Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 13492–13497

Online Laboratory Manager for Remote Experiments in Control

Matej Rábek and Katarína Žáková

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

Abstract: The aim of the paper is to contribute to the area of online experiment management. Although such systems already exist, the proposed solution tries to enhance their features by providing users with an extended user interface. The paper suggests the most efficient architecture and explains the reasons for using each component. After the examination of the system interfaces and their role in the overall data flow, the paper focuses on the achieved functionality and its benefits. The attention is devoted to the modification of control algorithms by the user, running experiments in prescheduled time and also in the batch mode. The experiments are supported by various simulation environments – e.g. Matlab, SciLab, OpenModelica. However, the use of classical program language, as the C language, is also supported. What differentiates this system from others like it, is the added value of social networking enabled by several communication tools.

Keywords: Remote control, control education, online experiments, software tools, open technologies.

1. INTRODUCTION

Whether proving a hypothesis or making uncertain first steps on a journey to master a subject, everyone can appreciate the benefits of a well-equipped and easy to access laboratory. While some may take it for granted, others might find it rather difficult to visit such place. Living too far away or an inability to walk are just some of the reasons why one can abandon their dream to pursue scientific research.

One way to solve this problem is creating an online laboratory which provides its users with real devices capable of running various experiments. It also has to return and display the results or any other output that the scientist can find beneficial. The ability to communicate, collaborate and share ideas among multiple participants is also integral to a working laboratory.

The number of internet based experiments is growing steadily, with examples from numerous areas of technical education, such as Physics, Electronics, Signal processing and Control engineering. Some examples are summarised in Table 1. Very detailed overview of publications about online experimentation is done in Heradio et al. (2016). The introduced online experiments are mainly from Control Engineering, Chemistry, Electrical and Electronic

Table 1. Examples of provided remote experiments

Project Title	Provider	Topic	Web page
UNILabs	National Distance Education University (UNED), Madrid, Spain	Physics, mechatronics, optics, robotics, control	http://unilabs.dia.uned.es/
Online experimentation	Faculdade de Engenharia da Universidade do Porto, Portugal	Electronics, metrology, mechatronics, environment	https://remotelab.fe.up.pt
Remote Laboratories @ UP	Universidade do Porto, Portugal	Physics, mechanics, metrology	http://remotelabsup.fe.up.pt/experiments.htm
WebLab Deusto	University of Deusto, Bilbao, Spain	Electronics, physics	http://weblab.deusto.es/website/
OpenLabs	Blekinge Institute of Technology, Sweeden	Antenna theory, electronics, security, vibration analysis	http://openlabs.bth.se/
Internet shared instrumentation laboratory	University of Genoa, Italy	Electronics	http://isilab-esng.dibe.unige.it:9999/ISILabWeb/ISILab.aspx
NetLab	University of South Australia	Electronics	http://netlab.unisa.edu.au/index.xhtml
Relle	Federal University of Santa Catarina, Brazil	Electronics, optics, physics	http://relle.ufsc.br/labs/
Grid of Online Laboratory Devices (GOLDi)	Ilmenau University of Technology, Germany	Robotics, mechatronics	http://www.goldi-labs.net/
REX controls	REX controls, Czech Republic	Control	https://www.rexcontrols.cz/virtualni-laboratore
iLabs	Massachusetts Institute of Technology (MIT), USA	Microelectronics, chemical engineering, signal processing	https://icampus.mit.edu/projects/ilabs/
Remotelab	University of Trnava, Slovakia	Electronics, physics	http://remotelabN.truni.sk/ where N = 1, 2, …11

10.1016/j.ifacol.2017.08.2332

measurement area. Authors introduce 189 references and they reference not only experiments, laboratories and environments, but also the used methodology. So, it is clear, that the interest in this area is notable.

2. REQUIREMENTS

Experiments, that are available over Internet, are very often built from scratch, as standalone applications. However, there already exist tendencies to build complex frameworks to support faster, easier and safer development of remote laboratories. WebLab-Deusto (Garcia-Zubia et al., 2008; Orduña et al. 2011) and iLabs Shared Architecture (Harward et al., 2008) represent just two examples of such initiatives.

The first one is the remote laboratory management system developed at the University of Deusto. It is offered as open source and free software on GitHub under BSD 2-clause license.

The second represents a web service infrastructure, that was developed to provide a unifying software framework, which can support access to a wide variety of online laboratories. It is available on SourceForge under MIT iLab Software License.

This paper tries to introduce another solution. There can arise questions “Why?”, “Is it still necessary? or “What new do you bring?”. Our aim was not only to share online experiments but also:

- to enable feedback control of experiments by setting own control algorithm,
- to run experiments not only in scheduled time slot but also in batch mode,
- to let the users choose the background simulation environment according to their own preferences,
- to maintain modularity and scalability of experiments,
- to enable various forms of communication among users.

3. HARDWARE AND SOFTWARE

The aim of this paper is to propose a general framework for remote controlled experiments. The experiments will be oriented mainly to the area of Automatic control. The structure of the proposed ecosystem should consider various types of devices and software solutions. The tested alternatives are introduced in this section.

3.1. Connected Devices

The thermo-optical system TOS1A (Huba et al., 2014) shown in Fig.1 enables to control the light intensity and the temperature inside of the device body. These variables can be influenced by 3 inputs: performance of a light-emitting diode, a light bulb and a fan. The user can measure several outputs such as internal and external temperature, light intensity, rotation speed of the fan. The system enables to set several control tasks, for example keeping constant internal temperature while either the fan or the light bulb output fluctuates. The plant can be used during the basic control theory courses since it enables testing of various control algorithms (PI, PID, pole assignment design, etc.).

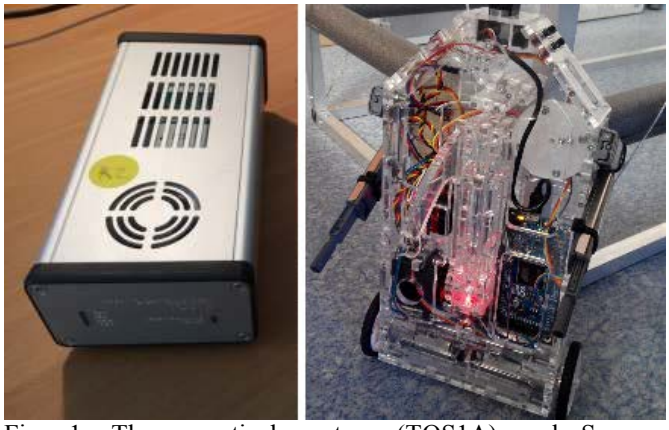

Fig. 1. Thermo-optical system (TOS1A) and Segway laboratory model.

The Segway device (Kňažek, 2016) illustrated in Fig. 1 belongs to systems with faster unstable dynamics. Therefore, it is more difficult to control. It can be considered an inverse pendulum on a two-wheel undercart that is driven by two DC motors. The centre of gravity can be shifted in vertical direction according to the user’s specification by means of another servo motor. The Segway model has built-in gyroscope and accelerometer. The behaviour of the vehicle is controlled by an Arduino Due microcomputer. The plant can be used in courses dedicated to control of nonlinear systems.

The last device is not as typical for the control area. The introduced 3D LED cube is used mainly for teaching the basics of programming, where students can learn to use various loop structures and conditional commands (Žáková, 2016). Visualisation of commands can be more attractive for students. The plant is made of 512 light emitting diodes that are controlled via an Arduino Uno. The cube is connected to a Raspberry Pi computer that serves as a web server enabling remote control of the experiment.

3.2. Simulation Environments

Development of online experiments can be done by two approaches:

- with support of supplementary software packages such as Matlab, Maple, LabView, OpenModelica, Scilab, etc.
- without support of additional software.

In the first case, we can use the computing capacity of the chosen software package. The advantage is that these solutions are usually stable and can therefore focus mainly on Internet communication and control algorithm design itself.

The second approach enables direct access to the controlled systems and it removes an intermediate level between the application on the client side and server-side application. It is reflected on the speed of access to the controlled system.

Since we wanted to provide users with comfortable way of control algorithm adaptation, we selected the first alternative. We offer users several simulation environments to enable them to choose the most appropriate tool for their use. Not everybody is capable of using each of them because syntax and settings can differ for every solution.

Our attention was dedicated to environments that enable modelling and simulation of dynamical systems, regardless of whether it is a commercial solution or not. The proposed framework was tested with

- Matlab – well-known commercial product from The MathWorks, Inc., its simulation tool is called Simulink.
- SciLab – open source software for numerical computation released under the CeCILL license and in present time kept by Scilab Enterprises S.A.S, its simulation tool is called Xcos.
- OpenModelica – open-source Modelica-based modeling and simulation environment kept by non-profit, non-governmental organization Open Source Modelica Consortium (OSMC), its enhanced graphical user interface for simulation is called OMEdit.

4. PROPOSED ARCHITECTURE

The proposed solution should cover the architecture of the whole eco-system of online laboratory that includes not only one but several experiments. Making sure that new experiment could be incorporated seamlessly and the whole process is easily replicated is one of the biggest requirements. In addition, if an experiment stops working, it should not affect the overall functionality. There should also exist an option to have more than one device running the same type of experiment in parallel, in case one malfunctions or it can no longer satisfy all the requests.

For all the above-mentioned reasons a star topology, with the web server acting as the central node, was considered to be the best solution. This design pattern (Fig. 2) simplifies the process of adding new devices or removing an old one (Barranco et al., 2006).

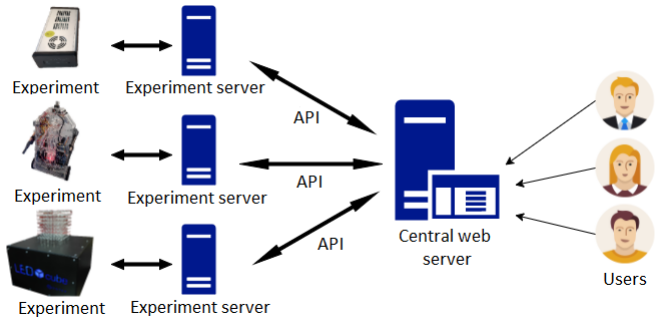

Fig. 2. Schematic representation of online laboratory interfaces and communication channels.

Each device has its own dedicated server that runs the experiment when requested, collects measured data and posts them to the central web server. The obtained measurements are then visualised in form of charts and graphs and presented to the user.

4.1. Server Installation

Both the central web server and the experiment servers are running Linux OS, specifically Ubuntu 16.04 LTS (long-term support). When it comes to the application software itself, a Model View Controller (MVC) architecture is a standard design pattern used in web application development. It is supported by an ever-increasing variety of PHP frameworks, one of which is Laravel – a framework used to develop the application.

4.2. Communication Interfaces

A system which tries to cover functionality of the real laboratory must be comprised of many discrete work stations using various software utilities that are best suited to solve the problem at hand. This creates a need to design and build interfaces which facilitates the communication.

The transfer channel needs to be fast, reliable and working in both ways – to issue commands from user as well as to gather data from running experiments.

The communication in the proposed web application is based on several interfaces that must efficiently transfer data between various nodes.

Faced with the problem of real-time communication, one must consider using WebSockets over HTTP request based messaging. WebSocket is a protocol that provides a bidirectional connection between a web server and a client. Combined with a Node.js server, WebSockets allow to achieve huge performance and architectural advantages (Zhangling and Mao, 2012).

The use of WebSockets is made possible thanks to the Socket.io JavaScript library, which is comprised of two parts: a client-side library that runs in the browser and a server-side library for Node.js.

The communication between Node.js and PHP servers is handled by Redis. Redis is an open source (BSD licensed), in-memory data structure store, which among other things, allows to establish channels to facilitate communication and send messages using publish-subscribe model (Burtica et al., 2012).

The online laboratory also provides tools for sharing information among users – an online chat, peer-to-peer video-chat and a public message board. In case of the online chat, the data transfer is once again handled by a Node.js and Redis servers. The public forum is based on a HTTP REST API and the audio-visual communication uses the WebRTC protocol (Rábek and Žáková, 2017).

WebRTC stands for Web Real-Time Communications. As it is entirely peer-to-peer, it achieves the highest performance and lowest latency possible (Dutton, 2012). One of the characteristics of WebRTC is that it does not include a signalling protocol. On one hand this gives the developer

freedom and flexibility to handle diverse use cases, but on the other hand it means a custom signalling protocol must be implemented, in order to make any use of WebRTC.

4.3. Data Flow

One of the biggest challenges faced while building the system was to implement a way to seamlessly transfer data between experiment servers, central web server and user's own computer. The flow of data through the system that takes place after initiating the experiment is illustrated in Fig. 3.

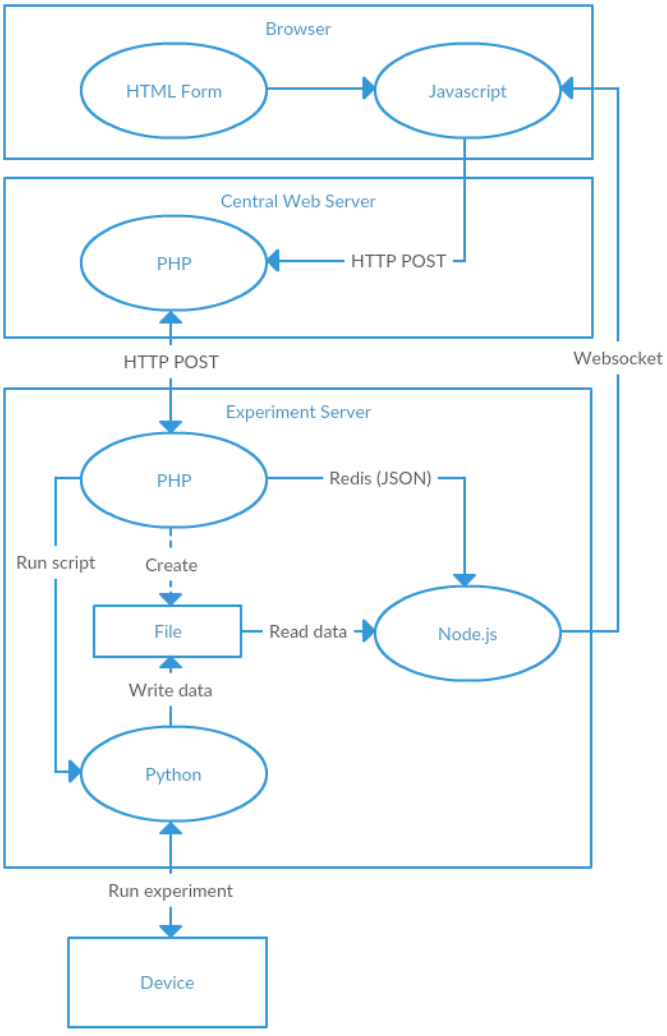

Fig. 3. Data propagation within the system

First, the user inputs all the data necessary to successfully run the experiment into an HTML form. Once submitted the information is sent to the central web server over HTTP using AJAX. Here the data are validated and if everything checks out, the user is notified that the experiment is commencing and the input parameters are sent to the experiment server again using a HTTP POST request method. This server-to-server communication uses JSON format. The reason why the data validation takes place on the central web server rather than on an experiment server is to confirm that only valid data are transferred thus limiting the amount of unnecessary traffic.

Once the command to start the experiment is processed, a log file on the experiment server is generated, where all the collected data will be stored.

The communication with the physical device running the experiment is executed via a Python script located on the experiment server. Since the server can operate different type of devices it was needed to ensure the variability of the processed operations. To maintain modularity of the application 4 different commands can be issued by the experiment server – `init`, `start`, `change` and `stop`. They control the experiment runtime.

Each of commands is operated by separate Python script. For example, the script that starts the experiment needs 3 arguments to run. First one is a number identifying a port used to communicate with the device. Second one is a name of the log file, where the experiment results are written and finally the third one is an array of experiment input settings defined by the user.

However, not all commands have to be implemented for each experiment – it depends on its nature. On the other side, the developer has also the option to chain multiple commands together into a sequence.

The use of these commands is specified in one of the configuration files. In Fig.4 it is possible to see what simulation environments were used for considered systems and what commands were implemented.

```
return [
  "tos1a" => [
    "openloop"  =>  ["start"],
    "matlab"  =>  ["start"],
    "scilab"  =>  ["start"]
  ],
  "segway"  => [
    "openmodelica"  =>  ["change","init","start"]
  ],
  "led_cube"  => [
    "c" =>  ["change"],
    "javascript"  =>  ["change"]
  ]
];
```

Fig. 4. Configuration file with a JSON object defining which software and commands is supported by each device.

The filename, along with an identification number of the user who started the experiment, is also published by the experiment server using Redis. A Node.js script, subscribed to a Redis channel with a predetermined name, awaits a JSON encoded PHP array indicating that an experiment has been started. Along with this information, it also receives a name of the log file with all output data collected so far. This causes a WebSocket channel to be established using the user's identification to ensure the data is transferred over to the correct recipient. The contents of the file are periodically sent in 100 millisecond intervals over the WebSocket while the experiment is executed. The JavaScript running on the user's computer draws a chart displaying the results.

As soon as the experiment finishes, the PHP application on experiment server publishes a corresponding message using

Redis channel. The Node.js script then ends the data transmission since no new data is being added to the log file.

Up until that moment, the user can see the results of the experiment, but the collected data are only accessible to him due to WebSocket connecting the Node.js server straight to client's browser (Fig.3). Once the experiment is over a separate HTTP POST message is sent from the experiment server to the central web server and a complete report is created and uploaded to the database.

4.4. Communication and Messaging

The real-time chat implemented on the website also uses Redis, Node.js and WebSockets to send messages among users. This is quite similar to the way the experiment server sends experiment results to the client's browser. Each chat room has its own identification number which also serves to identify the WebSocket channel. All clients subscribed to the channel will receive a message published to it. Using WebSockets (Socket.io) along with Node.js to create an instant messaging application works perfectly thanks to event-loop, a technology that allows multiple requests to be handled at the same time without creating new threads (Tilkov and Vinoski, 2010). Another advantage to a pure PHP solution would be the ability to use the server's memory, rather than the database, to store the names of users connected to a chatroom. Redis is used to send data from Node.js to PHP, where it is inserted to the database. It is essential to store the conversations in a database, so that when a new user enters a chatroom, the view rendered by the PHP server shows all the previously exchanged messages.

4.5. Supervisor

The system relies heavily on every one of its components to work properly. To ensure this, all software utilities such as Redis, Node.js and Laravel queue service are monitored by a Supervisor. Supervisor is a system providing control over processes. If a monitored process terminates or malfunctions, the supervisor will start it again. This is very useful with queues which stop working after a memory limit is exceeded. When a queue is no longer running, all the resources allocated to it are released and the supervisor can restart it.

4.6. Configuration

Configuration files are important for the system's modularity. All configuration files contain an object or an array in JSON format that can be easily modified if any changes should be made to the system. This means that it is possible to include a new command (Fig. 4), migrate the central web server to a different IP address or change the connected database.

5. USER INTERFACE

The system offers a wide variety of functions with the aim of enriching the user experience, yet at the same time they should not cause confusion or overwhelm anyone. It is important not to lose track of the project's main purpose, which is controller design and testing.

After a successful login and experiment reservation, the user is presented with a control panel, from which experiments can be executed (Fig. 5). Experiments can be also executed in batch mode, but this does not generate an immediate feedback as the experiment does not start until a suitable device is available. This confirms that no device is accessed by two different users at once and no data mix-up occurs.

Fig. 5. Control panel showing a visual representation of data measured during the runtime of an experiment in open loop

Depending on the nature of the experiment, it can run in open or closed loop. Running the experiment in an open loop enables the user to identify the system's dynamics. On the base of the measured step responses, the mathematical model can be determined. It can later serve for controller setting.

On the other hand, it is not sufficient to merely read the measured data in open loop. Students need to interact with the experiment and influence the control process. It can be done in closed loop by changing the control or simulation parameters or even by modifying the whole control algorithm.

Modification of parameters can be achieved by using a standard web form (Janík and Žáková, 2011). The advantage is that if all inputs are restricted to numerical values from a predefined interval such solution doesn't bring any potential danger for developed application and the server installation.

Internet based control applications that want to offer students better understanding of problem, usually enable to modify not only parameters, but also the control algorithm. The easiest way is to offer the users several algorithms or control structures so the student can choose one that is the most suitable. Another possibility is to enable the user to implement own solutions.

To prevent consistency of the simulation scheme it is suitable to separate it into two parts (Fig. 6) – the first one that cannot be changed by the user (displayed in green colour) and the

second one represented by the Controller block, which can be modified according to the user's preferences.

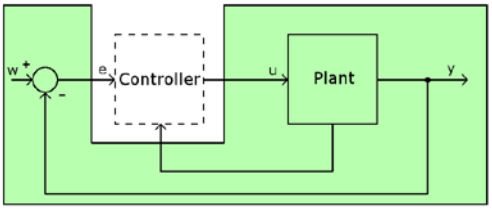

Fig. 6. Schematic block scheme of the experiment for simulation

The controller is set via the standard web form where users can edit the control algorithm code. The controller structure can also be uploaded in a separate text file. The text file can contain either a mathematical expression describing the controller behaviour or ASCII transcription of blocks describing the controller structure. In both cases, it is necessary to respect the format of the background software environment. Users can create their own private controllers and test them on available devices or they can run experiments using an already developed public controller.

The system was developed with an emphasis on communication among users. They can discuss various topics on message boards, join public chatrooms or create private ones and even video chat with each other. Trying to replicate some aspects of social networking, a notification system was implemented that informs the user if someone commented on their post or invited them to participate in an online chat conversation. There is also an option to receive these notifications by email if the user decides to do so. Detailed overview of these social networking aspects is presented in Rábek and Žáková (2017).

6. CONCLUSIONS

Access to a laboratory is an invitation to participate in the scientific research. The system described in this paper represents a contribution to building a unified platform, where various groups and teams can provide access to their devices to one another and to other people. While other web applications may seem similar, this one has its strength in a possibility of varying the control structure, socializing activities among users and finally, in modular design which is key in maintaining scalability. A way to effortlessly add a new simulation environment is also not a common feature among similar systems.

7. ACKNOWLEDGMENT

The work presented in this paper has been supported by the Slovak Grant Agency, Grant KEGA No. 025STU-4/2017 and VEGA No.1/0937/14.

The authors thank M. Gallovič, M. Kňažek, A. Dano and P. Novotný for their help with system implementation.

REFERENCES

Barranco, M., Proenza, J., Rodriguez-Navas, G. and Almeida, L. (2006). An active star topology for improving fault confinement in CAN networks. *IEEE transactions on industrial informatics*, vol. 2, no. 2, pp. 78-85.

Burtica, R., Mocanu, E. M., Andreica M. I., and Ţăpuş, N. (2012). Practical application and evaluation of no-SQL databases in Cloud Computing. *IEEE International Systems Conference (SysCon),* Vancouver, BC, pp. 1-6.

Dutton, S. (2012). Getting started with WebRTC. *HTML5 Rocks*, *23*.

Gallovič, M. (2016). Access management to online experiments. Diploma thesis. Slovak Technical University in Bratislava, Slovakia. (In Slovak)

Garcia-Zubia, J., Lopez-de-Ipina, D., Orduna, P., Hernandez, U., Angulo, I. and Irurzun, J. (2008). Acceptance, usability and usefulness of WebLab-Deusto from students point of view. *Third International Conference on Digital Information Management* (ICDIM), London, pp. 899-904.

Harward, V., del Alamo, J., Lerman, S., Bailey, P., Carpenter, J. and DeLong, K. (2008). The ilab shared architecture: a web services infrastructure to build communities of internet accessible laboratories. Proceedings of the IEEE, 96, 931-950.

Heradio, R., de la Torre, L., Galan, D., Cabrerizo, F. J., Herrera-Viedma, E. and Dormido, S. (2016). Virtual and remote labs in education: A bibliometric analysis. *Computers & Education*, Vol. 98, July 2016, pp. 14-38.

Huba, T., Huba, M., Ťapák, P. (2014). Thermo-optical plant TOS1A. *Technical User Guide*, Bratislava: Slovenska e-akademia, n. o. , 49 p., v0.02.(in Slovak).

Janík, Z., and Žáková, K. (2011). Online design of Matlab/Simulink block schemes. *International Journal of Emerging Technologies in Learning (iJET), Vol.* 6., No.S1, pp. 11-13.

Kňažek, M. (2016). *Online control design of Segway model.* Diploma thesis, Slovak Technical University in Bratislava, Slovakia. (In Slovak)

Orduña, P., Irurzun, J., Rodriguez-Gil, L., García-Zubia, J., Gazzola, F. and López-de-Ipiña, D. (2011). Adding New Features to New and Existing Remote Experiments through their Integration in WebLab-Deusto. *International Journal of Online Engineering (iJOE)*, Vol.7, No.S2, pp. 33-39.

Rábek, M. and Žáková, K. (2017). Communication in an online laboratory. *Int. conf. Distance Learning, Simulation and Communication,* Brno, Czech Republic.

Tilkov, S. and Vinoski, S. (2010). Node. js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing*, *14*(6), 80-83.

Zhangling, Y., and Mao, D. (2012). A real-time group communication architecture based on websocket. *International Journal of Computer and Communication Engineering* Vol. 1, No. 4, pp. 408-411.

Žáková, K. (2016). The Use of 3D LED Cube for Basic Programming Teaching. *11th IFAC Symposium on Advances in Control Education*, Bratislava, Slovakia.