Reconfigurable Distributed Network Control System for Industrial Plant Automation

Juan García, Francisco Rogelio Palomo, Antonio Luque, *Member, IEEE*, Carmen Aracil, José M. Quero, *Member, IEEE*, Daniel Carrión, Francisco Gámiz, Plácido Revilla, Juan Pérez-Tinao, Manuel Moreno, Pedro Robles, and Leopoldo G. Franquelo, *Senior Member, IEEE*

***Abstract*—Use of advanced communication technologies, highly integrated control, and programming platforms drastically increases the performance of industrial control systems. That is the case of Motronic, where the synergistic collaboration between industry and academia has led to an advanced distributed network control system. To be commercially successful, it needs to have a low cost and to be robust, even if this requirement implies that it is a custom design and not based on previously existing commercial solutions. Use of standards and off-the-shelf products lower development costs, but usually raise production costs. In this paper, we show that, in certain applications, design of a new system from scratch is more advantageous. This system comprises a set of dynamically reconfigurable local controller nodes, a graphical programming environment, a remote supervision and control system, and a fault-tolerant fiber optical network. TCP/IP connectivity is provided by the use of a local gateway. Motronic is currently being applied in the integrated control of large production plants and in energy and power management industries.**

***Index Terms*—Cost analysis, distributed control, internetworking, real-time systems, supervisory control and data acquisition (SCADA) systems, token networks.**

I. Introduction

THE first industrial plants automation projects, developed in the 1970s, used electrical logic hard-wired into cabinets. These cabinets commanded all electrical equipment located in the plant, and contained all logic needed to perform any operating sequence. Cabinets were modular and composed of local panels, each of which acquired several input signals, computed hard-wired logic equations, and set its outputs. Usually, no communication was established between panels. To realize this logic, operators usually drew electrical schematics and later implemented these schematics using discrete components, such as relays and other electromechanical devices (EMDs).

It is well known that this way of working has the obvious inconvenience of lacking flexibility. For modifying even a single logic equation, it is necessary to rewire several parts of the factory. The need of a reconfigurable control system then arises [1] and considerable effort was exerted in the 1980s to obtain such a system.

One possible way to cope with this problem is the use of *programmable logic controllers* (PLCs) and field buses. These devices are easily reconfigurable and can be adapted to a variety of situations. However, they must be programmed using a high-level description language or a graphical description of the algorithm [2], [3]. In this case, the physical viewpoint of the problem, that would be easily obtained with an electrical schematic, can be lost.

In other cases, the approach has been made to substitute hard-wired logic by microprocessor- or microcontroller-based logic [4]–[6], which are sensibly cheaper than PLCs, but this solution presents the same problem stated above.

Starting from previous work [7]–[9], a completely new system has been developed, having in mind current research trends in plant control. This system, called Motronic, implements a distributed control system with the ability to be connected to external networks. It is capable of presenting the user with a similar interface to the one of hard-wired logic, but with the reconfiguration capability and low cost associated with it being built using microcontrollers. This design philosophy helps operators accustomed to working with electrical schematics to migrate easily to microcontroller-based control. All described approaches of automating a factory are summarized in Fig. 1.

This solution's main advantages are: 1) a completely modular system that allows quick substitution of any damaged component; 2) important wiring and component savings in factory automation; 3) a robust fiber-optic-based network that provides remote communication between local controllers; 4) the possibility of having a supervisory industrial PC to monitor current plant state; and 5) immediate connectivity to external networks, all of this with a programming environment containing a schematics compiler capable of translating drawn schematics into instructions for the microcontrollers. Every part of the system has been designed with Internet capabilities in mind, this being one of the main requirements of today's plant managers, who want to be able to remotely control and monitor their plants without having to add any new software to them.

The single most contributing point to the final user's ability to completely modify the system in order to suit his/her needs is the use of the eXtensible Markup Language (XML) as an electronic document interchange (EDI) between all software components. This creates a flexible system that can be extended in almost any way. Different data representations are possible thanks to

Manuscript received August 29, 2003; revised May 31, 2004. Abstract published on the Internet September 10, 2004.

J. García, F. R. Palomo, A. Luque, C. Aracil, J. M. Quero, F. Gamíz, P. Revilla, and L. G. Franquelo are with the Electronics Engineering Department, University of Seville, E-41092 Seville, Spain (e-mail: quero@esi.us.es).

D. Carrión was with the Electronics Engineering Department, University of Seville, E-41092 Seville, Spain. He is now with Yaco Ingenieria S.L., E-41004 Seville, Spain.

J. Pérez-Tinao, M. Moreno, and P. Robles are with Instalaciones Abengoa, S.A., E-41007 Seville, Spain.

Digital Object Identifier 10.1109/TIE.2004.837871

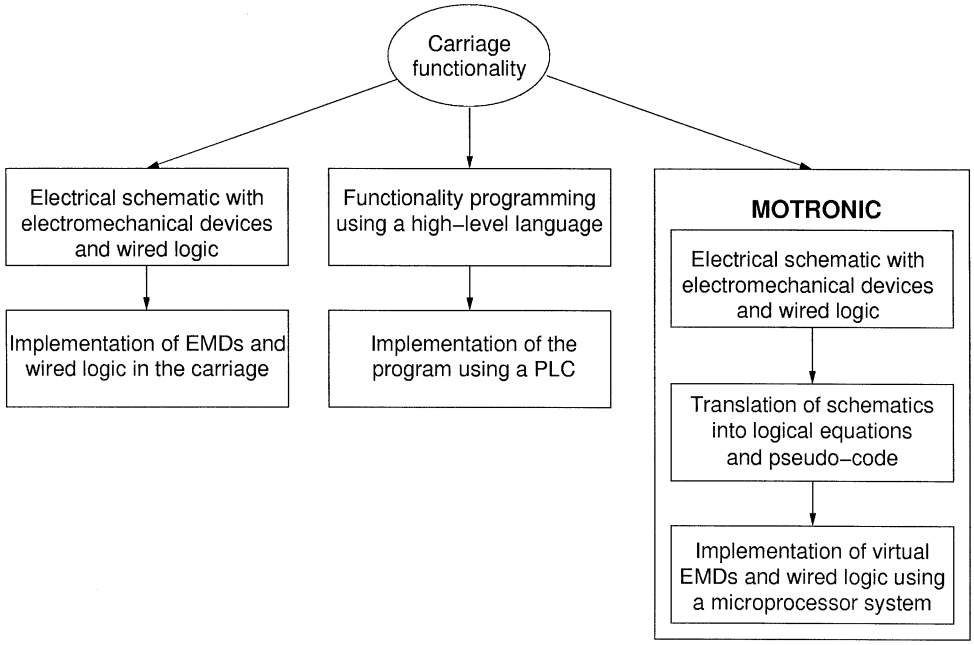


Fig. 1. Different ways to build plant automation: first, hard-wired logic approach, that lacks flexibility; second, solution with PLCs, that requires knowledge of a description language; and third, Motronic, described in this paper, combines the best of both solutions.

the use of XML, and they can be converted from one to any other, giving the system the possibility of interacting with any existing platform. Also, several standards in wide use for remote procedure calling and database query allow the system to be embedded into any other solution (for example, [10] and [11]).

Motronic is a generic control and acquisition system, able to replace many other existing solutions in a variety of industrial plants. It is being presently used to automate industrial facilities like oil plants, paper manufacturing plants, thermal power plants, and car production factories. The designed system is flexible enough to cope with a variety of different situations. The most used application where Motronic has been employed is electric motors control, which is exceptionally well suited to the kind of control Motronic is designed for. It has been proved that the use of Motronic in this kind of environment allows a very significant reduction in the cost of wiring maintenance.

Although it may seem obvious that previous solutions, based on commercial hardware and software products such as PLCs and supervisory control and data acquisition (SCADA) systems, can be used for automation of small to medium-sized factories, experience has proved that these solutions are not practical for the majority of such factories. These solutions have a high academic and theoretical value, but in some industrial sectors they are not practical enough (from an economical viewpoint) to be widely adopted. Almost always, the main concern of an industry management staff is to find a solution that is sufficiently cheap to justify its purchase and at the same time is robust and guarantees that it can handle all plant equipment. Such a solution is presented in the remainder of this paper.

II. System Design

An economical study carried out shows the different fixed and variable costs associated with the use of a commercial solution for control and actuation and the development of a solution from scratch. A typical approach to factory automation involves the use of PLCs networked between them and several central computers for control and monitoring. This solution has low fixed cost, but the variable cost grows rapidly due to the relatively high price of PLCs and licenses associated with SCADA software.

On the other hand, developing a new system has a high fixed cost, but, if it is well designed, every unit installed in the plant is quite cheap. The study carried out by the authors shows that every control unit described in this paper costs less than approximately half of an equivalent PLC. A simple calculation proves that a plant that uses about 50 control units, each one with six to eight local inputs and outputs, can benefit from using this completely developed approach. Later in this paper, the design of these control units will be described in more detail.

Motronic is a distributed control system, composed of four parts, as can be seen in Fig. 2: a programming environment, a hardware platform built by a number of local controller nodes (LCN), a supervision and control subsystem, and communications between all of them.

Architecture has been designed to replace a conventional factory automation system. There is graphical software that allows the user to draw a plant schematic. The programming environment takes electrical schematics as inputs, and compiles them, translating the behavior into logical equations. These equations are internally represented as an XML tree. In a subsequent step, the equations are translated into pseudocode that can be interpreted by the local controllers located in field. The programming environment is described in Section IV.

The hardware platform is composed of local controllers nodes (LCNs), placed in the factory plant. These controllers perform logical calculations, based on equations downloaded at setup time. The equations can depend on digital or analog inputs, and set controller outputs. Inputs to the controllers can be local,

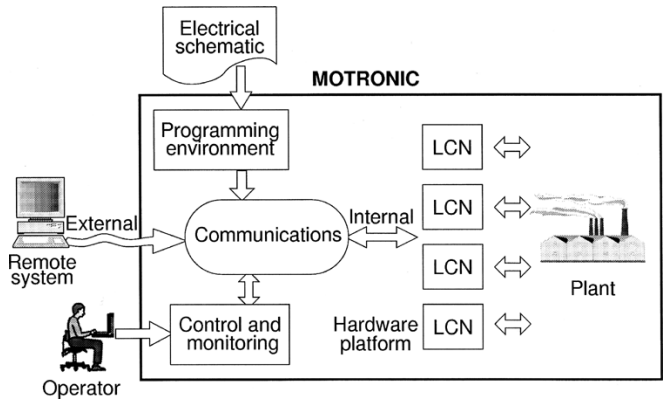


Fig. 2. Motronic system. The programming environment translates schematics into logical equations, the hardware platform implements virtual EMDs, the control system monitors the plant, and the communications provide links between them.

meaning that they are physically attached to the controller, or remote, i.e., physically attached to another controller. When a controller needs to know a remote input value, this value is transferred through the communication system, in a way that is transparent to the user. Internal consistency of the system is guaranteed by the use of a local array of variables, into which all inputs are copied and from which all outputs are obtained. This assures that there is no possibility of an input change in the middle of a calculation. A practical example, with experimental measures, of how the inputs are obtained and outputs are calculated is given in Section VI.

Computed equations in LCNs are intended to replace EMDs and to save wiring in the plant. It can be said that controllers implement virtual electromechanical devices, and the system will act as it would be composed of these, hiding its complexity from the user, who may be accustomed to working in a traditional way.

LCNs are implemented by using general-purpose microcontrollers that interpret pseudocode downloaded to them by the programming environment. Hardware is fully described in Section III.

A supervisory and control system has been developed entirely within the scope of this research project. The SCADA system needs to be able to operate directly on the schematics and to send and receive commands from local and remote controllers in a way that is fully transparent to the user, giving him/her the impression that he/she is commanding the plant directly. The control system, together with the programming environment and the rest of the software development, will be described in Section IV.

To effectively transfer remote variable values between LCNs, an efficient communication system must be employed. The communication system used must comply with the following requirements: it must not depend on specific hardware, but must run on any generic microcontroller; maximum delay time must be upper limited; topology can be either bus or star shaped; individual companies may modify network protocol to adapt it to its particular environment; it must be tolerant to single or multiple nodes failures; and it must be possible to implement it in a fiber-optic medium.

The fiber-optic and on maximum delay time requirements are imposed by the industrial environments in which the system will operate. They are extremely noisy due to the switching of power appliances that may be physically near the controllers and that can carry currents as high as several amperes. Fiber optic has the advantage of being immune to this electromagnetic noise and also of being cheap enough to wire great lengths at low cost, and it has a very broad bandwidth.

Also, sometimes operation can depend on a remote variable transmitted over the network, so the maximum delay time must be limited. These considerations led us [12]to implement a token-passing protocol, based on the standard for this kind of network.

It must be noted that the physical topology of the network is bus or star shaped, but it acts logically as a ring. From now on, in this paper, the network will be referred to as a token bus. The protocol is based on the standards, but some minor modifications have been made to better suit it to industrial environments. TCP/IP capability is also included to link a system to an open network, or several of them together. Communications will be described in Section V.

As will be shown in the remainder of this paper, all of the system, including hardware and software, has been built on top of existing architectures, widely used and heavily proved, such as XML [13], SAX [14] and IEEE Standard 802.4 [15]. Above these layers, the complete system has been designed with application optimizations in mind.

A. Logical Architecture

A typical Motronic control system is composed of a number of virtual modules. These modules usually represent traditional EMDs used in factory automation. All of these are implemented natively in Motronic, but the set of modules is easily expandable, due to the data-oriented nature of the whole system programming (see below).

As every module is internally described in terms of its inputs, outputs, and mathematical operations, new modules can be built by the user, in run time, without need of recompilation, or even restart of the system. These modules can be formed by arranging together two or more existing modules, or by designing completely new ones.

When a user draws an electrical schematic, Motronic SCADA creates an XML netlist document (called XGraph) and passes it to the Motronic compiler, which in turn filters it to an XML equation description document called XLogic. From this document, several applications are obtained. First, the controllers pseudocode is generated to set up the LCNs. Then, a control system is configured in the PC to monitor the system. This control system uses an associative map between field variables and their graphical representations.

At system power-on, the compiled equations are downloaded from the PC to each LCN, and they store them in nonvolatile memory, making the system immune to power faults. From this moment on, the PC is not necessary, and the LCNs can work on their own. Nevertheless, the PC can be very useful for online monitoring and control of the plant.

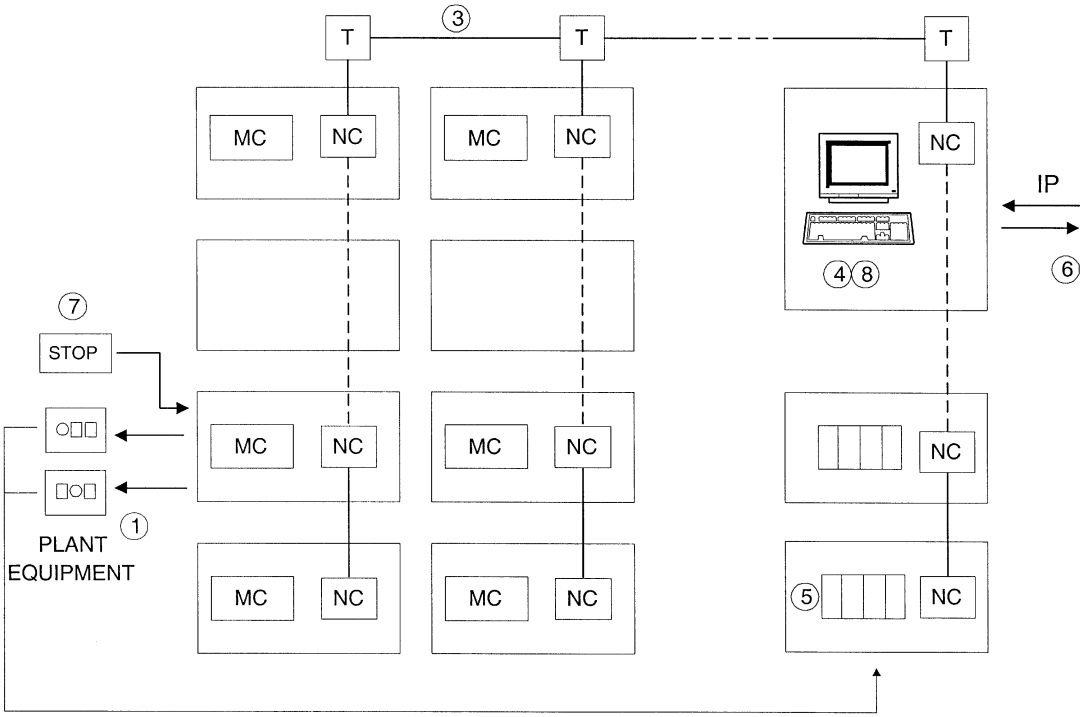


Fig. 3. General system architecture, showing cabinets and LCNs: *(1)* plant equipment; *(2)* LCN; *(3)* fiber-optic network; *(4)* system setup by a PC; *(5)* data acquisition; *(6)* remote access to the plant; *(7)* input signal; *(8)* online supervision and control.

The SCADA subsystem is used to perform a real-time monitoring of plant operation. Every change in a module can be displayed on a user's monitor and, of course, the user can remotely operate the modules.

B. Physical Architecture

The physical system architecture that Motronic is based on can be seen in Fig. 3. This system is composed of a number of LCNs, each one with several digital and analog inputs and outputs, and an intelligent software embedded in a microcontroller that makes all the necessary calculations and takes care of networking communication between LCNs. LCNs are physically arranged to form cabinets, typically used in traditional factories.

In Fig. 3, it can be seen that each LCN acquires digital inputs such as STOP button (7), performs logic calculations, and sets outputs, for example, turning plant equipment on and off (1). In the figure, inside of each LCN, Main (MC) and Node (NC) cards (explained below) are shown.

Sometimes, logical equations can refer to plant operating variables (for example, temperature, voltage, and current in a motor). These analog variables can be measured by the same or another LCN (5). Provision for calculating mean, rms, maximum, and minimum of each analog variable is implemented internally in the controllers.

Logical equations stored inside an LCN can reference local or remote variables without distinction. Communication of data between LCNs takes place over the fiber-optic network that connects each cabinet (3).

In place of an LCN, it is possible to attach a control PC that remotely configures each local controller node before turning the plant on (4), and monitors correct functioning while the plant is working (8). The system works transparently over a network by natively understanding the TCP/IP protocol (6). This way, each component can be operated remotely without the need to be physically near the cabinets. Several possible configurations will be discussed in Section V.

All LCNs are identical, and they are distinguished from one another only by the position they hold in the cabinet. Each LCN knows its position thanks to the Node Card (see below), which sends a position number every time the LCN is powered up. This way, if an LCN must be removed for maintenance, it is safe to replace it in the same location, because the LCN will remember its last location and will continue operating normally. Also, any LCN can be substituted by any other in case of error or damage. If a new LCN is inserted in an empty place, it will ask the PC for its equations before starting its operation.

III. Hardware Platform

A. Design Specifications

Hardware design was realized with three objectives in mind. The most important one is *reliability*. All elements and modules can be safely removed or replaced. System intelligence is distributed along all microcontrollers. An example of this is the continuous monitoring that each LCN makes of any remote input. If a remote LCN fails, local outputs are disabled for security.

Another target is *extensibility*. This application has been designed to be used in many different industrial environments, so the system must be extensible in order to be adaptable to any particular case. Finally, it is fundamental to build a *low-cost* system. We have designed the microcontroller cards in order to remove

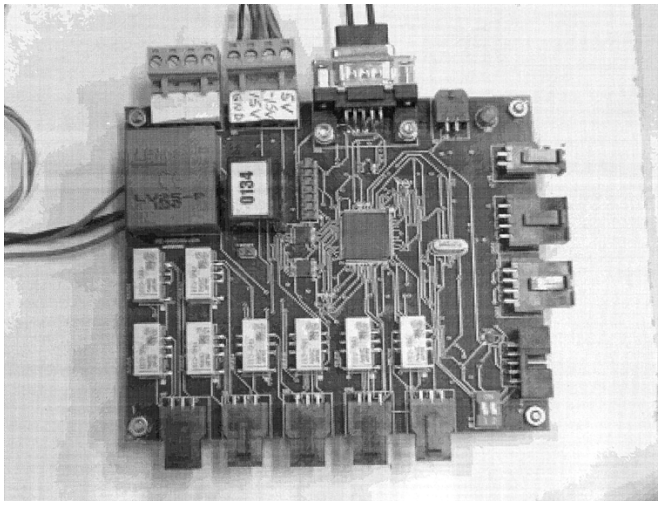


Fig. 4. Main Card. The card has a number of connectors for digital and analog inputs and outputs, and one port to connect it to Node Card through IrDA or fiber optic.

most of the electric wires in each LCN, reducing cost and simplifying the system.

The hardware platform has been divided into communication and controller modules. The completed design is made up of three cards, two of them (Node Card, T Card) supporting communications, and one, the Main Card, controlling the LCN.

As stated in Section II, the system is composed of a number of LCNs. Each LCN contains at least two microcontroller cards (MC and NC) designed within the scope of the research, as well as other standard components, such as power supplies, contactors, electrical protections, etc. The Main Card is movable and the Node Card is fixed in the cabinet.

B. Main Card

The Main Card acquires digital and analog input signals from electromechanical devices and stores them in its internal memory. On the other side, this card commands these electromechanical devices using digital and analog outputs. The relation between card inputs and outputs is established by a set of internal equations, as discussed in previous sections.

Analog input signals are current signals of 0–6 A, 4–20 mA, and voltage signals of 130 V. Output digital signals can be to LEDs and to relays. Digital inputs are optically coupled to protect the microcontroller. The Main Card supports a number of inputs and outputs designed to be sufficient in most applications. In certain cases, a larger number of inputs and outputs can be obtained using up to two more Main Cards in each LCN.

This card contains a fiber-optic connector and an IrDA transceiver to communicate with the system. Both can be used without distinction to set up communications between this card and the rest of the system. The two options present the advantage of being immune to electromagnetic noise, something that is almost always present in industrial environments.

The baud rate of the network can be increased by minor changes to the hardware. Presently, the network operates at 1 Mb/s.

Fig. 4 is a photograph of the Main Card. A widely known microcontroller manufactured by Motorola, Inc. is used.

C. Network Cards

The Node Card is an interface between the fiber-optic network and the Main Card. This Node Card is in charge of adapting the physical format of the message from the Main Card to the fiber-optic link or IrDA link communication and improving the mechanical flexibility of this connection.

This card contains a microcontroller, IrDA, and fiber-optic transceivers to set a communication link with the Main Card, an RS-232 transceiver for connecting the PC that configures the system, and two fiber-optic transceivers to connect with the system fiber-optic bus.

The main way to set a communication between the Main and Node Cards is by using the IrDA transceiver. By doing so, they can be separated, and communication remains active, thanks to the wireless IrDA channel. The distance between these cards can be up to 1 m. This is convenient for maintenance, because if a Main Card has a partial failure, an operator can extract it to find the error without losing the communication between cards. The fiber-optic transceiver allows the connection between both cards in case of no possible IrDA communication, for example, in an extremely dusty environment.

Another important function of the Node Card is to identify the different LCNs. To identify each card, an identification number must be assigned. Each card is connected to certain elements of the plant, so the identification number is fundamental for the right configuration of the system. Identification could be assigned to the Main Card, but it could be placed in a wrong location of the cabinet, so the system would be misconfigured. Therefore, identification is physically associated with the Node Card location. When an LCN is placed, the Main Card asks for the value of the identification number to the Node Card, connects to the fiber-optic communication link, and computes the logical equations to calculate the outputs from the inputs.

On top of every cabinet, a T Card is located, allowing data to flow between cabinets, and assuring that it is possible for each LCN to communicate with all others, even if they are in different cabinets. It allows adding new cabinets by expanding the fiber-optic communication link. This card include fiber-optic connectors and logical gates.

IV. Software Platform

A. General Software Architecture

The software architecture has to fulfill two main requirements: it has to define logic modules to substitute EMDs and to maintain the remote interaction between the user and the plant by means of drawings of the plant wired logic schematics. The virtual logic modules are running on LCNs, remotely programmed from a PC. The entry data for programming can be textual (we have defined a logical equations language named LoGiC) or graphical (using the same drawing schematics that define the visual interface).

Motronic software is composed of three different parts: a SCADA system, a compiler–interpreter pair, and communications stations. The SCADA system, PC resident, allows the user to draw schematics, to manage icon libraries, and presents the graphical user interface with the rest of the system. The compiler is located in the PC and generates, from the schematics,

the pseudocode that is later sent to the interpreter residing in the LCNs. Finally, the communications stations are in charge of managing the message passing between nodes in the network (in the case of the LCNs and PC) and between the network protocols that are supported (only in the PC).

For the planning of the architecture we utilize four design philosophies: object-oriented programming (OOP) for the SCADA, data-oriented programming (DOP) for the SCADA builder and for the virtual module compiler, real-time programming (RTP) for the whole communications procedure (on IP and on token bus, with gatewaying between the protocols), and data-flow-oriented programming (DFOP) for the software running on the LCNs.

Use of the DOP philosophy in industrial software is a recent innovation. The necessity of including a compiler to transform the LoGiC program or the drawn electrical schematics into pseudocode recommended the definition of a virtual document pipeline processing, so the DOP philosophy was the logical choice. The DOP paradigm is also used in several other parts of the system.

The DOP data format is XML. The W3 Consortium has normalized XML as a data description language [13]. The XML documents are sequential ASCII files in human-readable format. This feature, and its inherent hierarchical structure make it ideal for data storage and recovery in a way that is easily modifiable without having to recompile anything. This is one of the most interesting advantages of the system described here, and it allows one to develop the software platform and future additions to it with little effort.

Each XML document has to be processed (a task usually known as parsing) to incorporate the data contents into a client application. For XML processing we use the event-driven approach using an event-driven parser, based on SAX [14], called *expat* [16]. An event-driven parser helps the application to be very fast and economical in memory resources.

XML as a data format is intensively used by the Motronic application as text, as a serialization of other data structures, and as a logical tree. Several dialects of XML are used in Motronic: XIconLib, XGraph, and XLogic, and there is a dialect transformation from XGraph to XLogic in the compiling pipeline. XIconLib is used for external description of objects and system configuration, XGraph describes schematics and synoptics drawings and XLogic describes logical equations in postfix notation. Serialization of these dialects is also carried out by the software to store the data in computer files.

The inverse process, object reconstruction, is made through factory classes [17], [18]. The pattern of the objective class is embedded into its respective factory class. From the entry data, in XML format (for example, XIconLib), the factory class reconstructs the objective class using the pattern as a template. Object reconstruction is useful to increase application functionalities by adding new objects (of known patterns embedded in application factory classes) through their data description.

The programmer can add new patterned objects to the Motronic object library by writing documents in the XIconLib dialect. This is a modular mechanism to increase Motronic functionalities without recompiling the whole application. At the start of the Motronic application, an initialization sequence makes an object construction through the factory classes mechanism.

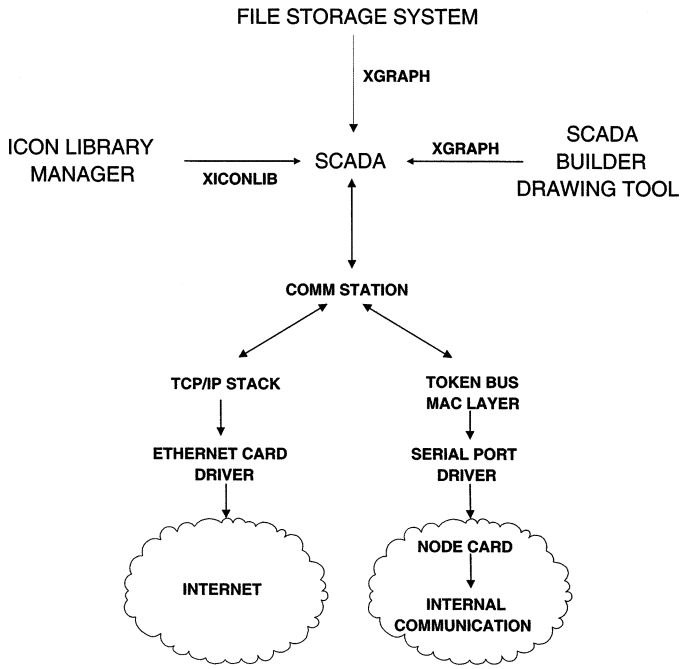


Fig. 5. SCADA action flow. Different ways to generate graphical interfaces are shown. Also, the two main communication channel are presented.

B. Control and Monitoring (SCADA)

The SCADA functionality manages three tasks: to show the plant events interactive drawings, to react to user events, and to transfer messages between the plant and the user. The interactive drawings are electrical schematics for the virtual logic modules (implemented by the LCNs) and plant structure synoptics. This SCADA action flow is shown in Fig. 5

The SCADA functionality, built using the OOP model in C++ language, has been designed following the Model View Controller (MVC) paradigm. The MVC paradigm proposes three principal sets of objects in the software: a Model (underlying logic structure of data), a View port (user interface), and a Controller (connection between View and Model).

Active icons are virtual representations of the wired logical elements, used for displaying the dynamical state of the plant and for providing a user–plant interface. They are designed as tiny automatons that evolve in response to incoming messages and generate outgoing responses.

The Motronic SCADA View port is a typical control GUI, similar to the programming environment (see below), following the same conventions as other CAD or SCADA interfaces. A screenshot of this GUI can be seen in Fig. 9.

A SCADA builder is included in the software, so that the user is not tied to any particular representation of the plant. It is in this builder that the user draws plant schematics. Having in mind the nature of most distributed control systems, the software has been designed in a way so that the user can draw schematic modules that can be reused and, thus, define a modular library of plant schematics that can be incorporated into a hierarchical plant representation.

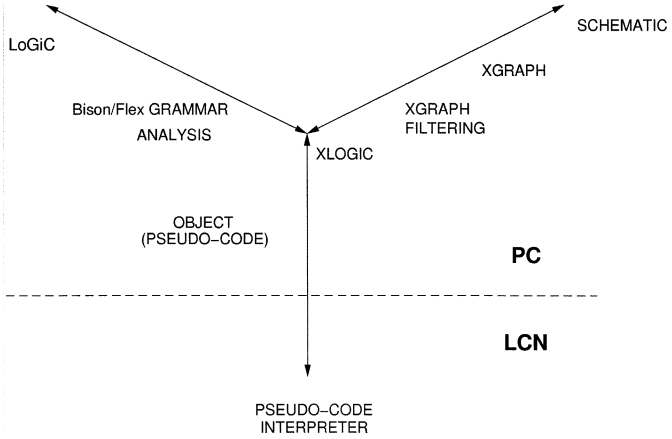


Fig. 6. Compiler pipeline. Inputs to the compiler can be electrical schematics, or textual description of the logic. These inputs are translated into XLogic, and from that into pseudocode downloadable to LCNs.

The set of objects (active icons) that the SCADA system incorporates is easily expandable by the use of the SCADA builder, which allows the user to add new objects to the icon library, using the XIconLib XML dialect.

C. Programming Environment

The programming environment comprises a compiler suite, the mission of which is the translation of electrical schematics (graphical programming) or high-level logic language (textual programming) into pseudocode [19].

The data flow of a compiler is very linear, starting from the source program, generating intermediate documents, and concluding in the object program. The compiler data flow is described as a pipeline for documents building. In Motronic, the compiling pipeline has a "Y" structure, as can be seen in Fig. 6.

The compiler final target, the pseudocode, codifies logical equations and EMDs. The pseudocode instruction set describes operations on a stack because any logical equations set can be written in postfix notation.

Starting from the electrical schematic, a XGraph document is generated. In a second stage the XGraph is parsed, building in memory the logical graphs that represent the wired logic. These graphs are processed with a reduction algorithm, using serial-parallel transformations and other reduction schemes until the coded equations have been extracted. The sets of equations, as postfix trees, are written in XLogic format. These equations are carefully generated by the compiler, in order to minimize the number of variables that must be sent between several LCNs, and so as to save network bandwidth.

D. Microcontrollers Software

The software running on local microcontrollers has to accomplish four different tasks: to implement the virtual electromechanical devices, to communicate properly with the rest of the nodes in the network, to sample analog inputs at a variable (user specified) rate, and to allow its own expansion (as described in Section III-B).

The implementation of virtual modules is made by the calculation of the logical equations downloaded from the PC at the beginning of operation that are mathematically equivalent to electromechanical devices behavior.

Network communication is guaranteed by a software implementation of the link layer of the token-bus algorithm. Communications will be described in Section V. The design of the application protocol has been oriented to achieve a high level of security in the process of sending remote variables through the network, in order to minimize the chance of losing synchronism between LCNs themselves and between LCNs and PC.

The sample of analog inputs and hold of analog outputs is made in parallel with the rest of the program, calculating at the same time the mean, rms, maximum, and minimum values of every input. Some inputs are voltages and some others are current, and the instantaneous power is continuously calculated by the microcontrollers. These values are displayed in the monitoring system running on the PC. This way, the LCNs can act as voltmeters, ammeters, and wattmeters.

Finally, each LCN can be expanded by adding more cards to it. Additional cards provide more memory and, more importantly, a greater number of inputs and outputs to the LCN. Software running on microcontrollers makes addition of these cards transparent to the user and the control PC.

These four tasks are performed by microcontrollers in parallel. A real-time architecture had to be developed to accomplish this, something quite difficult, given the limited resources present in the LCN. None of them has any external memory, in order to reduce costs, something that is fundamental for industry acceptance.

V. Communications

A. Token-Bus Optical Network

To properly configure and monitor the system, data must be exchanged between control PC and LCNs. Moreover, there are cross-referenced equations requiring two or more LCNs working in coordination. The time it takes to deliver such information is upper bounded by using a network protocol derived from the IEEE 802.4 Token Bus Specification [15]. Among the advantages of token bus networks are its excellent throughput performance and its capability to regulate the access to the medium [20]. Other protocol alternatives, such as CAN, Profibus, and Industrial Ethernet do not suffer from the inconveniences of token bus, namely, its complex algorithm, large overhead, and delay in the incorporation of new stations to the ring. These inconveniences can be reduced significantly by simplifying the protocol, as explained below. Having all these considerations in mind, finally, token bus was chosen as the network protocol for the system for its feature of having the transmission delay upper bounded.

Although basic concepts, algorithms, and finite-state machines described in the specification are fully accomplished, some minor functionality has been modified in order to suit system requirements. An example of this is the addition of an alarm frame, intended to allow LCNs to transmit an urgent signal without them waiting to acquire the token, to make sure that emergency situations are quickly handled. Another example of variation from the standard is the simplification of data frame formats to maximize

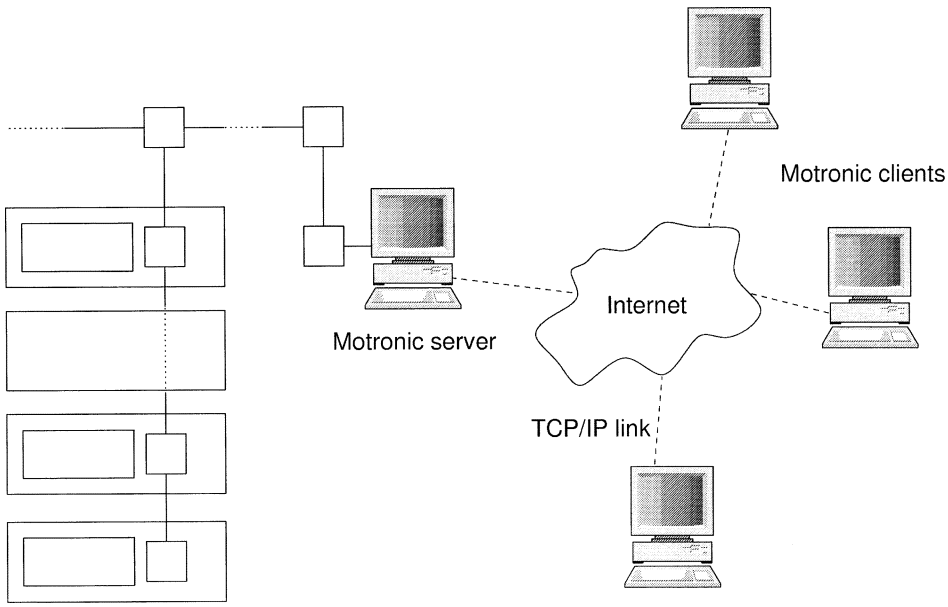


Fig. 7. Client/server setup for remotely controlling the plant over a TCP/IP network.

bandwidth savings in this application. The resulting code has been kept as simple and robust as possible, considering the limited resources of the MCORE microcontroller and the adverse conditions found in industrial environments.

Consequently, all nodes (LCNs or PC) are arranged in a logical ring fashion, where a token runs from node to node, allowing only one node to transmit data at a time, thus ensuring there is a bound on the time it takes to deliver a packet between any two of them. The network protocol is also designed to allow a short reaction time after a critical failure occurs in any of the nodes. This is accomplished by forcing each node to broadcast a call for alarms as soon as the token is received. Alarms are sent through the network with a high priority, something that is vital in some industries in which the time loss involved in communicating with a remote LCN can be critical. The configuration of alarm and priority signals is made at the beginning of operation, together with logical equations downloaded to each LCN. Should an LCN or fiber connection fall down, the system would create separate subnetworks and when the failure is cleared both subsystems will join together and continue working. Only one of these subnetworks can be connected to the local PC, but all of them can work locally.

The token-bus network only covers the link layer of the OSI protocol model. On top of it, an application layer has been designed that allows LCNs to communicate between them and with any external device. This application protocol is based on commands and acknowledgments. Each message is encapsulated in a token-bus packet, which includes a checksum to ensure message integrity. The link layer has the mission of delivering the message to its final destination, or returning an error message to the sender. Having an application layer on top of the link layer, it is possible (and has been realized, as will be shown in Section V-B) to send application messages over other kinds of networks, such as TCP/IP.

B. Internetworking

In addition to the token-passing network linking LCNs between them and with the PC, TCP/IP capability has been added to the system. The PC can act as a gateway between the industrial network and the outside Internet. This way, it is possible to remotely control the whole system from any computer or device connected to the Internet. The piece of software that is acting as a gateway on the PC runs in a separate thread from the control system. This way, communications cannot disturb normal functioning of the control part.

Internetworking also gives the system the capability of being operated using two or more remote PCs. The SCADA software running on PCs can behave as a client or as a server, in network terminology. When the PC physically attached to the fiber-optic bus is acting as a server, it will respond to commands coming from other PCs acting as clients. There can be a number of clients monitoring the plant state all over the world. Furthermore, these clients can send orders and the server PC will translate them into LCN commands, allowing true and effective remote control. Collisions are avoided by server software.

This last configuration, completely generic, is shown in Fig. 7. Among the possible configurations, there is the possibility of having any one of the TCP/IP links shown being wireless. For example, it is possible to have an isolated plant in a hardly accessible place. It is easy to remotely control it using a General Packet Radio Service (GPRS) link between the local Motronic server and the Internet. Moreover, it is also possible to control it from a hand-held device such as a mobile phone, provided it can be programmed to act as a Motronic client.

Another possible connection is presented in Fig. 8. In this figure, two factories belonging to the same company, and both managed by Motronic, are joined together through an external public network (for example, the Internet). The PCs on both sides of the Internet link act as gateways between the token-bus network and TCP/IP, in a way that is totally transparent to control LCNs. Of course, this solution can be extended to any number of plants. This way, it is possible to have a logical equation stored in an LCN to depend on a digital input located in another factory, and communication of the present state of that input is carried by the external network. This approach

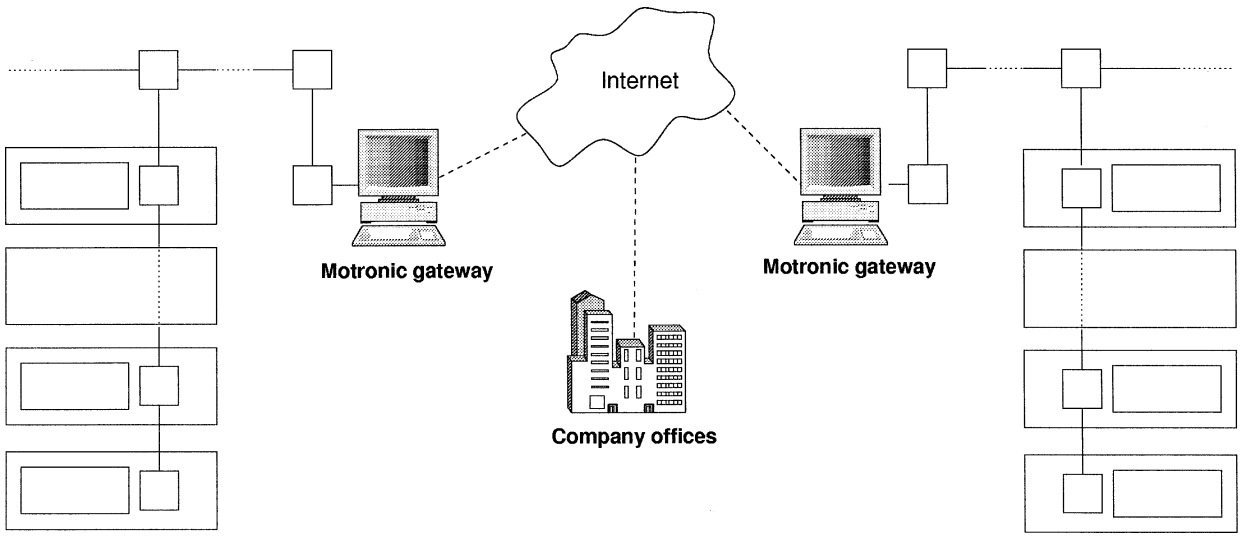


Fig. 8. Interconnection of two plants through the public Internet by using local PCs acting as gateways between local fiber-optic network and TCP/IP network.

can imply a significant savings in wiring for a company, which can effectively join all its plants using a public network. Additionally, talking between operators is also implemented in the application. The possibility of attacking the plants from the Internet is practically ruled out by the use of cryptography between the ends. All communication going through the public network is encrypted using well-known algorithms, creating a virtual private network (VPN) between the different locations [21]. As was mentioned earlier, use of XML throughout all of the applications makes it very easy to use HTTP for remote document and code interchange.

VI. Industrial Applications and Examples of Use

The original application for which the system was initially intended was the motor control cabinets [22] that are widely installed in all kinds of industrial plants. These cabinets are used to command all motors in a factory. Inputs for them are both digital and analog, for example, push buttons and current through motor rotors. Outputs are used to control motor states and to act over the whole system.

In a typical case, a control cabinet is used by an operator to control several similar motors. When the operator presses a "start" button, the motors are turned on. They can be turned off by the operator, or if security thresholds are reached in any of the operating variables that could cause damage. Normally, control logic is more elaborate, and it includes timers, counters, and several other devices that make plant operation semi-automatic.

To highlight some figures, we can mention the oil extraction industry, which makes intensive use of electric motors. In this kind of plant, dozens of motors are connected to each cabinet. For each motor, a number of operating variables need to be monitored (winding temperature, rotor current, state of protections, etc.), giving an overall of about 10 000 signals in the whole plant. To monitor all of these would imply a complicated system of numerous electrical connections. The way that this problem is solved in our system is by using a fiber-optic bus, drastically decreasing the cost and the complexity. In addition, the system is not only monitored but also controlled. Actions over the system are carried out with a graphical environment.

To give an example of application, a motor control cabinet usually implements a star–triangle start of large motors. This particular logic is easily accomplished in Motronic, making its setup and maintenance extremely easy.

The process begins by drawing the electrical schematic in a PC with the help of a complete palette of electromechanical devices. This schematic can be seen in Fig. 9. When the drawing is complete, a compiled form of it is downloaded to the LCN that will implement it. In the schematic of Fig. 9, real and virtual objects are represented. For example, pushbuttons shown at the top right do exist physically, while the switches surrounded by circles do not; they have the only function being activated by the operator on a PC. It is important to note that virtual devices and buttons can (and effectively do) affect the system's logic.

Other elements that appear in this screenshot are: timers, in the bottom-left corner, to commute between star and triangle connection; a flip-flop at the bottom right, to store the on–off state of the system; indicator lights, in the middle, that mimic the behavior of those installed in the field; a voltmeter, at the top right, that is connected to the mains supply; contactors, etc. Of these devices, timers, voltmeters, and flip-flops do not correspond to any real component, but are produced internally by Motronic.

The real system that implements this configuration is shown in Fig. 10. Components that manage power can be clearly seen. Only those devices really exist. The rest of the devices exist only inside the Main Card (which is located at the top-right corner of the image). There is no need to use discrete timers, voltmeters, or flip-flops, thus, considerable amount of space is saved.

With this system installed in the plant, a number of tests were made, measuring response times and verifying that consistency between all variables is always handled correctly. Delay measured between change of an input and activation of the corresponding output is shown in Fig. 11(a). This delay is about 500 μs, and it includes the time of acquiring the input, performing equation calculations, and setting the output that activates the

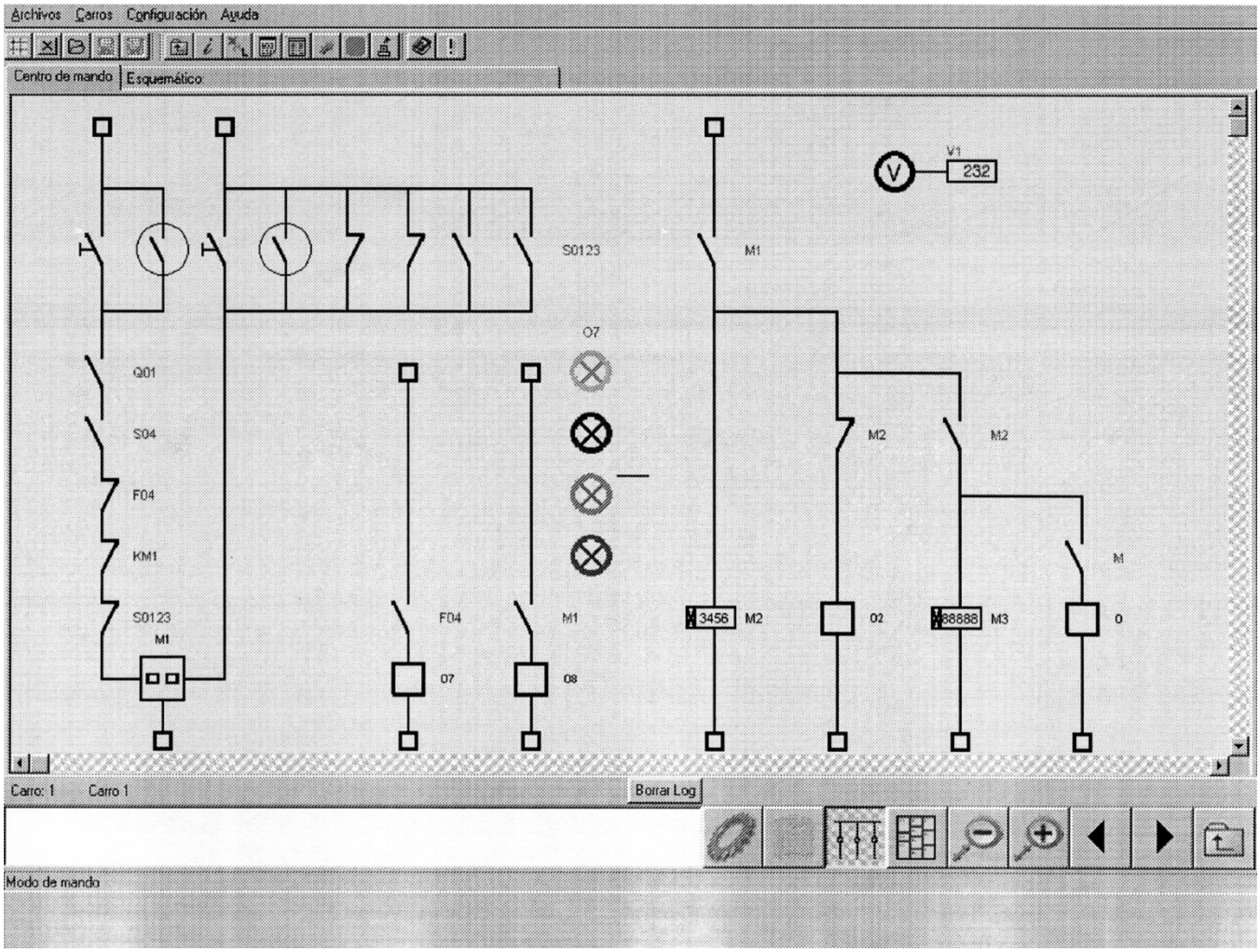


Fig. 9. Electrical schematic of a star-triangle motor starter. Active elements, such as switches and pushbuttons, are shown. Real components, like magnetic or thermal protections, are also shown, as are the virtual components that are implemented by software in LCNs, for example, timers and voltmeters. Logic is also implemented by LCNs.

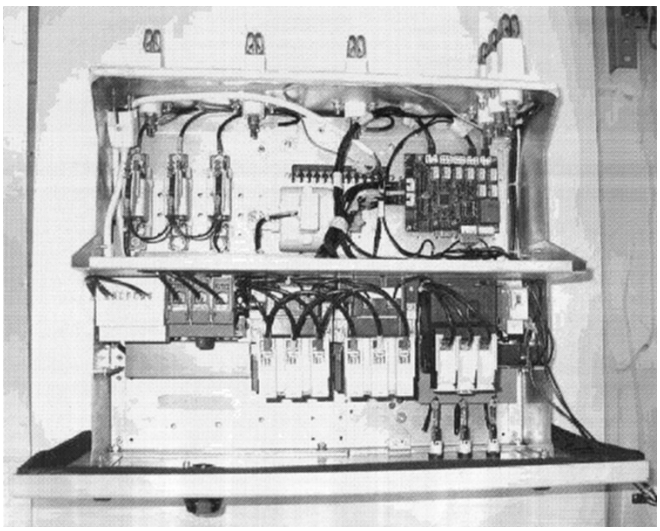


Fig. 10. Top view of the LCN that implements a star-triangle motor start. At top-right corner, the Main Card can be seen. The rest of the devices are relays and protections that are used to turn motors on and off. All of these devices are monitored by the microcontroller and can be actuated from local or remote PC.

motor. The graph displayed in this figure is obtained when the input is local in the LCN. If this input is to be obtained through the network from a remote LCN, then delay is somewhat higher, about 20 ms in the measured case [Fig. 11(b)].

When the electrical schematic is downloaded to the Main Card, the interface in the PC changes from an editable schematic to a control and monitoring system. In this interface, all electrical (analog and digital) variables can be monitored, knowing all system states at every moment. There also are active elements that can be actuated from the PC, for example, to remotely start or stop motors. As can be seen in the schematic, those active elements (marked with circles) are interspersed in the logic circuit, thus, there is no difference for a motor being activated locally in the field, remotely in the plant SCADA, or even from any PC on the globe connected to the plant through the Internet.

VII. Conclusion

A distributed industrial system with network capability has been presented. This system provides a reliable, flexible way to control an industrial plant, with sensible advantages over other solutions, such as PLCs. Motronic is an open and modular platform, completely microcontroller based and made up of a reduced set of hardware and software elements, which can be

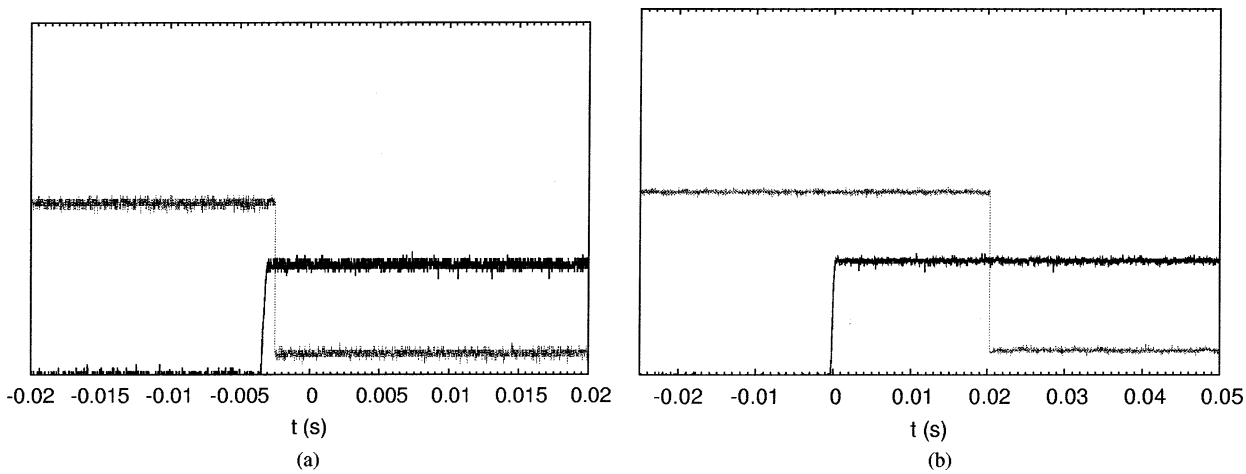


Fig. 11. Experimental delay between input change and output activation. (a) When both signals are local to the LCN. (b) When the activation signal is located in another LCN and transmitted through the network.

easily adapted to meet any given requirements at a reasonable cost.

Motronic is composed of four parts, tightly integrated with one another. Local control nodes provide access to plant equipment, the programming environment gives the user the capability to build plant automation from electrical schematics, the SCADA system allows real-time control and monitoring, and communications provide the necessary integration between the parts, and also the external network connectivity.

In a typical application, such as substituting hard-wired logic, LCNs compute their own equations without the need of any central node for functioning. Furthermore, LCNs can be aware of the current state of other LCNs, and act based upon that information as well. This provides an extremely powerful environment to design the actual operation scheme of the plant, which is much more flexible than those designed using traditional approaches such as hard-wired logic or PLCs.

The internal communication lies upon a token-bus network running over a mixed fiber-optic and IrDA medium which ensures flexibility and electric noise immunity.

A PC can be connected at any point of the network, to remotely monitor and control the plant. The Motronic application running on the PC has been designed with a data orientation, which makes this software easily reconfigurable and completely modular. The data layer has been built on top of a proved, industry-chosen standard, such as XML.

The PC also implements a graphic compiler, designed to translate electrical schematics into pseudocode downloadable to microcontrollers. This approach is extremely powerful, because it allows new devices to be easily connected to the network.

Reliability is guaranteed by a thorough hardware and software design, with a no-single-point-of-failure strategy. Also, methods to react rapidly to certain events, local or remote, are provided, so that operators can safely do their work.

As all protocols are based on standards, the system can be extended in the future to encompass virtually any device.

The system is presently in use in several factories, where it has proved its validity as a truly distributed and network-oriented control system.

The solution presented here is adequate for most factories that want a cheap and robust system and are not concerned about generic and standards-compliant solutions that are not practical enough to be successfully used in real environments. Industrial partners in the development of this system have experience in developing control and actuation systems, and their success in putting them in the market for more than 20 years backs the claims presented in this paper.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for all their valuable suggestions and comments on this paper.

REFERENCES

[1] A. Speck, "Reusable industrial control systems," *IEEE Trans. Ind. Electron.*, vol. 50, pp. 412–418, June 2003.

[2] G. Frey and L. Litz, "Formal methods in PLC programming," in *Proc. IEEE Conf. Systems, Man and Cybernetics (SMC)*, 2000, pp. 2431–2436.

[3] *Programmable Logic Controllers, Part 3: Languages*, International Standard 61131, 1993.

[4] N. Raghunandan, S. Sumithra, R. Anasuya, T. S. Natarajan, and G. Rangarajan, "Automatic control of Stirling cycle liquid nitrogen plant," in *Conf. Rec. IEEE-IAS Annu. Meeting*, vol. 2, 1992, pp. 1721–1723.

[5] J. P. Agrawal, E. Bouktache, O. Farook, and C. R. Sekhar, "Hardware software system design of a generic embedded controller for industrial applications," in *Conf. Rec. IEEE-IAS Annu. Meeting*, vol. 3, 1995, pp. 1887–1892.

[6] A. G. Malamos, K. Kalaitzakis, and N. C. Voulgaris, "A microcontroller-based system for monitoring and controlling the operation of an uninterruptible power supply," in *Proc. IEEE Int. Symp. Industrial Electronics (ISIE'95)*, vol. 2, 1995, pp. 610–615.

[7] J. García, J. M. Quero, R. Palomo, F. Manzanares, L. G. Franquelo, and J. Brey, "Distributed microprocessor controllers using optical fiber network," in *Proc. XV Conf. Design of Circuits and Integrated Systems (DCIS)*, 2000, pp. 557–560.

[8] J. M. Quero, R. Millán, M. Haidenthales, S. Fenoy, L. G. Franquelo, and R. Osuna, "Microprocessor board for industrial control of electric motors," in *Proc. XIII Conf. Design of Circuits and Integrated Systems (DCIS)*, 1998, pp. 352–355.

[9] J. García, A. Luque, C. Aracil, F. R. Palomo, D. Carrión, F. Gámiz, P. Revilla, and J. M. Quero, "Motronic: A configurable electronic controller for industrial power plants," in *Proc. XVIII Conf. Design of Circuits and Integrated Systems (DCIS)*, 2003, pp. 673–676.

[10] M. Gudgin, M. Hardley, J.-J. Moreau, and H. F. Nielsen. (2003, June) "Simple Object Access Protocol (SOAP)," World Wide Web Consortium (W3C)," W3C Recommendation. [Online]. Available: http://www.w3.org/TR/soap

[11] J. Robie, J. Lapp, and D. Schach. (1998, Sept.) "XML Query Language (XQL)," World Wide Web Consortium (W3C)," W3C Recommendation. [Online]. Available: http://www.w3.org/TR/soap
[12] F.-L. Lian, J. R. Moyne, and D. M. Tilbury, "Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet," *IEEE Contr. Syst. Mag.*, vol. 21, pp. 66–83, Feb. 2001.
[13] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. (2000, Oct.) "Extensible Markup Language (XML) 1.0," World Wide Web Consortium (W3C)," W3C Recommendation. [Online]. Available: http://www.w3.org/TR/REC-xml
[14] D. Brownell. (2003) SAX, Simple API for XML. [Online]. Available: http://sax.sourceforge.net
[15] *Information Processing Systems—Local Area Networks—Part 4: Token-Passing Bus Access Method and Physical Layer Specifications*, ISO/IEC Std 8802-4; ANSI/IEEE Std 802.4, 1990.
[16] J. Clark. (2003) Expat: XML Parser Toolkit. [Online]. Available: http://www.jclark.com/xml/expat.html
[17] F. Arciniegas, *C++ XML*. Indianapolis, IN: New Riders, 2001.
[18] A. Alexandrescu, *Modern C++ Design: Generic Programming and Design Patterns Applied*. Reading, MA: Addison-Wesley, 2001.
[19] J.-P. Tremblay, *The Theory and Practice of Compiler Writing*. New York: McGraw-Hill, 1985.
[20] W. Stallings, *Data and Computer Communications*. Upper Saddle River, NJ: Prentice-Hall, 2004.
[21] W. A. Arbaugh, J. R. Davin, D. J. Farber, and J. M. Smith, "Security for virtual private intranets," *IEEE Computer*, vol. 31, pp. 48–55, Sept. 1998.
[22] *Recommended Practice for Electric Power Distribution for Industrial Plants*, IEEE Std 141-1993, 1994.

Juan García was born in Seville, Spain, in 1965. He received the M.Sc. degree in electrical engineering and the Ph.D. degree from the University of Seville, Seville, Spain, in 1992 and 1995, respectively.

Since 1998, he has been with the Electronics Engineering Department, University of Seville. His research is currently focused on microelectromechanical systems, biosensors, and position sensors.

Francisco Rogelio Palomo was born in Seville, Spain, in 1969. He received the M.Sc. degree in fundamental physics in 1992 from the University of Seville, Seville, Spain, where he is currently working toward the Ph.D. degree in the field of stochastic resonance.

In 1999, he joined the Electronics Engineering Department, University of Seville, where he currently is a Research and Teaching Assistant. His research interests include signal processing, distributed control, and embedded systems.

Antonio Luque (M'04) was born in Seville, Spain, in 1976. He received the M.Sc. degree in electrical engineering in 2000 from the University of Seville, Seville, Spain, where he is currently working toward the Ph.D. degree in the field of MEMS.

From 2000 to 2002, he was granted with a research fellowship from the Andalusian Government. In 2002, he joined the Electronics Engineering Department, University of Seville, where he currently is a Research and Teaching Assistant. His research interests include microfluidics, BioMEMS, distributed control, and embedded systems.

Carmen Aracil was born in Seville, Spain, in 1974. She received the M.Sc. degree in physics in 2001 and the M.Sc. degree in electronics engineering in 2003 from the University of Seville, Seville, Spain, where she is currently working toward the Ph.D. degree in the field of BioMEMS.

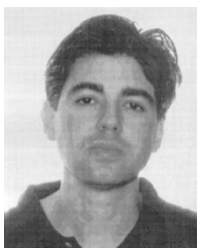
In 2002, she was a Research Assistant at the University of Seville. In 2003, she joined the Electronics Engineering Department, where she currently is a Research and Teaching Assistant. Her research interests include bioengineering, microfluidics, and control systems.

José M. Quero (M'97) was born in Seville, Spain, in 1963. He received the M.Sc. degree in electrical engineering and the Ph.D. degree from the University of Seville, Seville, Spain, in 1988 and 1990, respectively.

He is currently a full-time Professor in the Electronics Engineering Department, University of Seville. His research interests include microelectromechanical systems and industrial applications.

Daniel Carrión was born in Seville, Spain, in 1978. He received the M.Sc. degree in telecommunications engineering from the University of Seville, Seville, Spain, in 2003.

He is currently a Project Engineer with Yaco Ingenieria S.L., Seville, Spain. His research interests include communication networks and protocols, distributed control, and embedded and real-time systems



Francisco Gámiz was born in Seville, Spain, in 1971. He is currently working toward the M.Sc. degree in telecommunications engineering at the University of Seville, Seville, Spain.

A long-time Web designer and freelance programmer, his research interests include data-oriented programming, distributed applications, and electronic music.

Plácido Revilla was born in Cádiz, Spain, in 1978. He is currently working toward the M.Sc. degree in telecommunications engineering at the University of Seville, Seville, Spain.

He spends his spare time writing computer program. His research interests include signal processing, embedded systems, and hardware/software development.

Juan Pérez-Tinao is a Design and Project Engineer with Instalaciones Abengoa, S.A. (Inabensa), Seville, Spain, a leading company in factory automation, plant control, electrical installations, and solar energy.

Manuel Moreno is a Design and Project Engineer with Instalaciones Abengoa, S.A. (Inabensa), Seville, Spain, a leading company in factory automation, plant control, electrical installations, and solar energy.

Pedro Robles is the Logistics and Operations Director of Instalaciones Abengoa, S.A. (Inabensa), Seville, Spain, a leading company in factory automation, plant control, electrical installations, and solar energy.

Leopoldo G. Franquelo (M'85–SM'96) was born in Málaga, Spain. He received the Ing. Ind. and Doctor Ingeniero Industrial (Ph.D.) degrees from the University of Seville, Seville, Spain.

He is currently a full-time Professor in the Electronics Engineering Department, University of Seville. His current research interests are intelligent control of industrial drives and VLSI circuits for industrial control.