Int. J. Engng Ed. Vol. 19, No. 3, pp. 427–432, 2003
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2003 TEMPUS Publications.

Embedded Internet Laboratory*

F. NAGHDY, P. VIAL and N. TAYLOR
School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, NSW, 2522, Australia. E-mail: f.naghdy@uow.edu.au

A laboratory is developed in this work to teach embedded Internet systems to Bachelor of Internet Science and Technology students. The design and development of the system have been carried out based on the pedagogical outcomes expected from the laboratory and the subject. Accordingly, the laboratory designed for this subject does not only support all the objectives of the subject, but provides a platform to demonstrate all the principles associated with the concepts in a tangible manner. For example, the laboratory itself is designed as an embedded Internet system and access to the experimental device takes place remotely through the Internet from a client. The design and development of the laboratory is described and progress made is highlighted.

INTRODUCTION

COMPLETE on-line delivery of educational courses via the World Wide Web (WWW) is now an accepted norm and practice in areas such as Arts and Management, where hands-on experience and acquisition of practical skills are not essential. On the contrary, the introduction of engineering courses on the Web has been slow due to their high dependence on practical laboratories.

The Internet-enabled laboratory is the key technology that makes the on-line delivery of engineering degrees a viable option by providing remote access to an experimental rig via the Web.

The focus of this work has been on developing a cost-effective, robust and viable solution for setting up an Internet-enabled laboratory. It can be practically utilized for a wide range of applications in the home, industry and educational institutions.

In spite of its generic nature, the proposed solution is designed to provide remote access via the Web for an 'Embedded Internet Systems' laboratory. This subject is part of a Bachelor of Internet Science and Technology (BIST) degree offered in the Faculty of Informatics, University of Wollongong.

During the course of the paper, a review of the related work will be carried out. The overall architecture of the developed system will be then described and various hardware/software components of the system will be introduced. The progress made so far will be reported and some conclusions will be drawn.

BACKGROUND

The traditional computer-aided laboratory often suffers from technical limitations that arise from diversity in software packages, hardware, computer platforms and operating systems [1]. This becomes a source of confusion for students as well as laboratory assistants, and makes learning more difficult.

Web systems may be used to overcome these limitations, as well as providing students with remote access to laboratories. The recently developed web-based remote laboratory is an example of this. This tool consists of an actual laboratory attached to a web interface [2]. The student enters experiment settings via the web, and a software interface then converts these into a form that can be understood by the experimental rig. Data is recorded during the experiment and then reported back to the student via the web interface. A number of universities have recently succeeded in implementing various forms of remote laboratory.

The SBBT laboratory was developed by Oregon State University in conjunction with Netscape to provide remote access control of a three degrees-of-freedom robot arm. The main component of the laboratory is a Java server running on a Sun Sparc 5 workstation. The workstation is connected to a Motion Control Interface (MCI) and a control PC that are in turn connected to a robot arm. The server contains a lab manager application that controls access to the robot arm. UNIX shell scripts are used by the lab manager to interact with the MCI and the control PC. Students may connect to the robot arm via the lab manager, where they can then implement a control algorithm and observe the response of the robot arm to the algorithm [3].

Although it functions correctly, the major problem with this laboratory is cost. The use of a high-end Unix workstation prevents the laboratory from being within the budget of most learning institutions.

A remote laboratory was developed by Case Western Reserve University to provide students with access to a Bytronic Process Control unit (BPC) located within their Process Control and

* Accepted 22 February 2003.

Automation Laboratory. The BPC is a water flow system in which flow level and temperature can be measured and controlled. The remote laboratory consists of a Macintosh Power PC connected to the process rig via a PLC interface module. The PC contains a LabVIEW server, which is used to provide both static HTML pages and also a Common Gateway Interface (CGI) for executing programs. In this laboratory, students post experimental control data to the server via an HTML form. The server then runs the appropriate CGI program sending the data to the PLC module, which then starts the experiment. Output data is logged and made available to the student via an FTP site [4].

This system successfully provides remote access to the laboratory. It has an advantage over SBBT in that it is simpler and significantly more cost-effective. However, the use of LabVIEW server software greatly limits the laboratory functionality. Additionally, the system is not fully interactive.

Robotoy is a simple remote laboratory developed by the School of Electrical, Computer and Telecommunications Engineering (SECTE) at University of Wollongong. It allows remote users to control the operation of a robotic arm located within SECTE's Mechatronics Laboratory. It utilises a Linux PC to provide an interface between the remote user and the arm. An Apache web server installed on this PC allows users to request a particular action by the arm. The CGI scripts written in PERL are then used to process the request and run a C program with the appropriate arguments for that request. The C program communicates with the arm via a parallel connection between the PC and the arm. Visual feedback of the arm is provided via video cameras and a frame grabber card. Additional functions such as camera selection and status checks are performed via serial communications with a microprocessor [5].

PEDAGOGICAL ISSUES

The developed laboratory is meant to assist the students to develop a better understanding of Embedded Internet Systems defined as microchip-based control systems that are dedicated to perform specific tasks or groups of tasks. Embedded systems are now intrinsic components of mechatronic systems and are employed in all the modern medical, industrial and consumer products.

With advances in the Internet technologies, much attention is currently being paid to how real-time embedded systems can be linked and shared across the Internet. In such systems, people and devices easily and automatically communicate with each other with no one having to know about computers or software.

In embedded Internet systems there is a two-way communication linkage between the Net and virtually every mechanical/ electronic devices that touches the human life. This includes even life-critical devices which monitor/ assist major transplant organs in one's body by remotely monitoring, gauging and controlling them.

As mentioned before, the laboratory developed in this work is meant to support the teaching of a subject called 'Embedded Internet Systems'. The aim of this subject is to provide students with an understanding of the concept and typical applications of embedded Internet real-time systems; and to familiarise them with methodologies and tools used to design and develop them. The subject covers Web servers in embedded systems, embedded operating systems, embedded system configuration, real-time embedded databases, and design for embedded Internet.

Accordingly, the laboratory designed for this subject will not only support all the objectives of the subject, but provides a platform to demonstrate all the principles and associated concepts in a tangible manner. For example, the laboratory itself is designed as an embedded Internet system and access to the experimental device takes place remotely through the Internet from a client. Other pedagogical issues imbedded in the laboratory include:

- The design of the laboratory illustrates the primary components and configuration of an embedded Internet system and how they should be integrated.
- The communication protocols and methodologies of the laboratory show a typical design of the necessary interaction between different elements.
- The software tools and operating systems used in the experiments familarise the students with typical means which can be used to develop and operate embedded Internet systems.

SYSTEM OVERVIEW

The solution developed in this work is illustrated in Fig. 1. The embedded Internet device, a soccer robot in this study, is interfaced to the Internet via a TINI (Tiny InterNet Interface).

In the following sections different components of this system are defined.

TINI

This interface device provides an optimal and very low cost but universal solution that can be applied to any embedded Internet system used in a laboratory.

TINI is a low-cost microcontroller which connects directly to the Internet. The first implementation of it was in 1998 as a Java programmable device capable of controlling household electrical goods. TINI has been further developed by Dallas SemiConductors and the TINI SIG (Special Interest Group) and the result is a broad platform including software and hardware that can

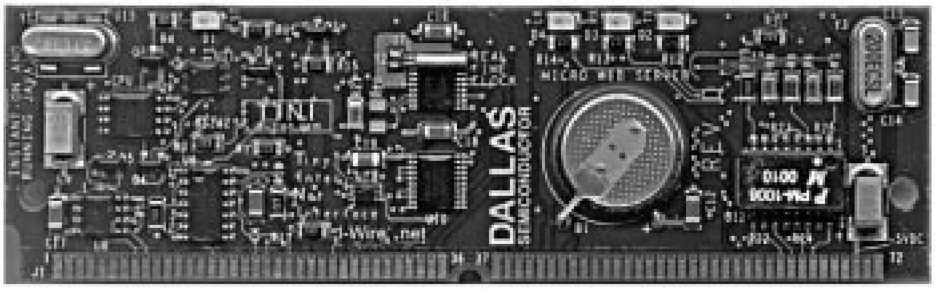


Fig. 1. The TINI board.

be used to create intelligent network devices. Targeted devices have a small footprint, low power consumption and are cost sensitive.

A TINI board is connected to the local area network via an Ethernet connection. Users connect directly to the TiniHttpServer running on the TINI board via a web browser. A java applet is downloaded to the user's computer and during the initialisation stage a socket request is sent to the TINI board. This socket request is answered by a java application running on the TINI board. This program performs Ethernet to Serial and Serial to Ethernet conversion for the connection between the user computer and the I/O System.

The soccer robot

The robot is illustrated in Fig. 2. In the final system, a web-cam connected to the TINI will provide a visual feedback of the robot and its location. This feature is not fully implemented at this stage.

The overall system is programmed in C and Java to provide communication between different components of the system and to enable the services required.

TiniTutor

In order to control the robot, TINI is used with a socket supplied from Taylec [6] called the TiniTutor. This socket is equipped with 8 digital inputs and outputs, one analogue input and one analogue output (amongst other features). The soccer robot requires four digital outputs (all asserted low) and one digital input (from the collision switch) for its control.

The code for the TiniTutor requires the use of the TiniHttpServer software freely available from [7]. This allows the setup of the Applet software on the TINI, which acts as a server for HTTP-based web pages. The Applet and source web page are thus downloaded to the TINI and stored as files on its operating system (Slush). Another Java program is written which runs concurrently on the TINI with the Java Applet called SoccerRobot.tini. It starts up waiting for a call from the applet on port 1000 (this being the port originally chosen for communication). Once the applet obtains a connection, the program SoccerRobot.tini waits for the applet to send a command using an Ethernet packet. When it receives the Ethernet packet it is decoded to locate the required operation.

For example, the command for moving the SoccerBot forward is '1'. When SoccerRobot.tini receives this, the digital output which drives the SoccerBot in the forward direction is asserted low. A separate thread is provided in SoccerRobot.tini to listen for an input from the collision switch. When it receives this input, it stops the soccer robot by asserting high on all the outputs and sends an Ethernet packet back to the Web browser running the applet to provide feedback to the user that the TINI has detected a collision event.

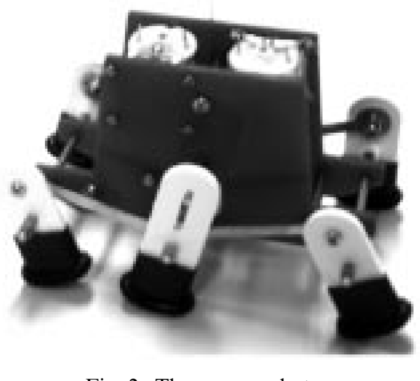


Fig. 2. The soccer robot.

Robot drive card

In order to drive the Robot, four 3 V, 300 mA signals are required. These output signals are only required to provide a constant voltage. This could be achieved through an I/O card using four pins of ioport1 which provide the commands required to drive the robot in four directions of forwards (F), backwards (B), left (L) and right (R).

TINI could not provide the current required to drive the robot. Hence a relay-based circuit was designed for this purpose.

The mapping of these inputs to the ioport pins is shown in Table 1. The inputs to the system all have a low assertion level. This board was tested and was found to provide a reliable means of controlling the movements of the robot.

The complete set-up including the soccer robot and TiniTutor is shown in Fig. 3.

Table 1. Mapping of directions to ioport pins

Circuit Input	Direction	Pin
Input 1	Forwards	Ioport1, pin 2
Input 2	Backwards	Ioport1, pin 4
Input 3	Left	Ioport1, pin 5
Input 4	Right	Ioport1, pin 3

COMMUNICATIONS SOFTWARE

The communications software integrates different components of the system and provides a means for the components and user to interact with each other. It consists of three main parts of the interface applet, the serial to Ethernet application and the control program. The architecture of the communications software is illustrated in Fig. 3.

The interface applet is the user interface that has been designed for the system. The TiniHttpServer hosts this applet on the TINI board.

TiniHttpServer is a web server that was developed by Smart Software Consulting for the TINI board and is capable of serving Java Applets, Java Servlets and HTML documents.

The serial to Ethernet application is a java application that runs on the TINI board acting as a transparent proxy between the interface applet and the control program. It has been designed as a multi-threaded application so that the serial/ Ethernet and Ethernet/serial conversion processes may run in parallel.

The system must be capable of dealing with:

- establishing a new connection to the system;
- control of the robot's direction by a user;
- feedback to the user from the robot;
- detection of a link failure.

A user can connect to the system via a web browser providing a simple, user-friendly and familiar interface from which to control the system. When the user wishes to connect to the system they redirect their web browser to the homepage of the TINI board. An HTTP GET request is then sent to the default web browser port (port 80) of the TINI board.

This HTTP GET request is received by the TiniHttpServer. As no particular web page has been requested the default web page index.html is sent back. This web page contains an applet tag, which references the interface applet. The web browser recognises this and requests a copy of the applet.

Once the interface applet has been retrieved the JVM on the client's PC begins to execute the Java applet. It executes the init( ) function which initialises the user interface and attempts to open a TCP/IP socket connection to the TINI board IP address on port 1000.

The connection request sent to the TINI board is received by the serial to Ethernet application that has a server socket listening to port 1000 for incoming requests. The serial to Ethernet application then stops listening to port 1000 for incoming requests and sends an acknowledgement back to the client's PC and opens the socket connection. The serial to Ethernet application and the interface applet then bind their InputStream and OutputStream to the socket connection.

Finally, the serial to Ethernet application opens a serial connection to the I/O board. Once this has been completed it binds another InputStream and OutputStream to this connection launchings the SerialReader and SerialWriter threads. The current thread is assigned as the maintenance thread. The SerialReader thread provides communication from the I/O board to the client's PC and the SerialWriter thread provides communication in the opposite direction. The maintenance thread monitors the status of both of these threads to ensure that no failure occurs.

The serial to Ethernet application is a Java application that runs on the TINI board acting as a transparent proxy between the interface applet and the control program. It has been designed as a multi-threaded application so that the serial/ Ethernet and Ethernet/serial conversion processes may run in parallel.

Once a user has established a connection to the system they can control the movements of the

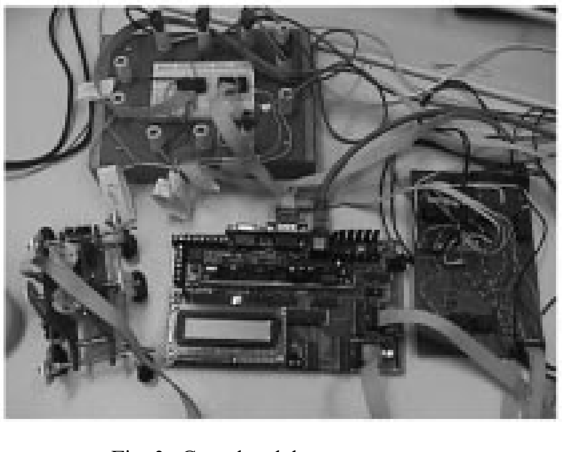
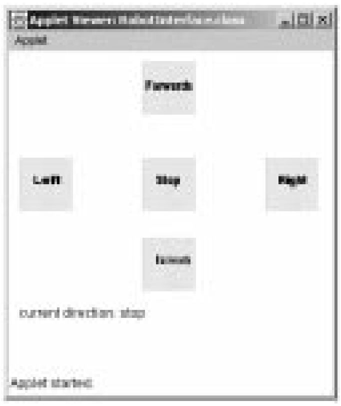


Fig. 3. Complete laboratory set up.



current direction: stop

Applet started

Fig. 4. The user interface.

robot via the interface applet. The user interface that they are presented with is shown in Fig. 4.

The user is able to move the robot forwards, backwards, left and right and can also stop the robot. The buttons that the users click on to achieve this are GIF images. A mouse listener instance has been created to detect if the users click within the boundary of these images. For example, if the 'Forwards' button is selected the mouse listener object detects this and sends the appropriate message to the serial to Ethernet application via the socket connection it previously opened.

The robot has been fitted with a collision detector. When this collision detector is tripped the interface board asserts pin 6 of ioport1. The Software Timer Interrupt polls the status of this pin every 200 ms. If it finds that the pin has been asserted, the control program stops the robot and then sends the value '5' back to the client's PC to indicate that the robot has collided with an object.

PROGRESS SO FAR

The design and development of all aspects of the project apart form its visual feedback are complete. The laboratory was used in the second session of 2002 for the first time. The laboratory notes were developed as a Web page

The students who attempted the embedded Internet laboratory came from different backgrounds. Some had experienced earlier laboratories using TINI and Java and thus were familiar with the embedded Internet platform that was used. Others were not so experienced and needed to learn some basic Java, getting familiar with TINI, and undertake the laboratory activities.

The students were asked to submit a laboratory report and undertake an exam on the laboratory contents. The reports were mostly satisfactory and showed that students had understood the concept and had developed skills to program TINI. The results of the exam were rather varied as some students did quite well whereas others showed poor performance.

The work of developing a systematic validation and study of the pedagogical outcomes of the project in relation to the subject and its objectives is still outstanding. The evaluation and validation will take place next session primarily by asking students to fill evaluation forms at different stages of their work. In the next phase, automatic monitoring of the students on line will be implemented to study the activities of the students on different aspects of the laboratory.

CONCLUSION AND FUTURE WORK

The primary goal of the project has been to build a cost-effective system for teaching embedded Internet systems to BIST students. The design and development of the system have been carried out based on the pedagogical outcomes expected from the laboratory. The laboratory itself is an embedded Internet system and provides remote access to control a soccer robot through a Web browser.

As far as the cost is concerned, the TINI and I/O card each cost around $US60. The total is substantially lower in cost than a PC-based solution, particularly if the number of required stations is large.

The laboratory is scheduled to be used in Autumn Session of 2002 as part of the subject, offered for the first time. The laboratory is set up on the Web and students can access the robot via the TINI.

Although the remote access to the laboratory is technically possible, due to the non-availability of Web cameras it is not provided for the time being.

The future work includes interfacing the Web cameras to the experimental rig and enabling the client to view the activities of the experimental rig. The pedagogical outcomes of the laboratory will be also evaluated through student's surveys and other validation methods.

The experience gained through this work can be incorporated in other engineering laboratories. This will have two major impacts. It will make the engineering practical laboratories more accessible to the students and hence provide them with more opportunities for experimentation and learning. It will also make the distance offering of engineering degrees a reality and will provide new opportunities for training of quality engineers.

REFERENCES

1. P. Doulai and M. Aldeen, Engineering and science laboratory courseware delivery using world wide web technology, *IEEE Int. Conf. Multi Media Engineering Education 1996*, Vol. 1, July 1996, pp. 339–344.
2. S. E. Poindexter, and B.S Heck, Using the Web in your courses: the how-to's and the why's, *Proc. 1998 American Control Conference*, Vol. 2, June 1998, pp. 1299–1303.
3. Bhandari and M.H. Shor, Access to an instructional control laboratory experiment through the World Wide Web, *Proc. 1998 American Control Conference*, Vol. 2, June 1998, pp. 1319–1325.
4. M. Shaheen, K. A. Loparo and M. R. Buchner, Remote laboratory experimentation, *Proc. 1998 American Control Conference*, Vol. 2, June 1998, pp. 1326–1329.
5. P. Ciufo, *Welcome to the Robotoy Homepage*, URL: http://roboty.elec.uow.edu.au/, accessed: January 1999.

6. Taylec Website, accessed March 2002, http://www.taylec.com
7. TiniHttpServer home page, www.smartsc.com, accessed March 2002.

Fazel Naghdy received his first degree from Tehran University in 1976. He then received an M.Sc. from the Postgraduate School of Control Engineering, University of Bradford, England, in 1980 and received his Ph.D. from the same University in 1982. Currently he is an Associate Professor at University of Wollongong, School of Electrical, Computer & Telecommunication Engineering. Fazel has extensive research experience in the areas of intelligent control, robotics and mechatronics. He has received many research awards and published around 145 technical papers in international journals and conferences. His current research interests include embedded Internet systems, haptic rendered virtual manipulation of clinical and mechanical systems, intelligent control and learning in non-linear and non-structured systems.

Peter Vial is an Associate Lecturer in the School of Electrical Computer and Telecommunications Engineering. He graduated with a Bachelor of Engineering Hons 2 (i) in 1987 and a Master in Telecommunications Engineering (Honours) in 1996. In 2000 he received a Diploma of Education (Mathematics). All qualifications were awarded by the University of Wollongong. His interests include issues in electrical engineering education, embedded internet systems, digital signal processing, modeling of telecommunication systems and wireless digital communications.

Nathan Taylor completed secondary education in 1997 and started University studies in Computer Engineering in 1998. He graduated with the first class Honours in December 2001 in Computer Engineering. His final year thesis was on embedded internet control of a SoccerBot. His work is currently used as part of a laboratory in an undergraduate degree.