Hydraulic Plant Remote Laboratory

V. Žilka, P. Bisták, P. Kurčík
Slovak University of Technology in Bratislava, Slovakia

***Abstract*—In this paper we present a three tank hydraulic system and possibilities how it can be used for the purpose of remote laboratories. The concept of our remote laboratory is based on client – server architecture. We compare two different approaches of remote control. First one uses a Java Server application implemented together with a control algorithm on the server side. In this case the control algorithm is realized in the Matlab/Simulink environment. There is necessary to exchange data between the Matlab and the Java Server application what is done through the COM technology. The second approach uses PC-independent network card which runs its own TCP server and the control algorithm is implemented on client side.**

***Index Terms*—hydraulic plant, remote laboratory, teleexperiment**

I. INTRODUCTION

Remote laboratories represent very useful tools in the engineering education. They enable to monitor and manipulate real systems from a distant location. A remote laboratory user can access the equipment from any point where the Internet connection exists and the equipment is available almost anytime.

The philosophy of a remote laboratory is usually based on client server architecture. This paper describes a remote laboratory for control purposes [2]. The controlled plant is a hydraulic plant. Plants of this type offer few advantages that designate them to be used in remote laboratories, like clear physical visibility of controlled values and time constants in range where sampling periods in order of hundreds of milliseconds are sufficient. Plant is controlled locally within Matlab environment as well as remotely using Java server or PC-independent network card. Graphical user interface on the client side was created in the Easy Java Simulations software [1,4].

II. PLANT DESCRIPTION

A. Plant hardware

The hydraulic plant (Fig. 1) consists of two electrical water pumps, three interconnected vertical tanks and the central tank serving as a water reservoir and five solenoid valves. Each of the pumps can supply water from The reservoir into one of three tanks. Reconfiguration can be done by inserting output hoses from the pumps into desired tanks. By default the first pump supplies first tank and the second pump the second one. Both pumps are continuously controlled by PWM signal generated on the control board. Connections between the tanks can be controlled by means of electrically controlled solenoid valves. Drain from each of the tanks to the central reservoir can be also controlled by three solenoid valves. Each tank has an emergency drain with high throughput to prevent overflow when user control algorithm fails. This setup allows us to use the plant as three, two, or one tank system, whenever the desired experiment needs it. The advantage of solenoid valves is that they can be, as well as electrical pumps, controlled remotely which designates the plant for use in teleexperiments.

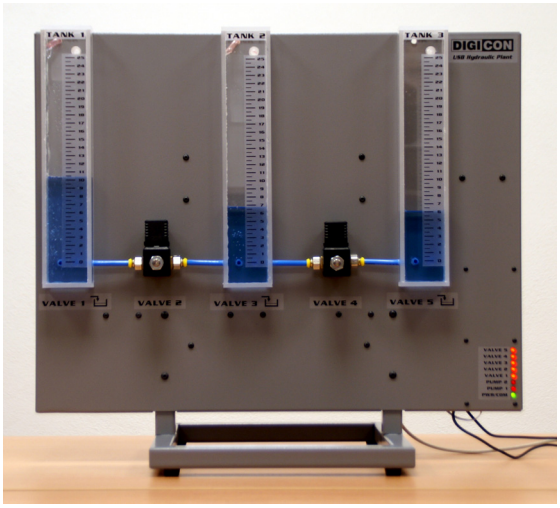


Figure 1. Front view of hydraulic plant

B. Communication interface

The plant has 7 input variables and 7 output variables totally. Plant structure is shown in the Fig. 2. Plant inputs include two integer inputs for controlling power of both pumps and five binary inputs for controlling the solenoid valves. Outputs consist of three values corresponding to levels in the three tanks, next the values represent filtered levels and last value is ambient temperature. The filter time constant is approximately 20s [5].

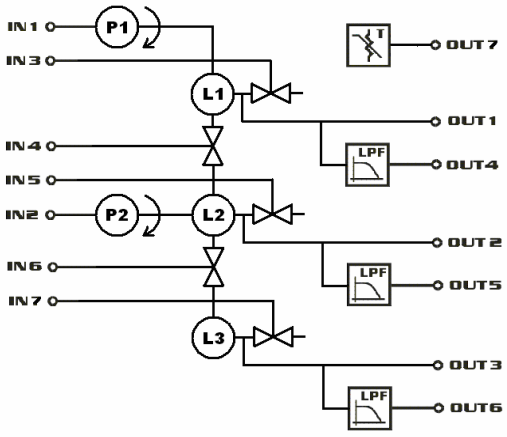


Figure 2. Block diagram of 3-tank hydraulic plant

The plant can be connected to the PC either via analog interface or preferably via USB interface. USB interface offers many advantages, compared to traditionally used analog interfaces and data acquisition cards. To name a few: plant initial setup is much less difficult, there is no need to open the computer case, it allows the use of plant with laptops and it is cheaper, because one does not need additional data acquisition card. We have developed one-block interfaces for Matlab / Simulink and for Scilab / Scicos.

Different approach we used to communicate with the plant does not need rebuilding of the simulation scheme after user change something in it and thus allows faster and more interactive work with the plant.

III. REMOTE CONTROL USING JAVA SERVER WITH MATLAB

A. *Client-server application*

This is the example showing the use of the hydraulic plant in virtual laboratory for remote experiment via Internet. For use in pedagogical process we designed a simple client – server application. Basic scheme for the client – server application is in the Fig. 3.

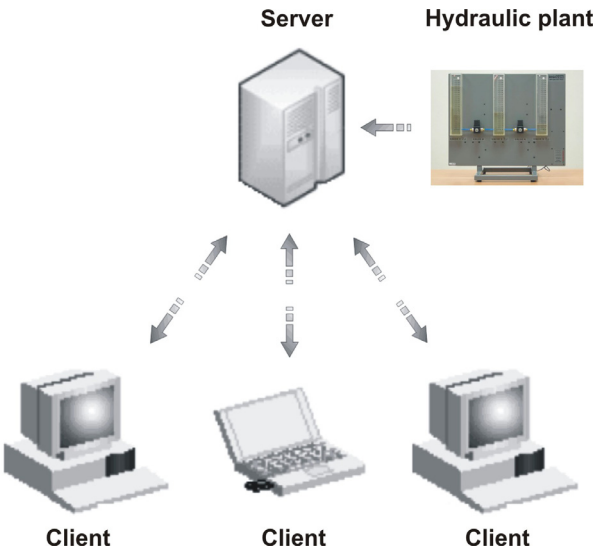


Figure 3. Client-server architecture

B. *Java Server application*

The server application is the central part of the whole system. It runs on the PC connected directly to a certain real plant. 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 cooperate with the Matlab/Simulink application that directly controls the plant. The server application is platform dependent. On the Windows platform there exists several possibilities how different applications can exchange data (DDE - Dynamic Data Exchange, ActiveX, COM - Component object model, …). For this purpose we used the JMatLink library [6] that was written for communication between Java applications and the Matlab application [4]. Our Java server application is based on the COM technology.

Component object model (COM) is technology developed by Microsoft. It defines a language-independent binary standard for component interoperability. It is used to enable interproces 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.

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 starts the Matlab Engine and waits for commands that are transferred immediately to the Matlab. The measured data are transferred back from the server to the client.

C. *Controller running on server side using MATLAB*

For the testing purpose we used the PD controller of dynamic class 2. Design of this controller was described in [8]. Controllers of dynamic class 2 can be very efficient with second order systems – e.g. two level hydraulic system. Block diagram of the controller in Matlab / Simulink is in the Fig. 4.

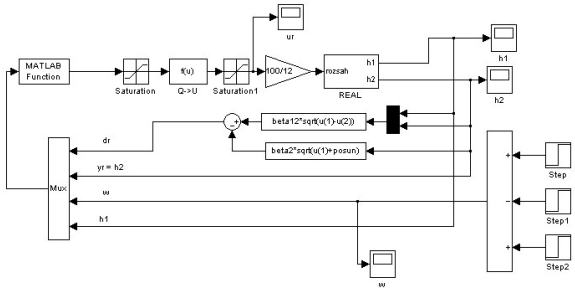


Figure 4. Block diagram for PD2 controller

D. *Easy Java Simulations*

For the creation of the client part we used the Easy Java Simulations (Ejs) development environment. Ejs is an open source tool, which provides a simplified structure for creation of the model of the simulation. User's responsibility is the design of the view, providing the variables and algorithms that describe the model of the simulation.

Ejs can generate the Java source, compile it into Java classes, pack the classes in a Jar file, and produce several HTML pages with the author-provided narrative and the ready-to-run applet for the simulation [1].

However we are not only using the simulations in Ejs, because it can cooperate with Matlab and Simulink when it is installed on the same computer. This possibility is included directly in the distribution of Ejs. Ejs can call Matlab functions, send and read data, variables, and also Simulink models, including setting of simulation parameters.

E. *Client description*

As we mentioned at the beginning of the paper, all parameters have electronic control, so we have full control over settings of the hydraulic plant. In the Fig. 5 we can see the control panel for a control of two levels. Visualization of the process is placed at the top of the control panel. It shows the current value of water levels in two tanks (range 0 – 0.3m), status of valves (we can change status with mouse click on valves before starting real experiment) and also the rotation speed of the crosslets in the pumps that is related with the real pump power. All pipes leading to pumps and from pumps are animated according to presence of water. Below the visualization we can select a controller from two available choices – manual control and PD2 controller for two level system. According to the selected controller the control panel which contains all necessary parameters and variables is displayed.

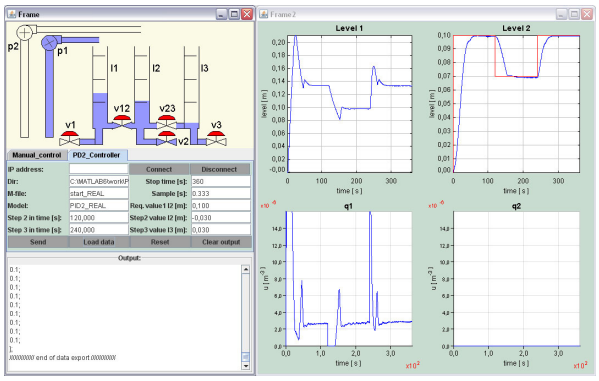


Figure 5. Remote user interface created with Ejs –complex view

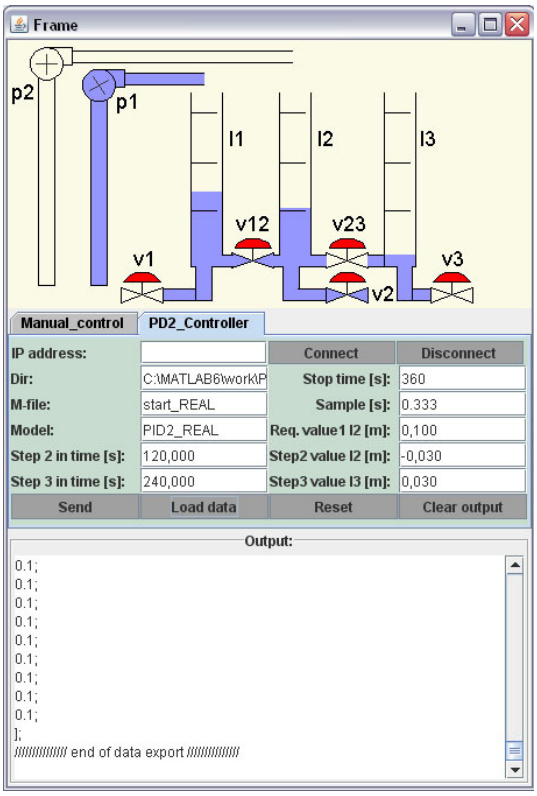


Figure 6. Remote user interface created with Ejs –detailed view

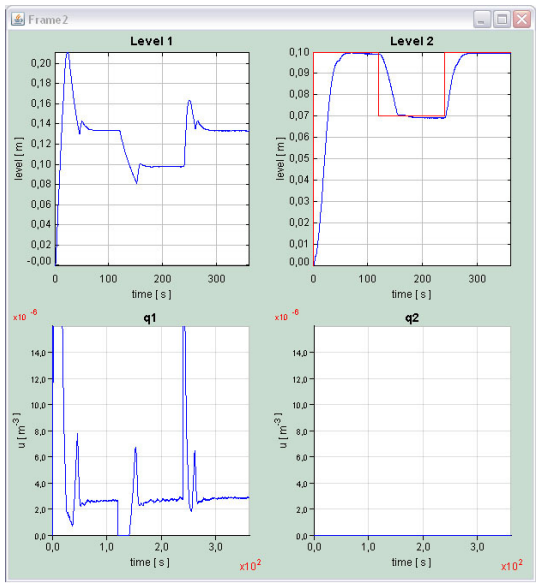


Figure 7. Client's plotting window

The default values of variables are predefined, but we can set new values of course. It is easy to set the IP address of the server, current directory, where the m-file and the model are located, name of the m-file, simulation time, name of the model, step size for simulation and for this regulator also requested value of the second level – there are three steps for better testing of the controller.

The buttons for starting the experiment with set values, load data, reset and the button for clearing the output window are located bellow. The analysis of the data from the experiment is also very important. This version of client application can export measured data from the Matlab workspace on the server to the status window (in Matlab format) and they can be easily selected by mouse, and copied to client's Matlab.

IV. REMOTE CONTROL USING HARDWARE BASED ETHERNET INTERFACE

A. *TCP server running on netLINK ethernet card*

To evaluate an alternative control approach, where the regulator itself is located at remote location we used the netLINK Ethernet interface card. We benefited from fact that the Hydraulic plant is beside of the USB and the analog interface also equipped with auxiliary serial communication port which we used to connect the netLINK card.

In this application the netLINK is configured as Ethernet to serial converter which runs its own TCP server. Due to card hardware limitations, the server is able to maintain only one active client session at a time. For the same reason we have not implemented any encryption methods which will probably slow down the card data throughput as it is only based on 8-bit RISC microcontroller. Advantage of this approach is however that it does not need a PC running server in Matlab to communicate with the plant.

To increase the safety of operation of the hydraulic plant we implemented certain hardware and software measures that prevent hazardous situations like a tank overflow. Having those is very important when the plant

is used remotely without any local supervision. Furthermore there is a timeout protection that shuts down all actuators if there is not any command received at least once per 10 seconds.

B. *Communication interface*

Basic connection scheme is in the Fig. 8. The communication chain contains the plant, the Ethernet to serial converter, a router allowing access from outside of the laboratory and the client application written in Ejs. To establish communication with the server, client needs to send the "open" command. After that the server registers the client's IP address and sends the "connected" string to the client. When the client wants to terminate the session it needs to send the "close" command. To prevent lockups there is a possibility to open new session even if previous was not closed by the "close" command. Basically the last client that sends the "open" command takes control over the plant. For the practical use in remote laboratory where multiple users may want to use the plant some access management at higher level needs to be implemented.

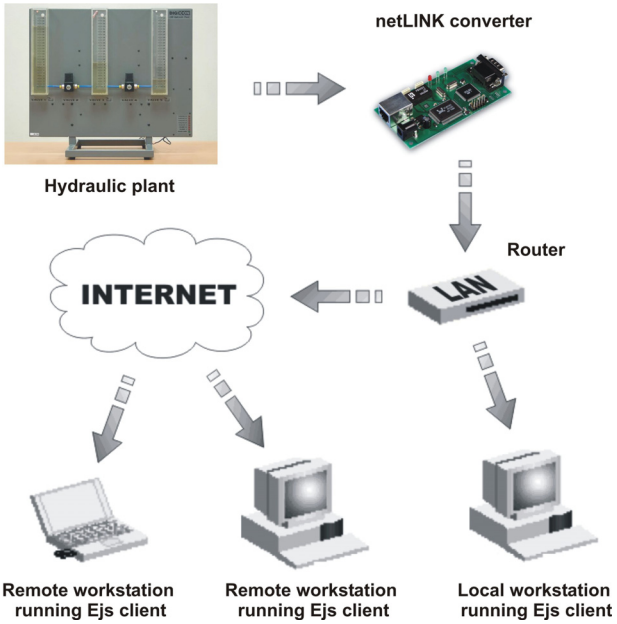


Figure 8. Communication using the netLINK converter

After communication session has started, client is able to control the plant by sending control strings in the same format as is used in communication through USB interface. Strings need to have the ">" character added at the beginning which indicates that this data needs to be relayed to the netLINK's serial port. Similarly, measured data from the plant are relayed to the client at each sample period. There is not any sampling clock in the plant itself or the netLINK card; instead the client is controlling the sampling rate by rate at which it sends the control strings to the plant. When the plant receives the control string it immediately responds with the string that contains the current measured values.

C. *Client description*

The client interface is again built in Ejs software which has been described in previous paragraphs. The visual part is very similar to the previous client interface depicted in the Fig. 5. Several buttons used with Matlab are not needed. The structure of the client is however different as in the approach with Matlab based server part. The main parts of the Ejs client are outlined in the Fig. 9.

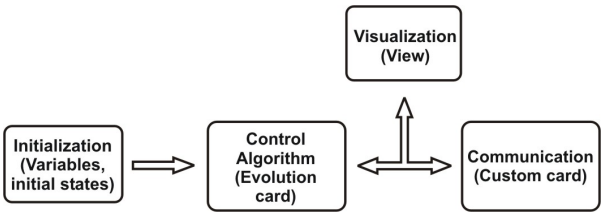


Figure 9. Block diagram of the client side

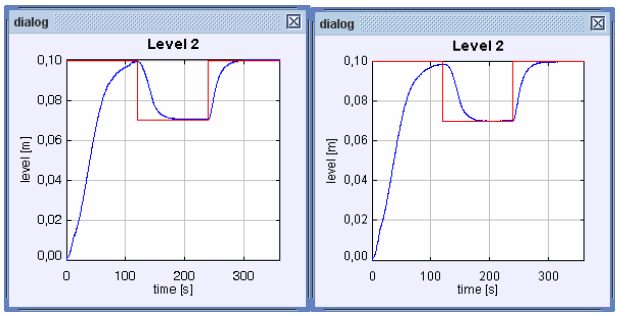
The most important part of the client is the communication thread. Its main purpose is to establish and then maintain the connection to the hydraulic plant. After pressing the Connect button, client tries to establish TCP session with the server. When the connection is successful, client registers its IP address at the server by sending the "open" command. 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. To prevent the plant from safety actuator cut-off, thread needs to send control string to the plant at least once per 10 seconds.

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

The control algorithm is running in the Evolution card. This is the main difference with the previous approach where the control algorithm has been calculated on the server side. The type of the algorithm is the same as in the previous case, i.e. it corresponds to the Matlab/Simulink block diagram in the Fig. 4 but this time it is realized in the Java language and calculated on the Evolution card of the client built in Ejs. The control algorithm is called by the 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. This is also difference with the previous case where data have been downloaded after the control process has finished. For the visualization part we used the same graphical interface as we used in the previous case.

V. COMPARISION OF BOTH APPROACHES

The network control represented by the case using the netLINK card is now the problem very often discussed in the control community. Using the described experimental setups it is possible to compare the advantages and disadvantages of both approaches. The controlled process has been represented by the hydraulic plant that belongs to the slow processes. Therefore it is possible to apply also the network control. From the responses of controlled value (the height of the level in the second tank) one can see, that there is no significant difference within the responses shown in the part III.



a) b)

Figure 10. Control value responses for netLINK solution. Client has been running a) within the same LAN, b) outside the LAN using mobile Internet connetion

It is necessary to mention that the client using mobile Internet connection (Fig. 10, b) has encountered a great number of timeouts caused by the network lag. With the sampling period of 0.333s there was obviously not enough time to receive, calculate and send data in every sampling period. Timeouts occurred almost in every second sampling period. By increasing the sampling period two times, 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. 10, a). Due to relatively slow behavior of the process, we can conclude that the number of timeouts does not significantly 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 use Matlab.

VI. CONCLUSION

The paper has shown that there are several possibilities how to build remote laboratories. We presented two approaches and compared them. First one based on the Java server connected to the real equipment through the Matlab allows running fast control algorithms but it is not as flexible when concerning the changes of controller structure, because there are problems with uploading m-file and model to the server. There is also security hazard, because Matlab can execute some dangerous commands. For this reason it will be very important to have some automatic control for m-files, which will filter any potentially dangerous commands. Beside of that there is need to implement authorization and scheduling system to allow only one user at a specific time. Some of these problems will be solved, when this teleexperiment will be integrated into WebLab [7]. WebLab is an online tool for administration of users, and scheduling. In some cases price of the Matlab which is necessary for realization of this approach can be disadvantage as well. In the future we plan to replace the Matlab by free products like Scicos/Scilab.

The second approach where the control algorithm is calculated on the client side has no limitation to the controller design but the controller must take into account transport delays caused by network connection. Therefore the second approach is not suited for fast dynamical systems. The advantage of the second approach is also highlighted by the fact that the remote user does not need to upload the control algorithm to the server side and thus his/her intellectual property of controller algorithm design is kept.

REFERENCES

[1] R. Pastor, J. Sánchez, S. Dormido, *Web-Based Virtual Lab and Remote Experimenttion Using Easy Java Simulations,* 16th IFAC World Congress, 2005, Prague, Czech Republic

[2] Ch. Smid, *Internet-basiertes Lernen,* Automatisierungstechnik, 51, No. 11, 2003, pp. 485-493.

[3] M. Huba, M. Kamenský, P. Bisták, M. Fikar, *Blended Learning Course Constrained PID Control*, IFAC Conference Advances in Control Education ACE'06, 2006, Madrid.

[4] P. Bisták, *Remote Control of Thermal Plant Using Easy Java Simulations,* Int. Conf. on Interactive Computer Aided Learning ICL'06, 2006, Villach, Austria.

[5] P. Kurčík, V. Žilka, M. Kamenský, *Hydraulic Plant for Education and Practicing,* 8th Int. Conf. Virtual University, 2007, Bratislava, Slovakia.

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

[7] M. Huba, M. Šimunek, *Modular Approach to Teaching PID Control*. IEEE Transactions on Industrial Electronics, VPL.54 No.6 December 2007, pp.3112-3121.

[8] M. Huba: Theory of Automatic Control 3. Constrained PID Control. STU Bratislava 2006 (in Slovak).

AUTHORS

V. Žilka is with the Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, STU FEI Bratislava, Ilkovicova 3, 812 19 Bratislava. (e-mail: vladimir.zilka@stuba.sk).

P. Bisták is with the Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, STU FEI Bratislava, Ilkovicova 3, 812 19 Bratislava. (e-mail: pavol.bistak@stuba.sk).

P. Kurčík is with the Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, STU FEI Bratislava, Ilkovicova 3, 812 19 Bratislava. (e-mail: peter.kurcik@stuba.sk).

The work has been financially supported by the ESF project JPD 3 2005/NP1-047 „PhD students for Modern Industrial Automation in SR", code No. 13120200115, and VEGA Project 1/3089/06 Development and Integration of Methods of the Nonlinear System Theory

This article was modified from a presentation at the REV2008 conference in Düsseldorf, Germany, June 2008. Manuscript received 1st July 2008. Published as submitted by the authors.