Development and Validation of on-board systems control laws

Development and validation of on-board systems control laws

Giovanni Medici, Nicole Viola, Sabrina Corpino and Marco Fioriti
Department of Aerospace Engineering, Politecnico di Torino, Torino, Italy

Abstract

Purpose – The purpose of this paper is to describe the tool and procedure developed in order to design the control laws of several UAV (Unmanned Aerial Vehicle) sub-systems. The authors designed and developed the logics governing: landing gear, nose wheel steering, wheel braking, and fuel system.

Design/methodology/approach – This procedure is based on a general purpose, object-oriented, simulation tool. The development method used is based on three-steps. The main structure of the control laws is defined through flow charts; then the logics are ported to ANSI-C programming language; finally the code is implemented inside the status model. The status model is a Matlab-Simulink model, which uses an embedded Matlab-function to model the FCC (Flight Control Computer). The core block is linked with the components, but cannot access their internal model. Interfaces between FCCs and system components in the model reflect real system ones.

Findings – The user verifies systems' reactions in real time, through the status model. Using block-oriented approach, development of the control laws and integration of several systems is faster.

Practical implications – The tool aims to test and validate the control laws dynamically, helping specialists to find out odd logics or undesired responses, during the pre-design.

Originality/value – The development team can test and verify the control laws in various failure scenarios. This tool allows more reliable and effective logics to be produced, which can be directly used on the system.

Keywords Simulation, Control systems, Design, Unmanned aerial vehicles, Control system design, Flowcharts, Programmable logic controller

Paper type Research paper

Nomenclature

Definitions, acronyms and abbreviations

FCC = flight control computer
FS = fuel system
GCS = ground control station
HALE = high altitude long endurance
IMA = integrated modular avionic
LASE = low altitude short endurance
LDG = landing gear system
MALE = medium altitude long endurance
NWS = nose wheel steering system
PS = pressure sensor
SOV = shut off valve
SSC = supervision and coordination station
SMAT = advanced environment monitoring system
TCS = tactical control station
UAS = unmanned aerial system
UAV = unmanned aerial vehicle
WBS = wheel braking system
WOW = weight on wheels

Introduction

In the last 20 years the use of electronics in aviation industry has seen a dramatic growth. When a complex system is designed, each subsystem's logic controller must be created. Once the subsystem's control laws are designed, they must be validated. During control laws test and validation, considerable cost reduction can be achieved (Zhang *et al.*, 2000), by using digital models of the system.

When considering remotely controlled devices, reliability and safety requirements grow (Loh *et al.*, 2009). Unmanned aerial vehicle (UAV), due to the lack of a pilot on the aircraft, shall manage autonomously many systems and routines. Moreover, data-link delays (due to satellite or distance) leave to the remote operator little time margins. For this reason a considerable amount of procedures and routines of various systems must be stored in (and invoked autonomously by) the flight control computers (FCCs). This amount of avionics needs a huge integration effort. Single failure of any component (including one FCC) may lead to a downgraded system operation, but never to total system loss. Integrated modular avionics (IMAs) is an answer to complex system design, and its use in UAVs is rising (Lopez *et al.*, 2008). IMA has been widely used since the beginnings of the 1990s, both in aeronautics and space applications (Doss *et al.*, 1996). Being used for the first time in Boeing 777, this technology has led to step improvement

The work has been performed through a close cooperation with Alenia Aeronautica staff, with constant technical meetings and a continuous information exchange. In particular the authors wish to thank Dr Maria Airoldi, Mr Marco Mantovani, Dr Alessandro Pasquino and Dr Massimiliano Paternoster.

in system quality, reliability, safety, and maintainability, while reducing cost (Morgan, 1991).

This paper describes the method used to develop and test the control laws of several systems on a UAV: Landing Gear, Nose Wheel Steering, Wheel Braking, and Fuel System (FS). SMAT UAS project is described in the first section, while the following sections illustrate the development of the method and the Status Model. Eventually some conclusions are drawn.

The SMAT UAS project

The authors have developed the present method in the framework of SMAT UAS project. SMAT project is currently under way and has now completed the first phase (named SMAT-F1), with a Demo flight on September 30, 2011.

The aim of SMAT is to define, design and develop an Advanced Environment Monitoring System, based on UAS. Within the UAS, UAV and their ground control stations (GCSs) are coordinated and managed by a Supervision and Coordination Station (SSC); a summarized conceptual overview of SMAT system is shown in Figure 1.

The analysis of innovative solutions to improve performances of future UAV is a remarkable activity within SMAT project. Some of these novel approaches have already been tested on some Alenia Aeronautica Technological Demonstrators (Chiesa *et al.*, 2010a, b; Farfaglia *et al.*, 2009). Both technological and knowledge-based issues have been widely tested during the development of the SKY-Y, a Medium Altitude Long Endurance technological demonstrator of Alenia Aeronautica. The UAV, which flies at the highest altitude (more than 12,000 m) in SMAT project (Figure 1), is not yet available as Alenia Aeronautica's Product, but will be developed on the basis of SKY-Y (Table I and Figure 1 collect some technical data and the design layout of the SKY-Y UAV).

The performances of SKY-Y are reduced, if compared to the envisaged performances of the final product. Nevertheless, studies for a derivative UAV are on going. The authors have been working on these improvements.

The UAV is designed to be single failure proof; its FCCs have been developed to share both software and hardware. Depending on the position they have on the rack, the software loaded can be: FCC1, FCC2 or FCC3. The redundant hardware design allows, in the event of FCC failure, to keep control of every system. The dual-redundant mission management computer has been used in various UAVs (Loegering and Evans, 1999), such as the Global Hawk.

The UAV has been developed using a modular design. If any failure occurs (FCC loss included), the system (and each subsystem) keeps on operating in normal or in a downgraded mode. Let us focus our attention on an example: FCC1 powers the landing gear's actuators. In an event of FCC1 Failure, Landing Gear System (LDG) extension function cannot be performed. In order to avoid this event (LDG extension failure), FCC3 has the authority to release the up-locks, allowing the landing gear's unpowered actuators to free-fall (due to gravity and drag). Although the function is downgraded (there is no control on the actuators), the landing gear subsystem withstands the FCC1 failure.

This approach leads to a safer and more reliable airplane. Safety and reliability have always been fundamental requirements in the aviation field, and by now are key-points along the path to certification for civil use of UAVs (use in a civil air traffic control scenario).

The relatively new growth of those particular airplanes has led to the need to define certification requirements from scratch. The certification authorities are working hard and side by side with the manufacturers to fill this gap. Software should be certified too, and certification authorities are interested in graphical visualization of the algorithms for their purposes. The Status Model uses widely the graphical approach.

The control laws development procedure

During the development of SMAT-F1 project the authors and Alenia Aeronautica staff have been requested to define the control logics for various subsystems: LDG, Nose Wheel Steering System (NWS), Wheel Braking System (WBS) (both Hydraulic and Electrical Powered), and FS. Each system has been designed by using the same procedure.

Figure 1 The SMAT system, an overview and design layout

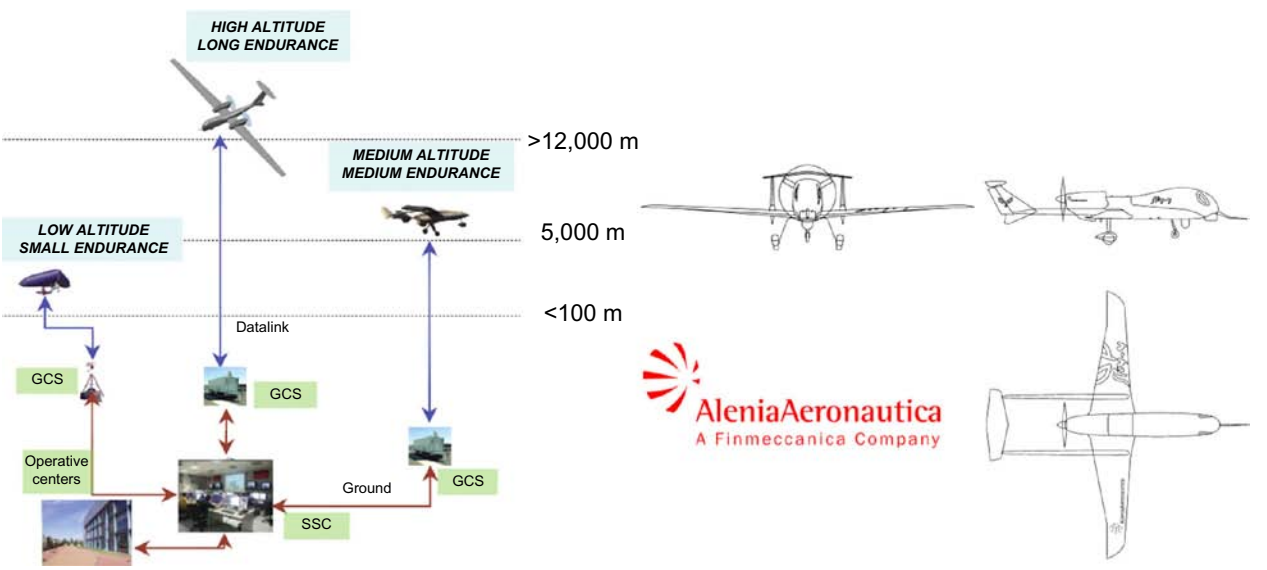


Table I SKY-Y male technical data

Dimensions	*(m)*	*Weight*	*(kg)*
Length	9.725	MTOW	1,200
Span	9.937	OEW	800
Wing area	10.785	Max fuel	250
		Max payload	150
Payloads		*Performances*	*(km)*
EO/IR sensor		LOS radius	185
Hyper spectral sensor		Max range	925
Synthetic aperture radar		Altitude	>7.6
Propulsion one dieseljet TDA 1.9 JTD 8 valve diesel aviation engine			
Endurance 14 (h)			

The design method has three main steps: control laws algorithm definition, code development, and validation (Figure 2 shows the main stages of the procedure).

Initially the main control laws logic has been defined by using a fast, easy to modify and extremely direct visual language, i.e. the flowchart approach. This well known approach (ISO Standards, 1985) is flexible and valid in the preliminary design phase. It has been effective during the initial brainstorming sessions, aiding specialists to focus on the key safety issues, and troubleshoot failures modes.

Once the general structure of the algorithms had been defined, the code to be used in the various systems has been developed. This part of the procedure aims at defining every detail of the control laws. The flowcharts have been used as reference for the code's development, for which a standard programming language has been utilized. The intention is to use the same control laws already developed (the code) inside the FCCs; for this reason ANSI C has been used (FCC runs ANSI C compiled programs). The code is easy to read and can be used with minor changes on the Status Model.

The purpose of the last step of this method is to test and validate the control laws. In aerospace industries it is common practice to build a test-rig of the system (iron bird), upload the control laws on the FCCs and verify various scenarios by using a checklist. Any control law error may cause the test-rig system to fail mechanically (development cost increase) or in the worst case, a system redesign (which affects budget too).

A virtual model of the system has been developed using Simulink, with the purpose of testing and validating the control laws. It has proved to be a fast and almost inexpensive way to test different failures scenarios. The tool, called Status Model, is described in the next section.

The Status Model

The Status Model is a tool to model, test and validate the control laws. It uses the same code developed in the previous steps and tests it. The Status model is a Matlab Simulink based model. An embedded Matlab function block, the core, models the FCCs (which contains the same control laws that have been previously developed). Each component of the system is defined through a virtual model, and every FCC I/O interface is preserved with respect to the real system.

An overview of the Status Model layout is shown in Figure 3. The Simulink Model file contains all components of the UAV systems (e.g. actuators, relays, pumps, valves, etc.). By using a Matlab function block, the core models the three FCCs (central block in Figure 3). The FCC block contains the ANSI C code (minor conversion from C to Matlab syntax is required), receives the inputs (signals: discrete or analog) from the various components, and sends back commands/outputs. The interfaces of the virtual model and the real system are identical; for example the power relay, which feeds the nose landing gear's leg-actuator, receives a discrete input from the FCC2; the same interface is modelled in Simulink; the FCC2 block outputs a discrete to the power relay model.

In the simulation, during every time step, FCCs control laws (and software, stored in the Matlab Function Block) run simultaneously. Three FCCs receive all input signals, process them, and send the outputs signal back.

This particular layout allows testing the control laws that have been previously designed, by implementing them directly inside a modelled Mother Board (modelled with the Matlab Function Block). The outputs of such blocks are linked to the simplified system components models (modelled through other Simulink Blocks).

Specialists are able to create a model using a reusable set of components. Matlab Simulink has a huge block library, which contains fully scriptable switches, relays, transfer functions blocks and much more. Those items can be used "out of the box", in order to model some simple subsystem components. Block-oriented approach offers some features valuable for building customized systems, and has been used widely in system control development (Gilberl and Diehl, 1994;

Figure 2 Procedure used to develop the control laws

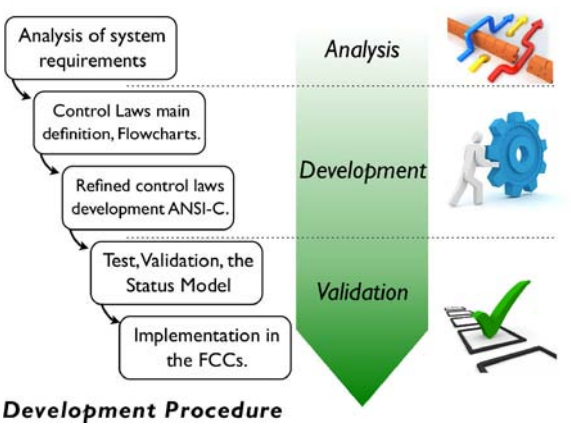
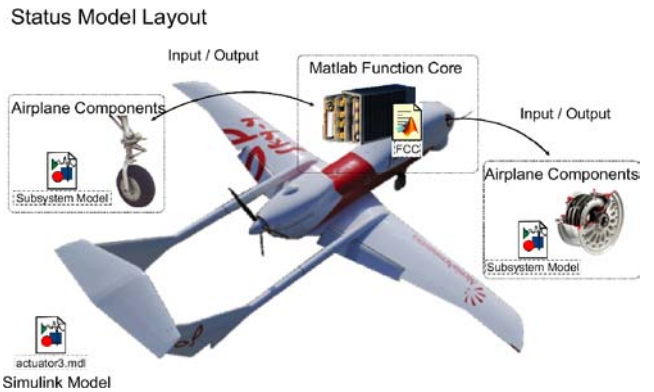


Figure 3 Status model layout



Ji *et al.*, 2006; Keller, 2006; Renfrow *et al.*, 1994). The basic object is the component: a reusable, self-contained entity that requires inputs and produces outputs. It is also possible to group and reuse customized components or even subsystems. This allows specialists to build up a customized library, which satisfies their needs. A component can be controlled (by other components) only through its inputs (external interface). Consequently, its internal (private) data and methods are hidden (encapsulated) from the other components (or Matlab Embedded Function Block). Even the icon of a component may be customized in order to recall the real component shape. Once the model is complete, specialists can test and validate their controller logics. Since the model uses signal as input for the Matlab Function Block, it is possible to test the system behaviour when a failure occurs.

For example, an input signal can be forced to reach a null value or to take a weird behaviour; FCCs will act consequently. The Status model is suitable for multi-system integration, so that the FCCs' control laws stiffness can be tested crosswise. This approach allows the operator to check how a faulted component may affect other functions or systems, and helps the troubleshooting inside complex and integrated systems.

General layout

In the next sections the Status Model will be described through a detailed overview of the model. Once the operator starts a simulation and opens the Simulink Model File, three main windows show up (Figure 4):

1 *Display window.* This window displays all lights and status indicators of the model, which changes colour dynamically during the simulation.
2 *Log window.* This window is a console; it displays all text messages (warning, status, counters) in a verbose mode. Saving log sessions is possible too.
3 *Command window.* From this window the user can send commands and/or invoke failures dynamically (in real time) during the simulation.

Display window

The display window has different lights and indicators, as reported in Figure 5. It is basically a customized Matlab Figure, with a status handle for each element. This panel summarizes various vital information of each subsystem.

Figure 5 summarize every light and indicator through a numbered list; a brief description of each item is provided in the next paragraph:

- LDG system: (1). Emergency LDG command: depending on its colour, this light indicates normal, potential emergency (as previously stated, in the event of FCC1 or FCC2 failure, the landing gear can be extracted only through an emergency command) or emergency command sent (2). Override LDG Command: it indicates either override or normal mode in action. The present command overrides any pre-retraction checklist and forces the landing gear retraction. This routine is important during maintenance and in some emergency procedures (11). Weight on wheels (WOW) Aircraft Status: reports the WOW status, can be Air, Ground, unknown/failure (12). LDG Command: the light depicts the LDG command status, and indicates inactive command, a gear-up/gear-down command, or a system failure (13). LDG Position: this indicator represents the down-lock, up-lock sensors status on each leg. Locked down status is displayed as green, while a red status indicates a not locked position (failure or moving LDG), once the leg-actuator has reached the locked-up position, the lights turn off (gray).
- FS: (3). Fuel Full light: it points out normal or full fuel tank level (4). Fuel Bingo light indicates whether the Bingo Fuel level is reached or not. Bingo is the minimum amount of Fuel that allows the UAV to turn back and head immediately to the base (5). Fuel low light: this light flashes when the Fuel is low. In this condition the system will soon end to work properly (engine power loss) (8). Fuel Temperature light: it indicates Fuel Temperature to be or not within the defined thresholds.
- WBS system: (6). Braking power indicator: these two bars report the left and right braking demand of the aircraft.
- NWS system: (7). Steering angle nose LDG command and position: those lights point out the angle command of the nose LDG's leg (solid yellow thin line) and position (solid blue thick line) (10). Steering centred signal: indicates un-centred/centred nose LDG's leg.
- Boundary conditions data (9). Aircraft Speed Indicator: reports the speed of the aircraft. The Status model does not simulate any physic model of the aircraft; speed-reading is used to trigger some checks (such as LDG retraction, NWS system activation), and can be adjusted as desired.

Figure 4 Status model simulink window

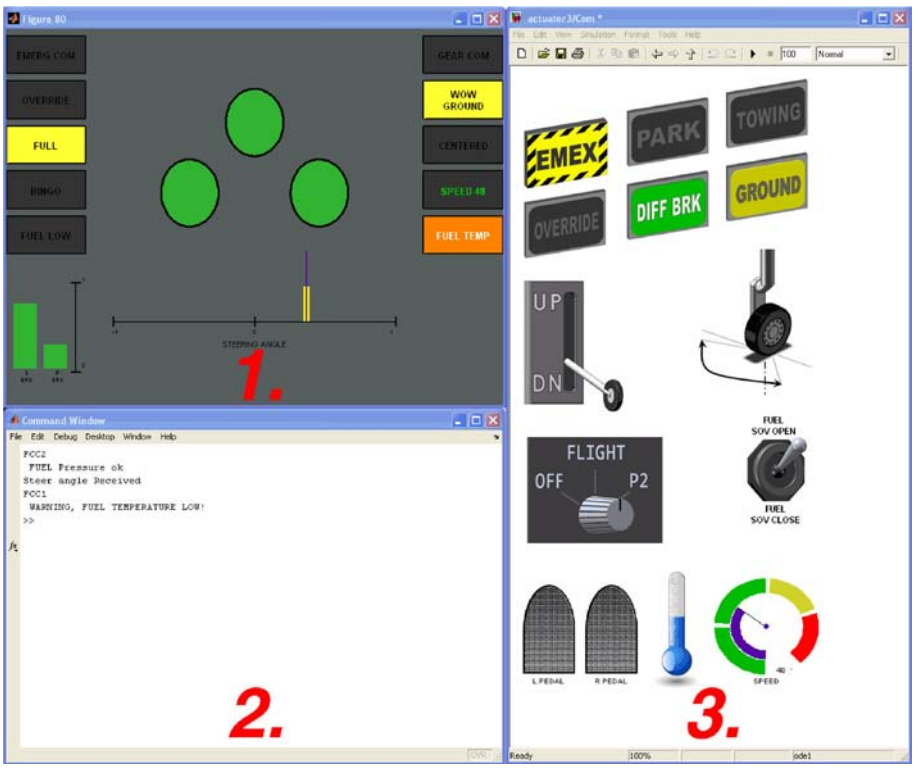
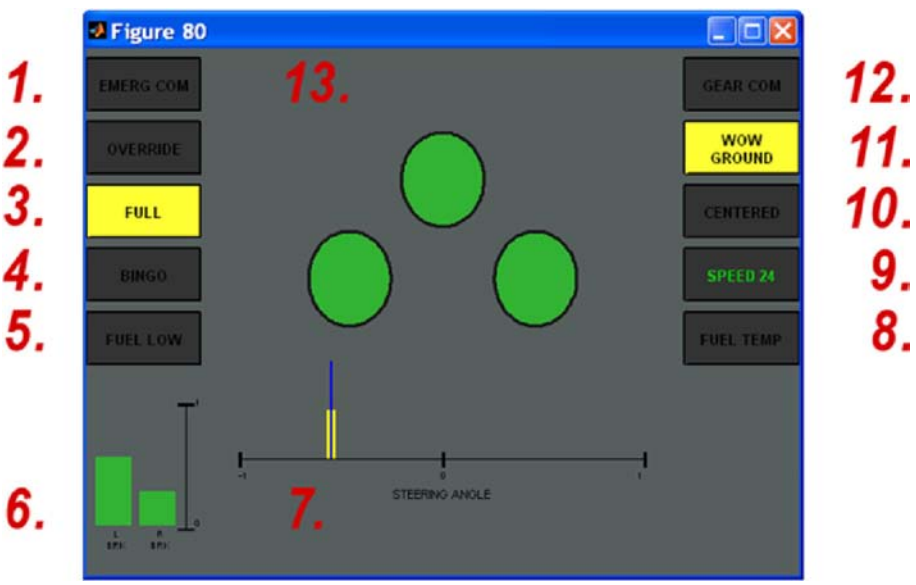


Figure 5 Status model display window



Log window

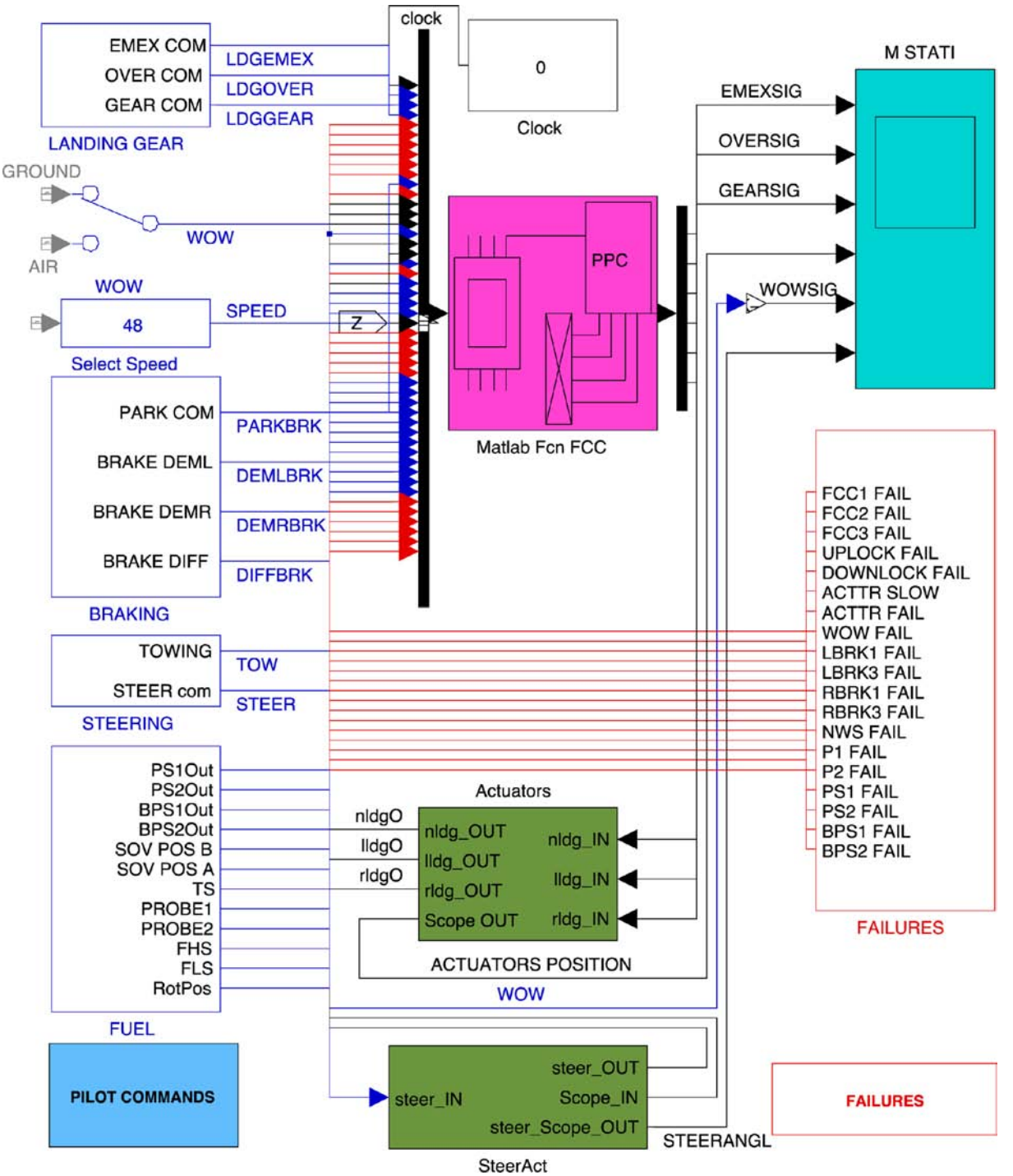
The log window writes the verbose log of the status of each sensor, relay, and FCCs command. It is used to monitor the systems' status and to check when a failure is detected by the FCC. The Log window collects every message, warning or information that the FCC produces and sends to Tactical Control Station (TCS). The Status Model contains also a routine that dumps the data to a text file, in order to be used as simulation post processing tool.

Command window

The command window embodies the Simulink model, this is the core of the tool; it contains several sub-masks and is briefly explained in the next section (Figure 6).

The Command Window splits the screen in three major parts. The central part houses the core block, FCC (pink box in Figure 6); all other components are connected to this block (as shown in the general overview of Figure 3). Various systems masks (blue and green boxes in Figure 6) are

Figure 6 Status model structure



present in Figure 6: LDG, NWS, WBS, and the FS. On the right hand side at the bottom in Figure 6 there is the failure panel (red box), while at the top there is the scope. It collects and plots useful information (turquoise box), like for instance, relay status, pressures, and LDG leg position.

During the simulation the operator simulates the TCS, or Pilot inputs. The Pilot Command Window (Figure 4, point 3) contains a customized graphical user interfaces (GUI), which the operator uses to interact with the Status Model. In order to give to the user a better and more intuitive experience while using the Status Model, customized icon (which look like the real system's buttons) have been used as masks for subsystems and blocks in general. The Pilot Command Block collects every command the pilot (or TCS) may send to the virtual model. The integrated command interface presents several controls (14), which derive from the four systems modelled. In the bullet list below these commands are grouped depending on the subsystem they are related to:

- *Landing gear.* This system handles three TCS commands to the LDG: emergency extension (EMEX), override retraction (OVERRIDE), normal mode (UP DN).
- *Braking system.* Several braking modes can be activated: parking brake (PARK), EMEX, and normal braking (through the left and right BRK pedals).
- *Steering system.* Steer demand, and towing. The NWS icon opens the steering angle command interface; by clicking on the (TOWING) switch, the NWS actuator disconnects from the wheel, so that the UAV can be moved on ground freely.
- *FS.* Fuel pumps, and shut off valve (SOV) switch. The fuel pumps rotary switch has three positions: OFF, Flight, P2. In the off position both fuel pumps are turned off, and pressure sensors are neglected. In Flight mode fuel pump one is turned on; if Pressure Sensors (PS1 or PS2) detect out of range pressure, the FCC powers on fuel pump two. The P2 mode overrides any PS check, and powers on fuel pump two.
- *General commands.* Few commands can change the environment the UAV is flying in, which in turns affect various subsystems (e.g. temperatures, speed, WOW, etc.).

The block-oriented design allows the subsystem model to be visually easy to understand. Specialists build the Status model using components, and drawing the connections between each element, with the same layout as pipes and wires join the components in the real system. During the simulations the operator can open each block and visually check the status of the relays (in this way is possible to estimate the system status in real time). The operator can send system specific commands directly from the subsystem block window (Figure 7), and look at the system reaction.

Subsystem blocks

In the next section the hydraulic braking subsystem and its respective virtual model are taken as example, and described in detail; a comparison between the real system and the Status Model is provided; then a single component (valve) model is sketched out.

Figures 8 and 9 show, respectively, the hydraulic braking system layout of the UAV and the overview of the hydraulic brake subsystem interface in Matlab Simulink (Status Model).

The hydraulic brakes system of the UAV is based on three operating modes: park brake, normal mode, and emergency mode. The park brake mode supplies full brake power with no regulation on the demand. A bistable valve provides pressure to the brakes line. This particular valve must receive a discrete command to change its position, in this way, even when the UAV is powered off, it can still keep brakes active, as long as the braking system pressure remain within the designed thresholds. The bistable valve, named Parkvalve in Figures 8 and 9, is powered by two relays (Park Open Rel and Park Close Rel). Normal brake mode is regulated by the isolation valve (Isovalve), which returns feed readings (Isofeed). One relay powers the valve (Iso rel). The servo-valve and shuttle valve provide the brake demand regulation on the left and right brake wheels. Emergency brake mode is activated by the emergency valve (Emexvalve), a relay provides power to the valve (Emex rel).

Only one valve shall be open at the same time. FCCs must check that all valves are switched off prior to power-on any valve's relay. The latter requirement has been the main issue in the control laws design. Various failures and simultaneous commands scenarios have been tested, by using the Status Model, in order to validate the braking system control laws. When hydraulic pressure in the brake power module is lower than a defined threshold or the LDG is down, the hydraulic pump is powered on by the dedicated relay (Pow rel).

In the Status Model is common practice to split the subsystem blocks in two major parts (as shown by the red numbers in Figure 9):

1 on the left hand side the user accesses the commands; and
2 on the right hand side there is the braking system model.

In the hydraulic brakes block the user, during the simulation, can switch to the desired mode by clicking the manual switches on the left hand side of the subsystem, and watch the sequence of actions performed by FCCs. On the right the Simulink model traces any reaction of the relays (power relay, isolation valve relay, parking valve open and close, and emergency valve relay) and draws it automatically. The relays position changes accordingly to the FCCs commands, so that the user can visually decode the system status, directly by looking at the relays symbol. In the Status Model each component has been modelled with a degree of detail depending on its role in the entire system.

The purpose of the Status Model is to validate the control laws in various systems, using a virtual model, without building a test bench. The model of each component should be as detailed as required. Transients, non-linearity, noise and other phenomena should be included in the model only if they affect the control law's response. Therefore, a description of a component model (valve) is here presented.

The various valves blocks in the hydraulic brakes subsystem window (Figure 9) contain a transfer function. Let us consider the isolation valve: its inlet is the hydraulic pump's outlet. The blue relay on the right (iso rel) commands the valve (either open or closed). The FCCs command directly the relays. When the valve receives a discrete, there is a transient (due to the transfer function), that models the valve opening sequence, and then the new position is reached (valve open). A variable gain models the shuttle valves; it regulates the pressure supply, in order to satisfy the brake demand. Since FCCs do not receive any output from this component, a very simple model has been used.

Figure 7 Pilot commands window

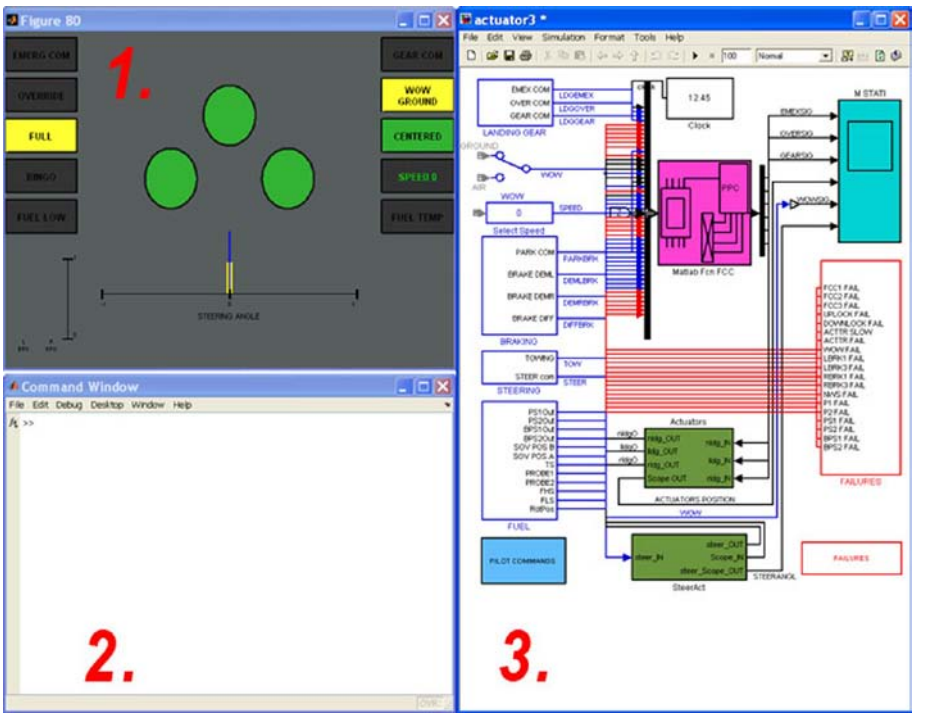


Figure 8 Hydraulic brakes system

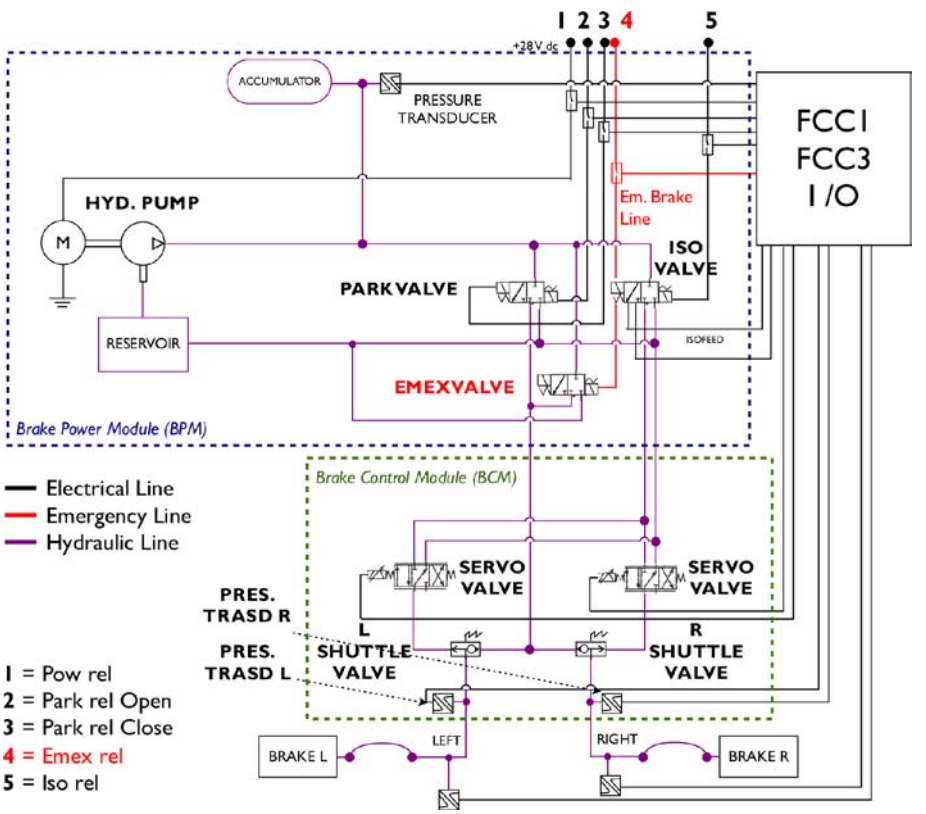
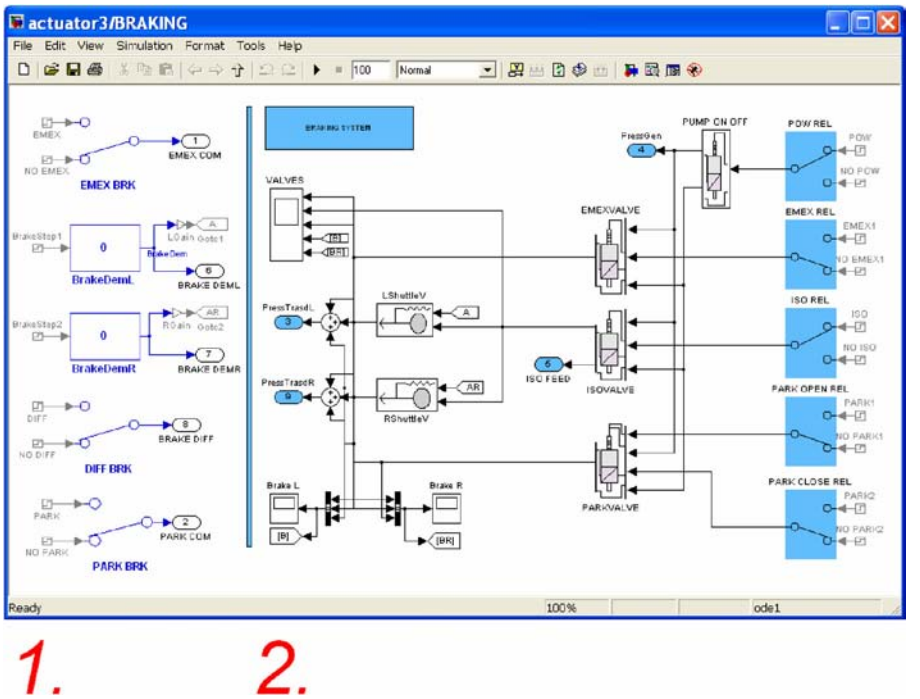


Figure 9 Hydraulic brakes block



These brief examples show the procedure used to model simple and complex components, but once a valve component is created, it can be reused in any other application, which requires that degree of detail.

Failures

The Status Model's tool enables to verify exactly how the FCCs' logics respond in the event of failure. It is possible to cut a discrete input (0-28 V) to the FCCs or force it to a weird, random behaviour. Since FCCs are modelled as embedded Matlab Function Blocks, they just receive components inputs (relays, sensors, etc.). Thus, FCCs' logics must estimate the system status. There are actually 19 different possible failures modelled in the Status model.

Failures are grouped in various classes: FCCs failures, LDG and NWS failures, WBS failures and FS failures. One FCC may fail, the landing gear actuators may block or slow their movements, pumps and sensors can block or freeze. Since all components are blocks, modelled outside the FCCs core function, their model can be changed freely to simulate the malfunction. The UAV is designed to withstand a single failure without any system/function loss; by using the Status Model is possible to verify how the system reacts to single or even double failure (and eventually improve the logics in order to overcame some critical failures combinations). A key point of the Status Model is the integrated environment. Integration is important since a failure of the component of a subsystem may affect the control laws of another subsystem; in the Status Model all subsystems run simultaneously.

The validation activity of the control laws has led to the analysis of 30 failure scenarios, and various improvements of the control laws of the systems have been designed and implemented.

Outputs and post processing

A set of post processing tools has been developed to manage the results of a simulation. The Simulink model collects both graphical and textual information. Each subsystem plots the most important variables vs time in a Scope block. By analysing the plots is possible to monitor how the system has reacted to the operator's commands; system failures can be monitored too.

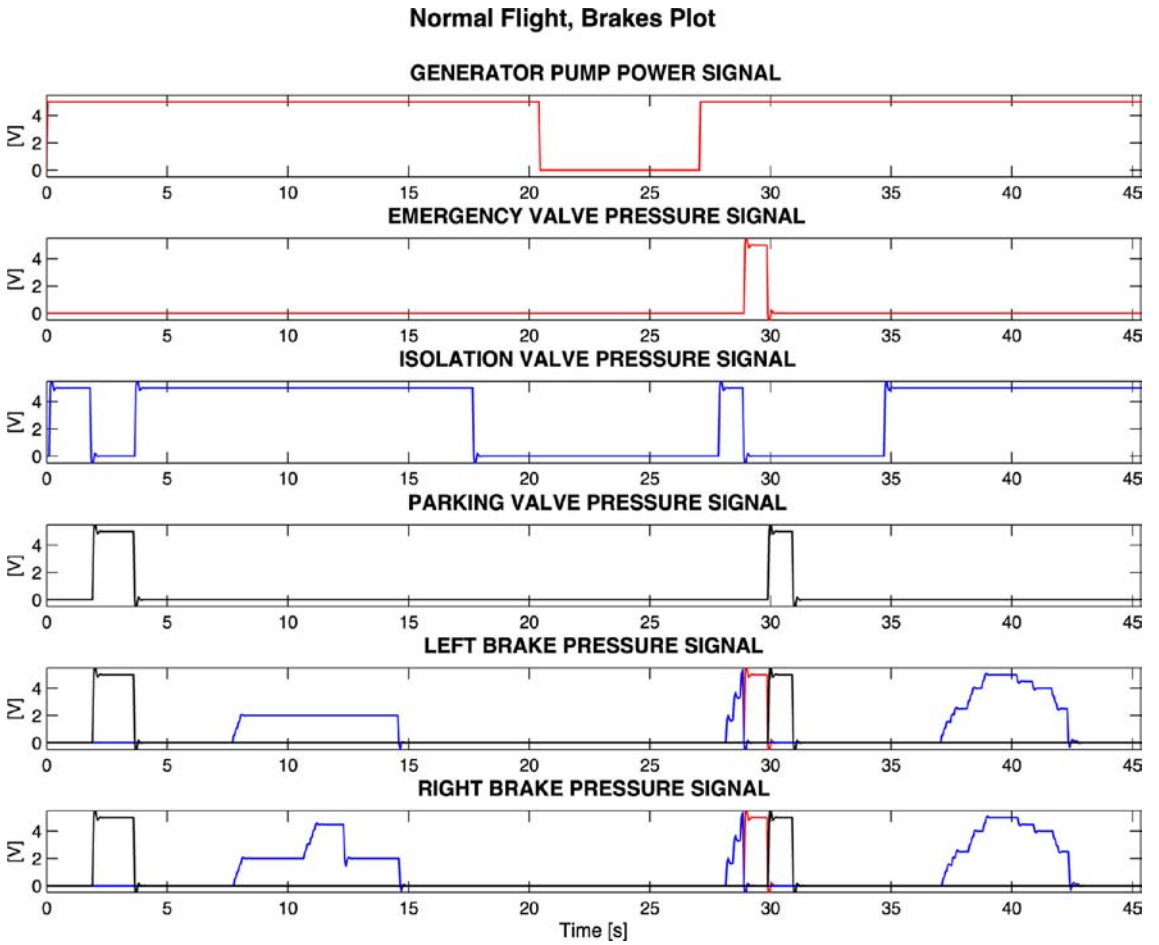
A Matlab routine saves all these plots in several formats, so that they can be easily attached to any document. Figure 10 shows an example: the hydraulic brakes plot. The plot depicts the braking system scope; it collects all the WBS vital information, such as: the generator pump discrete signal, the outlet pressures of emergency, isolation and parking valves, and the pressure (brake power) on the left and right brake.

Graphical plots combined with the textual log file offer a powerful summarizing tool to the test and validation purpose. The time history shown in Figure 10 shows a typical flight profile. FCCs initialise the system (verify it is powered on), and then send the Park Brake command. Park brakes are released shortly afterward, and the UAV starts the taxi operation. During taxi isolation valve is open, and some brake commands are sent to the left and right brake. Once the UAV takes off, the LDG is retracted and the brake system is powered off. Prior to landing, once the LDG extension sequence is completed, a built-in test (BIT) on the WBS system starts; the three braking modes: normal (33, 66, 100 per cent brake pressure) mode, emergency mode, and park mode are tested.

Conclusions

A new programmable logic controller development procedure has been defined and tested on a UAV. This method

Figure 10 Hydraulic brakes plot



is a valuable tool for specialists and software development team, which helps integration and validation of the system logics. Communication and data exchange between specialists may be supported by dynamic simulation video or test cases. The modular nature of the Status Model enables the creation of a customized-blocks library. Various Failure scenarios can be simulated and tested, reducing time and costs of a test-rig development. Finally the tool allows faster and more reliable controller logics development, integration and validation.

References

Chiesa, S., Corpino, S. and Medici, G. (2010a), "System programmable logic computer aided development procedure", *8th ACD2010 European Workshop on Advanced Control and Diagnosis*, pp. 229-34.

Chiesa, S., Farfaglia, S. and Viola, N. (2010b), "Design of all electric secondary power system for future advanced MALE UAV", *Proceedings of the 27th International Congress of the Aeronautical Sciences, International Congress of the Aeronautical Sciences*, p. 10.

Doss, M., Liebel, K., Lee, S., Calcagni, K. and Crum, R. (1996), "Migration of integrated modular avionics to space", *Proceedings of the 15th Digital Avionics Systems Conference* (*AIAA/IEEE*), *Atlanta, GA, USA*, pp. 131-7.

Farfaglia, S., Tranchero, B., Chiesa, S., Ragusa, C., Scavino, G. and Viola, N. (2009), "The SAVE project: hypothesis and investigation strategies for alternative energy based systems for MALE UAV", *20th National Congress AIDAA*, pp. 1-18.

Gilberl, J.G. and Diehl, G.R. (1994), "Application of programmable logic controllers to substation control and protection", *IEEE Transactions on Power Delivery*, Vol. 9, pp. 384-8.

ISO Standards (1985), "Documentation symbols and conventions for data, program and system flowcharts, program network charts and system resources charts, ISO 5807:1985", *Information Processing, ISO Standards Handbook 1*, International Organization for Standardization, Geneva.

Ji, K., Dong, Y., Lee, Y. and Lyoul, J. (2006), "Reliability analysis safety programmable logic controller", paper presented at SICE-ICASE International Joint Conference.

Keller, J.P. (2006), "Interactive control system design", *Control Engineering Practice*, Vol. 14, pp. 177-84 (Special Section on Advances in Control Education Advances in Control Education Symposium).

Loegering, G. and Evans, D. (1999), "The evolution of the global hawk and MALD avionics systems", *Proceedings of the 18th Digital Avionics Systems Conference*, Vol. 2, pp. 1-8.

Loh, R., Yi, B. and Roe, T. (2009), "UAVs in civil airspace: safety requirements", *Aerospace and Electronic Systems Magazine, IEEE*, Vol. 24 No. 1, pp. 5-17.

Lopez, J., Royo, P., Barrado, C. and Pastor, E. (2008), "Modular avionics for seamless reconfigurable UAS missions", *Proceedings of the 27th IEEE/AIAA Digital Avionics Systems Conference, DASC*, Vol. 1, pp. 1.A.3-1-10.

Morgan, M.J. (1991), "Integrated modular avionics for next generation commercial airplanes", *Aerospace and Electronic Systems Magazine, IEEE*, Vol. 6 No. 8, pp. 9-12.

Renfrow, J., Liebler, S. and Denham, J. (1994), "F-14 flight control law design, verification, and validation using computer aided engineering tools", *Proceedings of the Third IEEE Conference on Control Applications*, Vol. 1, pp. 359-64.

Zhang, H., Ning, J. and Schmelzer, O. (2000), "Integrated landing gear system retraction/extension analysis using ADAMS", paper presented at the North American ADAMS User Conference, June 19-21.

Further reading

Birbir, Y. and Nogay, S.H. (2008), "Design and implementation of PLC-based monitoring control system for three-phase induction motors fed by PWM inverter", *International Journal of Systems Applications, Engineering & Development*, Vol. 2, pp. 128-35.

Davidson, C.M. and McWhinne, J. (2000), "Engineering the control software development process", *Factory 2000 – The Technology Exploitation Process, Fifth International Conference*, Vol. 1, pp. 247-50.

Jeong-Woo, J., Ki-Chang, L., Don-Ha, H. and Yong-Joo, K. (2002), "Development of a dynamic simulator for braking performance test of aircraft with anti-skid brake system", *Proceedings of the 2002 IEEE International Symposium*, Vol. 2, pp. 518-23.

Lu, L. and Lei, J. (2010), "Design and reliability prediction of a distributed landing gear control system", *Aircraft Engineering & Aerospace Technology*, Vol. 82 No. 1, pp. 15-22.

About the authors

Giovanni Medici is a PhD student of Politecnico di Torino (Department of Aerospace Engineering) and Alenia Aeronautica. The main topic of his PhD is "Advanced control system: thrust vectoring on an UCAV". He is mainly involved in: propulsion, simulation, and fuel cells heat management. He graduated in 2009 with a thesis in collaboration with ETSIA Madrid. Giovanni Medici is the corresponding author and can be contacted at: giovanni.medici@polito.it

Nicole Viola has been working as an Assistant Professor at the Department of Aeronautics and Space Engineering, Politecnico di Torino, since March 2008. She had been working as Researcher on aeronautics and space systems design at Politecnico di Torino since April 2000. She obtained her PhD in Aerospace Engineering in 2004 on the "Conceptual definition of transatmospheric and space vehicles". She is author of papers published in books, journals and international and national proceedings. She is currently teacher of the course "On-board Equipments and Avionic Systems" of the third year of the first level degree in Aerospace Engineering.

Sabrina Corpino obtained her degree and PhD in Aerospace Engineering from Politecnico di Torino. She is now Assistant Professor at Politecnico di Torino in the field of Aerospace System Engineering and is involved mainly in the definition of design methodologies for both aeronautical and space systems and in RAMS techniques definition, development and application in the aerospace field. In particular, her main interest is the research in the simulation of aerospace systems with innovative methods such as hardware in the loop techniques for verification and validation of the design.

Marco Fioriti, PhD, graduated from Politecnico di Torino in Aerospace Engineering. He is working as a Research Assistant of Aircraft System Design at Politecnico di Torino and has been collaborating with Alenia Aeronautica Preliminary Aircraft Design Department for five years on Systems, Technology Innovation and Life Cycle Cost.