Design and Implementation of a Framework for Remotely Accessible Instruments

Bernhard Fechner

Department of VLSI and Parallel Computing
FernUniversität Hagen
Bernhard.Fechner@fernuni-hagen.de

Abstract

Remotely accessible instruments and experiments enhance the experience of students by providing a remote interface to a physical device. Photorealistic rendering of these devices makes the graphical interfaces easy to understand, to use and to learn. Students perform their exercises at home just as in a real lab so they can apply their experience directly when working on location. Needless to say: remote labs are a key to raise the student's interest and improve self-paced learning. The implementation of a generic remote control interface for existing instruments and the acquisition and visualization of data is one milestone on the road to the future in distance learning. To avoid obstacles on this way, the system architecture should be understandable, modular and easy to extend, so laboratories with different hardware could be integrated in the lab environment. The client software should be platform independent so the users can quickly gain hands-on-experience because soft- and hardware problems (e.g. installations) are of no concern. We report experience in developing such a software framework and review some of its capabilities by using an actual deployment as an example.

Keywords: Software design, distance learning, remote laboratories, remote control

1 Introduction

The need for multimedia technology in distance teaching and learning is being recognized worldwide. Network-based multimedia and computer-related technologies improve and enhance the quality of teaching and learning. Colleges, universities and distance-learning institutions are faced with a growing number of students particularly at introductory levels, urging to learn more. In Hagen, undergraduate courses in computer science serve a large amount of students (1049 in 2005) with various backgrounds and abilities. The challenge is to motivate and excite these students so that each performs to their fullest potential. This objective can be best achieved by projecting the lab environment in a web-based, multimedia technology because students learn science best by experimenting, gaining hands-on experience, raising questions and by solving problems. As the instruments in the real lab can be rather fragile and expensive, remote labs are a reasonable low-cost solution. Just like their real equivalents, remote instruments respond to students' manipulations, but rejecting the input of incorrect data such as the choice of a frequency or amplitude outside the specified limits of the connected lab equipment which could lead to system damage. A remote instrument is hereby denoted as a projection of the functionality and look and feel of a *remote* device such as a function generator or an oscilloscope (*server*) onto a virtual device (*client*). All controls on the virtual device correspond to the control on the real device. Thus, a virtual device is not just a simulation of a real device. Simulations can not be used for the modeling of physical effects, e.g. the development of a clock signal at higher frequencies or the influence of temperature on a circuit. Remote labs can be used to augment real laboratory experience in science and technology. When done effectively, this will increase the access of students to knowledge and enhance performance as well as the quality of the educational out-

come. The paper is organized as follows. Section 2 reviews some related work. The proposed *Telematik* framework is compared with commercial and non-commercial solutions. Section 3 presents the software architecture. Section 4 gives examples of implemented remote instruments. Section 5 concludes the paper.

2 Background and Related Work

There are several interactive virtual/ remote labs available on the web [1-8]. Physics 2000 [8] comprises interactive Java applets through where students can explore elementary physical phenomena. Some of these labs are implemented as Java applets or dynamic web-pages - but without generic methology for the development of such labs. Researchers are pursuing the development of remote labs, where robots receive commands from students and reproduce virtual experiments in a fully equipped real lab. Hagen's faculty of electrical engineering supports a lab exists where students can remotely control robots [14][15][16]. After the experiment is finished (the students reached the goal to control the robot such in a way that it reaches a certain target) the robot returns to a pre-defined state. Visually advanced computer science lab exercises are offered through the Howard Hughes Medical Institute by *Bio-Interactive* [7]. *Bio-Interactive* is a collection of learning modules where students can interactively explore topics in cardiology, neuro-physiology and the human immune system. Here, user-action is restricted mostly to clicking on the various instruments. For a realistic learning experience, the users should be able to perform the exact operations as they would on the real instrument. Slider widgets are used in most of the aforementioned work. For the actual instrument, this might not be the case. For instance, we might need to rotate a dial to set a particular value rather than slide a marker across a slider.
Current solutions from academic institutions have the following disadvantages:

- Insufficient availability (downtimes, browser dependencies)
- No generic methology (special solutions for existing lab equipment)
- Partially unavailable source-codes
- Elements such as knobs, etc. are not freely definable.

Some solutions are based on platform-independent Java applets. But it is a general (known) problem that the execution of Java applets can be browser dependent. Since we use Java *applications*, we are only depended on the Java version the student is running, but not on the browser.
Current commercial products have a number of disadvantages compared to our solution:

- They are not free
- They are platform-dependent
- The source code is not available and therefore not adaptable
- The programmer has to learn a (non-standard) graphical programming language – therefore the development time will increase in comparison with traditional programming languages like C, C++ or Java
- No code-inheritance
- Elements such as knobs, etc. are not freely definable.

It should be mentioned that commercial solutions offer a wide variety of drivers for different devices and a large library of definable elements. With the graphical programming language, code can be inherited by using the same basic elements. Source-code availability is important because it makes the software less error-prone and more maintainable. Furthermore, the user is not dependent on a company, where the products may quickly change or will not be supported any more. We overcome these shortcomings through a number of advantages:
Documentation

- Source availability $\Rightarrow$ better documentation/ improvement
- Direct control over the existing device, no simulation

Data Management:

- Data won from experiments can be quickly and easily exported and analyzed using external solutions like MathLAB, Excel, MathCAD or SPSS etc.
- Printing to any printer including network printers and faxes is possible

Cost:

- No additional, and/or hidden costs, e.g. no GPIB-cards (**G**eneral **P**urpose **I**nterface **B**us)[1], web-cams, etc. have to be bought to get the system to its fully functionality
- Existing experiments can easily be ported/re-used to remotely controlled labs
- Compared with graphical programming languages, the coding of device drivers is accomplished faster when using an object-orientated programming language like Java when sufficient documentation is available. Therefore the development costs are reduced
- Easy adaptation to newer technologies and low development costs through code-inheritance

The use of creative renderings of objects and their behaviors allows the student to freely experiment in the virtual world. The module content - complexity of problem solving and sophistication of technical skills are vertically scaled so that each student can move through the module depending on the preparation. Developers can use a graphical composition tool from modern graphical Rapid Application Development (RAD)-Tools to set up the user-interface and basic communication. The only thing the developer has to do is to implement a subset of the commands the device supports. Code inheritance makes the implementation of device-family based device drivers easy and fast. To accomplish this and to ensure the re-use of code [9], we used Java Beans[10][11].

3 The *Telematik* Architecture

The development and implementation of remote laboratories is a very tedious and costly task. A central challenge is to develop a starting point to allow the rapid, easy and flexible creation of remote laboratories. To access many different devices, a high level of abstraction from the developer's point of view must be reached. The *Telematik* framework simplifies the integration of new real hardware into a virtual computer science laboratory, furthermore the integration of existing hardware by distinguishing a device dependent and independent part.

We have identified two levels of abstraction for a device:

- Communication: here, we separate device-specific communication (commands, hardware interface) and general communication over TCP/IP (Transmission Control Protocol/ Internet Protocol) [12][13].
- Front-end: we separate a device-specific part (knobs, buttons with certain properties) of the Graphical User Interface (GUI) to control a remote device and a generalized version, so code inheritance can be used for the simple and fast implementation of new front-ends.

The modeling of a device can be explained by the ring model in Figure 1.

[1] The GPIB specification was developed by Hewlett-Packard and is also known as IEEE-488.

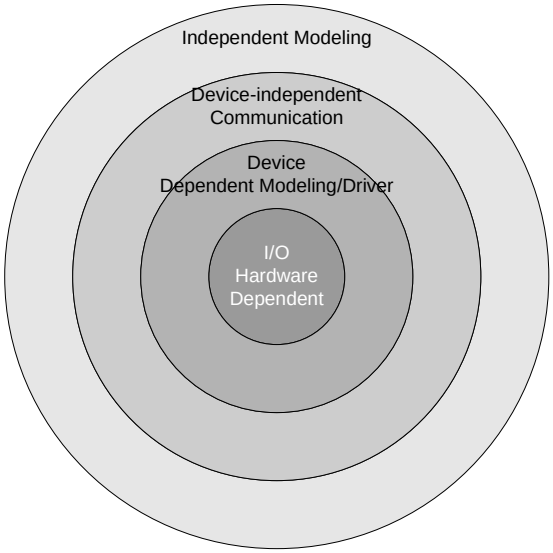

Figure 1: The ring model with different levels of abstraction

In the centre all hardware dependent I/O is modeled. Since we have used java, we use the Java Native Interface (JNI) at this level to model communication ports such as RS232. A java native method can use native hardware, because it is a part of e.g. a compiled C-program, using the native method interface from JNI. We use javax.comm from Sun Microsystems for the communication over serial and parallel ports. However, the modeling of other communication protocols such as GPIB is possible by using JNI. In the next ring the device driver is modeled as a set of java methods. Here, a subset of the device's commands has to be specified. This set of commands is integrated in the client and not part of the server, although RMI (Remote Method Invocation) could be used. Since this methology would increase development time and communication overhead, we omitted the use of RMI. By using a subset, it is possible to increase security, because it is not possible to send system-control messages from the client. The next ring contains the device-independent communication which is done over TCP/IP. The server accepts commands as plain text messages, since we have no sensitive data which must be encrypted. Thus, it is possible to test the server implementation without having to implement a GUI for the device. Figure 2 shows the basic client/server based architecture.

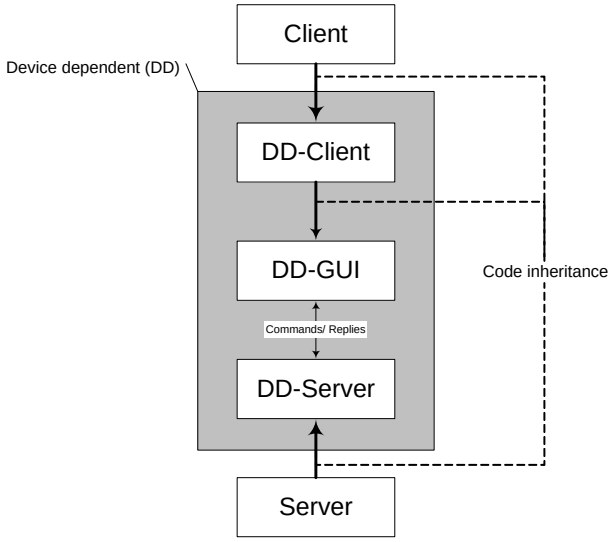

Figure 2: The client/server-based architecture

4 Exemplary deployment of a lab using the *Telematik* architecture

Our remote lab consists of a function generator, a DSP (digital signal processor) development board and an oscilloscope, each with specific pre-programmed behaviors, interface, look-and-feel and commands to control the device. To load programs onto the development board, the extraction of the communication protocol was necessary. As every device-specific part, the protocol was implemented directly in the server structure. Students select the binary file they produced at home. The file is uploaded using the client software. The server receives the binary file, extracts the data, converts it in the device-specific protocol and uploads the data to the development board. Then it will start the program execution. When the program starts, a feedback is given to the user. The student interacts with the devices in order to attain a set of given goals, i.e. the study of DSPs, microcontrollers, communication protocols, etc. Users can access the lab via wireless LAN (local area network) or any other network connection. Figure 3 shows the experimental setup. It consists of two main servers (telematik1, telematik3). The server telematik2 is only listed for completeness. A primary firewall protects the subnet against malicious attacks. All web-based accesses go to the webserver, running on server telematik1. Telematik1 also runs the servers for the function generator and the digital oscilloscope, both connected to the serial ports. All servers run Microsoft Windows 2000. With few changes, the system is able to run on any java-capable system where the javax.comm interface is available. Every server is running a personal firewall, blocking all accesses from non-local IP addresses. Over a virtual private network (VPN) connection, students are able to get a local IP and access the lab. Telematik1 also runs the experiment selector. The selector is a development from the technical lab in Hagen. It integrates simple filter experiments such as high-, band- and lowpass on a printed circuit board. A PHP-script is used to access the board and to switch between experiments/outputs. Additionally, the output can be switched to the DSP (digital signal processor) development board from analog devices. This board holds the popular ADSP2181 [20] signal processor. A DSP-client, which is based on the *Telematik* framework is able to load and execute programs on the board. The output of the DSP board is connected to the experiment selector. Remember that the outputs of the high-, band-, lowpass and the DSP-board are connected to the selector. The output of the function generator is connected to the experiment selector and to channel 1 of the oscilloscope. The output of the experiment selector is connected to the oscilloscope (channel 2). So, the outputs from the filters and the DSP can be manipulated by the function generator, depending on the experiment selection. The original and the modified signal can be observed on the oscilloscope. To do this in real time a webcam is connected to Telematik3. Over an integrated web-interface, all functions of the lab can be accessed without knowing the experimental setup. In case of a non-terminating DSP program (e.g. an endless loop, waiting infinitely for an external event) there must be the possibility to remotely reset the DSP-board. There are several ways to do this. We discovered the possibility to apply the reset signal externally and used the experiment selector for this operation. The selector produces the necessary reset signal at a certain signal combination on the parallel port. For remote administration, a virtual network computer service is running on all servers.

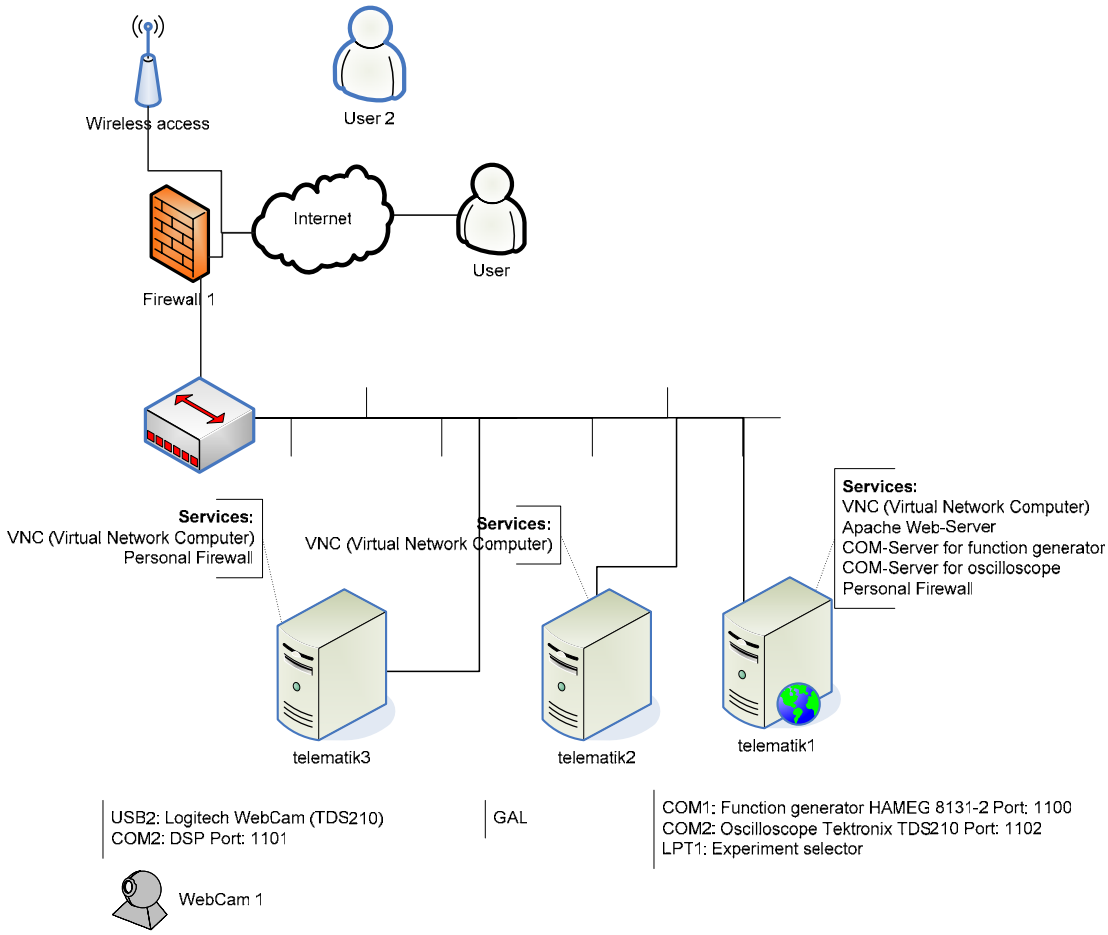

Figure 3: Experimental setup

4.1 The Function Generator Client Application

A function generator is a signal source which provides precision waveform signals (e.g. sine, square or triangular) over a frequency range. The instrument also provides a continuously variable amplitude level. Figure 4 shows the function generator interface. For better operation, all basic default settings are integrated in the client.

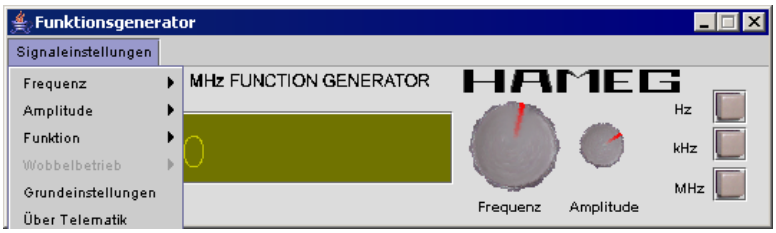

Figure 4: The HAMEG8131-2 function generator application

4.2 The Oscilloscope Client Application

The oscilloscope (Figure 5) draws the graph of an electrical signal. In most applications, the graph shows how signals change over time: the vertical (Y) axis represents voltage and the horizontal (X) axis represents time. An oscilloscope is therefore a linear voltage indicating device. A spot on a display screen is caused to deflect in proportion to the applied voltage. The vertical position of the spot depends on the 'Y-amplifier' input while horizontal position of the spot is determined by the waveform applied to the 'X-amplifier' input. The oscilloscope has a time base generator circuit that generates a voltage that varies linearly with time. When this voltage is applied to the X-amplifier, the voltage applied to the Y-amplifier can be observed with respect to time. The time base generator circuit is also connected to the 'trigger' circuitry which controls the starting instant of the waveform. Any output from the oscilloscope can be captured and copied in the report document as pure data.

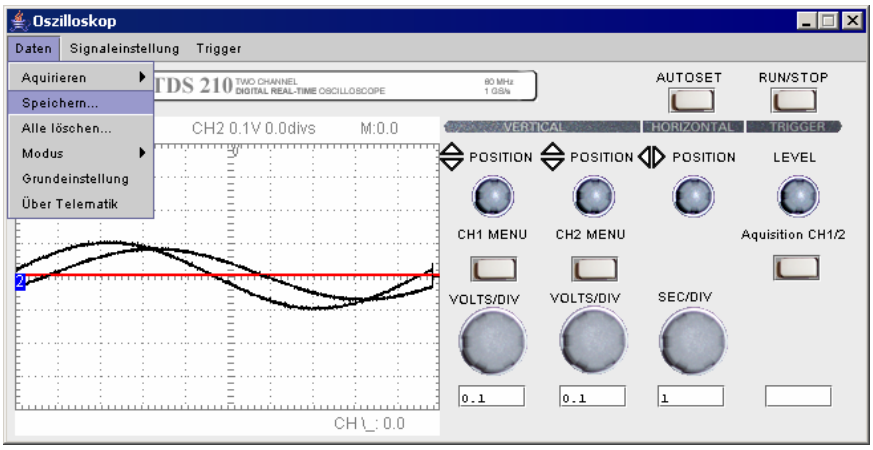

Figure 5: The Tektronix TDS210 oscilloscope application

5 Conclusion

The benefits of remote labs over actual laboratories are found in their increased portability, cost effectiveness, reduced need for teacher intervention, increased student interest and control, adaptability to various learning styles and learning rates, web-ready software and self-testing. Remote labs will satisfy a growing need for truly interactive learning software. This paper presented software architecture for the development of remote laboratories. The architecture can be used to develop tools to support scientific laboratories that allow the sharing of unique or expensive instruments. Our first experimental findings called for a user manual, covering the most frequently asked questions. When doing web-based exercises, it is deceiving to play around and not to do any useful work. We suggest a strict level-based concept to solve this problem. When doing experiments the students can gain credits. The credits are summed up and added to a basic amount of credits. The higher the level of questions, the more credits a student can get. This is a good measurement for the level of preparation, because non-prepared students might open the labs and simultaneously refer to their textbooks in order to understand the concepts. More motivated students get more credits and hence more time. The deployed lab is currently a single-user lab because not enough hardware is available. With more hardware, the support of multiple-users is possible. A future extension will be the support collaborative work [17][18][19]. This enables researchers and students to work together across geographic and organizational boundaries to solve complex, interdisciplinary problems by accessing remote resources. The short-term goal of remote labs is to serve as a preparation and supplement for actual labs. The students are familiarized with devices and procedures before they actually go into the lab and perform experiments. The rehearsal of an experiment with little complexity is a cost-effective

preparation for the use of limited and expensive lab facilities. The lab is continuously available on our website.

References

[1] Deutsches Museum München, *remote_lab - Das ferngesteuerte Labor im Internet.* Eberhard von Kuenheim Stiftung, Deutschland. http://www.remote-lab.de/

[2] California State University, Center for Distributed Learning (CDL), http://www.cdl.edu/

[3] M. W. Davidson, K. I. Tchourioukanov, and M. Abramowitz, *Virtual scanning electron microscopy applet*, Olympus America Inc. and The Florida State University, 1998. http://micro.magnet.fsu.edu/primer/java/electronmicroscopy/magnify1/index.html

[4] M. Duguay, *The TeleLearning Experience*, http://www.telelearn.ca/

[5] Virtual Laboratory, National University of Singapore (NUS), http://vlab.ee.nus.edu.sg/vlab/authfinal.html

[6] M. V. Goldman, "Physics 2000 interactive applets", University of Colorado, Boulder, CO. http://www.colorado.edu/physics/2000

[7] Howard Hughes Medical Institute, "Virtual laboratories" http://www.biointeractive.org/

[8] LEYBOLD DIDACTIC GmbH, Remote-Lab Versuch, http://remote-lab.leybold-didactic.de/cgi/nm_rclShow_cgi.exe

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley Longman, Inc., Reading, MA, 1995.

[10] IBM, Inc., "Bean markup language," At: http://www.alphaWorks.ibm.com/tech/bml/

[11] Sun Microsystems, Inc., "JavaBeans API specification," At: http://www.javasoft.com/beans/

[12] Defense Advanced Research Projects Agency (DARPA), Transmission Control Protocol, RFC: 793, Sept. 1981, At: http://www.faqs.org/rfcs/rfc793.html

[13] Defense Advanced Research Projects Agency (DARPA), Transmission Control Protocol, RFC: 791, Sept. 1981, At: http://www.ietf.org/rfc/rfc0791.txt

[14] C. Röhrig, A. Jochheim: Remote Control of Laboratory Experiments, 19th World Conference on Open Learning and Distance Education, ICDE-1999, Vienna, Austria 1999

[15] W. Laaser, M. Gerke, H. Hoyer: Teaching Control Theory by Multimedia, 19th World Conference on Open Learning and Distance Education, ICDE-1999, Vienna, Austria 1999

[16] C. Röhrig, A. Jochheim: The Virtual Lab for Controlling Real Experiments via Internet, IEEE International Symposium on Computer-Aided Control System Design, CACSD'99, Kohala Coast-Island of Hawaii, Hawaii, August 1999

[17] A. Haake, M. Bourimi, J. Haake, T. Schümmer, B. Landgraf, Endbenutzer-gesteuerte Gruppenbildung in gemeinsamen Lernräumen, *DeLFI 2004: Die 2. e-Learning Fachtagung Informatik*, GI-Edition Lecture Notes in Computer Science , GI e.V., Bonn, 2004, pp. 235-246.

[18] A. Haake, M. Bourimi, J. Haake, T. Schümmer, B. Landgraf, CURE – Eine Umgebung für selbstorganisiertes Gruppenlernen, *i-com Zeitschrift für interaktive und kooperative Medien*, 2004.

[19] A. Haake, M. Bourimi, J. Haake, T. Schümmer, B. Landgraf, Supporting flexible collaborative distance learning in the CURE platform, *Proceedings of the Hawaii International Conference On System Sciences (HICSS-37)*, IEEE Press, 2004.

[20] Analog Devices, DSP Microcomputer ADSP-2181, http://www.analog.com/UploadedFiles/Data_Sheets/505104853ADSP2181_d.pdf, 1998.