Int. J. Engng Ed. Vol. 19, No. 4, pp. 586–592, 2003
Printed in Great Britain.

OPC-driven Data Exchange between MATLAB and PLC-controlled System*

STOJAN PERSIN, BORIS TOVORNIK and NENAD MUSKINJA
Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia. E-mail: stojan.persin@uni-mb.si

Many manufacturing, and mechatronics systems, are controlled by PLCs and SCADA systems. These systems provide very limited possibilities for research, optimisation, testing and simulation. The main advantage of this proposed solution, which uses OPC, is an integration of the numerical, computational and graphical strengths of MATLAB® with the robustness of industrial systems. The experimental setup, the processing of the data and the results are presented with particular emphasis on student use in a laboratory environment but the techniques employed in the experiments will likely to be seen by the students in their subsequent employment after completion of their college careers.

INTRODUCTION

INDUSTRIAL AUTOMATION has reached the point where one cannot progress further without applying industrially tested, robust and market-accessible automation products such as sensors, actuators, programmable logic controllers (PLC) and supervisory control and data acquisition (SCADA) systems [1] which enable the acquisition of large quantities of data. Process automation is particularly important in isolated areas where the physical presence of humans is undesirable, as for example in the production of electronic components [2] or in the pharmaceutical industry. Such manufacturing systems operate with various degrees of success and are rarely modified [3]. Every alternation within any industrial process is followed by extensive documenting, testing and validating of the new system, sometimes even by certified authorities such as ISO or FDA. Moreover, in the case of modification quite often production must be stopped entirely which gives rise to substantial costs for the entire operation [4].

On the other hand there are a number of research laboratories where various approaches are constantly the subject of change, development and testing in order to bring about improvements. This research work often implements mathematical and simulation software such as Matlab (Matrix Laboratory) which enables the solving of complex mathematical operations, simulations, optimisations, etc. Their main disadvantage, however, is that they are substantially limited in their connectivity to other systems and applications [5].

Automated processes include many procedures where large amounts of data could be processed, examined and optimised: regulation loops, optimal production structuring, calculation of parameters at various points, etc. The equipment presently used in the industry, such as PLC and SCADA, is unsuitable for performing such data processing. Instead, Matlab would provide much better results [6] if only it were possible to feed it with real-time data from the manufacturing process. By applying this technology a wide spectrum of possibilities for testing and analysis emerges which can contribute to the improvement of the whole process. This system can also be upgraded to perform a real-time parallel optimisation procedure. In this configuration the manufacturing process would be running independently whereas Matlab would be performing in parallel with all the necessary mathematical operations, the results of which would be fed back into the process. There are two difficulties in implementing such real-time connections: the problem of connectivity, and the difficulty of presenting a sufficient universal solution that would not require altering our existing manufacturing systems, especially when new research requires updating or changing our existing configurations.

One possible solution would be to use the OPC (OLE for process control) standard [7] and the associated technology. This article presents a solution by showing how to use an OPC client in a Matlab environment. The former is used to exchange data as shown in Fig. 1. The OPC is being increasingly used [1, 8–10] which lends our solution the required level of universality and relevance to engineering education.

The following sections contain an outline of some of the most frequently used techniques in the real-time mathematical analysis of process data, a summary of the main properties of the OPC, and a plan and an implementation of the connection between Matlab and a PLC. In

* Accepted 22 August 2002.

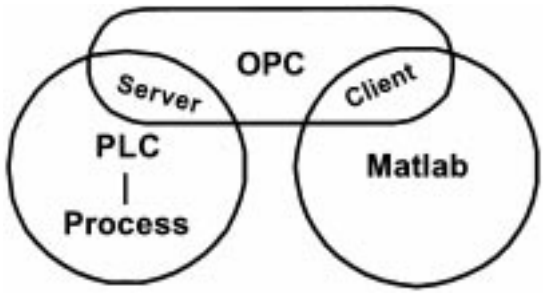

Fig. 1. The connection between Matlab and PLC using OPC.

addition, there is also performance testing of OPC and descriptions of two examples for use in the education process. The final section contains an analysis and a summary of the main properties of our approach.

CURRENT SOLUTIONS FOR REAL-TIME ANALYSIS OF PROCESS DATA

The application of user-defined programmes in one of the higher programming languages available with the majority of SCADA solutions appears, at first sight, to be quite a simple solution for additional mathematical analysis. Difficulties only arise when complicated mathematical operations must be programmed.

One of the more common solutions for feeding real-time data into Matlab is the use of special PC I/O modules [11, 12], supported by Matlab, for example those by Burr-Brown or National Instruments. Apart from product integration, the main advantage of such configuration is the speed of data sampling which can be faster than 1 ms per piece of data. Such systems, however, are purely laboratory systems and are rarely used in industrial automation because every hardware purchase is unavoidably assessed by its integration difficulties in the manufacturing process. Every such interface card must be physically connected to the existing sensory equipment by a parallel or serial linkage to the PLC. This presents the problems already mentioned above, namely the costs that arise during production standstill, documenting and validating the new system. A similar solution is offered by using measurement instruments with the GPIB and VISA communication options which is also supported by Matlab.

In Matlab it is also possible to access serial bus which enables the construction of APIs (application programming interface) which listen to the traffic on the PLC network and, if necessary, return some data. The main advantage of this solution is that it does not directly interfere with the process itself. Its main disadvantage is that the construction of such an interface for an industrial bus is costly in time and the solution is not universal [6].

The development of suitable solutions leans more and more in the direction of software applications which offer data exchange rather than actual data sampling. These measurements are continuously performed anyway for the purposes of PLC and SCADA, which means that they do not need to be further duplicated. The simplest data exchange can be implemented through an ASCII file which is fed by both SCADA and Matlab. To a limited extent, this application is universal, although some modification of the SCADA programme is necessary, and the data transfer is also slow.

By using OLE (object linking and embedding) it is possible to connect Matlab to Microsoft Excel which is available through the Excel Link toolbox in Matlab. Excel fully supports DDE (dynamic data exchange) which is also supported by SCADA systems and hence represents a solution that is one of the most universal. However, DDE never obtains a stronghold in the process control since it is inefficient and unreliable in transferring large quantities of data [13]. Another difficulty in using DDE is that it is necessary to alter the SCADA system used for sending DDE data, assuming that DDE is not originally supported by the SCADA.

The application of OPC (OLE for process control) for data exchange between Matlab and PLC represents a logical continuation in the development of the above techniques and is described in detail in the following section.

EXPERIMENTS FOR OPC PERFORMANCE TESTING

The Matlab mathematical software and the Simulink package were used in creating test environment for the experiment, as shown in Fig. 2. The purpose of the experiment was to accomplish an online transmission of process data from a PLC (Fig. 3) into Matlab. In the majority of industrial environments PLCs and SCADA systems are the only acceptable solution. However, the Matlab package with Simulink is a very powerful and efficient mathematical tool which offers much better data analysis than any SCADA system. Therefore, it appears sensible to include it in the system of process control.

Siemens S7–414 PLC was used in the experiment, although the whole system was also tested with General Electric, OPTO 22, Honeywell, Omron and Allen-Bradley PLCs. The aim was to read and write online process data from the PLC into Matlab. PLC was connected to a computer through MPI/RS 232 (Multi Point Interface) with a maximum transfer rate of 19.2 kbit/s. Even better results could be achieved by using a Profibus which would yield significantly higher rates of data transmission. The data was acquired using the OPC Server. It was then necessary to create a communication channel, to select a device where data could be captured and to define the group of variables to be monitored. The OPC client for

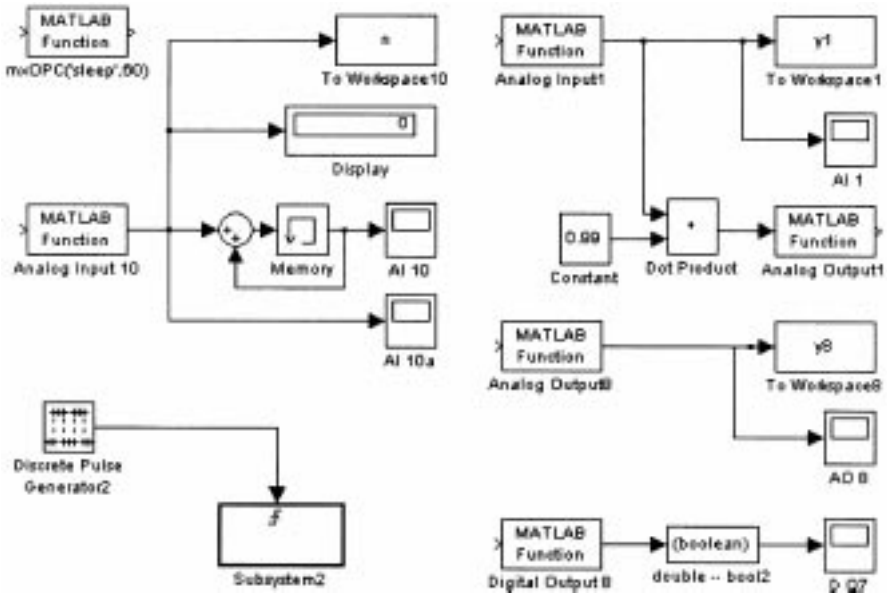

Fig. 2. A part of our testing environment in Matlab.

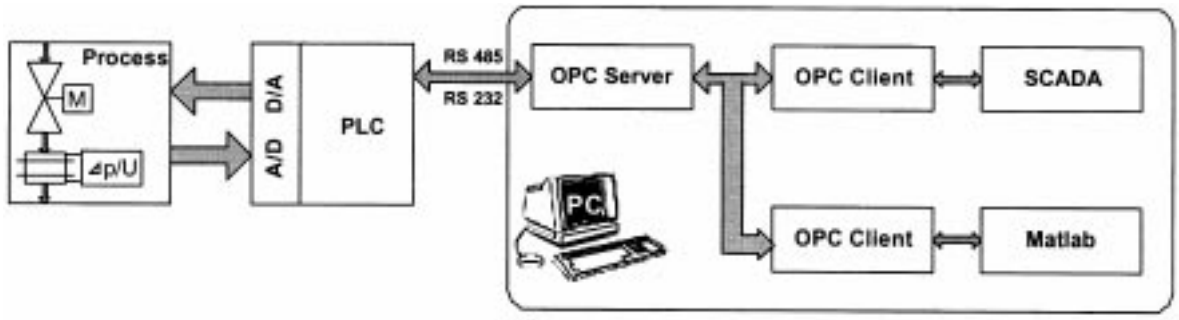

Fig. 3. Testing environment for the experiment.

```
i=0;
while(i==0);
hr=mxOPC('open','KEPware.KEPServerEx.V4','localhost',10);
mode=mxOPC('readmode','cache');
hr=mxOPC('setdoublecache','Channel1.Device1.Group1.analog_vh1',1,0);
hr=mxOPC('setdoublecache','Channel1.Device1.Group1.analog_izh1',1,0);
hr=mxOPC('Startdoublenotify','Channel1.Device1.Group1.analog_vh1');
hr=mxOPC('Startdoublenotify','Channel1.Device1.Group1.analog_izh1');
hr=mxOPC('readcache');
hr=mxOPC('writecache');
i=i+1;
```

Fig. 4. Matlab commands for communication with one input and one output.

Matlab was used which enabled communication between the OPC and Matlab. This is a collection of additional commands which are available in the Matlab environment and enable connection to any OPC server. It is necessary to enter a few commands into the Matlab command window in order to configure this package and Fig. 4 contains an example for a communication with one input and one output.

When testing the data transfer rate from a physical device into Matlab, a continuous sinus signal was used as shown in Fig. 5 where

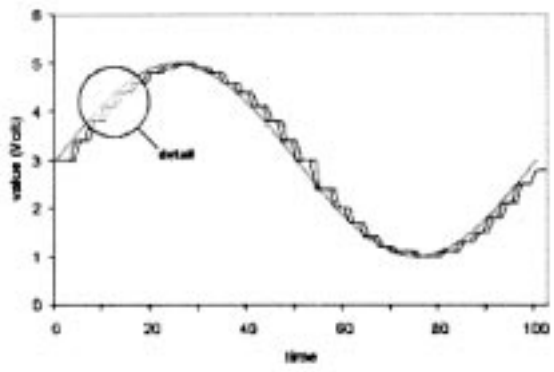

Fig. 5. Signals for testing the data transfer rate.

discreteness as a result of sampling was also noticeable. All measurements and tests were performed under the most unfavourable conditions for the entire system which yielded data on its marginal capabilities. The server captured data from the controller, checked its quality and evaluated it for a change of state. It then forwarded it to the client which used the same route to return the data once it was processed. Synchronous reading was used for the 'worst case' scenario, where each piece of data was changed every time and the same computer ran a server, two clients, as well as SCADA and Matlab software. Figure 6 shows the signals in detail, whereby *a* is the sinus signal, *b* is the server data and *c* is the client data. The total transfer time is represented by t_3 whereas the initial or the fixed part of the transfer time is given by the constant $t_1 = 300$ ms which is the shortest time needed for continuous refreshing of our OPC. The difference between t_1 and the measured time is then due to the server load which was on average $t_2 = 11$ ms for 11 signals, i.e. 1 s for approximately 1000 data. The results obtained are fully compliant with the specifications of the OPC standard as well as with observations made in other experiments performed on industrial SCADA systems [8, 10, 14].

The second part of the test was to establish communication reliability by rating the relationship between successful and unsuccessful transfers

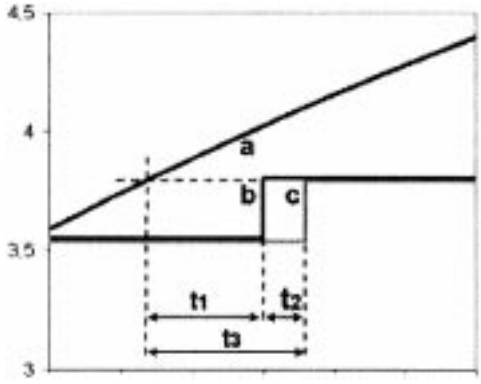

Fig. 6. Signals in detail.

of data. The testing in a closed loop PLC-Matlab-PLC was performed by altering 'sleep' time in Matlab, as well as the load on the computer being changed by executing various processes. It appeared that the highest load for the system was image manipulation. The processing of data in Matlab was executed in fixed time intervals and the relationship between completed and uncompleted cycles was monitored. If a cycle was incomplete and hence a data transfer was unsuccessful, it was assumed the computer was busy with other processes and, as a result, it was overloaded. It is important to emphasise at this point that even industrial systems for process automation do not show 100% reliability in communication, and data transfer failures can occur during their normal operation. This problem is solved by re-requesting the data for which transfer has failed (retry).

The test was performed at 20 ms and 60 ms time intervals between executions of data processing cycles in Matlab, with various loads on the operating system (OS), and by processing eleven pieces of actual process data. The results presented in Table 1 show that with execution intervals of 60 ms and more the communication reliability achieves an acceptability rate which would satisfy industrial environments. The speed of data transfer does not represent any particular problem since in the majority of industrial processes the required data transfer rate from PLC to the computer is in the range of seconds.

The third part of the experiment was done in the same way as the second one, with a closed loop PLC-Matlab-PLC. The total time of data transfer (t_3 in Fig. 6) was measured because the speed of Matlab was known. It was concluded that its average was 720 ms which is still within the acceptable interval for supervisory control of industrial processes.

In the fourth part of our experiment an alternative method was sought for transferring data into Matlab. There exist a possibility of transferring the data through the KEPServerEx interface to Microsoft Excel where Matlab can access it in real time using the Excel link which is, similarly to the OPC Client, a group of additional commands for Matlab. A system as shown in Fig. 7 was constructed and data transfer rates were measured.

It became apparent that data travels from the controller to Matlab faster than in the opposite direction. The reason was in the macro used to write data to the OPC server or the controller. The connection was also very susceptible to the

Table 1. Results from testing the execution of a Matlab application—the percentage of incompleted cycles.

	Cycle execution in Matlab	
	20 ms	60 ms
Unloaded OS	0.12%	0.02%
Loaded OS	1.69%	0.61%

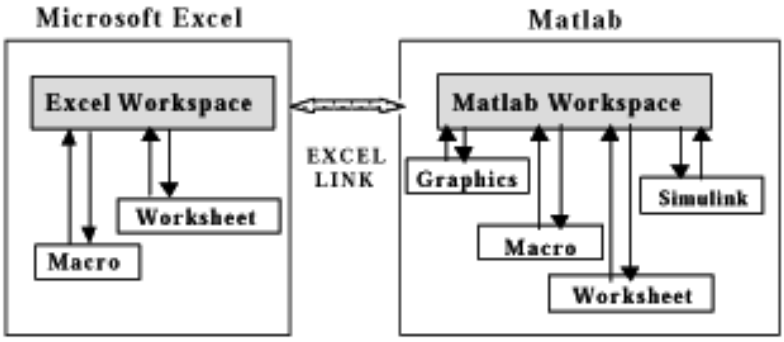

Fig. 7. Interfacing Matlab and Excel.

computer load since data transfer rates ranged from 600 to 1200 ms. Furthermore, the computer load had a higher influence on reading data into Matlab rather than writing it into the PLC.

Finally, the experiment was performed under conditions for which the proposed solution is the most suitable: a transfer of data into Matlab for simulations, optimisations and testing. In all these processes data had already been acquired by a SCADA system (see Fig. 2). Thus, a standard industrial automation system, but now with a link to Matlab, was used. It is important to emphasise that in the described solution there is no time delay between the data in Matlab and that in the SCADA system since both of them are connected to the same OPC server. Furthermore no delays in the operation of the system as a whole were noticed due to the additional OPC Client. This option can, therefore, be used for educational, experimental and research purposes through having all the advantages of reliable and verified industrial solutions.

EXAMPLES OF USING OPC IN A TEACHING PROCESS

The aim of the first example is the implementation of a gain scheduling and adaptive control system for use in undergraduate engineering education. The system was developed to demonstrate the benefits of using a connection between Matlab and PLC and this is what senior students of mechatronics and electrical engineering are taught in this course.

The process, which is connected to the PLC, is composed of a flow meter and an electrically-driven control valve. The overall system is shown in Fig. 3 and was described in a previous section. The aim is to control the mass-flow and a PID algorithm was chosen because it represents simple classical control which can also be 'upgraded'. In addition, at this point in the educational process, the students have had significant exposure to this type of controller and should be relatively comfortable in applying it to the mass-flow control problem. Several tests were performed during the course of this study. However, for brevity, only salient results are reported. Students began a study of the process in detail and observe the nonlinear steady-state characteristic (Fig. 8) which was obtained using Matlab and OPC connection. Then a PID controller was designed in Matlab and the controller parameters were downloaded directly into PLC. As shown in Fig. 9 the control was not equally appropriate over all operating points which indicates a need for gain scheduling control. Gain scheduling is then a two-step procedure where firstly, local linear controllers are designed based on linearisation of the nonlinear system at several different operating points and then a global nonlinear controller for the nonlinear system is obtained by interpolating or scheduling among the local operations point design. The combination of PLC, OPC and Matlab was used again, and Fig. 10 represents a control using gain scheduling PID. For advanced students there is

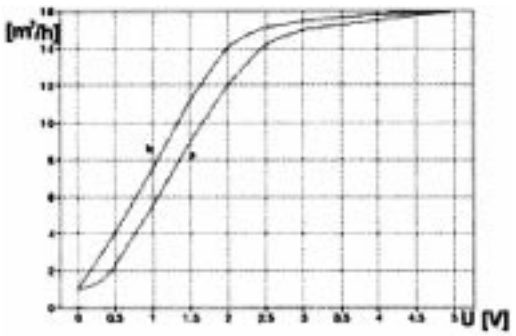

Fig. 8. A nonlinear steady-state characteristic of the process.

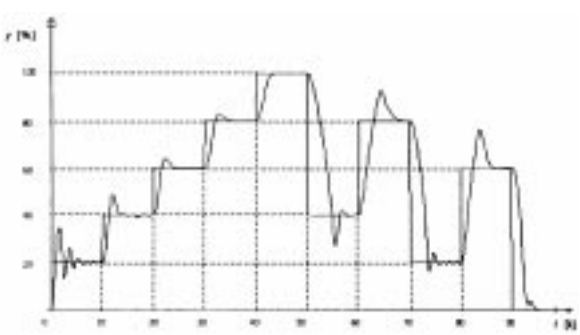

Fig. 9. Classical PID control.

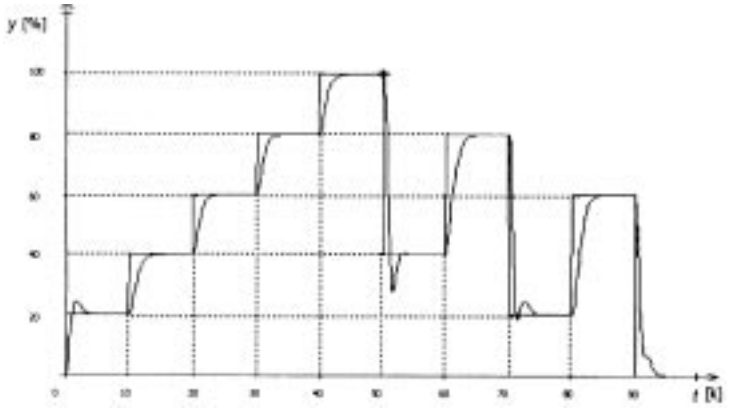

Fig. 10. PID control with gain-scheduling.

also a possibility to design and test an adaptive PID controller using on-line identification procedure in Matlab. We have to remember that this process is controlled by PLC and realisation of identification in PLC-code would be very difficult.

Our second example of using OPC is also suitable for mechatronic educational purposes. We have a six-axis Kuka robot at our faculty. The whole system is not very open and the students have found experimenting, evaluating results and recording the data quite difficult. On the other hand, there is a vendor-made OPC Kuka Server available. On-line connection between the robot and the Matlab could make these things easier. Furthermore, in Matlab there is also on-line and off-line analysis available for comparing the results, optimisation, etc. Moreover, we are seriously considering the feasibility of adding some additional sensors (acceleration, force . . .) and connecting these sensors via different PLCs and OPC servers onto the same Matlab/Simulink window. In this case the amount of information about the whole system would increase significantly, and there is also a possibility for OPC to become a universal platform, as was proposed by Mintchell [15].

CONCLUSIONS

In this paper a solution was proposed on how to connect an existing system to the Matlab environment by using the OPC. Such a connection can be used for monitoring industrial process data which can then be used for the analyses and optimisation of procedures, early fault detection and diagnosis, further data processing or data documentation. The proposed solution is particularly suitable for systems where the industrial process must remain unchanged, or must not be stopped.

The performance testing system consisted of an OPC server which captured data from the controller, it checked the data's quality, evaluated it for a change of state, and then forwarded it to the OPC client who returned it along the same route once it was processed. Synchronous reading was used for the 'worst case' scenario, where each piece of data was changed every time and the same computer ran a server, two clients, as well as the SCADA and Matlab software. The measurements performed show that, due to the fixed dead time, the worst results are obtained when transferring a single piece of data only. A rate of approximately 1000 data per second can be expected with large quantities of data. The process data is available to Matlab virtually at the same time as to the SCADA system which makes the application suitable for industrial environments.

Two examples for use in undergraduate engineering education were also described. The first example describes a procedure where the OPC connection between PLC and Matlab was used in the educational process for teaching control. The real process was connected to the PLC and SCADA system which is quite common in industrial practice. The OPC link to the Matlab was then established and used for several purposes: to record the nonlinear steady-state characteristic of the process, to download controller parameters into PLC and also for On-line identification of the process. The second example describes a proposal for how to use the OPC connection between Matlab and Kuka robot to make students' workload easier. The use of OPC in laboratory experiments has generated many positive results. Student reaction to the experiments has been very good and interest has been increased. The students seem to appreciate the 'feel' that they gain from the laboratory experiments, as opposed to a computer simulation.

The proposed solution demonstrates technologies that are just gaining widespread support today in industry and the examples illustrate that OPC can form a flexible universal data platform for factory floor automation. The techniques employed in these laboratory experiments will likely to be seen by the students in their subsequent employment after completion of their college careers. The quality of graduates is improving and the industry is receiving engineers trained in solving real-world problems. Our goal is to reduce the training period for new engineers when they start work and we believe that our paper is a contribution to this aim.

REFERENCES

1. M. Janke, OPC-plug and play integration to legacy systems, *Pulp and Paper Industry Technical Conference*, Atlanta, USA, 228–295 (2000)
2. Mitsugu Kishimoto, Optimized operations by extended X-Factor theory including unit hours concept, *IEEE Transactions on Semiconductor Manufacturing*, **14**(3), 2001.
3. T. G. Kirner Quality requirements for real-time safety-critical systems, *Control Engineering Practice*, **5**, 1997, pp. 965–973.
4. W. David Coit, Tongdam Jin, Prioritizing System-reliability prediction improvements, *IEEE Transactions on Reliability*, **50**(1), 2001.
5. Linda Wills *et al*., An open platform for reconfigurable control, *IEEE Control Systems Magazine*, **21**(3), 2001.
6. J. Rodgerson, Teaching systems and control using Matlab, *Int. J. Electical Eng Educ.*, Manchester Univ. Press, Manchester, **6**, 1992.
7. OPC Foudation, *Data Access Automation Interface Standard 2.04*, OPC Foundation (2001).
8. Wu Sitao, Using device driver software in SCADA systems, *Power Engineering Society Winter Meeting*, Singapore (2000).
9. P. Studebaker. ALL For One—The Top 10 Trends in Process Control. *Control Magazine* 1, (1998).
10. Al Chisholm, Intelluton Inc. OPC/OLE for Process Control Overview, *World Batch Forum* (1998). www.intellution.com/opchub/opc_wbfoverview.asp
11. M. Stianko, Merz. OPC Server according to the Data Access 1.0 and 2.0 Specification: Technical Descripton, *Czech Avzomatizace Magazine*, **11** (2000). www.merz-sw.com/articles/opc_da.php3
12. Mohamed Am, *et al*., Real-time implementation of a robust H-infinity controller for a 2-DOF magnetic micro levitation positioner, *Journal of Dynamic Aystems Measurement and Control Transactions of the ASME*, **12**, 1995.
13. Al Chisholm., A Technical Overview of the OPC DataAccess Interfaces, *Intellution Inc*. (1998).
14. Frank Iwanitz, Jurgen Lange, *OLE for Process Control*, Huthig Gmbh Heidelberg, Germany (2001).
15. G. A. Mintchell, OPC integrates the factory floor, *Control Engineering*, **1**, 2001.

Stojan Persin received the B.Sc. degree in electrical engineering from the University of Maribor, Slovenia, in 1995. From 1995 to 1999, he was with Metronik, Ljubljana, Slovenia and was a senior engineer for industrial automation systems and afterward a manager at the branch office. Since December 1999, he has been with Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia. He is currently working towards a Ph.D. in electrical engineering. His research interests include industrial automation, building automation, fault detection and diagnosis and intelligent systems.

Boris Tovornik received the B.Sc. degree in Electrical Engineering, from the University of Ljubljana, Slovenia and M.Sc. and Ph.D. from University of Maribor, Slovenia in 1984 and 1991, respectively. He was a teacher of Electrical Engineering school, Maribor 1973, engineer for automation, Ironworks Store 1975, engineer for automation at Drava river Power plants 1977. Since 1978 he has been with Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia. He is Head of Laboratory of Process Automation and holds the rank of Associate Professor. His fields of research interests are computer control of industrial processes, modelling and process identification, fuzzy control, intelligent systems and fault detection.

Nenad Muskinja received the B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Maribor, Slovenia, in 1988, 1992, and 1997, respectively. Since 1989, he has been a faculty member in the Department of Electrical Engineering and Computer Science, University of Maribor, Slovenia, where he currently holds the rank of Assistant Professor. His research interests include industrial automation, adaptive control, sampled-data control, fuzzy control, and intelligent systems.