Control System for Unmanned Aerial Vehicles

Ondřej Špinka, Štěpán Kroupa, Zdeněk Hanzálek

Abstract— **An open project, dealing with autopilot design for autonomous Unmanned Aerial Vehicles is introduced in this paper. Networked hierarchical distributed control system is being proposed and its hardware and software structure is briefly described. Mathematical model of a small rotorcraft is presented and identification methodology and state estimation using Extended Kalman Filter are discussed. Control algorithms, based on PI, LQG and SDRE approaches, focused on rotorcraft UAVs are proposed, including a complex hierarchical autopilot design. Real data, measured during test-flights of an experimental UAV, are presented.**

I. Introduction

A. Motivation

UNMANNED aerial vehicles (UAVs) are becoming more and more popular in a wide field of applications nowadays. Although being developed mainly for military purposes in the past, it becomes obvious that there are a lot of other areas where they might prove useful. Consider agriculture for example, where they may be used for field observations or for chemicals distribution. They can patrol over wide forest areas as a fireguard, or they can be used for traffic observation in the cities. In cartography, small UAVs might be used for automatic landscape photographing, being much more cost-efficient compared with traditional aerial snapshooting. UAVs are also very interesting for the academic research, as they can be used for various purposes - as flying laboratories, proving ground for control algorithms, or as an education tools for students.

Therefore, there is a growing demand for UAV control systems, and many projects - either commercial or academic - destined to design a UAV autopilot were (and are) held recently. A lot of impressive results had already been achieved, and many UAVs, more or less autonomous, are used by various organizations. Sadly, they are still unattainable for many potential users, as stock products are very expensive, and academic researchers are often reluctant (understandably) to give away specific construction details of their inventions. Many teams had published their results, but (according our knowledge) none of them had released full technical documentation of their control systems for public usage. Hence, every newcomer to this field, unable to afford a commercial product, is compelled to start his project from scratch, again and again. The complexity of the whole design process may discourage many of them, and a lot of time and resources are wasted by re-inventing things already known to more experienced teams. That is why we designed RAMA.

RAMA stands for Remotely operated Aerial Model Autopilot. It is an open project, held at the Department of Control Engineering, Faculty of Electrical Engineering of the Czech Technical University in Prague, which purpose is to design a universal lightweight and compact control system for small UAVs. RAMA is intended to be used mainly in the academic environment, and shall serve as a good starting point for anyone willing to build his own UAV. Main distinctive feature of the RAMA project is that it is totally open, meaning that the whole technical documentation - including wiring diagrams, PCBs (Printed Circuit Boards), software source codes, vehicle controller designs, vehicle mathematical models and real flight data from our UAV - is available at the project website [24]. Therefore, anybody with necessary technical background should be able to build his own RAMA system for whatever purpose, or could use RAMA as a reference for his own project, saving a lot of work he (she) would otherwise spend developing his (her) own system.

From engineering point of view, UAV autopilot design poses a lot of challenges. It is very complex multidisciplinary process, covering disciplines from hardware design, sensors and measurement, programming, networking, etc. to mathematic modeling and control theory, artificial intelligence, image and signal processing etc. Therefore it is very interesting for researchers from various fields, and there is still plenty of room for improvements and new approaches, as this field is relatively fresh and unexplored.

The RAMA project had been running since 2004 and had already brought some interesting results. From hardware design and system programming point of view, it is almost complete by now. Currently, extensive work is being carried out on the mathematical modeling and controller designs. A couple of stabilizing controllers had already been introduced and tested with promising results, and dozens of test flights were performed. The project is comprehensively documented at the project website [24].

B. Related work

A lot of work had already been done in the UAV field by other teams. Most interesting works in the academic environment were proposed by following teams: H. P. Geering and M. F. Weilenmann from ETH Zurich presented a stabilizing controller for a rotorcraft in [14]. Another Swiss team followed in their footsteps with considerable success [15]. At the University of California in Berkeley a military UAV rotorcraft was developed in the course of the DARPA project [16]. Their vehicles are even able to perform coordinated group flights. At the MIT (Massachusetts Institute of Technology), V. Gavrilets, B. Mettler and E. Feron developed a complex non-linear mini-helicopter mathematic model and verified it successfully using two small rotorcrafts [6]. A. Bogdanov, E. Wan and G. Harvey from the OGI School of science and engineering at Oregon later used this model to develop a promising SDRE (State-Dependent Riccati Equation) governor

[5]. An interesting work dealing with dynamics identification of a small helicopter using frequency domain methods was published by B. Mettler [4]. A group led by G. Buskey investigated the possibilities of "artificial intelligence" control methods, such as neural networks, fuzzy and neuro-fuzzy controllers (see [17] for example). Another interesting R-UAV (Rotorcraft-UAV) is being developed at the TU Berlin [18]. Advanced mathematical models and control methods were presented by a team from the Aalborg University [2] [3] [7] [8] [9].

From non-academic projects, let us at least mention the Yamaha RMAX R-UAV [19], probably the most advanced commercial UAV currently available at the market. It is a completely autonomous R-UAV, used widely all around the world in such projects as environmental observations, atomic plant surveillance, infrastructure maintenance etc.

C. Paper organization

This paper is organized as follows: In the following chapter, basic structure of the RAMA control system is presented. Its distinctive feature compared to most other projects known to us [8] [14] [15] [17] [18] is the hierarchical networked architecture we are introducing, bringing undisputable benefits (see II-B.1). In chapter II-C, the software architecture is very briefly described. Its main asset is very general and easily extendible structure, as well as the possibility of rapid software upgrades without the need to physically connect to the system. The chapter III introduces basic mathematical models we use, and controller design is described in the chapter IV, introducing our complex hierarchical autopilot, utilizing PI, LQG and SDRE control methods. Results achieved during the flight tests are discussed in V, followed by the conclusion.

II. Control System Architecture

A. Flying Platform

Fig. 1. Hirobo Freya helicopter with the RAMA control system

Hirobo Freya 60 RC helicopter (Figure 1) [23] is being used as the flying platform for the RAMA system. Freya is a two-blade machine with Bell-Hiller stabilizer, incorporating H1 type of swash plate management. Powered by O.S. Max SX-61 Heli two-stroke engine and with carbon-fiber 680mm FAI symmetrical blades, it can bear about 2 kilograms of payload. The overall weight of the machine (including avionics and fuel) is approx. 7kg, main rotor diameter 1560mm, length 1375mm, width 200mm and height 543mm.

B. Control System Hardware

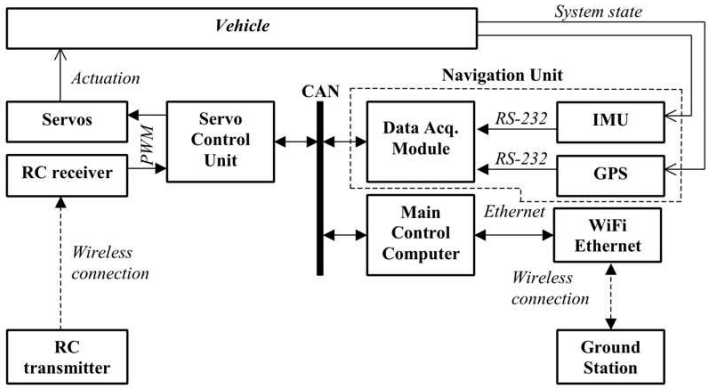

Fig. 2. Control System block diagram

1) Specifications: RAMA should be a fully autonomous control system for any kind of UAV. It is lightweight (approx. 1.5kg) and compact (330x160x65mm), to allow its usage even in the small UAVs. Although we use the RAMA system to steer a hobby helicopter, it can be adopted for fixed-wing aircraft if needed. All vital parts of the system are airborne. This includes the data acquisition system (sensors and their management), control system and communication system. The *Ground Station* (GS) serves only to gather and visualize the on-line telemetry. Generally speaking, RAMA should therefore be able to fulfill a pre-programmed mission, or at least enter some kind of fail-safe mode (emergency landing, hover or orbiting) when loosing the wireless connection with the GS. This is the ultimate goal for the future.

To be easily extendible, RAMA is designed as a networked hierarchical distributed system, as is pretty much common nowadays. It consists of several basically independent functional blocks, as can be seen from the block diagram (Figure 2). Only the RC transmitter/receiver and the servomotors are standard modeler equipment, all other parts are our own contribution. Those blocks (or *nodes*) are interconnected by the CAN 2.0b (Controller Area Network), a well-known serial bus, used mainly in the automotive industry. This architecture not only allows relatively easy expansion of the system, but also contributes to its reliability (failure of one of the nodes does not necessarily compromise the function of others) and simplifies the testing and debugging of the system. As will be shown later, RAMA has only one crucial node, whose malfunction would lead to an inevitable crash of the controlled vehicle. All other nodes may fail, but at least the manual controllability would be preserved.

100 kbps baudrate is being used for CAN communication in our case. The bus utilization is 20%, so there are no problems with extensive packet delays. The real-time behavior is also ensured by packet priorities and non-destructive arbitration, so no problems related to jitter were encountered. The bus overall behavior and mean/extreme data delays were evaluated statistically.

The core member of the RAMA system is the *Main Control Computer* (MCC), where the control and communication algorithms run. MCC is connected to the GS (Ground Station) via

Wireless Ethernet (IEEE 802.11b, WiFi) link. The data rate is 11Mbps, which is more than enough for our application, as the telemetry data packets are only some 200 bytes long and are sent 32 times per second. This results in approximately 0.6% of bus utilization, so there are no concerns with real-time issues. The telemetry is a soft real-time task anyway. In case of a communication loss, the data are stored in a buffer and sent off-line as soon as the data link is re-established. The WiFi communication is non-critical, as it is used only for telemetry and remote monitoring of the system. The commands for the autopilot (automatic/manual control mode switching and desired values for the controllers) are issued via the modeler RC system, which is much more reliable and has a lot wider range. The WiFi link will be replaced in the future by some better system. It is meant to be used only temporarily, to speed-up the development process.

Controller input data, such as current position and attitude of the vehicle in space, are provided by the *Navigation Unit* (NU). This unit includes an *Inertial Measurement Unit* (IMU) and a *Global Positioning System* (GPS). Currently, it being extended by a three-axis magnetometer to simplify the flight angles determination.

By now the flight angles are determined from the measured accelerations. The acceleration of gravity can be decomposed into three components, parallel with vehicle axes. Those components are measured by the accelerometers and the angles might be determined by triangulation (with one degree of freedom still left, i.e. one of the angles would be unsure and has to be determined by other means, e.g. angular rate integration). The disadvantage of this method is that the accelerometers measure not only the g acceleration, but also the vehicle accelerations, which are superponed on the g. Vehicle accelerations might be filtered using a low-pass filter, but this introduces latency into the process. More convenient method is to use three mutually perpendicular magnetometers, which measure the Earth magnetic field vector components. The magnetometers are not influenced by the vehicle dynamics and the angles might be determined in the same manner as from the accelerometers. Moreover, one can use both accelerometer and magnetometer data, and pecify the computation using a Kalman filter.

The actuators (that are, in our case, the servomotors driving the collective and cyclic controls, the tail rotor pitch and the throttle) are controlled by the *Servo Control Unit* (SCU). All servos and a RC receiver are connected to this unit. Its purpose is to control the servos according the commands from the MCC or the RC receiver and to switch between automatic and manual control modes. This is the only crucial point of the RAMA system, whose failure would be fatal.

2) Basic Facts and Ideas: Basically, the system can operate in two modes. Either in the *Manual Control Mode* (MCM), when the hosting 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, sometimes it is convenient if automatic control could be applied only to selected actuators, while the others are controlled manually; that's the reason why this *semi-automatic* option was implemented.

Regardless of the operational mode, all *Operational Data* (OD) (that are measured vehicle variables and control system variables) are gathered and sent to the GS (Ground Station) on-line. Those OD (Operational Data) comprise:

Measured vehicle variables

- Angular rates around lateral (*pitch rate*), longitudinal (*roll rate*) and yaw (*yaw rate*) axes of the vehicle
- Accelerations in the lateral (x), longitudinal (y) and yaw (z) vehicle axes
- Altitude above sea level and flight speed (decomposed into x, y and z components)
- Geographical position (latitude and longitude)
- Actuators position
- RC control sticks position

Control system variables

- Operational mode (automatic / manual)
- Controller(s) state variables
- Error notifications

The OD are sampled synchronously at the rate of 32Hz. Synchronization CAN message is sent periodically to all nodes to ensure that all data are sampled at the same time. All vehicle variables are sent to the MCC. When operating in ACM, the action value is computed and applied to the actuators as soon as the computation is finished. In MCM, the controller state variables are set so that the action value equals current manual action value, to ensure bumpless control transition in case of the manual-to-automatic mode change. Afterwards, the telemetry message containing the OD is composed and sent to the GS (Ground Station) instantly.

3) Main Control Computer: To shorten the design time, it was decided not to develop a tailor-made MCC for the project, but to buy a stock product instead. An extensive market research followed, and at the end of the day the EXM32 SH7760 product line from a German company MSC [20] was chosen.

The EXM32 system features a 64MB of SDRAM, up to 32 MB of FLASH memory, and a lot of peripherals (2x CAN controller, 10/100 Ethernet, USB, Compact Flash socket, 6x UART, VGA and others). With small Eurocard PCB footprint (160x100mm) and relatively low-power consumption (approx. 250mA@12V DC, that is 3W, in our case), the EXM32 forms an ideal computing platform for the RAMA system, able to execute complex control algorithms in real-time.

4) Servo Control Unit: The Servo Control Unit (SCU) is a simple purpose-developed embedded computer, built around the Renesas 2638F MCU, serving to control the servomotors. Basically, the SCU can operate in two modes; either in Automatic or in Manual Control Mode (ACM and MCM). In the ACM, the signals from the RC receiver are read and sent to the MCC, and the signals for the servomotors are generated by the SCU, according the commands from the MCC. In the MCM, the signals from the RC receiver are read and sent to the MCC, and also sent directly to the servomotors. Therefore, should the rest of the RAMA system fail and only the SCU would prevail, it is still possible to maintain manual control.

5) Navigation Unit: The Navigation Unit (NU) consists of three basic parts. The core member is the *Data Acquisition Module* (DAM)), based on the Philips LPC2119 ARM-core MCU, whose purpose is to acquire data from various types of sensors, preprocess them and send the results to the MCC. On top of that, DAM also provides the system with the time synchronization message. Currently, two sensing units are connected to the DAM; the first one being the Inertial Measurement Unit (IMU), and the other is the Global Positioning System (GPS) receiver.

MICRO-ISU BP-3010 [21] is being used as the Inertial Measurement Unit (IMU). It is a handy little unit, measuring just 35x22x12mm and weighing tiny 30g, with 0.5W power consumption, which contains 3 accelerometers and 3 gyroscopes, providing accelerations and angular rates in all 3 vehicle axes. Measuring ranges are +-10g / +-300 ° /s at 64Hz sampling rate.

Garmin GPS 16-HVS receiver [22] is being used as the GPS receiver. It is an OEM GPS module with integrated antenna and WAAS (Wide Area Augmentation System) capability. The data are sampled at 1Hz rate.

C. Control System Software

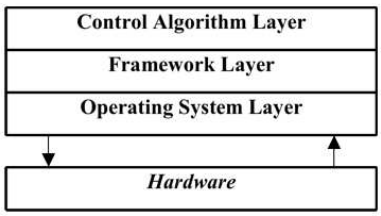

Fig. 3. System software structure

1) MCC Software: The software running on the MCC (Main Control Computer) could be separated into several layers (see Figure 3). Its main tasks are following:

- Gather the OD (Operational Data) from the sensors and actuators
- Compute the control action and deliver it to the actuators
- Communicate with the Ground Station (GS)

The uppermost layer is the *Control Algorithm Layer* (CAL). Its task is to generate actuation signals according OD.

The layer beneath is the *Framework Layer* (FL), detaching the controller from the operating system and the hardware. The CAL is communicating with the FL using standardized interface, and the FL does the rest of the work - that is to gather the OD from the underlying CAN nodes, to send the controller commands to the actuators and to communicate with the GS.

The *Operating System Layer* (OSL) represents the OS. Currently, we are using Linux 2.6.14 for that purpose.

The MCC software can be easily updated via the Wireless Ethernet (WiFi) link without the need to physically connect to the system. This greatly contributes to the ability to test various software and controller versions in the course of a flight-day, as the whole software can be changed within seconds. Even the firmware of other system nodes can be accessed and updated remotely (see the next section for detail).

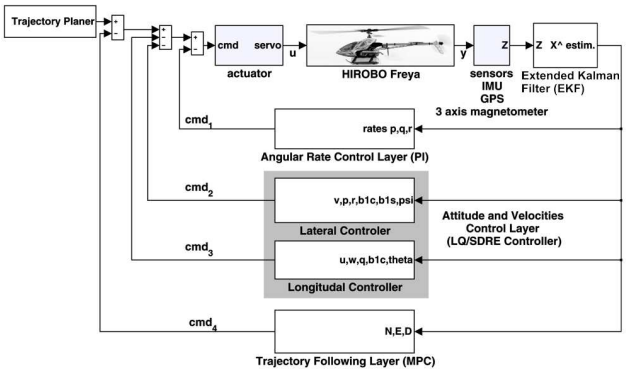

Fig. 4. Autopilot hierarchical structure

2) DAM and SCU Software: The DAM (Data Acquisition Module) and SCU (Servo Control Unit) are running systemless. Their programs are kept as simple as possible in order to maintain reliability, especially in the case of SCU, as it is the most crucial part RAMA's.

To enable remote software updates when the system is mated to the UAV, both modules are equipped with bootloaders, to enable firmware uploads via the CAN bus. Firmware updates can be performed using the wireless link - new firmware is uploaded to the MCC, and then to the modules using CAN. Therefore, all software can be changed remotely, without the need to de-mate the system.

III. Helicopter Mathematical Model and Parameters Identification

A helicopter is strongly non-linear, unstable system with 6 Degrees Of Freedom (DOF). Its mathematical model could be derived trough the extension of a rigid body dynamics with external moments and forces acting on it [6]. Basically, a helicopter has 4 inputs (longitudinal and lateral cyclic, collective and yaw) and 6 outputs (3 dimensions of the position in space and 3 Euler angles). From a control engineer point of view, it is one of the hardest systems to cope with. To make things worse, there are also some interdependencies, for example the longitudinal - lateral dynamics is interrelated.

Several mathematical models, more or less complex, have been studied and tested. For the initial controller design a well-known 10 state space model [11] was chosen: $[u, v, w, p, q, r, \theta, \phi, \beta 1_c, \beta 1_s]$, where u, v, w are the body velocities, p, q, r are the angular rates, $[\beta 1_c, \beta 1_s]$ describe the main rotor tip plane flapping, and finally θ, ϕ are the Euler angles. The identification of model parameters for the Freya helicopter was limited by the signal preprocessing, necessary due to high vibration noise superponed on the observed signals. Its elimination was solved using the Kalman filter for Gaussian colored noise [24]. The model had been implemented using the Matlab symbolic toolbox as a state-space model with external noise (n4sid [13]), therefore its modification (addition of new states for estimation of unknown/unprecise model parameters, sensor biases and IMU axis misalignment) could be done easily.

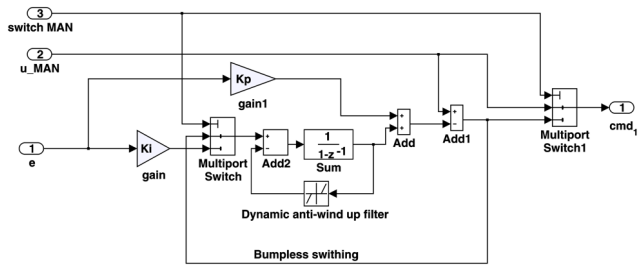

Fig. 5. PI-type angular rate stabilizer structure

IV. Helicopter Controller Development

Proposed R-UAV autopilot is structured hierarchically (see Figure 4). It consists of three layers. The first layer (Angular Rate Control Layer, ARCL) is responsible for angular rates stabilization. The second layer (Attitude and Velocities Control Layer, AVCL) is responsible for attitude and velocity stabilization. The uppermost layer (Trajectory Following Layer, TFL) is responsible for the trajectory following.

The OD (Operational Data), obtained from the IMU (32Hz), GPS (1Hz) and potentially a 3-axis magnetometer (32Hz) (as soon as it would be implemented into the hardware), are used for the system states estimation [1]. An *Extended Kalman Filter* (EKF) [12] is applied, using quaternion Euler angles representation. Estimated system state vector $\hat{x}$ is fed to respective controllers. Control commands (denoted cmd) are then fed (hierarchically) to the actuators.

Actually, only the first control layer (ARCL) had been implemented and flight-tested by now. The higher-level controllers had been designed and simulated in matlab, but are not implemented in the RAMA software and had not been flight-tested yet.

The ARCL consists of three totally independent controllers. Those are roll, pitch and yaw angular rate stabilizers. The roll controller actuates directly the lateral cyclic servo, the pitch controller actuates the longitudinal cyclic servo and the yaw controller drives the tail rotor pitch servo. All three controllers have the same internal structure, only their parameters are different. PI-type controllers with anti-windup filter are used (see Figure 5). To ensure bumpless control transition when switching from MCM to ACM, the summator initial value is set so that the controller actuation equals the manual actuation at the moment of MCM/ACM switch. See the section V for the experimental results.

The AVCL could be implemented in several ways. LQG and SDRE [5] control structure was chosen in our case. The LQG attitude stabilization is based on the decomposition of the dynamics into the longitudinal and lateral modes, as is common in aircraft control. Cross dependencies must be considered in order to enhance performance. The SDRE method utilizes a complex form of state-parameterized model, describing lateral and longitudinal dynamics. Main advantage of this approach is better differentiation of the flight dynamics in hover and cruise flight regimes.

The last layer (TFL) would probably be implemented using a Model Predictive Controller (MPC). This layer had not been designed yet.

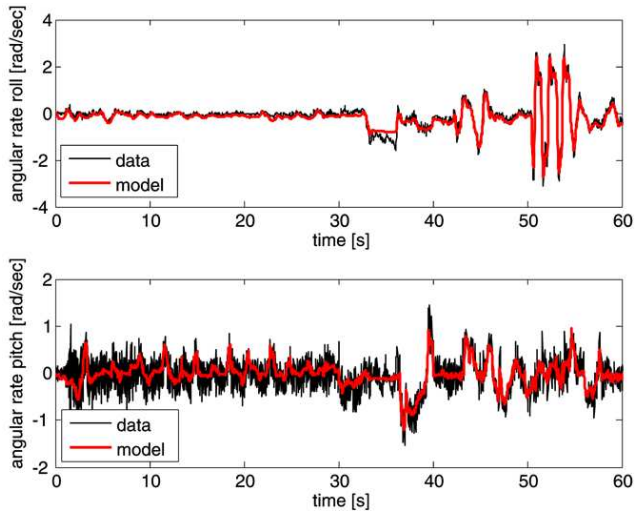

Fig. 6. Comparison of identified mathematical model vs. real flight data

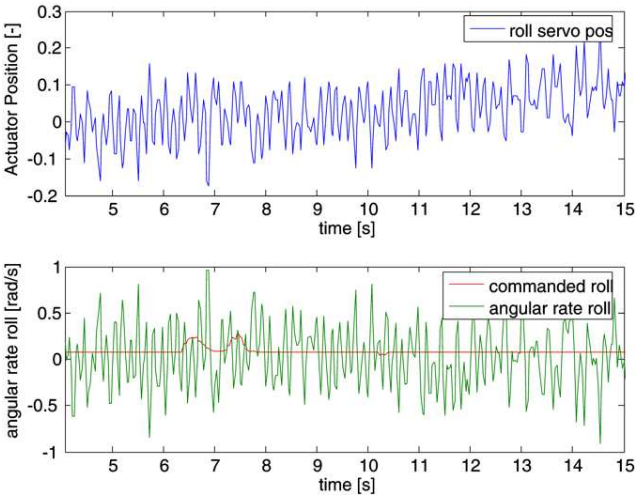

Fig. 7. Roll rate stabilizer performance in flight

V. Flight Tests

Dozens of flight tests were performed to identify the mathematical model parameters and to test the control system. The identification experiments were primarily focused on angular dynamics identification, as the first control layer (ARCL) was designed. Good results were achieved (see Figure 6, showing the conformity between the simulated and measured roll and pitch dynamics), despite high amplitude noise present in the measured data.

Fortunately, most of this noise (caused by airframe vibrations) could be filtered relatively easily using polynomial low-pass filter, although the results are still not completely satisfactory. The problem is that the filtering in real-time introduces latency into the control loop, limiting the quality of the filtering (only the first or second order filters are suitable to be implemented in real-time). Off-line filtering with higher order filters is working well, but the real-time filtering is still something that needs to be improved.

Good quality of the flight data and excitation of all system modes is needed for successful identification. The identifica-

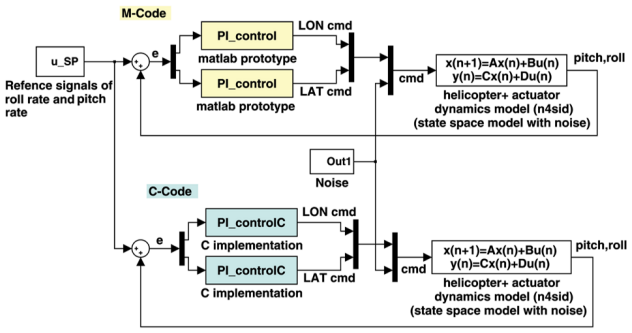

Fig. 8. Controller implementation testing

tion experiments were focused on covering both hover and cruise flight regimes.

After having the mathematical model identified, the angular rate stabilizers (ARCL in the hierarchical structure) were designed. Wide range of PI controllers (from careful to aggressive) were tried out. Couple of them had shown promising performance, although some of the earlier tests were hindered by various teething problems, related to the hardware (electromagnetic disturbances affecting the RC apparatus, vibration-related issues with IMU, GPS-related problems). Figure 7 shows the roll rate stabilizer performance during one of the experimental flights. Note that the measured roll rate is greatly deteriorated by vibration-related noise, which compromises the controller performance.

To ensure that the control algorithm was implemented properly and no bugs were added during the process of the Matlab-to-C code transfer, a testing scheme (Figure 8) was introduced. The response of the C-implemented and Matlab/Simulink-implemented controllers is thoroughly evaluated prior to flight tests.

The measurement records with raw/processed flight data, detailed experiments descriptions and video records are available at the project website [24].

VI. Conclusion and Future Work

RAMA (Remotely operated Aerial Model Autopilot), an open control system for Unmanned Aerial Vehicles, was introduced in this paper. After approximately 2 years of development, some promising results had been achieved - the system is now flown on regular basis, HW and SW development is almost finished, basic mathematical models had been developed and parameters identified, and low-level stabilizing controllers had shown promising results.

For the future, the HW and system SW would probably be changed only slightly; the Navigation Unit is being extended by a three-axis magnetometer to improve angles determination. Extensive work would be carried out on the controller design, as higher layers of the autopilot would be added and tested. More complex mathematical models are also to be introduced, to allow design and testing of more complex control methods.

The ultimate goal, that is to develop a fully autonomous vehicle with fault-tolerant control system, is still very distant. Currently, only the lowest control layer (angular rates stabilization) had been designed, and even that had still not been properly tested (only some initial tests had been performed). Higher layers (angular and velocity stabilization, trajectory following) are still to be designed and tested. But we are confident that this goal can be achieved, hopefully not only to our benefit, but also to the benefit of other researchers in the UAV field.

Comprehensive information on the project and its current state may be found at the project website [24].

Acknowledgment

This work was supported by the Ministry of Education of the Czech Republic under Project 1M6840770004.

References

[1] Merwe R., Wan E., Sigma point Kalman Filters for integrated navigation. report IAAA, Oregon Heath and Science University. url: http://speech.bme.ogi.edu/publications/ps/merwe04a.pdf
[2] Pettersen R., Mustafic E. and Fogh M., Nonlinear Control Approach to Helicopter Autonomy. Thesis, Aalborg University, 2005
[3] Pettersen R., Mustafic E., Fogh M., Developement of real time flight simulator for an experimentalmodel helicopter. Thesis, Georgia Institute of Technology, School of Aerospace Engineering, 1998
[4] Mettler B., Identification, Modeling and Characteristic of Miniature Rotorcraft. Kluwer Academic Publishers, Dordtrecht, The Neteherlands, isbn 1-4020-7228-7, 2003
[5] Bogdanov A., Wan E., Harvey G., SDRE Flight Control For X-Cell and R-Max Autonomous Helicopters. Proceedings of the AIAA Guidance Navigation and Control Conference, OGI School of Science and Engineering, Austin, USA, 2004
[6] Gavrilets V., Mettler B., Feron E., Nonlinear model for a small-size acrobatic helicopter. Proceedings of AIAA Guidance Navigation and Control Conference, Montreal, Canada, 2001
[7] Pettersen R., Mustafic E., Fogh M., Nonlinear Control Approach to Helicopter Autonomy. Technical report, pp.136, Aalborg University, Alborg, Denmark, 2005
[8] Jensen R., Nielsen A., Robust Control of an Autonomous Helicopter. Technical report, pp.78, Aalborg University, Alborg, Denmark, 2005
[9] Hald U., Hesselbok M., Holmgaard J., Report: Autonomous Helicopter - Modelling and Control. Technical report, pp.162, Aalborg University, Alborg, Denmark, 2005
[10] Prouty R., Helicopter Performance Stability and Control. Krieger Publishing Company Inc., Malabar, Florida 32950, pp.560, ISBN: 1-57524-209-5, 2003
[11] Munzinger Ch., Development of a real-time flight simulator for an Experimental Model Helicopter. Technical report, pp.130, Georgia Institute of Technology, 1998
[12] Extended Kalman Filter description, url: http://www.cs.unc.edu/ welch/kalman/
[13] n4sid model description, url: http://www.mathworks.com/access/helpdesk/help/toolbox/ident/ident.html
[14] Weilenmann, M. F. and Geering, H. P., A test bench for rotorcraft hover control. Journal of Guidance, Control and Dynamics, vol. 17, pp. 729–736, 1994
[15] Chapuis, J., Eck, C., Kottmann, M., Sanvido M. and Tanner, O., Control of Helicopters. Control of Complex Systems, pp. 359–392, 2001, url: http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=375
[16] Shim, D. H., Kim H. J. and Sastry S., A Flight Control System for Aerial Robots: Algorithms and Experiments. IFAC Control Engineering Practice, 2003
[17] Buskey, G., Roberts, J. M. and Wyeth, G., Autonomous helicopter hover using an artificial neural network. IEEE International Conference on Robotics and Automation, pp. 1635–1640, 2001
[18] MARVIN UAV project, url: http://pdv.cs.tu-berlin.de/MARVIN
[19] Yamaha RMAX UAV, url: http://www.yamaha-motor.co.jp/global/industrial/sky/solution/rmax/index.html
[20] MSC Company website, url: http://www.msc-ge.com
[21] BEC-NAV Company website, url: http://www.bec-nav.de
[22] Garmin Company website, url: http://www.garmin.com
[23] Hirobo Company website, url: http://model.hirobo.co.jp/english/index.html
[24] RAMA UAV project, url: http://rtime.felk.cvut.cz/helicopter