SoftCOM 2000
events

WORKSHOP ON CONTEMPORARY COMMUNICATIONS

SoftCOM 2000

October 10-14, 2000
Split, Rijeka (Croatia)
Trieste, Venice (Italy)

Edited by:

Nikola Rožić
Dinko Begušić
Marija Vrdoljak
University of Split
FESB Split

Published by FESB-Split

Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture
University of Split
R. Boškovića b.b.
HR-21000 Split
Croatia

Text design: Jurica Ursić, Hrvoje Dujmić

Cover design: Zoran Perdić Lukačević

CIP – Katalogizacija u publikaciji
Sveučilišna knjižnica u Splitu

UDK 621.39 : 004](063)

INTERNATIONAL Conference on Software, Telecommunications and Computer Networks (8 ; 2000 ; Split...[etc.])
Workshop on contemporary communications / SoftCOM 2000, Split – Rijeka, Trieste – Venice, October 10 – 14, 2000 ; edited by Nikola Rožić, Dinko Begušić, Marija Vrdoljak. – Split : University of Split, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, 2000. – VI, 173 str. : ilustr.; 29 cm

Na vrhu nasl. str.: SoftCOM 2000 events. – Prema predgovoru, knjiga je radni priručnik osme konferencije. – Bibliografija iza svakog rada. – Kazalo

ISBN 953-6114-39-9

1. Rožić, Nikola

ISBN: 953-6114-39-9

EMBEDDED WEB SERVERS

MAJA BOŽANIĆ, ZVONKO ČIĆ, DARKO STIPANIČEV

UNIVERSITY OF SPLIT
Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture
R. Boškovića bb, 21000 Split
tel. 305 828, 305 813, fax. 563 877
e-mail: maja@fesb.hr, zcic@fesb.hr, dstip@fesb.hr

Abstract: *Embedded system applications are moving toward interconnection and networking. Integrating distributed systems through Internet is possible using dedicated gateways or web enhanced embedded servers. In this article hardware and software architecture of embedded web servers is analysed. Processor, memory and interface requirements are defined. Software development tools are specified. TINI embedded web server from Dallas Semiconductors is presented. The experimental data acquisition system is developed and embedded web server functionality is verified.*

Keywords: Embedded system, Embedded web server, Internet, TINI

INTRODUCTION

Almost any device containing a computer can benefit from a network connection in terms of functionality and both exploitation and initial price. In the last few years with the rapid growth of the Internet and World Wide Web as a public resource, people have recognised the usefulness of attaching various devices to the network. This makes them accessible from almost anywhere for the purposes of data display, remote control, testing, configuration, etc. Using WWW browser on workstation computer allows us to use large screen, menus, buttons and helpful information to guide us through the process of monitoring and control of the device instead of using dedicated complex grafical presentation software products. To date, the machinery and the network have remained largely separate entities. Embedded processors have communicated within local limits to control lighting, heating and cooling units, doors, etc. These could be connected to Internet through complex devices like gateways, Figure 1.

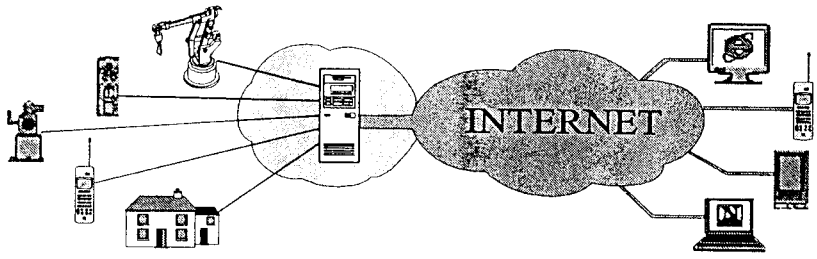

Figure 1. Device networking infrastructure

Embedded web servers are designed to bridge the gap between sensors and Internet to allow machinery to talk to user workstations using standard network protocols like TCP, IP and HTTP without making the bridge too big or too expensive, Figure 2.

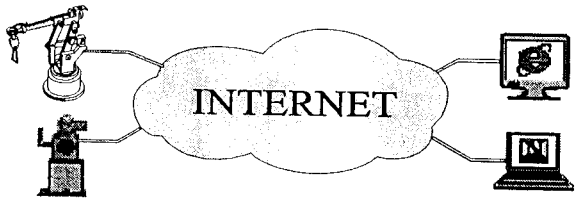

Figure 2. Embedded device connectivity

In this paper the embedded systems functioning as web servers are described. In Chapter 1. system design, protocols and software and hardware requirements for embedded web servers are presented. In Chapter 2. the real system is introduced. In Chapter 3. an application is illustrated through the embedded web server based monitoring system. The conclusions are given in Chapter 4.

1. SYSTEM DESIGN

Embedded systems require web server software that enhance their existing functionality without taking up vital system resources and without provoking costly software and hardware redesign. Web enabling of devices is possible in two different ways:

- by adding web server software to existing embedded system,
- by connecting another specialised device for that purpose.

Adding web server software to existing embedded system is inconvenient because embedded systems are cost constrained and they do not have much excess memory or computing power. Therefore, web server software that is being used has to be shrunk and written with many restrictions, and it must not influence original software operation in any way.

Connecting specialised devices bring many advantages and new possibilities to existing systems. These devices have their own processors and memory and are not restricted with "home system" capabilities. Connecting dedicated web servers to embedded systems is simple, through serial port that can be found in all embedded systems.

Requirements for embedded web servers are:

- Small memory footprint: Server must use very little memory and it must not fragment memory. Many embedded devices use simple memory allocators that

cannot manage memory effectively. This problem is usually solved by using statically allocated or pre-allocated memory blocks.

- Dynamic page generation: Since the content of the pages served will be status information and sensor readings part or all of the web pages will have to be generated on the fly.

- ROMable web pages: Since embedded systems do not have disk drives or any other memory storage units, web pages and other web content have to be stored in ROM or flash.

Reducing embedded web server capabilities to minimal set of necessary functions makes hardware requirements as well as energy consumption very small. Therefore dimensions and the price of the device are miniature compared to the desktop PC with full functionality preserved at the same time.

1.1. Protocols

Although TCP/IP (Transmission Control Protocol/Internet Protocol) is the main communication protocol that enables connection of different devices in networks, it is not implemented in full. Only the necessary parts are used and parts that are not needed for this sort of communication are excluded. Furthermore, TCP/IP protocol is basis for upper level network protocols which will be used in embedded systems. If the system uses RS-232C or modem link then point-to-point (PPP) or serial-line Internet protocol (SLIP) is needed. File transfer protocol (FTP) is used for uploading new files and programs to the system. Opposite to the receiver-driven information flow through Web browser, information flow could be device-driven. Using Simple Mail Transfer Protocol (SMTP) application can periodically send information via e-mail.

At the time being, there are at least three upper-level protocols, but although they are significantly overlapping, they still cannot interoperate [3].

Jini™ [4] was developed by Sun Microsystems, based on Java programming language. Jini promises well-defined space in which devices can support each other and procedures are very orderly laid out for devices to advertise and use capabilities on network.

HAVi [5] is a set of protocols and procedures defined by a consortium of consumer electronics to allow multivendor equipment to interoperate. This includes control information that should allow devices to send and receive data as well as control operation functions.

Universal Plug-and-Play is a protocol set defined by Microsoft. It is a set of protocol formats with basic rules for communication, including the use of XML [6] as a data interchange format. The specific data scheme for each message is defined by a consortium of companies with particular interest in that type of device.

This is the field in which embedded systems can be developed at most. Formulating a new, unified protocol that will allow full interaction between all sorts of devices no matter who produced them is the next step that will advance the development of this technology.

1.2. Software

A minimal set of software for embedded Internet system includes an operating system and application software, a http server, a TCP/IP stack, and drivers for communication hardware. The use of Internet allows the embedded system to offer substantial online capabilities without using system resources. With HTML pages pointing to other network locations, the system can offer more online documentation and richer graphics than the system hardware could otherwise support. In addition, supplying the raw data to more powerful "partner" on the network and presenting the results of "partner" calculations can virtually increase the systems processing power. As we can see with clever programming and the use of Internet, systems capabilities can be virtually increased to a higher level. The presence of embedded web server must not obstruct the systems primary real-time operation in any way and therefore system response must be compromised to main function.

The Internet software should not be the limiting factor for the speed of data flow between the server and the browser. The HTML pages should not be burdened with extensive graphics. The presence of too many graphical elements consumes much more memory than text and therefore takes much more time to download to browser. The page download time should be acceptable for the user, not longer than couple of seconds. [8]

1.3. Hardware

Hardware requirements result from the given selection of requirements and a set of software to resolve those requirements. The minimal setup for network enabling of embedded systems includes 8-bit microprocessor, Ethernet and/or serial interface and enough memory to store and run applications. No I/O devices are needed since all this operations are managed through Internet. Further system improvements depend on desired system price and capabilities. These 8-bit devices can address up to 16MB memory and often include hardware-accelerated interpreters for higher-level languages, which make them adequate for this purpose. Memory

storage devices such as hard disk drives are definitely optional and most of these devices will run software directly out of ROM or flash instead of loading it into RAM memory. This enables the system to boot in just a second or two as opposed to the minutes that a desktop Windows system can take.

1.4. System integration

In a new system design, system hardware will be chosen according to the overall requirements of the system. Application software needs to be developed in order to achieve the required system functionality. System designer must compromise between hardware and software solutions for each problem. Hardware solutions are easier to design, faster and more accurate in operation. Software solution results in smaller production cost and system modifications are simpler. Particularly, connection to Internet must be considered. When embedded processor is located in the vicinity of local network installation, 10Mbps Ethernet interface is chosen. Otherwise, serial port is used to transfer data over telephone or radio channel.

For the program development any software development kit can be used, with the additional libraries supplied by the hardware manufacturer. Since the software will be developed on a different platform, portability is very important for the embedded systems. Java, as object oriented, platform independent programming language, eliminates the need to recompile applications when moving to a different processor. Since a complete set of standard class libraries is available, there is less code to write and maintain. HTTP and FTP client support is built in along with support for handling URLs, IP addresses and name resolution.

To put a web server on a device, the server must be able to generate dynamic web content based on data from the device. Although it is not necessary to support a large number of users, the Web server must process multiple, simultaneous browser requests.

2. TINY INTERNET INTERFACE

On its broadest level, the networking problem concerns how to gather and route data from local monitoring devices for centralised readout and control. If data can be gathered and delivered to a processor, control commands can also be issued back out over the network. But first, a data transfer medium (interface) and a unified logic for dealing with disparate elements are needed. Between existing interface devices, minimal configuration suitable as a working interface needed for low-cost networking is TINI™ [9].

The TINI board is a Java™ computer that uses a TINI chip set plus 512 Kbytes commodity SRAM and interface circuitry in a 68-pin SIMM stick form factor.

TINI's three-chip chip set consists of a DS80C390 processor, 512 Kbytes Flash ROM containing the firmware and an Ethernet controller.

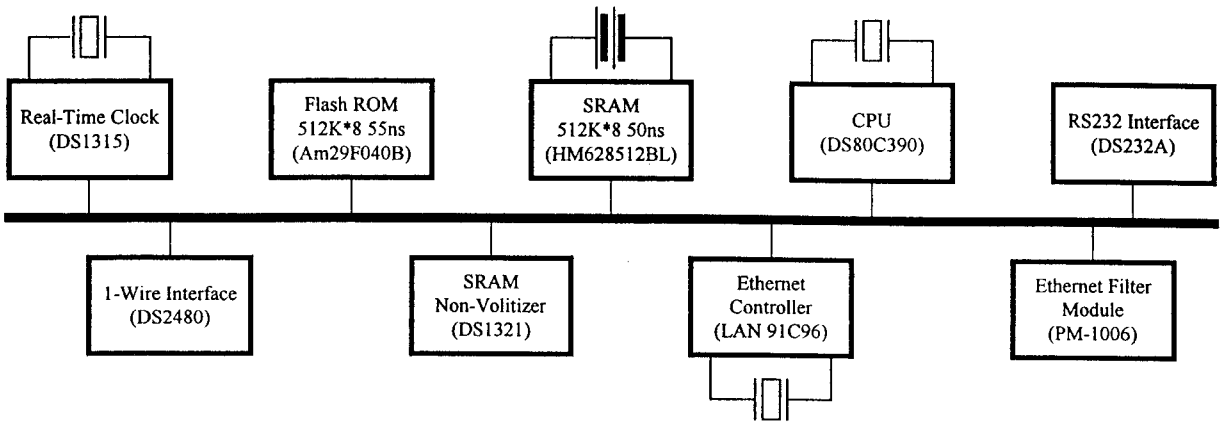

Figure 3. TINI hardware architecture

Optimised for the embedded Java environment, the processor supports 24-bit addressing, an 8/32-bit CPU/ALU, and high clock rates (~60 MHz and beyond), and other Java enhancements.

TINI's I/O ports include:

- Ethernet 10Base-T interface
- Dual 1-Wire® net interface
- CAN interface (with option for a second CAN interface)
- Dual serial port (one RS-232 level and one +5V level)
- I2C port
- Expansion bus allowing nearly unlimited parallel ports and miscellaneous digital and analog I/O.

TINI software is divided into categories: run-time code in Flash ROM (the RTOS, TCP/IP stack, Java™ VM and API packages) and high-level networking protocols (FTP, TELNET, DHCP, DNS), development tools (JDK) and Java applications.

The Java VM on TINI conforms to Sun's Embedded Java platform, version 1.1 of the Java API.

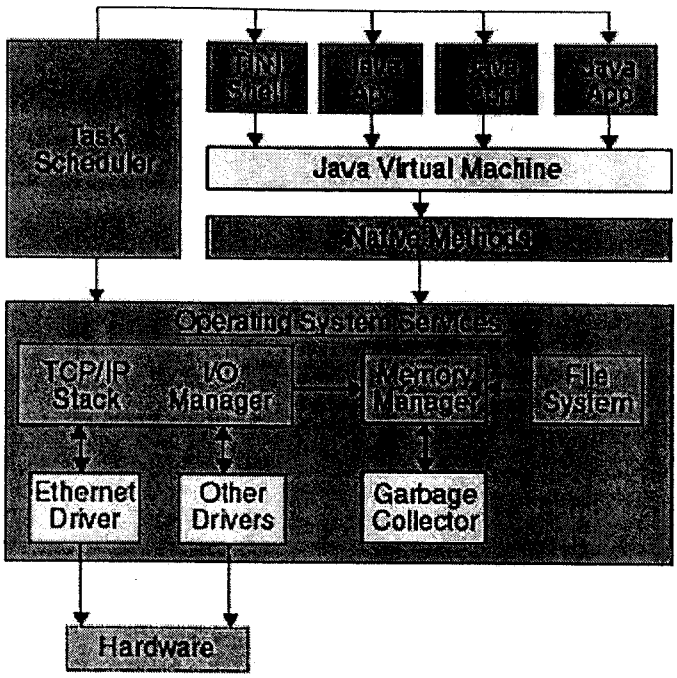

Figure 4. TINI software architecture

TINI provides a small command shell called "slush" which provides access to remote hosts. This Java application is started during the last phase of system initialisation. During construction, slush creates threads that maintain server sockets to listen for and accept FTP and Telnet client connection requests. When TINI is being used without a network controller or when there is no network configuration information, access to slush is obtained through a TTY login.

An instance of slush uses very little CPU bandwidth until client requests a Telnet or FTP connection. After a successful logon, the remote user can execute UNIX-style shell commands to manipulate the file system, set or get configuration information, and start or stop other Java applications. Slush is also extensible. User applications may add and remove commands to slush by invoking methods on an instance of the shell's base Java class.

TINI OS is very small and provides basic services such as task scheduling, a file system, and memory and I/O managers. Unlike most small, embedded controller operating systems, TINI OS is designed to switch heavyweight tasks. Specifically, it is optimised to switch between multiple executing instances of a Java byte code interpreter. This provides the foundation required for running multiple Java applications. The task scheduler is a simple round robin scheduler which provides constant 4 ms time slices. With the exception of the garbage collector, all tasks driven by the OS are Java applications. Multiple native/kernel processes are managed through cooperative multitasking and are therefore subject to tight execution time requirements.

3. EMBEDDED WEB SERVER IN MONITORING (DATA ACQUISITION)

Embedded web server functionality is verified implementing experimental data acquisition system which communicates weather station data to Internet workstation. System hardware consists of a weather station and a TINI board, both from Dallas Semiconductors (Figure 5.).

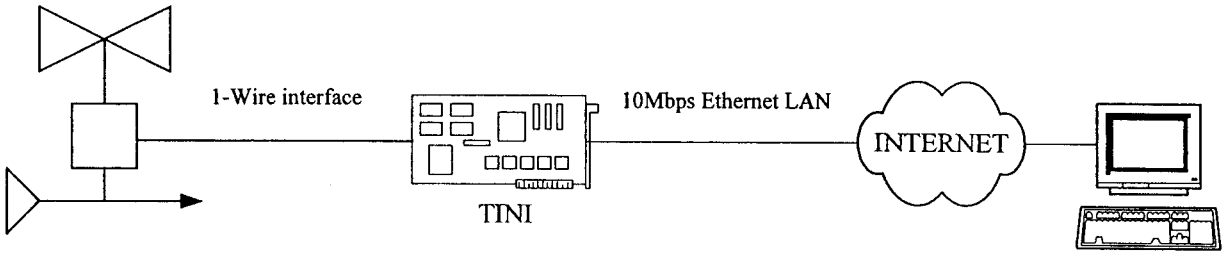

Figure 5. Experimental data acquisition system structure

The weather station is attached to the TINI board using 1-Wire interface, proprietary designed by Dallas Semiconductors. System connection to Internet is achieved over the 10 Mbps Ethernet LAN.

Communication software routines and HTTP server are built in a single Java application.

3.1. Weather station

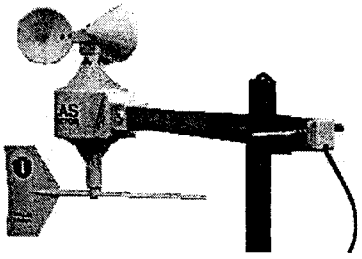

Figure 6. 1-Wire weather station

Dallas Semiconductor 1-Wire weather station (Figure 6.) for measuring temperature, wind speed and direction is a network on its own composed of many individual chips all sharing the same wire [10]. It is called a 1-Wire Net [11]. The chips in 1-Wire weather station are so low power that they can be powered by the 1-Wire communication signal. Because 1-Wire Net uses just one signal for both its power and network connection, the economy of connections lowers the cost of distributed sensors. Each 1-Wire chip has a unique ROM registration number that contains a family code, serial number and a CRC for checking the correct delivery of the registration number. They digitise the measurement at the source to make an inexpensive communication link to a host computer without loss of accuracy.

3.2. Data acquisition software

Since the TINI board runs Java programs, software can be created using freely available Java Development Kit from Sun Microsystems.

TINI's flash memory contains standard java.net, java.lang, java.io and java.util packages, and com.dalsemi packages for accessing the TINI command shell slush, the 1-Wire bus and several system parameters. RS232 serial port support is contained in the Java communication API. These packages should be built into the application. Through include statements, TINIconverter takes necessary class files and converts them into a single tini file that can be executed on TINI. The new program is uploaded to TINI using FTP. The program can be started from telnet session or automatically by putting commands in the .startup file.

This particular experimental data acquisition system is based on a software module with dual-function: data acquisition and http server. Data acquisition software identifies 1-Wire sensors in the weather station and reads their status. Http server interprets template web page stored in TINI board RAM. The HTML page contains basic layout and Java applet that collects data from data acquisition software and creates graphical presentation of measurement results.

The final layout of the web page (http://tini.fesb.hr) with weather report is shown in Figure 7.

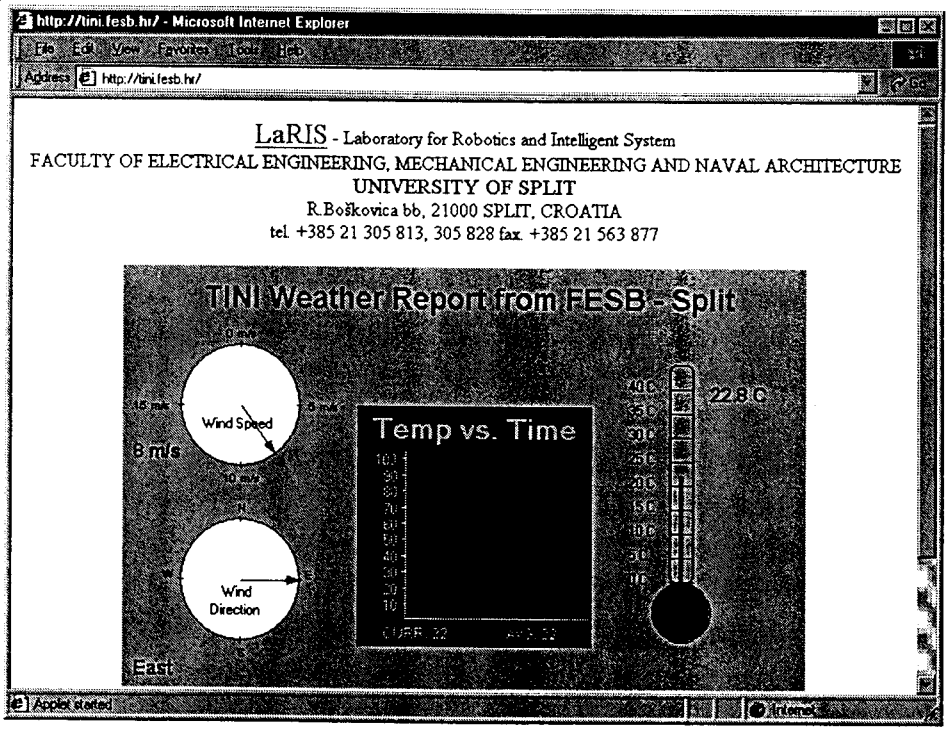

Figure 7. TINI weather report

4. CONCLUSION

In this paper we presented embedded systems that gather and route data from local monitoring devices for centralised readout and control through Internet.

The developed experimental system, based on TINI board, shows that a system for meteorological data acquisition could be successfully connected to the Internet through embedded web server. Dimensions and the price of the embedded web server used in this system are miniature compared to the desktop PC with full functionality preserved at the same time. Our further work with described system is oriented toward measurement data based control of the green house.

On the other side, there are so many issues for embedded systems application and development. Development of the network systems and protocols have substantial importance due to the Internet accessible possibilities. Defining new unified protocols for network interoperability is the field in which embedded web servers can be developed the most. In the field of operating systems for embedded devices, contribution is expected from Linux community, based on availability of powerful 32-bit embedded processors. Introducing Linux based embedded systems will add to development of unified interaction protocols.

REFERENCE

[1] J. Turley: "Microprocessors for Consumer Electronics, PDA's and Communications", Embedded Systems Conference, September 26.-30. 1999.
[2] M. O'Brien: "Embedded Systems Programming", www.embedded.com/internet/9911/9911ia2.htm
[3] L. Mittag: "Why now is the Time for Internet Appliances", www.embedded.com/db_area/embeddedinternet/artic01.html
[4] Jini™, www.sun.com/jini
[5] Home Audio/Video Interoperability (HAVi), www.havi.org
[6] Extensible Markup Language (XML), www.w3.org/TR/REC-xml
[7] S. Fink: "Hardware/Software Tradeoffs in Microcontroller Based Systems", Embedded Systems Conference, Spring 1999.
[8] R.A.Quinnell: "Web servers in embedded systems enhance user interaction", EDN Magazine, April 10. 1997.
[9] Tiny InetrNet Interface, www.ibutton.com/TINI
[10] D. Awtrey: "The 1-Wire Weather Station", Sensors, June 1998, pp. 34-40
[11] D. Awtrey: "Transmitting Data and Power Over a One-Wire Bus", Sensors, February 1997, pp. 48-51
[12] W. Webb: "Linux strafes the embedded landscape", EDN Magazine, June 22. 2000. pp. 58-68