EMBEDDED SYSTEMS HOME EXPERIMENTATION

Wilfried Elmenreich
Institute of Computer Engineering
Vienna University of Technology
Treitlstrasse 3/182-1
Vienna, Austria
email: wil@vmars.tuwien.ac.at

Christian Trödhandl and Bettina Weiss
Institute of Computer Engineering
Vienna University of Technology
Treitlstrasse 3/182-2
Vienna, Austria
email: {troedhandl,bw}@ecs.tuwien.ac.at

ABSTRACT
To cope with increasing numbers of students in embedded courses, providing means for distant learning from the student's homeplace is an appealing idea. This paper[1] presents two approaches for experimentation with real embedded hardware at the student's homeplace. In the labkit approach, the students receive a media bag with hardware and software that can be used at their computer at home. In the remote workplace approach the students connect to a target board via a server on the Internet.

An evaluation of two case studies shows that both ways have their justification. Functional requirements like accessibility to wiring, Internet access, and tutor support have to be taken into account. Which approach is more economical depends mainly on the cost of a single board: if a single board is rather cheap a labkit approach is favorable; if a single board is more expensive the remote workplace approach is better.

KEY WORDS
embedded systems, distance learning, Knoppix.

1 Introduction

Embedded systems represents a growing segment in industry and domestic electronics, leading to an increased demand on education on this topic. Experiences in teaching embedded systems have shown that a complementary approach having students performing practical exercises additionally to theoretical courses is vital for effective learning [1, 2, 3].

In order to cope with increasing numbers of students in embedded courses, there are three possibilities: (i) increasing the number of workplaces in the lab, (ii) utilizing simulators instead of real embedded hardware, and (iii) providing means for distant learning from the student's homeplace. While the first approach reaches natural limits regarding room size and equipment cost, current experiences have shown that simulators cannot accurately model the intrinsic behavior of embedded systems like real-time constraints and second order effects of electric circuits [4]. The third method can be further distinguished into two cases: If the hardware is located physically at the university, it is necessary to provide a remote access interface to the system. This usually involves a web interfaces and remote instrumentation procedures. Examples for such systems can be found in [5, 6, 7].

If the hardware is relatively cheap and small it is more advantageous to produce embedded handout boards which can be taken home by the students for a deposit. This approach has also the advantage of preserving the possibility for hands-on exercises involving change of physical wiring among the target system components. However, this approach has the disadvantage that it is necessary to provide a full development system in software and hardware (except for the target board) at the student's PC at home.

This paper describes two case studies for experimentation with real embedded hardware at the student's homeplace comparing the two approaches. Current results show that both ways have their justification depending on the boundary conditions like accessibility to wiring, tutor support and economical parameters.

The remainder of the paper is structured as follows: The following Section 2 briefly describes the constraints and requirements for our case studies. Section 3 describes two courses that both deal with embedded hardware programming and provide different means for home experimentation. The two approaches are compared and evaluated in Section 4. The paper is concluded in Section 5.

2 Requirements for Embedded Systems Home Experimentation

In contrast to other more software-centered domains, like web programming or computer graphic applications, where the typical student is able to provide the necessary equipment, i. e., a desktop PC, at home, the field of embedded systems comes with more specific requirements.

Since many embedded systems come with specific development tools, it is required to provide depending on the target system a specific development software. Making this software available to the students confronts us with two

[1]This work is part of the "Seamless Campus: Distance Labs" project and received support from the Austrian "FIT-IT Embedded systems" initiative, funded by the Austrian Ministry for Traffic, 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.

problems:

First, the development system must not have a restrictive or expensive software license, since the students expect the development system to install easily and not to come with license costs. Even if the student is considered to be an expert in administration of his or her own computer, the more time it requires to set up the development system, the less time the student has left for the course itself.

Second, the software has to support the students' systems at home, which means different operating systems and different computer performance. Some development tools for writing the Flash memory of a microcontroller connected to the computer via RS232, USB, or Printer Port even require an older operating system like Windows 95/98, since direct access to computer interfaces has been made more restrictive in newer operating systems. Additionally, hardware drivers, e. g., for USB devices have to be provided for different versions of operating systems.

Thus, the environment for experiments at home should be easy to install, economical, and run on the majority of the students' computers.

3 Two Case Studies

In the following we describe the relevant properties of two embedded systems courses being held at the Vienna University of Technology.

3.1 Microcontroller Course

The Microcontroller course is an undergraduate course designed to give second year computer engineering students an introduction to microcontroller programming in C and Assembler.

The course consists of a theoretical part covering microcontroller architectures, I/O interfaces, and sensor/actuator peripherals and a practical part where students implement three sets of 3-4 short exercises. Motivated students are encouraged to improve their grade by submitting bonus exercises.

The target hardware consists of a board containing an ATmega16 microcontroller, its supportive logic (power jack/regulator, reset button, programming connector, and oscillator), and I/O connectors in single pin layout.

In order to enable the students to work at home, a labkit has been developed, which contains a controller board and an I/O board, a set of wires, a power supply, a programming interface, a CD, and a brief documentation (Figure 1). The labkit can be borrowed by students for a deposit of 70 Euro, interested students can also buy the kit for this price.

The included CD contains a bootable auto-configuring Linux system based on *Knoppix* [8] that supports various PC hardware (desktops as well as notebook types). The CD is self-contained, the user does not need to have Linux or any other Software pre-installed

Figure 1. Labkit for Microcontroller course

on his or her computer. As development software we use the GNU tool chain including the avr-gcc cross-compiler, the uisp programmer software, and the GNU debugger frontend *insight* to debug programs running on the microcontroller nodes.

Thus we have an environment that fulfills the above described requirements for experiments at home:

Operating system independence: Since the operating system is included with the labkit, the operating system installed on the user's system has no influence on the program compatibility.

Zero installation effort: The standard Knoppix edition has been extended by the necessary compiler, programming and debugging tools for the target system.

No software license costs: We strictly used freeware or open source software on the CD, thus we do not have to cope with software license costs. This way we also avoid tedious software registration procedures.

Thus, the enclosed CD enables students to start instantly working on their exercises and experiments. Since the software environment on the target system is also identical, many sources of configuration errors and problems of interference between different programs are avoided. The Knoppix Linux system comes with a standard graphical desktop system which is familiar also to users that have not worked on Linux before (see Figure 2).

3.2 Embedded Systems Programming Course

The Embedded Systems Programming (ESP) course is an undergraduate course designed to introduce third year computer engineering students to design and programming of distributed embedded computer systems. The course builds on the contents learned in the Microcontroller course.

The course consists of a theoretical part covering an introduction to data acquisition, communication, networks,

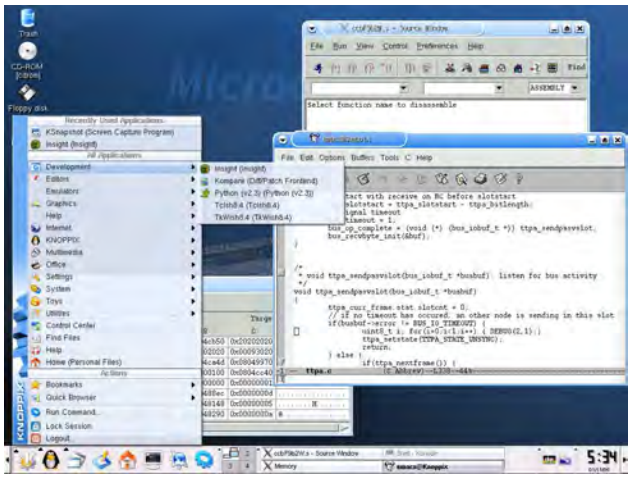

Figure 2. Development environment for Microcontroller course

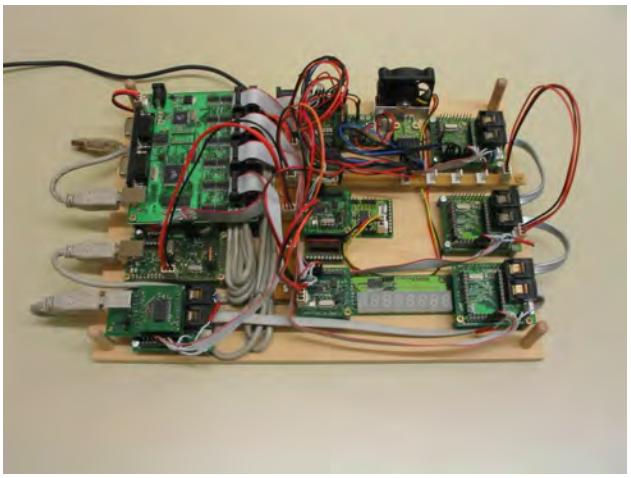

Figure 3. Remote workplace target board for ESP course

control theory and sensor data processing and a practical part where students implement three complex exercises (I/O, transducer network, and a feedback controller). Motivated students are encouraged to improve their grade by submitting a bonus exercise.

The target hardware consists of a board containing four microcontrollers, a programmer supporting subsequent programming of multiple controllers [9], a display, an electric fan with integrated rotation speed sensor, a light bulb, and sensors for luminance and temperature. Due to the relatively high cost of about 300 Euro, the hardware is not lent or sold to students.

In order to enable the students to do some of the work at home, we are planning to provide remote access to a subset of the boards. These remote boards are extended by a measurement framework that monitors the I/O of the target system. The measurement framework is based on the real-time communication network TTP/A [10] that enables an efficient transfer of periodic real-time data. The boards are connected via USB to a server that provides access to the development tools and the monitoring data from the target system. Figure 3 depicts such a remote workplace target board, which in addition to the four microcontrollers also contains the electronics for the measurement framework. With the extra electronics, the cost for a board is about 600 Euro.

As it is the case in the Microcontroller course, the development system will be brought to the user in form of a bootable Knoppix CD. The CD contains the development tools, a local client software, and visualization software. The students at home will start a session by booting the development system on their PC. The student then uses the client software to contact the authentication server, and – if the login succeeds – is connected to a free target.

To assure fairness, students can reserve target time. Otherwise, access is granted on a first come – first served principle.

The visualization software will connect to the target server via a secure TCP/IP connection and display the measured values in a graphical visualization of the target board (e. g., the light bulb's brightness shown on the screen will correspond to the measured values at the physical light bulb).

The system supports also remote group working, where other students can join the target in a read-only mode. The same mechanism can be used for tutoring, where a tutor joins a target system, if the student requests help.

4 Evaluation

The two case studies present two models for embedded systems home experimentation. Both enable the students to work with real hardware outside the laboratory. However, the question arises why we follow two different approaches instead of choosing the approach that is more economical in both cases.

Our main reason is that both approaches have their merits and drawbacks, and which version to favor depends on the goals that should be met. For instance, a significant advantage of take-home labkits is that students have physical access to the hardware. This might be advisable for beginner level courses to foster familiarity with actual hardware. Another major advantage compared to remote workplaces is that each student has his or her own target and can work in his or her own time. This implies that a particularly motivated student may do fun projects without taking valuable target time from other students. Similarly, a student who needs more time to grasp the concepts can work as much as necessary. Labkits also offer the possibility to let students buy their kits at the end of term. In that case, students have the opportunity to further develop their understanding and skills, and it may encourage more students to experiment and do their own projects on "their"

	Microcontroller	ESP
Lab workplaces	$8\times70 = 560$ Euro	$10\times300 = 3000$ Euro
Labkit	$56\times70 =$ **3920** Euro	$40\times300 =$ **12000** Euro
Remote Workplace	$7\times370 = 2590$ Euro	$5\times600 = 3000$ Euro
Server	3000 Euro	3000 Euro
Total	**5590** Euro	**6000** Euro

Table 1. Hardware costs for labkit and remote workplace approach

boards even during the term, which will improve retention and grades.

Similarly, there are good reasons to favor remote workplaces over labkits. For instance, since we expect about 120 students, it is certainly a lot less time-consuming to produce and test 20 remote workplaces than to provide 120 labkits. Furthermore, since there is no physical access to the remote workplaces, they remain in prime condition for years, whereas labkits will show wear and will need to be replaced eventually. Other reasons to prefer remote workplaces, especially in more advanced courses, are that they can contain more expensive and elaborate hardware, only have moderate constraints with respect to their size and weight, and need not be particularly robust and foolproof.

The reason for the labkit approach is that in the Microcontroller course one teaching goal is that the students should also do the wiring of the target hardware, thus requiring physical access to the board.

An argument in favor of the remote workplace approach used in ESP is that the cost for a labkit with an ESP board is quite considerable for a student's budget.

Moreover, when considering the total hardware costs, both approaches can be justified: Currently, we have 8 Microcontroller workplaces and 10 ESP workplaces in the laboratory. Our goal is to teach 120 students each in both courses. We assume that one laboratory workplace can be timeshared by 8 students. Thus, we need to get an extra capacity for 56 and 40 students, respectively, outside the lab rooms.

In the case of the Microcontroller course, this is achieved by providing extra boards for each student who wants to work at home, thus hardware costs for this approach are 56×70 Euro for the labkits, making a total of 3920 Euro.

Using the remote workplace approach, we would need only 7 extra boards (since those are timeshared as well), but circuitry for remote monitoring would approximately cost another 300 Euro per board. Moreover, there are the costs for the web server and infrastructure of about 3000 Euro, making a total of 5590 Euro. Thus, for the Microcontroller course, the remote workplace approach would be more expensive than the labkit approach.

In the case of ESP, the remote workplace approach requires costs for 5 remote workplaces with circuitry for remote monitoring (5×600 Euro) and the costs for the web server and infrastructure of about 3000 Euro, making a total of 6000 Euro.

If we chose the labkit approach for ESP, the costs for 40 labkits will be $40\times300 = 12000$ Euro, thus, for ESP the remote workplace approach is more economical. Table 1 shows a comparison of the calculations.

When considering costs, it is also prudent to think about the cost of changes in the hardware. Clearly, a major redesign in the hardware that invalidates the existing system is costly in both approaches, with the labkit approach being more costly according to Table 1, although the scales will dip in favor of the labkit if enough students buy the kit. If the new design can be used in parallel with the existing one, the labkit approach is clearly superior since every sold old labkit can simply be replaced with a new one.

5 Conclusion

We have presented two approaches for experimentation with real embedded hardware at the student's homeplace. In the labkit approach, the students receive a media bag with hardware and software that can be used at their PCs at home. In the remote workplace approach the students connect to a target board via a server on the Internet. In both cases, the software environment for the development system is shipped on a CD with the autonomous Knoppix Linux system. Thus, there is no installation procedure for the user. When comparing the two approaches, each has advantages and disadvantages. The possibility to directly manipulate the hardware (for example to change the wiring) can be a desirable or unwelcome feature. In contrast to the labkit approach the remote workplace approach requires Internet access to work, but it enables a tutor to directly monitor and assist the student's work.

Remote workplaces can be more elaborate, and thus lend themselves to advanced courses with expensive hardware, whereas the hands-on aspect of simple labkits is particularly suited for beginner courses.

Which approach is more economical depends mainly on the cost of a single board: If a single board is rather cheap as it is the case in the Microcontroller course, a labkit approach is favorable; if a single board is more expensive as it is the case in the Embedded Systems Programming course, the remote workplace approach is better.

References

[1] D. Beetner, H. Pottinger, and K. Mitchell. Laboratories teaching concepts in microcontrollers and hardware-software co-design. In *Proceedings of the 30th Annual Frontiers in Education Conference*, volume 2, pages S1C/1–S1C/5, Kansas City, Missouri, USA, October 2000.

[2] R. Bachnak. Teaching microcontrollers with hands-on hardware experiments. *Journal of Computing Sciences in Colleges*, 20(4):207–213, 2005.

[3] M. J. Callaghan, J. Harkin, C. Peters, T. M. McGinnity, and L.P. Maguire. A collaborative environment for remote experimentation. In *Proceedings of the 2003 IEEE International Conference on Microelectronic Systems Education (MSE'03)*, 2003.

[4] D. Gillet, C. Salzmann, H.A. Latchman, and O.D. Crisalle. Advances in remote experimentation. In *Proceedings of the 19th American Control Conference (ACC00)*, Chicago, USA, June 2000.

[5] G. Mužak, I. Čavrak, and Mario Žagar. The virtual laboratory project. In *Proceedings of the 22th International Conference on Information Technology Interfaces*, pages 241–246, Pula, Croatia, June 2000.

[6] F. J. Genzález-Castañoa, L. Anido-Rifón, J. Vales-Alonso, M. J. Fernández-Iglesias, M. Llamas Nistal, P. Rodríguez-Hernández, and J. M. Pousada-Carballo. Internet access to real equipment at computer architecture laboratories using the Java/CORBA paradigm. *Computers & Education*, 36(2):151–170, February 2001.

[7] M. J. Callaghan, J. Harkin, T. M. McGinnity, and L.P. Maguire. An internet-based methodology for remotely accesses embedded systems. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, October 2002.

[8] K. Knopper. Building a self-contained auto-configuring Linux system on an iso9660 filesystem. In *Proceedings of the 4th Annual Linux Showcase & Conference*, Atlanta, Georgia, USA, October 2000. USENIX Association.

[9] W. Haidinger. A tool for programming AVR microcontroller networks – the ZEUS programmer. Research Report 51/2002, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, 2002.

[10] H. Kopetz et al. Specification of the TTP/A protocol. Research Report 61/2002, Technische Universität Wien, Institut für Technische Informatik, Vienna, Austria, September 2002. Version 2.00.