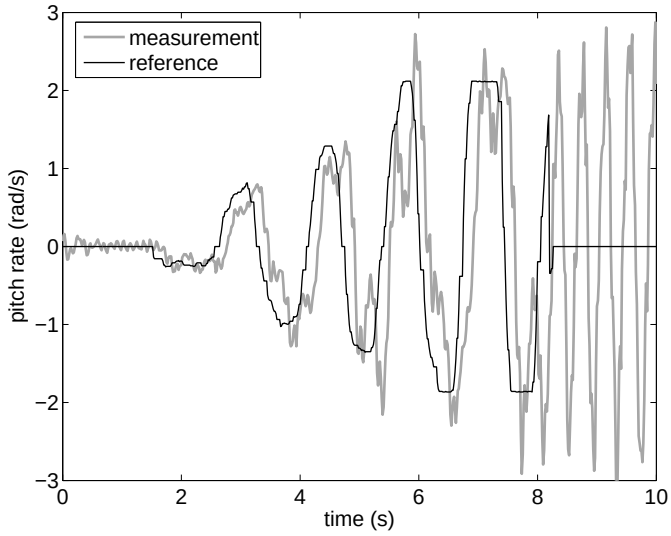


Figure 7.10: Pitch rate PI controller oscillations. The reference tracking is still poor, although the controller is clearly over-aggressive.

in Figure 7.10.

The D component in the controller was not very helpful and the Ziegler-Nichols rule didn't work. In fact, the attempt to determine the natural frequency and critical gain indirectly led to a loss of a vehicle, as will be described later (see section 7.3.5).

The PI controller with a rather large i constant proved to be the most promising solution. It was steady and stabilized the vehicle perfectly in hover, but suffered considerable lag in response to the reference change (as shown in Figure 7.11). This problem was ultimately solved by introducing the feed-forward loop into the controller. The feed-forward allows a part of the reference signal to be applied directly to the actuator, ensuring rapid response to the reference change; the P and I branches work as expected, stabilizing the vehicle. Considerable improvement in response to the reference variations using the feed-forward technique is shown in Figure 7.12 - compare that to the performance shown in Figure 7.11.

The final setup of the pitch rate controller can be found in table 7.1.

7.3.4 Roll Control

The roll rate control characteristics is naturally similar to the pitch, although there is a little less momentum in this axis. Therefore, similar problems were encountered when trying to setup the roll rate controller. The lag issue was present in the roll rate control too, although it was not so severe compared to the pitch (see Figure 7.13). The Ziegler-Nichols rule did not work for the roll rate controller either. The final solution for the roll was achieved in the same way as for the pitch rate - by introducing the

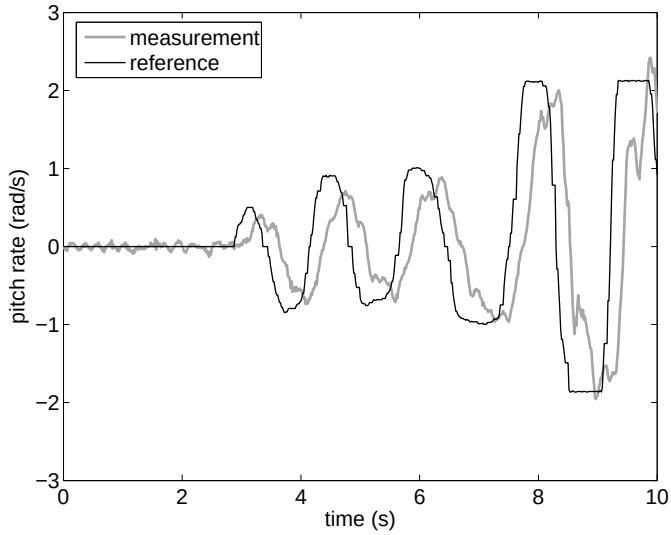


Figure 7.11: Pitch rate PI controller in-flight performance. The reference tracking is delayed in phase by approx. $\pi/2$.

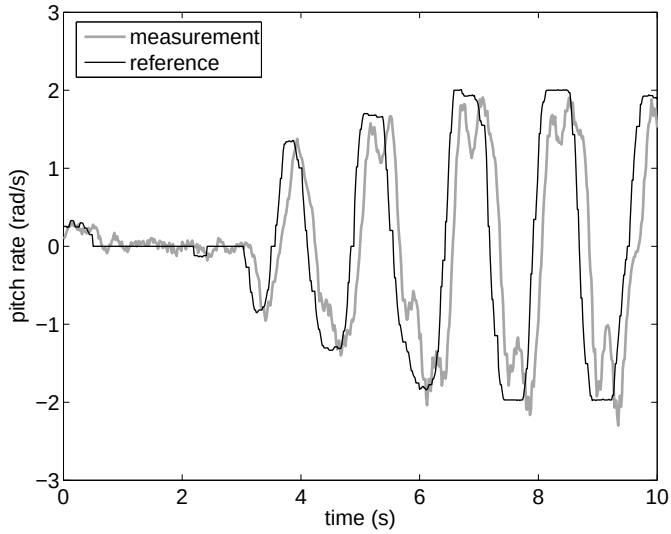


Figure 7.12: Pitch rate PI controller with feed-forward in-flight performance. The reference tracking is still delayed a little, but generally much better than in previous cases.

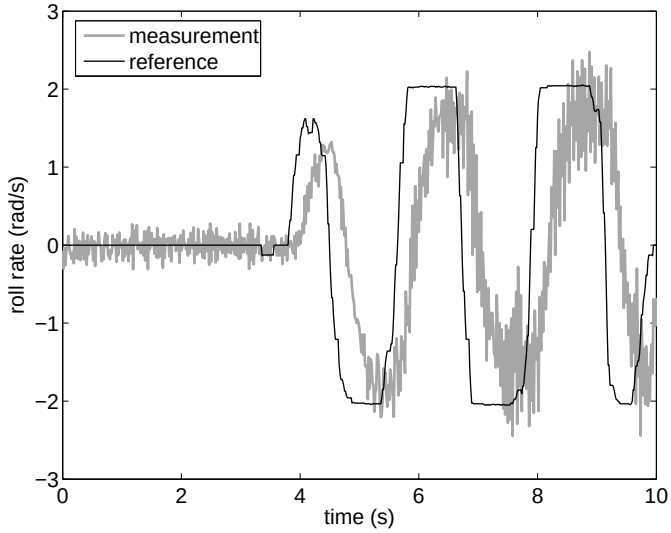


Figure 7.13: Roll rate PI controller in-flight performance. The reference tracking is considerably delayed in phase.

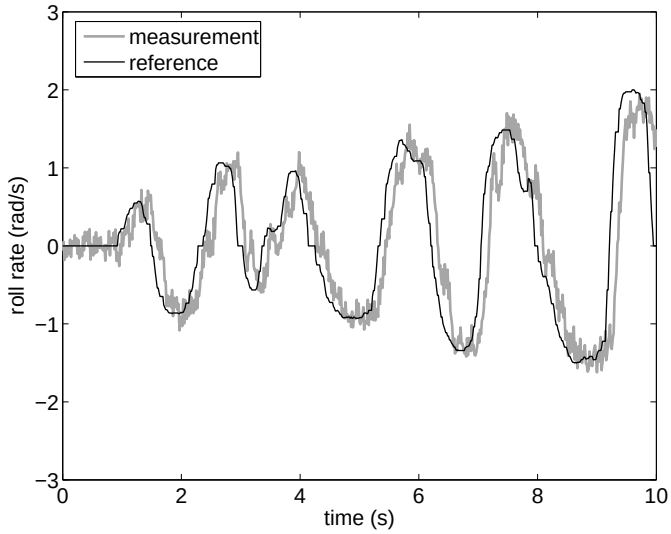


Figure 7.14: Roll rate PI controller with feed-forward in-flight performance. The reference tracking is clearly improved over the previous case.

Basic Parameters			
par.	yaw	pitch	roll
k	0.075	0.025	0.072
ω_i	5.2	40	30
ω_d	17.66	300	10000
Reference Filtering			
par.	yaw	pitch	roll
ω_y	10000	10000	10000
Feed-Forward Gain			
par.	yaw	pitch	roll
m	0	0.15	0.1
Reference Weighting			
par.	yaw	pitch	roll
b	0.75	1	1
c	1	1	1
D Component Filtering			
par.	yaw	pitch	roll
N	15	15	15
Output Restrictors			
par.	yaw	pitch	roll
I_{low}	-0.25	-0.5	-0.5
I_{high}	0.25	0.5	0.5
D_{low}	-0.32	-0.5	-0.5
D_{high}	0.32	0.5	0.5
Out_{low}	-0.5	-0.5	-0.5
Out_{high}	0.5	0.5	0.5

Table 7.1: Angular Rate Control Layer PID controller settings

feed-forward branch into the controller. The flight performance of a properly set roll rate controller can be seen in Figure 7.14, and the final parameters are noted in table 7.1.

7.3.5 Roll and Pitch Identification Experiment

When trying to setup the pitch and roll rate controllers, an identification experiment was carried out in order to determine the natural frequencies and critical gains of the vehicle in both axes. Instead of a P controller, a bang-bang controller was used for that purpose. By the term “bang-bang” a two-state controller is meant, actuating either fully to one side or the other, according the control error; an infinite-gain P controller would work analogically.

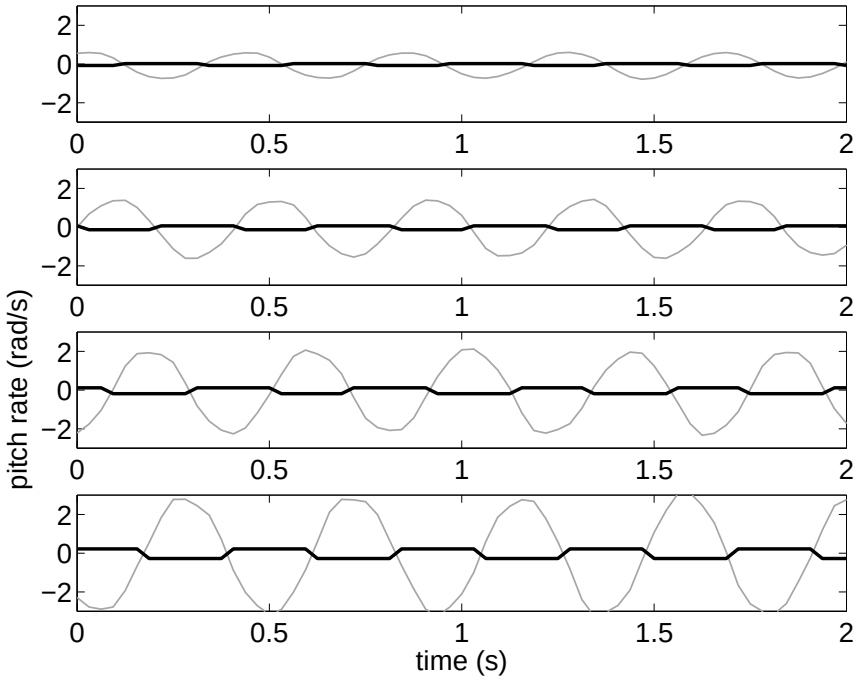


Figure 7.15: Pitch rate bang-bang controller in-flight performance. Maximum actuator throw (black line) is restricted to 0.1, 0.2, 0.3 and 0.4 of the maximal throw in the figures going from up to down. The gray line is the measured angular rate.

The gain (i. e. the allowed actuator throw) of the bang-bang controller was gradually increased and the response of the vehicle was measured. The results can be seen in Figures 7.15 and 7.16. The identification experiment was successful, or at least seemed to be, but right in the next flight the testing vehicle crashed badly due to the mechanical failure of the powertrain. The engine revved up suddenly, but the rotor lost all power and slowed down. The pilot contributed to the failure by not engaging the auto-rotation soon enough and let the momentum accumulated in the rotor to dissipate. The crash was then inevitable and destroyed most of the vehicle, badly damaging the avionics in the process. It was later determined that a needle one-way bearing in the transmission cracked and failed, causing the sudden loss of power. The root cause was traced back to the identification experiments.

The one-way bearing is situated inside the main gear, transmitting the engine torque to the main rotor shaft (see Figure 7.17). Therefore, all forces and moments, induced by the main rotor to the fuselage, are transferred through this bearing. When the identification experiments, described above, were performed, the bearing was overstressed as a result of relatively long-term high-frequency periodic oscillations, induced by the bang-bang controller. Also, the fuselage is much heavier and therefore

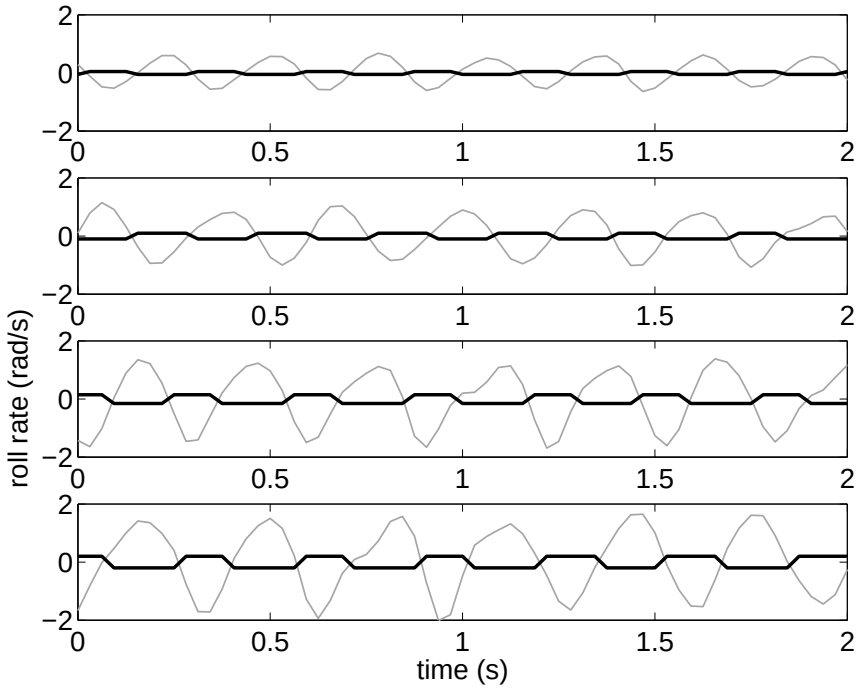


Figure 7.16: Roll rate bang-bang controller in-flight performance. Maximum actuator throw (black line) is restricted to 0.1, 0.2, 0.3 and 0.4 of the maximal throw in the figures going from up to down. The gray line is the measured angular rate.

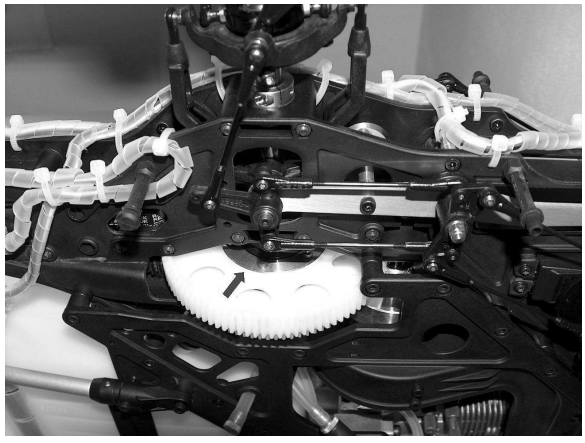


Figure 7.17: The improved metal housing of the one-way bearing is clearly visible in this picture (pointed by an arrow)

has larger moments of inertia than what the bearing was designed for by the vehicle manufacturer, because of additional weight of the on-board avionics. The housing of the bearing is made of plastics and can flex, not protecting the bearing sufficiently. This resulted into particularly high rate of attrition for the bearing, causing it to fail eventually.

The problem was fixed by introducing a non-flexible metal housing for the bearing, protecting it much better than the original plastic housing, as can be seen in Figure 7.17.

Chapter 8

Conclusions

8.1 Summary and Contributions

RAMA, the control system for Unmanned Aerial Vehicles (UAVs), was presented in this thesis. Its design philosophy, construction and implementation were shown, along with long-term operation experience and lessons learned during the project. The abilities and functions of the system were demonstrated on real-life flight data.

The goals, which were set in the beginning of the project, were completely fulfilled:

- A Rotorcraft-UAV was built and the control system was designed and developed.
- It is now operational and tested in the course of several hundred flights, conducted since 2005. All major hardware components are in place and the software framework is implemented, although this does not implicate that the system is perfect and won't be developed in the future (see section 8.2).
- The control scheme was designed and the basic control laws were implemented and tested, as was shown in chapters 4 and 7. The flight tests were conducted, data were acquired and the feasibility of proposed solutions was demonstrated (chapter 7).
- Last but not least, the whole project is now well documented through the project website [Špinka(2007)].

Main contributions and originality of this work were already mentioned in section 1.2. The original system architecture has shown its assets and shortcomings, as well as the safety features and the control algorithms. Vast pool of flight data and relevant technical information are now accessible to everyone through the project website

[Špinka(2007)]. The results of the project were also presented at numerous international conferences and a prestigious journal paper [Špinka et al.(2009a)].

The system had undergone several major redesigns and overhauls since its first incarnation and further improvements are waiting in the pipeline, as is shown in the next section.

8.2 Future Research

There are many areas that require further research and improvements. On the hardware side, the Inertial Measurement Unit should be replaced and the sampling and working rate of the system increased, as was mentioned in sections 3.3.2 and 4.4. A barometric altimeter will also be added, along with an ultrasound ground proximity sensor.

The attitude determination algorithm (section 4.2) will be soon improved by incorporating a Kalman filter, which is currently being finalized, and ultimately improved to provide not only attitude, but complete navigation data. The Attitude Control Layer (section 4.4.2) would be the next task in order and shall be operational in the near future. The higher control layers will follow.

It is clear that there is still a lot of work to be done on the project, and surely many other tasks, not yet foreseen, will emerge in the future too, as they always do.

Appendices

A.1 Contents of Enclosed DVD

Disclaimer - this DVD contains the technical documentation, flight data and photo gallery for the RAMA project, in the state it was when this thesis was finished. For latest versions and information, please visit the project website http://rtime.felk.cvut.cz/helicopter.

list of directories:

- thesis - directory contains this thesis in pdf format
- hw - hardware schematics, blueprints and pcb designs
- sw - software source codes
- flight_data - flight data, along with the vizualization scripts for Matlab
- gallery - photo and video gallery (contains only selected photos and videos from the RAMA archive)

A.2 List of the Telemetry Messages

Table A.1: Definitions of the Telemetry Messages

ID	Meaning	Contents
Asynchronous Messages		
0x02	IMU error	**uint8** error code
0x04	GPS error	**uint8** error code
0x05	SCU error	**uint8** error code
0x10	control mode	**uint8** mode (0x00-MCM 0xFF-ACM)
0x30	data composition error	**uint32** error code
0x64	DAM unexpected reset	**uint8** number of occurrences
Synchronous Data Messages		
0x35	all data sampled @64 Hz	**struct eth_data_msg_type** data
0x40	GPS data sampled @1 Hz	**struct cpo_pvt_data_type** gps_data

```
typedef struct {
   uint32_t sample_num;
   uint8_t servo_mask;
   uint8_t reserve_1;
   int16_t throttle_stick_pos;
   int16_t roll_stick_pos;
   int16_t pitch_stick_pos;
   int16_t gyro_stick_pos;
   int16_t collective_stick_pos;
   int16_t yaw_stick_pos;
   int16_t reserve_2;
   float des_roll;
   float des_pitch;
   float des_yaw;
   int16_t throttle_servo_pos;
   int16_t roll_servo_pos;
   int16_t pitch_servo_pos;
   int16_t yaw_servo_pos;
   int16_t collective_servo_pos;
   int16_t reserve_3;
   float params_roll_rate_Xi;
   float params_roll_rate_Xdr;
   float params_roll_rate_Xdy;
   float params_pitch_rate_Xi;
   float params_pitch_rate_Xdr;
```

```
   float params_pitch_rate_Xdy;
   float params_yaw_rate_Xi;
   float params_yaw_rate_Xdr;
   float params_yaw_rate_Xdy;
   float batt_voltage_actuator;
   float batt_voltage_avionic;
   float roll_angle;
   float pitch_angle;
   float yaw_angle;
   float angular_rate_roll_filtered;
   float angular_rate_pitch_filtered;
   float angular_rate_yaw_filtered;
   float angular_rate_roll;
   float angular_rate_pitch;
   float angular_rate_yaw;
   float accel_x_filtered;
   float accel_y_filtered;
   float accel_z_filtered;
   float accel_x;
   float accel_y;
   float accel_z;
   float mfi_x_filtered;
   float mfi_y_filtered;
   float mfi_z_filtered;
   float mfi_x;
   float mfi_y;
   float mfi_z;
} __attribute__ ( ( packed ) ) eth_data_msg_type;
```

```
typedef struct {
   float alt;
   float epe;
   float eph;
   float epv;
   int16_t fix;
   double gps_tow;
   double lat;
   double lon;
   float lon_vel;
   float lat_vel;
   float alt_vel;
   float msl_hght;
   int16_t leap_sec;
   int32_t grmn_days;
} __attribute__ ( ( packed ) ) cpo_pvt_data_type;
```

A.3 List of the Vehicle Bus Messages

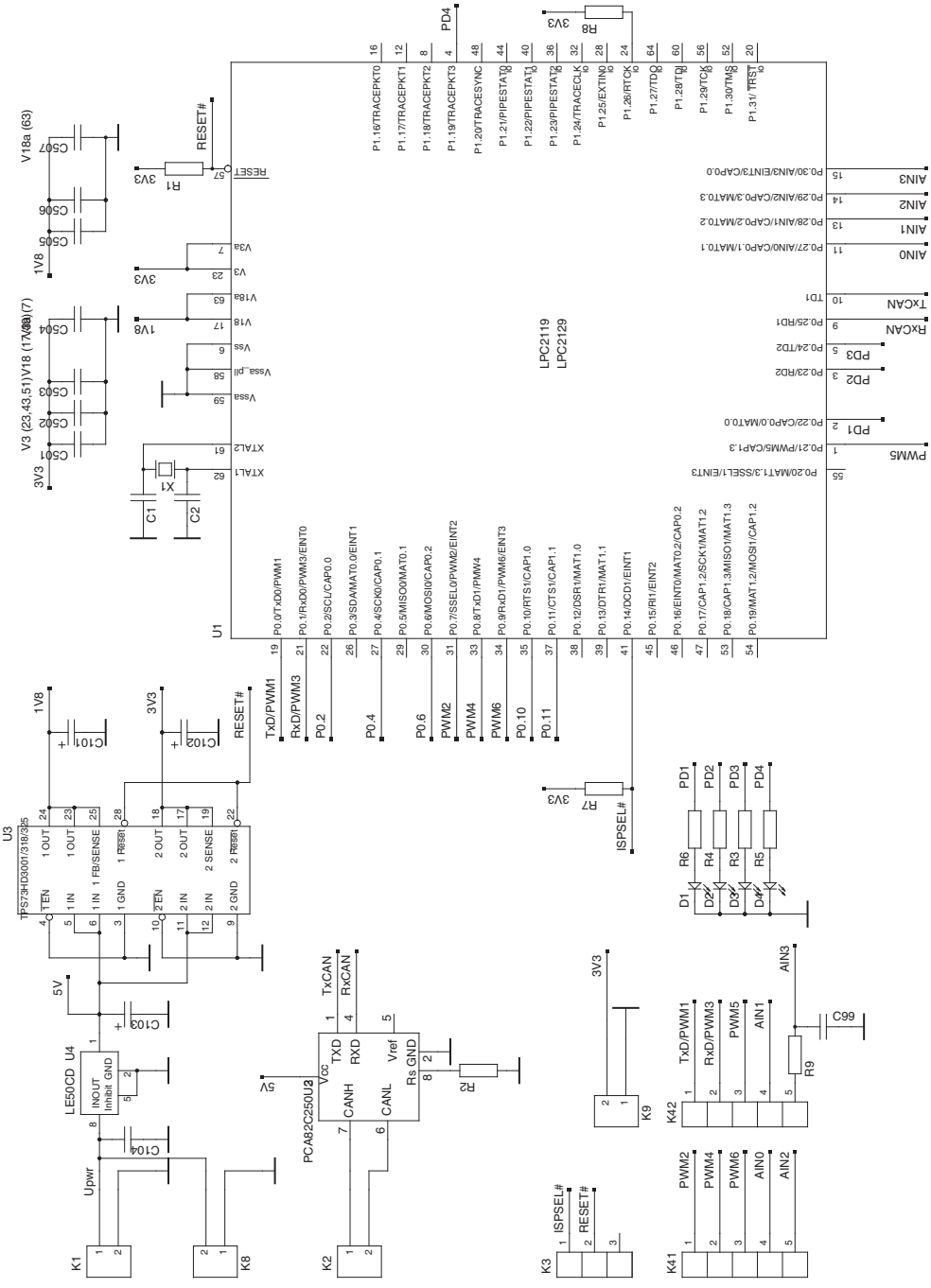
Table A.2: Definitions of the Vehicle Bus Messages

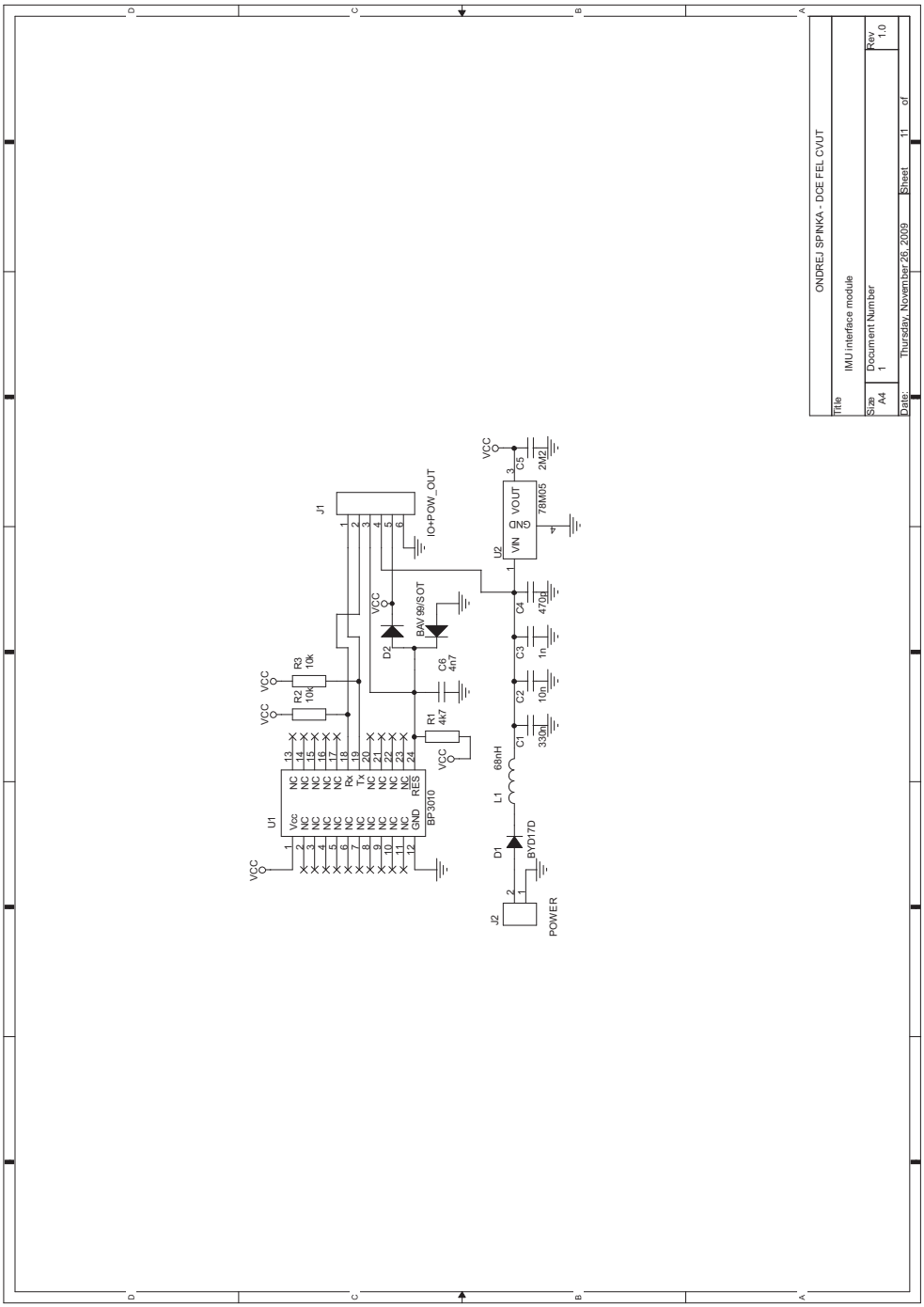
ID	Meaning	Contents
DAM Status Messages - out		
0x0002	IMU error	**uint8** error code
0x0004	GPS error	**uint8** error code
0x0064	DAM boot-up	no data
DAM Data Messages - out		
0x0014	TSM Message	no data
0x0020	x magnetic intensity	**float** raw data **float** filtered (miligauss)
0x0040	y magnetic intensity	**float** raw data **float** filtered (miligauss)
0x0060	z magnetic intensity	**float** raw data **float** filtered (miligauss)
0x0056	x acceleration	**float** raw data **float** filtered (milig)
0x0057	y acceleration	**float** raw data **float** filtered (milig)
0x0058	z acceleration	**float** raw data **float** filtered (milig)
0x00C4	angular rate roll	**float** raw data **float** filtered (rad/s)
0x01C4	angular rate pitch	**float** raw data **float** filtered (rad/s)
0x03C4	angular rate yaw	**float** raw data **float** filtered (rad/s)
0x0400	GPS raw data 1	**uint64** raw data
0x0200	GPS raw data 2	**uint64** raw data
0x0180	GPS raw data 3	**uint64** raw data
0x0100	GPS raw data 4	**uint64** raw data
0x0300	GPS raw data 5	**uint64** raw data
0x0500	GPS raw data 6	**uint64** raw data
0x0700	GPS raw data 7	**uint64** raw data
0x0090	GPS raw data 8	**uint64** raw data
DAM Command Messages - in		
0x0059	calibration sample number	**uint16** number
0x006E	DAM reset request	no data
0x006F	DAM run request	no data
0x0070	roll rate filter a coef.	**float** coefficient
0x0071	roll rate filter b coef.	**float** coefficient
0x0072	pitch rate filter a coef.	**float** coefficient
0x0073	pitch rate filter b coef.	**float** coefficient
0x0074	yaw rate filter a coef.	**float** coefficient
0x0075	yaw rate filter b coef.	**float** coefficient
0x0076	accel x filter a coef.	**float** coefficient
0x0077	accel x rate filter b coef.	**float** coefficient

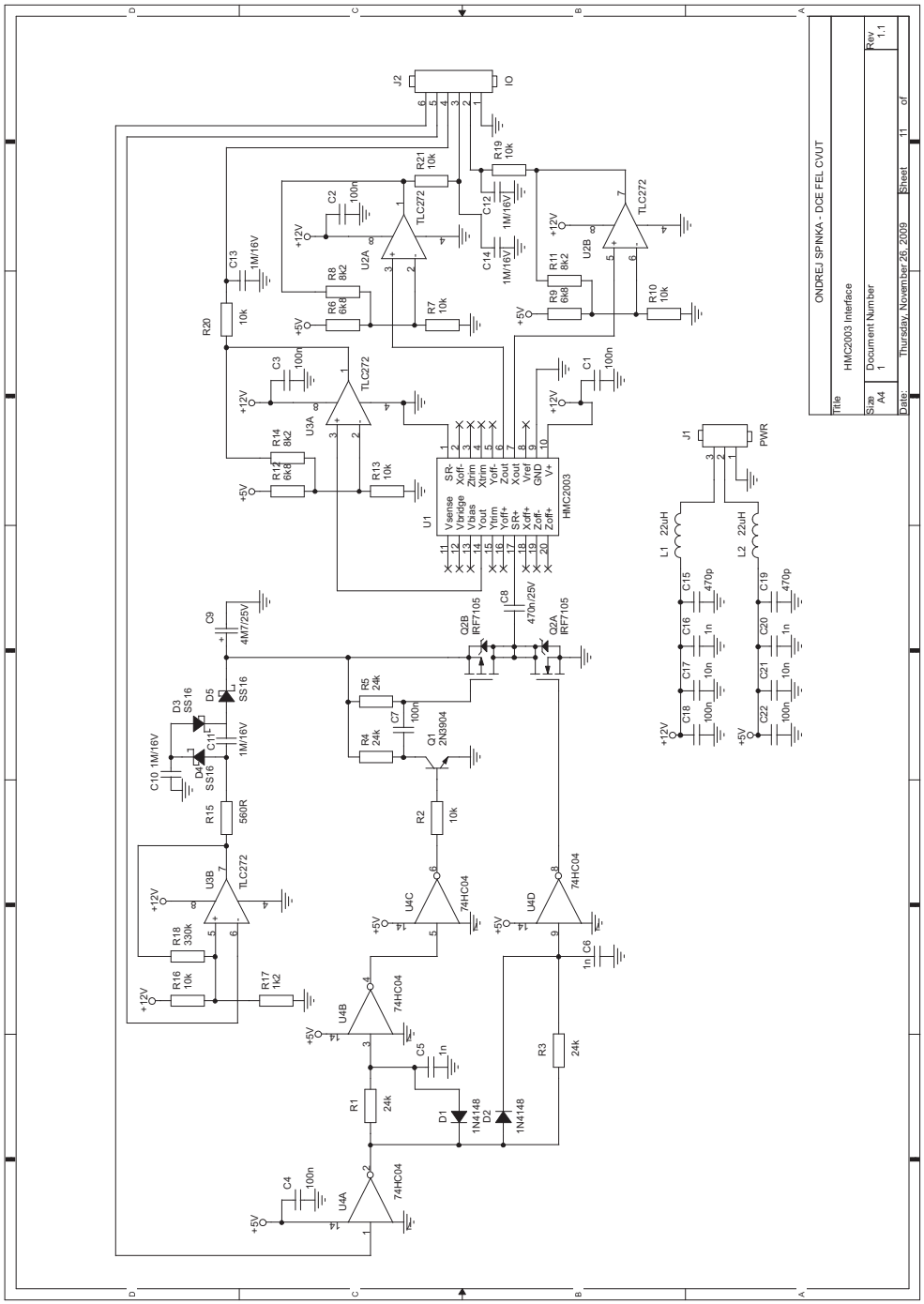
continued on the next page

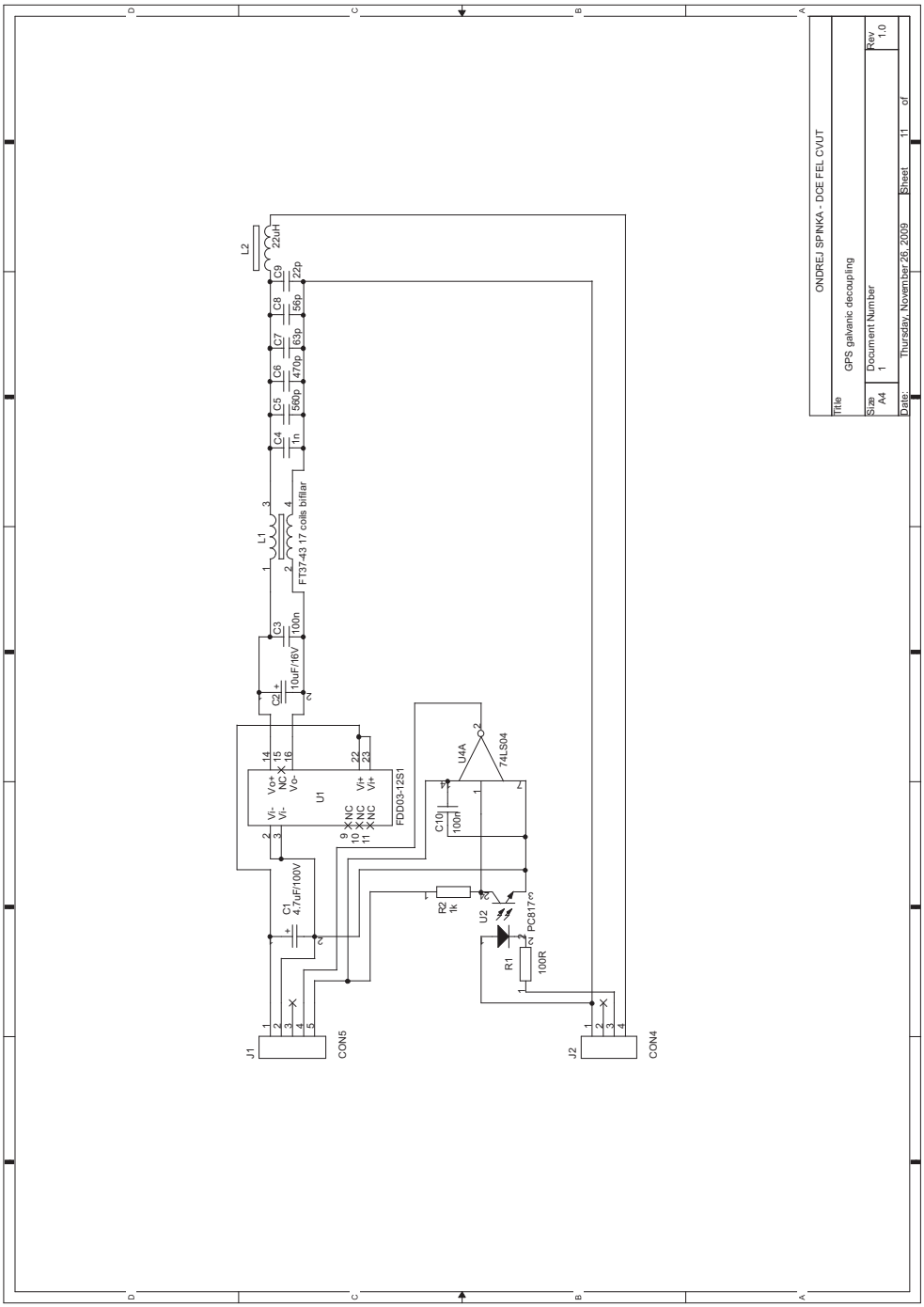
Table A.2 – continued from previous page		
ID	**Meaning**	**Contents**
0x0078	accel y filter a coef.	**float** coefficient
0x0079	accel y rate filter b coef.	**float** coefficient
0x007A	accel z filter a coef.	**float** coefficient
0x007B	accel z rate filter b coef.	**float** coefficient
0x007C	TAM x filter a coef.	**float** coefficient
0x007D	TAM x rate filter b coef.	**float** coefficient
0x007E	TAM y filter a coef.	**float** coefficient
0x007F	TAM y rate filter b coef.	**float** coefficient
0x0080	TAM z filter a coef.	**float** coefficient
0x0081	TAM z rate filter b coef.	**float** coefficient
0x0082	TAM x offset	**float** offset
0x0083	TAM y offset	**float** offset
0x0084	TAM z offset	**float** offset
SCU Status Messages - out		
0x0005	SCU error	**uint8** error code
0x000A	ACM engaged	no data
0x000B	MCM engaged	no data
SCU Data Messages - out		
0x0023	stick positions 1	**uint8** servo mask **uint8** reserved **uint16** throttle **uint16** roll **uint16** pitch
0x0024	stick positions 2	**uint16** gyro **uint16** collective **uint16** yaw **uint16** reserved
0x0032	battery voltages	**uint8** ACB voltage **uint8** AVB voltage
SCU Command Messages - in		
0x0019	servo positions 1	**uint8** servo mask **uint8** reserved **uint16** throttle **uint16** roll **uint16** pitch
0x001A	servo positions 2	**uint16** gyro **uint16** collective **uint16** yaw **uint16** reserved

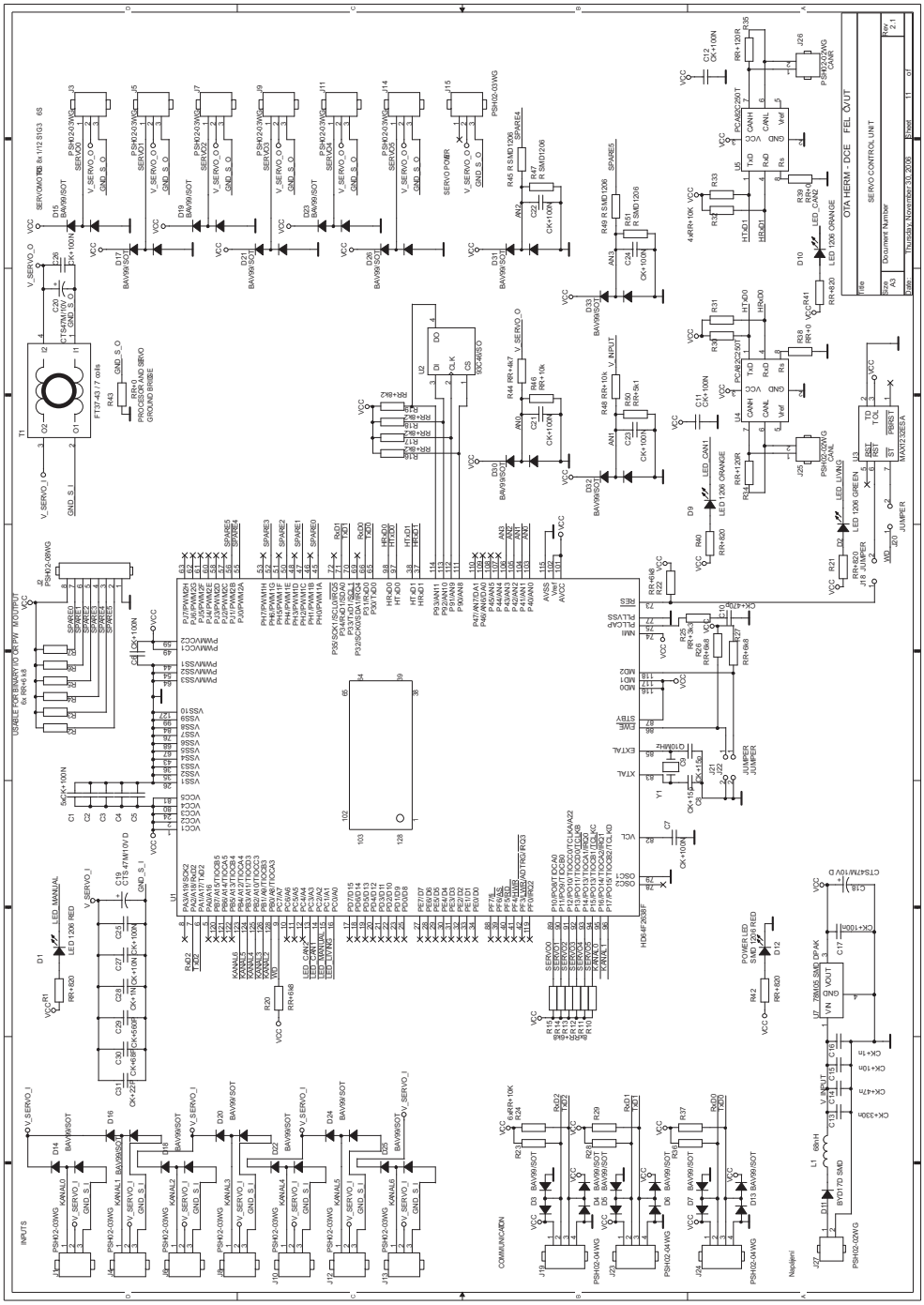
A.4 Schematics of Control System Modules


U1
LPC2119
LPC2129
P0.0/TxD0/PWM1
P0.1/RxD0/PWM3/EINT0
P0.2/SCL/CAP0.0
P0.3/SDA/MAT0.0/EINT1
P0.4/SCK0/CAP0.1
P0.5/MISO0/MAT0.1
P0.6/MOSI0/CAP0.2
P0.7/SSEL0/PWM2/EINT2
P0.8/TxD1/PWM4
P0.9/RxD1/PWM6/EINT3
P0.10/RTS1/CAP1.0
P0.11/CTS1/CAP1.1
P0.12/DSR1/MAT1.0
P0.13/DTR1/MAT1.1
P0.14/DCD1/EINT1
P0.15/RI1/EINT2
P0.16/EINT0/MAT0.2/CAP0.2
P0.17/CAP1.2/SCK1/MAT1.2
P0.18/CAP1.3/MISO1/MAT1.3
P0.19/MAT1.2/MOSI1/CAP1.2
P0.20/MAT1.3/SSEL1/EINT3
P0.21/PWM5/CAP1.3
P0.22/CAP0.0/MAT0.0
P0.23/RD2
P0.24/TD2
P0.25/RD1
P0.27/AIN0/CAP0.1/MAT0.1
P0.28/AIN1/CAP0.2/MAT0.2
P0.29/AIN2/CAP0.3/MAT0.3
P0.30/AIN3/EINT3/CAP0.0
P1.16/TRACEPKT0
P1.17/TRACEPKT1
P1.18/TRACEPKT2
P1.19/TRACEPKT3
P1.20/TRACESYNC
P1.21/PIPESTAT0
P1.22/PIPESTAT1
P1.23/PIPESTAT2
P1.24/TRACECLK
P1.25/EXTIN0
P1.26/RTCK
P1.27/TDO
P1.28/TDI
P1.29/TCK
P1.30/TMS
P1.31/TRST
RESET
XTAL1
XTAL2
Vss
Vss_PLL
V18a
V3
V3a
RESET#
PD4
AIN3
AIN2
AIN1
AIN0
TXCAN
RXCAN
PD3
PD2
PD1
PWM5
R8
R1
X1
C1
C2
C501
C502
C503
C504
C505
C506
C507
V18a (63)
V3 (23,43,51)
V18 (17,39)
1V8
3V3
U3
TPS73HD3001/3018/3025
1 EN
1 IN
1 FB/SENSE
1 GND
1 OUT
1 RESET
2 EN
2 IN
2 SENSE
2 GND
2 OUT
2 RESET
C101
C102
C103
C104
5V
Upwr
LE50CD
U4
INOUT
Inhibit
GND
K1
K8
PCA82C250U3
Vcc
TXD
RXD
CANH
CANL
Vref
Rs
GND
TxCAN
RxCAN
R2
K2
K3
ISPSEL#
RESET#
R7
K9
TxD/PWM1
RxD/PWM3
PWM5
AIN1
AIN3
R9
C99
K42
PWM2
PWM4
PWM6
AIN0
AIN2
K41
D1
D2
D3
D4
R6
R4
R3
R5


ONDREJ SPINKA - DCE FEL CVUT
IMU interface module
Rev 1.0
Size A4
Document Number 1
Date: Thursday, November 26, 2009
Sheet 11 of
J1
IO+POW_OUT
U1
BP3010
U2
78M05
VIN
VOUT
GND
J2
POWER
D1
BYD17D
D2
BAV99/SOT
L1
68nH
C1
330n
C2
10n
C3
1n
C4
470n
C5
2M2
C6
4n7
R1
4k7
R2
10k
R3
10k
VCC
RES
GND


ONDREJ SPINKA - DCE FEL CVUT
Title
HMC2003 Interface
Size
A4
Document Number
1
Rev
1.1
Date:
Thursday, November 26, 2009
Sheet
11
of
J2
IO
J1
PWR
U1
HMC2003
U2A
U2B
U3A
U3B
U4A
U4B
U4C
U4D
TLC272
74HC04
Q1
2N3904
Q2A
IRF7105
Q2B
IRF7105
D1
1N4148
D2
1N4148
D3
SS16
D4
SS16
D5
SS16
L1
22uH
L2
22uH
+12V
+5V


ONDREJ SPINKA - DCE FEL CVUT
Title
GPS galvanic decoupling
Size
A4
Document Number
1
Rev
1.0
Date:
Thursday, November 26, 2009
Sheet
11
of
L2
22uH
C9
22p
C8
56p
C7
63p
C6
470p
C5
560p
C4
1n
L1
FT37-43 17 coils bifilar
C3
100n
C2+
10uF/16V
U1
FDD03-12S1
U4A
74LS04
C10
100nf
C1
4.7uF/100V
R2
1k
U2
PC817
R1
100R
J1
CON5
J2
CON4



Bibliography

[1] J. Cham, "The Ph.D. Comics." http://www.phdcomics.com.

[2] "The yamaha RMAX UAV website." http://www.yamaha-motor.co.jp/global/industrial/sky/solution/rmax.

[3] "Paparazzi - The free autopilot." http://paparazzi.enac.fr.

[4] B. Mettler, *Identification, Modeling and Characteristics of Miniature Rotorcraft*, vol. ISBN 1-4020-7228-7. Dordtrecht, The Neteherlands: Kluver Academic Publishers, 2003.

[5] B. Mettler, M. B. Tischler, and T. Kanade, "System identification modeling of a small-scale unmanned helicopter," *Journal of the American Helicopter Society*, October 2001.

[6] V. Gavrilets, B. Mettler, and E. Feron, "Nonlinear dynamic model of a small-size acrobatic helicopter," in *Proc. of the AIAA Conf. On Guidance, Navigation and Control, Montreal, Canada*, 2001.

[7] A. Bogdanov, E. Wan, and G. Harvey, "SDRE flight control for X-Cell and R-Max autonomous helicopters," in *Proceedings of the 43rd IEEE Conference on Decision and Control*, December 2004.

[8] M. F. Weilenmann and H. P. Geering, "A test bench for rotorcraft hover control," *Journal of Guidance, Control and Dynamics*, vol. 17, pp. 729–736, 1994.

[9] J. Chapuis, C. Eck, M. Kottmann, M. Sanvido, and O. Tanner, "Control of Helicopters," *Control of Complex Systems*, pp. 359–392, 2001.

[10] M. Sanvido, "A Computer System for Model Helicopter Flight Control," tech. rep., Technical Memo Nr 3: The Software Core. Technical Report, Department of Computer Science, ETHZ., 1999.

[11] G. Buskey, J. M. Roberts, and G. Wyeth, "Autonomous helicopter hover using an artificial neural network," in *IEEE International Conference on Robotics and Automation*, pp. 1635–1640, May 2001.

[12] G. Buskey, J. M. Roberts, and G. Wyeth, "Online learning of autonomous helicopter control," in *IEEE International Conference on Robotics and Automation*, 2001.

[13] D. H. Shim, H. J. Kim, and S. Sastry, "A Flight Control System for Aerial Robots: Algorithms and Experiments," *IFAC Control Engineering Practice*, 2003.

[14] B. Horowitz, J. Liebman, C. Ma, T. J. Koo, A. Sangiovanni-Vincentelli, and S. Sastry, "Platform-based embedded software design and system integration for autonomous vehicles," in *IEEE Proceedings*, vol. 91, 2003.

[15] O. A. Rawashdeh and J. E. Lumpp, "A Technique for Specifying Dynamically Reconfigurable Embedded Systems," in *Proceedings of the IEEE Aerospace Conference*, vol. ISBN: 0-7803-8870-4, pp. 1–11, 2005.

[16] D. Seto, B. Krogh, L. Sha, and A. Chutinan, "Dynamic control system upgrade using the simplex architecture," *IEEE Control Systems*, vol. 18(4), pp. 72–80, 1998.

[17] E. A. Strunk and J. C. Knight, "Dependability Through Assured Reconfiguration in Embedded System Software," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 172–187, 2006.

[18] E. A. Strunk, J. C. Knight, and M. A. Aiello, "Distributed Reconfigurable Avionics Architectures," in *Digital Avionics Systems Conference*, vol. 2, 2004.

[19] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real-Time Scheduling Theory: A Historical Perspective," *Real-Time Systems*, vol. 28, no. 2-3, pp. 101–155, 2004.

[20] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, "Elastic Scheduling for Flexible Workload Management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289–302, 2002.

[21] P. Balbastre, I. Ripoll, and A. Crespo, "Minimum Deadline Calculation for Periodic Real-Time Tasks in Dynamic Priority Systems," *IEEE Transactions on Computers*, vol. 57, no. 1, pp. 96–109, 2008.

[22] P. Caspi, A. Sangiovanni-Vincentelli, L. Almeida, A. Benveniste, B. Bouyssounouse, G. Buttazzo, I. Crnkovic, W. Damm, J. Engblom, G. Folher, M. Garcia-Valls, H. Kopetz, Y. Lakhnech, F. Laroussinie, L. Lavagno, G. Lipari, F. Maraninchi, P. Peti, J. de la Puente, N. Scaife, J. Sifakis, R. de Simone, M. Torngren, P. Verissimo, A. J. Wellings, R. Wilhelm, T. A. C. Willemse, and W. Yi, "Guidelines for a graduate curriculum on embedded software and

systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 4, no. 3, pp. 587–611, 2005.

[23] N. S. Kumar and T. Jann, "Estimation of attitudes from a low-cost miniaturized inertial platform using Kalman Filter-based sensor fusion algorithm," *Sadhana Journal*, vol. 29, no. 2, 2004.

[24] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, "An Integrated GPS/MEMS-IMU Navigation System for an Autonomous Helicopter," *Aerospace Science and Technology*, vol. 10, no. 6, pp. 527–533, 2006.

[25] J. H. Kim, S. Sukkarieh, and S. Wishart, "Real-Time Navigation, Guidance, and Control of a UAV Using Low-Cost Sensors," *Springer Tracts in Advanced Robotics*, vol. 24, no. 7, pp. 299–309, 2006.

[26] Laboratory for Autonomous Flying Robots - TU Berlin, "The MARVIN UAV - project website." http://pdv.cs.tu-berlin.de/MARVIN.

[27] R. Jíra and J. Ticháček, *Aerodynamika a mechanika letu*. Naše vojsko, 1960.

[28] C. Mill, "Practical Theories." http://www.w3mh.co.uk.

[29] L. Euler, "De Motu Corporum Circa Punctum Fixum Mobilium," *Commentatio 825 indicis ENESTROEMIANI, Opera posthuma*, vol. 2, pp. 42–62, 1862.

[30] F. Markley and M. Shuster, "A New Angle on the Euler Angles," in *Flight Mechanics (Estimation Theory Symposium)*, pp. 395–403, 1995.

[31] D. Hoag, "Apollo Guidance and Navigation - Considerations of Apollo IMU Gimbal Lock - MIT Instrumentation Laboratory Document E-1344," tech. rep., Massachusetts Institute of Technology, 1963.

[32] E. M. Jones and P. Fjeld, "Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas," tech. rep., National Air and Space Agency (NASA), 2006.

[33] A. R. S. Bramwell, *Bramwell's Helicopter Dynamics*. AIAA, 2001.

[34] W. R. Hamilton, "On quaternions, or on a new system of imaginaries in algebra," *Philosophical Magazine*, vol. 25, no. 3, pp. 489–495, 1844.

[35] A. B. Chambers and D. C. Nagel, "Pilots of the future: human or computer?," *Communications of the ACM*, vol. 28, no. 11, pp. 1187–1199, 1985.

[36] C. E. Billings, "Toward a Human-Centered Aircraft Automation Philosophy," *International Journal of Aviation Psychology*, vol. 1, no. 4, pp. 261–270, 1991.

[37] K. Etschberger, *Controller Area Network*. IXXAT Press, 2001.

[38] “The analogue & Micro company BOA 5200 website.” http://www.analogue-micro.com/BOA5200.html.

[39] M. Peca, “Walking robot spejbl - project website.” http://rtime.felk.cvut.cz/ pecam1/spejbl.

[40] M. Peca, M. Sojka, and Z. Hanzálek, “Spejbl - The Biped Walking Robot,” in *Preprints of 7th IFAC International Conference on Fieldbuses and nETworks in industrial and embedded systems, Toulouse, France*, pp. 63–70, 2007.

[41] M. Peca, “Kráčející robot Spejbl - Diplomová práce,” tech. rep., CTU FEL DCE, Prague, 2008.

[42] “The BEC-NAV company website.” http://http://www.becnav.co.uk.

[43] “The Analog Devices ADIS 16350 IMU website.” http://www.analog.com/en/mems/imu/adis16350/products/product.html.

[44] “The Garmin company website.” http://www.garmin.com.

[45] “The Honeywell Aerospace company website.” http://www.honeywell.com.

[46] O. Herm, “Návrh a realizace řídicí jednotky servomotorů pro model vrtulníku - Diplomová práce,” tech. rep., CTU FEL DCE, Prague, 2006.

[47] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transaction of the ASME - Journal of basic Engineering*, pp. 35–45, 1960.

[48] P. Corke, “A Robotics Toolbox for MATLAB,” *IEEE Robotics and Automation Magazine*, vol. 3, pp. 24–32, Mar. 1996.

[49] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems*, vol. ISBN 0-13-314899-8. NJ, USA: Prentice Hall, 3rd edition ed., 1997.

[50] “The SocketCAN poject website.” http://developer.berlios.de/projects/socketcan.

[51] P. Svoboda, “Zpracování a vizualizace letových dat pro model vrtulníku - Diplomová práce,” tech. rep., CTU FEL DCE, Prague, 2009.

[52] D. Day, “Moving swashplates and CCPM.” http://www.iroquois.free-online.co.uk/ccpm.htm.

[53] J. G. Ziegler and N. B. Nichols, “Optimum setting for PID controllers,” *Transactions of ASME*, vol. 64, pp. 759–768, 1942.

List of Publications

[Špinka et al.(2009a)] O. Špinka, O. Holub, and Z. Hanzálek. Low-Cost Reconfigurable Control System for Small UAVs (**co-authorship 70% - accepted paper**). *IEEE Transactions on Industrial Electronics, (**IF 5.468**)*, 56, 2009a.

[Špinka et al.(2009b)] O. Špinka, J. Åkesson, Z. Hanzálek, and K. E. Årzén. Open Physical Models in Control Engineering Education (**co-authorship 45% - paper in review**). *International Journal of Electrical Engineering Education, Manchester University Press, (**IF 0.118**)*, 2009b.

[Špinka(2008)] O. Špinka. Low-Cost Reconfigurable Control System for Small UAVs (**authorship 100%**). In *1. Národní konference Civilní bezpilotní systémy, Prague, Czech Republic*, 2008.

[Špinka et al.(2007a)] O. Špinka, Š. Kroupa, and Z. Hanzálek. Control System for Unmanned Aerial Vehicles (**co-authorship 75%**). In *Kolokvium CAK 2007*, 2007a.

[Špinka(2007)] O. Špinka. RAMA UAV Autopilot - Project website (**authorship 100%**). http://rtime.felk.cvut.cz/helicopter, 2007.

[Špinka et al.(2007b)] O. Špinka, Š. Kroupa, and Z. Hanzálek. Control System for Unmanned Aerial Vehicles (**co-authorship 65%**). In *5th IEEE International Conference on Industrial Informatics, Vienna, Austria*, volume ISBN 1-4244-0864-4, pages 455–460, 2007b.

[Špinka et al.(2006)] O. Špinka, J. Krákora, M. Sojka, and Z. Hanzálek. Low-cost avionics system for ultra-light aircraft (**co-authorship 50%**). In *11th IEEE International Conference on Emerging Technologies and Factory Automation Proceedings, Prague, Czech Republic*, pages 102–109, 2006.

[Špinka(2006)] O. Špinka. Design and Construction of an Automatic Unmanned Helicopter Model (**authorship 100%**). In *CTU Reports Proceedings of Workshop, Prague, Czech Republic*, volume ISBN 80-01-03439-9, 2006.

[Krákora et al.(2005)] J. Krákora, P. Píša, P. Smolík, O. Špinka, and J. Kelbel. Open Components for Embedded Real-Time Applications - OCERA (**co-authorship 5%**). In *LinuxExpo, Prague, Czech Republic*, 2005.

[Sojka and Špinka(2005)] M. Sojka and O. Špinka. ECS Deployment and Validation (**co-authorship 20%**). In *Proceedings of the Graduate Course on Embedded Control Systems, Valencia, Spain*, 2005.

[Špinka and Hanzálek(2005)] O. Špinka and Z. Hanzálek. Control system of an autonomous helicopter model (**co-authorship 90%**). In *Process Control, Štrbské Pleso, Slovakia*, pages 102–109, 2005.

[Špinka(2003)] O. Špinka. Aerodynamika a mechanika letu vrtulníku (**authorship 100%**). Technical report, CTU FEL DCE, Prague, 2003.

Curriculum Vitae

Ondřej Špinka received a degree in Electrical Engineering from the Czech Technical University (CTU) in Prague in 2004. From 2004 on, he is a Ph.D. student at the same university. He was involved in several research projects (IFIBO, OCERA, ARTIST) and worked as a lead designer in a project on integrated avionic system for ultra-light aircraft. From 2004 on, he works as a research fellow with the Center of Applied Cybernetics.

His teaching activities include a broad scope of courses. Currently, he lectures a course on programming for embedded systems, and is a leading teacher for this course. In the past, he was a leading teacher and a lecturer for a course on Aircraft avionic systems. He also teaches in a course on computer systems and programming in Linux. In the past, he taught courses on logical systems and industrial automation and a basic course on computer systems. Ondřej Špinka supervised a large number of student projects and theses; these include 10 diploma theses and 8 bachelor theses, many semestral projects and 2 student projects under the CEPOT (CEntrum Pro POdporu Talentů). He attended an ARTIST2 Graduate Course on Embedded Systems in Valencia, Spain in 2005, and taught one of the classes here.

His research interests include real-time embedded systems, Unmanned Aerial Vehicles, autopilots and vehicle control and guidance. His results were presented at multiple international conferences, including two prestigious IEEE conferences, and an IEEE journal (Impact Factor 5.468). Another paper is currently reviewed for publication in another international journal. One of his papers has a registered citation in the ISI database ([Špinka et al.(2007b)] - this paper was cited by Erdos D., Watkins S. E., UAV autopilot integration and testing. In *IEEE Region 5 Conference, Kansas City, USA, pages 94-99*, 2008). Ondřej Špinka also worked as a reviewer for multiple IEEE conferences and journals.