Online Control of Thermo-Optical Plant via OpenModelica

http://dx.doi.org/10.3991/ijoe.v8iS3.2195

Ladislav Szolik and Katarína Žáková
Slovak University of Technology, Bratislava

***Abstract*—The paper deals with the web-based implementation of the thermo-optical plant control. The control of the plant is based on the open-source Modelica-based modeling and simulation environment OpenModelica which originally is not designed for web-based applications. The paper shows one possible way how to circumvent this limitation.**

***Index Terms*— computer aided engineering, control design, online services, student experiments.**

I. INTRODUCTION

Experimental work is an inseparable part of engineering education at universities. Experiments help to understand theory, relations between incurred tasks and problems and they give a hand to form, confirm and reject hypotheses that lead to the problem solution. In spite of the fact that some experimental work can be accomplished using various animations and simulators, the best possibility is to offer students the work with real devices. Unfortunately, such experiments cannot be usually available for the whole day. The rooms with experiments are usually opened only during limited time. It means that except of travelling to the university buildings students also need to check the availability of laboratories. Therefore there arises a question how to offer and deliver experiments to all students whenever they need them. One possibility is to use Internet as a medium and to provide experimental work via remote access to the laboratory.

The paper demonstrates one possible way of remote control of the thermo-optical plant presented in the next section.

II. PLANT

The considered thermo-optical laboratory plant uDAQ28/LT (Fig.1) presents a system that is very suitable for all forms of the education process because of its easy manipulation and a good portability. It can be connected to the control computer via a USB interface and no special A/D card is required. It enables to control two physical variables – the temperature inside a plastic cylinder and the intensity of a light source.

The plant [5] has three inputs – the bulb voltage (the heat & light source), the ventilator voltage (system cooling) and the light diode voltage (the second possible light source). There also exist two parameter inputs for adjusting the sampling period and the time constant of the built in derivative filter. The user can use eight measured outputs: the system temperature measured by a PT100 sensor, the light intensity (both measured directly, or with a preliminary filtration by the filter of the 1st order) and its derivative, the ambient temperature, the current and the rotation speed of the ventilator.

Figure 1. The combined thermo-optical laboratory plant

The high number of measured outputs enables to accomplish a variety of experiments. However, students mostly control only the light intensity that can be influenced by the voltage on the light diode or the temperature inside the plastic cylinder that is influenced by the bulb heating and the ventilator cooling. The temperature is usually controlled by the bulb voltage with the ventilator being considered as a disturbance factor. Of course, the control when the temperature is influenced by the bulb heating and the ventilator cooling at the same time together is also possible. In addition to control, students also have to solve tasks that are connected with the plant identification, input-output data manipulation and communication with outer computer environment.

The communication with the computer runs via the string exchange. The data transfer rate is 250kbit/s. The plant enables to use a sampling period 40-50 ms whereas considering the dynamics of the presented system 1 second it should be sufficient for its quasi-continuous control. The whole plant is supplied by 12V/2A DC external adapter.

III. OPENMODELICA ONLINE SUPPORT

OpenModelica [3] (developed on the base of open and free technologies) enables to model and simulate the behavior of the dynamical systems. In this way it is similar to Matlab, LabView or SciLab.

It is to say that OpenModelica is usually used for computations that are accomplished on the computer locally. However, the increased expansion of Internet together

with the growing support of online education raised a question how to exploit capabilities of OpenModelica for these purposes, too. In such a case, one installation of OpenModelica placed on the remote server could serve for several clients whereby the client can be represented by a person or an application.

We started to test OpenModelica for various online interactive examples and simulations. Later we tried to use it for a remote experiment, too. Such approach is also presented in this paper.

IV. REALISATION

For development of the presented web portal we decided to use LAMP technologies (Linux – operating system, Apache – web server, MySQL - database, PHP and Python – script languages). The backend of the application is also supported by the already mentioned OpenModelica environment that runs the whole experiment. Except of that we also used the supplementing program module omniORB that offers communication interface between Python and OpenModelica. The driver for the plant is written in C language.

The client side of application was developed using PHP script language that dynamically generates HTML pages by means of Smarty templates. The database abstraction was done via Dibi library. The graphical layout was built using cascade styles and Javascript language helped us to achieve interaction with the web application. Except of that we used some Javascript libraries that facilitated the programming of some tasks: jQuery [12] (simplifying the work with page elements), Flot [8] (graphical visualization of results), jQuery Week Calendar [9] (graphical user interface for experiment time reservation) and Canvas2image [16] (export of data to .png format).

A. *Experiment running*

The communication with the server is provided via the web browser form where the user enters all necessary parameters that are sent later to the OpenModelica engine installed on the server. The same web page can also be used for the result visualization.

Each user that wants to run experiment has to log in into the web application environment and to start with allocation of a time slot for the experiment. Without this reservation the experiment cannot be executed. It should ensure that in each moment only one user will have the full access to the experimental plant.

Then according to preferences specified by the user the application builds the control structure for the experimental work. Only then the experiment can be executed.

The created application requires accomplishing following tasks in parallel:

- Accepting the data from the plant that are in successive steps saved to variables and globally available in the application. Afterwards they can be used for continuous visualization of experimental results e.g. in the form of graphical dependences.
- Enabling the change of controller parameters during the run of experiments.
- Following the simulation time that was specified by the user or by the application.
- Receiving information messages that are sent by the plant model.

The experiment can be stopped by two ways. It can be done manually by the user who presses the Stop button. The experiment is also concluded after the predefined simulation time elapses. In both cases all running processes are ended and the experiment report is completed by the last measured experimental values. The communication between the server and the client side is closed only after the results are visualized on the client side. Later, the client can send new data that the server has to process again.

B. *Web Application*

The structure of the whole application is sketched in Fig.2. It consists of 3 main parts.

The presentation and application layer is realized using PHP scripting language that is widely used for the development of dynamical web applications.

The simulation layer was built using Python scripting language. The main motivation for the change of the programming language was the fact that PHP (in contrast to Python) doesn't allow building of native multi-threading applications. Since we need to accomplish several tasks in parallel one possible choice how to realize them all at once was the selection of the programming language that supports multi-threading in the native manner. We didn't want to use any alternative solution for PHP (e.g. cURL extension).

Except of that Python programming language has the ability to cooperate with CORBA interface that can be used for communication with OpenModelica environment.

The presentation layer defines the appearance and layout of the client application using Smarty templates. It helps to manage requests and responses for users. The data that are visualized in the presentation layer are dynamically prepared in the application layer.

The application layer processes the requests from users and ensures that the user receives the corresponding feedback information. After the user starts the experiment and data are sent to the server, they have to be transformed to the form that can be understood by OpenModelica. This layer also support data manipulation using MySQL databases and takes care about export of experimental results to various formats according to the specification of the user.

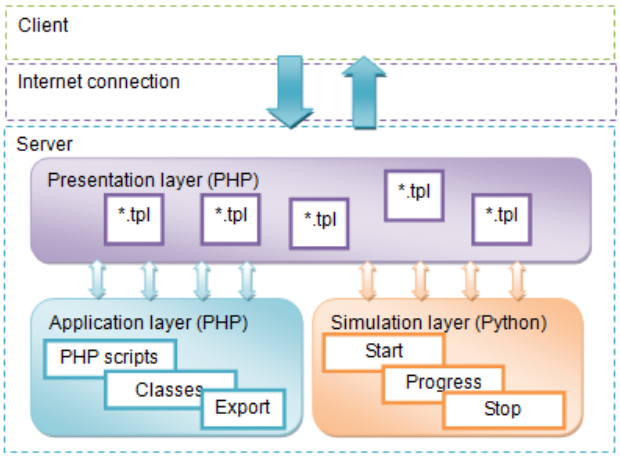

Figure 2. The web application structure

The simulation layer covers the communication between the real plant and OpenModelica environment that is used for the control of the remote experiment. OpenModelica is running in the interactive mode that enables to change parameters during the experiment. Without this feature it would be not possible.

Before running the experiment the simulation layer creates its control structure on the base of requirements from user. It consists from several model components that are summarized in Fig.3.

The structure of all model components, their nesting and connection results from the Modelica language possibilities. However, there are 2 essential parts:

The omReadWrite block covers the communication with the thermo-optical plant. Its task is to write and to read data from the system. The functionality requires importing the driver that ensures approach to the real device. The driver was written in C language using the open source libusb library [2] that enables an unified approach to USB interface based equipments on the user level.

The UdaqCont model block describes the block scheme that is used for the experiment control. It consists from two main blocks:

The Udaq model block allows to specify and set inputs and outputs of the plant. This block is written in Modelica language and defines an interface between the real plant and control algorithm calculations that are realized in the OpenModelica environment.

The Controller model block enables to compute the value of control signal according to the chosen control algorithm. The application also allows running the experiment without a controller. This setting permits to measure step responses of the physical system and to identify its dynamics. In such a case the Controller model block is substituted by the open loop connection.

C. *User Interface*

The online experiment can be realized using Web portal shown in Fig.4. After registration and login it enables user

- to allocate time slot for experiment
- to set suitable parameters
- to select type of experiment (open loop, control structure with PID controller or control structure with own control algorithm)
- to follow results in form of graphical dependencies
- to export numerical results to various graphical or text formats that can be used outside of the presented web portal.

Our aim was to enable users to use not only a set of predefined controllers but also to be able to experiment with own control algorithm that is set up according to personal preferences.

Since the structure of the OpenModelica file describing the controller is quite complicated we decided to facilitate the design as much as possible. The structure requires specifying parameters, inputs, outputs and all other variables of the controller at the beginning of the controller model description. Their number depends on the type of the controller and therefore has to be flexible.

The implemented graphical user interface (Fig.5) allows adding a new parameter or variable by simple click-

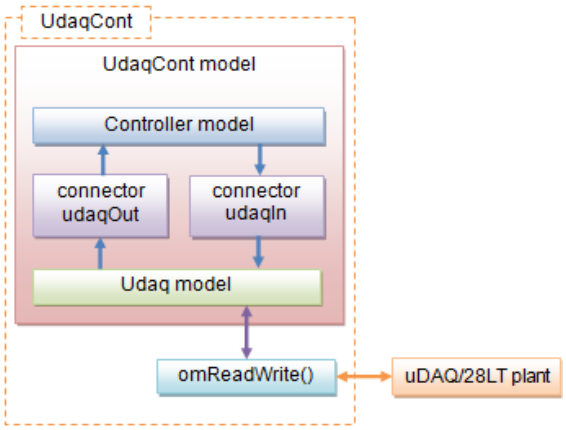

Figure 3. Control structure model components in OpenModelica

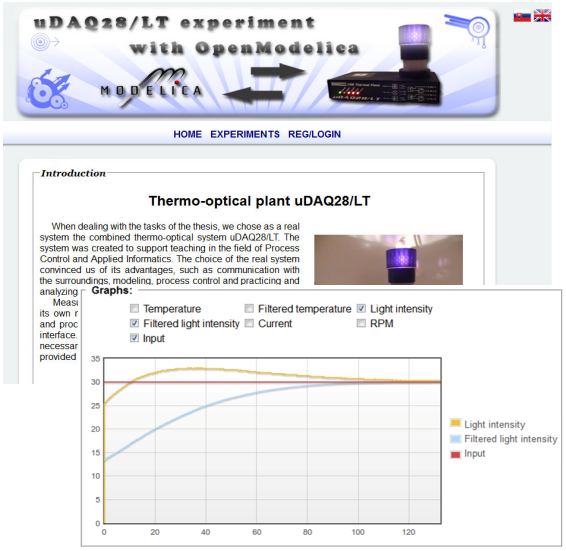

Figure 4. Web portal for online control

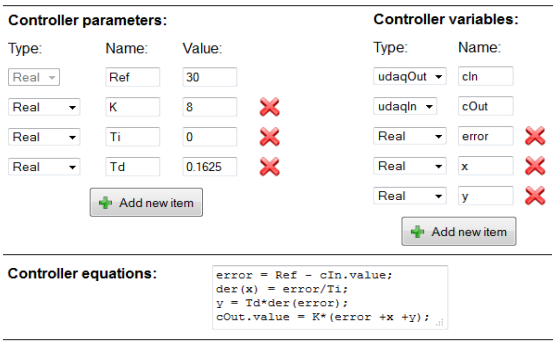

Figure 5. GUI for the definition of own controller

ing on a button whereby the user can define its type, name and in the case of the parameter its default value, too. The keywords for various types of parameters and variables are predefined and therefore the risk of misspelling is minimized. Variables have no associated values since their values are changed in each time step of the experiment.

By specifying parameters and variables the structure of the controller model starts to be created (Fig.6 – blue sections). Finally, the controller algorithm equations have to be determined. We used the classical text area input element (Fig.5) for their definition. The syntax of equa-

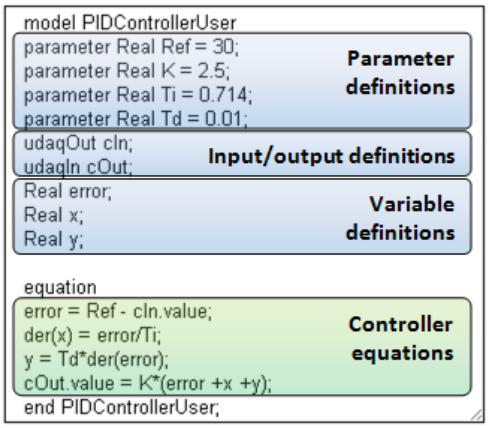

Figure 6. The resulting controller code for OpenModelica

tions has to follow the OpenModelica specifications. Unfortunately the misspelling errors in equations cannot be automated checked so easily. The control algorithm equations (Fig.6 – green section) form the last part of the controller model structure. In Fig.6 it is possible to see the example of PID controller defined by the user. Afterwards the controller can be saved and it is prepared for the use. It means the experiment can start.

Running the experiment the user can set up several parameters. Except of determining the required value of output he or she can set up the sampling period, the run time of experiment and parameters that were defined in the controller. The values from the controller structure are used as default parameter values but they can be changed still before or later during the experiment. The experiment duration can be determined as a concrete value in seconds or by the string "inf". In such a case the experiment is running till the moment when it is stopped by the user or when the reservation period elapses.

During the experiment the user can follow all available variables. They are visualized in the form of graphical dependencies that can be either switched on or switched off (Fig.4). In the case that the user receives outputs that are not expected the experiment can be stopped still before the specified time or as it was already told the parameters of the controller can be changed. The structure of the controller (i.e. the controller algorithm) can be changed only after the running experiment is ended.

All experiment data are saved to MySQL database and can also be viewed later. Except of that the measured values can be exported to several formats (.txt, .xml, .json, .png) what enables processing of experimental results in other software environments that can be more familiar to the user.

V. CONCLUSIONS

The implementation of the presented remote control of experiment was realized in frame of diploma work. Our purpose is to support "learning by doing" method that helps students to understand better the problem they meet. In this case it was necessary to master not only programming techniques, algorithmization, way of communication but also all topics connected with control of the real plant.

ACKNOWLEDGMENT

Authors thank to Zoltán Magyar for his help and discussions.

REFERENCES

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

[2] J. Erdfelt, D. Drake, "*LibUSB Homepage*", http://www.libusb.org/

[3] P. Fritzson, et al., "*OpenModelica Users Guide*", Linköping University, Sweden: Wiley-IEEE Press, 2004.

[4] I. Gustavsson, K. Nilsson, J. Zackrisson, L. Hakansson, J. G. Zubía, G. R. Alves, U. Hernandez, R. J. Costa, T. Lago, I. Claesson, "The VISIR Open Lab Platform 5.0 - an architecture of a federation of remote laboratories", 8th *Intern. Conference on Remote Engineering and Virtual Instrumentation* (REV'11), Brasov, Romania, 2011.

[5] M. Huba, "*Thermo-Optical Laboratory Plant uDAQ28/LT*", http://www.eas.sk/mod/product/show.php?ID=59, 2008.

[6] M. Huba, M. Šimunek, "Modular Approach to Teaching PID Control", *IEEE Transactions on Industrial Electronics*, ISSN 0278-0046, Vol. 54, No. 6, pp. 3112-3120, 2007.

[7] M. Jáno, K. Žáková, "SciLab Based Remote Control of Thermo-Optical Plant", *Int. Journal of Online Engineering* (iJOE), Vol. 7, No. 4, pp. 10-15, 2011.

[8] O. Laursen, "Flot - Attractive Javascript plotting for jQuery", Online, http://code.google.com/p/flot/

[9] R. Monie, "jQuery Week Calendar", Online, https://github.com/robmonie/jquery-week-calendar/wiki, 2010.

[10] Nette Foundation. Dibi is Database Abstraction Library for PHP 5, Online, http://dibiphp.com/cs/.

[11] New Digital Group, Inc. Smarty Template Engine, Online, http://www.smarty.net/.

[12] The jQuery Project. jQuery: The Write Less, Do More, JavaScript Library, Online, http://jquery.com/.

[13] M. T. Restivo, J. Mendes, A.M. Lopes, C.M. Silva, F. Chouzal, A Remote Lab in Engineering Measurement, *IEEE Trans. on Industrial Electronics*, vol. 56, no.12, pp. 4436-4843, 2009.

[14] F. Schauer, M. Ožvoldová, F. Lustig, "Real Remote Physics Experiments across Internet – Inherent Part of Integrated E-Learning" , *Int. Journal of Online Engineering* (iJOE), 4, No 2, 2008.

[15] Chr. Schmid, "Internet - basiertes Lernen", *Automatisierungstechnik*, 51, No. 11, p. 485-493, 2003.

[16] J. Seidelin, "Canvas2Image", Online, http://www.nihilogic.dk/labs/canvas2image/.

[17] I. Zolotová, M. Bakoš, L. Landryová, "Possibilities of communication in information and control systems", *Annals of the university Craiova, Series: Automation, Computers, Electronic and Mechatronic*, Vol.4(31), No.2, pp.163-168, ISSN 1841-062, 2007.

[18] J. G. Zubía, G. R. Alves (eds.), "*Using Remote Labs in Education: Two Little Ducks in Remote Experimentation*", University of Deusto, Bilbao, ISBN: 978-84-9830-335-3, 2011.

[19] K. Žáková, M. Sedlák, "Remote Control of Experiments via Matlab", *Int. Journal of Online Engineering* (iJOE), 2, No. 3, 2006.

AUTHORS

L. Szolik is with is with the Accenture s.r.o., Plynárenská 7/C, 824 86 Bratislava, Slovakia, (e-mail:szolik.ladislav@gmail.com).

K. Žáková is with the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia (e-mail: katarina.zakova@stuba.sk).

This work has been supported by the Slovak Grant Agency, Grant VEGA No. 1/0656/09. It is an extended and modified version of a paper presented at the International Conference on Remote Engineering & Virtual Instrumentation (REV2012), held at University of Deusto, Bilbao, Spain, July 4-6, 2012. Manuscript received 18 July 2012. Published as resubmitted by the authors 14 November 2012.