Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation, and Mathematics

PROCEEDINGS

17th International Conference on Process Control 2009

Hotel Baník, Štrbské Pleso, Slovakia, June 9 – 12, 2009

ISBN 978-80-227-3081-5

http://www.kirp.chtf.stuba.sk/pc09

Editors: M. Fikar and M. Kvasnica

Bisták, P.: Virtual and Remote Laboratories Based on Matlab, Java and EJS, Editors: Fikar, M., Kvasnica, M., In *Proceedings of the 17th International Conference on Process Control '09*, Štrbské Pleso, Slovakia, 506–511, 2009.

Full paper online: `http://www.kirp.chtf.stuba.sk/pc09/data/abstracts/069.html`

VIRTUAL AND REMOTE LABORATORIES BASED ON MATLAB, JAVA AND EJS

P. Bisták

Slovak University of Technology in Bratislava, Fac. of Electr. Eng. & Inf. Technology

Ilkovičova 3, 812 19 Bratislava, Slovakia

fax : +421 2 654 29 521 and e-mail :pavol.bistak@stuba.sk

Abstract: Online learning becomes more active and virtual and remote laboratories have significant contribution to this process. They are growing with expansion of Internet. The paper will describe basic characteristics of virtual and remote laboratories suited for education of control engineers and will give recommendations how to build such laboratories. Their design is based on a Java client server application and the Matlab/Simulink software package. Also the special case when the control algorithm is computed at the client side is considered. The Easy Java Simulations (Ejs) free software package is briefly mentioned in order to create the client interface rapidly and comfortably.

Keywords: virtual and remote laboratories, Matlab, Java, Ejs, COM.

1 INTRODUCTION

Rapid development of Internet influences also education. Online education or e-learning has become very popular. These changes are also reflected in the education of engineers. Traditionally, practical exercises in laboratories are very important for engineering students. Although during this process students are getting technical skills there is still possible to reduce the stay in laboratories. Students can be better prepared for laboratory works in advance when they visit a corresponding virtual laboratory with simulations. A remote laboratory offers even better preparation because students can work remotely with real equipment and measure real signals. And after getting necessary skills for laboratory works from face-to-face lessons students can again use remote laboratories and thus continue experimentation with real equipment.

The principle of presented virtual and remote laboratories is based on Matlab/Simulink and the COM technology for data exchange within the Windows operating system. First we suppose the real system is directly controlled from the Matlab/Simulink. For the purpose of a remote control a Java client server application has been developed. It covers communication and offers an user friendly interface for control of the remote plant and visualization of measured data. Another approach discussed at the end of this paper is represented by the computation of control algorithm at the client side.

The paper is organized in six chapters. After introduction the architecture of virtual and remote laboratories is depicted. Following there is a chapter about virtual laboratories. Then two Java server applications designed for remote laboratories are discussed. The fifth chapter demonstrates the server application facilities when examples of different client applications are shown. The paper is finished with short conclusions.

2 STRUCTURE OF VIRTUAL AND REMOTE LABORATORIES

Virtual and remote laboratories have roots in classical control engineering laboratories where students and researchers carry out experiments in the presence form (Žáková *et al.* 2000). These are usually equipped with sets consisting of a control PC connected through the AD/DA converters to the power electronic of the real system and of course the real dynamical system itself. There is the Matlab/Simulink software installed on the control PC that enables to design different control algorithms but also run identification procedures and get and process signals from the data acquisition cards. Matlab/Simulink offers several toolboxes for the control of fast dynamical systems (e.g., Real Time Control, Windows Target and xPC Target). In industrial solutions Matlab/Simulink is usually replaced by other more professional tool (e.g., LabView) but for the purposes of education Matlab/Simulink is better choice. Thus we can say that the Matlab/Simulink directly controls the real plant. Then we can define the local control (Fig. 1) that is represented by a PC with Matlab/Simulink and a data acquisition card, power electronics and the real system. There are many experiment from the control area using this platform (Müller *et al.* 1999, Safaric *et al.* 2004, Žáková *et al.* 2006).

Another important part of virtual and remote laboratories is given by a client server application that is able to access the data produced by the local control and transfer them to the client side and vice versa transfer the data, commands and requirements of the client to the local control. The client server application uses Internet connection (TCP/IP) and is Web based as many similar products (Bisták *et al.* 2003, Bisták 2006, Huba *et al.* 2006, Schmid 2003, Žilka *et al.* 2008). This enables to the remote client to enter the virtual or remote laboratory form his/her familiar Web browser environment. In fact there are two communication channels that are used by the server application to intermediate communication between the client and the local control. The communication between the server and the client is realized over the Internet and so the quality is namely influenced by the speed of the connection. On the other side the server application has to communicate with the Matlab/Simulink application that is a part of the local control. This communication depends on the operating system under which the server and Matlab/Simulink are running. Under Windows operating system we used the COM technology for the data exchange between two applications.

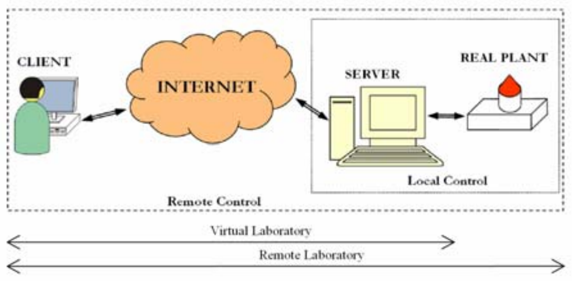


Fig. 1: Remote laboratory architecture

To complete the structure of the virtual and remote laboratories it is necessary to mention the administrative system that is useful to cover activities connected with user accounts, booking of experiments, editing Web pages related to the description of experiments, help and support system, safety issues, etc. Then we can summarize the structure of virtual and remote laboratories into following points (starting from the local control towards the client):

- Local control (different control algorithms running on a local PC with Matlab/Simulink equipped with AD/DA converters together with the real system)
- Server application (it creates a bridge between the remote client and the real plant, it transfers remote user's commands and transmits responses in the form of measured data and messages)
- Client application (remote user interface that enables to control a specific real plant usually in the form of JAVA applet running within Web browser)
- Management (the application that covers administrative issues like the access to the remote laboratory, reservation of a specific real system for remote experimentation, upload of files, help and support, safety issues, etc).

The architecture of virtual laboratories is simplified because there is no real plant present (Fig. 1). The simulation can run on the server side or can be downloaded and run on the client side (Huba *et al.* 2007).

In this paper we will briefly describe the idea of virtual laboratories and then we will concentrate on remote laboratories designed for control engineering purposes.

3 VIRTUAL LABORATORIES

To build a virtual laboratory is not so complicated task. Usually the virtual laboratory can be approached through the Internet from the browser's interface. The simplest way is to program Java applet that can be placed on the Web server. A remote client downloads the applet and runs it within his/her browser. Java offers good possibilities for programming numerical calculations and creating visualization suitable for simulations. As an example the Java applet for simulation of two tank hydraulic system can be seen in Fig. 2.

Of course no everybody can be a Java programmer. For those it is recommended to use a software tool called Easy Java Simulations (Ejs) (Martin *et al.* 2005). Ejs can be downloaded for free from http://www.um.es/fem/Ejs/. This software package is suitable for producing simulations in Java without knowing Java programming. Java applets are one of the outputs of Ejs. Instead of writing Java code the "programming" in Ejs is based on graphical interface when a user fills an appropriate card with variables, equations and the visualization consists in selection of graphical elements and setting their properties (Fig. 3). As an example graphical user interfaces for the magnetic levitation system and the thermo-optical system produced in Ejs are shown in the Fig. 4 and Fig.5.

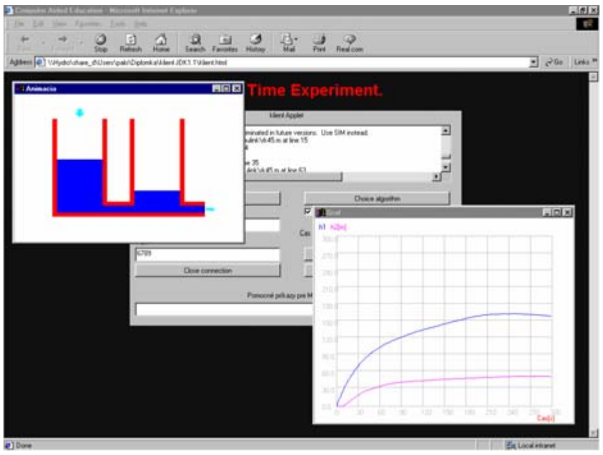
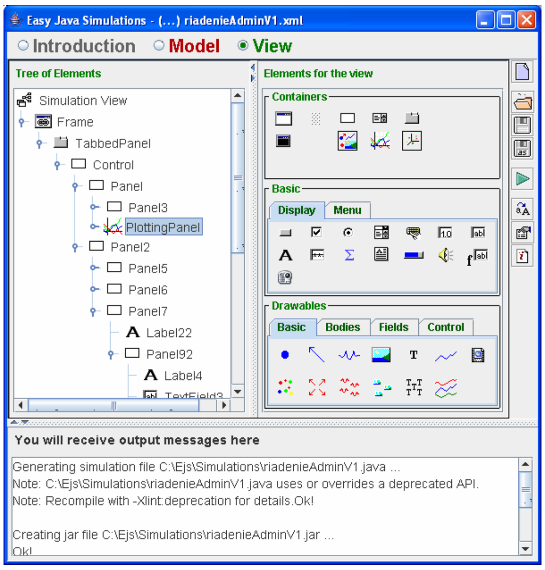


Fig. 2: Java client for the two-tank system

We have to admit that there are some limitations using Ejs. The remote user interface can be designed by a Java programmer more user friendly because he/she can manipulate it in details and thus cover almost all requirements. But in many cases the Ejs is sufficient.



You will receive output messages here

```
Generating simulation file C:\Ejs\Simulations\riadenieAdminV1.java ...
Note: C:\Ejs\Simulations\riadenieAdminV1.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.Ok!

Creating jar file C:\Ejs\Simulations\riadenieAdminV1.jar ...
Ok!
```

Fig. 3: Easy Java Simulations View

4 SERVER APPLICATIONS FOR REMOTE LABORATORIES

In order to enable the control of the real systems via Internet we had to create a bridge between the Matlab application (that really controls the real plant) and a remote client. Such a bridge is represented by a server application.

The server is an application that listens at a particular port and responds to clients. The server application is the central part of the whole system. In our case its main function is to transfer commands and send the output data from the controlled plant to the client. Another necessary feature is that the server should co-operate with the Matlab/Simulink application that really controls the plant. On the Windows platform there exists several ways how different applications can exchange data (DDE, ActiveX, COM, etc.).

4.1 COM Technology

Matlab supports several possibilities for data exchange with other applications. The DDE technology is no more developing from the Matlab version 5.1. The other technology is called Component object model (COM). COM technology has been developed by Microsoft. It defines a language independent binary standard for component interoperability. It is used to enable interprocess communication and dynamic object creation in any programming language that supports the technology. Using COM, developers and end users can select application specific components produced by different vendors and integrate them into a complete application solution. The essence of COM is a language neutral way of implementing objects such that they can be used in environments different from the one they were created in, even across machine boundaries.

Using COM technology Matlab can control another component or be controlled by another component. In the second case Matlab plays a role of the server and the component is a client. Matlab can run as the Engine Server or the Automation Server. In the second possibility it enables the use of Real Time Workshop toolbox within the Matlab environment. When Matlab is running in the mode of the Engine Server the Simulink block diagrams could not be recompiled and built. For fast real plants the use of Real Time Workshop is necessary.

4.2 Java Server Applications

To provide an open solution we decided for the platform independent Java programming language (although for the server application it is not so important as for a client application). The server application runs on the PC connected directly to the certain real plant and its operating system determines the platform. Following parts describe two Java server applications depending on the mode in which the Matlab is running. Both these application are based on the COM technology. In the first case Matlab is acting as the Engine Server and in the second one as the Automation Server.

Java does not support COM technology directly (instead it supports the later CORBA technology that is missing in Matlab). We avoid this by using the JMatLink library and Microsoft SDK for Java package.

4.3 Java Server with JMatLink Library

According to its name, JMatLink is a library of classes which allows the communication between programs written in Java and Matlab. The library was created by Stefan Müller and it can be downloaded from Web page http://jmatlink.sourceforge.net for free. When using JMatLink library the Matlab runs as Engine server.

The JMatLink enables the access from the server application written in Java to the Matlab environment. It allows to assign commands to the Matlab as these were written in the command line of the Matlab. In the similar way it is possible to get values from the Matlab Workspace. The functions of the JMatLink enable to run the Matlab Engine, execute Matlab commands, get multidimensional variables from the Matlab Worskpace, return the Matlab output of the previous Matlab command and stop the Matlab Engine.

The server application communicates with the client application through the TCP/IP connection. After the client establishes the connection with the server application the server runs the Matlab Engine and waits for the next commands that it immediately transfers to the Matlab. Of course, the necessary condition is that there must be a set of Matlab commands that can fully control the real time experiment. Most of such commands are realized through the call of the set_param Matlab function.

Matlab Engine is useful as computing power but when using Matlab Real-Time Workshop we faced a problem. Matlab Real-Time Workshop needs to compile block diagrams to C language using internal or some of the external compilators. This compilation can not be run from Matlab Engine because it does not support this option. Therefore, using JMatLink, we can control only those real systems, which communicate with Matlab in different way (for example using Windows Real-Time Toolbox) and their control block diagrams do not need to be compiled. To cope with this problem we developed the next Java server application.

4.4 Java Server with Microsoft SDK for Java

Dan Adler created project JACOB (Java Com Bridge) that enables the access to COM Automation components from Java applications (Web page: http://sourceforge.net/ projects/jacob-project). Similarly to JMatLink, it uses native methods JNI. The JACOB project has been inspired by another product called Microsoft SDK for JAVA that was our final solution. When compiling Java source with JACOB libraries Java Virtual Machine does not reported any mistake but we were not able to get any matrices data from Matlab into our Java application. Therefore we changed JACOB libraries for Microsoft SDK for JAVA libraries and compiled the source using Microsoft jvc.exe compiler. In this way we were able to fully control Matlab from our Java application.

MATLAB Automation server capabilities include the ability to execute commands in the MATLAB workspace, and to get and put matrices directly from and into the workspace. Beside this they also allow to control the server window and to stop the program.

This server application is suitable also for control of real systems using Matlab Real-Time Workshop. Thus, for instance, the magnetic levitation system can have been involved in our remote laboratory system.

5. CLIENT APPLICATIONS FOR REMOTE LABORATORIES

For the client part it is very important to be independent of the operating system. In other way the number of users is limited. Therefore the final form of the client application is the Java applet that can be easy executed within the Web browser environment.

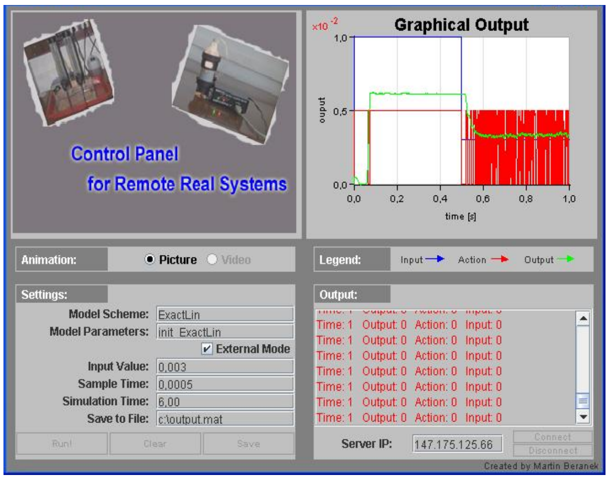
The client is an application that connects to a specific port on the server and exchanges data. Our client application should enable

- to connect and disconnect to the server
- to modify controller or experiment parameters and structure
- to start and stop the experiment
- to gather and visualize measured data in the user friendly environment

After establishing connection with the server the client sends Matlab commands written by the user and waits for responses. The responses can be displayed in the form of text, numerical data, graphs, animations or video clips.

To enable comfortable modification of the client's graphical user interface and visualization of measured data the Java client application has been created within the Easy Java Simulations software package. Only the communication part of the application had to be written in the Java code and was placed on the Custom card of Ejs development environment.

The rest of the client application has been easily designed using graphical tools that the Ejs offers. In the future this enables the quick modification of the client application also for those who do not know Java programming what is the main advantage of this approach.

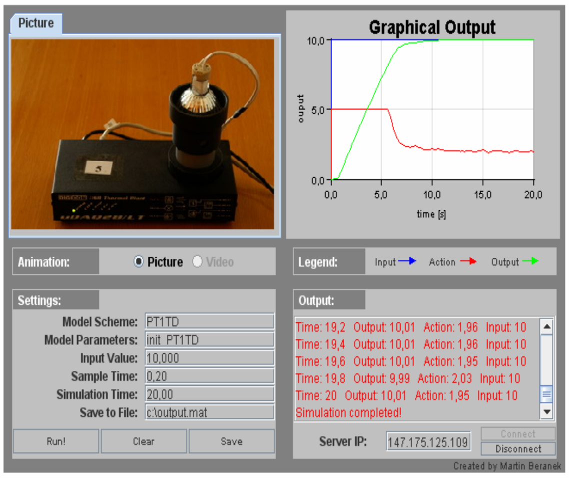


Settings:
Model Scheme: ExactLin
Model Parameters: init_ExactLin
✓ External Mode
Input Value: 0,003
Sample Time: 0,0005
Simulation Time: 6,00
Save to File: c:\output.mat

Output:
```
Time: 1   Output: 0   Action: 0   Input: 0
Time: 1   Output: 0   Action: 0   Input: 0
Time: 1   Output: 0   Action: 0   Input: 0
Time: 1   Output: 0   Action: 0   Input: 0
Time: 1   Output: 0   Action: 0   Input: 0
Time: 1   Output: 0   Action: 0   Input: 0
Time: 1   Output: 0   Action: 0   Input: 0
```
Server IP: 147.175.125.66

Fig. 4: Magnetic Levitation System client application

In the Fig. 4 and Fig. 5 one can see two client applications designed for the demonstration of the Java server application facilities. The first one (a) corresponds to the fast dynamical system of magnetic levitation. The second one (b) represents the relatively slow dynamical system of thermo-optical plant. We have shown that the developed Java server applications are suitable for both systems.



Settings:
Model Scheme: PT1TD
Model Parameters: init_PT1TD
Input Value: 10,000
Sample Time: 0,20
Simulation Time: 20,00
Save to File: c:\output.mat

Output:
```
Time: 19,2   Output: 10,01   Action: 1,96   Input: 10
Time: 19,4   Output: 10,01   Action: 1,96   Input: 10
Time: 19,6   Output: 10,01   Action: 1,95   Input: 10
Time: 19,8   Output: 9,99   Action: 2,03   Input: 10
Time: 20   Output: 10,01   Action: 1,95   Input: 10
Simulation completed!
```
Server IP: 147.175.125.109

Fig. 5: Thermo-optical System client application

In our case the window of a client application is usually divided to the visualization part (the upper panel) and the data part (the bottom panel). The visualization consists of animation or video frames and several graphs. The data part is created by the field of control buttons, the field of parameters, and the area of numerical outputs of the experiment. The remote user can interact with the experiment using several control buttons (Connect, Disconnect, Start, Stop, Save data, Clean parameters.) The other possibility is to change parameters. Thus the user can change the time of the experiment, the sampling period, desired values and choose a different type of controller from the given list. Using an FTP server the remote user can add to this list his/her own controller. The user can also transfer the simple Simulink block diagram that can perform identification or different tasks necessary for running the control experiment.

5.1 Control algorithm computed at the client side

Client interface is again built in Ejs software which has been described in previous chapters. Structure of the client is however different as in the approach with Matlab based server part.

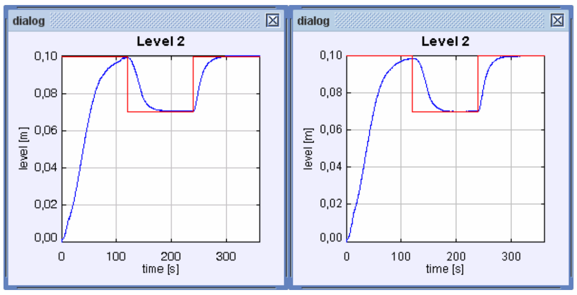
The client communicates with the remote system through the TCP/IP protocol. In this case the remote system does not need a PC server. The communication and the simple server application is implemented in additional interface card that extends the electronic part of the remotely controlled real system. After the connection is successfully established, the client registers server's IP address and sends the "open" command. The connection is then maintained by periodically sending (at given sampling rate) the control string to the plant, while actual measured data are retrieved from the returned string (Žilka *et al.* 2008).

A network lag can cause (especially at higher sampling rates) that returning data from plant arrive later than next sampling period has begun. In that case inputs from the previous sample are used to calculate controller action and a user is notified with the timeout message. A sampling rate should be set appropriately to prevent timeouts as much as possible, while maintaining a sufficient sampling rate for the used control algorithm.

The control algorithm is running in the Evolution card of Ejs. This is the main difference with the previous approach where control algorithm has been calculated on the server side. This time the algorithm is realized in the Java language and calculated on the Evolution card of the client built in Ejs. The control algorithm is called by internal Ejs command _step() each sampling period. The time is controlled by the client. Due to network lag it can happen that new measured data are not available as mentioned above. The measured and sent data are continuously visualized during the control process.

Using this approach the network control problem can be studied. It is possible to compare the advantages and disadvantages with respect to the control realized at the server side. The control process has been represented by the hydraulic plant that belongs to the slow processes. Therefore it has sense to apply also the network control. From the output responses (the height of the level in the second tank) in the Fig. 6 one can see the controller calculated at the client side works well in this case.

It is necessary to mention that the client using mobile Internet connection (Fig. 6, b) has encountered a great number of timeouts caused by network lag. With sampling period of 0.333s there was obviously not enough time to receive, calculate and send data in each sampling period. Timeouts occurred almost in each second sampling period. By increasing the sampling period to double value, the number of timeouts has been decreased significantly. Several timeouts (approximately one per one hundred of samples) appeared also when the network control was carried out within the same local area network (Fig. 6, a). Due to relatively slow behavior of the process, we can conclude that the number of timeouts does not reasonably influence the quality of the control process. On the other hand the network control offers remote users to design and modify their own control algorithms and they are not limited to Matlab.



a) b)

Fig. 6: Output responses. Client has been running a) within the same LAN, b) outside the LAN using mobile Internet connection

6 CONCLUSIONS

The paper showed that there are several possibilities how to build virtual and remote laboratories. We found Easy Java Simulations software as a very useful to create Java applets and to design remote user interface as well. The proposed Java server applications based on COM data exchange are suitable for remote experiments with slow systems (hydraulic plant or thermo optical plant) or for fast systems (magnetic levitation or inverted pendulum). The network control approach is suitable for slow systems and again the client with computing the control law can be realized using Easy Java Simulations. The additional costs for building remote laboratories are low. We use platform independent Java solutions and except of Matlab there is no commercial product. In future we plan to replace Matlab by Scilab/Scicos.

ACKNOWLEDGMENTS

This work was supported in part by the Slovak Scientific Grant Agency (VEGA) Project No. 1/0656/09.

7 REFERENCES

Bisták, P., Žáková, K. (2003): Organizing Tele-Experiments for Control Education. *The 11th Mediterranean Conference on Control and Automation MED'03*, Rhodes, Greece.

Bisták, P. (2006): Remote Control of Thermal Plant Using Easy Java Simulation. *Int. Conf. on Interactive Computer Aided Learning ICL'06*, Villach, Austria.

Huba, M., Bisták, P., Fikar, M., Kamneský, M. (2006): Blended Learning Course "Constrained PID Control". *7th IFAC Symposium on Advances in Control Education ACE'06*, Madrid, Spain.

Huba, M., Šimunek, M. (2007): Modular approach to teaching PID control. *IEEE Transactions on Industrial Electronics*, Vol.54 No.6 December 2007, pp.3112-3121.

Martin, C., Muñoz, R. (2005): A Distance Learning Course on Virtual-lab Implementation for High School Science Teachers. *6th International Conference Virtual University VU'05*, Bratislava, Slovakia.

Müller S., Waller H. (1999): Efficient Integration Of Real-Time Hardware And Web Based Services Into MATLAB. *11th European Simulation Symposium*, Erlangen, Germany.

Schmid, Ch. (2003): Internet-basiertes Lernen. *Automatisierungstechnik*, 51, No. 11, pp. 485-493.

Safaric, R., Uran, S., Trunic, M., Hedrih, I.(2004): Remote Controlled Mechatronics Device via Internet using Matlab. *1st Int. REV Symposium*, Villach, Austria.

Žáková, K., Huba, M., Zemánek, V., Kabát, M. (2000): Experiments in Control Education, *IFAC Int. Conf. on Advances in Control Education ACE 2000*, Gold Coast, Australia

Žáková, K., Sedlák, M. (2006) Remote Control of Experiments via Matlab. *International Journal of Online Engineering (iJOE)*, Vol. 2, No. 3.

Žilka, V., Bisták, P., Kurčík, P. (2008): Hydraulic Plant Remote Laboratory. *International Journal of Online Engineering*, ISSN 1861-2121, vol. 4, Special Issue 1: REV2008, July 2008, pp. 69-73.