CAN-based modules for A320 flying simulators

Ana Antunes, José Sousa, Victor Antunes,
Instituto Politécnico de Setúbal / ESTSetúbal, Portugal

Flight simulators are used for entertainment or training purposes. These systems reproduce the cockpit of an aircraft as realistically as possible. Nowadays there is a thriving community of users of this kind of products and several companies that provide hardware blocks to build the cockpits. Traditional architectures of flight simulators use several personal computers and point-to-point connections to link the cockpit devices with the simulation software. This paper describes the development of CAN-based modules for A320 flight simulators. These modules were developed for the Virtual Aviation Systems company and comprise blocks for the throttle, engine panel, gear, flaps, spoilers, pedals, FCU and EFIS systems. All the modules provide transparent RS-232 and CAN connections in order to be used individually or in a CAN network. The application layer follows the CANaerospace interface specification and provides users with the ability to ask for the identification of the devices, to change the Node-ID, to configure some of the modules and to automatically configure the baud rate.

Training flight simulators are part of commercial airline operations since the 1960's because of both their training effectiveness and their safety [1]. The interest of entertainment flight simulators accompanied the evolution of personal computers and the decrease in their cost, making it possible for amateurs to have flight simulators similar to the ones used for training. The market of home aviation enthusiasts was recognized since the beginning of the computer industry with the advent of personal computers [2].

Early "flight simulation" programs were marketed as games and the user's input was limited to the keys of the computer keyboard. Within a few years the companies were selling flight sticks, control yokes and rudder pedals to be used as interface devices for the flight simulation programs. Visual systems for flight simulation have also evolved. More recently there are PC-based flight simulators designed specifically as instrument flight trainers for those that already have a pilot's certificate and there are others on the verge of games that are marketed as entertainment although providing serious flying features. Some of the more popular programs offer a variety of aircraft flight dynamics and visual representations allowing the user to define the aircraft model to be used and the visual environment and conditions in which to fly and their capacities exceed the power of full-scale flight simulators of the 1970s [2]. The market followed the users desire for realistic fly operation conditions and started to provide improved methods of interface and visual conditions. Currently there are add-on systems to enhance the visual system of the existing PC-based flight simulators or to add additional flying characteristics. Methods of interface with the user have also evolved providing tactile feedback, emulating the behaviour of the real cockpit instruments.

With the decrease of the prices of electronic hardware and the increase of the processing capacities of modern personal computers amateur pilots are buying more hardware interface devices to emulate the different instruments in a cockpit and by doing so, increasing the complexity of these flight simulator systems.

The electronic I&D group at the Polytechnic Institute of Setúbal was approached by a local company in the field of entertainment flight simulators to develop the hardware for the interface modules of the cockpit of an A320 entertainment aircraft. The commissioning included the mandatory feature of the modules being as similar as possible to the real instruments both in the visual and in the behavioural aspects.

This paper describes the development of CAN-based interface modules to assemble a realistic cockpit for entertainment A320 flight simulators. The modules developed so far are the following: throttle and engine panel, gear, flaps, spoilers, pedals, FCU and EFIS.

The remainder of this paper is organized as follows: section 1 provides a description of traditional entertainment flight simulation systems; section 2 describes two alternative CAN-based architectures; section 3 presents the generic features of the modules; section 4 presents the CANaerospace implementation; section 5 described the local gateway CAN-to-RS-232 feature of the modules; section 6 describes the modules and section 7 concludes this paper.

Traditional flight simulation systems

Traditional entertainment flight simulation systems use a personal computer to run the flight simulation program, other personal computers to run add-on software, usually providing additional visual details and flying features, and interface with several hardware devices that emulate the real instruments of the aircraft cockpit. The personal computers are usually connected using a client-server Ethernet connection. The server runs the main flight simulation software and the clients run the add-ons. The hardware devices that provide the interface with the pilot contribute decisively to the realism of the simulation. Usually the simulator is built in incremental steps and is assembled with different blocks from different manufacturers. These blocks are traditionally connected point-to-point with the main computer using a serial connection either RS-232 or USB.

Figure 1 presents the block diagram of the traditional entertainment flight simulator architecture.

The point-to-point connections used to connect the interface devices with the main computer increase the amount of cabling and limit the expansion of the system.

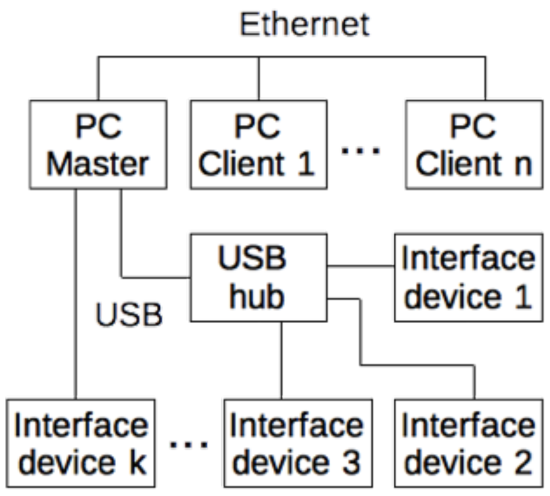

Figure 1: Block diagram of the architecture of a traditional flight simulator system

A CAN-based flight simulation architecture
In order to simplify the cabling structure and to take advantage of the other well known characteristics of distributed systems like modularity, better resource utilization, incremental growth and easier reconfiguration, diagnostic and repair, the hardware interface modules that were developed result in a flight simulator architecture that uses a CAN bus [3][4][5] to connect all the interface devices. Figure 2 shows the block diagram of the architecture that was developed.

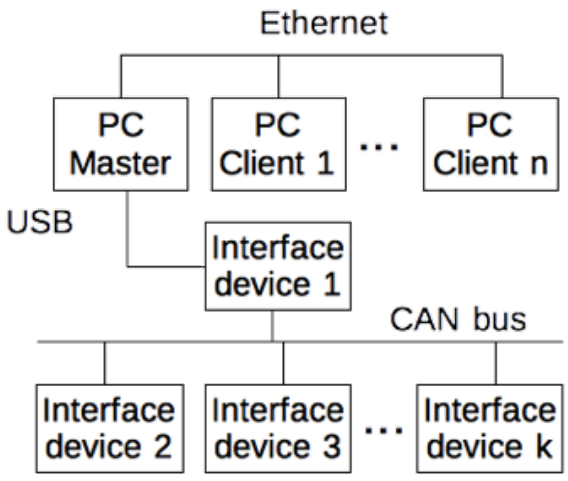

Figure 2: Block diagram of the architecture proposed for the flight simulator system

This architecture uses a gateway CAN to Ethernet to connect the interface devices' CAN bus to the main simulation computer. It is adequate for systems with a large number of interface devices that generate a large amount of traffic in the CAN bus.

In order to reduce the costs of the system an alternative architecture was developed adequate for systems with less interface devices, that generate an amount of traffic that can be dealt with using a USB connection. This architecture is presented in figure 3.

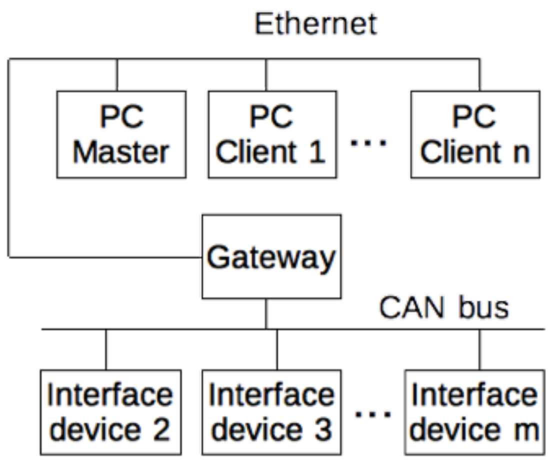

Figure 3: Block diagram of the alternave architecture proposed for the flight simulator system

This architecture relays on the modules' gateway CAN to RS-232 features. The connection between the modules and the computer's USB ports can be made using a RS-232/USB conversion cable that is inexpensive and widely available.
The architectures proposed can only be applied when all the interface devices have appropriate CAN features.
The modules that were developed can support both of these architectures.

Generic features of the modules

Because amateur pilots usually assemble their cockpits using different modules from different manufactures the modules should be able to work efficiently both in a CAN bus or standing alone.
They should also be able to communicate using an adequate message format preferably standardized or well known in the aviation domain.
All the modules are issued with a bidirectional gateway CAN to RS-232 in order to provide the ability for the modules to be connected directly to the main computer using a serial interface or to be connected in a CAN bus.

Concerning the application layer and, given the domain of application of the modules, the choice was made to use the CANaerospace specification [6]. Other flight simulator manufacturers made the same option [7].

CANaerospace implementation

The CANaerospace specification defines 7 message types. Their identifier ranges are shown in table 1.

Table 1: CANaerospace message types

Message type	Identifier range
Emergency Event Data	0x000 - 0x07F
High priority Node Service Data	0x080 - 0x0C7
High priority User-Defined Data	0x0C8 - 0x12B
Normal Operation Data	0x12C - 0x707
Low priority User-Defined Data	0x708 - 0x76B
Debug Service Data	0x76C - 0x7CF
Low priority Node Service Data	0x7D0 - 0x7EF

The CANaerospace default identifier distribution for normal operation data messages was used whenever possible and user-defined identifiers of the low priority user-defined data range where attributed to the new messages that had to be created.

In the CANaerospace specification the general message format uses four bytes for the header and four bytes for message data. The standard header format is shown in figure 4.

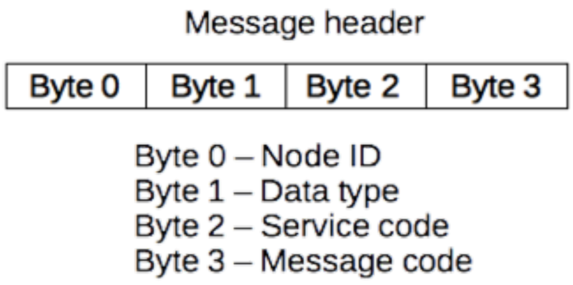

Figure 4: Standard CANaerospace message header structure

The node-ID is the local node identifier that is in the range [0,255]. The broadcast identifier has node-ID equal to 0.
The data type specifies the coding of the message data according to the specification.

For normal operation data messages the message code is incremented for each message, allowing the receiver to order the messages upon reception.
The CANaerospace node service protocol provides a connection-oriented communication using a handshake mechanism [6].
The modules implement four node services. These services are listed on table 2.

Table 2: Node services implemented

Node Service	Service Type
Identification	Protocol defined and mandatory
Node-ID setting	Protocol defined
Module configuration	Protocol defined
Automatic baud rate configuration	User-defined

Following the CANaerospace specification the services are requested using channel 0 and the responses are sent on the same channel.
The mandatory CANaerospace node identification service is used to request a “sign-of-life” response as well as configuration information of the addressed nodes.
The node-ID setting service is used to set the local node-ID number.
The module configuration service was tailored to set the type of service that a module performs and is used in the modules based on a multifunctional hardware board. This service is intended for manufacturer configuration. It is always used in the initial configuration of the multifunctional modules and can be used sporadically to change the function of an already operational module. This procedure requires knowledge of the existing hardware and to prevent improper configurations the service is secured with a password.
By default the modules are set with a baud rate of 125 kbps.
The automatic baud rate configuration service is user-defined and was developed to allow for the modules to adapt automatically to an existing network testing for the most used baud rate settings of 125 kbps, 250 kbps and 500 kbps.

Local gateway CAN-to-RS-232

The modules were developed to be as generic as possible.
They were provided with a bidirectional gateway CAN-to-RS-232 in order for them to be used alone outside of a CAN network (in the traditional architecture presented in figure 1) or in the architecture presented in figure 3 where the connection to the main simulation computer is made using a serial connection.
This gateway is not fully transparent and does not automatically route all traffic to the other network. It was tailored to support the CANaerospace specification in a way that extends the CANaerospace network to the main computer.
Node service messages are sent across the gateway when the node-ID addressed by the service is different from the node-ID of the module, or when the node-ID is 0 (the broadcast node-ID). If the node-ID addressed is the same of the module the service is delivered and the message is not routed across the gateway.
Other messages are always routed across the gateway.
Normal operation data is always checked in the modules.

Modules’ description

The modules are used to reproduce the features of the different sections of the cockpit of an A320 aircraft. They implement a control system that is used to collect information from different kinds of sensors like encoders, rotational selectors, keyboards, tactile or toggle switches, and actuate upon LEDs, 7-segment displays, step motors, etc. Potentiometers are used to sense the position of certain devices as analog variables.
Some of the modules act only as output devices and others as input/output devices.
The control actions are usually taken by the simulation computer. Nevertheless some modules have local controllers to control specific hardware like displays or stepper motors.
The modules’ hardware architecture is based on a PIC microcontroller with CAN features. The microcontrollers’ ports are used to connect sensors and actuators using the appropriate signal conditioning circuits. In some of the modules a matrix structure is used to scan keyboards or to actuate LEDs. Communication is provided through the CAN bus and through a RS-232 serial connection.

The power circuit was designed to use 12 V power sources. The block diagram of the modules' architecture is presented in figure 5.

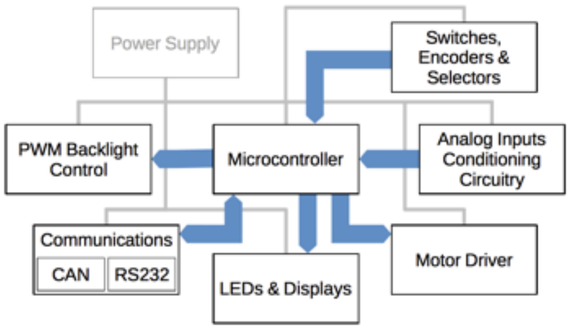

Figure 5: Block diagram of the modules' architecture

Initially the design focused on the modules in the pedestal of the cockpit.
After analyzing the requirements of the modules related to the pedestal it was decided to develop a printed circuit board (PCB) that includes all the hardware circuits needed to implement the functions of the following modules: throttle, engine panel, gear, flaps, spoilers and pedals. Hence this is a multifunctional common board that uses the module configuration service to configure its function by software. The configuration is made by the manufacturer prior to the assembly of the module's casing.
A picture of the PCB of the common board is presented in figure 6.
A user-defined data message was created to control the backlight of the pedestal. This message is recognized by all the modules based on the common PCB and has the CAN identifier 0x708.

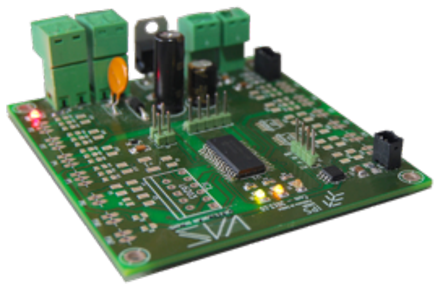

Figure 6: PCB of the common board

In the flowing paragraphs the modules of the A320 cockpit that use the common board will be described.
A picture of the flaps module is presented on figure 7.

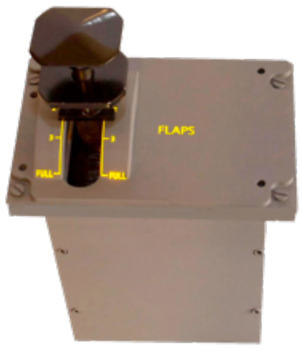

Figure 7: The flaps module

This module is an output module. It actuates on the flaps of the plane that increase the lift of the wings of the plane.
It uses the normal operation data message Flaps Lever Position with the CAN message identifier 0x1AE.
A picture of the gear module is presented on figure 8.

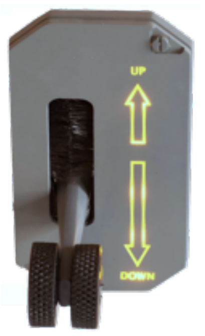

Figure 8: The gear module

This module is an output module. It actuates on the landing gear of the plane.
It uses the normal operation data message Gear Lever Switches with the CAN message identifier 0x497.
A picture of the spoilers module is presented on figure 9.

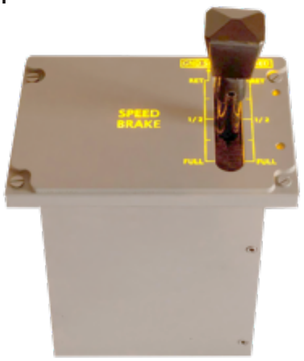

Figure 9: The spoilers module

This module is an output module. It actuates on the spoilers of the plane that are devices that reduce the lift of the wings of the plane.

It uses the normal operation data message Speedbreak Lever Position with the CAN message identifier 0x1B1.
The throttle module is an input and output module. This module comprises three devices: the throttle levers, which control the engine power, the trim wheel, which is used to adjust the vertical stabilization of the plane, and the engine start panel which includes the commands for engine startup.
A picture of the throttle module is presented in figure 10.

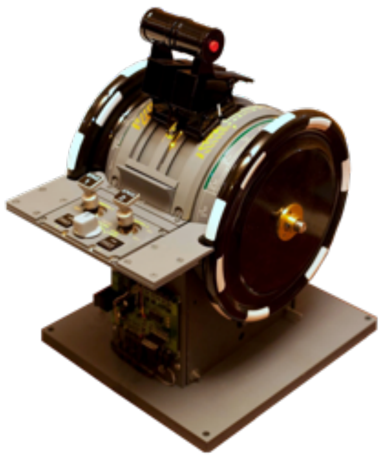
Figure 10: The throttle module

The messages used by this module are presented in table 3.
In table 3 NOD refers to the normal operation data messages defined by the CANaerospace specification and UD to user-defined data messages.

Table 3: List of messages used by the throttle module

Message name	Type	ID
Engine 1 throttle lever position	NOD	0x19E
Engine 2 throttle lever position	NOD	0x19F
Pitch control position	NOD	0x190
Engine 1 status 1	NOD	0x22C
Engine 2 status 1	NOD	0x22D
Autothrottle	UD	0x70A
Engine start control	UD	0x70C
Trim input	UD	0x70B
Trim position input	UD	0x716
Trim configure	UD	0x717
LED status 1	UD	0x70D
LED status 2	UD	0x70E

A picture of the pedals module is presented on figure 11.

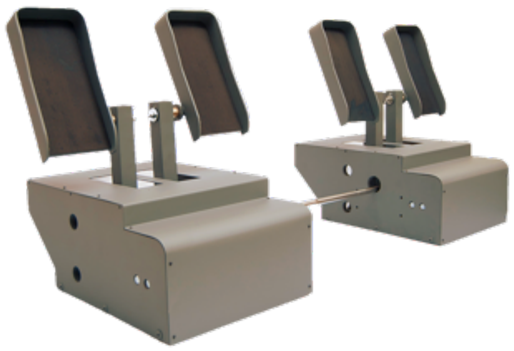
Figure 11: The pedals module

The pedals module is an output module. It actuates on the rudder that is the vertical surface use to control the plane's yaw axe and the wheel brakes.
The messages used by this module are presented in table 4. All of them are normal operation data messages.
The former modules belong to the pedestal section of the cockpit except for the gear module, which belongs to the forward section.
Another cockpit section is the glareshield. Two modules were developed for this area: the FCU and the EFIS modules.

Table 4: List of messages used by the pedals module

Message name	ID
Pilot left break pedal position	0x1B3
Pilot right break pedal position	0x1B4
Copilot left break pedal position	0x1B5
Copilot right break pedal position	0x1B6
Rudder position	0x132

Due to the complexity of the hardware required three dedicated PCB boards were designed: one for FCU and two for the EFIS (pilot side and flight officer side). The pictures of these boards are shown on figure 12 and figure 13.

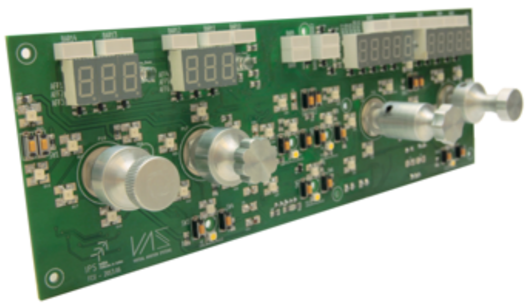
Figure 12: PCB of the FCU board

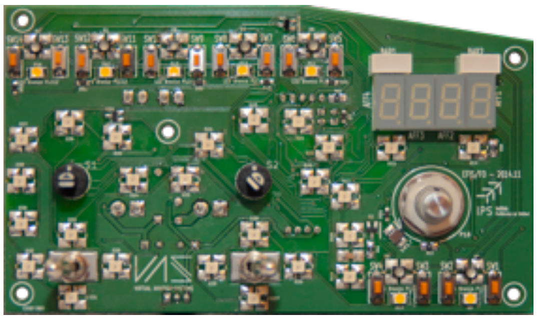
Figure 13: PCB of the EFIS board

Another user-defined data message was created to control the glareshield backlight with the CAN identifier 0x70F.
A picture of the FCU module is presented on figure 14.

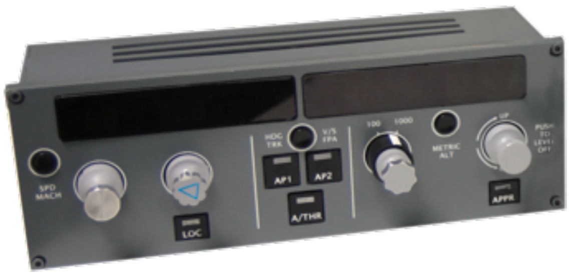
Figure 14: The FCU module

The FCU module is an input and output module. FCU stands for Flight Control Unit which is used to set flight parameters like heating, speed, altitude, vertical speed as well as to control the auto-pilot and auto-throttle features of the plane. Actions are input by the pilot through push-pull encoders and tactile switches and results of these actions are visualized by LED indicators and numerical displays.
The messages used by this module are presented in table 5.

Table 5: List of messages used by the FCU module

Message name	Type	ID
True airspeed	NOD	0x13C
Mach number	NOD	0x13E
Heading number	NOD	0x141
True altitude	NOD	0x14C
Computed vertical velocity	NOD	0x464
Selected glidepath angle	NOD	0x462
Speed or mach display	UD	0x711
Heading display	UD	0x712
Altitude display	UD	0x712
V/S or FPA display	UD	0x714

A picture of the EFIS module is presented on figure 15.

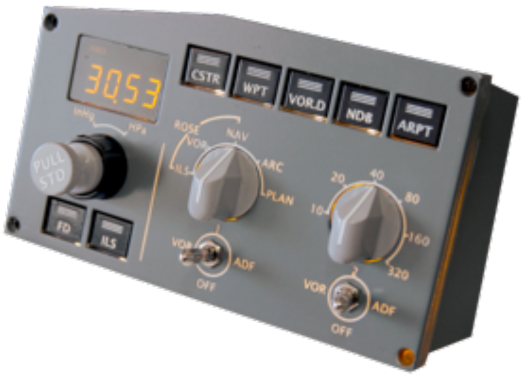
Figure 15: The EFIS module

The EFIS module is also an input and output module with interfaces similar to the ones used in the FCU.
EFIS stands for Electronic Flight Instrument System and is used to input commands to the primary flight display, like the type of information it shows and its range. It is also used to set the reference pressure that is used in the altitude calculations.
The messages used by this module are presented in table 6.

Table 6: List of messages used by the EFIS module

Message name	Type	ID
Baro correction	NOD	0x13F
EFIS status input	UD	0x71C
EFIS status output	UD	0x71D
Baro reference display	UD	0x71E

Conclusion

This paper presented the development of hardware modules for the cockpit of an A320 aircraft entertainment simulator.
The modules can be used either in a CAN network of interface devices or as standalone interface devices directly connected to the main simulation computer.
At this moment the modules for the flaps, gear, spoilers and throttle are already being commercialized and the FCU and EFIS modules are in the final phases of test and production.
Simultaneously other modules are currently being developed in order for our client company to be able to purpose a complete A320 cockpit as soon as possible. Examples

of the modules modules under development are the side panel (both pilot and flight officer sides), parking brakes, ECAM (Electronic Centralized Aircraft Monitor) and switching panel. Some of the boards are presented in figure 16.

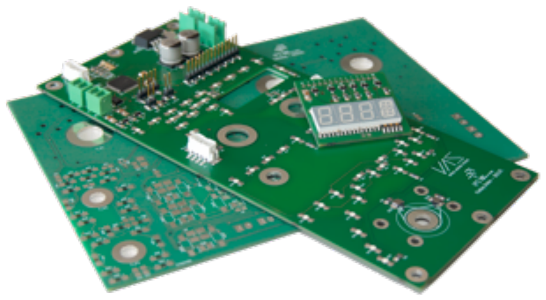

Figure 16: Some of the modules under development

References

[1] Page, Ray L., A brief history of flight simulation, SimTecT Proceedings, 11-17, 2000.

[2] Koonce, J.M., Personal Computer-based flight training devices, The International Journal of Aviation Psychology, 8(3), 277-292, 1998.

[3] Lian, F-L, Analysis, design, modeling and control of networked control systems, PhD thesis, University of Michigan, 2001.

[4] Jianyong, Y., Shimin, Y. and Haiqing, Survey on the performance analysis of networked control systems, Proc. of the 2004 International Conference on Systems, Man and Cybernetics, 5068-5073, 2004.

[5] Road vehicles - Controller Area Network (CAN) – Part1: data link layer and physical signalling, ISO 11898-1, ISO, 2015.

[6] CANaerospace Interface Specification for airborne CAN applications V1.7, Stock Flight Systems, 2006.

[7] CANaerospace/AGATE data bus, Presentation, Michael Stock Flight Systems, 22, 2005.

Ana Antunes, José Sousa, Victor Antunes
Instituto Politécnico de Setúbal,
Escola Superior de Tecnologia de Setúbal
Campus do IPS, Estefanilha
2914-761 Setúbal, Portugal
Tel.: +351265790000
Fax: +351265790043
ana.antunes@estsetubal.ips.pt
jose.sousa@estsetubal.ips.pt
vitor.antunes@estsetubal.ips.pt
www.est.ips.pt