Remote Target Monitoring in Embedded Systems Lab Courses using a Sensor Network

Christian Trödhandl, Markus Proske, and Wilfried Elmenreich
Vienna University of Technology,
Institute of Computer Engineering,
Treitlstrasse 3, 1040 Vienna, Austria
{troedhandl,proske}@ecs.tuwien.ac.at, wil@vmars.tuwien.ac.at

Research Report 93/2006

Abstract – *This paper describes an architecture for remote monitoring of a distributed embedded system via Internet. The data at the target system is gathered with a time-triggered sensor network which transmits the measured values to a local target server. The sensor network approach makes the system easily adaptable to different embedded target systems.*

The sensor network is connected to a target server that communicates via Internet with visualization and programming tools at the monitoring computer. The visualization clients provide a live display of the parameters of the observed system.

The target server acts as a gateway between target system and monitoring clients and provides security and authentication features for connecting monitoring clients. One target server is able to serve for multiple target systems.

As a case study, the presented system will be used in an embedded systems lab course where students are requested to implement various applications on an embedded target board. Using the remote monitoring feature, a student is able to do the work from his or her home place.

1 Introduction

The increased use of embedded applications in the domains of automation, process control, transportation systems, ubiquitous computing, etc. calls for means to monitor and control the embedded system without the need to be physically at the place of the observed system. We will follow a remote monitoring approach using Internet technologies as it is used in applications in the automation domain [1, 2, 3, 4] and in railway transportation systems [5].

Typically, the monitored target system has real-time properties, which puts real-time requirements onto the monitoring system. When also considering aspects like flexibility, extensibility and, since the monitored data is usually transmitted over the Internet, security, the design of the monitoring system becomes a challenging and difficult task.

This paper presents a distributed remote monitoring system based on a real-time fieldbus network, an Internet server, and a data visualization system on the user's PC. The real-time fieldbus network interconnects several smart transducers that periodically gather measurements from the process variables of interest. The fieldbus approach supports the flexible adaption of the system to different target systems. The resulting data is forwarded to a target server that communicates the data via Internet to the user's PC where a dedicated visualization software displays the actual system state.

As case study, we present a distant learning application which enables students to implement and test embedded software on real embedded systems hardware. The case study supports the parallel monitoring of several target boards which are connected to a target server. An authentication server takes care that students can access only their admitted target boards. For the communication between the target server and the visualization and programming tools at the students PC, we have developed the so-called Remote Workplace Protocol (RWP).

The remaining parts of this paper are organized in the following way: Section 2 elaborates general requirements for remote monitoring of embedded applications. Section 3 describes the monitored system used in the case study. Section 4 presents the time-triggered fieldbus approach used to do the real-time gathering of the monitored data. The authentication and data transmission and visualization is described in Section 5. Section 6 describes the local client system. Section 7 describes how remote monitoring will be used in our embedded systems lab-courses. Section 8 briefly discusses related approaches providing remote access to embedded systems hardware. Section 9 concludes the paper.

2 General Requirements for Remote Monitoring

A typical remote monitoring application comes with several requirements in real-time capabilities, architectural requirements, and security/collaboration issues:

Firm real-time support: In order to get sufficient information, it is necessary to periodically and regularly sample the real-time variables of the target system with appropriate speed. In theory we need at least Nyquist frequency [6], that is the double of the highest frequency in the monitored signal, to reconstruct the signal dynamics. In practice an oversampling of about ten times of the signal with highest frequency is convenient.

Synchronized snapshots: The time difference between two concurrent measurements of different variables should be minimal in order to receive consistent snapshot views of the target system.

No interference at the target system: The monitoring system must not interfere in the functionality of the measured system. Note that in an embedded system such a probe effect [7] can arise from inserted monitoring code as well as from active measurement methods (e. g., ultrasonic sensors emitting a scanning signal), heat dissipation, and shared resources such as common power supplies or, especially in wireless system, shared bandwidth of the communication system.

Bandwidth: The bandwidth of the fieldbus system and the Internet stream must be sufficient in order to transmit the monitored data.

Temporal accuracy: The freshness of the data must be provided at the user's PC, especially when the user is intended to perform feedback actions on the observed system.

Target system flexibility: The remote monitoring system should be adaptable to different target applications without the need to perform changes throughout the whole monitoring architecture.

Target system extensibility: The number of components in a typical embedded applications is likely to increase, therefore the system should support monitoring of large embedded systems with a high number of process variables. Moreover, it should be possible to use the system concurrently for multiple processes.

Flexible client system: Since the idea of remote monitoring is to enable the access from an arbitrary place in the Internet, it would be counterproductive to require an extensive setup of the specific visualization and communication software at the clients computer.

Secure access: We require a save authentication and data transmission for the monitoring session in order to avoid interception or interference of data from an outside attacker.

3 Target System

Our target system to be monitored contains four 8-bit microcontrollers (Atmel AVR ATmega128) which instrument a display, a small light bulb, a photo sensor, a temperature sensor and a small cooling fan. Furthermore, the nodes are interconnected by an ISO k-line communication bus. Figure 1 shows a target board with microcontroller nodes, add-on hardware, JTAG debugging interface board, and measurement network. Unlike standard debugging boards, our JTAG debugging board was especially developed in order to support an electronic switching between multiple target processors without the need to locally re-plug the debugging cables.

Thus, the target system contains the following data sources to be monitored:

Seven-segment display: The seven-segment display consists of 8 separate seven-segment digits which are refreshed with a frequency of 100 Hz. A single digit has 8 connections for the cathodes of the LEDs (7 segments and one decimal point) and one for the common anode. The multiplexing is done by applying supply voltage to one of the eight anodes and thereby activating the corresponding digit. A segment is lit if supply

Table 1: Sensor inputs and expected data rate

I/O	Sampling frequency	data size	req. Bandwidth
7 segment display	100 Hz	24 bytes	2 400 bytes/s
Light bulb input	50 Hz	3 bytes	150 bytes/s
Photo sensor	50 Hz	1 byte	50 bytes/s
Temperature sensor	50 Hz	1 byte	50 bytes/s
Motor input voltage	50 Hz	1 byte	50 bytes/s
Motor rotation speed	160 Hz	2 bytes	320 bytes/s
ISO k-line signal (compressed)	–	–	10 000 bytes/s

voltage is applied to the common anode and the cathode is pulled to ground. The current content of the display can therefore be stored in an array of 8 bytes. A second array can be used to store the activation time of each digit to represent the brightness of each digit.

Light bulb input signal: The brightness of the light bulb is controlled by pulse-width modulation (PWM). The interesting parameters for PWM are the signal frequency (two bytes) and the duty cycle percentage (one byte). Since the light bulb reacts rather slowly to the control signal, a sampling frequency of 50 Hz is sufficient for monitoring the light bulb signal.

Photo sensor: The photo sensor converts the light emitted by the light bulb to an analog voltage. This voltage is digitized to an 8 bit value. A sampling frequency of 50 Hz is sufficient for monitoring the photo sensor.

Temperature sensor: The temperature of the light bulb is measured by a temperature sensor and also converted from an analog voltage to an 8 bit value. A sampling frequency of 50 Hz is sufficient for monitoring the temperature sensor.

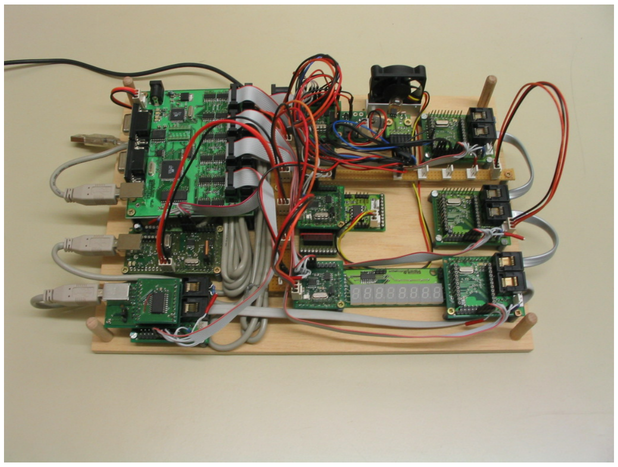


Figure 1: The remote workplace target board

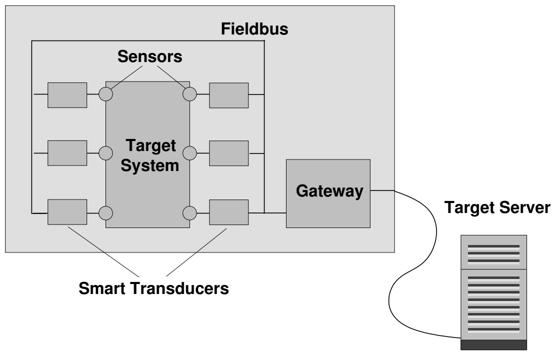


Figure 2: Monitoring network system

Motor input voltage: The supply voltage of the fan motor is measured by an 8bit analog/digital converter. We have specified a sampling frequency of 50 Hz for the monitoring the supply voltage.

Motor rotation speed: The motor supplies a rotation speed signal with one impulse per revolution. The rotation speed of the motor can be controlled in a range of 580 to 9600 revolutions per minute, thus, the expected maximum data rate if one assumes a 2 byte integer for the speed is 320 bytes/sec.

ISO k-line bus: The four microcontrollers used in our target system are interconnected by an ISO k-line bus. The bus communication will be compressed by run-length encoding, the compressed signal is expected to require a bandwidth of about 10 kbytes/s. Different from the other sensor inputs, the ISO k-line signal is captured by

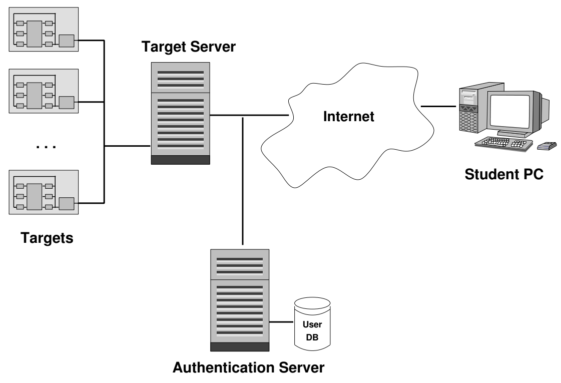


Figure 3: Overview of the remote workplace setup

the gateway node, thus, it does need to be transmitted via the fieldbus network.

Table 1 summarizes the expected data rates for the different sensor inputs.

The measured values (display content, motor speed, ...) are transmitted via the gateway to the target server by a USB connection. The target server transmits the data via Internet to the monitoring PC, where the current system state is visualized.

A first idea was to apply a digital video camera viewing the target board that streams the information over the Internet. While this approach would be rather generic (changing or extending the target system would not require a change in the video system set up), we decided to use a dedicated measuring approach for the following two reasons:

- The overall data rate in our application is about 13 kbytes/sec — far less then the usual data rate for a video stream. Thus, the transfer of measurement values and debugging data uses less network bandwidth than the transfer of a video stream.

- The camera would not show all information of interest. Due to frame rate restrictions, fast changes on the display or motor would not be displayed properly. Furthermore, some data sources do not have a visual feedback at the target board like the temperature sensor or the state on the communication bus (this would require to connect an oscilloscope and other measurement devices for visualization).

Thus, we decided to use a specific measurement system that collects the data of interest.

4 Monitoring Fieldbus Network

The monitoring fieldbus system interconnects several *smart transducers*. A smart transducer is the integration of an analog or digital sensor or actuator element, a processing unit, and a communication interface. In case of a sensor, the smart transducer transforms the raw sensor signal to a standardized digital representation, checks and calibrates the signal, and transmits this digital signal to its users via a standardized communication protocol [8].

Each smart transducer periodically performs measurements on several target system variables and transmits these to a gateway that is connected to the target server via a USB interface. The smart transducers are interconnected via a time-triggered TTP/A [9] network. The time-triggered approach uses a global schedule for all communication, computation, and action schedules. All nodes are synchronized to a global time which enables synchronized actions, as for example triggering of measurements and a predictable real-time data transfer using a predefined conflict-free TDMA scheme.

Figure 2 depicts the monitoring network architecture. The collected data is transmitted from the smart transducers to a gateway node that forwards the data to the target server. Note that our architecture supports multiple monitoring networks per target server as indicated in Figure 3.

5 Authentication and Data Transmission

The remote workplace software has to handle the following tasks:

- Authentication of students: Login requests have to be checked in the database.

- Assignment of time-slots: To assure each student a fair portion of the available target time, students can reserve target time. Without reservation, access time is assigned according to the “first come - first serve” principle.

- Data transmission: Measured values and debugging information have to be transmitted over the Internet and dispatched to the assigned target. Furthermore the students must be able to upload their programs to the target.

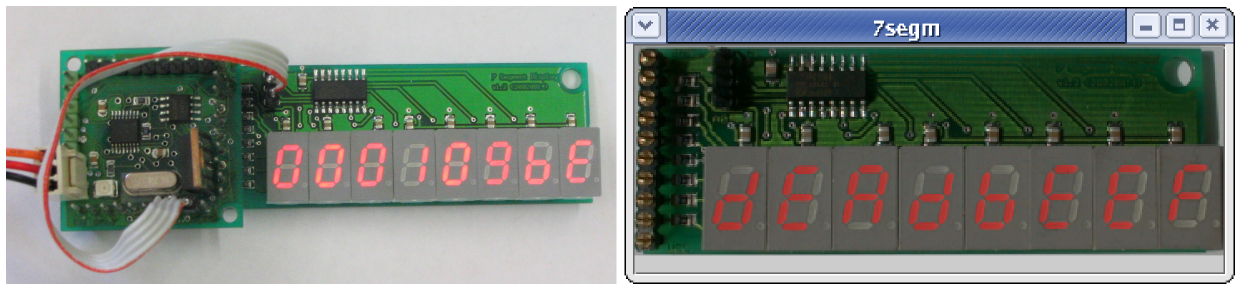


Figure 4: Microcontroller node with attached seven-segment display and visualization of a display module

- Visualization of the target board: On the client side, the remote workplace software has to visualize the state of the target system.

Figure 3 shows the different parts of the remote workplace setup: The authentication server, the target server, the login client, and the visualization software.

For the communication between authentication server, target server, and local client software we developed the RWP protocol. Requests and responses between the programs are encoded using XML, optionally followed by a binary data stream which is separated from the XML part by a single NUL character. First the client software initializes a connection to the authentication server. If the authentication succeeds, a target is assigned to the student and an authentication cookie is transmitted back. This cookie is used by the visualization software to set up data connections to the target server. If the assigned time-slot of the student expires the session is closed.

The RWP protocol builds the concept of so-called endpoints – virtual sensors that form a hierarchy (similar to the directory hierarchy in a computer filesystem) that is later mapped onto the hardware by the target server. Since the RWP protocol is independent of the underlying sensor hardware, different types of interfaces can be easily integrated by adding a new driver module to the target server.

6 Local Client Software

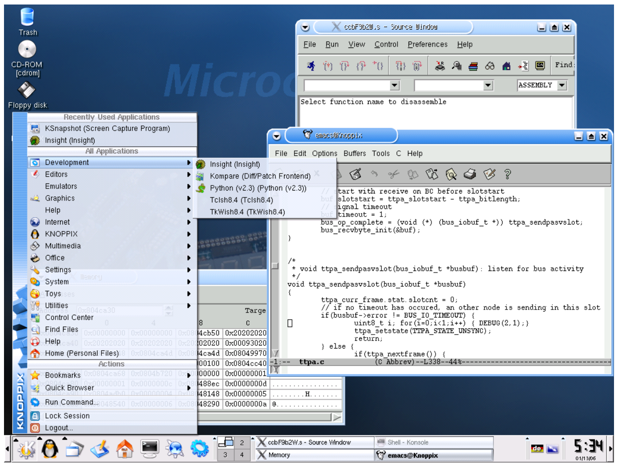
For our remote workplace setup we needed to get two things to the students: The development environment and the visualization software. Our development toolchain is based on the Free Software Foundation GNU tools in order to be able to distribute the client system without costs for software licenses.

The client set-up contains the following software: The cross-compiler *avr-gcc* (the normal gcc compiler set up for cross-compilation for Atmel AVR 8-bit microcontrollers), the automation tool *make*, and the GNU debugger *gdb* with the graphical frontend *insight*.

Additionally, a visualization system that provides a virtual dashboard [10] to allow the students to remotely monitor the current state of the target board is included within the client software. Using a web browser as visualization client at the user's PC is a appealing approach, however experiences with remote monitoring applications running as Java applets have shown that standard settings of communication and access rights for Java applets in typical web browsers are very restrictive and hinder the communication with local software (e. g., debuggers). Therefore, the visualization software runs as a stand-alone Java program.

All the described software packages are available for several operating systems like Windows, Linux, and Mac OS X. However, our lab course requires to establish about 120 students with the correct software setup. In order to avoid problems with heterogenous environment and system software, we have decided to provide all the necessary software on a bootable CD featuring the auto-configuring Linux system based on *Knoppix* [11]. Knoppix supports various PC hardware (desktops as well as notebook types). Knoppix users do not need to have Linux or any other Software pre-installed on the user's computer. The Knoppix system fully runs from CD and does not install any software on the user's computer. However, it supports access to local harddisk in order to store the user's private files.

The Knoppix system we use has been customized in order to provide our development toolchain and the visualization software, which is tailored to visualize the state of the target system. After booting the Knoppix environment, the user starts a session by running the local client software. The software contacts the authentication server and – if the login


ccbF9b2W.s - Source Window
File Run View Control Preferences Help
ASSEMBLY
emacs@Knoppix
File Edit Options Buffers Tools C Help
Insight (Insight)
Kompare (Diff/Patch Frontend)
Python (v2.3) (Python (v2.3))
Tclsh8.4 (Tclsh8.4)
TkWish8.4 (TkWish8.4)
Shell - Konsole
Memory

```
Select function name to disassemble
```

```
        // start with receive on BC before slotstart
        buf.slotstart = ttpa_slotstart - ttpa_bitlength;
        // signal 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 5: Snapshot of the development environment on a customized Knoppix CD

succeeds – is connected to a free 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.

All connections between the local client software and the authentication- and target servers are accomplished with the RWP protocol, using secure socket layer (SSL) connections to achieve a secure communication channel between client and server. The target server will transmit all measured values from the target to the visualization client and will forward the GDB debugging stream which enables the students to debug their programs over the Internet. The authentication server keeps track of all user connections and terminates the session if the assigned time-slot elapses.

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 similar as if was filmed by a camera (e.g., the picture of the display will show the currently measured values and the light bulb's brightness shown on the screen will correspond to the measured values).

7 Application Scenario

We are planning to deploy a remote workplace setup that uses a graphical visualization of the target boards in the "Embedded Systems Design and Programming" course in the winter term 2006/07. 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. In the practical part of the course, the students have to implement three exercises like using a multiplexed seven-segment display, controlling the speed of a fan, or measuring the brightness and temperature of a light bulb on an embedded system, consisting of four 8-bit Atmel AVR microcontrollers with 128 kbytes program and 4 kbytes data memory that are connected through a fieldbus network.

Students that want to use a remote workplace are supplied with a customized Knoppix CD that includes a full, pre configured development environment for our remote target boards. To start a remote workplace session, the student inserts the Knoppix CD into his/her computer and reboots the system. On start-up the Knoppix system establishes an Internet connection using the standard DHCP protocol.

The student starts up the login client program that contacts the target server using the account data that the student received at the course registration. If the login succeeds the local client software is connected to a free target and the student can start with his or her programming tasks.

The course programs are written and compiled locally and then sent to the remote target system. Figure 5 shows the desktop of a customized Knoppix CD with various development tools.

For debugging the programs the GNU Debugger frontend *insight* is used that communicates with the target server using the GDB protocol. Visualization clients are used to monitor the physical outputs of the tested software (see Figure 4 for a prototype visualization of a seven-segment display). If a course example is finished, it can be submitted electronically over the Internet. The remote workplace session either ends if the student logs out or if the assigned time-slot elapses.

8 Related Work

At the University of Technology in Sydney, Moulton et al. have developed an embedded systems lab with remote access [12]. Their target system is equipped with a master server providing access to the development board and a camera server monitoring the target board. Master and camera server are accessible remotely via Internet. The students' computer require an SSH (Secure Shell) client and a web browser.

Tzafestas et al. describe a remote robotic laboratory featuring an industrial robotic manipulator and a camera accessible via a web-based graphical interface connected to a robot and video Internet server [13].

The NetLab approach [14] at the University of South Australia provides remote access to measurement equipment that is interfaced via IEEE 4888.2, a standard interface to measurement instruments. Using a LabView client, students are able to access the instruments via Internet in order to gather measurement data. Additionally, a camera provides visual access to the set-up. However, since the target system is neither a programmable microcontroller nor a remote configurable hardware, the possibilities of interactions are limited.

González-Castaño et al. describe a remote lab featuring target boards accessible through the Internet via CORBA [15]. While in this approach students work with real hardware, there is not much interaction with a physical environment except for a module that allows students to remotely press physical buttons. The module is connected directly to the target server via the printer parallel interface.

Callaghan et al. use a remote desktop approach to access a PC with connected measurement hardware interfaced by IEEE 4888.2 and various target systems such as Microcontrollers, Field-programmable gate array, Digital signal processors, etc., which are interfaced by an RS232 connection [16]. While the remote desktop approach easily establishes a remote interface, the approach is resource-demanding since each working student monopolizes one workstation during experiments.

The Virtual Laboratory project [17, 18] at the University of Zagreb employs an architecture that is similar to our proposed approach. The architecture specifies a CAN network that connects two C167 development kit to an Internet server. One C167 board acts as development board for the students while the other one is used to monitor the effects on the physical process environment.

The Internet server provides a web interface for lab time reservation, gives access to the development board, and forwards the data from the monitoring node. The students' computer are running a Java visualization client that displays the current state of the target system.

Besides technical differences regarding the type of fieldbus network and employed web techniques, the main difference to our approach is that the target system in the virtual lab is a single computer instead of a distributed system.

9 Conclusion

We have presented a remote monitoring architecture on the example of a distant learning application. The system consists of a monitoring network system of networked smart transducers that collect data about the target system. The data is made available on the Internet via a target server using our RWP protocol. The software on the client computer visualizes the current state of the target system. In order to be independent of the client computer's software setup, we propose a self-contained Knoppix system that contains all the necessary communication and visualization software.

The general properties of our architecture make the approach also interesting for different applications where real-time monitoring, flexibility with respect to the target and the client systems and authenticated access is required. The used software is fully either open source or developed by our group so that it is easily possible to extend the course size or set-up the presented system at other universities without licensing costs. The remote workplace will be used in the course "Embedded Systems Design and Programming" in autumn 2006 at the Vienna University of Technology.

Acknowledgments

This work was supported by the SCDL project (`http://www.ecs.tuwien.ac.at/Projects/SCDL/`) from the Austrian "FIT-IT Embedded systems" initiative, funded by the BMVIT and managed by FFG (grant 808210).

We would like to thank Leo Mayerhofer for giving technical advice on computer interfaces.

References

[1] A. Weaver, J. Luo, and X. Zhang, "Monitoring and control using the internet and java," in *Proceedings of the 25th Annual Conference of the IEEE Industrial Electronics Society (IECON'99)*, vol. 3, 1999, pp. 1152–1158.

[2] M. P. de Albuquerque and E. Lelievre-Berna, "Remote monitoring over the internet," *Nuclear Instruments & Methods in Physics Research*, vol. 412, no. 1, pp. 140–145, 1998.

[3] K. Kusunoki, I. Imai, H. Ohtani, T. Nakakawaji, M. Ohshima, and K. Ushijima, "A corba-based

remote monitoring system for factory automation," in *Proceedings of the First International Symposium on Object-Oriented Real-time Distributed Computing*, Kyoto, Japan, Apr. 1998.

[4] F. Olken, H. A. Jacobsen, C. McParland, M. A. Piette, and M. F. Anderson, "Objects lessons learned from a distributed system for remote building monitoring and operation," in *Proceedings of the Conference on Object-oriented Programming, Systems, Languages and Applications*, Vancouver, Canada, Oct. 1998.

[5] T. Nieva, A. Fabri, and A. Wegmann, "Remote monitoring of railway equipment using internet technologies," Faculté I&C, School of Computer and Communication Sciences, Tech. Rep. Publication ID:200118, 2001, available at icwww.epfl.ch/publications/documents/IC_TECH_REPORT_200118.pdf.

[6] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the A.I.E.E.*, vol. 47, pp. 617–644, Feb. 1928.

[7] C. E. McDowell and D. P. Helmbold, "Debugging concurrent programs," *ACM Computing Surveys*, vol. 21, no. 4, pp. 593–622, Dec. 1989. [Online]. Available: http://www.acm.org/pubs/toc/Abstracts/0360-0300/76897.html

[8] H. Kopetz, M. Holzmann, and W. Elmenreich, "A universal smart transducer interface: TTP/A," *International Journal of Computer System Science & Engineering*, vol. 16, no. 2, pp. 71–77, Mar. 2001.

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

[10] W. Elmenreich, C. Trdhandl, and M. Proske, "Virtual dashboard specification," Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, Research Report 76/2006, 2006.

[11] 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: USENIX Association, Oct. 2000.

[12] B. D. Moulton, V. L. Lasky, and S. J. Murray, "The development of an enviroment for remote embedded systems: Feedback from students and subsequent enhancements," *World Transactions on Engineering and Technology Education*, vol. 2, no. 1, pp. 65–68, 2003.

[13] C. S. Tzafestas, N. Palaiologou, and M. Alifragis, "Virtual and remote robotic laboratory: Comparative experimental evaluation," *IEEE Transactions on Education*, vol. 49, no. 3, pp. 360–369, Aug. 2006.

[14] Z. Nedic, J. Machotka, and A. Nafalsk, "Remote laboratories versus virtual and real laboratories," in *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference*, Boulder, CO, USA, Nov. 2003, pp. T3E1–6.

[15] F. J. González-Castaña, L. Anido-Rifón, J. Vales-Alonso, M. J. Fernández-Iglesias, M. L. 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*, vol. 36, no. 2, pp. 151–170, Feb. 2001.

[16] M. J. Callaghan, J. Harkin, T. M. McGinnity, and L. Maguire, "An internet-based methodology for remotely accesses embedded systems," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, Oct. 2002.

[17] G. Mužak, I. Čavrak, and M. Žagar, "The virtual laboratory project," in *Proceedings of the 22th International Conference on Information Technology Interfaces*, Pula, Croatia, June 2000, pp. 241–246.

[18] N. Perić and I. Petrović, "Virtual laboratory for automatic control and supervision - challenges and opportunities," in *Proceedings of UN-ECE International Conference on Technology Transfer for Economic Development*, Zagreb, Croatia, June 2000.