Research Report 56/2006, TU Wien, Institut für Technische Informatik, April 24, 2006

Anytime, Everywhere — Approaches to Distance Labs in Embedded Systems Education

Markus Proske
Vienna University of Technology,
Institute of Computer Engineering,
Vienna, Austria
`proske@ecs.tuwien.ac.at`

Christian Trödhandl
Vienna University of Technology,
Institute of Computer Engineering,
Vienna, Austria
`troedhandl@ecs.tuwien.ac.at`

Abstract

Embedded Systems education is a vital part of the computer engineering curriculum. Increasing numbers of students stress our lab capacities, time constraints encumber working students whereas handicapped students often have difficulties attending the lab. Our project "Seamless Campus: Distance Labs"[1] *introduces remote-teaching into our hardware-centered courses. These courses are slightly different from other programming courses that focus on the development of typical application software where setting up a development environment on the students PCs is a relatively easy task. With this paper we will investigate the additional requirements that arise for different types of embedded system labs. We examine three possible concepts for implementing distance labs. A crucial aspect for the acceptance of remote labs is that as much as possible of the "feeling" of the on-site workplace is retained with the remote workplace setup. Therefore we will give a guideline how the "hands-on" aspect of hardware-centered courses can be retained with remote labs.*

Keywords— *Embedded systems education, remote-teaching, distance-labs.*

1. Introduction

The fundamentals of the practical work in the computer engineering curriculum are the lab courses that teach students different aspects of embedded systems programming and design. These courses are typically characterized by a high demand on specific hardware equipment. A typical approach taken by many universities to increase the lab capacity of programming courses is to let students do certain programming tasks at home, which at the same time also helps working or handicapped students to attend the course.

We investigated three possible concepts: enabling remote-access to our existing hardware labs, mass production with distribution of hardware to our students and the use of simulators as in [1]. In [2] the advantages and disadvantages of remote labs versus simulation and real laboratories are compared:

Real Lab: The main advantage of real labs is the interaction with real equipment that provides realistic data. Collaborative work is easy and there is direct interaction with the supervisor. The main disadvantages of real labs are time and place restrictions (needs scheduling of lab slots) and the expensive equipment that requires supervision.

Simulation: The advantages of simulation are the low costs and the lack of time and place restrictions. The main arguments against simulation are the missing interaction with real equipment and the use of idealized data — simulation often behaves slightly different than the real hardware.

Remote Lab: The remote lab approach combines the main advantages of real labs (realistic data, interaction with real equipment) without the disadvantage of time and place restrictions. Typically the costs of a remote lab are between that of the real lab and the simulation. The disadvantage of a just "virtual presence" in the lab is dependent on how well the user interface of the real lab is mapped to the remote lab.

Although a simulation-only approach provides a cost-effective way to realize remote workplaces in hardware-centered courses, we think that it is inevitable to deploy real hardware to convey the technical skills to our students that are needed for the development of embedded systems. Therefore we discarded simulators in an early stage because we believe that nothing can substitute "hands-on" experiences since simulators do not capture all problems that arise with real hardware.

Evaluations of deployed remote labs (see [3], [2]) suggest that these are comparable in effectiveness to real laboratories. In some cases remote labs are even rated higher by the students because of their flexible accessibility. An important point for the acceptance of remote labs is a good user interface. It should be structured in a way that the "feeling" of the on-site workplace is retained with the remote workplace setup.

Online engineering work is not restricted to remote teaching but also gains acceptance in industrial environments [4]:

- Use of expensive equipment from different locations.

[1]This work is part of the SCDL "Seamless Campus: Distance Labs" project, which received support from the Austrian "FIT-IT Embedded Systems" initiative, funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) and managed by the Austrian Research Promotion Agency (FFG) under grant 808210. See http://www.ecs.tuwien.ac.at/Projects/SCDL/ for further information.

- Complex experimental systems can be directly controlled from the scientist's office.
- Team members can work on the same problems from different locations.
- Long-term trials can be supervised from home, e.g., at weekends.

The paper is structured in the following way: Section 2 will show the requirements for different types of hardware-centered lab courses. Section 3 presents different approaches to deploy distance-labs for these courses. Additional components of the remote learning framework are presented in Section 4. Section 5 is dedicated to case studies on the implementation of distance-labs in three of our lab-courses and Section 6 will conclude the paper.

2. Requirements for Different Types of Embedded Systems Labs

Since the range of embedded systems is very diverse, the requirements for different kinds of embedded system labs vary over a wide range, much more than for a typical application-programming lab. Unlike the more software-centered domains where a typical student is able to provide the means for software development himself (e.g., a desktop PC, standard compilers), the embedded systems field has more specific requirements.

First is the development software: Since different embedded systems require distinct development tools, the setup of the development software is specific to the deployed hardware. Often, the used software packages are accompanied by a restrictive and/or expensive software license. This effectively prevents the usage of such software in a remote setup, since most students would not accept to buy expensive licenses or use a system where the setup of the software is very complicated.

The second difficulty in embedded system labs is the hardware itself: As already mentioned there is a wide range of hardware that is used — from very cheap microcontrollers to very expensive logic analyzers. Additionally there are very different options of how to connect the hardware to the computer: Serial port, USB, printer port, or specific interfaces like JTAG, that require additional interface hardware. This brings up the problem of providing driver software for these interfaces.

To get an overview about the possibilities of distance-labs, we looked at the requirements for three of our hardware-centered courses in greater detail:

Digital Design: In the Digital Design course, students learn how to design hardware by using programmable logic. As target technology we utilize Field-Programmable Gate Arrays (FPGAs). In recent years the use of FPGAs and with it the adoption of hardware description languages like VHDL for hardware design has found a rapid increase of application throughout the industry. The course is divided in several parts: The students learn, how to use logic analyzers and the debugging of hardware. The debugging process of hardware is fundamentally different from the debugging methods used in software development — mostly the internal state of the hardware cannot be accessed directly. Other parts are the design flow of the hardware design process (specification – coding – simulation – synthesis) and the VHDL programming itself. The software used in the design process (simulator, synthesis tool) is very specific to the deployed hardware architecture and is bound to restrictive license terms. Software license costs are about 1000 EUR per workplace. Another aspect is the hardware price tag. A single Digital Design workplace (PC, logic analyzer, hardware target, see Figure 1) accounts for 17500 EUR.

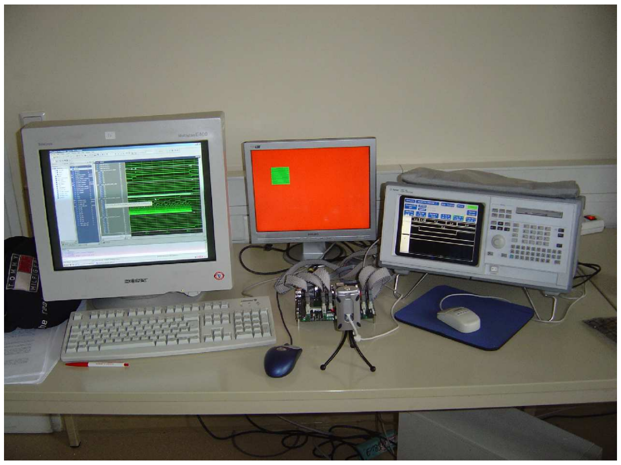


Figure 1: A Digital Design workplace with development system, target system with attached VGA monitor, and logic analyzer

Microcontroller: For most of our students the Microcontroller course is the first contact with hardware-near programming and microcontrollers. It requires basic electrical engineering knowledge and C programming skills. The course starts with an introduction to microcontroller architecture, assembly programming, and the programming environment. The second part teaches basic microcontroller features such as I/O ports, timers, a/d converters, and interrupts. The last part concludes the course with a more elaborate microcontroller application. A fundamental part of the programming examples is the setup of the board where the students have to connect different subcomponents of the microcontroller to the additional hardware to form a working system. Therefore the main aspect of this course is to give the students a real "hands-on" introduction into microcontroller programming.

Embedded Systems Programming: The Embedded Systems Programming (ESP) course addresses students who have already completed the Microcontroller course. It teaches various aspects of embedded systems programming in C programming language. In the ESP setup we use multiple 8-bit microcontroller nodes that are connected through a fieldbus network. Attached to each microcontroller node are hardware modules. The configuration of the hardware stays the same throughout the course. This is a significant difference to the Microcontroller course where the hardware configuration changes with every example. A complete ESP target board costs about 300 EUR. The main aspect of this course is to teach the students how to build distributed microcontroller applications and to use the fieldbus network for communication between different microcontrollers.

The comparison between the the different courses shows that the requirements for the courses are different from each other which will lead to different approaches for the setup of distance labs.

3. Different Approaches to Distance Labs

Remote-accessing hardware can be achieved in different ways. In a simple approach we could take a remote PC to remotely control an existing workstation in our lab. The workstation contains all the software necessary for development. Target boards as well as measurement equipment are connected to the workstation. There are several tools available for this solution, a remote-desktop is even included in Windows XP. The main drawback of remote-controlled workstations is the one to one matching of local and remote clients that makes that approach not very scalable. Furthermore, a bulk of workstations creates a high effort for software updates and account management. But such a setup has advantages too: As all software is bound to the workstations, there are no additional problems with licenses or distribution of software. Using existing software for remote control, the setup of a distance lab requires few to none modifications of hard- and software.

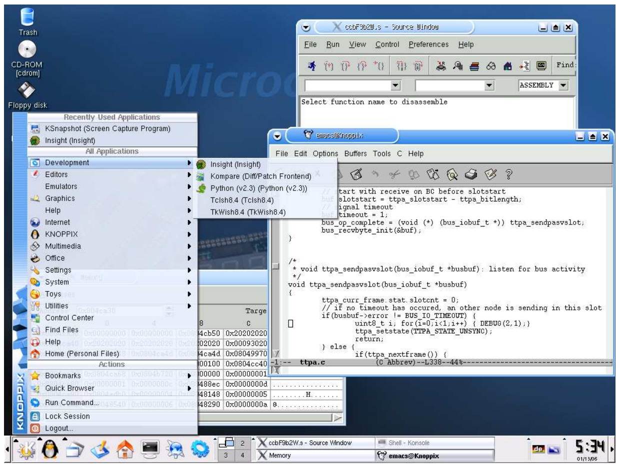
Alternatively, we could go for a client-server architecture. In this approach several target boards with integrated measurement hardware are connected to a server that manages the data transfer between boards and remote clients. The server handles user management and automatically assigns free target boards to authenticated users. In our client-server architecture, just measurement values and debugging streams are transferred via the Internet — all software necessary runs on the client's side. This reduces bandwidth usage by quite a large factor compared to the remote-controlled workstations. As software is needed at the client's side, licensing of software might be an issue as usually one server handles a large number of clients. Another advantage of the client-server approach over the remote-controlled workstations lies in scalability as additional boards are not only cheaper than additional workstations, they also account for less room capacity needed. As noted above, administering a couple of servers is preferable over managing the factor ten amount of workstations.

Our second strategy aims at getting the lab directly to our students. This approach requires cheap and robust hardware for mass production. The most crucial benefit is scalability. Once production is cheap, it's just a matter of logistics. Furthermore, labkits — as we call them — give students directly that precious hands-on experience. Participating in this kind of distance lab does not require any Internet connection and — on our side — no investment in hardware or rooms is needed.

4. Supporting Distance Labs

Of course, deploying distance labs is more than just planning hardware and access. We needed to build a framework to support the different approaches (labkits, remote workplace) for our courses.

Both the client server approach and the labkits require a software environment at the client's side. Usually, students use a broad variety of operating systems. Each student would have to install and setup the specific software needed for the course. Despite not having compatible software and documentation for all those systems, this moves quite a bunch of work to our students before they can start with their exercises. To simplify our student's lives we introduced a modified version of the Knoppix CD. The Knoppix CD is a bootable CD containing a complete Linux environment[2]. Students can just insert the CD and reboot their computer to run a preconfigured Linux environment. In addition to the operating system our version of the CD contains all software needed for their exercises. Without any installation procedures students can directly start with their tasks. All documentation and course materials are available on the CD, including our "electronic slides" — slides, texts and videos enriched with the professors voice comments. The graphical desktop of our Knoppix CD showing the development environment of the Microcontroller course is depicted in Figure 2.



```
Select function name to disassemble
```

```
        //  tart with receive on BC before slotstart
        buf  slotstart = ttpa_slotstart - ttpa_bitlength;
        //  ignal timeout
        buf  timeout = 1;
        bus_op_complete = (void (*) (bus_iobuf_t *)) ttpa_sendpasvslot;
        bus_recvbyte_init(&buf);
}


/*
 * void ttpa_sendpasvslot(bus_iobuf_t *busbuf): listen for bus activity
 */
void ttpa_sendpasvslot(bus_iobuf_t *busbuf)
{
        ttpa_curr_frame.stat.slotcnt = 0;
        // if no timeout has occured, an other node is sending in this slot
        if(busbuf->error != BUS_IO_TIMEOUT) {
                uint8_t i; for(i=0;i<1;i++) { DEBUG(2,1);}
                ttpa_setstate(TTPA_STATE_UNSYNC);
                return;
        } else {
                if(ttpa_nextframe()) {
-1:--  ttpa.c          (C Abbrev)--L338--44%------------------------
```

Figure 2: Development environment for Microcontroller course

In our former closed-lab courses students normally had supervised lab hours. As this concept has proven to work well in the past, we transferred it to the distance labs using voice/video over IP-technologies — the birth of "video-tutors".

On the organizational side, we will deploy a new web portal handling enrollments, submissions, grading, evaluation, and communication, similar to that described in [5]. Especially the communication between students over the web portal is an important aspect, since questions and answers from other students will give a deeper insight and help to build knowledge (see [6]). Students will receive reminders on upcoming deadlines and see private rankings comparing their own performance anonymously with those of their colleagues. The new internal administration

[2]See `http://www.knopper.net/knoppix/index-en.html` for further information on Knoppix.

system eases course manager's lives in simplifying and reducing monotonous, time-consuming tasks. Both systems are still in development and will be fully operational by the next winter term.

5. Case Studies: How Distance Labs are deployed in our Lab Courses

At our department, we picked three of our undergraduate courses in embedded systems for deploying the concepts.

The Digital Design employs the "remote-controlled workstation" concept for the reason of expensive soft- and hardware — the major criteria for using a remote workplace approach are a very efficient utilization of the workplaces and to protect the expensive equipment from physical damage.

The setup of the workplace is the same as for the onsite course: A Windows workstation runs the proprietary VHDL development tools. Attached to the PC is the FPGA target board and a USB webcam. The webcam is used to capture the output of the target board that is displayed on a monitor — the task of the students is to implement a VGA controller on the FPGA.

The logic analyzer that is connected to the target board to measure its outputs can be attached to the network and is capable to be controlled via a remote X-Window connection. Therefore we installed a Cygwin environment[3] with included X-Server on the workplace PCs.

For the students to control the workstations remotely, we use the Terminal Services provided by the Windows operating System. It uses the Remote Desktop Protocol (RDP) to communicate with client software like *rdesktop* (Linux) or *mstsc* (Windows). The output of the webcam and the logic analyzer are also displayed via the RDP client.

Alternatively to the webcam we could use a framegrabber card that captures the output of the target board directly.

Other than the Digital Design course, Embedded Systems utilizes the client-server architecture: self-developed boards equipped with a measurement fieldbus network sending data to a server in combination with a self-developed visualization tool at the client's side.

The system consists of an extended version of our currently used target boards. The target hardware consists of a board containing four microcontrollers that are programmed by the students with additional hardware (a display, an electric fan with integrated rotation speed sensor, a light bulb, and sensors for luminance and temperature). A programmer that supports subsequent programming and in-circuit debugging of multiple microcontrollers is also present. Additional the programmer also works as low-speed version of a logic analyzer, monitoring the fieldbus network. We use a second fieldbus network with sensor nodes to capture the outputs of the nodes programmed by the students. Figure 3 displays such a target board.

[3]Cygwin is a free software package that provides a full GNU environment under Windows operating systems. See `http://www.cygwin.com` for more information.

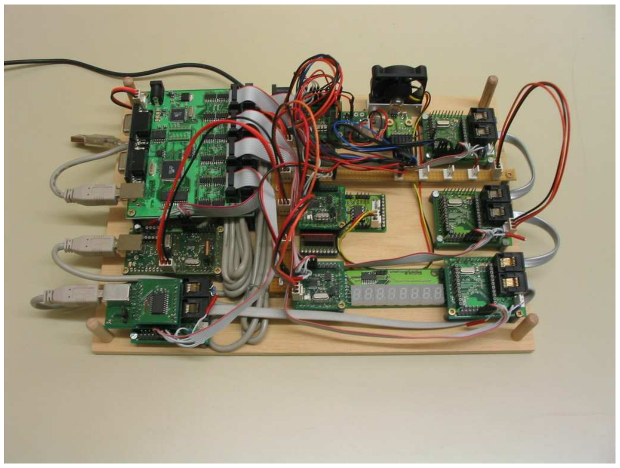


Figure 3: The remote workplace target board

The measurement framework is based on the real-time communication network TTP/A that enables an efficient transfer of periodic real-time data. The measured real-time data is sent to a gateway node that is connected via USB to a target server.

The development environment on the client side is brought to the user in form of a bootable Knoppix CD. The CD contains all required development tools, a local client software, and visualization software. Remote students will start a session by booting the development system on their PC. Afterwards, they use the client software to contact the authentication server. If the login succeeds, the user is connected to a free target, automatically managed by the target server.

The visualization software will display the measured values in a graphical visualization of the target board (e.g., the visualized display on the screen will show the same contents as the real display). Figure 4 shows the different parts of the remote workplace setup: The authentication server, the target server, the login client, and the visualization software.

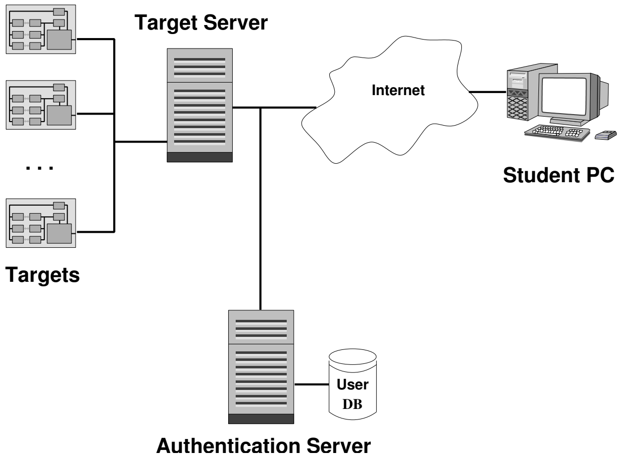


Figure 4: Overview of the remote workplace setup

Finally, the Microcontroller course builds upon the labkit concept. To be affordable even for the low budget we needed to find inexpensive target boards. Since we did not find any hardware that fulfilled our requirements, we developed our own a 8-bit microcontroller board. This self-developed hardware was optimized for cost-efficiency enabling mass production at around 70 EUR making our approach high-scalable. The usage of free software (GCC) as development environment also contributes to the low cost factor.

A main argument for not using a remote workplace approach for this course is that its concept is based on the premise that the students have to set up the connections on the target board — a aspect that is not possible for a remote workplace setup. The labkit contains all the items that are needed during the course: It will include our self-developed target board (microcontroller + I/O board), a set of wires, a power supply, cables to connect the board to the PC, the customized Knoppix CD, and a short documentation. The contents of the labkit is shown in Figure 5.

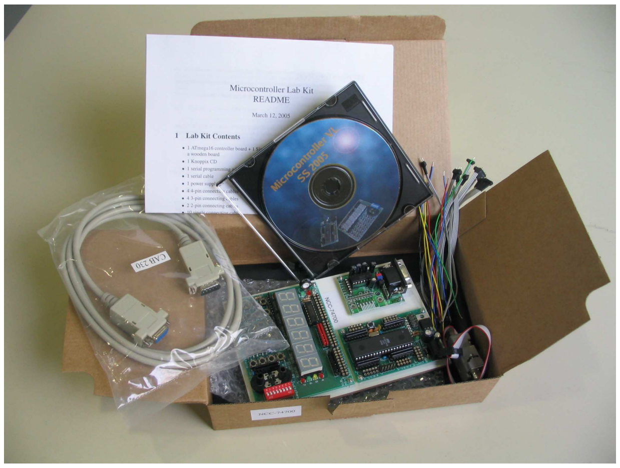


Figure 5: Contents of the labkit

To start with their programming tasks, the students simply have to connect the target board to both, power supply and PC, set up the connections on the board, and reboot their computer with the Knoppix CD — again the whole development environment is already set up and ready to use.

The Microcontroller labkit was deployed in a test run last year and found great acceptance among the students — many of them even bought the labkit at the end of the course. The Digital Design remote workplaces have also been deployed during the last year's course and received very positive student feedback. The Embedded Systems version will be evaluated in winter term.

Testing and Grading: As with all e-learning concepts one issue stands to be solved: testing and grading. We addressed the problem in our Microcontroller course concept. As stated in [7] especially for courses that primarily focus on programming, it is often more effective to hold online tests than using written exams. We have both practical and theoretical exams in our regular course. For the practical exams the individual tasks are very easy and generally consist of at most 20 lines of code. To place exam tasks within complex settings while still having rather simple tasks we provide a skeleton where students just have to fill in a couple of lines. During those tests our tutors are available to help with hardware troubles and even with debugging problems if students can prove that they just suffer from a blackout. At the end of the test, those tutors also check programs manually for correctness. Our theoretical exams are conducted using multiple choice tests with yes/no questions and a grading of +1 point for a correct answer, -1 point for a wrong answer and 0 points for no answer.

To deploy our course structure within the distance labs, some enhancements had been necessary [8]. Our theoretical exams are now held online using our multiple-choice framework. To assure the quality of our questions the student's tests are not instantly graded (although our framework could do that). After all students have finished their tests we run a statistical evaluation on the tests filtering out questions with a high percentage of wrong or no answers. Those questions are carefully examined for difficulty, clearness and errors in the database. Questions not passing this examination are consequently removed and thus not graded. For our practical exams we developed an automated test system [9] which can provide the microcontroller with specific test stimuli and verify its responses. The output of these tests is visualized on the LEDs and thus can be controlled by coaches not familiar with microcontrollers. Of course the results of a test can easily be taken over by any other external system if visualization is not required.

Both practical and theoretical exams make heavy use of tutors. In fact, they have two tasks: helping students and supervising the test. The first one, helping students, can be moved to our "video tutors". Supervision can be performed by any "trusted person", even if that person does not have any microcontroller skills. After practical exams, trusted persons can verify correctness of their student's programs using the automatic test system and its test cases. Using this approach we can have students all around the world participating in our lab and attending exams at their local university, guarded by trusted persons and supervised by our staff remotely from the Vienna University of Technology.

6. Summary and Conclusion

A fundamental part of today's engineering education is the "hands-on" experience gained from practical work in laboratories. The growing number of students confronts us with the problem of supplying the individual student with a sufficient amount of lab time. We evaluated the different requirements of our hardware-centered lab courses. To overcome the limitations of an on-site lab, we investigated various kinds of remote-learning concepts and distance labs.

With the labkit approach, used in the Microcontroller course, students receive a media bag with hard- and software that can be used with any PC, everywhere, so that each student can have his/her own laboratory at home. This approach is primary suitable for courses where the hardware can be acquired for a low price.

We also studied two different approaches for remote workplace setups. In the Digital Design course we opted for a remote desktop solution that uses the standard configuration of our on-site workplaces utilizing the built-in Remote Desktop Protocol to transmit the content of the desktop to the students. This approach was taken because of the expensive hardware and license issues with the proprietary development software.

With the Embedded Systems Programming course we used a client-server approach where a dedicated target server handles multiple connections from different users. In this case client-side visualization software is used to display the status of the target and the software development is done locally with a pre-configured development environment.

Additionally, we provided a software environment that can be used with all of our courses. One part of this framework is a customized Knoppix CD that features a pre-configured development environment and additional course materials.

An important aspect in distance labs is automatic testing and grading of practical and theoretical exams. We addressed this issue with the development of an automatic test system for software on embedded systems and a framework for multiple-choice exams. Finally, with our web portal we support students and staff in the field of course organization, with discussion groups, and learning material.

The comparison between our different hardware-centered lab courses shows that there is no single "one fits all" approach for deploying remote teaching concepts. In cases where the needed hardware is cheap, each student can be supplied with a labkit. In other cases a remote workplace approach is favorable — either in the form of a client-server or a remote desktop application.

Depending on license issues, software restrictions or the hardware used, one or the other approach will better fit the course requirements.

7. References

[1] M. Amirijoo, A. Tešanović, and S. Nadjm-Tehrani, "Raising Motivation in Real-time Laboratories: The Soccer Scenario," *35th SIGCSE Technical Symposium on Computer Science Education*, March 2004, pp. 265–269.

[2] Z. Nedic, J. Machotka, and A. Nafalsk, "Remote Laboratories versus Virtual and Real Laboratories," *33th ASEE/IEEE Frontiers of Education Conference*, Boulder, CO, USA, November 5–9 2003, pp. T3E1–T3E6.

[3] J. E. Corter, J. V. Nickerson, S. K. Esche, and C. Chassapis, "Remote Versus Hands-On Labs: A Comparative Study," *34th ASEE/IEEE Frontiers of Education Conference*, Savannah, GA, USA, October 20–23 2004, pp. F1G–17–F1G–21.

[4] M. Auer, A. Pester, and D. U. C. Samoila, "Distributed Virtual and Remote Labs in Engineering," *2003 IEEE International Conference on Industrial Technology*, Maribor, Slovenia, December 10–12 2003, pp. 1208–1213.

[5] T. G. Cleaver and R. L. Toole, "Design of a Web-Based Education Environment," *9th ASEE/IEEE Frontiers in Education Conference*, San Juan, Puerto Rico, November 10–13 1999, pp. 12A3/1–12A3/5.

[6] W. Braga, "Evaluating Students on Internet Enhanced Engineering Courses," *32nd ASEE/IEEE Frontiers in Education Conference*, Boston, MA, USA, November 6–9 2002, pp. S1B–1–S1B–6.

[7] A. N. Kumar, "The Design of Online Tests for Computer Science I and Their Effectiveness," *29th ASEE/IEEE Frontiers in Education Conference*, San Juan, Puerto Rico, November 10–13 1999, pp. 13B3/1–13B3/5.

[8] B. Weiss, G. Gridling, and M. Proske, "A Case Study in Efficient Microcontroller Education," *Workshop on Embedded Systems Education (WESE'05)*, September 22nd, 2005, pp. 36–43.

[9] V. Legourski, C. Trödhandl, and B. Weiss, "A System for Automatic Testing of Embedded Software in Undergraduate Study Exercises," *Workshop on Embedded Systems Education (WESE'05)*, September 22nd, 2005, pp. 44–51.