Information Sciences and Computer Engineering, Vol. 1, No. 2, (2010) 42–47

PHARUS

International Journal of
Information Sciences and Computer Engineering

journal homepage: http://www.ijisce.org

New Aspects of Implementing a Distributed System

Marius-Constantin Popescu[a,*], Cristinel Popescu[b]

[a] *Faculty of Mathematics and Informatics, University of Craiova, Romania*
[b] *Constantin Brancusi University of Targu-Jiu, Romania*

Abstract– **Nowadays the implemented distributed systems have architectural varieties which have similar characteristics and development problems. This paper presents the architectural detail of a distributed system and the concepts which enable the development of distributed applications.**

Keyword: Distributed system, "ball on beam" process, Lab Windows implementation.

1. Introduction

In the last decade, the concept of distributed system was assigned as the most powerful computing system, increasingly replacing the old systems [1, 2, 3]. Evidence indicates that over 99% of applications that require a massive volume of mathematical operation and which before were designed without any doubt to high power computers now can be run with far superior results to distributed computing systems [4, 5]. A major reason in the development of distributed systems was their low cost because in most of the cases computers or components may be usual PCs linked together to achieve a distributed computing. Design and implementation of distributed systems requires models, techniques and proper ways communication, adapted to the requirements and restrictions arising from the nature of applications [6, 7, 8, 9].

This work aims at presenting concepts to enable the development of distributed applications and designing an architecture using Lab Windows / CVI product of National Instruments Company [10].

2. Proposed

Description of distributed systems: Managing processes develop a lot of complex algorithms that require a lot of information about the process and also processing these information is very complex and must be performed in real time [11, 12, 13]. This requires a horizontal distribution of performing tasks by several smart devices (microprocessor systems), which corresponds to dividing the process into several independent sub processes. The general objective pursued in the architectural design of the system aims to design a structure able to meet present or future requirements; to be sure that operating system is adaptable, manageable and efficient. A good architectural design will be translate into a simple to implement, test and change system. Based on microprocessor, digital equipment for management of industrial processes began to fill but also to replace analog equipment. An obvious advantage of digital equipment, in addition to flexibility, is that it can easily interconnect with each other through communication lines industry.

Models of allocating work tasks in a distributed system are reflected directly on the performance and efficiency of the resulted system [14, 15]. Locating the components of a distributed system is determined by issues of performance, operating safety, security and costs. Starting from this, to ensure safety requirements in operation, increase system reliability, increase the processing information speed, optimal management of industrial processes have been built distributed and hierarchical management systems as in Figure 1.

By working in parallel information with more process equipment, is obtained a smaller total processing time (it tends to real time restriction) and also obtain a reduction of equipment complexity, leading to an increased reliability for automation equipment as a whole.

Ranking algorithms for data processing on vertically is realized in the way that complex algorithms (time consuming) depart from the direct management of the process, while simple algorithms are closer to the direct management of the process. A possible hierarchical and distributed structure on 5 levels is shown in Figure 2.

Within this structure can be seen a modular assembly, in which communication between devices on the same level and between levels is achieved by specialized bus just to increase the speed of vehicular information. This distributed management ensures a distribution of tasks and also a specialized driving equipment or software, which allows parallel processing of information leading to increase speed of assembly work. Each level out a category of specific tasks and also provides levels specialization such as hardware equipment and application programs.

*Corresponding author:
Email address: popescu_ctin2006@yahoo.com, Ph: +40 745438287

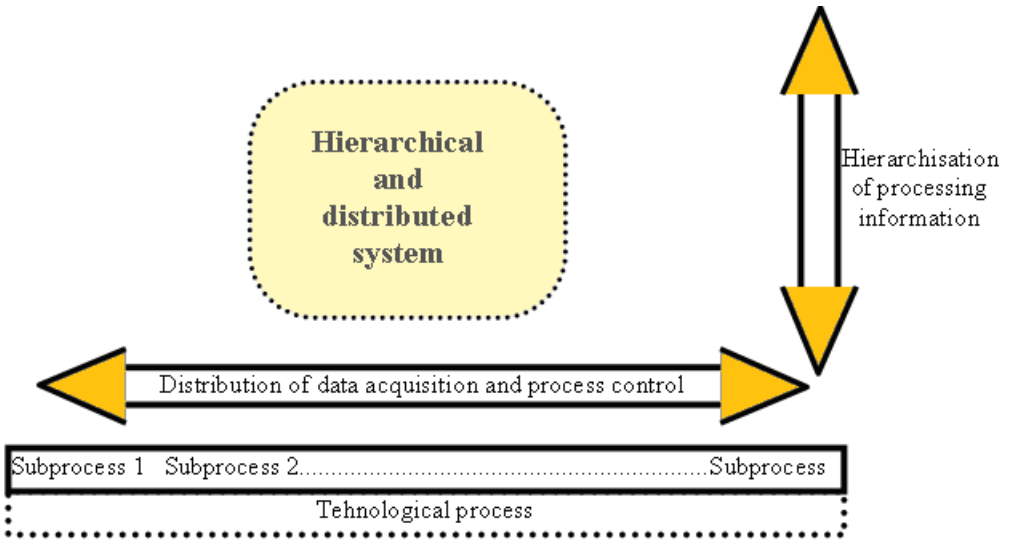

Fig. 1. The principles of a distributed and hierarchical system.

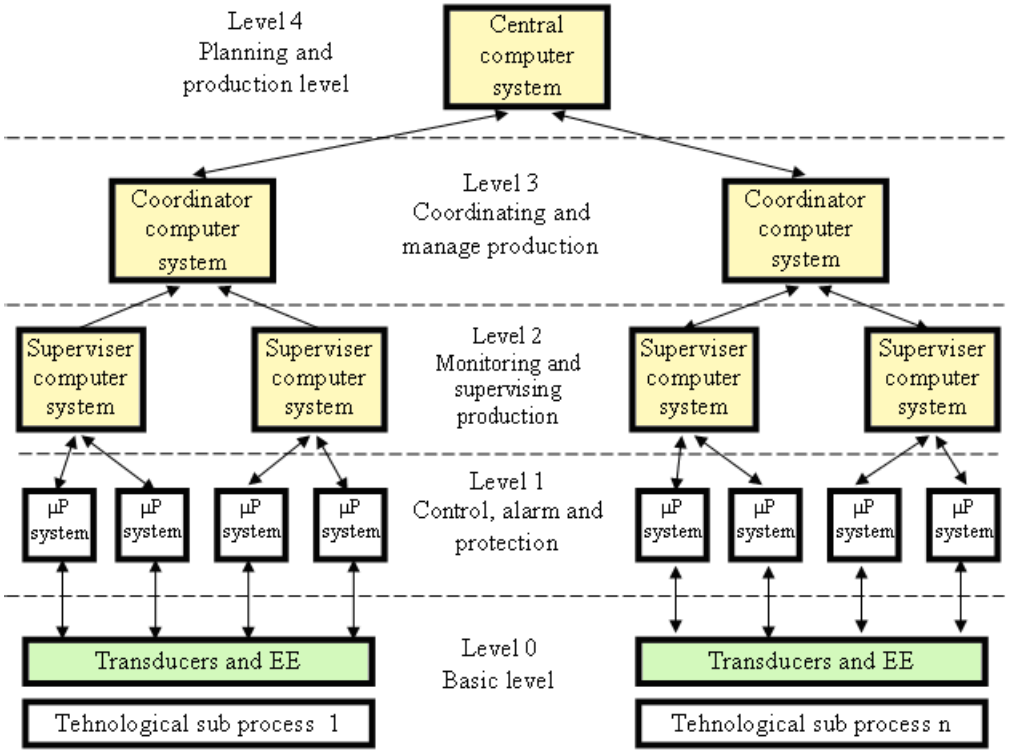

Fig. 2. A hierarchical and distributed structure.

Description of "Ball on Beam" process: In Figure 3 can be seen the "ball on beam" process that was chosen for practical implementation of distributed system. We could choose any process but we decided on this because is the last we studied and we want to improve his control using a distributed system. The objective of this system is math modeling of a free movement of a ball on a bar, to implement a control system of position. The movement is initiated by balancing the bar with a DC motor, located at one end of the bar, which is designed to compensate for the position of the ball [16].

The system is implemented using state reaction. From the description of objects as state variables, it is known that state variables are successive derivatives on higher order of output, variables that are within the process or object to manage. If these variables are measurable then we can achieve a control system with state feed back [17]. The main advantage is that such a structure takes into account changes of all states and not only of one state (the exit). All state vector components are available (measurable). Under these conditions, entry in the comparison element (the reaction way) is considered proportional to the vector of state:

$$u = -k \cdot x \tag{1}$$

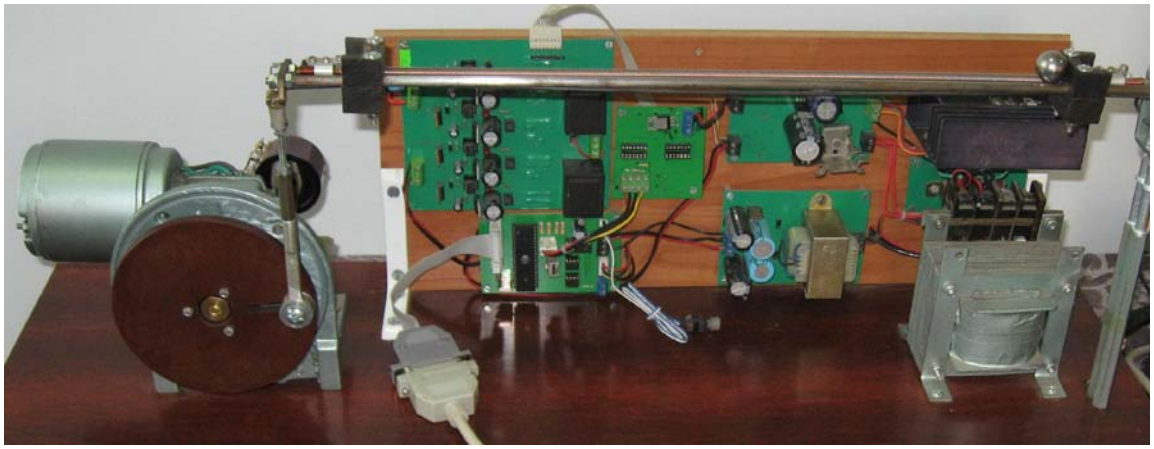

Fig. 3. "Ball on beam" process.

The control structure of "Ball on Beam" process that will be implemented is presented in Figure 4.

Implementing "Ball on Beam" as a distributed system: The Transmission Control Protocol/Internet Protocol (TCP/IP) suite is a set of protocols that govern how data is transferred between networked computers [18]. TCP/IP is the primary protocol of the Internet, the Web, and many private networks and local area networks. Networking components rely on TCP transport for many activities, including the following operations: internet access using HTTP, access to remote file servers and printers, Distributed Component Object Model (DCOM) services. Typically, network communication using TCP involves client-server architecture. A client or server can be any device on the network identified by a logical IP address. Before creating a TCP network connection, the server application must register on the network. Registration establishes the port through which client applications can access the server. After registration, the TCP server application listens on the network for incoming client requests. To issue a request to a server, the client must know the name or network address of the host machine on which the server application is running and the server port number.

When a server receives a request, it processes the request and replies to the client. Figure 1 shows the interaction between a TCP server and TCP client application. Because TCP is connection-based, the server and client must connect with each other before they can exchange data.

Figure 5 shows how to use the Lab Windows/CVI TCP Support functions to create a TCP server and client. The communication process begins when an application registers itself as a valid server. The client then connects to the server using the port number specified in the server registration function. When the client connects to the server, the server application receives the TCP_CONNECT event. Once the connection is established, both the server and the client applications can receive TCP_DATAREADY events, which indicate to the applications to call their respective read functions. When the server disconnects from the client, the client application receives the TCP_DISCONNECT event. Similarly, when the client disconnects from the server, the server application receives the TCP_DISCONNECT event. The client and server applications also can receive the TCP_DISCONNECT event if the connection terminates because of an error. You can use TCP only for one-to-one communication. You can connect a server or client application to several other client or server applications concurrently; however, only one server and one client can exchange data in a single communication session.

In the Lab Windows/CVI TCP Support Library, a *conversation handle* represents an individual communication session. In a client application, the ConnectToTCPServer and ConnectToTCPServerEx functions return conversation handles when the client connects to the server application [19]. In a server application, the Lab Windows/CVI TCP Support Library passes the conversation handle to the TCP callback function when the function receives a TCP_CONNECT event, which occurs when a new client connects to the server.

As an example we use "Ball on beam" process, which control's is realized using state feed back.

The state vector x used in its transposed form is given by relation:

$$x = [\Delta\dot{\alpha} \quad \Delta\alpha \quad \Delta\dot{l} \quad \Delta l]^T, \tag{2}$$

so will have the following equivalents:

$$\begin{cases} x_1 = \Delta\dot{\alpha} \\ x_2 = \Delta\alpha \\ x_3 = \Delta\dot{l} \\ x_4 = \Delta l \end{cases} \tag{3}$$

As it can be observed from figure, the system output is given by:

$$y = \Delta l = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{4}$$

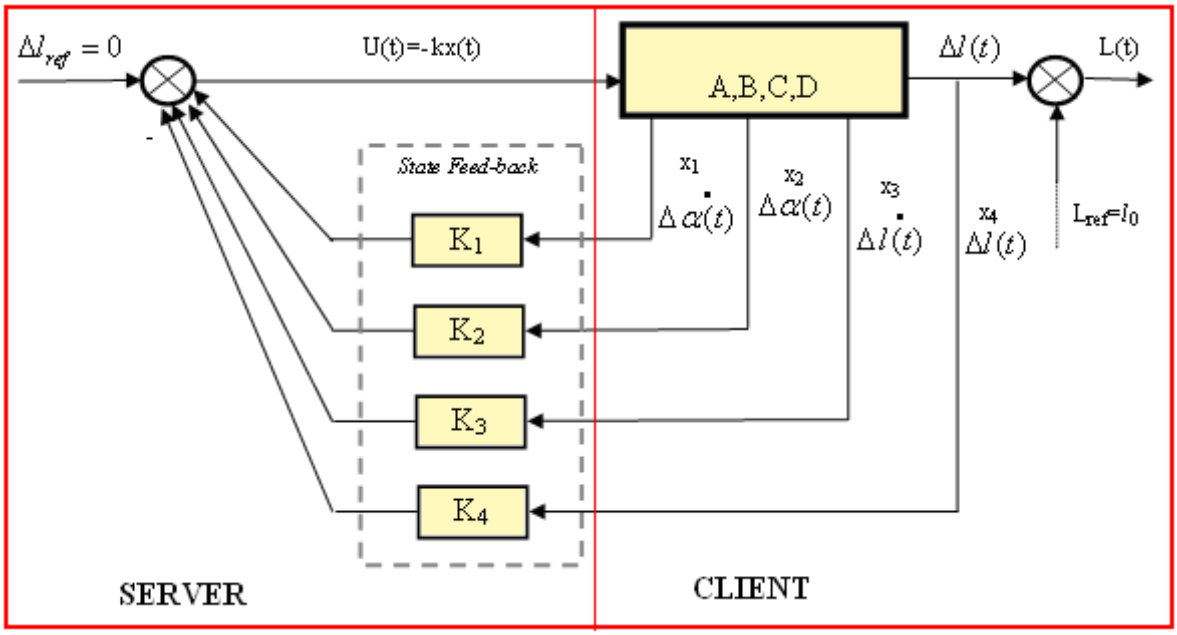

Fig. 4. The control structure of "Ball on Beam" process.

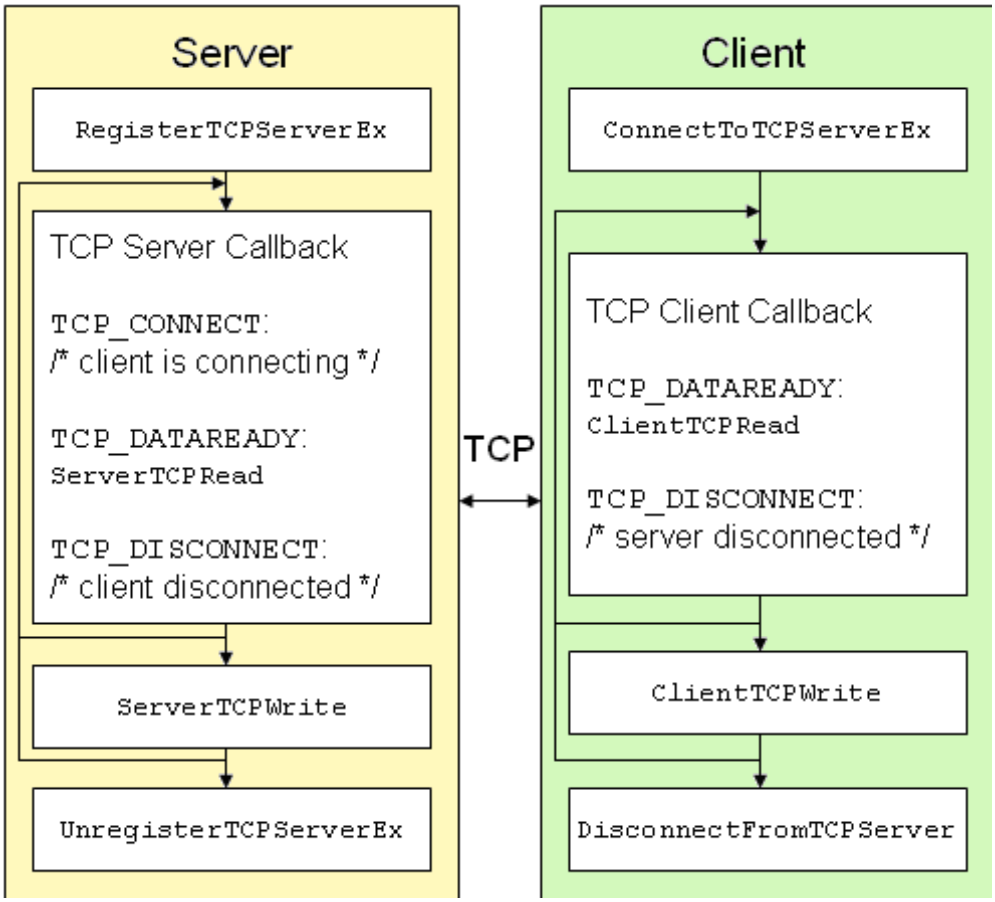

Fig. 5. Lab Windows/CVI TCP Support Functions.

On server will implement the state reaction, the effective control of the selected system, and on client computer will implement the process. Basically our system of regulation will be distributed using Lab Windows / CVI through client and server applications. To realize distribution more steps that are presented below are required.

Convention: To make the application more accessible will use separate files for each function used. These files have the extension .h and are included in the source file whose extension is .c.

Transmitting and receiving information are essential for distribution operations. Thus for the server application will use a func-

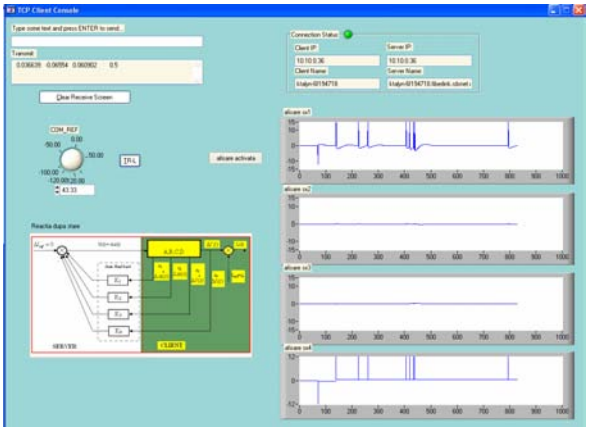

Fig. 6. Client interface.

Fig. 7. Server interface.

tion to receive data from the client called *reception_from_client*. Data can be sent by the client only when its program has called function dealing with this task, named *transmission_to_server*. Note that transfer data between the client and server is based on the TCP interrupt. For selected process data received from the client are the 4 states of state vector which are read from the process: $x_1 = \Delta\dot{\alpha}$ which is the angular velocity of rotation of the bar, $x_2 = \Delta\alpha$ which is the angle between the horizontal and bar, $x_3 = \Delta\dot{l}$which is the speed with who the bal is moving on the fixed pivotal and $x_4 = \Delta l$ the distance from the fixed pivot. A second operation necessary to achieve the desired distribution is the transmission of data from the server to the client performed with function *transmission_to_client* and the reception with *reception_from_server* function. Data to be sent to the client are: reference specified by parameter Δl_{ref} and system error (command). It must be specified that entry into comparison element, as we observe in Figure 8, is considered proportional to the vector of state:

$$u = -K \cdot x, \tag{5}$$

where K is the state reaction matrix (usually is generated using Matlab environment).

3. Simulation

In Figure 6 and Figure 7 are presented the interfaces used for implementing the distributed system using Lab Windows environment and TCP/IP client/server communication.

First of all we connect the client to server so that the connections led are green. Then we start the application from server pressing the *StartAplicatie* button. The button *afisare_locala* is used if we want to see the graphics on the interface. As an advantage if we don't push it, the transfer speed will increase so the transfer of data will be made quickly.

The client receives information from the process (the states x_1, x_2, x_3, x_4) and through his functions sends the information to server. When receives data, server analyze and elaborate through his functions the control law (multiply states with k_1, k_2, k_3, k_4 coefficients, compare the results with reference and elaborate the control law). After this, server sends its result to the process and this situation repeats. So the control of our process is realized using this distributed application. Another important thing is that we can run the application using only one computer but this is indicated only if we simulate the process as we did, not if we have a real one.

On the client interface, under *connection status* led we have information about client and server name and about the IP from the computers containing the server and the client. If we simulate the process on the client interface we have the button *COM_REF* similar with a potentiometer from where we send different commands to server (position where we want to place the ball on bar). On graphics with blue lines we have the command (the 4 states) and with red lines we have the results that server elaborate and

send to client. We can easily observe that results are following reference and that stationary error is null. The simulation ends when pressing *Quit* button from server interface.

4. Conclusion

Distributed systems offer several advantages such as exchanging information, sharing resources, increased security in operation and most important advantage: performance improvements. Distributed design of processes leads to higher reliability for automation equipment as a whole. Algorithms used in distributed systems (as in all systems) must be fair, flexible and efficient. Linearization reaction tends to cancel the non-linearity of a nonlinear system so that the closed loop system dynamics have the linear form. Central to this approach is to algebraically transform nonlinear system dynamics (or partially) in some linear so that it can be applied to linear control techniques.

The control structures with reaction by state variables are extremely important, their result consist in an outstanding performance as precision, response time, speed, but are harder to implement. TCP/IP is the primary protocol for the Internet and many local or private networks. Normally, communications using TCP, involves client-server architecture. Lab Windows/CVI real-time environment is particularly useful in implementing distributed systems due to its client-server application.

References

[1] F. Boian, "Programarea distribuita in internet," *Editura Albastra, Cluj-Napoca*, 1998.

[2] G. Coulouris, J. Dollimore, and T. Kindenberg, *Distributed Systems- concepts and design*. Addison-Wesley, 2002.

[3] D. Grigoras, *Modele de calcul distribuit*. Iaî, Romania: Editura Specturm, 1999.

[4] O. Olaru, M. Popescu, L. Popescu, F. Grofu, and A. Mihailescu, *Sisteme de reglare automata - Teorie si aplicatii*. Editura SITECH - Craiova, 2001.

[5] M. Sloam and J. Kramer, *Distributed Systems and Computer Networks*. Prentice Hall, 1987.

[6] F. Grofu, M. Popescu, and L. Popescu, "Data acquisition system for vibration signal," in *International Journal of Computers, Communications & Control*, (Oradea, Romania), pp. 251–255, CCC Publications, June 2006.

[7] O. Olaru, M. C. Popescu, and V. Balas, "A study of oscillation for signal stabilization of nonlinear system," *Proceedings of the 10th WSEAS Int. Conf. on Automation & Information*, pp. 430–437, March 2009.

[8] J. J. E. Slotine and L. Weiping, *Applied Nonlinear Control*. Prentice Hall, 1991.

[9] A. Tanenbaum, *Reele de calculatoare, Editura Byblos, editia a IV-a*. 2004.

[10] National Instruments CVI, Home Page: http://www.natinst.com/cvi.

[11] O. Olaru, L. Popescu, and M. Popescu, "Sistem modern de reglare automata a turatiei unui motor de curent continuu realizat cu microcontrollerul pic 16f84," in *International Conference on Naval and Marine Education*, (Constanta), pp. 190–196, Romanian Naval Academy, November 2002.

[12] M. C. Popescu, E. V. Balas, M. M. Balas, and O. Olaru, "Algorithm for virtual reality," *Proceedings, 4th International Symposium on Computational Intelligence and Intelligent Informatics*, pp. 125–128, October 2009. IEEE Catalog Number: CFP0936C-CDR, Library of Congress: 2009909581.

[13] M. C. Popescu, A. Petriôr, and A. Drighiciu, "Fuzzy control algorithm implementation using labwindows – robot," *WSEAS Transactions on Systems Journal*, vol. 8, pp. 117–126, January 2009.

[14] M. C. Popescu and N. Mastorakis, "Testing and simulation of a motor vehicle suspension," *International Journal of Systems Applications, Engineering & Development*, vol. 3, no. 2, pp. 74–83, 2009.

[15] M. C. Popescu, "Three connectionist implementations of dynamic programming for optimal control," *Journal of Advanced Research in Fuzzy and Uncertain Systems*, vol. 1, pp. 1–16, March 2009.

[16] F. Grofu and M. P. C. Vilan, "Sistem numeric pentru mentinerea echilibrului unei bile aflata pe o tije orizontala," in *10th International Conference, Universitatea Constantin Brancusi*, (Tg-Jiu, Romania), November 2005.

[17] M. C. Popescu, N. M. I. Borcosi, and L. Popescu, "Asynchronous motors drive systems command with digital signal processor," *International Journal of Systems Applications, Engineering & Development*, vol. 3, no. 2, pp. 64–73, 2009.

[18] D. Comer and D. Stevens, *Internetworking with TCP/IP:Client-Server Programming and Applications*, vol. III. New Jersey: Prentice Hall, 1993.

[19] M. C. Popescu, O. Olaru, and N. Mastorakis, "Processing data for colored noise using a dynamic state estimator," *WSEAS Transactions on Communications*, vol. 8, pp. 321–330, March 2009.

Marius-Constantin Popescu was born in Gorj – Tismana, on the 19-th of may 1965. In 1990, has gratuated the University of Craiova, Romania, Faculty of Automation and Computer and in 1995 he graduated from the Faculty of Mathematics and Informatics, University of Craiova. Author and co-author of 175 scientific papers, 10 textbook and 11 books. In 2003 has received scientific degree Ph.D, he is currently Associate Professor of University of Craiova, Romania. Professional skills: measurements in industrial processes, fuzzy system modeling, artificial intelligence, optimization and automation of industrial processes, web intelligence, intelligent communication network control.

Cristinel Popescu was born in Gorj, on the 28-th of May 1967. In 1997, has graduated the University of Targu Jiu, Romania, Faculty of Power System. Author and co-author of 59 scientific papers, and 6 books. In 2005 has received scientific degree Ph.D., He is currently Associate Professor of Constantin Brancusi University from Targu Jiu, Romania. Professional skills: electrical energy transport and distribution and electrotechnic.