Int. J. Engng Ed. Vol. 20, No. 5, pp. 879–885, 2004
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2004 TEMPUS Publications.

MicroLab: A Web-based Multi-user Remote Microcontroller Laboratory for Engineering Education*

A. KUTLU
Department of Technical Education, University of Suleyman Demirel, 32260 Isparta, Turkey.
E-mail: akutlu@tef.sdu.edu.tr

This paper discusses the implementation of a Web-based multi-user remote microcontroller laboratory designed for electric-electronic engineering education. The MicroLab has been developed to provide programming and monitoring microcontroller modules through the Internet for undergraduate students. Students can access the MicroLab individually and simultaneously. The MicroLab consists of a server with web cams and microcontroller modules which have 8051 core architecture, and are equipped with a CAN (Control Area Network) as its peripheral architecture.

INTRODUCTION

EIGHT-BIT MICROCONTROLLERS are usually preferred for teaching-learning in the electric-electronic engineering laboratories, due to their simplicity and low cost. Using simulation techniques to understand the nature of microcontrollers are preferable [1]. Nevertheless, it is necessary to have 'real' circuits and experiments to make sure programs on the microcontrollers are working efficiently. As discussed in the study of Alhalabi et al., simulation techniques can never replace the need for a real laboratory [2]. In order to conduct the required experiments, the students must be present in the laboratory during the experiment. Building experiments on experimental modules are sometimes time consuming. Because of time consideration, it is preferable to conduct the experiments over the Internet at any time and from any location without spending too much time.

Remote laboratories have already been used in control engineering, [3–5] robotics, [6–8] and chemical engineering [9]. In this project, the main expectation from a remote laboratory is not only to supply the students with one experimental module but also to provide each student their own experimental module simultaneously. In order to achieve multi-user facility, a remote laboratory should build up an integrated environment for users as controlling the real devices from the client side and conducting the actual experiments in the remote laboratory through a real-time network.

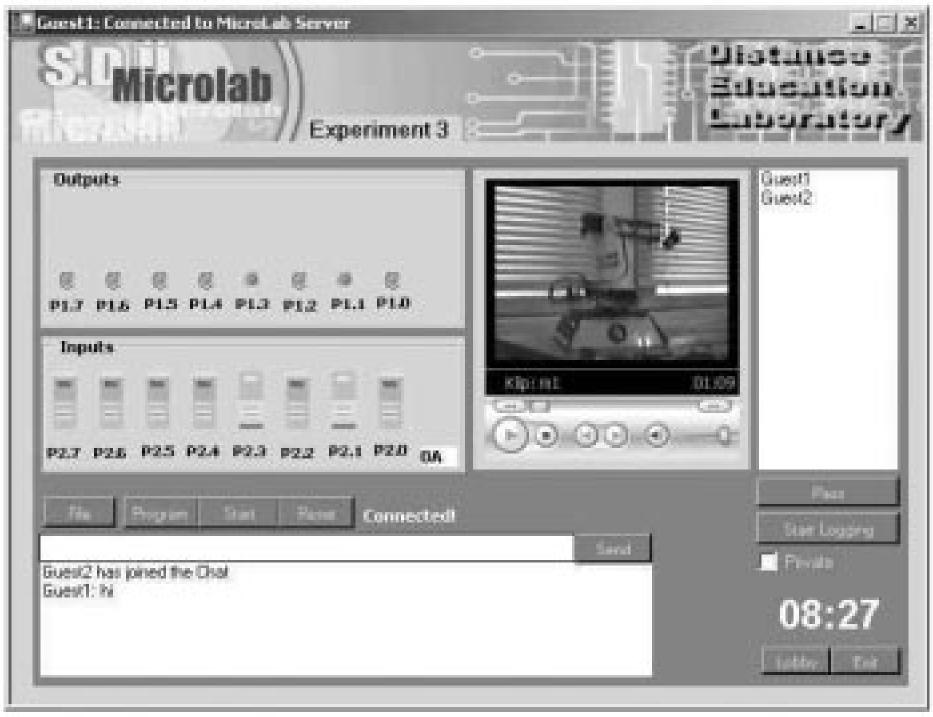
This paper introduces MicroLab remote laboratory which meets the requirements of multi-user and web connection. The user interface run on a student's computer is an application program called MicroClient. MicroClient provides connection with an experimental module through a server application program called MicroServer. Whenever the connection is established to an experimental module, the student can upload the software written for the microcontroller and can give instructions required by uploaded software. Students can also monitor their results on the screen either by using simulation workplace or web camera. Once the students enter the session using their ID and password, they can access the experimental module they need. On the other hand, the experimental module can be conducted by only one student. The system allows guest users to enter the experiment process only as an observer.

THE USER INTERFACE AND SESSION MANAGEMENT

Microcontroller experimental modules consist mainly of port controls. In the MicroLab, four different experimental modules are implemented. Therefore, the user interfaces are designed along with the existing experimental modules. As an example, the user interface of Robotic arm experiment is shown in the Fig. 1. MicroClient and MicroServer are designed by using C# programming language.

When the MicroClient program is run, the user student is asked to provide username and password for authentication. If the user doesn't have an account, it is still possible to access the session as a guest user. When the student enters the session as a guest, he/she will be given an automated sequential guest number to distinguish the new guest user from the other guest users.

* Accepted 21 February 2004.


Guest1: Connected to MicroLab Server
S.D.Ü Microlab
Experiment 3
Distance Education Laboratory
Outputs
P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.0
Inputs
P2.7 P2.6 P2.5 P2.4 P2.3 P2.2 P2.1 P2.0 0A
Klip:nt 01:09
Guest1
Guest2
Pass
Start Logging
Private
08:27
Lobby
Exit
File
Program
Start
Reset
Connected!
Send

Guest2 has joined the Chat
Guest1: hi

Fig. 1. User interface for an experiment.

After authentication, the student is lounged to the main entrance called lobby which lists the available experimental modules. The lobby also provides web links to the information for experimental modules. At this stage, the user is asked to choose one of the experimental modules by clicking joint buttons next to the experiment names. After clicking one of the joint buttons, students enter the experimental module interface. First connected student is selected as the operator of the experimental module by the MicroClient. The operator is the only one who can program and conduct the experimental module. Other present users are only able to see the results of the experiment and discuss it through using chat tool. As shown in the Fig. 1, the user interface is designed to allow students to chat with each other. Operator session-time is limited to 10 min, and can be adjusted by the MicroLab server operator. Remaining time is shown on the user interface in minutes and seconds. When the time is up, the next available student who connected the experimental module after the first student becomes the new operator. Another way of passing control of an experimental module is to choose a student from the user list and give control to the chosen student by clicking on the pass button. The operator is the only user who can use the buttons. For the other users, unless they obtain the control of the experimental module, the buttons are inactivated.

When the student finishes his work on an experimental module, he/she can still attend another experiment. The only step students need to take for another experiment is to go to the lobby by clicking lobby button and to choose another experiment.

To start working on an experimental module, the student needs to upload his/her source hex file written for the 8051 microcontroller. In order to upload the file, the students must use the file button to locate the source file. Only Intel hex format is acceptable. Any other file formats will be rejected by the MicroClient. Client program automatically uploads the hex file to the experimental module via the MicroServer program. If the student logs off or passes control to another student, his/her program is automatically deleted from the experimental module.

The switches and LEDs are designed to visually monitor the experiments. As the operator turns the input switch on/off, all the users connected to the experimental module receive the same status of the input switch. The same procedure is applied to the outputs as well. Whatever the operator's 8051 program running on the experimental module is, the same result appears on the output LEDs on all MicroClient software. Since the robotic arm and LCD experiment need visual inspection, user interface(s) on these modules are designed to receive the live video of the experiments.

MICROLAB HARDWARE ARCHITECTURE

As Fig. 2 shows, the designed MicroLab is composed of two main parts. The first part is the substructure by which students are connected to the server. The second part is the substructure that connects server to the experimental modules. The first part consists of a server equipped with an Ethernet card that uses IEEE 802.3 standard and campus network which provides connection to the

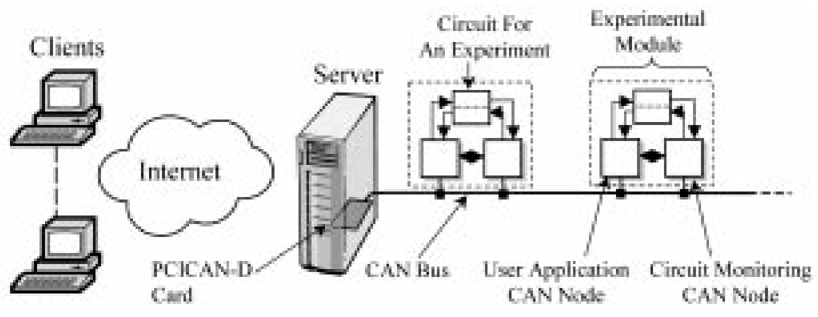

Clients
Internet
Server
Circuit For An Experiment
Experimental Module
PCICAN-D Card
CAN Bus
User Application CAN Node
Circuit Monitoring CAN Node

Fig. 2. Hardware architecture.

Internet through ULAKNET, a backbone network for the universities in Turkey [10].

In the second part, PCICAN-D card, created by KVASER Company, is used in the server for connecting the server to the experimental modules. This PCI card provides a real-time communication protocol called CAN (Control Area Network) between experimental modules and the server.

Experimental modules consist of two CAN nodes as shown in Fig. 3. One of the two CAN nodes, User Application CAN Node, is for downloading and running the student's source code. The second node, Circuit Monitoring CAN Node, is for monitoring the circuit and executing the user's input parameters. In order to communicate experimental modules to the server, each node must use the same communication protocol as PCICAN-D card uses.

The device used in the experimental module must also provide remote programming to allow students to send source codes to the experimental modules. Both CAN nodes use 89C51CC01 Microcontroller. The 89C51CC01 is the first member of the ATMEL's CANary TM family of 8-bit microcontrollers support CAN network applications and it provides remote programming via the CAN bus or the UART [11]. Flexible In-system Programmer (FLIP), a software program, is provided by Atmel in order to the 89C51CC01 device using UART or CAN bus. Unfortunately, FLIP software is not capable of programming the CAN nodes via 'PCICAN-D' card. Therefore, programming software for the CAN nodes is designed for PCICAN-D card and integrated into MicroServer. Figure 4 shows the example experimental modules.

CONTROL AREA NETWORK

The Control Area Network (CAN) is a serial data communication protocol primarily used for real-time automotive applications. It is based on a multi-master bus configuration which incorporates the first two layers of the ISO/OSI model: the physical layer and the data link layer. The first silicon CAN chip became available in 1987 after an improved investigation of the communication protocol. It has been approved by ISO (ISO 11898) as a high speed in-vehicle network (>125 KBit/s) [12]. Currently, the CAN protocol is used in industrial applications as well as in vehicle applications. Aircraft washing-systems, robot control, concrete paving systems, and sensor-based solutions for automation are some of the examples of CAN applications by industry [13, 14].

The CAN uses the CSMA/CR (Carrier Sense Multiple Access with Collision Resolution) access method with a bit-wise arbitration. The arbitration guarantees the delivery of the highest priority frame without delay even in a collision situation.

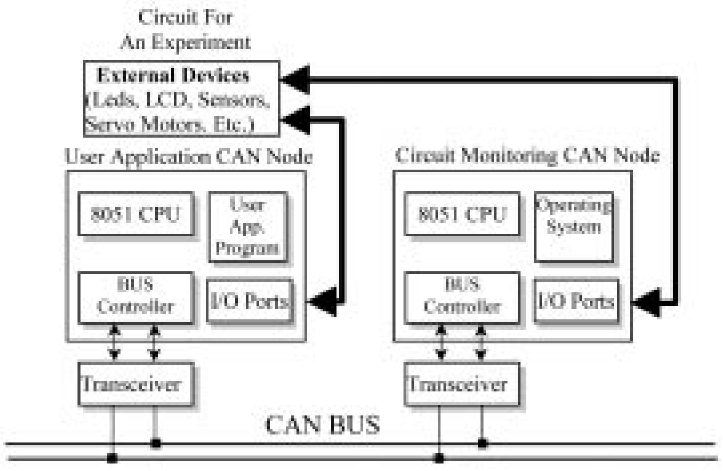

Circuit For An Experiment
External Devices (Leds, LCD, Sensors, Servo Motors, Etc.)
User Application CAN Node
8051 CPU
User App. Program
BUS Controller
I/O Ports
Transceiver
Circuit Monitoring CAN Node
8051 CPU
Operating System
BUS Controller
I/O Ports
Transceiver
CAN BUS

Fig. 3. Experimental module architecture.

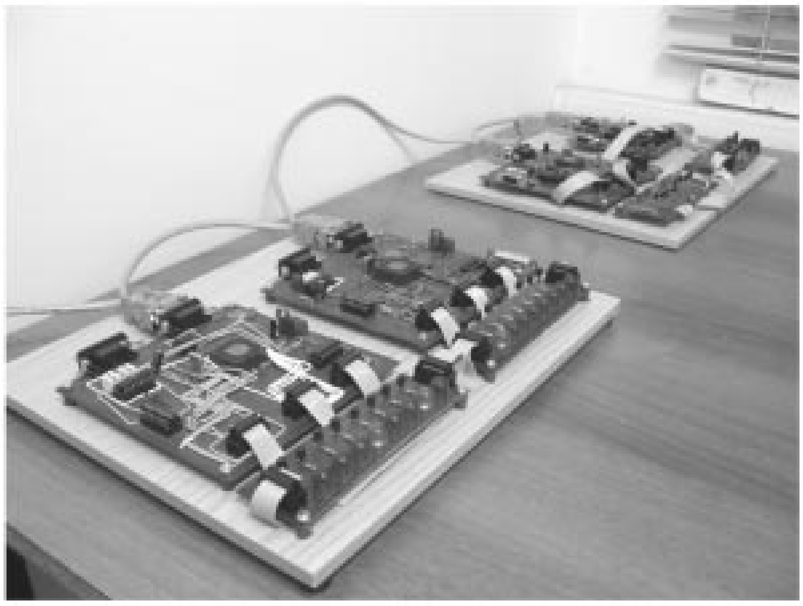


Figure 4. General I/O Experimental Modules

Each message used in the control area network has a unique identifier defining the type of data such as engine speed, temperature, pressure, or any data in an application. Furthermore, the digital value of the identifier also automatically implies the message priority for the bus access [15]. Identifiers used in this project are shown in Table 2 .

The number of nodes to be connected physically in a CAN network depends on the CAN transceiver parameters [16].

SOFTWARE ARCHITECTURE

The software architecture of MicroLab is illustrated in Fig. 5. Control of the experimental modules is implemented by employing both designed MicroServer control functions and the experimental module operating system.

For each experimental module, there is a unique control function on the server that converts user TCP commands into CAN messages and vice versa. To conduct experiments, user TCP commands sent by the MicroClient must be converted into CAN messages and then the converted CAN messages must be transferred to the CAN Bus. The transfer functions have been accomplished by using KVASER's Canlib SDK (Software Developer Kit) C# libraries.

As described above, each CAN message has a unique identifier. The identifiers used in this project define the CAN message behavior according to the designed operating system running on each experimental module. For example, if the experimental module operator presses the port switch icon, MicroClient converts the switch objects data into TCP command as 'IPORT' and transfers it. After the server receives the 'IPORT' message, it converts the message into CAN ID as 012h. This data can only be obtained as a result of operating the system on experimental module1. The other experimental modules receive the message with ID 012h but immediately discard it

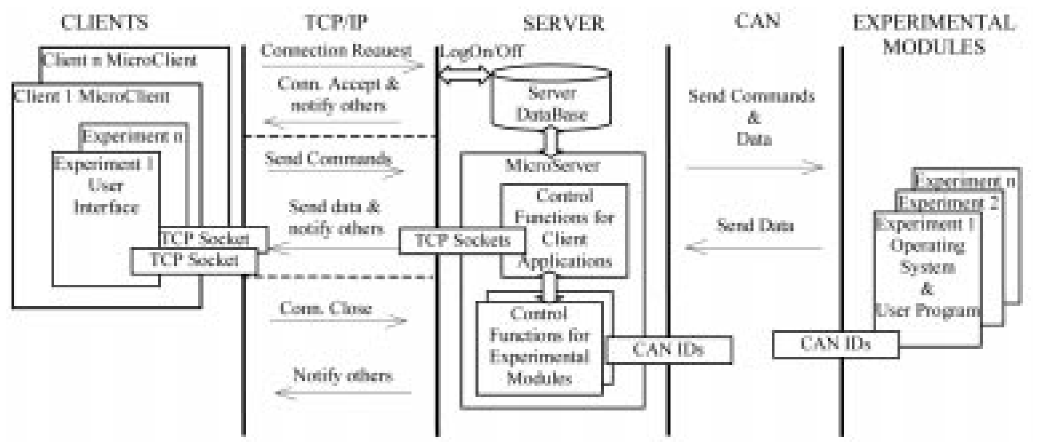
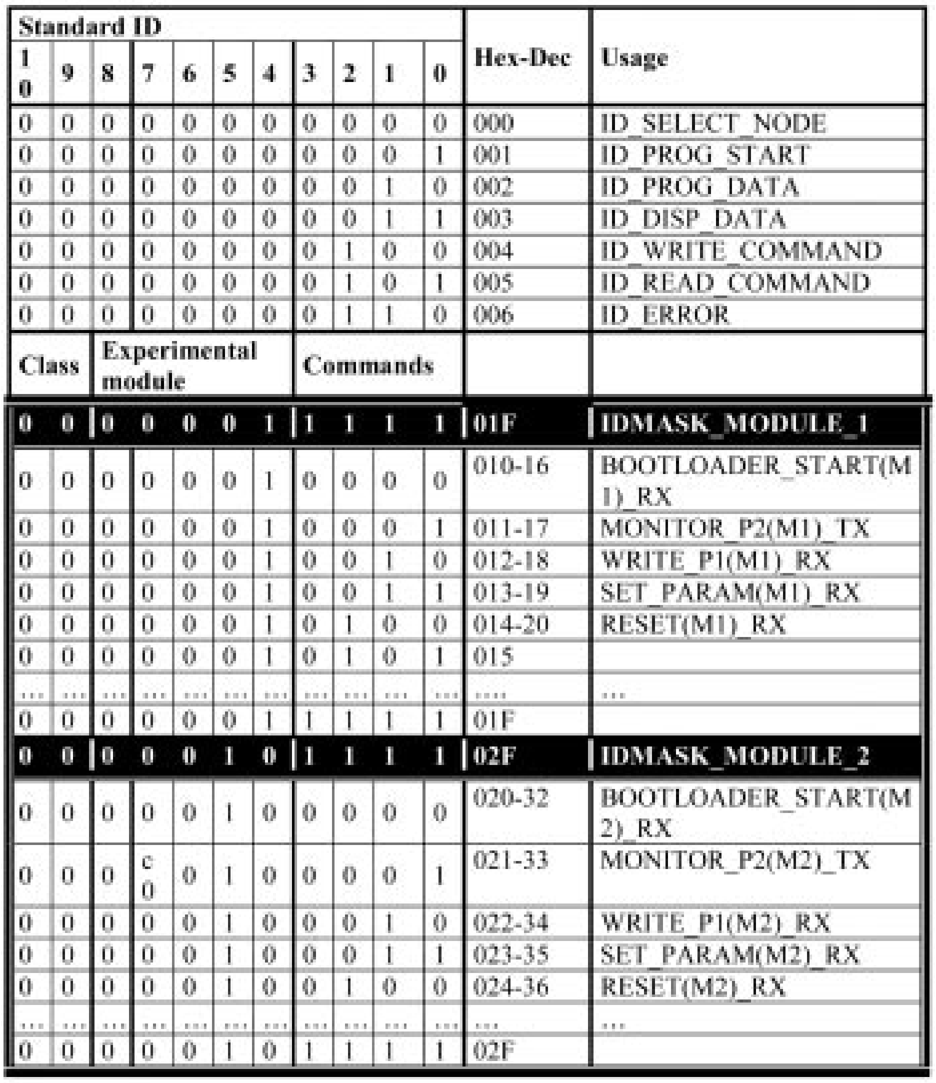


Fig. 5. Software architecture.

Table 1. Client and server commands

TCP command name	Sender	Activity
CONN	Client	Connection request
JOIN	Server	Connection accept
LIST	Server	Send User List to connected client
GONE	Client	Abort connection
CHAT	Client	Send text message
CHAT	Server	Transfer text to clients
PRIV	Client	Send private text message
PRIV	Server	Transfer private text message to client
PASS	Client	Pass control of module
PASS	Server	Pass control of module
IPORT	Client	Send switches status
SWTC	Server	Transfer switches status to clients
LEDS	Server	Transfer experimental module's data to clients
START	Client	Start Experimental module
RESET	Client	Reset Experimental module
PROG	Client	Start programming the experimental module
FILE	Client	Upload user program to server
OK	Server	Transfer completed module programmed

Table 2. Identifier MAP of experimental modules



Standard ID											**Hex-Dec**	**Usage**
10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**		
0	0	0	0	0	0	0	0	0	0	0	000	ID_SELECT_NODE
0	0	0	0	0	0	0	0	0	0	1	001	ID_PROG_START
0	0	0	0	0	0	0	0	0	1	0	002	ID_PROG_DATA
0	0	0	0	0	0	0	0	0	1	1	003	ID_DISP_DATA
0	0	0	0	0	0	0	0	1	0	0	004	ID_WRITE_COMMAND
0	0	0	0	0	0	0	0	1	0	1	005	ID_READ_COMMAND
0	0	0	0	0	0	0	0	1	1	0	006	ID_ERROR
Class		**Experimental module**					**Commands**					
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**01F**	**IDMASK_MODULE_1**
0	0	0	0	0	0	1	0	0	0	0	010-16	BOOTLOADER_START(M1)_RX
0	0	0	0	0	0	1	0	0	0	1	011-17	MONITOR_P2(M1)_TX
0	0	0	0	0	0	1	0	0	1	0	012-18	WRITE_P1(M1)_RX
0	0	0	0	0	0	1	0	0	1	1	013-19	SET_PARAM(M1)_RX
0	0	0	0	0	0	1	0	1	0	0	014-20	RESET(M1)_RX
0	0	0	0	0	0	1	0	1	0	1	015	
…	…	…	…	…	…	…	…	…	…	…	….	…
0	0	0	0	0	0	1	1	1	1	1	01F	
0	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**02F**	**IDMASK_MODULE_2**
0	0	0	0	0	1	0	0	0	0	0	020-32	BOOTLOADER_START(M2)_RX
0	0	0	c 0	0	1	0	0	0	0	1	021-33	MONITOR_P2(M2)_TX
0	0	0	0	0	1	0	0	0	1	0	022-34	WRITE_P1(M2)_RX
0	0	0	0	0	1	0	0	0	1	1	023-35	SET_PARAM(M2)_RX
0	0	0	0	0	1	0	0	1	0	0	024-36	RESET(M2)_RX
…	…	…	…	…	…	…	…	…	…	…	…	…
0	0	0	0	0	1	0	1	1	1	1	02F	

since it doesn't belong to them. Client and server TCP commands are listed in Table 1.

TheCAN identifier is 11-bits long in CAN 2.0A and 29-bits long in CAN 2.0B [17]. In this project, CAN 2.0A is preferred since it has enough identifiers to maintain classes, experimental modules, and commands.

The two most significant bits are allocated to define the number of classes in the system. In this case 4 different classes can be implemented. Five of the 11 ID are allocated to define experimental modules. This provides 32 different experimental modules to implement in a class and also defines the size of the class according to the maximum simultaneous connections. Finally, 16 different commands can be implemented for each experimental module via the use of 4 least significant bits. Each message ID can have data from 0 to 8 bytes.

In order to see the result of the student's program running on the User Application CAN node, Circuit Monitoring CAN node must periodically transfer the status of the experiment to the client side through the server. This is done by periodically transferring xx1 ID on each experimental module to the server. To reduce the message traffic, the message is sent only if the status of the experiment changes. Sampling rate of an experiment can be adjusted by MicroServer through passing parameters to the experimental module using xx3 ID.

To program the User Application CAN node, message ID xx0 is used. The ID numbers from 000 to 006 is reserved for programming the 89C51CC01 microcontroller by the ATMEL. Therefore, these ID numbers can only be used for programming the chip. The required programming procedure is applied to the MicroServer according to the Atmel definitions [18].

CONCLUSIONS

This project introduces the multi-user remote microcontroller laboratory and discusses its applicability for engineering education purposes. Based on the methodology utilized in the paper, client and server side applications are developed. Client applications offer efficient interaction between students on the experimental modules. It enables students to access different experimental modules any time and from any location. Distributed architecture of experimental modules provide real-time simultaneous connections to the experiments. This design architecture can be applied and extended to conduct other types of experiments.

The experimental modules allow students to improve their programming knowledge by engaging them in real experiments. Using this approach also provides a fast learning process since it gives the opportunity to work with the experimental modules whenever needed.

The MicroLab is primarily designed for class-attending students at Suleyman Demirel University and can be accessed at the Web address http://microlab.sdu.edu.tr.

Acknowledgement—This project has received financial support from the Suleyman Demirel University under the grant of 03-YL-754.

REFERENCES

1. A. D. Rio, J. J. Rodriguez and A. A. Nogueiras, Learning microcontrollers with a CAI oriented multi-micro simulation environment, *IEEE Trans. Education*, **44**(2) 2001, pp. 76–86.
2. B. A. Alhalabi, M. K. Hamza and S. Anandapuram, Real Laboratories: an innovative rejoinder to the complexities of distance learning, *The Open Praxis Journal of International Council for Open and Distance Education*, Vol. 2, (1998).
3. C. C. Ko, B. M. Chen, J. Chen, Y. Zhuang and K. C. Tan, Development of a Web-based laboratory for control experiments on a coupled tank apparatus, *IEEE Trans. Education*, **44**(1) 2001, pp. 76–86.
4. C. C. Ko, B. M. Chen, S. Y. Hu, V. Ramakrishnan, C. D. Cheng, Y. Zhuang and J. Chen, A Web-based virtual laboratory on a frequency modulation experiment, *IEEE Trans. Systems, Man. and Cybernetics, Part C: Applications and Reviews*, **31**(3) 2001, pp. 295–303.
5. http://chem.engr.utc.edu/webres/stations/controlslab.html
6. S. You, T. Wang, R. Eagleson, C. Meng and Q. Zhang, Low-cost Internet-based telerobotic system for access to remote laboratories, *Artificial Intelligence in Engineering,* **15**(3) 2001, pp. 265–279.
7. E. G. Guimares, A. T. Maffeis, R. P. Pinto, C. A. Miglinski, E. Cardozo, M. Bergerman and M. F. Magalhaes, REAL—a virtual laboratory built from software components, *Proc. IEEE*, **91**(3) 2003, pp. 440–448.
8. S. H. Chen, R. Chen, V. Ramakrishnan, S. Y. Hu, Y. Zhuang, C. C. Ko and B. M. Chen, Development of remote laboratory experimentation through Internet, *Proc. 1999 IEEE Hong Kong Symposium on Robotics and Control*, Hong Kong (1999) pp. 756–760.
9. R. Moros, F. Luft and H. Papp, Virtual laboratory course in chemical engineering and unit operations (VIPRATECH) tutorials, simulations and remote process control*, Int. Conf. Computers in Education*, 2002, pp. 1447–1448.
10. http://www.ulak.net.tr/?lang=en
11. *Enhanced 8-bit MCU with CAN Controller and Flash Memory,* Datasheet 89C51CC01, Atmel (2003).
12. K. W. Young and R. T. Mclaughlin, Low-cost SLIO CAN-based body control system, *ICC'94, Int. CAN Conf. Proc.,* 1994, Germany, pp. 8-2–8-9.

13. K. D. Rupp and O.Wurst Putzmeister, Implementation of CAN system in truck-based aircraft washing system, *ICC'94, Int. CAN Conf. Proc.,* Germany, 1994, pp. 4-13–4-21.
14. J. Uphoff, Theories and practical experience with CAL-based industrial control, *ICC'94, Int. CAN Conf. Proc.,* 1994, Germany, pp. 6-2–6-31.
15. *Intel 82527 Serial Communications Controller Architectural Overview,* Intel, (1993)
16. H. Eisele and E. Jöhnk, PCA82C250/251 CAN transceiver, *Application Note (AN96116)*, Philips Semiconductors, Germany, (1996).
17. J. Randhahn and H. Beikirch, Controlling and monitoring CAN via Web interface, *CAN Newsletter*, (2003) pp. 68–69.
18. 89C51CC01 Datasheet, *CAN Bootloader,* Atmel, (2003).

Akif Kutlu is currently an Assistant Professor of Computer Systems Education, Faculty of Technical Education at Suleyman Demirel University in Turkey. He is the head of the department since 1997. He received his Ph.D. degree from Sussex University in 1996. His interests are industrial computer networks and microcontrollers.