A Grid Concept for Reliable, Flexible and Robust Remote Engineering Laboratories

http://dx.doi.org/10.3991/ijoe.v8iS3.2263

K. Henke, St. Ostendorff, H.-D. Wuttke and St. Vogel
Ilmenau University of Technology, Germany

***Abstract*—Within this paper, we will describe a grid concept to realize a universal remote lab infrastructure as well as different operation modes based on this concept. This new infrastructure consists of three parts: an internal serial remote lab bus, a bus protection unit to interface the remote lab bus and to protect it from misuse and damage as well as a protection unit, which protects the physical systems (the electro-mechanical models in our remote lab) against deliberate damage or accidentally wrong control commands and which offers different access and control mechanisms. The interconnection between the Web-control units and the selected physical system during a remote lab work session (experiment) as well as the user management is done by the lab server, which also handles the webcams. The implemented remote lab infrastructure is based on the iLab architecture of the MIT, which allows to interconnect remote labs and to exchange remote lab experiments among different universities worldwide.**

***Index Terms*—control engineering education, laboratories, Web-based education, virtual and remote labs, Web-based design tools, distance learning.**

I. INTRODUCTION

Our Integrated Communication Systems Group at the Ilmenau University of Technology has many years of experience in integrated hard- and software systems and over 10 years of experience in dealing with Internet-supported teaching in the field of digital system design ([1] and [2]). We have developed a new teaching concept, called "Living Pictures" [3] that we use in several phases of the learning process. Living Pictures are highly interactive Java applets that can be used for demonstrations as well as for experimental purposes, and also serve as tools in certain steps of the design process of digital systems. To complete the learning outcomes by own experiences, the students have to pass a practical examination in a lab. A task during this examination is to design an algorithm for a digital control system that controls one of various physical systems, for instance an electro-mechanical model of an elevator, a production cell or a positioning table, etc.

For all students, hands-on experiences are important to deepen their knowledge about topics they learned during lectures. At our university we offer a remote laboratory, which gives the students the possibility to work on real world systems without the need to stand in line at a lab or the need to take care of opening hours.

With our remote lab we want to offer the students a working environment that is as close as possible to a real world laboratory. Under real laboratory conditions, disturbances can appear and lead to failures of the control algorithm that cannot be detected under virtual lab conditions. Therefore, it is important to include such real disruptive factors for a closer relation to practical conditions. Furthermore the remote lab should offer the students stimulus with regards to the design of safety critical control systems. By conception of our lab (described in the next section) the virtual worlds were combined with a real laboratory surrounding.

In principle, we would like to concentrate on giving students the chance to check their prepared control algorithms

- via Web-based simulation,
- via Web-based remote control of the existing physical system (that means before the tutorial course),
- in real environmental conditions (which can be interactively influenced by themselves)

and to correct or modify the received results.

Currently, there are no existing common standards for the architecture of remote labs. That is the reason why universities develop their own remote lab solutions – suitable for their specific requirements, for example [4], [5], [6] and [7]. This handicaps a networking of different remote labs among each other and also a usage by different institutions. Missing uniform interfaces prevent the programming of universal plug-ins and other software modules. Most of the existing laboratories can be divided in

- batched remote labs and
- interactive remote labs.

A. *Batched Remote Labs*

Batched remote labs do not necessary possess a real-time access and are therefore used only if something has to be computed or processed in a "start-stop" manner with no interaction or communication needs between start and stop. In this case no time-slot regulated login procedure is necessary. The individual tasks are stored in a stack, successively processed or computed and the result will be sent back [8]. Nearly any Web-protocol can be used for transmission, since only an exchange of text files takes place. This is not the focus of this article and therefore not discussed. For an example and more details see [9].

B. *Interactive Remote Labs*

Interactive labs will be used, when

- an experiment requires real-time processing or

- the user wants to observe the whole physical system (the electro-mechanical models) during the experiment or
- some parameters (e.g., input variables to a control system) need to be changed interactively.

An example for an interactive remote lab, supporting all the design steps for complex control tasks, is the *REAL*[1] system, developed at the Department of Integrated Communication Systems at the Ilmenau University of Technology [10] – see Figure 1. As mentioned in many papers (e.g., [11], [12] and [13]), interactive remote labs can open opportunities which allow an experimental approach for a wider audience and also an independence of opening times of the laboratory room.

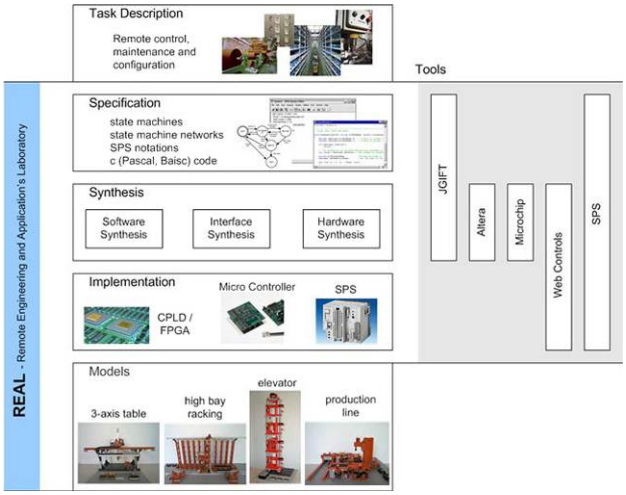

Figure 1. Overview of the *REAL* system

The *REAL* system is a hybrid[2] interactive remote lab where students can design a control algorithm to control some electro-mechanical models. It offers various features like visualization and animation, which allows to observe and to test all the properties of the design. In connection with formal design techniques, simulation and prototyping are used to establish a foundation for the development of a reliable system design. To check the functionality of the whole design, some special simulation and validation features are included as integral part of the *REAL* system. This offers various possibilities for the execution of simulations, such as

usage of simulation models of the physical system for visual prototyping,

step by step and parallel execution of these prototypes,

visualization of the simulation process with the tools also used for specification,

features for test pattern generation and

code generation for hardware and software synthesis.

REAL offers a Web-based environment supporting the above mentioned features to generate and execute a design by using simulation models. An example will be given in Section III.A.

As discussed in the beginning, it is furthermore necessary for students to test their design under real environmental conditions in the remote lab. For this purpose, we have developed a Web-interface to implement the control algorithm on the control unit described in the following section. By using this Web-interface (as shown in Figure 6), the student is able to:

download the synthesized hardware and/or software control algorithms – generated and already validated by the *REAL* environment – to test his control algorithm on the physical system in the lab room,

handle the experiment (e.g., start, stop, reset),

change environmental variables if necessary and

watch the experiment by manipulating environmental variables inside an I/O monitor or by observing the control of the physical system directly via a webcam.

At any time the students have the chance to adjust their algorithms in case of faults. Therefore, they are able to achieve a fault free solution (a validated control algorithm) step by step. For more details see [2], [14] and [15].

II. EXTENSIONS OF THE *REAL* SYSTEM

Our remote lab is used for teaching practical lessons as well as giving hands-on experiences for the development of embedded electronics. This is not done using on-site lessons, but remote via the Internet. This has the advantage that courses can be offered internationally world-wide and gives students from different countries, speaking different languages, the same access and equal possibilities in the lab. Additionally, even for local students the lab offers extended opening hours (twenty-four-seven) when compared to a regular lab. Besides the advantages for students, this also reduces the costs for academic teaching and improves the quality by offering more practical training possibilities.

One implementation problem is to protect the physical systems in the lab against wrong control algorithms of unskilled students without defining too many constraints. Students should be free in their decisions and develop own creative solutions. They can implement their own design strategies. A reference design and a method to check the students' design against this reference is needed [16] to protect the physical system in the remote lab. The reference design should be independent of the used control unit and the development tools. This is done by the protection unit of the physical system.

This basic idea was the starting point for our extension of the *REAL* system and the design of a standalone protection unit for each physical system, which will be described in the next section (see Figure 3).

In the classic setup of the *REAL* system, every physical system in the remote lab is connected to one specific control unit (see Figure 2). The user can connect to the selected control unit (e.g., a microcontroller, Beck IPC, FPGA or a PLC) over the Internet. All functionality, including the reference model checking, has to be executed in the control unit. This imposes some constraints on the students' work:

[1] [1] Goal of the *REAL* (**Re**mote and **A**pplications **L**aboratory) system is to show new ways and chances of remote controlling and remote observation of real processes (e.g., in the fields of control engineering, robotics, tele-control engineering), dealing with integrated and interactive usage of modern Internet and intranet technologies, like WWW, HTML, Java, etc.

[2] [2] Hybrid remote labs provide both remote experiments and real electro-mechanical models in the remote lab as well as simulation models of the physical system.

- the reference design has to be included at design time,
- the student is forced to use a certain set of tools and project structure and
- the correct use of the reference model in the compiled code has to be checked to prevent any intended or unintended manipulation.

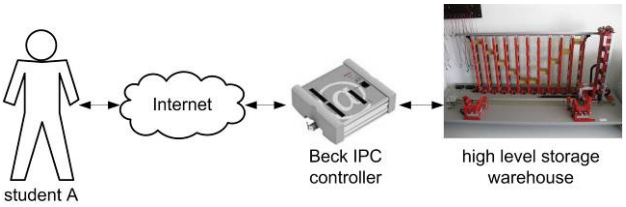

Figure 2. Classic setup of the remote lab

In our new approach, we want to eliminate a possible bypass of the protection functionality by taking it away from the control unit completely. This simplifies the development process of the control algorithm at the student side, because no special constraints apply. The student is completely free in using any control unit and development tool he likes [17].

The implemented remote lab infrastructure is based on the iLab architecture of the MIT, which allows to interconnect remote labs and to exchange remote lab experiments among different universities worldwide. Details about this lab server are beyond the scope of this paper. Please refer to [8] for further information.

A. *Internal Serial Remote Lab Bus*

For a reliable, universal and more flexible usage of our remote lab we installed an internal serial ***remote lab bus***. All Web-control units and all the physical systems are interconnected via this bus. With this bus concept we are able to add additional Web-control units on the one side as well as further physical systems on the other side to build an internal remote lab grid, where any combination of physical hardware and control units can be used for experiments.

B. *Bus Protection Unit*

The ***bus protection unit*** receives commands from a control unit and simply checks them for bus validity. This is done by verifying the transport protocol. The content of the transmitted data and addresses are not checked, because this depends on the physical system used and is

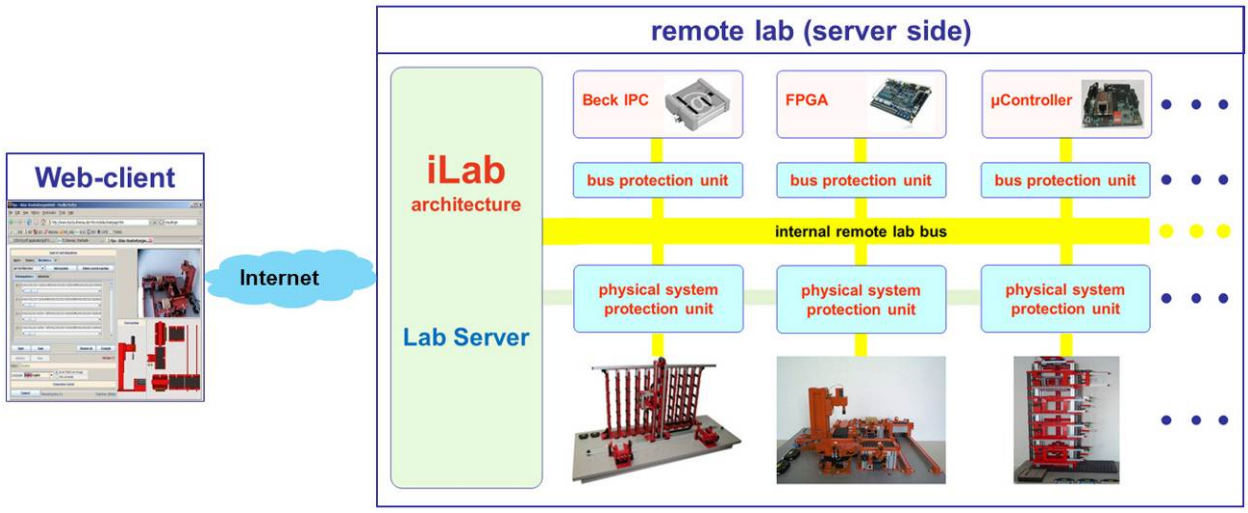

Figure 3. Grid architecture of the remote lab of the new *REAL* system (server side) with Web-client

In the following, we will describe a grid concept to realize a universal remote lab infrastructure, shown in Figure 3. This new infrastructure consists of three parts:

- an internal serial ***remote lab bus*** to interconnect all parts of the remote lab,
- a ***bus protection unit*** to interface the control units to the remote lab bus and to protect the bus from blockage, misuse and damage as well as
- a ***physical system protection unit***, which protects the physical systems (the electro-mechanical models in the remote lab) against deliberate damage or accidentally wrong control commands and which offers different access and control mechanisms.

The interconnection between the Web-control units and the selected physical systems during a remote lab work session (experiment) as well as the user management is done by the lab server, which also handles the webcams. therefore done by the specific protection unit of each physical system. The function of the bus protection unit is to prevent a control unit from blocking the bus and causing other experiments to be affected. The bus protection unit is based on the same hardware as the protection units for the physical systems in the remote lab but uses a different population of components to simplify production and maintenance. A secondary task for the bus protection unit is to interface various control units like PLCs to the bus. These units, for example, can only supply simple I/O control which is intended to interface the physical systems directly, therefore no protocol is implemented. The serialization of the I/O signals and the implementation of the protocol are handled by the bus protection unit. The level shifting, in case the control unit is not compliant with the bus voltage, is done by the bus protection unit as well.

Furthermore the bus protection unit acts as programmer for the connected control units. It receives the programming files from the Web-client via the lab server.

C. Protection Unit for the Physical Systems

The ***physical system protection unit*** is necessary when students execute their algorithms directly on the control unit and want to be free in their choice of design tools. Compared to the approach used so far, there was no possibility outside of the control unit to check, if the executed commands are safe for the physical system. This means, damage could be caused by invalid commands. The task of the protection unit is to check for command safety by filtering all commands. Only commands that will not cause any malfunction are executed (see Figure 4). All others are discarded and optionally reported as an error condition to a learning management system (LMS).

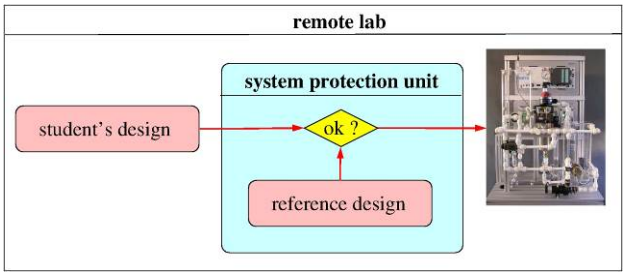

Figure 4. Observation of the student's design by the physical system protection unit

This concept can be used with all Web-control units like microcontroller, FPGA, PLC, etc. Using such a universal protection unit gives the students the largest degree of freedom for their design, because no precautions have to be taken into account. Therefore, no additional security framework (workbench) within the software and hardware control design is required to prevent malfunctions of the physical system. The complete design flow is carried out at the students' side, giving them a more authentic look at a real world project design flow. The valid behavior defined by the physical system protection unit is compared against the commands received from the control unit to determine any unsafe control commands of a faulty student's control algorithm. Commands that are valid for the current state of the physical system are passed. Commands that are not valid and could cause a failure or any other unwanted behavior are discarded and optionally reported.

Besides pure Web-based training for students, controlled by the control units, the physical system protection unit also offers the possibilities to be accessed by the teachers over the Internet for initialization of experiments or be accessed by local control. It is also possible to mix signal sources to get certain sensor signals from the physical system and supply others via a Web-interface. This can be used in cases that require a user input at the physical system to test the system behavior (like a key press at an elevator).

D. Additional Features – LMS Interface

Besides the features already mentioned in this article, even more functionality can be added using the new concept of having a Web-based protection unit that checks the user input against a reference model.

The protection unit can be connected to a learning management system like "moodle" to forward any experimental results of the user. For an effective usage of the *REAL* system within learning management systems, the reference design and a method to check the student's design against this reference design step by step will be traced by the LMS. Figure 6 shows this idea.

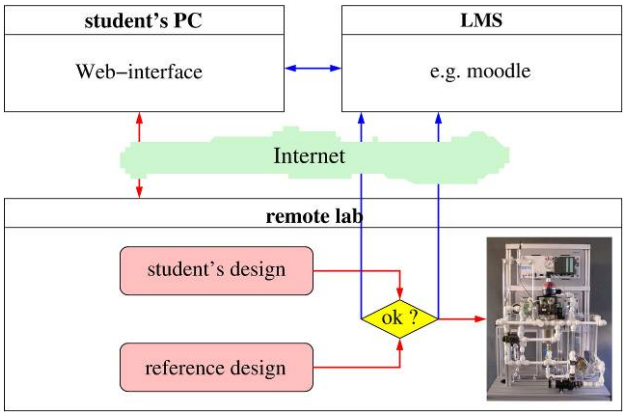

Figure 6. Observation of the student's design under LMS control

If the student's design as well as the reference design produces the same output value, the resulting action can be executed by the remote lab. Otherwise, the LMS will be informed and the student gets some hints to correct his design. For a detailed description see [17], [18] and [19]

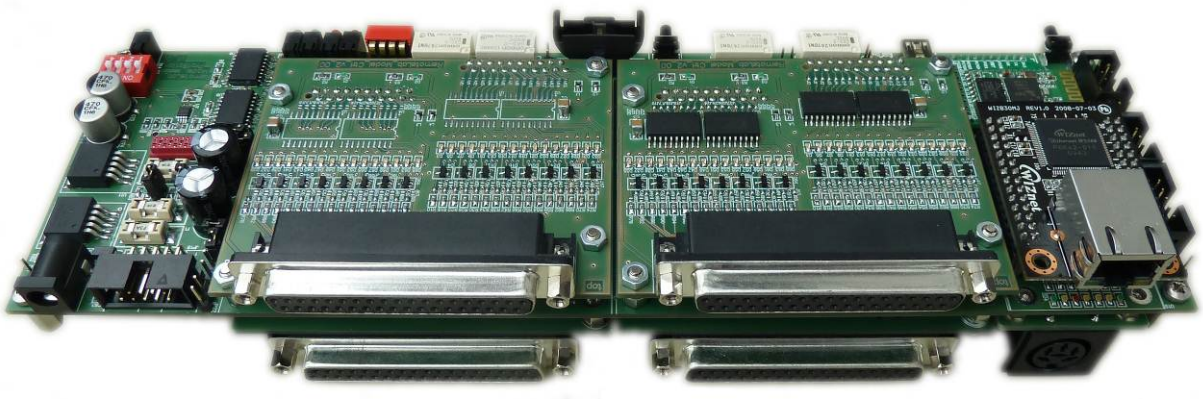

Figure 5. Hardware of the physical system protection unit

III. OPERATION MODES

Based on this flexible remote lab structure we offer different operation modes to test the developed control task for a universal usage of our *REAL* system. Simulation and visual prototyping help to find functional errors. Before starting practical work on real systems, simulations and animations in "virtual worlds" are often used to verify the developed solutions. The behavior of the physical system that should be controlled, as well as its environment, will be emulated as a simulation model. The student can influence this "virtual world" and analyze the caused reaction of his control algorithm. Figure 8 shows an example of such a simulation model.

These steps have to be executed until no more errors are detected. But there is an essential disadvantage in this method. Real disruptive factors (e.g., failure of single components, mechanical problems or process variations) cannot be recognized by the underlying virtual environmental model. Generally, only a simulation of predetermined malfunctions is possible. After some time, all these effects are well known in the student's community. Unconsidered sources of errors lead to undetected failures of the control because the corresponding environmental situation was not simulated before [15]. That is why a fault free design algorithm finally should be tested on real physical systems (e.g., the water level control, shown in Figure 8) in the remote laboratory as well.

In the following we will describe possible operation modes of the complete *REAL* system based on the schematic view in Figure 3. The complete schematic view (client and server side) of the remote lab architecture is shown in Figure 8.

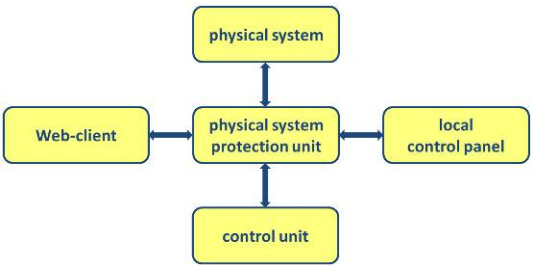

Figure 8. Schematic view of the remote lab components

Via the *Web-client* (e.g., a Java applet running on the student's PC at home) the *physical system* (e.g., the electro-mechanical model of the water level control) can be controlled by using different *control units* (e.g., microcontroller, FPGA, PLC). By using this Web-interface, the student is able to:

- handle the experiment (e.g., start, stop, reset),
- change environmental variables if necessary and
- watch the experiment by manipulating environmental variables inside an I/O monitor or by observing the control of the physical system directly via a webcam.

Via an optional *local control panel* the student is able to manipulate the lab environment (e.g., the water level in the tank) or the actuators (e.g., the pump) when working on-site. Heart of this architecture is the *physical hardware protection unit*, which is described in section II.C in detail.

A. Stand-alone Mode – Visual Prototyping

In case of using finite state machines for specification, based upon an automaton graph, a student can use the JGIFT design environment [20] of the *REAL* system.

Assuming the student achieved a validated design, he gets the required next state and the output equations. By accessing the Web-browser interface of the *REAL* system, he is able to enter his algorithm (the received equations), handle the laboratory experiment (e.g., start, stop, reset) and change environmental variables if necessary. The control algorithm is executed by an interpreter, running inside the student's client PC (e.g., implemented as Java applet). No Internet connectivity to the physical systems in the remote lab is necessary for this mode (see Figure 9).

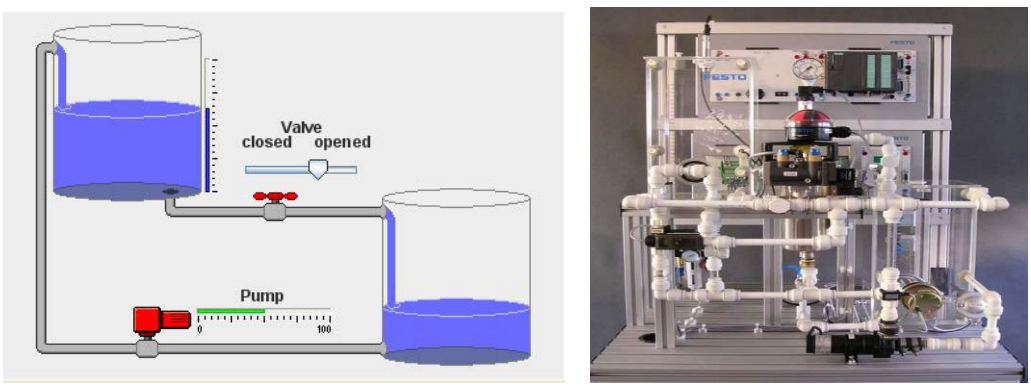

Figure 7. Simulation model and physical system of a water level control

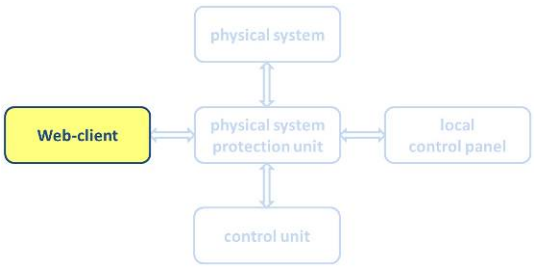

Figure 9. Stand-alone mode for visual prototyping

Figure 10 gives an impression of the verification and simulation features of the Web-based environment of the *REAL* system. The simulation model will be driven directly through the input/output signals by the control

algorithm running on the embedded interpreter within the Java applet.

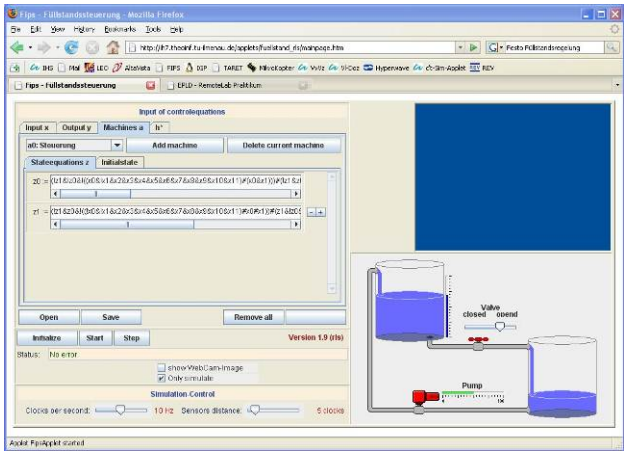

Figure 10. Offline regulation (visual prototyping) of the water level control - without Internet connectivity to the remote lab

B. *Remote Control Mode – via Web-client*

This operation mode is also for FSM based specification based on equations. In this case, the physical system will be controlled via the Internet "from a distance" through the interpreter, running inside the student's client PC. No additional control units in the remote lab are necessary (see Figure 11). In this case, only the input and output signals of the physical system will be transferred via Internet.

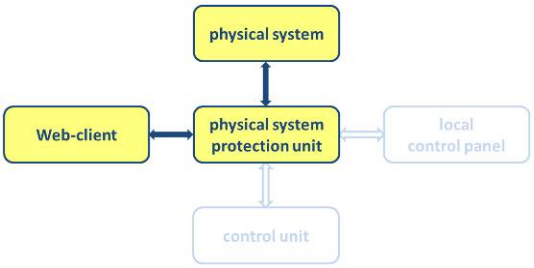

Figure 11. Remote control mode via a Web-client

An example of such a Web-client is shown in Figure 12.

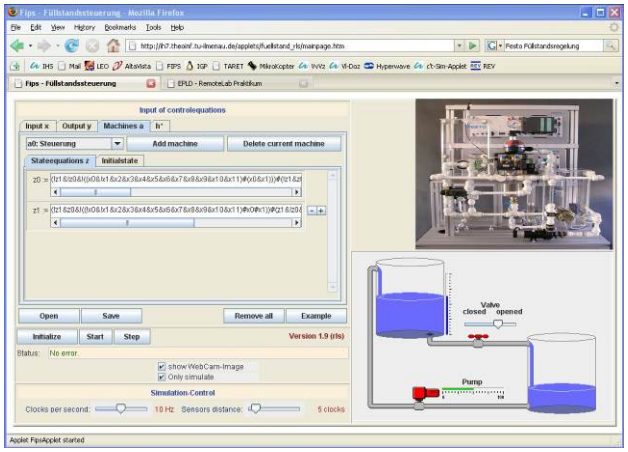

Figure 12. Online regulation of the water level control - with Internet connectivity to the remote lab

C. *Remote Control Mode – via Control Unit*

This mode can be used to realize software or hardware oriented control tasks. Students can implement their control algorithm directly into a microcontroller for a software-oriented implementation. Therefore, they use common (non-commercial) development tools, for example MPLAB IDE and/or C18 C-compiler by Microchip [21], or AVR Studio by Atmel [22], to develop Assembler- and/or C-coded software projects. After compilation, the generated software control algorithm is transferred via *REAL* Web-interface to the remote lab, where the hex code is programmed into the microcontroller (see Figure 13). Now, the student can begin with his experiment to check, if his algorithm fulfills the requirements of the given control task.

If a student prefers an exclusive hardware-oriented design using an FPGA and applying a hardware description language like VHDL as specification technique, he can prepare his design with common development tools, for example ISE, Quartus II, Diamond or others. The generated bit file is uploaded via the *REAL* Web-interface to the remote lab, where the FPGA will be programmed (see Figure 13). After programming the connected FPGA, the FPGA board operates as control unit for the designed control algorithm, and the student can start his experiment.

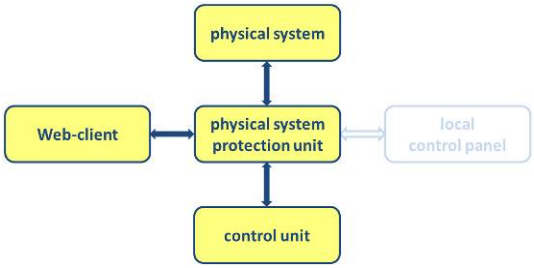

Figure 13. Remote control mode via a Web-client

D. *Virtual Control Mode – Visual Prototyping*

This mode is comparable to the operation mode A (for visual prototyping). In this case, the simulation model is not connected to the interpreter, running inside the client. The visual model is connected via the Internet to the real control units which are running inside the remote lab. In this case, the student can test his prepared software or hardware oriented design on the corresponding control unit (microcontroller or FPGA) without the need for a real physical system – before he will use operation mode C (remote control mode – via control unit) to test his design task on the physical system in the remote lab.

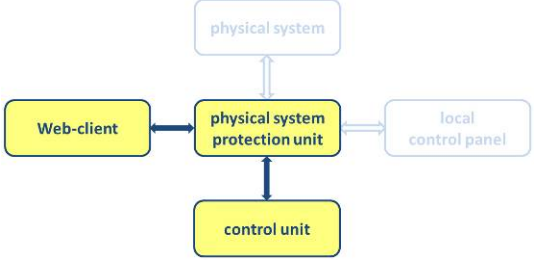

Figure 14. Virtual control mode for visual prototyping

E. *Virtual Control Mode – Test Mode*

This operation mode (see Figure 15) is mainly for debugging, testing and the maintaining of the protection unit.

By using this operation mode, it is possible to check the implemented reference design (as seen in Figure 4) in the physical system protection unit without the need to use a control unit or a physical system. This will be done by transmitting input and output pattern from the Web-client to the protection unit and by analyzing its response.

Because this mode is not preferential to execute lab experiments, it is not explained in detail.

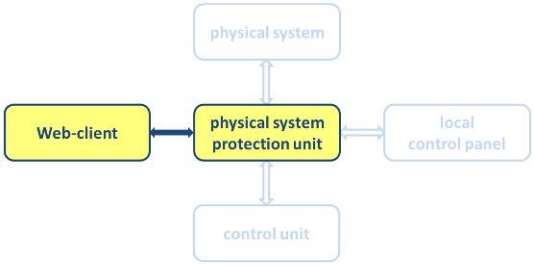

Figure 15. Virtual control mode to test the protection unit

F. *Local Control Mode – via a Control Unit*

Besides the possibility to work in the lab remotely, the following two operation modes explain the usage of the *REAL* system for on-site experiments or demonstrations.

During an on-site lab experiment students can observe the whole hardware setup as well as all the physical system and environmental variables directly in the lab room. In this case they can check their control task (running on a connected control unit). They have access to the whole lab setup via a connected control panel. Figure 16 illustrates this operation mode.

Besides for student experiments, this mode can also be used to demonstrate the remote lab on guided tours during open house presentations to inspire new students to study engineering courses.

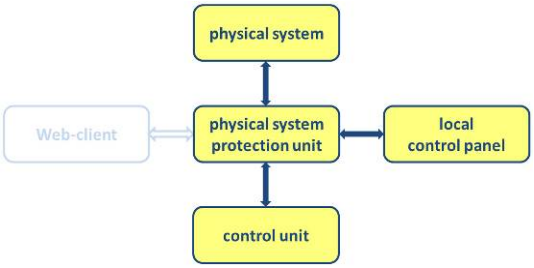

Figure 16. Local control mode via a control unit

G. *Local Control Mode – manually*

This operation mode (see Figure 17) can be used for demonstrations and maintenance of the connected physical system. By activating the actuators (e.g., a water pump) manually without any control algorithm the sensor signals can be observed via the connected control panel. This operation mode is not preferential to execute a lab experiment.

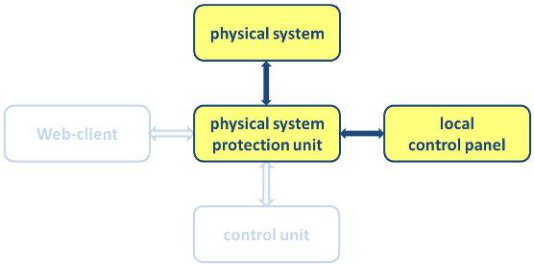

Figure 17. Manually local control

IV. CONCLUSION

We discussed a new grid concept for our remote lab, which mainly consists of an internal remote lab bus, a bus protection unit and a protection unit which protects the physical systems against deliberate damage or accidentally wrong control commands. These extensions offer possibilities for a flexible, scalable and robust usage of remote labs, giving the students a high degree of freedom for doing practical work inside the lab. Limitations and constraints like restricted opening hours, preselected design tools and/or workbenches as well as language limitations for international students) are reduced significantly by the new grid architecture.

This flexible grid-based structure of our remote lab offers a number of different operation modes for a universal usage within our *REAL* system. Besides simulation-based experiments also on-side and off-side experiments with real physical systems can be offered for students.

A connection to a learning management system is also possible. This can give feed-back to the students for every experimental result taken by their implemented control algorithms.

Our Integrated Communication Systems Group at the Ilmenau University of Technology is involved in different national and international e-Learning projects [23], [24] in which it is increasingly necessary to allow and organize a shared use of equipment. That is why, the main focus of the *REAL* system is

- a Web-wide usage of different design tools and control units to control different physical systems in the lab room,
- a robust, fault-protected access to any connected physical system and
- an LMS-coupling for all control units and physical systems used in the remote lab.

All these requirements can be fulfilled using the concept and the new components presented in this paper.

ACKNOWLEDGMENT

Parts of the *REAL* project are supported by the society “Friends of the Faculty of Computer Science and Automation”.

REFERENCES

[1] S. Sire, F. Geoffroy and D. Gillet: “A Virtual Assistant for Sending Hints and Perturbations to Students based on an Electronic Laboratory Journal (eJournal)”, Proceedings of the ITHET’03, Marrakech, Morocco, July 7-9, 2003.

[2] K. Henke and H.-D. Wuttke: "Web-based educational tool access", IASTED International Conference Computers and Advanced Technology in Education – CATE 2003, Rhodes, Greece, June 30 - July 2, 2003.

[3] H.-D. Wuttke and K. Henke: "Living Pictures – tool-oriented learning modules and laboratory for teaching digital via Internet", Proceedings of the ICEE-2002 International Conference on Engineering Education UMIST, Manchester, Great Britain, August 18-22, 2002.

[4] The iLab Project, from: https://wikis.mit.edu/confluence/display/ILAB2/Home

[5] WebLab Deusto, from: https://www.weblab.deusto.es/web.

[6] VISIR, from: http://openlabs.bth.se/index.php.

[7] LabShare Sahara, from: http://www.labshare.edu.au/project/index.php.

[8] MIT iLab Service Broker, Project homepage, from: http://ilab.mit.edu/iLabServiceBroker.

[9] K. Henke and S. Krug: "Web-based Rapid Prototyping of Digital Systems", International Conference on Interactive Computer-Aided Blended Learning – IMCL2011, Antigua, Guatemala, November 02-04, 2011.

[10] REAL: "Remote Engineering and Applications Laboratory", http://lanai.theoinf.tu-ilmenau.de/applets/index.htm.

[11] Y. Torroja, et al.: "A Modular Environment for Learning Digital Control Applications", Microelectronics Education, Marcombo, S.A, 2002.

[12] T. A. Fjeldly, J. O. Strandman, R. Berntzen and M. S. Shur: "Advanced Solutions for Laboratory Experiments over the Internet", Engineering Education and Research-2001, Begell House Publishing.

[13] H.-D. Wuttke, K. Henke and N. Ludwig: "remote labs versus Virtual Labs for Teaching Digital System Design", Proceedings of the Int. Conf. On Computer Systems and Technologies CompSysTech'05, Varna, 2005.

[14] K. Henke, H.-D. Wuttke and T. Braune: "Virtual and remote labs in the Educational Process", International Conference on Remote Engineering and Virtual Instrumentation, REV2007, Porto, Portugal, June 25-27, 2007.

[15] K. Henke, H.-D. Wuttke and S. Hellbach: "Laboratory via Internet – new ways in education and research", Int. Journal of Computers and Applications, vol. 25, ACTA press, 2002.

[16] K. Henke, H.-D. Wuttke and T. Braune: "Rapid Prototyping Modules for Remote Engineering Applications", International Conference on Remote Engineering and Virtual Instrumentation, REV2008, Düsseldorf, Germany, June 23-25, 2008.

[17] H.-D. Wuttke, R. Ubar, K. Henke and A. Jutman: "Assessment of Student's Design Results in E-Learning-Scenarios", 8th Conference on Information Technology Based Higher Education and Training (ITHET2007), Kumamoto City, Japan, July 10-13, 2007.

[18] K. Henke: "Reusable Assessment Objects for Learning Management Systems", Computers and Advanced Technology in Education"(CATE 2007), Beijing, China, October 8–10, 2007.

[19] K. Henke, St. Ostendorff and H.-D. Wuttke: "A Flexible and Scalable Infrastructure for Remote Laboratories - Robustness in Remote Engineering Laboratories", The Impact of Virtual, Remote and Real Logistics Labs - ImViReLL2012, Bremen, February 28 – March 02, 2012.

[20] JGIFT: "Java-based Graphical Interactive FSM Tools", http://wcms1.rz.tu-ilmenau.de/fakia/index.php?id=780, TU Ilmenau.

[21] Microchip Corp., www.microchip.com.

[22] Atmel Corp, http://www.atmel.com.

[23] M.E. Auer, I. Grout, K. Henke, R. Safaric and D. Ursutiu: "A Joint Master Program in Remote Engineering", International Journal of Online Engineering (iJOE) [Online], Vol. 2, No. 2, April 30, 2006.

[24] TRE – International Summer School in Technologies for Remote Engineering, http://www.fh-campuswien.ac.at/en/international/summer_school

AUTHORS

Karsten Henke is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: karsten.henke@tu-ilmenau.de).

Steffen Ostendorff is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: steffen.ostendorff@tu-ilmenau.de).

Heinz-Dietrich Wuttke is with the Ilmenau University of Technology, Faculty of Computer Science and Automation, Integrated Communication Systems Group, 98684 Ilmenau, Germany, POB 10 05 65 (e-mail: dieter.wuttke@tu-ilmenau.de).

Stefan Vogel is with the Ilmenau University of Technology, Master Student in Computer Engineering at the Faculty of Computer Science and Automation, 98684 Ilmenau, Germany, POB 10 05 65. He obtained his BSc. in Computer Engineering in 2011 (e-mail: stefan.vogel@tu-ilmenau.de).

This article 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. Received 12 September 2012. Published as resubmitted by the authors 14 November 2012.