A Simple Architecture for Embedded Web Servers

Miguel Domingues

Universidade do Minho
4710 - 057 Braga, Portugal
mig@idite-minho.pt

Abstract. Older technologies can still play an important role in embedded systems. Complex applications such as a Web server can be embedded implemented, with some restrictions and assumptions, and still be efficient for current industry demands. This communication makes an incursion into the hardware architecture behind an embedded Web server based on simple 8051-type processors. It presents and discusses architectural features, limitations, performance and trends.

1 Introduction

Computer communication systems and especially the Internet are playing a rapidly increasingly important role in our everyday environment. Today this is not only a domain of personal computers or workstations. We are beginning to see the Internet and associated technologies manage our work and home environments through the use of intelligent embedded devices. Using this knowledge, many applications are imaginable. Home automation, utility meters, appliances, security systems, card readers and building controls that can be easily controlled using either special front-end software or a standard Internet browser client from anywhere around the world. Imagine applications that are able to control hardware via a standard Web browser, to transmit and visualize the state of sensors or automatically generate and send emails on the occurrence of special events, for example, for security purposes. There are not just routers and switches and network infrastructure but devices used in everyday life, from security systems to vending machines.

Hewlett-Packard filed the following patent "Embedding Web access mechanism in an appliance for user interface functions including a Web server and Web browser," on October 25, 1996. The patent was issued on September 21, 1999. The abstract is as follows: "Web access functionality is embedded in a device to enable low cost widely accessible and enhanced user interface functions for the device. A Web server in the device provides access to the user interface functions for the device through a device Web page. A network interface in the device enables access to the Web page by a Web browser such that a user of the Web browser accesses the user interface functions for the device through the Web page" [1]. The idea behind this patent is closely related with the new paradigm for embedded systems. It states that networks have to become flexible and easily integrated, with the user getting closer to the device without supplementary efforts, using large networks like the Internet. Such device, which consumes a few bytes of memory and is specifically designed for microcontroller-based embedded systems, allows designers to create modular components that can be connected to the Internet and controlled remotely using a standard Web browser. By adding Web server technology, the manufacturer gains an immediate competitive advantage through standardized access, both in terms of protocol and client application. With the explosion of the Internet and Web services, companies that have provided proprietary solutions for networking are rushing to add Internet technologies and

embedded Web servers to their product lines. It provides a more open and economical alternative of the networking devices, reduces development costs and increases functionality. Communication solutions about industrial networks, such as Powerline, RS485, RS232, CAN or 12C have their special qualities and their functional space in real-time application worlds, where communication reliability and efficiency is an imperative request, but in other way, they represent an additional handicap when it comes the time to maintain and support the applications. Actual application designers need a standard communication model that integrates with the rest of the world, even if it is so small like the local back office.

Accepting this model for small devices, introduces a reverse engineer process that has to happen. It is necessary to reduce the Web server to its essential components, requiring just a few bytes of memory at the device. In larger devices, where more memory is available, the server should still be kept as small as possible, allowing more room for the device's application software. When interfacing with embedded devices, getting the information to and from the device quickly and easily is essential, so reducing the amount of information sent to and from the device makes the Web a more efficient communication medium. The big advantage of a Web server in an embedded environment is that the Web browser manages the whole user interface. The visualizing of information is possible by sending Hyper Text Markup Language/American Standard Code for Information Interchange (HTML/ASCII) strings to the client, therefore minimal resources are required. Thanks to the increased versatility and flexibility of microcontrollers such as Philips 80C51 and XA, Infineon C500 and C166, Texas Instruments MSP430 ultra low-power microcontroller and others, manufacturers can create solutions for their customers inexpensively. Companies are now able to connect large-scale networks to embedded devices using 8 and 16 bit microcontrollers.

It seems like everything is serving Web pages these days, so why not an 8051? This report presents a simple architecture for an embedded Web server and its requirements. Finally, an overview of a 8051-type implementation is described and some benchmarks states its valid functionality.

2 Hardware Architecture Presentation

The application requisites of a Web server are the most important key to determine the characteristics of any implementation. In fact, issues like file Input/Output (I/O), network communication model, memory cache, page processing, instruction complexity and virtual memory directly determine the goals that the architecture has to achieve for a plenty success.

In a generalist implementation of a Web server the available operations are to serve static or dynamic content as response to a Uniform Resource Locator (URL) request. In the static case, it serves the page's ASCII code and binary for any other document. A file I/O instruction gets the requested content and serves it over Hyper Text Transfer Protocol (HTTP). If the content is dynamic a search and replace is applied. This determines that the application needs a simple file system for page storage, a small memory footprint for search and replace and a communication model on which HTTP relies to serve the final page content. The search and replace operation is a simple set of instructions that do not require a great calculus power nor complex scientific functions. Catching pages in memory is an advantage for static and dynamic pages, but there is no need of virtual memory because some bytes of ASCII tags. With this description, the framework is totally characterized, but it must consider that the host hardware is embedded based. So, some considera-

tions impose additional limitations. An embedded Web server architecture limits the scope of the design for the simple reason that file system is not supported [2]. Remember that the device must be of easy integration and has low-cost per production unit.

Observing the facts, let us conclude that a conventional 8051-type architecture could take every of the discussed aspects. The only one that it does not guaranties, has to be with the network communication model, of course, some assumptions and restrictions had to be taken, and they will be described later on the article. To illustrate a possible implementation of a simple embedded Web server architecture, it follows an example that is used to understand the feasibility of device.

The two main components of the demonstration board are the Local Area Network (LAN) and Microprocessor Unit (MPU) controllers. The MPU runs the code responsible for the Web server. The LAN controller offers the physical connection to the Internet with a downsized Transmission Control Protocol/Internet Protocol (TCP/IP) stack, and guaranties the needs for the network communication model. Its functionality is encapsulated by an easy-to-use application programming interface, and by using this Application Programming Interface (API), creating new applications or modifying existing ones becomes an easy task.

This architecture uses the C8051F005 MPU from Cygnal and the CS8900A Ethernet controller from Crystal™ Semiconductor Corporation. [3]

The 8 bits Cygnal 8051 is fast with its 25-MIPS peak performance, and it has a 12-bit A/D converter and 2.4 KB of RAM [4]. The 32-KB flash memory is large enough for a reasonably sized program plus a few Web pages and therefore this makes it a good choice for storing and transferring Web pages. It also has general-purpose input/output ports that can be used for interfacing to the LAN controller. At first, it seems like its lack of a conventional bus with the LAN controller could be a problem, but it turned out to be no problem at all.

A 22.1184-MHz crystal was soldered onto the board and a new function was written to make the Computer Processor Unit (CPU) use it instead of the slowest on-chip oscillator. Most of its Cygnal 8051's instructions execute in one or two cycles, as compared to 12 or 24 cycles for standard 8051 chips. A good performance is expected, and indeed the C8051F005 runs the Sieve benchmark about 19 times faster than a standard 8051. It also ran faster than most 16-bit CPUs that were tested, which is impressive because many of the Sieve operations are 16-bit.

Table 1. CPUs Benchmark

CPU	Crystal (Mhz)	Sieve 10 loops (secs)
Cygnal C8051F005 (8 bits)	22.1184	0.43
Intel 80C51 (8 bits)	11.0952	8.2
Intel 80C196 (16 bits)	18.4320	1.3
Philips XA-S3 (16 bits)	22.1184	1.0

For the Ethernet controller, the Cirrus Logic CS8900A with 4 KB of RAM is enough to hold a satisfactory number of frames. It adds additional buffering capability for incoming frames, which is the key for allowing the CPU time to process a frame while more are received. Browsers running on fast machines can easily fire out two or three Ethernet frames within a millisecond. The CS8900A is a low-cost Ethernet LAN controller optimized for Industrial-Standard-Architecture Personal Computers (PCs). The features that made it very suitable for this project are its highly integrated design, which reduces the amount and cost of external components, and its very easy-to-handle bus interface. Most LAN controllers that are on the market have a Peripheral Component Interconnect bus interface. The

CS8900A bus interface is simple to interface with a microcontroller directly [5]. The CS8900A includes an integrated 10Base-T transceiver. It contains all the analog and digital circuitry needed for implementing the LAN interface by the use of a simple isolation transformer. [6]

Because the CS8900A has more Random-Access Memory (RAM) than the 8051, it checks for new frames by polling and reads the most recent one. This way the CS8900A can queue a number of frames while the 8051 pulls them out. This is desirable to hold the entire segment in RAM to compute its sum. It can also handle IP processing in a small memory space, as long as it does not try to reassemble and process all at a time. CS8900A is configured to capture only the frames directed to its Media Access Control address, plus broadcast frames. If the 8051 had to deal with every Ethernet frame on a busy network, it would be in big trouble.

A standard RJ45 patch cable can be used to connect the module to either a 10 Mbps or 100 Mbps hub. A 100 Mbps hub automatically switches down its transfer speed to 10 Mbps if it detects the CS8900A running at 10 Mbps. [7]

3 Network Layer Analysis

Today, IEEE 802.3, also known by Ethernet, contains the most common medium access control to transfer data in a LAN [2]. It belongs to the Network layer in the Internet reference model. The standard IEEE 802.3 defines possible bit rates, the physical realization of bit coding, and the frame format used. Over it, IP is designed for use in packet-based networks such as the World Wide Web. It provides mechanisms for transmitting datagrams from a source to a destination, and for fragmentation if necessary for transmission through small-packet networks. Finally, the hypertext transfer protocol (HTTP) is an application level protocol. It is a generic, stateless, object oriented protocol that can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods. It uses a client-server relationship and is based on a stream-oriented transport layer, such as TCP. It works with the principle of request and response. A client establishes a connection to a server and requests a content referred by URL that specifies the address, path and name of the resource [2]. After decoding the request, the server starts transferring the resource to the client. Commonly, this is done by navigating with a Web browser.

At this point, a reasonable question is - can a CPU with only 2 KB of RAM really handle large Ethernet messages and the complexities of protocols such as TCP and Address Resolution Protocol (ARP)? Surprisingly, it turns out that even a few hundred bytes would suffice. Small RAM footprints work with TCP because you can tell the other end to limit the message size. TCP can, for example, advertise a maximum segment size of only 100 bytes. If it could guarantee that it only responds to TCP segments, rather than initiating them, then a massive simplification in the TCP/IP stack would result. It can also handle IP processing in a small memory space, as long as it does not try to reassemble fragmented incoming messages, and it will not occur because the other machine's TCP layer will limit the size of the message it sends. [10]

ARP is the last one needed to complete the serving process. A Web server must handle ARP requests because it will receive them when the other end wants to find out its hardware address. The inverse operation also happens in order to find out the hardware address of the device it is sending to. ARP is a simple protocol that can cause big problems for a small server. An ARP request must be sent, and a reply received, while a regular message is waiting to be transmitted [11]. One send buffer no longer suffices. It needs more space

and must buffer the outgoing and incoming ARP messages. The good thing, is that ARP messages are only 64 bytes long, so buffering is still possible.

With all significant compromises described above, these are the TCP/IP protocol stack limitations that a small device can not handle: no reassembling of fragmented incoming IP frames; no buffering of TCP segments which are delivered out-of-order; no support for IP type-of-service and security options; ignoring of any other special TCP options.

Next table gives a list of several computers and operating systems where data exchange over TCP/IP was possible and efficient even with last limitation parameters:

Table 2. Compatible Communication Systems

Computer System / CPU	Operating System / TCP/IP Stack
PC / Athlon™/ 1 GHz	Windows 2000
PC / Athlon / 1 GHz	Linux, kernel v 2.2.16
PC / Pentium™ / 233 MHz	Windows 98
PC / 486DX2 / 66 MHz	Windows 95
Apple™ Macintosh™ / 68030 / 50 MHz	System 7.5, Open Transport™ 1.1.2
AT Amiga™ / 68030 / 50 MHz	Kickstart 3.0, Miami 2.1
Cassiopeia™ / MIPS / 150 MHz	Windows CE 3.0

Compatibility is achieved by implementing only the important parts of the protocol specifications, but also is due to the tolerance of the other TCPs.

The maximum transfer speed of the module cannot be defined exactly, as it largely depends on the other TCP. Normally, a TCP that fully implements the protocol specification is able to receive and buffer more than one segment at a time. Because the C8051 has a relatively small amount of memory compared with, for example, personal computers, it can maintain only one receive and one transmit buffer. It needs to wait for an acknowledgement (ACK) from the other TCP before the overwriting of buffer contents is allowed and new data can be exchanged. Because of the round-trip time of packets sent over the Internet and the delaying of ACK segments by some TCPs, transfer speed varies significantly. During evaluation of the TCP/IP stack, transfer speeds between 2 and some dozens of Kbps were measured. But most of the applications for an embedded Web server do not require transfer speeds of several Mbps. If data is transferred in eXtensible Markup Language format, smaller messages with more information are produced and that is more than sufficient to any embedded system.

In spite of the fact that during software development many compromises were made, the compatibility of the stack in communicating with other TCPs is very good. No other TCP had problems establishing a connection during the software evaluation. [3]

4 Web Server Analysis

As an example of how to use the previously described module, a demonstration HTTP server was implemented. The module must have been powered on, properly connected to LAN and the TCP/IP settings of the local host correctly configured. Then, the embedded Web server is ready.

The server provides an HTML Web page that is stored in MCU flash memory. The module waits for an incoming connection, transfers the Web page, closes the connection and waits for another client to connect. The content of this Web page is adapted dynamically with analog values. Before sending a segment of TCP data, it searches the transmit buffer for special strings. If such a string is found, it is replaced by an A/D converter value.

The page has three HTML labels that display Analog-to-Digital (A/D) values such as CPU/air temperature and operating voltage and a radio button pair that toggles the main board Light Emitting Diode (LED) state. One purpose of a small Web server is to make a product easy of use. This page is bidirectional in that it both displays device information data and controls the board LED on or off. The new state of the LED is sent to the Web server in a post message. There is an image, which the browser loads after the HTML portion.

Most of the tested browsers establish a single connection to load both parts of the page, but others, open two separate connections to the server. Each connection comes from a different port on the browser's machine. To handle this situation, all connection specific information, such as client IP address, port number, sequence number, ACK number, and TCP state is stored into a structure that is indexed by the connection number. Each element of this structure can be thought of as a connection. When a TCP segment arrives, if it matches with an existing connection, then it uses its state information. This method allows the Web server to handle simultaneous connections from the same PC or from multiple PCs. [3]

Next photo shows the Web page, as presented on a common browser:

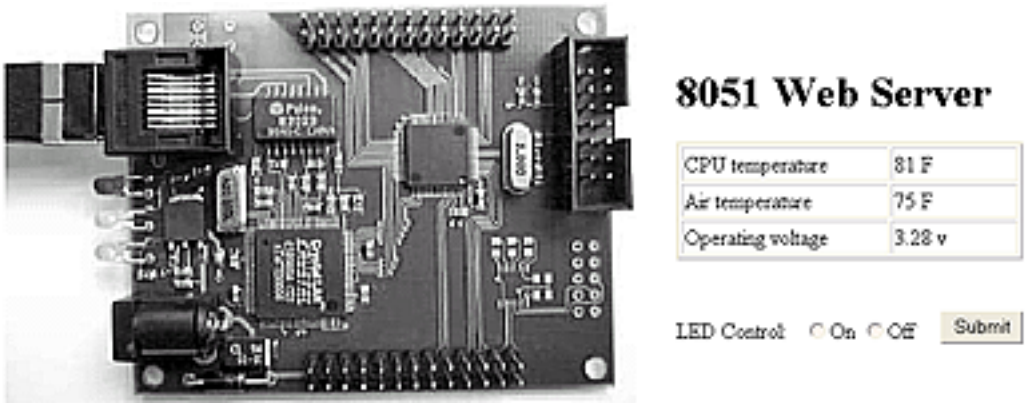

Fig. 1. Browser Web Page

The 0.8 KB HTML portion of the page and the 6.2 KB JPEG graphic combine for a total of 7 KB. This page is used as the basis for later comparisons in the article. Here is an example of the page's ASCII code:

```
<HTML>
<HEAD><TITLE>8051 Web Server</TITLE></HEAD>
<BODY>
<FORM ACTION=\"/index.html\" METHOD=\"POST\">
<TABLE BORDER=\"0\">
<TR>
<TD><H1>8051 Web Server</H1>
<TABLE BORDER=\"2\">
<TR>
<TD WIDTH=\"150\" HEIGHT=\"25\">CPU temperature</TD>
<TD WIDTH=\"90\" HEIGHT=\"25\">TAG:TMP1</TD>
</TR>
<TR>
<TD WIDTH=\"150\" HEIGHT=\"25\">Air temperature</TD>
<TD WIDTH=\"90\" HEIGHT=\"25\">TAG:TMP2</TD>
</TR>
<TR>
<TD WIDTH=\"150\" HEIGHT=\"25\">Operating voltage</TD>
<TD WIDTH=\"90\" HEIGHT=\"25\">TAG:VOL1</TD>
</TR>
</TABLE>
<BR><BR>LED Control: 
<INPUT TYPE=\"RADIO\" NAME=\"switch\" VALUE=\"1\" TAG:CHK1>On
```

```
<INPUT TYPE=\"RADIO\" NAME=\"switch\" VALUE=\"0\" TAG:CHK2>Off
  <INPUT TYPE=\"SUBMIT\" VALUE=\"Submit\">
</TD>
<TD><BR>  <IMG SRC=\"photo1.jpg\"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

Web pages in HTML format are for human consumption. Therefore it is reasonable to say that 100 ms response time is a good reference value. The measured time required for this 8051-type Web server to serve the 7 KB Web page to a 500 MHz Pentium machine running Microsoft Internet Explorer 5.5 was 60 ms, 40 ms less under the reference value. In contrast, it takes to a 100 MHz Pentium server running Apache about 32 ms to serve the same page. This demonstrates that the response of the 8051 is acceptable. The page related tasks and its response time values can be observed in the next table:

Table 3. Page Tasks and Response Times

Task	Times Run	Time (ms)
Search and replace tags	6	23.8
Copy buffers	8	9.1
Write to CS8900A	11	4.6
Parse incoming HTTP headers	2	4.5
Compute checksums	38	4.4
Read from CS8900A	8	0.6
TCP state machine	8	0.4
Total		**47.4**

How to explain the difference between the announced 60 ms to the presented 47.4 ms? The answer is so surprisingly that when added all the intervals between the 8051 sending an Ethernet frame and the browser's 500 MHz Pentium responding, the result sum is 8.5 ms. Here is the big difference. The 8051 is waiting for a Pentium! [3]

About memory usage, the lower 256 bytes of the 2.4 KB C8051 RAM are used for frequently accessed variables. The 2048 byte area of additional on-chip memory is addressed as External Data memory. Incoming and outgoing message buffers are dynamically allocated from this space. The server execution code uses 22 KB and more 7 KB are for the Web page content, what makes a total of 29 KB of the 32-KB on-chip flash memory. Access to on-chip flash memory is fast and the page could be transferred into the 8052 in about 30 ms, this is about 50% of the total time needed to serve the page. The details are shown in the next table:

Table 4. Memory Usage

Description	Code Space (KB)
TCP/IP	9.5
Web Page (plus image)	7.0
HTTP Server	3.8
ARP	2.5
C Library	2.9
UDP	1.4
CS8900A I/O	1.0
RS-232	0.5
Analog	0.3
Priority Task Switcher	0.3
Total	**29.2**

5 Summary

This paper has reviewed the key areas associated with a simple embedded Web server design and implementation, such as typical microcontroller hardware and its constraints, the network protocols needed for a Web server and the implementation techniques to minimise resource usage. The presented example has sufficient resources to support the creation of useful Web pages, including dynamic data. The real challenge in designing a workable and extensible embedded Web-based server is the data layer.

The 8051 microcontroller family remains one of the most popular processors in the world. Its ease of use and its relatively high performance make it ideal for many applications, including portable and handheld products. The introduction of a new line of high-performance derivatives has many positive implications for improving the power efficiency of 8051-based designs. Its low cost is an advantage when designing embedded systems for high volume applications. Perhaps there is no need to invest much more to find a new solution, it has ever been there and only requires a redesign for the new appliances requirements. The main reason of this paper was to show the performance, reliability, low cost, adaptability and power of the old 8051 architecture in the new embedded systems applications. One point has been clarified about the place it could take in these new solutions with old technology: 8051 is respectable.

As more and more devices appear, the issue is no longer if it will have embedded Web technology but, how and to what extent it will be used. Device designers need to pay close attention to the features and functions that are allowed through these remote interfaces, and what are the industrial requirements that make the device production feasible. It is clear, that the technology was already invented. The work that follows, must prepare and advice the companies in order to take the right decisions to achieve the right demands.

References

[1] Casterline, R.: Serving Up Web Pages from an Embedded Device. Lighthouse Solutions, LLC, Embedded Systems Conference, http: www.lhsolutions.com, (2000)

[2] Bentham, J.: Creating a Miniature Web Server from Scratch. Iosoft, Ltd., (2000)

[3] Brady, J.: Build Your Own 8051 Web Server. Circuit Cellar Magazine, Issue 146, http: www.circuitcellar.com, (2002)

[4] C8051F005 Mixed Signal MCU. Cygnal Integrated Products, Inc., http: www.cygnal.com, (2000)

[5] Using the Crystal CS8900A in 8-Bit Mode. Cirrus Logic, Inc., (2000)

[6] CS8900A Product Data Sheet. Cirrus Logic, Inc., (1999)

[7] CS8900A Ethernet Controller Technical Reference Manual. Cirrus Logic, Inc., (2001)

[8] Dannenberg, A.: MSP430 Internet Connectivity. Texas Instruments (2001)

[9] Hall, E. A.: Internet Core Protocols. O'Reilly & Associates, Inc., Sebastopol, CA, (2000)

[10] Bentham, J.: TCP/IP Lean - Web Servers for Embedded Systems. CMP Books, R&D Developer Series, (2000)

[11] Washburn, E.: TCP/IP Running a Successful Network. Addison Wesley (1996)