# Networked Embedded Greenhouse Monitoring and Control

Darko Stipanicev, *Member, IEEE* and Jadranka Marasovic, *Member, IEEE*

***Abstract*— Networked embedded systems have become quite important nowadays, especially for monitoring and control of distance and dislocated objects. Small greenhouses are typical examples. First, they are usually located far away from the owners house, and second, the plant grow is an example of the process which need constant 24 hours monitoring.**

***In this paper networked embedded greenhouse monitoring and control based on simple embedded web servers and 1-wire protocol for connecting sensors and actuators is described. Hardware and software architecture of embedded web servers is described and the experimental results of monitoring and control of laboratory greenhouse model is presented.***

***Index Terms — Embedded web server, Greenhouse Monitoring and Control, Internet, Networked Embedded Systems, TINI, 1-Wire protocol***

## I. Introduction

Today, advances in sensors, actuators and microprocessor technology, both on hardware and software level, have enabled distributed implementation of sensor and control actions over sensor/actuators networks. If we connect such local sensor/actuators, private networks to global network (Internet) additional features could be exhibit. The monitored and controlled system could become accessible from almost anywhere. The process parameters data display, remote control, system testing and system reconfiguration could be done using standard browsers on workstation computers. That allows us to use large screens, menus, buttons, and on-line helps, instead of simple alpha – numeric displays usually connected to such embedded devices.

The bridge between distributed sensors and actuators on one side and Internet on the other side could be **Embedded Web Servers**. They are low cost devices, which allow us to use standards network protocols like TCP/IP, HTTP, PPP telnet, ftp.

Authors are with Laboratory for Modeling and Intelligent Systems (http://laris.fesb.hr), Faculty of Electrical Engineering, Machine Engineering and Naval Architecture, UNIVERSITY OF SPLIT, R.Boskovica bb, 21000 Split, C r o a t i a.
D.Stipanicev (e-mail: dstip@fesb.hr ), J.Marasovic (e-mail: jadranka.marasovic@fesb.hr ).

In this paper greenhouse monitoring and control system based on simple 1-wire sensor/actuators local private network and embedded Web servers for connecting this network to Internet is described.
System design, protocols, software and hardware requirements for embedded Web servers are presented, detail description of greenhouse monitoring and control system is given together with experimental results of laboratory greenhouse model monitoring and temperature control.

## II. Embedded Web Servers System Design

Embedded systems require web server software that enhances their networking functionality without taking up vital system resources. Web enabling of devices is possible by adding web server software to existing embedded system. 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. Using statically allocated or pre-allocated memory blocks usually solves this problem.
- 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 [1,5].

0-7803-7729-X/03/$17.00 ©2003 IEEE.

### A. 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.

### B. Software

A minimal set of software for embedded Internet system includes operating system and application software, an 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. [2, 3].

### C. 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.

### D. System Design

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.

## III. NETWORK EMBEDDED GREENHOUSE MONITORING AND CONTROL

Network embedded greenhouse monitoring and control system is a small-size network of tightly coupled physical/information system components (sensors, actuators, processing and communication resources) with limited reliability and changing physical topology. The system core is **Embedded Web Control Unit** connected on one side to sensor/actuator local network and on the other side to Internet.

Between existing interface devices, minimal configuration suitable as a working interface needed for low-cost networking is TINI™ [4,5], so we have choose him as a central part of our Embedded Web Control Unit.

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.

Optimized 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. One of the most interesting TINI features is its I/O ports. TINI has Ethernet 10Base-T interface, 1-Wire® net interface and dual serial (RS-232) ports. Ethernet interface is used for permanent Internet connection, serial port for dial-up modem Internet connection and 1-Wire net interface for sensors and actuators connection.

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.

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. Multiple native/kernel processes are managed through cooperative multitasking and are therefore subject to tight execution time requirements.

All sensors and actuators are connected using Dallas Semiconductor 1-Wire Net [6]. 1-Wire Net uses one signal for both, power and network connection. Each sensor or actuator has its own 1-Wire chip with a unique ROM registration number that contains a family code, serial number and a CRC for checking the correct delivery of the registration number.

The sensors used in our experimental greenhouse model were temperature and humidity sensors, but parallel we have monitored the weather conditions outside the greenhouse (temperature, wind speed and wind direction) having in mind future predictive control algorithms. Inside the greenhouse the temperature was measured in the air, on the ground surface and 5 cm inside the ground. Other type of sensors, like barometric pressure or lighting sensors could be connected too. The temperature and humidity inside the greenhouse could be controlled using heaters, overheat protection devices (ventilators) and water wetting system.

Laboratory, experimental greenhouse is schematically shown on Fig.1. Fig.2. shows the laboratory greenhouse model photo.

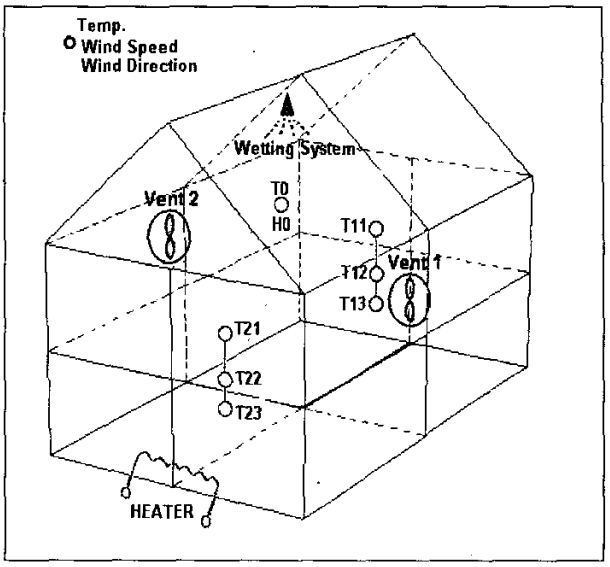


Fig. 1. T0, T11, T12, T13, T21, T22, T23 are temperature sensors and H0 is the humidity sensor. Actuators are heater, wetting system and vent1 and vent2 for overheat protection.

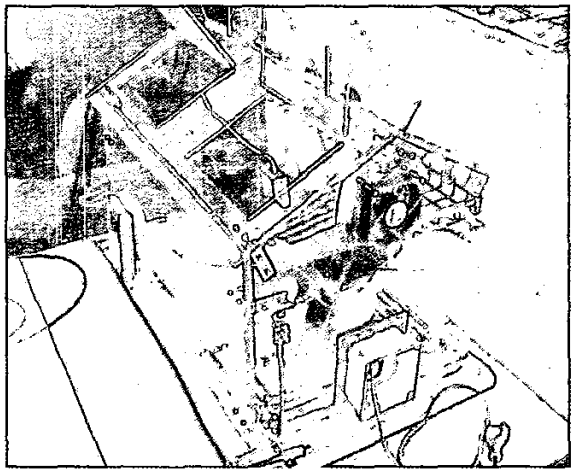

Fig. 2. Laboratory greenhouse model used in our experiments

Embedded Web Control Unit is shown in Fig.3.

1-wire sensors could be parallel connected, so almost unlimited number of sensors could be used. Of course, practically, there are some limitations because the processor has limited capacity. We have successfully worked with 30 sensors, and that is more than enough for any greenhouse application.

Embedded Web Controller software is conceived as a collection of software agents - servlets, running in parallel, and each of them performing one specific task. The servlet structure is schematically shown in Fig.4. The central one is **MonitoringServlet** responsible for reading 1-Wire sensors and displaying data in the form of dynamic HTML page.

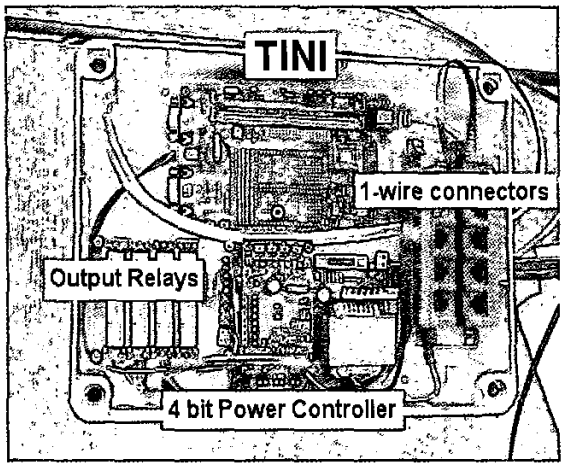


Fig. 3. Embedded Web Control Unit based on TINI embedded Web server

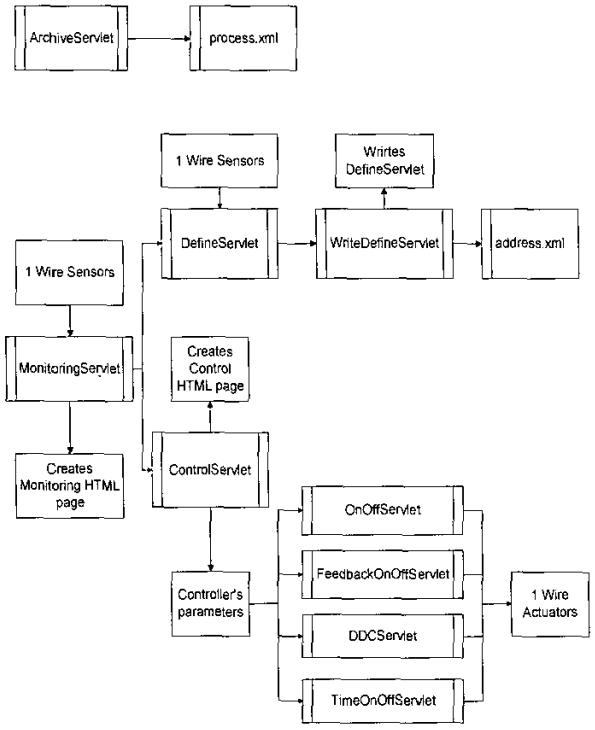


Fig. 4. Embedded Web Control Unit Servlet Structure

Fig.5. shows Monitoring HTML page dynamically created by MonitoringServlet.

The important feature of the designed system is the possibility of 1-Wire topology changing without any influence to system functionality. The special software agent **DefineServlet** check the 1-Wire network, detect the actual sensor/actuator network configuration and after that **WriteDefineServlet** write identification numbers in xml file address.xml Detail sensor description (min, max values, units) could be inserted in the same file using special Web form. In address.xml file root element is 'address' and the child elements are 'variable', each of them having subchild elements 'name', 'id', 'min', 'max' and 'unit'.

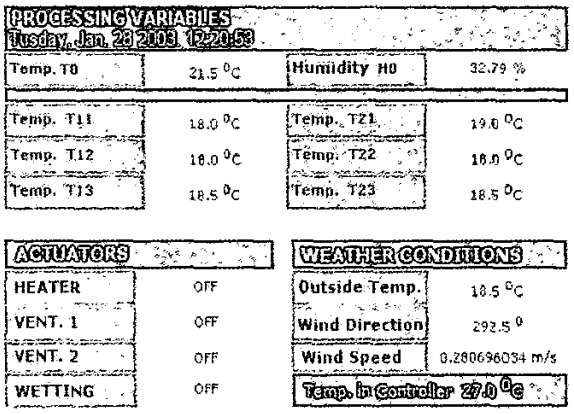


Fig. 5. Monitoring HTML page created by MonitoringServlet

An example of one child element (temperature sensor) of address.xml is:

```
<address>
  <variable>
  <name>Temp. T0</name>
  <id>3A0008000F514010</id>
  <min>-10</min>
  <max>45</max>
  <unit>°C</unit>
  </ variable >
  . . .
</address>
```

Before displaying process parameter values DefineServlet reads the sensor identification No. from address.xml file.

The third servlete is **ArchiveServlet**, which is responsible for process data archive. All process variables values are stored in predefined intervals to process.xml file:

```
<process>
   <sample>
   <id>3A0008000F514010</id>
   <value>21.0</value>
   <time>12:39:13</time >
   <no>1</no>
   </sample>
   < sample >
   <id>720008000F469810</id>
   < value >21.5</value >
   < time >12:39:17</time >
   < no >1</no >
   </sample >
   ...
</process>
```

As the Embedded Web Server memory is not big enough to store archive files, they are stored to another standard Web server. JavaScript/JavaApplet program was written to display the archive data. Fig.5. shows one example of displaying archive temperature data.

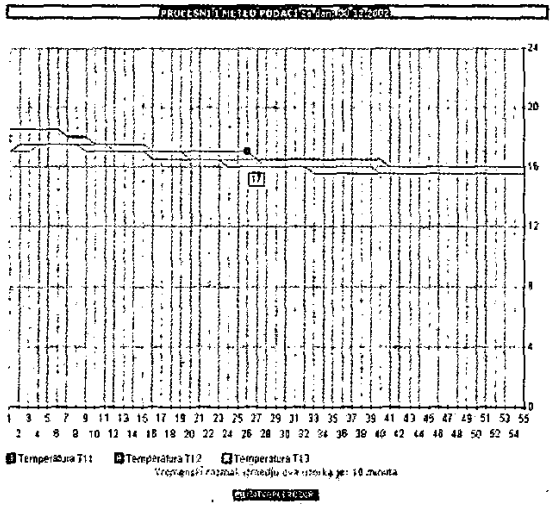

Fig. 5. Displaying archive temperature data using JavaScript/JavaApplet application

For control purpose, four control servlets were developed, each of them responsible for its type of control. Fig.5 shows the control dynamic HTML page created by central **ControlServlet**. The ControlServlet is activated from the main monitoring page and UserID and Password protect it from unauthorized access.

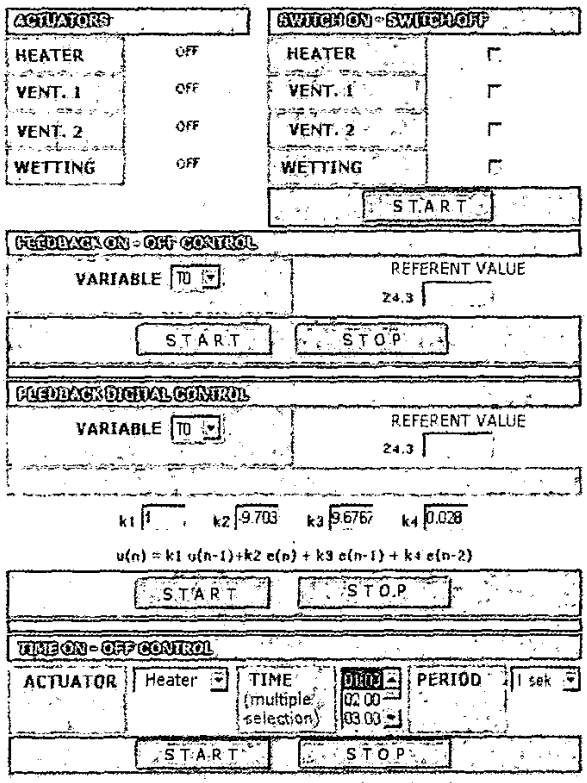


Fig. 5. Control HTML page created by ControlServlet

Four types of control procedures were applied:

a) Simple switch on – switch off control of all actuators using output relays. Any actuator could be remotely switch on and switch off by clicking in the appropriate checkbox in the Control Web page (**OnOffServlet**).

b) Feedback on – off control. The user chose the controlled variable (for example temperature T0 from Fig.2) and its referent value and the servlet maintained the referent temperature using simple on-off control procedure (**FeedbeckOnOfServlet**).

c) Feedback discrete PID and DDC control based on difference equation:

$u_n=k_0*u_{n-1}+k_1*e_n+k_2*e_{n-1}+k_3*e_{n-2}$

where $u_n$ and $u_{n-1}$ are control signals in nT and (n-1)*T time instant, $e_n$, $e_{n-1}$ and $e_{n-2}$ are error signals in nT, (n-1)*T and (n-2)*T time instant and $k_0$, $k_1$, $k_2$ and $k_3$ are regulator constants defined by user through standard Web form (**DDCServlet**).

d) The time on – off control. This type of control is especially important for wetting system, but it could be used for any actuator. The user choose the time moments when one of actuators would be switched on and time duration of its active state. For example the wetting system could be switched-on at 6:00 , 12:00 and 18:00 for 20 seconds (**TimeOnOffServlet**).

Fig.6. shows two control examples. The first one, shown in Fig.6a) is temperature control using heater and the second one, shown in Fig.6b) is temperature control using cooling devices (ventilators).

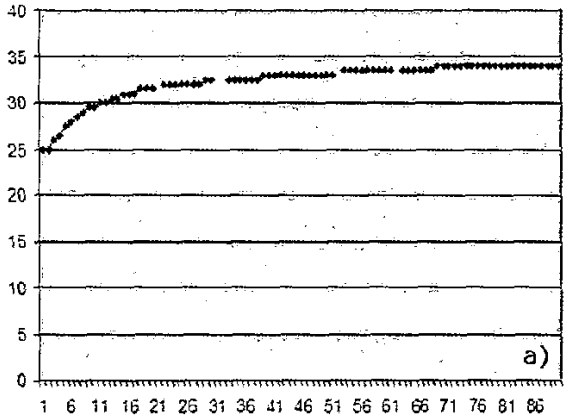


Fig. 6a) Temperature control using heater

Each Embedded Web Control Unit has its own IP address and it could be connected to Internet permanently using Ethernet or temporary using modem dial-up connection, because TINI has both Ethernet and serial port. We have realized two additional Java programs. The first one works as a PPP server and it is used when the Embedded Web Control Unit is working as a pull device. The Unit is waiting for modem call, and after the PPP connection is established, the MonitoringServlet could be triggered and dynamic Monitoring or Control HTML page generated. The second one works as a PPP client and it is

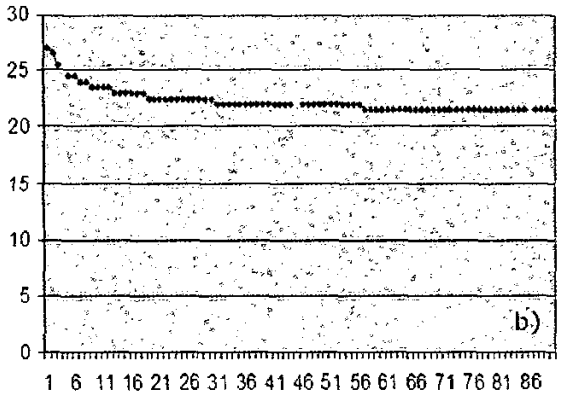

Fig. 6b). Temperature control using ventilators

used when the Embedded Web Control Unit is working as a push device. The Unit itself is responsible for establishing PPP connection with any ISP provider. After that the thread is automatically started, responsible for transferring process.xml file to predefined server for archiving process parameter data. The users could collect sensor data in any time from that archiving server. Dial-up modem Internet connection is quite important for practical use in real greenhouses, because mobile telephone service is widely available.

## IV. CONCLUSION

In this paper we have presented network embedded greenhouse monitoring and control based on embedded Web server unit which gather and route data from local sensor/actuator network to global network - Internet.

The developed experimental system, based on TINI embedded Web server, collect data from distributed sensors and activate connected actuators using simple 1-wire local network. On the other side Web server is connected to Internet through Ethernet or dial-up network.

The developed system shows all advantages of Network Embedded System Technology (NEST), like the possibility of changing physical topology and low dimensions and cost in comparison with PC based system, preserving the full functionality at the same time.

## V. ACKNOWLEDGEMENTS

This work was supported in by the Ministry of Science and Technology of Republic Croatia under Grant TP-01/0023-02 "Embedded Web servers in Monitoring and Control of dislocated systems" and in part under Grant 0023008 "Intelligent Agents in Modeling and Control of Complex Systems". The project Web page (in Croatian) is http://laris.fesb.hr/tehnoloski.htm .

Especially thanks to Maja Stula, Maja Cic, Mirjana Bonkovic, Ljiljana Bodrozic, Zvonimir Torba and Kaja Radic for their contribution to project realization .

## VI. REFERENCES

[1] J. Turley: "Microprocessors for Consumer Electronics, PDA's and Communications", Embedded Systems Conference, September 26.-30. 1999.

[2] M. O'Brien: "Embedded Web Servers", Embedded Systems Programming, November 1999. www.embedded.com/internet/9911/9911ia2.htm

[3] R.A.Quinnell: "Web servers in embedded systems enhance user interaction", EDN Magazine, April 10. 1997.

[4] Tiny InetrNet Interface, www.ibutton.com/TINI

[5] M.Bozanic, Z.Cic, D.Stipanicev, Embeded Web servers, SoftCOM'2000 (Internacional Conference on Software in Telecommunications and Computer Networks), Split, Rijeka (Croatia), Trieste, Venice (Italy), October 10-14 2000, pp. 63-73

[6] D. Awtrey: "Transmitting Data and Power Over a One-Wire Bus", Sensors, February 1997, pp. 48-51