TOWARDS A MODELING DISCIPLINE FOR BUILDING AND RESIDENCE AUTOMATION

José Reinaldo Silva
Dept. of Mechatronics, University of São Paulo
Av. Prof. Mello Moraes, 2231, São Paulo, SP, 05508-900, Brazil
reinaldo@usp.br

Marco A. Poli
Dept. of Mechatronics, University of São Paulo
Av. Prof. Mello Moraes, 2231, São Paulo, SP, 05508-900, Brazil
marco.poli@poli.usp.br

Pedro Angel Restrepo
Dept. of Mechatronics, University of São Paulo
Av. Prof. Mello Moraes, 2231, São Paulo, SP, 05508-900, Brazil
pedro_luis_angel@yahoo.com

Abstract. *Building and residence automation, specially the first, is a promising area of research and business since the beginning of the 90´s. However only part of the features envisaged have being implemented (although with success) and even a small partr of them have being treated more formaly. Thus, there are very few publications concerning building and residence automation, and the success obtained so far is based in a transposition from techniques and equipment used in industrial automation. The result is a costly design and implementations where viewpoint requirements concerning the final user are negleted. We claim that building and residence automation are an specific application area and should be faced as that. Implications are that instead of just transfering industrial models and equipments the goals of the area should come first, as well as modeling techniques, based on sound theory. From that framework special equipement should demanded as part of the requirements and necessities. The payoff of the automation process and its antropomorphic nature must be taken as key issues in the modeling phase.*
The paper proposes a discipline for modeling Building and Residence Condominia justifying the use of the terms horizontal and vertical environments and its differences.

Keywords: *building automation, discrete events, supervisory, modeling and design, domotics*

1. Introduction

Building and residence automation has become one of the most promising business and academic application for new architectures and methods in discrete event systems in the last decade. First, because the pressure for comfort and efficiency of construction with controlled price is too high that were transformed in a competitive issue by large companies. Second, because those are applications of discrete control where the characteristic time is large enough to be observed without any artificial help, while the objective of the automation procedures are very clear and thus easier to integrate.

Therefore, we have a very good test bed for new ideas and architectures in discrete event control, supervisory systems, and for general hybrid systems. Curiously, this systems has not being well addressed in the market for reasons such as: i) there is not a project discipline specific for this application neither a general one is commonly used in the market; ii) there is not a study about the proper equipment and the technical needs of the area, generating a poor cost benefit relation in many projects, what reduces building automation more to a marketing issue instead of adding value to the business; iii) in the academic world there is not enough challenge in the field of building automation, where it is granted that the old techniques already existing would be enough to treat new applications without any special development.

In this paper we explore the possibility to propose an advanced architecture that can also be applied to other situations in automation, but that is very suited for building and residential applications. Also, Building and Residential Automation (BRA) can be a very interesting testbed for architectures proposed to adapt different degrees of flexibility and decentralization, depending on the target behavior of the overall system.

The proposed architecture is based in the hollonic concept of integron. An integron is a special kind of subsystem that has all the general properties of its master system, is fully integrated to the master system, although it has not similar behavior. The idea is that the master system (behavior) is the result of the composition of all its integrons.

Integrons were first defined in the field of Biology and Medicine by François Jacob (1970) and fits the idea of similarity between the whole and parts that are its components.

For us, an integron is a special object which has a specific behavior, denoted by its control algorithm and a passive plant that is its control object. Thus, each integron knows exactly what to do and can control its plant without any help or dependency. In other words, only a message would be enough to start procedures in an integron plant.

We claim that this architecture is very suited to complex applications where we include some BRA´s such as very large buildings and residential condominiums. Also, integrons can be represented by objects, which is a very good issue to the modeling technique of discrete systems.

In previous works (Silva 98)(Ramos 98) presented an extended object oriented net system with features such as : i) preservation of the duality between active and passive elements; ii) preservation of property analysis and a linear state

equation. The same system has also some facilities. Hierarchy is implemented by an inheritance from a general class called root which can be specialized in a class box and a class activity as shown in Fig 1. From that point other subclasses represent special cases of boxes and activities that could be added to the library of components depending on the convenience of the designer. In section 2 we will briefly show some definitions of GHENeSys, as a general representation for integrons.

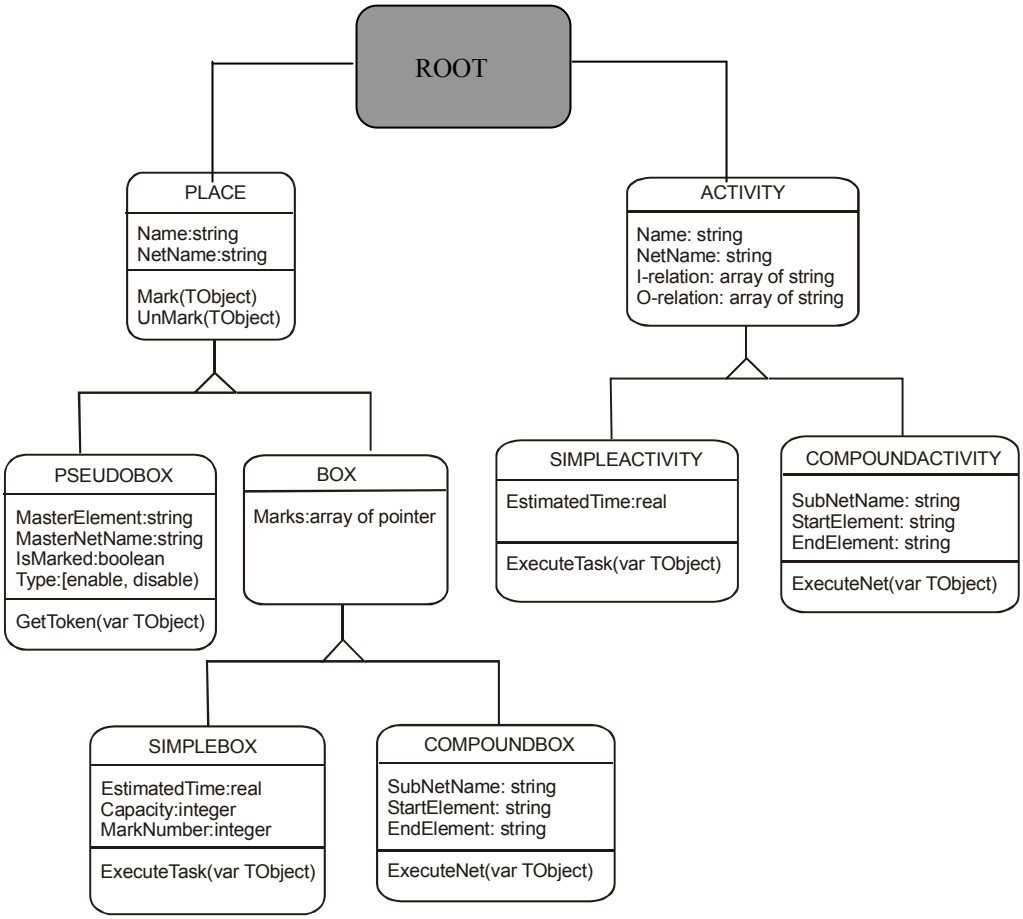

Figure 1. Class diagram for the elements of GHENeSys

Section 3 will present some general ideas about integrons and its formal representation. Next, in section 4 the hardware implementation for an integron, based on dedicated controllers is explained. Finally, in section 5 we will return to the general discussion to analyze the results achieved until the submission of the present work, as well a discussion about further results and other experiences already in course.

2. Building Models Based on Integrons with GHENeSys

GHENeSys is a computational tool that implements the GHENeSys (General Hierarchical Enhanced Net System) proposed by Del Foyo et al. (2000). A GHENeSys net is an extended and Hierarchical Petri Net, formally a GHENeSys net is defined as $N = (L, A, F, K, M)$,

i) $L \cap A = \varnothing$

ii) $L \cup A \neq \varnothing$

iii) $F \subseteq (L \times A) \cup (A \times L)$

iv) $B \cup P = L$

v) $K : B \rightarrow N^{+}$ for all $p \in P, K(p)$ is finite.

vi) $M : L \rightarrow N^{+}$ being that: $M(l) \leq K(l)$ for all $l \in L$.

Where:
L: The Places (composed by the union of two Sets **B** and **P,** Boxes and Pseudoboxes respectively).
A: Represent the activities.
F: Flow relation.
K: Is the Capacity function, indicate the max capacity allowed in a specific Box.

M: Initial marking of the net

The elements of the set L have different functions in the equation of state of the net: the Pseudoboxes P represent places with persistent marking

The flow relation is defined for: $F \subseteq (B \times A) \cup (A \times B) \cup (P \times A)$

Where:
$$F = \begin{cases} 1 & \text{for } (B \times A) \\ 1 & \text{for } (A \times B) \\ 1 & \text{for } (P \times A) \quad \text{(Enabled)} \\ -1 & \text{for } (P \times A) \quad \text{(Disabled)} \end{cases}$$

The elements of the subgroup **P** represent external information to the net in question, but can be an element do **B** Set form another Net. If a net it is divided in nested modules, the use of this element is to model the transmission of information among different parts of a same system, as well as the influence of the extern information, allowing this way that elements in the net are in communication with the outside world.

The GHENeSys Net offers facilities for the construction of models based in integrons, due to following the characteristics:

1. Is a hierarchical and structured net, incorporating the concept of integrons in a natural way.
2. The Object Orientation, allows the use of previously designed models of integrons, reusing the previously made models of mathematical objects with physical meaning.
3. The obtained model using the GHENeSys is structured.

The Class diagram of GHENESYS is presented using UML in the Fig. (1).

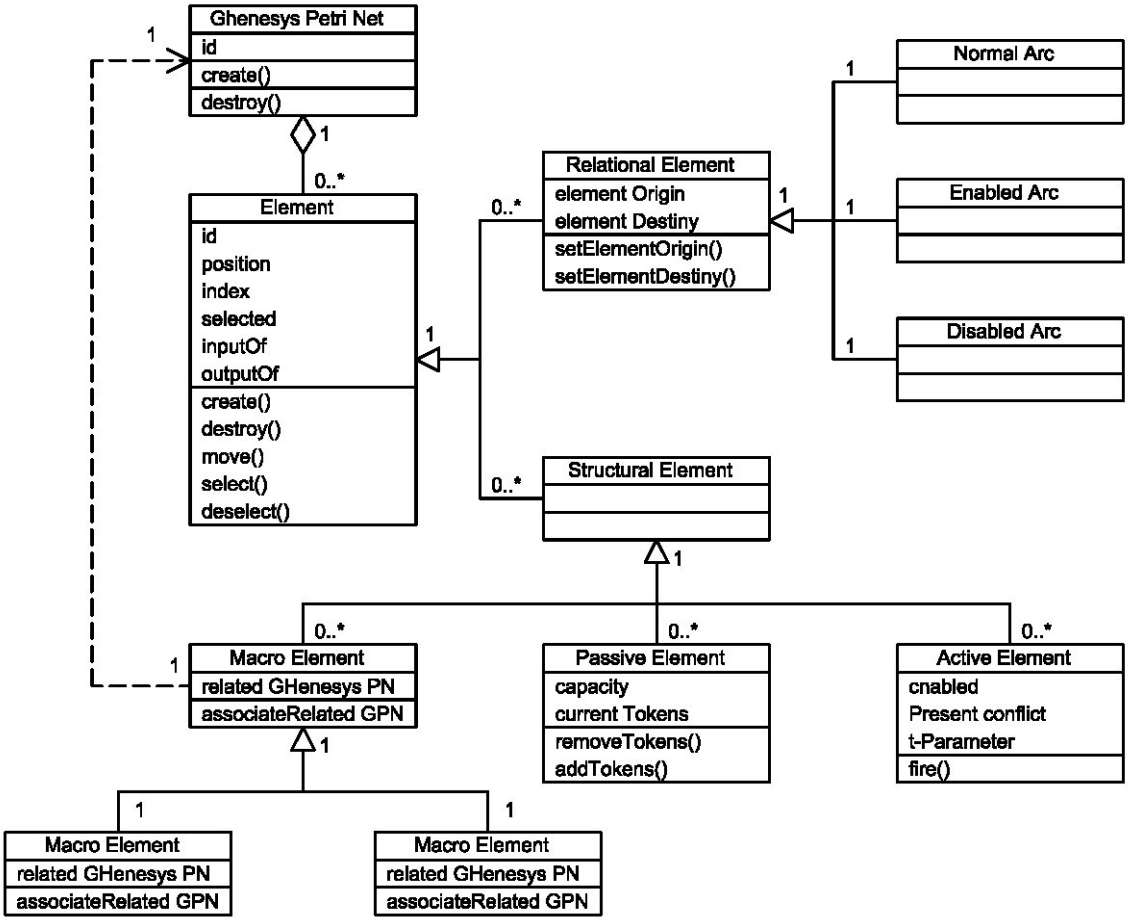

Figure 2. Class's diagram of the computational tool GHENESYS.

Note that the macro elements, represent instances of the "GHENeSys Petri Net" class, allowing that the focus Top-Down can be used, modeling the behavior of the integrons that constitute the system.

The following Fig (2) and Fig. (3) illustrate the process of assigning a previously created Net to a Macro-element.

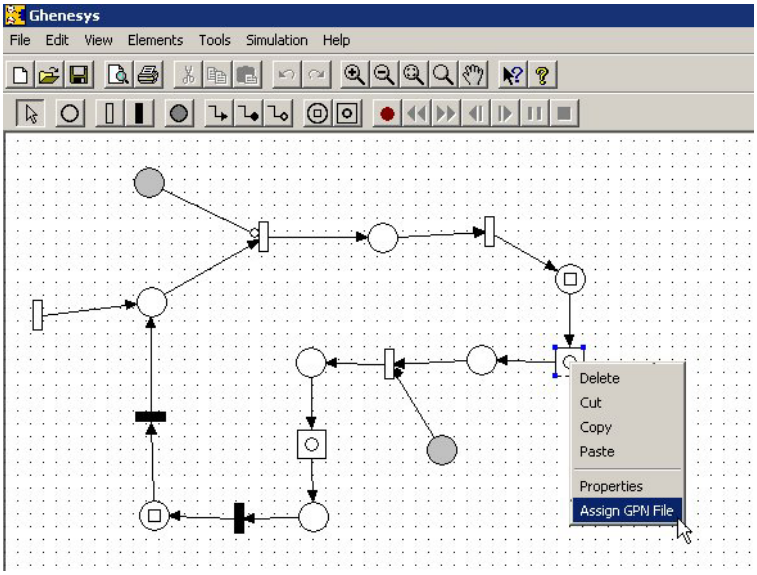

Figure 3. Assign a GHENeSys Net to a Macro-Element, part 1.

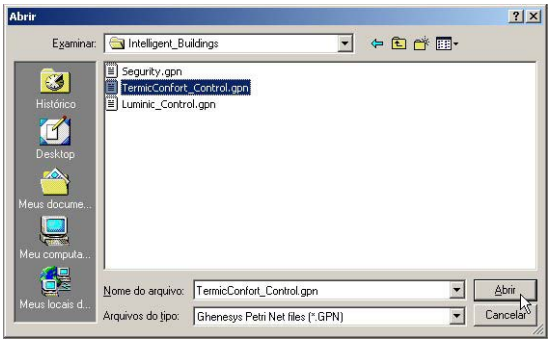

Figure 4. Assign a GHENeSys Net to a Macro-Element, part 2.

The models built in GHENESYS can be executed in independent form of the tool used for the design, using the Dynamic Link Libraries (DLL) form any other computational tool.

3. The Integron Based Architecture

A complex dynamic system is an aggregate of almost independent and open subsystems. The global behavior of such a system results not only from the individual behavior of the subsystems but also from the internal interaction among them and the external interaction with the environment. If the emergent observable behavior indicates some type of organization or a well defined set of functions, the interaction between the subsystems is of cooperative nature and it can be said that they are integrated (Ramos and Silva, 1998, 1997; Silva, 1998).

Thus, *integration* is the process of creation of state/event correlations and control restrictions over subsystems, leading to complex stable configurations. The integration capacity and the internal possible action sequences of each subsystem impose a limit on the resulting configurations and determine the participation or the local functions of the subsystem into the whole system.

The events that contribute to the formation of the structure have a meaning in this context and, thus, carry *information*.

Biochemical networks are complex dynamic systems formed by the coupling of multiple chemical reactions occurring in biological systems far from equilibrium. The mutual affinities between components determine the possible configurations and the interactions with the environment determine the “surviving” ones, leading to cells, organs, tissues and organisms. The Nobel Prize winner biologist François Jacob called these components, organized hierarchically according to nested structures, *integrons*.

The “surviving” configurations are open systems, in the thermodynamic sense, thus exchanging substance and energy with the environment. Despite the external perturbations, they have a property called homeostasis, closed related to stability. Therefore, the behavior of each biological integron follows a cyclic path or sequence of internal states, never entering a state that could lead to its “death”.

These interesting characteristics of the biochemical networks are common to a large class of structures associated to intelligent behavior, like neural and immune networks. Thus, a mathematical model emulating coupled biochemical reactions can be a powerful tool to the analysis and synthesis of artificial intelligent systems in industrial automation and robotics.

To construct such a model, the architecture of an integrated system must reflect the way the subsystems interact, so the proposed one for the considered class of complex systems, is a tree structure, in the sense of the graph theory, hierarchical and decentralized, with the so called primitive components forming the bottom level. These components are locally controlled dynamic systems of a few types, exhibiting a cyclical behavior and oscillating between a finite number of states. They are integrated in groups according to interaction rules associated to their mutual affinities, generating local properties, yielding new components, called *integrons*, which are mathematical objects that emulate the biological integrons.

The integrons interact in a recursive way along the several levels of the tree, yielding the complete structure, with the whole system being represented by a single integron, the root of the tree at the top level. Each integron is formed by a group of interacting integrons of the lower level and can participate in the formation of a single integron at the upper level. The interactions among integrons preserve the openness and the stability properties. Therefore, successful internal interactions are those leading to configurations accepted by the environment.

The interactions among components are modeled by imposing restrictions on the possible event or state sequences through an integration medium. Thus, the components don't interact directly but through dynamic entities called supervisors, which model the integration medium (like the synapses in neural nets or the extra-cellular matrix in tissues). Figure 1 shows the recursive integron-based architecture.

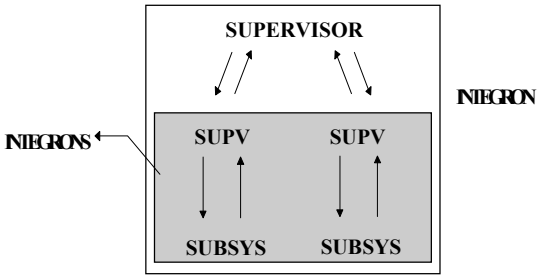

Figure 5. The recursive integron-based architecture, showing the subsystems, supervisors and integrons.

The proposed hierarchical and decentralized structure of an integrated complex system can be formalized through a tree graph (a connected graph without cycles), where the vertices represent integrons and the links represent the formation relation among integrons at successive levels. The primitive components constitute the bottom level and the root integron represents the integrated system. The integrons are constituted from the primitive components according to a bottom-up scheme.

Each integron has local properties that define its participation in the global function of the whole integrated system. These properties emerge from the integration of inferior level components. The connected graph assures that any integron can interact or share resources with other integrons through the upper levels.

Therefore, the integron tree which models an integrated complex system can be represented in a recursive form by the quadruple (I_n, I_{n-1}, P_n, R_n), n = 1..N, respectively the set of n-level integrons, the set of (n-1)-level integrons, the set of local properties associated to the n-level integrons and the set of n-level integration maps. I_0 is the set of primitive components and I_N represents the whole system. Thus, the following results:

$i_{n,j} = r_{n,j}(i_{n-1,k})$, $k \in \{1,\ldots,l_{n-1}\}$,

where l_{n-1} is the number of (n-1)-level integrons or primitive components, $i_{n,j} \in I_n$, $i_{n-1,k} \in I_{n-1}$.

The relations $r_{n,j}$: $2^{I_{n-1}} \rightarrow I_n$ are structural relations, generated by the integration capacities (affinities) of the integrons and the relations $f_{n,j}$: $I_n \rightarrow P_n$ are emerging functional relations.

Each n-level integron is obtained from a relation over a subset of (n-1)-level integrons and it owns an emerging local property, which contributes to the global behavior. Each (n-1)-level integron can participate in the formation of only one n-level integron.

The integration maps $r_{n,j}$ are associated with the action of supervisors, which execute control over the integrons, generating a local behavior or property.

This descentralized architecture is the key word for adjusting the architecture to the proper demandas of the target system. Thus, it is possible to choose, since a master slave architecture (only one integron) until a flat architecture of independent integron (or agents). The possibility to have inteligent integrons is connected with the behavior of the integron: if an integron is inteligent it has a procedure self which can answer its general objectives and that is capable to accept or assing tasks to itself or broadcast messages looking for another integron that could do it better.

To implement such architecture it is important to have more flexibility in the programming than normaly is found in PLC programming. In the next session we will analyze another possibility to produce controlers more agile and cheap to implement the integron architecture.

4. The Hardware Issue

Beginning some 30 years ago, automation equipment suppliers created custom computers, running proprietary software to replace hard-wired control panels. Over the years, these systems have evolved into complex systems capable of controlling a wide variety of processes. The core technology, however, remains proprietary in both hardware and software (Gee, 2003).

A modern *Programmable Logic Controller* (PLC) is a computer-based device designed to control a process. It relates information coming from sensors that monitor the state of a process, with the status of some actuators that are capable of changing it.

PLCs were designed to replace *relay panels*. These are custom made controllers dedicated to a particular application. They can be expensive for complex systems, cannot be easily reconfigured, are difficult to troubleshoot, consume lots of energy, have a relatively moderate speed of operation, and have low reliability (Rullan, 1997).

Commercial PLCs manufactured today are very reliable, fast response, flexible and expensive controllers designed for industry applications. Industry applications differ from building automation in several ways: the response time for industry processes can be in the order of milliseconds or less, when in building automation, there are a little number of processes that require fast response, as most of the actions of the control system are ruled by the timing of people working or living in the building environment; building automation does not require very high reliability equipment in most of its applications; in all control that does not involve threat to human lives, as most of its applications, for example, light level control, door lock control, temperature sensing, *Heat and Air-conditioning* (HVAC) and others, an equipment failure can be promptly sensed (high coverage) and some corrective action can be taken without major damage to people or the company business.

In the 1970s and early 1980s, before the present “global imperative” sparked the race for total quality and ever increasing productivity gains, inefficiencies in many manufacturing companies could be hidden by the impact of inflation which allowed incremental price increases. In today's era of aggressive cost controls, ever increasing price competition, and shrinking profit margins, price increases are not only unacceptable to customers, but also they can cause disastrous business results. Commonly, suppliers are being forced to sign contracts that guarantee stable or decreasing prices, with terms of up to five years. Manufacturers must find the ways to meet these tough requirements, or some competitor, somewhere in the world, will meet them (Gee, 2003).

Building automation of large companies’ buildings, hotels and large office buildings are designed using equipment specially developed for building automation, like air-conditioner tubes flow control valves, infrared passive motion sensors and others, but the design of the control system for this automation is still based in industry PLCs. These PLCs are quite expensive due to factors of time response and reliability already covered in this text. The high prices make automation of smaller buildings, offices, factories, or even houses not economically viable. Thus, a large slice of the potential automation market is neglected due to the high prices of the industry-oriented controlling equipment in use today.

Besides industry-oriented PLCs, other architectures are available for using as control modules and control systems for building automation. One option that can be considered is the use of PCs *(Personal Computers)*. According to Gee (2003):

“Traditionally, either proprietary (PLC and/or (*Distributed Control Systems)* DCS) solutions or custom hardware/software solutions have controlled automation systems. PC-based control technology now offers the reliability and functionality of the traditional equipment with capabilities that allow substantial productivity gains.”

PC architectures like the PC-104 can be used very effectively in building automation: requires low-maintenance, as it can be used as a full solid-state device, capable of handling a huge number of sensors and actuators with relatively low-cost. But still, for cost efficiency, a PC-104-based controller would have to act on sensors and actuators located not only in the surroundings, but possibly quite far from the controlling-station. Analog-data sensors information deteriorate when traveling long distances, so, controllers acquiring data from analog sensors have to be positioned near the sensors, making a more decentralized architecture very desirable.

Sensors and actuators used for building automation can be placed distant one from another. Communication between these sensors and actuators and between the other components of the control chain is an issue to be considered as crucial.

Industry-oriented PLCs usually communicate using Fieldbus. Fieldbus is industry-proven for real-time high reliability shop-floor communication. There are some works available on the integration of shop-floor Fieldbus and administrative Ethernet systems, aimed at providing broader integration between data collected from the processes and administrative ERPs *(Enterprise Resource Plannings)*. However, the application of fieldbus has been limited due to the

high cost of hardware and the difficulty in interfacing with multivendor products. In order to solve these problems, the computer network technology, especially Ethernet, is being adopted by the industrial automation field (Lee et al., 2002). Based on the results of the works of Jasperneite (2001), Lee et al. (2002) and Vitturi (2001), we can affirm that Ethernet shows itself as a alternative to be considered even in higher reliability and faster response control systems, as the ones found in factories, considering the appropriate configuration and load calculation of the network. These works show that switched Ethernet communication shows itself adequate for real-time systems until the theoretic bandwidth limit of the interfaces.

Often, building automation elements include digital and analog sensors. Digital sensors are, for example, open-doors or open-windows sensors, keypads for user interaction, smoke detectors, infrared movement sensors etc. Analog sensors are room temperature sensors, light level sensors, humidity sensors, pressure sensors and others. For this reason, it is important that the automation control system be able to understand the data generated by both, digital and analog sensors.

The actuators in building automation are commonly digital devices. For example door locks, light control, air conditioning, power outlets control and so on can all be driven by digital outputs. But there are some cases of analog actuators like propotional flow valves or light dimming. For this reason, both digital and analog outputs are required in a building automation module.

4.1 The Proposed Architecture

The proposed architecture for the control system consists of a number of small, inexpensive local controllers, micro-PLCs, able to gather data from several digital and analog sensors and to drive actuators as needed, communicating on standard Ethernet (IEEE 802.3) networks using the standard TCP/IP *(transmission control protocol/Internet protocol)* and BACnet/IP building automation protocol.

The proposed micro-PLC consists of one core microcontroller, 96 digital I/O ports *(input-output)*, 16 analog I/O ports, a serial RS-232 interface, and a TCP/IP communication module. The proposed micro-PLC alone should be able to acquire data and to actuate over a small building floor, a house, an office etc.

In Fig. (4), a block diagram of the core module and its components is shown.

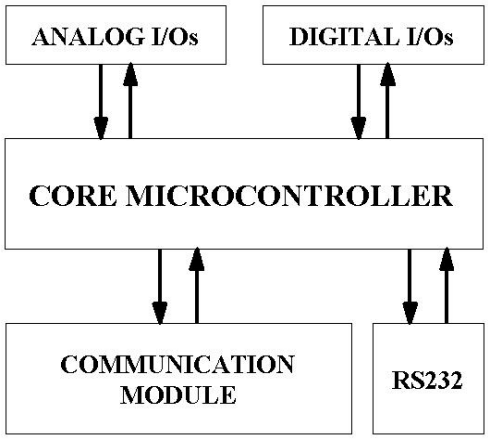

Figure 6. Block diagram of the core module

The TCP/IP communication module is composed of one microcontroller, implementing the TCP/IP protocol, a RS-232 interface, and an Ethernet controller. Communication between the communication module and the core controller is carried by a fast communication medium like I2C, SPI or USB.

Communication between the micro-PLC modules and a supervisory control is made in a three-layer model, shown in Fig. (5).

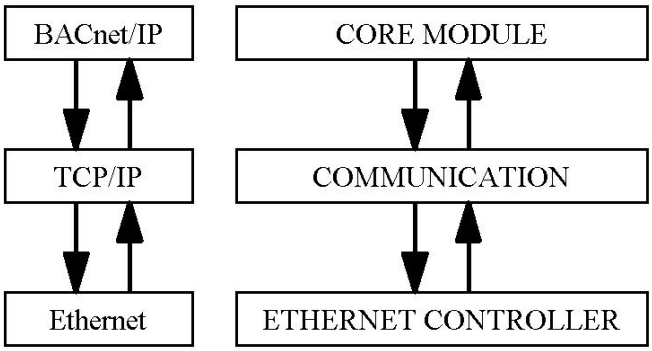

Figure 7. Three layer model for communication.

The core microcontroller understands and generates the BACnet/IP packets. These packets are then sent to the communication module that encapsulates them in TCP/IP and sends them to the Ethernet controller that in turn encapsulates in Ethernet II and sends them to the wire, detecting possible collisions. TCP is in charge of assuring a reliable channel between the sender and the recipient, detecting possible problems in transmission/reception.

Using the configuration proposed by Jesperneite (2001) and making the correct calculations for the amount of data to be send by each communicating end-point, Ethernet shows itself as the chosen solution for the communication needs of the proposed control system, also considering the very low cost of equipment and the high availability of support personal for executing the physical installation and maintenance of the network.

5. Results and Work in Progress

The architecture proposed in the previous session was tested in a laboratory facility involving the following signals and processes: security cameras pass key recognition, door lock, presence sensors and air conditioning set point. A short view from reception room and the main corridor is showed in Figure 8.

Figure 8. View of the security cameras of the Nucleus of Dynamics and Fluids (NDF) lab.

Functions in the controller were programmed in C and can respond to solicitors through the internet. For instance, a researcher can choose the default of this office as “door locked” meaning that the locker will be activated to this room as soon as the door is closed. Otherwise the lock would remain unlocked and the door could be opened by a solicitor.

The flexibility is in the possibility of different users be able to set his or her own options to the dependencies were they are installed. Compatibility among all options will be checked in a central computer that store demands and building policies. Due to space limitations we will not include in the paper some of the programs of inserted in the controller, but the set of functions is very easy to be installed and maintained.

Also, the granularity of the automation can be a feature of the project. That is, since the controller is cheap and easy to program, it is possible to choose as a project feature how many controllers to use to control a set of dependencies and to cover a set of building functions. Thus, if one of the controllers fails, only a small part or the building will be unassisted. Also the association of controllers can fit very well the integron architecture presented here.

5.1 Different implementation for integrons

Another interesting experiment in building automation was performed with a maquette of a building showed in Figure 9, composed of two floors, a main lob served by an elevator.

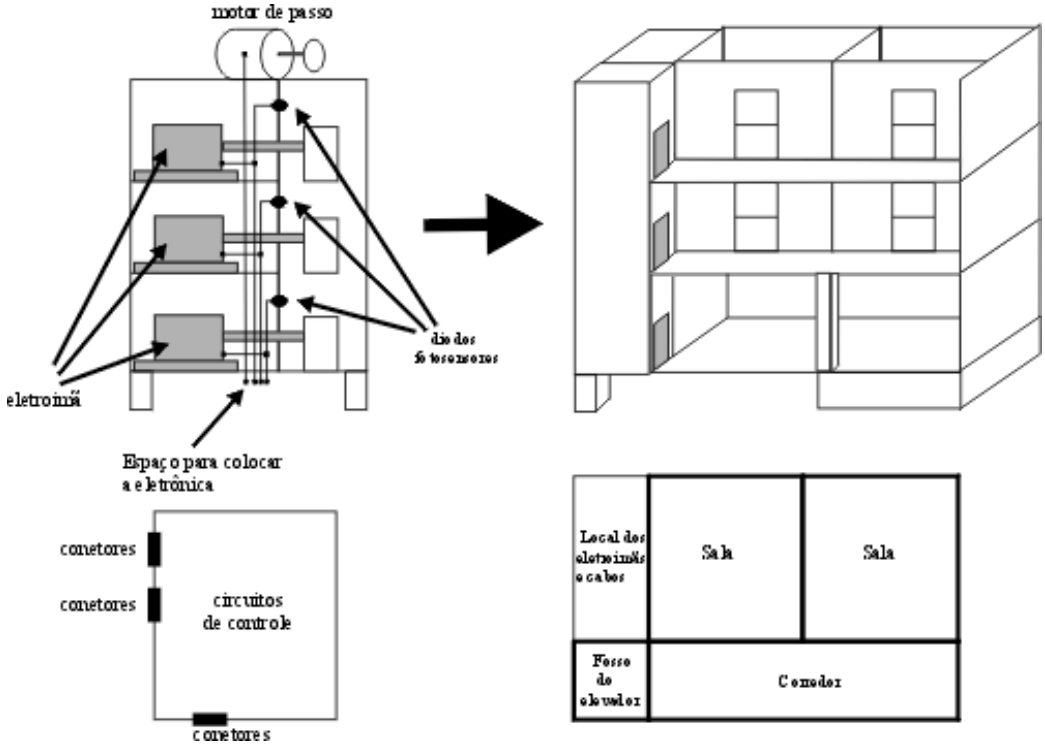

Figure 9. Model for the maquette of a two floor building

A first attempt to implement functions to automate the maquette was done dividing the functions in three modules : i) the security model, composed of presence sensors, smoke sensors and temperature sensors; ii) power saving function, that also use the presence sensors to turn off lights in unoccupied rooms; iii) the HVAC (Heating, Ventilation and Air Conditioning) compose of temperature sensors, fans, etc, to control temperature of the rooms.

It should be noticed that functions used redundant set of sensors what could lead to the repetition of equipment. On the other hand, the integrated architecture allows the sharing of sensor signals controlling unnecessary spend in equipment, and making the overall control policy more rational.

This experiment was implemented in two PLC´s PS4 and a computer with a National Instruments device. The result was what we call a functional implementation of the integron architecture. A projection of this experiment to real cases would also lead to a difficult cabling and the risk of signal lost, mainly for very high buildings. Thus the necessity of associate modularity with some local structure leaded to the new architecture of integrons based on controllers (which are also cheaper and easier to implement).

A new experiment is now in course in the Design Lab where the functions just mentioned are now encapsulated in three controllers that are integrated by a computer where a database is installed. In this new architecture one controller will have all functions (security, power saving and HVAC, now transformed in project goals) structured in one controller responsible for the main lobby, another controller for the two office floors and another controller dedicated to the elevator.

The integrator, the computer can also have a static view of the building by accessing the database or processing the data originated in each controller. Each controller has a local objective which are now keep thermal comfort, save power and keep he security policy of each of the environment it was assigned. Of course special environments would introduce differences in the programming otherwise neglected in the normal programming or implemented with redundancy of equipment or low performance.

The results so far are very promising and we can show that a general policy for the building is now easier to implement and also very intuitive. Real applications such horizontal condominiums could explore this advantages and the flexibility of the automation with controllers by choosing a proper distribution of controller to monitor a large area and introducing a local hierarchy in order to control the overhead of information and what is called the entropy of information until it reaches the central computer.

Other possibilities are large parking lots, shopping centers and all large and public areas (big museums, and so on). The difficulty in modeling and designing the general control policy can be faced with tools like GHENeSys presented in this work, associated with the known engineering design techniques.

6. References

del Foyo, P. M. G., Silva, R., 2003, "Towards a Unified View of Petri Nets and Object Oriented Modeling.", to appear in 17th International Congress in Mechanical Engineering, São Paulo.

Gee, D., 2003, "The Hows and Whys of PC-Based Control", plantautomation.com, http://www.plantautomation.com /content/news/article.asp? docid={f7b804b9-540c-11d4-8c54-009027de0829}.

Jacob, F., 1970, "La Logique du Vivant: Une Histoire de L´Heredité", Editions Gallimard, Paris.

Lu, H. Z., Ying, Z., Liao, T. W., 1992, "Simulation of Programmable Logic-Controller", Computers & Industrial Engineering, Volume 23, Number 1-4, pages 351-354.

Rullan, A., 1997, "Programmable Logic Controllers versus Personal Computer for process control", Computers & Industrial Engineering, vol. 33, number 1-2, pages 421-424.

Jasperneite, J., Neuman, P., 2001. "Switched Ethernet for Factory Communication", 8th International Conference on Emerging Technologies and Factory Automation, IEEE.

Lee, K.C., Lee S., 2002, "Performance evaluation of switched Ethernet for real-time industrial communications", Computer Statndards & Interfaces, Volume 24, Pages 411-423.

Ramos, R.L.C.B. and Silva, J.R. (1998). A Formal Model for Integrated Complex Dynamic Systems. *Proceedings of 5th. IFAC Workshop on Intelligent Manufacturing Systems - IMS'98*, Gramado/Canela - RS, Brazil.

Silva, J.R. (1998). Interactive Design of Integrated Systems. *Proceedings of BASYS'98,* Prague, Czech Republic.

Vitturi, S., 2001, "On the use of Ethernet at low level of factory communication systems", Computer Standards & Interfaces, Volume 23.

5. Copyright Notice