Defining scenarios in home and building automation: CANopen and the virtual entities approach

Francisco J. A. Cardoso, Universidade de Coimbra, Coimbra, Portugal
Paulo M. R. Falcão, Instituto Pedro Nunes, Coimbra, Portugal
Luís M. A. Oliveira, Instituto Pedro Nunes, Coimbra, Portugal
Nuno J. M. Rocha, Instituto Pedro Nunes, Coimbra, Portugal
Alejandro N. Cruz, Instituto Pedro Nunes, Coimbra, Portugal
Bruno J. F. Ribeiro, Universidade da Beira Interior, Covilhã, Portugal

The approach to both the architecture and the underlying technologies of a highly distributed systems platform for home and building automation is presented here. This concept is based on the capability to integrate a number of deeply embedded devices that share information among themselves and act as a single system, extracting knowledge from the data collected at the various locations and using this information to intelligently react to events and circumstances in the surrounding area, so as to economically provide comfort, safety and security to people working or living in that controlled environment. Special emphasis is put on the human interfacing to technology, by lifting up the traditional focus from the level of applications to that of services, therefore aiming at serving user's needs, but relieving them from low-level platform concerns. Middleware is used to address the issue of development barriers for services, firstly by providing means for graphical programming in order to rapidly create in-house living scenarios (i.e., complex applications, involving different, interacting variables and conditions) by simply drawing a block diagram that, with recourse to a library of pre-constructed models, addresses the various scattered units and invokes the respective local actions. Units in the system are of two different kinds, as they fit in electrical switchboards (where are concentrated the protective devices to power circuits) or, alternatively, are dispersed throughout the buildings, in relatively larger quantities. The former being integrated over CANbus, and the latter over a wireless cluster tree network based on IEEE 802.15.4 (ZigBee) technology, a unified approach to high-level remote procedure calls was attempted, thus encompassing both wired and wireless networks, which, being based on CANopen high-level protocol, allows a more effective co-operating service-computing platform.

1. Introduction

Market response to products on offer for home and building automation has been relatively poor so far, clearly falling short of industry's expectations *vis-à-vis* serious market forecasts, especially in the home automation business. This fact has impaired investment, thus slowing down the pace of product developments and, all very naturally, the market reached a point close to stagnancy. Recently, the diagnosis of such a standstill was undertaken by a number of researchers and industrialists, especially throughout Europe, so as to bridge the gap between offer and demand and, eventually, realize the potential of a market of such a huge dimension.

In line with the perspective above, the rationale and the results put forward in the present paper are the outcome of co-operative work with a firm operating in the area of information technologies, and, in that capacity, with strong links to real estate promoters and contractors. This very much allowed a constructive blend of different knowledge as much as a subtle balance of the two traditional trends of technological push and the market pull, thus leading to what we believe can be a significant step ahead in this line of work: focus on common housing as the main target market, instead of the *élitist* niche market of top and upper-middle class buildings. Actually, we believe that such a shift in the underlying marketing strategy, seeking the respective 'democratisation',

is the crucial issue in the process of changing this industry, an idea that could be realized only through a far better adequacy of products to people's needs and purses.

The new objective of addressing a mass-market had serious and challenging implications in both functional and technical system specifications, which substantiated and led to a significant expansion of the very concept of home and building automation. Thus, this perspective has led us to firstly address the following key factors: (i) the use of highly distributed systems, made of small, cheap, easy to install and use units, (ii) operational flexibility, so as to rapidly adapt existing resources to changing requirements, without having to resort to computing skills, and (iii) straightforward human perception of who actually masters the house, which is a complex feeling very much dependent on the amount of trust in the technology there involved, and on the degree of friendliness provided by the system(s) in order to support human interaction – all issues above contributing to the ability of "mass customisation", which is required in order to meet the new challenge.

The above factors benefit from the level of modularity that can be practiced in integrated designs. Therefore, our systems platform was structured with special attention to modularity rules, which were used to define and articulate modules. Thus, the whole system was designed conceptually as a set of inter-communicating tasks, which are logically grouped into functional sub-systems that are mapped onto physical processor modules according to their processing requirements. Each task executes in a private protected environment and implements a single abstraction for the rest of the system. Messages being used by modules to request services from other modules, a message system provides an efficient mechanism for passing information between processors and synchronising local processes, and encourages strong logical separation between tasks.

Such considerations inspired the implementation of a universal platform for the purpose of easily supporting a multitude of application cases. Thus, control tasks are to be realised with existing hardware and software modules, and applications setting-up can be carried out with no further development of applications software, so as to keep the standard of software modularity compatible to the one more easily obtainable for hardware. Actually, this line of concern addresses the specific issue of how procedures can be remotely invoked throughout the different unit levels, thus involving both code allocation criteria and the communications media to be used, which were dealt with as follows: (i) concerning code storage, executable code corresponding to the possible role of each module, according to its hardware contents, resides permanently in non-volatile memory there located – hence, in the course of initialising an application, short message strings are passed along the system, each message unit consisting of a command code and the respective parameters – and, (ii) the integration of a number of independent modules in a distributed system, usually deeply embedded, led to the choice of CAN serial bus as a primary communication mechanism adequate to link the different units in a system, irrespective of time stringency and distances involved, even though complemented by short-haul, low-power, wireless networking.

These decisions implied the use of CANopen high-level protocol, which proved to be a very suitable tool to support the communication semantics required by what then became a complete strategy of virtual instrumentation.

The rationale and the solutions found in order to tackle the string of problems that have arisen from that new general perspective are embodied in a full systems platform, which is described next in terms of both architecture and technologies.

2. The platform

Typical applications in this line of work impose stringent requirements regarding both functional flexibility and time management, thus reflecting the potential diversity of variables involved, as much as

the diversity of people and their circumstances. In order to cope with this common difficulty and, still, keep costs at a reasonable level, a highly modular system was devised, implemented, and used under common working conditions.

2.1. Architecture

Topology was determined by (i) a primary decision of creating a sensor/actuator wireless network, consisting of very small nodes that can independently self-organise, as they equipped with modules for data acquisition and processing, communication and power supply, and (ii) the fact that power circuits concentrate in electrical switchboards, where are located the respective protective devices, making it sensible to fit the respective control units in there, as well. In this manner, a wired backbone network connecting a few units (relatively large, in both size and I/O capabilities) located in the switchboard(s) links to a myriad of tiny units that are dispersed throughout the house, in large quantities. Criteria of functional partition and allocation were affected also by these considerations and, therefore, the overall system consists of a number of fully autonomous units, which are briefly described as follows:

- The central unit, a PC/Windows-based machine, which is responsible for data gathering, archival and presentation, as well as for interactive human interfacing (applications programming and follow-up);
- Switchboard-integrating units which, besides the relatively large I/O handling capabilities – 24 DO, 32 DI, and 4 AI – also consolidate and appropriately route the information to/from lower level units, through CANbus;
- Field-integrating units, fitting in sensors for temperature, relative humidity, and light, in addition to which provide general purpose local I/O handling capabilities – 6 DO, 4 DI, and 2 AI – also consolidate and appropriately route the information to/from other units, through both CANbus and ZigBee and, thus, can bridge communication between the two networks;
- Smart transducers and actuators, fitting in sensors for temperature, relative humidity, and light, plus some more limited general purpose local I/O handling capabilities – 2 DO, 4 DI, and 2 AI – and one single communication device of either type, CANbus or ZigBee, in which latter case can act as master or slave.

The key to system architecture is a basic cell structure comprising one unit of a certain level and a number of units of the immediate lower level, which adequately replicates so as to encompass all variables and devices in each and every application. Embedded communication within a cell is carried out so that (i) lower level units do not communicate with each other, and (ii) the top-level unit in the cell also acts as gateway to the respective network (of either type), at the next higher level. Thus, this dual-ported operation facilitates a multi-buffering scheme, which ensures constant-rate data flow across the whole system and, therefore, predictable time in either case of remote data access and I/O human-triggered operation, as required by real-time operation.

Another key factor for the success of this design was the basic choice of communication technologies to support the different communication needs throughout the system: (i) CANbus, a fieldbus technology that is particularly suitable in cases like this one, where the network hierarchy implied in the architecture serves the purpose of distributed computing at device level, besides its outstanding capabilities of error detection, thus leading to increased reliability, real-time response, simplicity and low cost, and (ii) ZigBee-inspired, IEEE 802.15.4 wireless embedded networking platform, allowing flexible configuration (topology), high security, low cost, long battery life, reasonable data rate, and effective network management, thus producing networks that are reliable, flexible, secure, and easy to use.

Both communication mechanisms being specified at the two lowest levels of ISO/OSI only, CANopen was considered in order to support a complementary role consisting in addressing functions whose resources (both hardware and software)

are already made available in the modular units. Such a high-level approach lends to minimising the effort involved in system integration by adopting the concept of device profiling, thus standardising the communication contents (both syntax and semantics) that are to be transferred to each other.

Being a natural add-on to CAN networks, CANopen model to describe devices (including functionalities and the mechanism of communication to explore them) was extended to the wireless network, with the logic adaptation to the particulars of a different architecture, especially to accommodate message routing.

2.2. Technology

Both hardware and software supporting technologies were selected and implemented so as not to compromise the final outcome, vis-à-vis the ultimate objectives and the system architecture previously defined.

Thus, starting from the top central unit, it was based on a commercial PC, for its role mainly consists in providing extensive historical data storage, and supporting the appropriate local human interfacing to the overall system for in-house operation, as well as a number of web services required for remote operation. All other units referred to above are built using microcontrollers of the latest generation as basic building blocks, to the benefit of size, cost, reliability, power consumption, and simple and effective overall integration, in result of the commonly available software development tools of good quality.

In fact, especially owing to the paraphernalia of built-in devices in new microcontrollers, these tend to be more reliable and compact, as well as efficient and versatile. In this line, the STMicroelectronics ST10F269 device appears as a paradigmatic case, especially as it integrates five major facilities: (i) “intelligent” interfacing to field variable signals through capture/compare inputs and PWM driving outputs, which, when duly associated with internal timer units, can be used for plain timing, pulse width measurement, event counting, etc., and one analogue data acquisition chain including a 16-input analogue multiplexer, sample-and-hold and 10-bit ADC, (ii) two CANbus controllers, implementing low-level (physical and data-link layers) standard protocols in hardware, (iii) FLASH memory in generous amount (256 kbyte), thus providing the long sought flexible (on-site read/write), non-volatile memory for both programs and data, plus 12 Kbyte of RAM, (iv) real-time clock, so that tasks can be synchronised without the intervention of the central unit, and (v) hard real-time capabilities deriving from the built-in multiple priority interrupt manager that allows ‘clean’ prioritising of interrupt sources, and by fast context switching in response to interrupts through increased flexibility in register bank handling.

Therefore, the ST10F269 chip was selected to support all switchboard-integrating units, for these are always involved in real- and limited-time operation, handling locally a wide range of I/O, and communicate over CANbus. Also, as a ‘relative’ of the well-known Infineon C166 microcontroller family, software could be developed with recourse to a set of tools (assembler, C and C++ compilers, and debuggers) from TASKING, Inc. Finally, in what concerns the physical implementation of switchboard-integrating units, these being devised so as to easily fit in electrical switchboards, the electronics of this type of unit lay on a single board with the footprint of 8x16 cm, already including the respective fast-on connectors, which is boxed so as to clip on DIN-rails.

In addition, in order to support the embedded units other than switchboard integrators, the Texas Instruments MSP430F149 device was selected, mostly for it qualifies as ultra low-power device (<1 mA @ 3.3 V, in active operation) and, therefore, greatly contributes to extend battery life in battery-fed smart sensors and actuators, and the 8-input analogue multiplexer in connection with a 12-bit ADC, 2 kbyte of RAM, and 60 kbyte of FLASH memory (with flexible handling for code changes or data logging), make it perfectly suitable for supporting either type of field-integrating units and smart

sensors/actuators. The IAR Embedded Workbench suite of tools, from IAR Systems AB, was very successfully used to support software development in all units sharing the same microcontroller technology.

In order to reduce the overall size of both unit types, they are implemented with recourse to tiny PCBs. Thus, in case of smart sensors/actuators, 3 different PCBs _ made circular, of 4 cm diameter, to best adapt to wall boxes that are part of existing electric circuitry _ are stacked together, that can be described as follows: (i) the 'motherboard', thus containing the microcontroller, on-board sensors, and supply conversion and management devices, (ii) one board with interfacing devices to field signals, and (iii) one board with the appropriate communication controller of either type, ZigBee (CC2420, from Chipcon) or CANbus (MCP2510, from Microchip Technology). Field-integrating units share the same approach to building blocks, with the minor exceptions that the 'motherboard' here is shaped as rectangular (4.5 x 6 cm) and permanently includes a CANbus controller, the add-on communications board being required only in case of integration in the wireless network.

2.3. The real-time operating system

In what concerns software structuring, a mechanism for the appropriate handling of software tasks running in each of the different system modules was also designed, so as to keep up with the hardware structuring in what concerns the level of modularity. Thus, a package of software was designed and implemented with the objective of providing support to the basic functions of both resource and task executive management at each unit in the distributed system (except the top central unit), with respect for the stringent time constraints affecting the majority of tasks running there. This software is, therefore, a real-time operating system that was specially devised to fit in the small amounts of memory that are available in embedded units.

This operating system is based on a kernel that was conceived and developed in order to support multi-tasking, pre-emptive task management, and was organised as a collection of small modules structured functionally. As a result, users can have it duly fit to their specific needs concerning both the available functions and resources.

The kernel in our operating system provides the classic primitives for inter-task synchronisation, communication, and resource sharing. Furthermore, the kernel constitutes a platform for embedded software in the form of multiple, independent and concurring tasks, which provides these tasks with their private executing environments. Three levels of processing have been devised, as shown in figure 1: the interrupt level, the system logical level, and the task level. Execution priorities in the interrupt level depend only on the CPU architecture, and they are statically assigned in the course of applications programming. Priorities at task level were made to vary between 1and 62, the priority level 1 being the highest. Level 0 was assigned to an "internal" system task, called *system logical level*, which integrates the intermediate processing level.

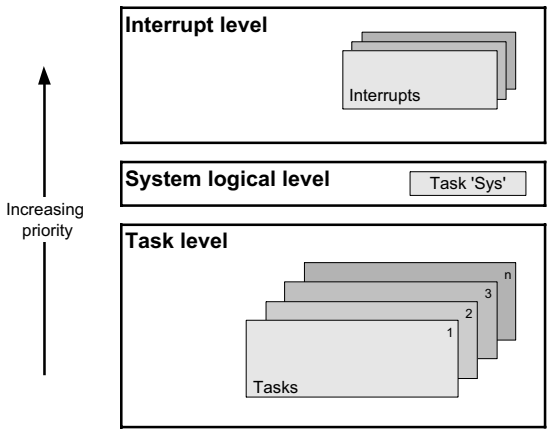

Figure 1. Processing levels in the RTOS kernel

In fact, in order to tackle the complex software in real-time units, namely those involved in data gathering and control functions, it is sensible to split the whole application into small task units, which are to be executed according to their specific time constraints, in a sequence duly managed at the level of the operating system kernel. So, here, besides its contents of executable code and data variables, each task is allocated its own

virtual CPU, with specific program counter, general-purpose registers, and stack pointer. Situations may exist where the same code segment serves different concurring tasks.

A simple mechanism of management was implemented here, with few straightforward rules, for the sake of both speed and reliability. Thus, in the progression of one given task, it skips from one state to another according to its own logical activity. As shown in figure 2, a three-state scheme was devised here, involving four state transitions, which proved to be both efficient and reliable in multi-tasking operation. As depicted there, transition 1 corresponds to a situation where the task cannot continue and, therefore, a system call ought to be made for it to be declared as *blocked*. Transitions 2 and 3 are both induced by the scheduler: transition 3 corresponds to a pre-emption of the current task, and transition 2 occurs whenever the task under consideration is promoted to the active executing state, thus replacing the previous one in that same situation. Transition 4 is triggered by a specific condition indicating either an external event or the availability of a given recourse, which, in any case, kept the task pending. Finally, whenever there is no other task executing, or this considered task is of the highest priority, transition 2 occurs immediately, as a default case.

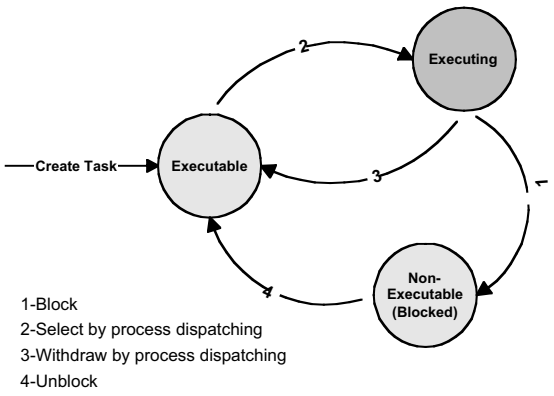

Figure 2. State-transition diagram of one single task

This simple, yet effective and reliable, task management scheme, along with the hardware structuring and the basic communication mechanism, paved the way for an overall well-built platform to suit the needs for the remote programming of a number of embedded controllers that are involved in the setting-up of in-house living scenarios, i.e., complex applications, encompassing interacting variables and conditions, at different locations.

3. Applications programming

The ultimate objective here consists in the capability to rapidly set-up scenarios in a house _ by definition, a *scenario* provides a combined ‘space-time’ understanding of the environment, which maps on a set of factors and parameters of environmental, temporal and personal nature.

The ability to effectively achieve such a goal depends on the implementation of three major technological services, as follows:

1. In the first place, provision for ordinary people to specify a set of conditions and/or operating points for environmental factors, which determine the way of living in a certain space, at a certain time, with recourse to a library of pre-constructed models, thus in line with the virtual entities approach _ specification is given graphically at the central unit, where icons representing physical modules and logical functions are associated with graphs that represent the logical links between them, altogether constituting a meaningful chain of events and functions;
2. Still at the level of the central unit, allocation of functions to the various units dispersed in the environment, within the framework of a programmed scenario, and remote invocation of the respective local actions over the networks;
3. Involving all units performing functions to the benefit of a programmed scenario, co-ordinated interaction, over the networks, amongst the scattered units integrating the system, so that they can decide upon and implement the appropriate actions in due time (task synchronisation), as much as real-time state diagnosis (task and transaction monitoring).

From the different service levels above, it becomes clear that special emphasis is put on the human interfacing to technology, by lifting up the traditional focus from the level of applications to that of services – applications are devised and built for machines, and services are built for people – therefore aiming at serving user's needs, but relieving them from low-level platform concerns. Hence, the use of techniques within the scope of virtual entities contributes to (i) facilitate human interaction, and (ii) reduce the number of messages required to maintain consistent state among many units distributed across the network, in the sense that update messages are sent only to units with entities that can perceive the change.

This way of devising and implementing complex systems is supported directly by the CANopen high-level protocol, as it provides the means to support the communication semantics required by the remote set-up of devices and functions.

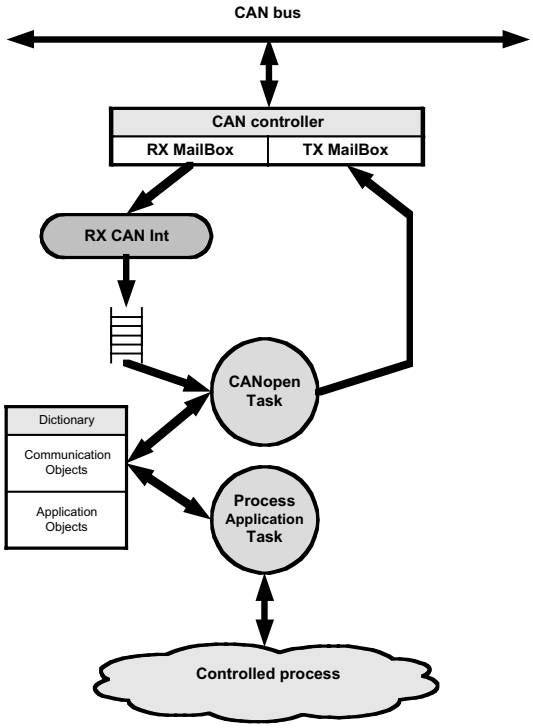

Figure 3. Flow of information within CANopen

The diagram depicting the information flow in CANopen software (figure 3) shows how information interacts with the different functional layers in a CANopen node. It can be noticed that there is no direct link between process application tasks and the communication management task, data being transferred through object variables located in the object dictionary; this one consists of a communication profile, which is common to all devices in the family, and applications related data in one or more device profiles specifying device attributes. In this manner, devices become uniquely and universally represented by an electronic data sheet (EDS) and a device configuration file (DCF), both supplied by makers/vendors.

Therefore, within CANopen, the concept of device profiling was adopted to identify units in a system, thus generally standardising the communication contents (both syntax and semantics) that are to be transferred to each other. Thus, a CANopen profile family for a determined device type provides both descriptions on the actual devices' functionality and the communication mechanism to explore them, i.e., consists of device profiles and a communication profile.

This approach provides present flexibility and ensures room for future developments of device profiles, therefore recommending CANopen for the setting-up and maintenance of complex systems. In addition, CANopen provides standard network management and system services, such as synchronism, emergency messaging and system time provision. These are, synthetically, the reasons why CANopen was chosen as the paradigm of an equivalent high-level transactional service over the wireless network adopted here.

This led us to a layered approach to the respective protocol stack, as depicted in figure 4, where layers other than physical and media access control are used to support message routing, plug-and-play operation (mobility, in general terms), and local functionalities.

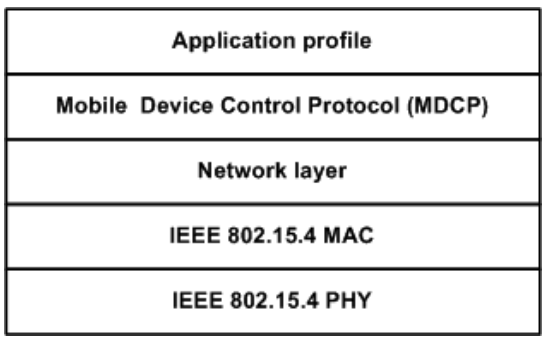

Figure 4. 'Mobile' device control protocol stack

Thus, having adopted a cluster-tree topology for our wireless network, so as to enforce the hierarchy throughout the network, message routing is pre-determined to the point where a faulty master node is found; then, another master node is instructed from the top so as to take its place, with a new list of 'tributary' nodes (figure 5).

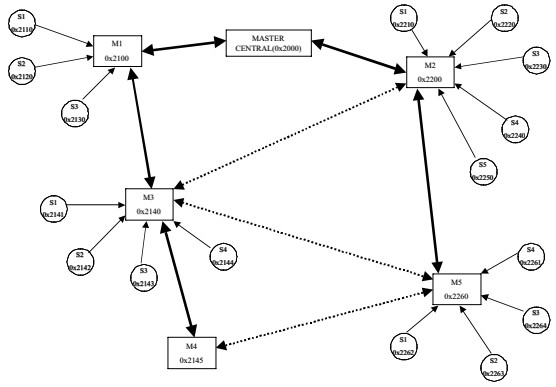

Figure 5. Message (re)-routing
(solid lines = normal paths; dotted lines = alternative paths)

In the same manner, new nodes being 'introduced ' to the network are 'welcomed' by masters that have been previously appointed to the job _ pre-routing regions are dynamically created, which limits the propagation of route requests. Therefore, the protocol handles both new route formation and route update circumstances.

4. Conclusion

This was devised as a hybrid ad hoc network, linking fixed infrastructures with potentially mobile devices, in a typical scheme of sensor network. Problems concerning both data communications and topology control had to be tackled, which led to solutions for service access and network organisation that are innovative and reliable. Pilot installations have been set up, so far in one office and in two apartments, in order to demonstrate the integrated platform for home and building automation that is presented here. Results having satisfactory, industrialisation should occur very shortly.

5. Acknowledgements

This work was carried out in co-operation with Instituto Pedro Nunes (Coimbra, Portugal) and LogicHome, Lda (Porto, Portugal).

Francisco J. A. Cardoso
Universidade de Coimbra – Dep. de Física
Rua Larga da Universidade
3004-516 Coimbra - Portugal
Phone: +351 239 410635
Fax: +351 239 829158
E-mail: fcardoso@ci.uc.pt
Website: www.uc.pt

Paulo M. R. Falcão
Instituto Pedro Nunes (UAII)
Quinta da Nora - Rua Pedro Nunes
3030-199 Coimbra - Portugal
Phone: +351 239 700962
Fax: +351 239 700 912
E-mail: pfalcao@ci.uc.pt
Website: www.ipn.pt

Luís M. A. Oliveira
Instituto Pedro Nunes (UAII)
Quinta da Nora - Rua Pedro Nunes
3030-199 Coimbra - Portugal
Phone: +351 239 700962
Fax: +351 239 700 912
E-mail: loliveir@ci.uc.pt
Website: www.ipn.pt

Nuno J. M. Rocha
Instituto Pedro Nunes (UAII)
Quinta da Nora - Rua Pedro Nunes
3030-199 Coimbra - Portugal
Phone: +351 239 700962
Fax: +351 239 700 912
E-mail: nrocha@ci.uc.pt
Website: www.ipn.pt

Alejandro N. Cruz
Instituto Pedro Nunes (UAII)
Quinta da Nora - Rua Pedro Nunes
3030-199 Coimbra - Portugal
Phone: +351 239 700962
Fax: +351 239 700 912
E-mail: aleja@portugalmail.com
Website: www.ipn.pt

Bruno J. F. Ribeiro
Universidade da Beira Interior
Dep. de Engenharia Electromecânica
6200-358 Covilhã - Portugal
Phone: +351 275 329757
Fax: +351 275 329972
E-mail: bruno@demnet.ubi.pt
Website: www.ubi.pt