Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Two Ways for Remote Plant Control

Katarína Žáková and Mikuláš Huba

Slovak University of Technology in Bratislava
Faculty of Electrical Engineering & Information Technology
Ilkovičova 3, 812 19 Bratislava, Slovakia
(e-mail: katarina.zakova@stuba.sk, mikulas.huba@stuba.sk)

Abstract: The paper presents several possibilities of remote control of experiments. One direction of the research is devoted to the use of Matlab software environment. There are number of ways how to approach Matlab remotely. They include e.g. Matlab S-function, COM, DDE, shared communication file or Virtual Reality Toolbox. One of these proposed concepts was used and tested for control of the inverted pendulum system. This approach was compared with the method enabling to approach the plant directly without using Matlab package. Finally, the gained experience is discussed in the paper.

1. INTRODUCTION

Technology is increasingly changing our lives. First, it was the emergence of the desktop computer, now it is the Internet. It enables access to never-ending quantities of new information and knowledge. As more individuals become connected, the Internet will penetrate deeper into our everyday activities, including the way we learn. The term e-learning covers a wide set of applications and processes, including computer-based learning, Web-based learning (online Learning), virtual classrooms, and virtual & remote collaboration. E-learning is much more than just the delivery of content via all electronic media, including the Internet, intranets, extranets, satellite broadcast, audio/video tape, interactive TV, and CD-ROM. It also offers tools for performing tasks, for communication, management, administration, assessment and evaluation. Feedback, interactivity and integration are the most important keywords characterising role of e-learning in our education.

In engineering education providing hardware and software components of existing laboratories via the Internet creates a base for establishment of virtual laboratories. The main motivation for using plants in the educational process is clear physical "visibility" of the controlled dynamics, and also authenticity enabling to exercise all design steps starting with the plant identification and ending with the evaluation of the control results achieved with the particular model.

Design of control applications that are available via Internet is oriented in two directions: control of virtual devices and control of real physical plants. The first possibility enables to simulate the virtual model on a computer and distribute it on CDs. Combined with Internet access and WWW it enables to offer it to students as an online animation, too. Simulations and animations are especially important in areas where the nature, safety and costs of experiment do not allow to experiment on real plants. However, using the animation models cannot substitute the work with real physical plants that always demonstrate some unmodelled dynamics, parasitic noise, delays, friction, etc. Unfortunately, the number of students is high in comparison with the number of available real plants. A possible solution of this problem is building of remote lab that opens learners access to laboratories via Internet.

For accomplishing the remote control of experiment it is necessary to create two applications: for the server and for the client.

The paper demonstrates how to use remote control for the inverted pendulum model. The inverted pendulum is one of the most important classical problems of control engineering. It is a well-known example of nonlinear, unstable control problem. The inverted pendulum is related to rocket or missile transport, where thrust is actuated at the bottom of a tall system.

2. PLANT

The inverted pendulum consists of a thin rod attached to a moving cart. Whereas the pendulum is stable when hanging downwards, the vertical inverted pendulum is inherently unstable, and must be actively balanced in order to remain upright, typically by moving the cart horizontally as part of a feedback system.

The plant is represented by a revolving pendulum mounted on top of a moving base [11]. By a DC-motor, a toothed wheel, a toothed belt and a clutch, the moving base can be driven along a guiding bar over a length of approximately 1.5m in such a way that the pendulum is stabilized upright at a preassigned position. The stabilization of the pendulum is accomplished by a digital controller. Based on measurements (the pendulum angle and the cart position obtained by incremental encoders), the controller generates a suitable signal, which controls the DC-motor by an electronic drive.

3. MATLAB POSSIBILITIES

The aim of this section is to discuss some of possibilities of a remote approach to the plant. Our interest was dedicated to use of Matlab like a simulation engine for the whole experiment. We do not mention here the standard solution – Matlab Web Server – since it is not more supported by The MathWorks, Inc.

3.1 Matlab Dynamic Data Exchange (DDE)

Dynamic Data Exchange is a technology developed by Microsoft Corporation that enables communication and data exchange among various Windows applications. DDE doesn't need to be installed separately, it is a part of Windows operating system. Matlab contains functions enabling full duplex approach of Windows applications to Matlab. These functions use Dynamic Data Exchange. However, the communication can usually run only locally.

As written e.g. in [12] the communication starts after establishing the so called DDE conversation. The application starting a communication is called a client and application answering to the client is a server, i.e. we can talk about the client/server application. The client has to identify two DDE parameters that are defined by server: the name of application that should answer to the client prompt – service name and subject of communication – topic. When the server receives a request for opening the conversation channel, it verifies the topic and if it is supported, it allows the client connection. The combination of two obligatory parameters uniquely identifies the conversation.

In general, Matlab can act both as a client and as a server as well – despite of the fact that the second possibility is maybe more frequent.

A client application can access MATLAB as a DDE server in the two ways, depending on the client application. If we can use an application that provides functions or macros to conduct DDE conversations, the simplest way is to use these functions or macros. If we decide to create our own application, we can use the MATLAB Engine Library or DDE directly [12].

The successful realized local communication with Matlab enables to step forward and to try to implement TCP/IP protocol to approach Matlab remotely. In this case we transform DDE client to TCP/IP server that can be approached by remote clients via Internet. The structure of such connection is shown in Fig.1.

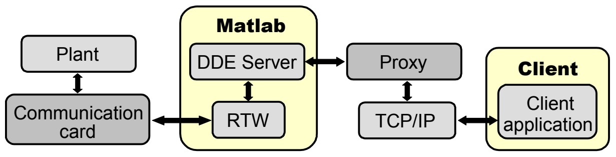


Fig. 1. Client/server communication via DDE

However, the realization of this solution is quite time demanding since it requires to program TCP/IP server and also client application. The client implementation is very often done in Java environment because of its platform independence. The advantage is that this development enables full optimalization of system tools and functions that are needed for communication.

3.2 JmatLink library

The communication with Matlab can also be realized using the dll library JmatLink [14]. It is a product from Stefan Müller programmed in C language that using ActiveX tool approaches data and services of Matlab. The library is implemented in the class of Java language that has the same name (JMatLink.class). In the case that we use Java for development of our client application we have available all methods of the mentioned class.

At the communication it is necessary to ensure data transfer in both directions: from client application to Matlab and from Matlab to client application. The client can be formed by a Java applet and the server can be designed as a Java application whereby both client and server communicate each to other using TCP/IP protocol. The plant is connected to the server where the Java application and Matlab is installed. The server guarantees the connection and data transfer with Matlab software. All needed m-files for simulation are placed on the PC server where Matlab is installed.

The connection between the server and Matlab can be established in two ways. It is possible to create an own class of native methods or the already created class of connecting methods can be used. We decided to use the second alternative - JmatLink library [14].

TCP/IP protocol that is used for communication in Internet ensures the failure-free data transfer. The data transfer between server and client can be accomplished using sockets that enable direct approach to network protocols of lower level. In general, when a socket of the server waits for connection with client, the application is blocked. Therefore it is necessary to create the server with several threads. It is able to open separate channel for each client. The server waits for client's request for connection. If there is a free thread (the number of attached clients is smaller than the allowed number of clients) the request of client is accepted and the communication channel between server and client is created. The server should keep information about the given IP address, control algorithm and simulation scheme.

Then, the server waits for instructions from client. They contain parameters of the system and the required simulation. The instructions are transformed to the Matlab format and sent to the specified Matlab workspace using JmatLink library. The problem can be to force Simulink to give data to workspace during the whole run of simulation and not only after the simulation stops. However, this task can be solved.

After simulation Matlab sends simulation results back to the server. The server transforms these data to the required format and sends them to the client where they can be visualized as an animation and/or a graph.

3.3 Component Object Model

Component Object Model (COM) is a software architecture developed by Microsoft to build component based applications that can be called up and executed in a Windows environment. A program can call the object whenever it needs its services. Standard applications, such as word processor and spreadsheets [13], can be written to expose their internal functions as COM objects, allowing them to be "automated" instead of manually selected from a menu. For example, a small script could be written to extract data from a database, put it into a spreadsheet, summarize and chart it, all without manual intervention. The similar functionality is also supported by Matlab. The user can approach the most of object parameters via interface that enables communication between COM client and COM server. Matlab supports the only interface and it is called IDespatch. Except of the interface the programmers need to know what approach methods they can use. The simplest one is the already mentioned Automation method also supported by Matlab.

3.4 Communication via file

This type of communication is in its nature very simple. It uses normal Web server where we can also have our web pages and Matlab functions that enable to read from files.

The communication functions on the base of shared file (see Fig.2). User can enter own simulation parameters and commands in the form that can be filled in via web browser. After its submitting they are written into the file placed on server. Simulation is running infinitely long time and Matlab reads the file in regular time instants.

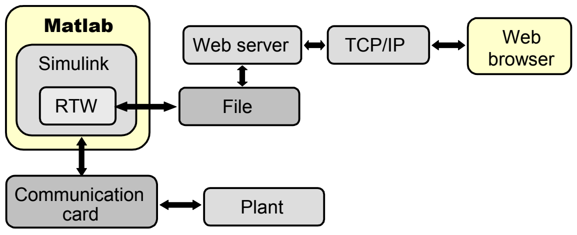


Fig. 2. Communication with Matlab via file

The big advantage of the method consists in the fact that the block scheme prepared in Simulink is attached to the real plant permanently and therefore visualization via Virtual Reality toolbox can also be used. However, there is also one drawback excluding this method. Real-time Workshop does not support I/O functions with files. Therefore this solution is suitable only for simulations without RTW.

3.5 S-function

Next possibility for remote communication with Matlab is to use S-function. S-function is a block in Simulink, i.e. Matlab graphical simulation environment. It can be written in one of compatible language, e.g. C, C++, Ada, Fortran, Matlab, etc. S-functions can be compiled as MEX files and dynamically linked to Matlab whenever it is needed.

S-functions use special syntax that enables interactions with Simulink and its ODE. The structure of S-function allows to create continuous, discrete and also hybrid functions. In this way user can create own blocks that can also be used with Matlab Real-Time Workshop.

Let's consider that we need to transfer data from Matlab to user. S-function can be realized in two ways. It can be used as a TCP/IP client or a TCP/IP server. In the first case the block representing the TCP/IP client is placed in the Simulink model. We have to ensure the transfer of measured signals to the user that can visualize results e.g. by means of Java applet in web browser. Then, it is necessary to create a application (e.g. Java application) that consists of two TCP/IP servers. The TCP/IP client in S-function connects to one of these servers and the client in Java applet to the second one. TCP/IP servers in Java application only move data between themselves and in such a way send data to user. The second solution is less complicated because it is not necessary to use an additional Java application. TCP/IP server realized in Matlab S-function can directly connect to TCP/IP client on the user side that can be again created as Java applet.

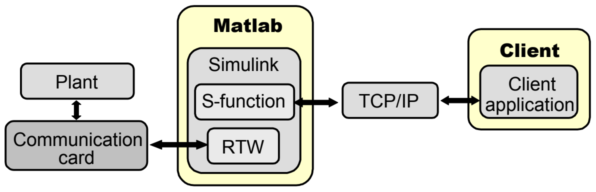


Fig. 3. Communication via TCP/IP using S-function

The disadvantage of S-functions consists in the fact that if we want to use them in combination with Real-Time Workshop we cannot use any Windows API functions, any I/O functions for communication with files and many other functions that could cooperate with timers. In Fig.3 we can see a basic block scheme illustrating communication between remote client and Matlab using S-function.

4. MATLAB REALISATION

The previous section summarised various alternatives how to use Matlab engine for remote control of plant. Now, our attention will be dedicated to the method where the communication is partially based on the use JmatLink library. Since it was necessary to build the server and also client side of experiment we will describe step by step both.

4.1 Server

The server is an application that listens at a particular port and responds to clients. It has 3 main tasks: it has to reply to all requests from clients, it sends information to Matlab and it receives information from Matlab.

The server application could be any of standard servers like http, ftp or a custom server made for the specific application. The another possibility is to create a Java proxy server that will interpret user commands from the client application to Matlab. For this purpose it is possible to use a dll library JmatLink [14] that accesses services and data of Matlab. The library is implemented to the simple class JmatLink.class that can be used inside of created Java proxy server.

Proxy server was realized as a Java application. On the one side there is a TCP/IP server that receives commands from the client (Java applet). These commands are transformed to the Matlab commands and sent to the Matlab workspace using methods of JMatLink library (Fig.5).

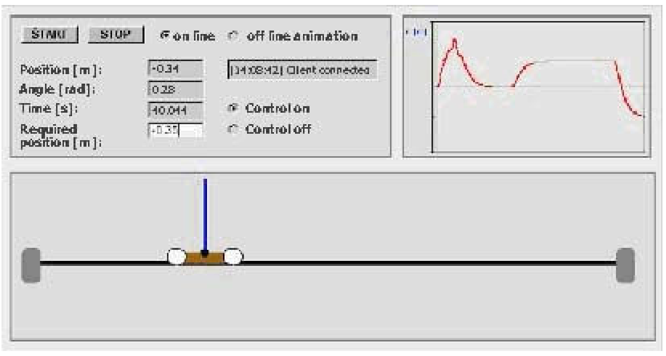
Proxy server has 3 basic functions: to start the real time simulation, to stop the real time simulation and to change the model parameters during the simulation running.

4.2 *Client*

In general, TCP/IP client is an application that connects to a specific port on a TCP/IP server and exchanges data either as a stream or text.

The client application enables user to run the simulation via the web page interface. In a general way the user can use own parameters and choose among presented controllers. The application can also give user a possibility to follow simulation results, save them and use them later in the same application or in another software environment. It is installed on a web server that can be approached by a student via Internet. It is usually realized as Java applet. In Fig.4 the applet for an inverted pendulum control is shown. After its opening in a user web browser, TCP/IP client connects to the proxy server and to the S-function in the Simulink model. An user is informed about the time of connection and the connection status. After the client is connected, everything is prepared for real-time simulation.

The user can enter a required position of the pendulum. The buttons „Start“ and „Stop“ serve for the control of simulation running. During the simulation user can follow numerical values of the carriage position, deviation angle of the pendulum, time of simulation and the graphical dependence of the carriage position. The simulation results can also be visualized via the model animation.



Position [m]: -0.34
Angle [rad]: 0.28
Time [s]: 40.044
Required position [m]: -0.35
[14:08:42] Client connected
[✓ Control on] Control off

Fig.4. The realised client application

Connection between Matlab and Java client can be realized directly using TCP/IP communication. TCP/IP server can be realized by Matlab S-function. It enables transfer of data from Matlab to the connected TCP/IP client (Java applet). The Matlab S-function operating as TCP/IP server is placed in Matlab/Simulink model whereby the input of this block is formed by signals that should be transferred to the client.

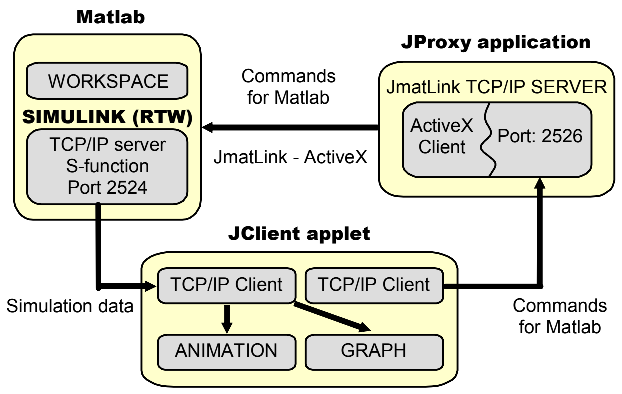


Fig.5. Communication between server and client

It is to say, this solution is demanding for hardware of server since Simulink has to run in real time and in the same time it has to save data to the Matlab workspace. In addition, ActiveX service also has to access to the same workspace. All these factors can freeze Matlab on slower computers.

5. REMOTE CONTROL WITHOUT MATLAB

In spite of the fact that Matlab brings a comfortable way of plant control, it is not always available. Therefore in the following sections we tried to devote our attention to the remote control of experiment that doesn’t require to use Matlab engine for simulation. Let’s start with summarising of facts that we wanted to take into account for server and client side of experiment.

5.1 *Server Side*

The choice of programming tools for the server side of experiment was influenced by several requirements:

- Expansibility of the experiment – the experiment should be easily spread without need for long study of the source code.
- Maximal use of system tools – the effort was to skip all intermediate stages between application and operating system.
- Accessible hardware and software – it was necessary to take into account the most used systems and software environments at the universities and their accessibility for users.

The first requirement can be fulfilled by the selection of the programming language with good support of the object-oriented programming. One can choose from many languages such as .NET, Basic, Delphi, C++, Java, Python, and JavaScript. Matlab with its toolboxes also offers an alternative to the design of remote control of plants [9].

The second requirement excludes the JavaScript and similar scripting languages since these languages are only the presentation languages. They don’t send commands to the operating system directly and therefore they need an additional tool for communication between application and operating system. The Java language brings the similar situation. In spite of the fact that Java is a multi-platform instrument it also needs an extra tool for communication with application. It can be easily understood that such indirect connection between the operating system and application limits the fluent running of the application and its velocity. Simply said, such applications are slower.

Finally, at the choice of the programming tool we had to realize that the most of computers at the university is equipped by the Microsoft Windows operating system and therefore the design should be oriented in this way.

Summarizing these facts we decided to use .NET technology since it is the product of the same company as the operating system that is installed on the computer taking care about the control of the inverted pendulum. This solution should provide the maximal compatibility between the operating system and the created application.

5.2 Client Side

The realization of the client side also requires considering various aspects of its design. It is needed to adjust requests of users, programmers and possible technical limits of application. The chosen tool should be suitable for animation, i.e. it should guarantee smooth animations. It should enable simple Internet communication (the communication can be implemented e.g. using sockets). Finally, the requirement is to create a multi-platform application. The platform independence is important because the application is presented to users via Internet. These requirements can be fulfilled by several solutions. Let's introduce some of them. One can use:

- combination of JavaScript and CSS. In this case the graphical layout of the HTML web site is ensured by the Cascading Style Sheets that can be dynamically changed by JavaScript. The disadvantage is that the animation is not very smooth and there is relatively complicated communication with the server.
- Java applet. Since Java enables a comfortable network programming, the communication is very simple. However, there still exist the problem with smooth animation without a blinking. This problem can be solved, but it is uselessly tricky and complex.
- Macromedia Flash animation. This software enables to create nice animations that are independent on the platform. In addition, using ActionScript offers a simple control of application and Internet communication on the base of XML sockets.

We decided to use the third mentioned possibility.

6. .NET & MACROMEDIA FLASH REALISATION

Similarly, as in the case of remote experiment supported by Matlab, let us describe server and client side of experiment separately.

6.1 Server Side

The server application enables parallel processing of several tasks. This is enabled by using several threads (Fig.6).

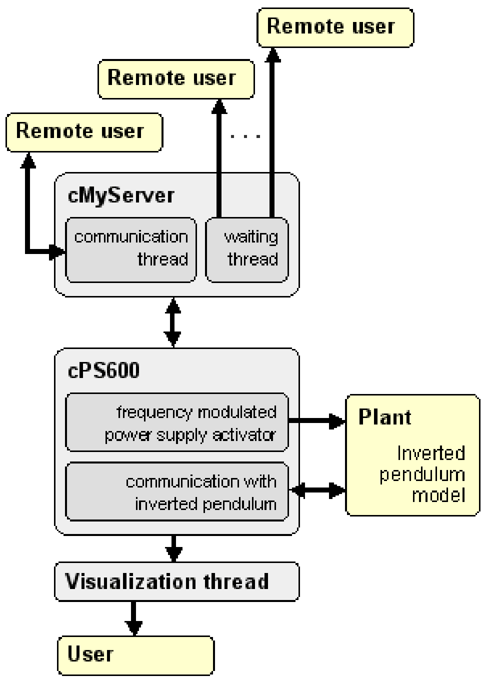


Fig.6. Overview of threads in the server application

Firstly, it is necessary to activate the power converter. The output of the A/D card Option 600-3 gives only the low-voltage unified signal ±10V. Since the used DC motor can work with much better performance that the card can supply the provided signal has to be amplified.

Next, one has to detect the actual state of remote experiment. If the control is running, the value of the control signal is computed and it is sent to analogue output of the A/D card. As soon as the time determined by the sampling period elapses, the whole cycle for computing the control signal is repeated.

The next thread is responsible for visualization of results to the possible local user. He or she can receive numerical information about the actual position of the pendulum and its angle. Moreover, there is also visualized a simplified version of the animation.

Further, there are 2 threads that ensure communication with the remote client realized by means of the Flash animation. The task of the first one is to wait for the request of the first remote user, to process this request and to give him or her back the asked data. Meanwhile, the second thread informs all next interested remote clients that the experiment is running and they have to try to connect later. As soon as the experiment ends the first thread is again prepared to serve to the first interested remote user.

The first step to start the remote experiment is to execute the server application. Its activating causes that the equipment is synchronized, i.e. it is initialized into the initial zero position situated in the middle of the pendulum path. The processor tick counter is also synchronized with the processor.

The server application presents a simple local graphical user interface that enables administrator to set and verify most of activities that will be later available to the remote user. The application enables to start and stop the plant control, there is the sketch animation and finally the area for the setting of parameters. The application seems to be very simple. However, their main task is to enable to client to connect to the inverted pendulum remotely.

6.2 Client Side

As it was already told the client application was prepared using Macromedia Flash. For the communication with server it is needed to create a new instance of the native class XMLSocket. Firstly, user has to connect to the server. For this purpose he or she has to specify the server IP and the number of the port that is used for communication. Then, the client-server communication is ensured by the following two methods (Fig.7).

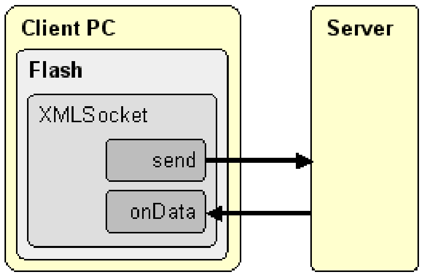


Fig.7. Client-server communication

The method onData() enables to receive and process data from the server. The data are in the XML form and therefore they have to be separated to single values using a XML parser. Only after parsing, data can be used for numerical and graphical visualization of results.

The send() method is used for sending all requests to the server. ActionScript processes the request for actual results from the remote plant and another requests are related to the setting of the remote plant. It is needed to set the pendulum required position, the sampling period and also to switch on and off the whole experiment.

The final client application enables to visualize results. The main attention is devoted to the model animation. User can also follow numerical values of the required position of the pendulum, its actual position, angle and the elapsed time.

Using the client application menu user can connect to or disconnects from the server, can start and stop the control of the plant provided by the predefined controller and he or she can switch between the setting and graph mode of the window. In the setting mode user determines parameters for the remote control. The graph mode shows the graphical dependencies of all followed variables (position, angle, control variable). The measured data can also be saved into the local file and visualized later in some external software environment, e.g. in Matlab, Excel, etc.

7. CONCLUSIONS

The paper presents two ways of the remote experiment realization. The first one exploits advantages of the standard simulation package Matlab. Using Java language it was possible to build a bridge between the server application and the real plant. The main contribution of the second realisation offers an atypical way of the solution. The client application is designed using the widespread multimedial format Flash. It enables to create a simple and effective animation that has only low requirements on the velocity of Internet connection. The server application working on the Microsoft Windows platform can also be easily used without a deeper knowledge of the programming code. From this reasons the presented solution can be a good alternative to other techniques that enable to design and accomplish experiments in remote laboratories.

ACKNOWLEDGMENT

The work has been supported by ESF Project JPD BA3, No. 13120120183 and by Slovak Grant Agency and Grant VEGA 1/3089/06. This support is very gratefully acknowledged.

REFERENCES

[1] P. Bisták, K. Žáková, “Organising Tele-Experiments for Control Education,”. *11th Mediterranean Conference on Control and Automation*, Rhodes, Greece, June 2003.
[2] P. Karagiannis, I. Markelis, K. Paparrizos, N. Samaras, A. Sifaleras, “E-learning technologies: employing Matlab web server to facilitate the education of mathematical programming”, *Int. J. of Math. Education in Science and Technology*, Vol. 37, No. 7/15 Oct. 2006.
[3] J. Liguš, J. Ligušová, I. Zolotová, “Distributed Remote Laboratories in Automation Education,” *16th EAEEIE Annual Conf. on Innovation in Education for Electr. and Information Eng.*, Lappeenranta, Finland, June 2005.
[4] F. Michau, S. Gentil, M. Barrault, “Expected benefits of web-based learning for engineering education: examples in control engineering”, *European Journal of Engineering Education*, Vol. 26, Number 2/June 1, 2001.
[5] S. Müller, H. Waller, ”Efficient Integration Of Real-Time Hardware And Web Based Services Into MATLAB”, *11th European Simulation Symposium*, October 1999, Erlangen, Germany.
[6] M. Repčík, K. Žáková, “Remote Control of Inverted Pendulum”, *Int. Conf. on Remote Engineering & Virtual Instrumentation*, Porto, Portugal, 23.-27.6.2007.
[7] Chr. Schmid, “Virtual Laboratory for Engineering Education,” *ICDE conf.*, Vienna, Austria, 1999.
[8] K.Žáková, M.Huba, V.Zemánek, M.Kabát, “Experiments in Control Education,” *IFAC Symp. on Advances in Control Education*, Gold Coast, Australia, Dec. 2000.
[9] K. Žáková, M. Sedlák, “Remote control of experiments via Matlab”, *Int. J. of Online Eng.*, Vol. 2, No. 3, 2006.
[10] http://www.learnframe.com
[11] http://www.amira.de
[12] http://www.mathworks.com
[13] http://www.answers.com
[14] http://www.held-mueller.de