Distance Labs – Embedded Systems @home

M. Proske
TU Vienna
proske@ecs.tuwien.ac.at

C. Trödhandl
TU Vienna
troedhandl@ecs.tuwien.ac.at

T. Handl
TU Vienna
handl@ecs.tuwien.ac.at

Abstract

Introducing remote-teaching into hardware-centered courses is a difficult endeavor – especially if one tries to retain the hands-on experience of a typical lab course. This paper introduces three different approaches for implementing distance labs in hardware-centered courses. With the *Microcontroller* course we will show an example where the hardware can be made inexpensive enough that every student can be supplied with his or her own hardware. An other approach is the implementation of remote workplaces, where targets are controlled remotely over the Internet. We used two courses with differing constraints, *Digital Design* and *Embedded Systems Programming* to evaluate two kinds of remote workplaces. This paper will give a guideline for what circumstances which approach is favorable.

Keywords: distance labs, embedded systems, e-learning, remote-teaching

1 Introduction

In the industry, there is an increasing demand for well trained and competent embedded systems engineers. Universities have to meet this demand in their courses [1]. The Vienna University of Technology has recently introduced the bachelor study "Computer Engineering" that contains several courses which focus on designing embedded systems. These courses are characterized by a higher demand on hardware equipment – especially if one strives to give hands-on experience with embedded hardware – compared to courses that focus solely on software development and use simulation to introduce students to the field of embedded systems. An approach for such a simulation-only course is found in [2].

This leads to the following demands that must be fulfilled by embedded systems courses:

- Real hands-on experience on hardware since simulations often behave slightly different than the "real thing".
- Improved accessibility of the course for handicapped and employed students.
- Open labs so that students with less experience in embedded programming can easily catch up with their colleagues.
- Profound knowledge requires an appropriate workload assigned to the students.
- Tight resource constraints (material and personal) of these courses.

To address these demands, we started the "Seamless Campus – Distance Labs" project[1], that solely focuses on the development of remote teaching concepts for embedded systems laboratory courses. The vision of this project is to use similar methods developed in the SCDL project in most of the embedded systems courses at the Vienna University of Technology.

[1]The "Seamless Campus: Distance Labs" project 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.

In this paper we will show different solutions for the realization of distance learning concepts in embedded systems courses: One that uses lab kits and two different solutions of remote workplaces. We introduced those in three different courses: *Digital Design*, *Microcontroller*, and *Embedded Systems Programming*.

This paper is structured as following: Section 2 will give an overview of the different concepts of the "Seamless Campus – Distance Labs" project. The following section will show, how these concepts are applied to the *Digital Design* course. Section 4 and Section 5 describe our results with the *Microcontroller* and *Embedded Systems Programming* course. Section 6 concludes the paper.

2 Concepts

2.1 Lab Kits

A lab kit is a complete collection of hard- and software that is needed to participate in a course. The most important aspect in deploying lab kits is to make them very inexpensive. This allows us to produce enough of those to be able to lend such a kit for a small deposit (under 100 Euro) to every student who attends the course – so every student can have his or her own little laboratory at home.

2.2 Remote Workplaces

Remote workplaces are laboratory workplaces that can be accessed over the Internet. The utilization of remote workplaces in the laboratory courses will have the following benefits compared to a local course:

- Students are not bound to fixed working hours. They have access to the remote workplaces from their home over the Internet at any time.
- Handicapped and working students are supported (e.g., working from home, working in the evening / on the weekend).
- Targets can be used outside the opening hours of the building – better utilization of the targets

In comparison to the usage of simulation, this approach uses the same hardware as the in-house course with the benefit that the development environment is the same, no matter if the student works in the laboratory or at home.

We will show two different examples how such remote workplaces can be implemented: The straightforward solution is to use the same setup as in the local laboratory course – a workstation, instruments, target board, and applications and use remote desktop software to forward the screen contents to the students. An other approach is to build dedicated remote targets and develop special software that handles the data transfer between the student's PC and the remote target server.

2.3 Electronic Course Material

Documentation and course materials play an important role in our distance lab concept. These materials include electronic course scripts, technical manuals, demonstration programs, but the major part are electronic slides and demonstration videos. We use *Techsmiths Camtasia Studio* to record slideshows or software demonstrations capturing the professor's computer screen. Additionally, his or her voice is recorded. Afterwards the electronic slides can be enhanced with text and graphics and saved in several standard media formats like quicktime or mpeg.

2.4 TI Portal

Our distance lab courses will be supported by our new TI-Portal that is currently being built. Uniting all computer engineering departments under one portal we provide course-based information at one single address[2]. Students will find full information, receive reminders, have their personalized management interface and functionalities suitable for distance labs as electronic submissions and self-assessment tests. Our course managers on the other hand can utilize a complex workflow-based administration system reducing stupid and time-consuming organizational tasks.

[2] `http://ti.tuwien.ac.at`, starting 2006

3 Digital Design

3.1 Overview

Digital Design is a compulsory lab for the bachelor study "Computer Engineering" held in the 3rd semester. One of our special aims is to provide fundamental knowledge and arouse interest in the field of modern digital hardware design. In parallel, a lecture is held in order to provide the basic knowledge and techniques in a more theoretical way. As a matter of fact, designing such a lab means having to cope with largely differing experiences with hardware on the student's side. Students are quite experienced with software design at this point, but only little knowledge of VLSI (Very-Large-Scale Integration) design can be assumed.

Another point influencing the design of a laboratory tutorial are the limited capacities in terms of rooms, tutors, and expensive measurement equipment. Actually, there are about 120 students attending this lab. So, beside knowledge, we have to provide maximum flexibility to practice (according to one's needs) and further on to solve the exercises. This also means maximal resource utilization, with an optimum of 24 hours per day. An additional problem to cope with is the fact that an "open lab" will always be subject to damage and eventually theft. As already mentioned above we talk about very expensive equipment.

3.2 Course Goals

In this course, FPGAs (Field-Programmable Gate Arrays) are used as target technology. Understanding programmable logic is one of the fundamental skills in the field of hardware design and is primarily taught in the accompanying lecture. In the laboratory, the students should gather practical experience with state of the art tools which are de facto industry standard. Also, first experiences with the hardware description language VHDL will be made. These experiences comprise not only syntax or semantics of a programming language, it is much more important to comprehend what the code written will look like in hardware. Additionally, not every statement that is syntactically correct can also be mapped onto a circuit.

Another major point is to gather hands-on experiences with logic analyzers as opposed to the simulation approach presented in [3]. We believe it is important to not only simulate logic circuits (although this is mandatory in the context of design flow, another very important aspect also covered in this lab) but to see that the design indeed becomes hardware and how it is working. For the same reason we decided to use a benchtop logic analyzer instead of a (cheaper) PC-card solution. A typical workplace is depicted in Figure 1.

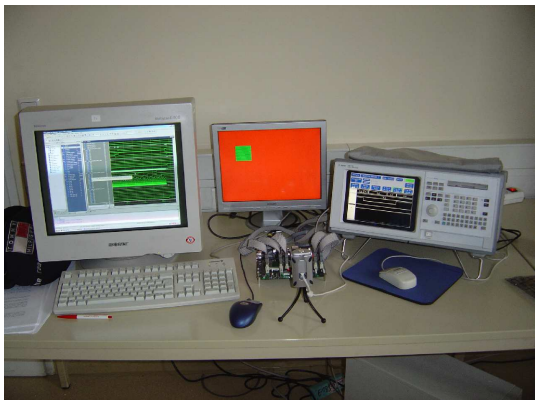

Figure 1: A Digital Design workplace

As far as the design flow, a methodology to design and build digital chips, is concerned, the students shall also learn how to debug hardware as the methods used here differ from those used in debugging software. Some keywords here are observability and accessibility (in fact, one can not see directly what happens inside a chip; this has to be considered in the design process as well as in the test phase).

3.3 Course Structure

The lab is divided into four parts, each one building on top of the other. In the first part, the students just have to download a pre-compiled design and learn the basic operation of the logic analyzer. They have to apply the methods described in the accompanying script.

The second part is centered around the design flow. After accomplishing a little modification of the source code, it is of major importance to comprehend what happens in each step of the design flow and what are the inputs and outputs of each step.

The third part incorporates the real introduction of VHDL, trying to convey all the points mentioned above by means of a slightly larger programming example.

The final part is concerned with debugging hardware. The students are confronted with a deliberately faulty design. It is their task to identify, locate, and remove all these faults. Thus, they have to apply all the knowledge gathered in the preceding three parts.

For all four parts it is important to understand the notion of finite state machines in a more practical way. Also, a basic understanding of models and abstraction is necessary throughout the course.

The target design has to be chosen very carefully. First of all, its usability for distance learning has to be considered. Every system response must be optically observable. Further, there must be no need for local manipulations, like there were with our former design, a printer. It also should be "cool" in the sense that students find it interesting and motivating. Everyone should be familiar with it – so no time consuming initial skill adaptation training is necessary. As a side effect, it also should be useful for the student's future. And finally, it should not be too complicated. The student's attention should remain focused on the main objectives of the lab. So we decided to use a design every computer engineer is familiar with and that has a rather low complexity of its own: a VGA controller (see Figure 2).

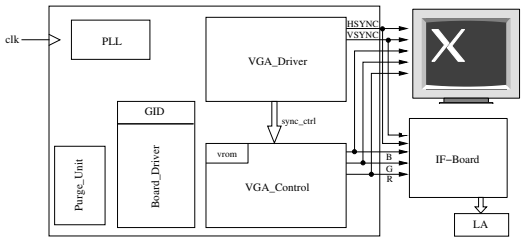

Figure 2: The VGA controller design

3.4 Grading

As mentioned above, the course is four-folded. Prior to beginning an example, each student has to do a PC-hosted multiple choice test. Each of these tests covers the topics the following as well as the last exercise were concerned with (except the first test of course; it deals only with the first example). The tests should ensure preparation of the students as well as their continuous learning. The four multiple choice tests make up 20% of the total score.

Associated with each solved exercise is a protocol the students have to write. It should cover the measurement results demanded in the script as well as the way they were achieved in term of screenshots of the logic analyzer and the simulation tool. Suggestions concerning the lab and constructive criticism are always welcome. The four protocols make up 30% of the total score and are submitted as a printed version. As a future prospect, we plan to set up an electronic submission.

At the end of the semester, after solving all the examples and multiple choice tests and after writing the protocols, an individual mandatory practical test has to be passed. This test is oriented at the fourth exercise, but due to the inherent stress of an exam it is kept slightly simpler. This test shall ensure that really every student is capable of performing the mentioned techniques on his or her own. It further shall guarantee that neither there is a group leader inhibiting the other group members from working nor that any group member is a hanger-on.

Concerning distance learning, there exists an obvious problem: nobody can hinder students from cheating when doing a multiple choice test remotely. Currently, these tests have to be performed with physical presence in the laboratory. As a future prospect, we plan to offer the multiple choice tests as a chance for self assessment. In that case, the grade will only be determined by the protocols and the practical test. As of today, the practical test still has to be performed in the laboratory. It has to be investigated by what means this test could be done remotely. Perhaps the *Microcontroller* approach described in the following chapter could be adapted in a feasible way.

3.5 Implementation Issues

When developing a distance learning lab, one must keep in mind that no manual operation has to be necessary. This fact especially concerns the following points:

- The logic analyzer: In our case, this tool is capable of connecting to a remote X-server. Unfortunately, the VLSI development software we use is Windows-based, as is the major part of this type of software (unfortunately, again). This fact forced us to use the *Cygwin* environment to establish an X-Server on a Windows machine and employ a connection to the logic analyzer.

- The fact to use *Windows* as operating system nevertheless reveals a rather elegant solution concerning the remote access of the workstations: we can simply use `rdesktop` (in the *Linux* environment) – or `mstsc` (in the *Windows* world). Both programs are free, so connectivity is no problem. It has to be stated, though, that a broadband Internet connection is strictly required to use distance learning in a meaningful way. This in fact is no restriction to generality, because there exist computer rooms, students can use at will. This implies their physical presence at the campus. It should also be stated, that local workplaces will still remain for those students who prefer to work in the lab.

- The images created by the VHDL design and modified / measured / tested by the students can easily be made visible using a simple USB webcam. The webcam's output can be viewed on the workstation's local screen and thus can be viewed from everywhere via the remote desktop application. More expensive, but also more elegantly, a framegrabber could replace the webcam, which does not need a mechanical construction to fasten it. Employing frame-grabbers is also one of our future prospects.

- Of special concern is the startup phase of each digital design. Due to the lack of physical accessibility of a (preferably denounced) reset button in case of remote learning, some alternatives guaranteeing a secure and deterministic system startup have to be provided. In our case, a special circuitry has been added to the design, a programmable power-on reset generator. This is a logic block activating the reset line for a certain amount of time after downloading the design. The active reset time itself depends on the type of FPGA used.

- Finally, it has to be ensured that besides providing accessibility from remote hosts, the network infrastructure must not be subjected to any kind of attack from the outside. This problem is solved by use of firewalls and rights management. The solution again creates new problems, e.g. the use of floating licenses for software packages.

3.6 Organization

Students attend the course in groups of three people. They can choose their partners freely as long as no conflicts arise. If no explicit request is uttered, the groups are formed randomly.

At the moment, supervision is done per schedule, four hours per workday. The remaining time is free and unscheduled; students can work as they want and as long as stations are available. Remote access is also possible; in either case, access is granted by the "first come, first serve" principle.

There still are plans to further stress remote working capabilities. Especially working people should be given the chance to obtain the same quality of teaching as students physically present. One way to do that would be, for example, to establish web-tutors or introduce a software based resource control system to allocate time slots and, in a broader sense, manage the course. The inherent single-user character of remote learning will in future make groups of students working on the same example obsolete.

4 Microcontroller

4.1 Overview

The *Microcontroller* course gives second-year computer engineering students their first impressions on microcontroller and hardware-near programming. We require basic electrical engineering knowledge, some C programming skills and basic knowledge of computer architecture. For most of our students this course is their first contact with hardware and Assembler programming.

The original course was first held in 2003 when the new bachelor study "Computer Engineering" was introduced at the Vienna University of Technology. Faced with the challenge of 100-150 students in an 8 workstation lab, the first course consisted of free lab access combined with free exercises, support using email and newsgroup, and a programming exam at the end. The course was subsequently modified the following years, experimenting with different mechanisms to balance the demands of the curriculum, the needs of our students and the constraints of available resources.

4.2 Course Structure, Grading and Organization

Basically, our course is built upon a mixture of programming and theory exams and both compulsory and voluntary exercises. The new course is divided into three parts. Part one is an introduction to microcontroller architecture, assembly programming and the programming environment. Part two teaches basic features as timers, analog module, interrupts, matrix keypad, and the numeric display. Finally, part three covers the advanced features like interfaces, motor control, and application programming. Both the second and the third part are done using C language.

Each part has a couple of compulsory exercises. These exercises cover the microcontroller features in rather simple tasks. The tasks are intentionally kept simple to prevent students to get the program working instead of getting it right. Students can additionally select voluntary tasks to deepen their knowledge. For those programs, additional points are rewarded depending on complexity of the task and quality of solution. Since there is no guarantee that students complete their tasks on their own, each part of the course is concluded by a two-tasks programming and a theory exam. For a passing grade, we require students to have at least two out of the six programming exams correct. For students who attend all three exams, the worst result is dropped. The final grade consists of all the points out of the programming and theory exams and the compulsory homework. Students can use their points out of voluntary homework to compensate some missed points in their exams.

The course uses a hierarchic system for student supervision to provide scalability and to minimize personnel costs. The course is managed by one assistant professor, one teaching assistant and one tutor per every 15 students[3]. Basically, we organize the course as an open lab – in fact, the lab is the students home. Additionally there is a supervised lab hour for every student once a week. Though attendance is not compulsory, it is highly recommended. Supervision is performed by tutors (senior students) who have 3-5 students per hour. Their task is to answer questions, help with debugging problems, have an open ear for students problems and keep up students motivation. Tutors also do the preliminary grading of both compulsory and voluntary homework. The teaching assistant checks for plagiarism and does the final grading. Furthermore, he is the first instance for tutors having questions or problems. The assistance professor sets up the course, creates the electronic course material, prepares programming and theory exams, and is responsible for final grading. In fact, most of this work is preparation, so after the course starts, the professors workload is pretty low.

4.3 Microcontroller Hardware

As we expect our students to work at home (or anywhere else) we have to get the lab to our students. Simulators like in [2] would have been very cost effective but the main drawback is the fact that simulators do not capture all problems that arise with real hardware [4]. Second, students gain a lot more confidence in using microcontrollers working hands-on instead of using simulators. Third our students have to do all the wiring themselves, again hands on – our components are not pre-wired like those in [5]. We learned our lesson in our first lab course in 2003 using a commercial HSC12 evaluation board with fixed connections, where we had a couple of students who did not understand the connection between a microcontroller pin and a LED. They just learned that writing 0x0F to port X turns on LED2.

[3] This number of tutors is prescribed by the Computer Science Faculty of the Vienna University of Technology

For our distance labs project we wanted to get the lab to our students. Instead of letting students remote-control our workstations or directly connecting to our targets, we decided for a different approach: we wanted to bring the lab to our students – physically. The main idea was using cheap hardware that is easy to reproduce and suitable for mass production. We did not find any suitable hardware on the market, so we developed our own hardware.

Main component is the interchangeable controller board that contains the microcontroller itself (an ATmega16), its supportive logic (power jack/regulator, reset button, programming connector, and oscillator), and I/O connectors as single pins. This approach enables the use of multiple controller boards together with the rest of our hardware. In fact we are currently working on three other boards using HSC12, ARM, and MSP430 processors to enable students to broaden their knowledge by working with more than just one processor during the course. The I/O board contains LEDs, switches, buttons, a matrix keypad, a numeric display, variable resistors, and a light sensor.

Most exercises can be done just using one of the main controller boards and the I/O board but for advanced projects, additional boards such as a motor board with dc- and stepper-motors or an interface board with serial, parallel, and CAN interface are available.

4.4 The Lab Kit

In order to get the lab to our students we introduced lab kits. A lab kit contains all the items needed during the *Microcontroller* course: hardware, software, and documentation (see Figure 3). We offer those lab kits for a deposit of 70 Euro, interested students can also buy the kit for exactly the same price. Each lab kit contains one controller and one I/O board, all wires needed, a power supply, a programming interface, a Knoppix CD and a short documentation.

The Knoppix CD plays a keyrole here – it contains a bootable *Linux* environment and all necessary software tools. Furthermore, a lot of documentation material is included. Other then the CD-ROM described in [6] that just contains learning material, the Knoppix CD enables students to start instantly with their exercises, without installing software or setting up their PC. All they have to do is insert the CD, reboot their computer, wire the boards, and load the demo program.

Figure 3: Contents of the lab kit

4.5 Exams and Grading

The lab kit allows our students to freely plan their "lab time". They only have to synchronize with us for exams and homework submission. Again, scalability and efficiency were main requirements.

Our programming exams consist of two tasks in two hours. The tasks are kept quite simple (at most 20 lines of code). To allow complex settings while keeping the actual task simple, students are provided a framework consisting of an C or Assembler skeleton, a makefile and object files. Students just have to add their code to complete the task.

Tutors are available to help students with hardware trouble. Tutors are also allowed to help with debugging problems if student can demonstrate that they understand the issues involved and just suffer from a blackout. After students finish the test, the programs are checked by the tutors. The main advantage of our programming exams is the fact, that students can try and debug their programs. They again work "hands-on". We use an "all-or-nothing" grading and honestly, students would not accept such with paper exams. Furthermore the results are instantly available, so there is no need for a later grading session.

The theory exams are set up as yes/no-questions instead of the traditional exam ques-

tions. We took this approach for the benefit of a fair grading scheme and the possibility of automatic grading. Before grading automatically we run a statistical evaluation of all exams. Questions with a high percentage of wrong or no answers are carefully reevaluated for errors or complexity and excluded (and thus not graded) from the test. This method uncovers errors and problematic questions in the database at a moderate level of effort making our exams more convenient for the students.

4.6 Anytime, Everywhere

The *Microcontroller* lab kits perfectly fit into the anytime, everwhere paradigm. Students can freely decide when and where to do their homework. In fact, we might even have students some thousand kilometers away. Homework and protocols can be submitted electronically, course materials including electronic slides are on the Knoppix CD. Two issues stand to be resolved: human support and trusted programming and theory exams.

Students at our university have their weekly supervised lab hour and additionally support via email and newsgroup. For students abroad we are currently preparing video tutors using software for voice/video communication over IP.

Problems with theory exams can be solved accordingly. All we need is a trusted person supervising the test abroad. Students log into our framework and can do the test remotely. The trusted person must not necessarily have any microcontroller knowledge as we once more can support the test using voice/video over IP.

In programming exams the support can be handled the same way. But afterwards our tutors check the correctness of the programs what trusted persons without microcontroller knowledge simply could not do. Therefore we developed a system that can carry out black-box tests to verify whether the embedded software running on a target system meets predefined requirements. This system consists of a special test board which is connected to the target system and software running on a host computer that assists in the generation of test-case descriptions and controls the test [7].

4.7 Experiences

We already deployed part of the lab kit in last summer term. Student feedback was very positive, over 70% of the students even kept the lab kit for private projects. Furthermore, we had the feeling, that the lab kit helped slower students to catch up and aquire the necessary programming skills. The high student satisfaction resulted in a good student retention rate of 70%. Out of those 70% of remaining students, 73% passed the course. Next summer term, the final version of the lab kit will be deployed in the *Microcontroller* course. We also have plans to conduct the course in a full distance learning setup far abroad with a small testing group to get experiences and feedback about ”trusted” exams and the automatic test system.

5 Embedded Systems Programming

5.1 Overview

The *Embedded Systems Programming* course addresses students in the 5th semester of the “Computer Engineering” bachelor study who have already completed the *Microcontroller* course. The laboratory course is held once every year and is attended by about 80 undergraduate students. It teaches various aspects of embedded systems programming in C programming language on 8-bit microcontrollers (Atmel AVR), such as accessing the built in hardware functions like timers, UARTs (Universal Asynchronous Receiver and Transmitter), analog/digital converters, access to peripheral hardware like motors displays, and the communication between several of these nodes using a real-time communication protocol.

5.2 Course Goals and Structure

The course consists of two parts: The laboratory part where groups of three students each solve three programming examples. Additionally there are two exams where the students answer theoretical questions and solve small programming tasks. To improve their grades, students can do bonus tasks.

Currently, 10 PCs in our laboratory are available for the *Embedded Systems Programming* course. Each group has about three hours per week where they can do their exercises under the supervision of a tutor. Additionally, students can also work on their tasks outside of these supervised hours.

5.3 Remote Targets

To improve the support for our students we are currently implementing a system that supports remote workplaces which we will deploy in a test run in the summer term of 2006. The system will consist of an extended version of our target boards that are able to capture the output of student-programmed microcontroller nodes and forwards it to a visualization software on the students computers. Furthermore, there is the software that handles the authentication of the students, the transmission of data between the students computer and the target board, and the development environment (C-compiler, libraries, debugger) on the client side.

The approach used in *Embedded Software Programming* is slightly different to the one used in the *Digital Design* lab. In *Digital Design* we deployed remote workplaces by using the normal workplace setup with PC, logic analyzer, and target hardware and installed additional remote desktop software to control the workstations over the Internet. This was the only option available for this course, since the hardware is very expensive and the needed software licences are bound to the PC.

The preconditions for the *Embedded Systems Programming* course allowed us to find an other solution:

- The *Embedded Systems Programming* target boards use relative cheap hardware (no expensive logic analyzer is needed) so that the number of remote workplaces can be easily increased.
- All software used is available under open-source licence. There is no legal problem to distribute the software among the students.
- More workplaces can be installed, because the number is no longer limited by the number of PCs since one server can handle multiple target boards.
- In *Digital Design* the same room capacity is needed for remote workplaces as for conventional ones – in fact, the same workplace is either used remote or local. With the *Embedded Systems Programming* remote target boards, lab space is no longer a limiting factor since there is no need for a per-target workstation or monitor, the target itself has a rather small form factor (about the size of a DIN-A4 page) and could be easily mounted in a rack and located in a server room.
- The setup of authentication server, target server and targets can be easier managed (update of software, account setup) than a bulk of workstations.
- The transfer of measurment values and debugging data uses less network bandwidth than the transfer of the whole desktop content.
- Forwarding of the desktop, as described in the *Digital Design* approach makes the internal network more vulnerable for security threats than the transfer of measured values and debugging streams.
- The *Embedded Systems Programming* remote workplace allow the sharing of the target between a group of users, enabling them to cooperate over the Internet or for a tutor to interactively help students with their programming tasks.

This shows that, if software licensing is not an issue (throught the use of open-source software) and the required hardware is relatively cheap (compared to logic analyzers) a more scaleable approach can be taken. This point is especially important, since thus we can handle increasing numbers of students with only small additional investment into more target boards.

5.4 Workplace Hardware

The remote workplace will feature the same type of microcontroller node, that is used in our current laboratory setup. The microcontroller node consists of an Atmel AVR ATmega128 microcontroller, an interface to a communication

bus, attached peripheral hardware, and a debugging interface. Several of these nodes (currently four) are connected through the communication bus and are programmed by the students. This setup is similar to the one used in our laboratory setup. Additional to these nodes, a second set of nodes is used to monitor the student-programmed nodes. The measured values (display content, motor speed, ...) are then transmitted via a gateway to the student PC were these values are visualized.

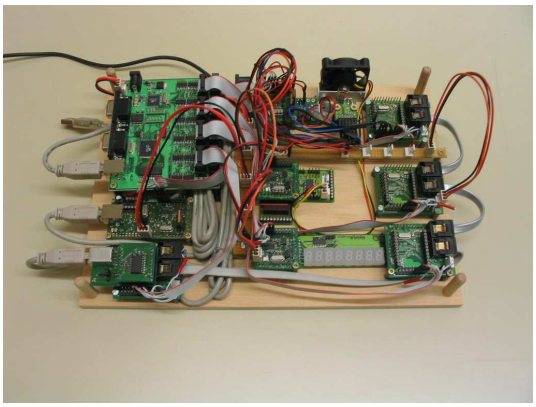

Figure 4: The remote workplace target board

The third component of a remote target board is the debugger board, that uses the JTAG interface of the nodes for in-circuit-debuging of the software on the nodes, one at a given time (see Figure 4).

5.5 *Authentication, Data Transmission, and Visualization*

The remote workplace software has to handle the following tasks:

- Authentification of students: Login requests have to be checked against the database.
- Assignment of time-slots: To assure each student a fair portion of the available target time, students can reserve target time-slots. Excess time is assigned according to the "first come - first served" principle.
- Data transmission: Various data (measured values, debugging information, and student programs) has to be transmitted over the Internet and dispatched to the assigned target.
- Visualisation of the target board: On the client side, the remote workplace software has to visualize the meaured values.

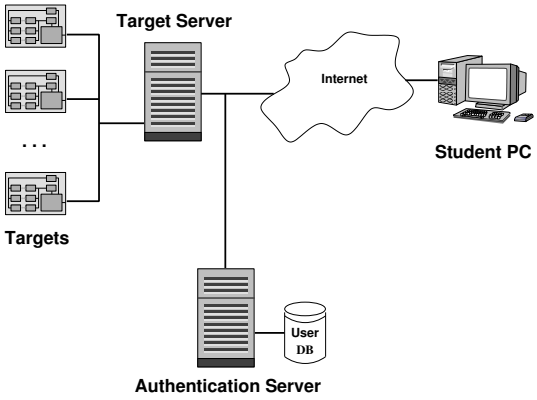

Figure 5: Overview of the remote workplace setup

Figure 5 shows the different parts of the remote workplace setup: The authentication server, the target server, the login client, and the visualization software.

The students starts a session by booting up the Knoppix CD. The local client software contacts the authentication server and – if the login succeeds – is assigned a target. Whenever a local program is started, it contacts the login client for the assigned session-cookie and uses this to connect to the target.

The target server will transmit all measured values from the target to the visualization client and will forward the GDB debugging stream.

The authentication server keeps track of all users and terminates the connection if the assigned time-slot elapses.

5.6 *Development Tools*

In our lab course we use the GNU toolchain, cross-compiled for the Atmel AVR target as development environment. The GNU Debugger (GDB) frontend named *insight* can be used to debug remote microcontroller nodes.

To relieve the students from the task of setting up the development environment themselves, all needed tools are supplied pre-configured on a bootable Knoppix CD simmilar to that used in the *Microcontroller* course. In addition to the development tools that are used on the *Microcontroller* CD, this CD will also include the ap-

plications that handle the login of the students onto a given target, the data transfer to the target, and visualization of the measured values. Additional all needed documentation (datasheets, manuals, lecture slides) will be bundled onto the CD.

The visualization software will display the measured values as a graphical visualization of the target board, where the hardware elements of the board are displayed in real-time as if the where filmed by a camera (e.g., the picture of display will show the currently measured values and the light bulb's brightness shown on the screen will correspond to the measured values).

6 Conclusion

With this paper, we described different approaches to enable remote learning without sacrificing the hands-on experience of designing and programming real hardware.

The advantages of the distance learning approach are manifold. First of all a much better resource utilization can be established. On the one hand this actually means that additional efforts have to be put into creating scheduling and reservation systems. On the other hand, students are provided a much more flexible working environment – especially in the time domain – while experiencing the same quality of education. Finally, after setting up such a system, the administrative overhead should be minimized. For example, no physical damage can be done to the lab equipment when it is only accessed remotely.

In our work we used two different solutions to support remote learning: Lab kits and remote workplaces. Lab kits are preferable if the hardware can be made cheap enought that a seperate kit can be provided to each student. For courses that are more hardware-intensive a remote workplace solution is preferable. Depending on the constraints of the course, different approches have to be taken.

Such a remote workplace can either be implemented by using remote desktop software to utilize the existing laboratory PCs over the Internet, or dedicated remote workplaces could be set up. In the case of the *Digital Design* course the expensive hardware and licence issues lead to the remote desktop solution. With the *Embedded Systems Programming* course we did not have such restrictions. Therefore we used a setup with a dedicated target server which improves the scalability.

References

[1] Wayne Wolf and Jan Madsen. Embedded systems education for the future. *Proceedings of the IEEE*, 88(1):23–30, January 2000.

[2] Mehdi Amirijoo, Aleksandra Tešanović, and Simin Nadjm-Tehrani. Raising motivation in real-time laboratories: The soccer scenario. In *35th SIGCSE Technical Symposium on Computer Science Education*, pages 265–269, March 2004.

[3] Jonathan Allen and Christopher J. Terman. An interactive learning environment for VLSI design. *Proceedings of the IEEE*, 88(1):72–80, January 2000.

[4] John W. McCormick. We've been working on the railroad: A laboratory for real-time embedded systems. In *36th SIGCSE Technical Symposium on Computer Science Education*, pages 530–534, February 23–27, 2005.

[5] Rafic Bachnak. Teaching microcontrollers with hands-on hardware experiments. *Journal of Computing Sciences in Colleges*, 20(4), April 2005.

[6] Jack M. Wilson and William C. Jennings. Studio courses: How information technology is changing the way we teach, on campus and off. *Proceedings of the IEEE*, 88(1):72–80, January 2000.

[7] Voin Legourski, Christian Trödhandl, and Bettina Weiss. A system for automatic testing of embedded software in undergraduate study exercises. In *Workshop on Embedded Systems Education (WESE'05)*, pages 44–51, September 22nd, 2005.