Remote Laboratory Hardware Modules Based on Networked Embedded Systems

Darko Fudurić, Mario Žagar, Tomislav Sečen and Marin Orlić
University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia

***Abstract*—Networked embedded microcontrollers integrated into remote laboratory are proposed. Hardware and software architecture responsible for functioning of the laboratory is discussed. Compatibility with existing networks is mandatory, so common TCP/IP stack layer is introduced. Open remote laboratory, based on embedded systems is accessible from anywhere on the Internet.. It is suitable for practical education in microcontroller fundamentals.**

***Index Terms*—e-Learning, Embedded systems, Remote laboratory.**

I. INTRODUCTION

Embedded, ubiquitous, pervasive, mobile, wearable, paintable are just a few keywords of modern computing. Total distribution and networking is a common denominator. Knowledge and understanding of basic principles and architectures of small computers, microcomputers, networked microcontrollers [13] is a necessity. One solution is to build massive laboratories, but in an educational environment (like we have at the University of Zagreb, Faculty of Electrical Engineering and Computing) with many students enrolled and curious to exploit the world of microcomputers through "hands on experience", building, maintenance, administrative organization and teaching support, this proves to be "mission impossible".

Unlike general purpose (PC) laboratories, in which many different exercises can be organized by simply replacing software, a microcomputer architecture [15] laboratory is specific and not very reusable.. Lack of physical space (not enough classrooms), low efficiency (one student working several hours with the equipment), lack of teaching staff, inconvenient time (early morning, late evening, big time gaps between the lectures and exercises), expensive maintenance (many wires, many people moving around) and many other reasons led to the idea of building a microcomputer laboratory available round the clock, all days a week, all weeks a year. And, of course, accessible from home or any other Internet enabled place. The first idea that naturally came to mind was "write or buy good software simulator installable on every PC, give it to the students and a lot of problems will be solved". However, there is no "hands-on experience" and it would be much like a cooking simulation - nobody would like to eat dishes prepared by the chef trained by a cooking simulator. You have to have a real experience. The second idea, to build a remote laboratory [12][14] for microcomputer systems and processes, came as a normal decision. The goal was to establish a laboratory in which everything is real except that you do not need to be there. In your client environment you must feel like you are there.

Several entirely different problems had to be solved such as the administration system, reservation system, a secure and unique access to the equipment (there is no possibility for time sharing of the small systems), visualization of the experiments, automatic examination and notification about the exercise results. And just when we think that we have counted in all the problems, a small and simple one arises – a student somehow wrote, downloaded and started binary program execution of the type:

```
START:    NOP
          JUMP START
```

and everybody (the student and teacher) is kilometres away. Who will press the Reset button or unplug - plug the power cord?

We have solved all the mentioned problems but their description would exceed the size of one article, and technologies and methodology used do not belong to a single aspect of computing. In this paper we have focused on the networked hardware modules in the implemented remote laboratory.

II. SYSTEM ARCHITECTURE

Microcontrollers are generally divided into simple modules with one type of simple serial communication interface [1] like RS-232 or I2C enabled communication and another type, more sophisticated, Internet enabled network microcontrollers. The first concept has been well-known for years, the latter is widely available recently. This is the reason we should not ignore the first group. In addition, smaller or simpler devices usually have less power dissipation, less architecture complexity, simpler maintenance, system implementation and evaluation process along with some features that are not necessary for educational purposes. For better understanding of course materials and exercises, students need general knowledge of microcomputer architecture with some elementary (basic) details, which is not specific for commercial vendor policy. Selecting one commercial architecture and one among dozen processors offered on the market that meets the majority of requirements noted above is not easy.

Proposed system architecture is shown in Fig. 1. As an exercise, students have to access the core of a remote laboratory called VLAB server. They can connect from one of the computers on Faculty premises or from a

computer on the other side of the firewall. In the first case, Faculty network security policy can be applied directly, but in the second case we deal with the so-called open Internet connection. One of the side effects is the necessity to restrict access to the remote laboratory resources (because of malicious activities and threats on the open Internet connection).

After successful access to the VLAB server, students have an account and exercises related options for term-based reservations. Within the reserved term (time slot) students have the possibility to upload and execute the code and see execution results. After several iterations, the final code should be uploaded before exceeding the total time limit allowed for the exercise.

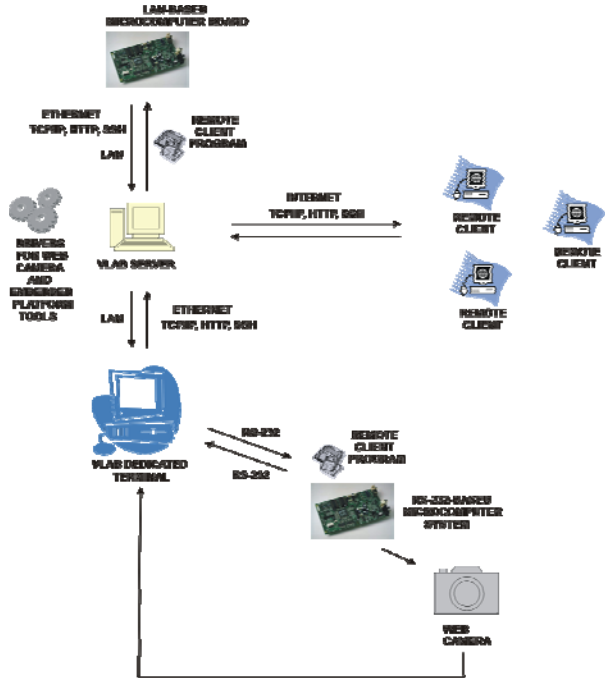

Figure 1. Remote laboratory concept

Access to resources is determined by the applied exercise profile. In the case of LAN-based microcomputer boards, resource access is enabled directly through the LAN, because target module has TCP/IP stack and already developed communication API-s and related tools. In the case of simpler boards with some type of serial interface instead of a LAN interface, LAN-to-serial bridge is required. We propose a VLAB-dedicated terminal (VLAB DT). This is a PC-based computer with installed tools [2], such as compilers, loaders, programmers, etc. and interfaces related to the equipment attached on it. VLAB DT can receive programs from remote students through VLAB server and upload them on the related equipment, according to the resource profile policy. Furthermore, VLAB DT can send commands in order to execute uploaded programs, and get back the execution results. The results are sent as data or video streams from an internet camera.

VLAB server is the focal point of remote laboratory that consists of ,many modules. This allows for new embedded networked modules to be attached when needed and in the same time preservs the reservation system.

From the user point of view, there are three steps in the connection to the VLAB server, shown in Fig. 2.

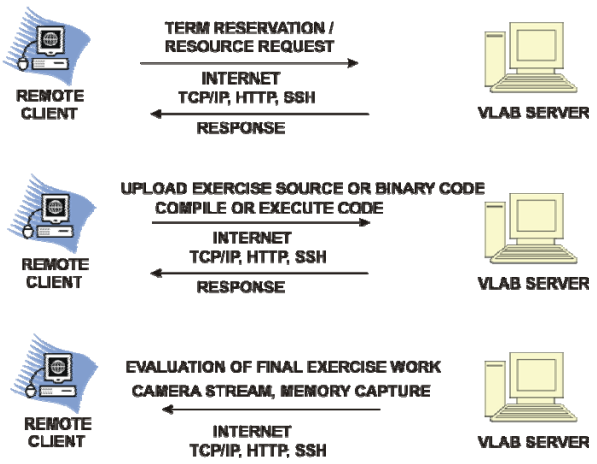

Figure 2. Three steps of VLAB server usage

In the first step, clients send a reservation requests to the VLAB server. The requests are registered and logged on the VLAB server. Global resource scheduler redirects each request to the relevant resource group (LAN or serial). When the request for execution arrives, resource time scheduler starts the processing. This is the beginning of step 2. Command and data messages are sent to the resource, equipped with a mechanism for accepting (user program) and executing such messages, as well as for retrieving execution results during the step 3.

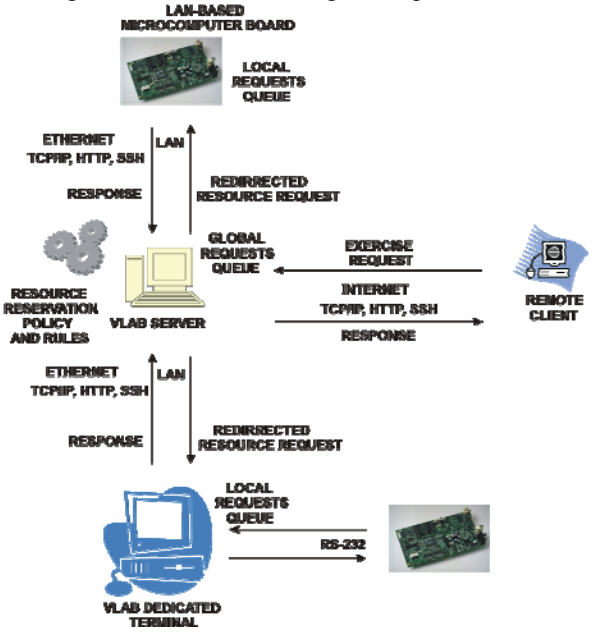

Figure 3. VLAB resource reservation policy

After the results are retrieved, resources are released and the next request processed. Therefore, users can subsequently execute programs on the remote microcomputer system.

Frequency of possible executions depends on the nature and complexity of the program, peripheral unit delays and environmental responses for a specific problem. We define system inertia as the time between sending binary program to the system and getting data and video stream back. This lasts about a few seconds in the case when program performs only simple operations in the memory, such as adding two operands or blinking LED diodes.

Resolving phase 2 and phase 3 depends on the microcomputer architecture, as it requires mechanisms for capturing all microcomputer resource states after a program is executed and returning back to the server. In order to return the current system state, user program has to include special service routines, which can provide memory stream output to the VLAB dedicated terminal. It is assumed that the user will include previously prepared functions in binary code during or after the compilation, but the problem is that these routines allocate some amount of program and RAM memory. The other problem might be if a user's main program enters into an infinite loop, as explained in the introduction. In that case, service routines will never be called.

Because there is a need for stopping execution and returning results, we need emergency mechanisms for resolving potential user infinite-loop-like programs. This can be accomplished by using hardware debugging interface such as JTAG. This interface enables supervision of user program execution through tracepoints, breakpoints, step-by-step execution, register, memory and processor state capture. There are many microcomputers on the market without JTAG port, but nearly all of them have some kind of a serial interface. We propose replacing the JTAG port by a serial port combined with firmware that runs in the microprocessor supervisory mode. This requires writing a supervisory program that can clear, upload, execute and block any user program, according to the given JTAG-simulated commands through the serial port. The accomplishment of this task requires self-programmable processor architecture feature, which means that the program can change itself. The supervisory program has to provide memory capture of the user memory area and special function registers (SFR) which are used in order to control the on-chip peripherals. Implementation depends on concrete microcomputer architecture, because we need to simulate JTAG hardware supervisory features by software. We have found that microcomputers with boot loader capability are a suitable replacement and the solution is given in the next chapter.

III. APPLICATION AND IMPLEMENTATION

Atmel AVR family architecture [7] is one of the preferred architectures suitable for educational and experimental purposes that we have in our lab, with several practical exercises. Almost every microcontroller from this family has FLASH, EEPROM and SRAM memories, auxiliary units such as counters, timers, memory, pulse-width modulation units, RTC clocks, A/D converters [6], PWM channels etc. Replacement with a serial port and boot loader is possible because many processors such as ATMEGA64 have a boot loader section placed at the end of FLASH memory. This section is suitable for our supervisory program named AVR Boot Loader (BL). Flow diagram is shown in Fig. 4.

The reset source can be distinguished from the BL or user program. For example, ATMEGA64 processor has five reset sources: Power-on Reset, External Reset, Watchdog Reset, Brown-out Reset and JTAG AVR Reset. We use the Power-on Reset and Watchdog Reset. In either case, BL is executed first.

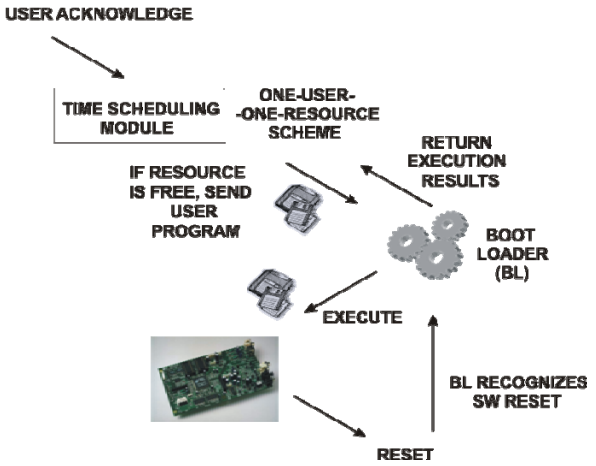

Figure 4. Supervisory boot loader architecture

In the event of power-on reset, the boot loader sends the "ready" message through RS-232 or USB port to the time scheduling module which responds with user binary code and the expected execution time interval. After receiving the user program, the BL overwrites the user program area with a received program (AVR architecture has a self-programmable feature), starts the Watchdog timer with the expected execution time interval just before it jumps to the first instruction of the user program. After the execution, every user program has to enter into the infinite loop or sleep mode, so the watchdog timer started by the boot loader will always reset the processor and enter into the boot loader program again. The BL then sends a command that signals the end of the execution to the time-scheduling module, and waits for memory capture commands. Capture commands regarding the memory type (FLASH, EEPROM or RAM) and memory address range are executed and memory image in INTEL HEX format is returned to the VLAB server. The time-scheduling module compares the returned memory and registers with the correct or expected values, and provides laboratory result marks.

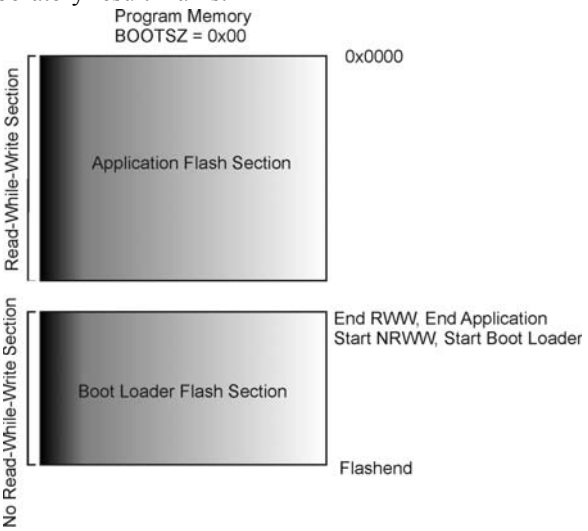

Figure 5. Boot loader and Application section

Tiny-based platform does not need the time scheduling module because it has multi-threaded OS which handles the user connections automatically. The AVR platform does not have an operating system.

The last point that should be noted is how to protect the BL from being overwritten by a user program. Regarding the different memory types and read/write permissions, AVR architecture has 11 levels of protection. We select one of the two suitable protection levels that protect the BL from being erased or overwritten from the application section (Fig. 5).

We have implemented a few practical exercises [3]. The target system consists of several parts: LED array with four LED BCD segments, STEP motor with drivers, temperature sensor and 16x2 text LCD unit, all shown in Fig. 6.

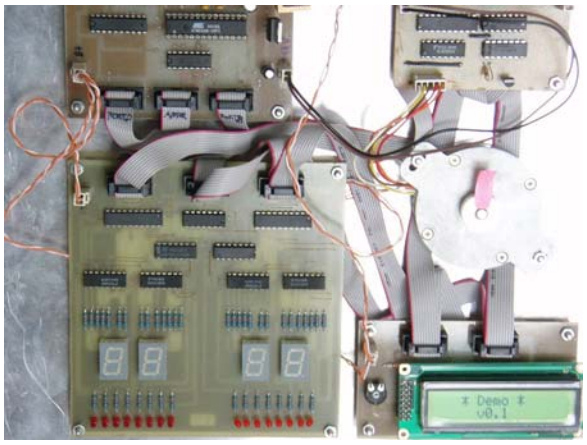

Figure 6. VLAB system hardware based on Atmel AVR microcontroller family

The main board consists of the target microcontroller ATMEGA8 from Atmel2 AVRTM family. Fig. 7 shows the physical realisation of the VLAB system concept.

The main board is connected to the other parts via a multiplexed bus, because it does not have enough I/O lines to support all the connected modules at once. Target microcontroller can be reprogrammed directly from the VLAB server, via the RS-232 interface, using the dedicated uploader tool.

VLAB DT server is PC with Linux Debian 3.1 OS, kernel version 2.6. Video LAN media server is used for video streaming.

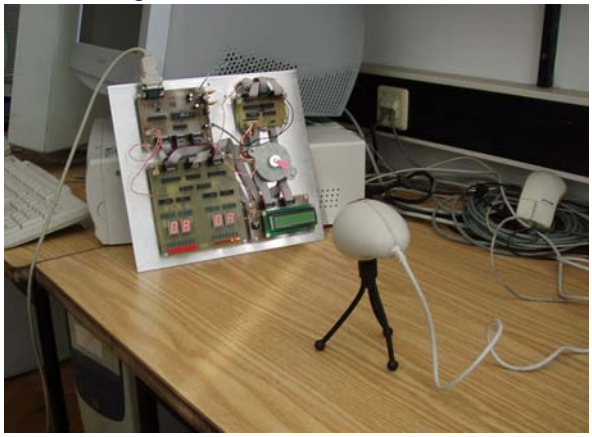

Figure 7. VLAB system hardware in action

Modules based on LAN interface used as parts of VLAB are shown in Fig. 8. One of them (Rabbit2000TM embedded module) is used for "hot reset" of the other (TINI S400 series module), if this system hangs up.

Figure 8. TINI S400 and RABBIT2000TM modules

IV. CONCLUSION

When speaking in terms of distributed and distance learning in computer science, remote laboratories are an important facility for learning and practicing. Various systems can be controlled remotely, allowing students to practice on real systems. This is available from anywhere and at any time. Reserving a real, hardware resource, a microcontroller of some kind, can be done in advance or dynamically in order to prevent collisions between different users. After making a reservation, students program remote resources in any supported language and tool. The exercise, if successful, can be graded automatically, thus reducing the workload of the teaching staff.

Web-based experiments have been developed for both engineering/students and science experiments. That way it is much easier and cheaper to learn or demonstrate experiments, since there is no need for the rather expensive equipment or for students to travel to the lab. The remote laboratory enables various models and simulations that are not feasible with regular equipment.

Remote laboratory can be applicable in various ways and fields, such as electronics, computer engineering, chemistry, physics, probability and statistics, medicine, robotics etc.

The broad-spectrum solution to the distance learning or working exercises is not currently resolved, because our system depends on the laboratory equipment features and configuration, as well as on the exercise problem domain. This means that solving distance learning or working problem for another type of laboratory, for example, chemistry or biology, cannot be done in the same way it is currently done, because the input vectors to the system and output results and data from the system are naturally different. There are two rules which should be fulfilled to make this fact irrelevant:

- transformation from the designed digital input vectors into problem domain input vectors can be done transparently and uniquely
- the effects or responses produced in the problem domain as a response to the input vectors can be transformed into the result domain transparently and uniquely.

It should be noted that transparently and uniquely means that the function between our digital domain and a

specific analog system domain can be implemented successfully and with expected performance, which should not depend on the digital domain. The digital domain can be improved in the future with additional or improved hardware, algorithms and methods for acquiring and fetching data from other system output (A/D conversion), which can also be achieved with better representation of digital input vectors via sophisticated D/A converters etc. For example, there may be a need for wide-range response types from the target system such as analog input signal frequency band and peak-to-peak level, or faster sampling frequency etc. If we find how to generalise these two subsystems in the future, we can say that we have reached a broad-spectrum distance learning and working solution.

ACKNOWLEDGMENT

The authors would like to thank all the people involved for extensive discussions in the initial phases of the work, participation and support during all related working activities and for comments provided on the early versions. Unfortunately, we cannot list them all.

REFERENCES

[1] T. Noergaard, "Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers," Elsevier, pp. 257–283, 2005.

[2] A. S. Berger, "Embedded Systems Design: An Introduction to Processes, Tools and Techniques", CMP Books: 1st edition , December 15, 2001.

[3] Embedded System Design: A Unified Hardware/Software Introduction, Wiley; I.S.ed edition, October 17, 2001

[4] P. Marwedel, "Embedded System Design", Springer; 1 edition, December 14, 2005.

[5] Marin Orlić, Tomislav Sečen, Mario Žagar and Darko Fudurić, "Enhanced learning by experimentation with remote laboratory equipment", REV 2007.

[6] Fudurić Darko, Davor Antonić, Žagar Mario, "Features of the Embedded Computer System for Data Acquisitions in Humanitarian Demining", MATEST 2005. NDT for the public benefit, 2005.

[7] R. H. Barnett, S. Cox, L. O'Cull, "Embedded C Programming And The Atmel AVR", Thomson Delmar Learning; 2 edition, June, 2006.

[8] Aravind Kumar, Alagia Nambi, "Implementation Of Mobile Information Device Profile On Virtual Lab," itcc , p. 612, 2003.

[9] Paul I-Hai Lin, Melissa Lin, "Design and Implementation of an Internet-Based Virtual Lab System for eLearning Support," icalt , pp. 295-296, 2005.

[10] S. Kolberg, T. A. Fjeldly, "Web Services Remote Educational Laboratory", International Conference on Engineering Education, October 16–21, 2004, Gainesville, Florida.

[11] R. Berntzen, J. O. Strandman, T. A. Fjeldly, M. S. Shur, "Advanced solutions for performing real experiments over the internet", International Conference on Engineering Education, August 6 – 10, 2001 Oslo, Norway.

[12] J. Rusten, S. Kolberg, "Online FPGA laboratory for interactive digital design", International Conference on Engineering Education, October 16–21, 2004, Gainesville, Florida.

[13] H. Shen, M. S. Shur, T. A. Fjeldly and K. Smith, "Low -cost modules for remote engineering education: performing laboratory experiments over the internet", 30th ASEE/IEEE Frontiers in Education Conference, October 18 - 21, 2000 Kansas City, MO.

[14] T. A. Fjeldly, M. S. Shur1, H. Shen, and T. Ytterdal, "Automated Internet Measurement Laboratory (AIM-Lab) for Engineering Education", 29th ASEE/IEEE Frontiers in Education Conference, November 10 - 13, 1999 San Juan, Puerto Rico.

[15] T. A. Fjeldly, J. O. Strandman, R. Berntzen, "LAB-on-WEB – a comprehensive electronic device laboratory on a chip accessible via internet", International Conference on Engineering Education, August 18–21, 2002, Manchester, U.K.G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," *Phil. Trans. Roy. Soc. London*, vol. A247, pp. 529–551, April 1955.

AUTHORS

D. Fudurić, BSc is with the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering (e-mail: darko.fuduric@fer.hr)

M. Žagar, PhD is with the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering (e-mail: mario.zagar@fer.hr).

T. Sečen, BSc is with the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering (e-mail: tomislav.secen@fer.hr).

M. Orlić, MSc is with the University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering (e-mail: marin.orlic@fer.hr).

Manuscript received July 18, 2007. This work is supported in part by the Croatian Ministry of Science, Education and Sports, under the research project "Software Engineering in Ubiquitous Computing".

Published as submitted by the author(s).

Paper presented at REV2007 conference, Porto, Portugal, June 2007.