

The Challenge of Embedded Internet Design

Embedded systems developers are increasingly facing a new challenge: What is the best way to implement an Internet or Intranet connection for their designs, and how to make the most of this connection whilst minimising memory and performance impact on the target system. Possibly the most powerful tool available to address this new challenge, Web server technology has the potential to become the most useful and the most pervasive of all the Internet applications for the embedded systems designer.

The embedded Web or Hypertext Transfer Protocol (HTTP) server allows embedded systems developers to take advantage of a network infrastructure that is already in place to enhance their designs and provide new features for their customers. It has gained acceptance as a protocol for gleaning information from a wide variety of sources and its open extensible nature makes it suitable for the full range of networked embedded systems. Many companies are already building Web-browser support into smartphones and network computers, but the other end of the connection also has great potential for many more classes of embedded system. Once fitted with an embedded Web server, a device can report its status to any client on the network and can even be reprogrammed remotely to perform new tasks.

TYPICAL APPLICATIONS

The first widespread use in embedded applications have been in office automation products where the Web server provides a set of pages that let a systems administrator configure a printer or multifunction peripheral attached to the network. The pages can also provide status information, such as the amount of paper remaining in each tray or whether toner needs replacing. If access to the printer is allowed beyond the Intranet, it could also relay engineering information to the supplier to allow preventive maintenance - or even re-order a toner cartridge just before it runs out.

The ability of a Web server to pass real-time status information to a client across a network suits it to process- and machine-control applications. Instead of writing dedicated SCADA applications for every possible kind of client machine, networked devices could display information using graphs and charts generated internally on any platform that can support a graphical Web browser.

Web servers also provide a more user-friendly way of updating systems in the field. Instead of working with a command-line interface, the embedded server could provide a set of form-based pages that let a service engineer tell the system where to pick up new program and data files. Once returned, the embedded system would use FTP to pick up the files and then update itself.

USING HTTP SERVERS IN EMBEDDED SYSTEMS

Unlike other remote user-interface technologies, HTTP does not demand that the remote clients use a particular piece of software and a proprietary protocol. Any compliant Web browser, whether it is running on a PC or another embedded system, can process and display the information delivered by HTTP. It also provides a standard mechanism by which user input can be returned to the server.

Many embedded Internet devices (EIDs) will use HTTP instead of providing a user interface through a local front panel. For example, an intelligent instrument might use an HTTP server to enable any Web browser to act as a virtual front panel, saving on the component cost of an LCD and its driver circuitry while making it easier to control or access data. This application could apply to a device as simple as a thermostat, letting itself be read and adjusted from any client that is attached to the same network.

The basic responsibility of an HTTP server is to respond to requests from one or more Web browsers by sending the requested file. However, there are important differences between a common desktop HTTP server, which acts mainly as a high-performance file server, and one designed for an embedded system.

The servers that support the Alta Vista search site are expected to cope with some 200 requests per second and must store connection information about each for up to four minutes. Although some EIDs, such as dedicated Web appliances, may be expected to support that level of workload, most designs will need only to support one or two simultaneous connections.

The software that implements the Web server must be scalable to support the right level of service without incurring the memory and performance overhead of a Unix or NT server. At the same time, the client independence offered by Web technology means that the embedded Web server must support the same standards as those employed by large-scale servers. That means adherence to standards such as HTTP.

A potential problem for any EID expected to run on an HTTP server is that the original standard was created for server-class computers that could be expected to have a file system: the pages are located as documents in the server's file system. A low-cost EID may

not have a file system. Therefore, it is important to provide the ability to create documents from application code at runtime.

In the context of an EID, Web technology comes into its own when it is used to dynamically create and react to HTML. For example, the page may be updated at regular intervals to display changes in temperature in a Web-enabled thermostat. An HTML page may be used to pass information from a client application back to the embedded system so that the user can change temperature setpoints.

A desktop-class HTTP server normally uses scripts written to the Common Gateway Interface (CGI) for this purpose. However, CGI is designed for UNIX scripting languages such as Perl, making it big, slow, inflexible and only loosely coupled to the server. Unless CGI compatibility is a requirement of the EID's design, it is generally inappropriate for use in embedded systems.

DESIGNING FOR THE EMBEDDED INTERNET

Fortunately, there are ways to make a server build and react to dynamic HTML pages without compromising compatibility with off-the-shelf Web browsers. The technique employed by Wind River Systems for its Wind Web Server uses an approach similar to the Server Side Includes (SSIs) that are often used with CGI programs. A typical HTML page consists of a combination of text and tags that are used to indicate changes in text styling, the position and target for hypertext links and where included graphics should appear on the page.

Pages containing SSIs are parsed by the HTTP server before transmission and used to insert variables or information from other files that may be generated on the fly by other programs running on the server. The WINDWEB SSI tags recognised by the Wind Web Server specify target functions that are called at runtime to generate the dynamic information that is used to replace the tag. Lines that do not contain WINDWEB tags are sent directly to the browser.

WINDWEB tags use the following syntax:

```
<WINDWEB FUNC=showValue
```

```
SYMBOL=mySymbolName>
```

```
</WINDWEB>
```

Using this syntax, the source HTML page can be written using a standard HTML editor. Almost all commercial programs make it possible to enter tags that are not directly supported by the editor. This lets non-programmers create the visual elements for the Web pages employed by an embedded system and makes it easy to change designs without affecting the behaviour of the program that supplies the runtime data.

When called, the showValue function looks up the new value of mySymbolName and passes it back to the HTTP server using the httpPrintf() function. This function allows any data to be entered in the HTTP output stream. It need not be an ASCII-encoded value; it could easily be an image or other multimedia object. This can be particularly important for systems that do not support a file system but need a way to display graphics on Web pages. Instead of storing a GIF or JPEG image as a file, it may be constructed on the fly by a page containing just the WINDWEB tag (e.g. by a digital camera). The tag is used to call a function that is designed to create and send the graphic to the HTTP output stream.

To make it easier to build Web pages for target systems without a file system, all HTML documents can be compiled into code using the PagePack utility. This software automatically creates C code and header files from input files created using an HTML editor. These files contain the byte code of the input files and a series of references which the Wind® Web Server uses to read the data out of memory at runtime.

HTML forms provide a convenient way to configure and manage an embedded system because information can be passed from the browser to the server. Form processing comprises two steps. The first is to send the page that describes the form to the Web browser; the second involves processing the form entries that are returned to the server.

Traditionally, default values in forms have been encoded into the HTML directly. Thanks to the way that it processes tags, the Wind Web Server can obtain the form defaults that should be shown on the generated

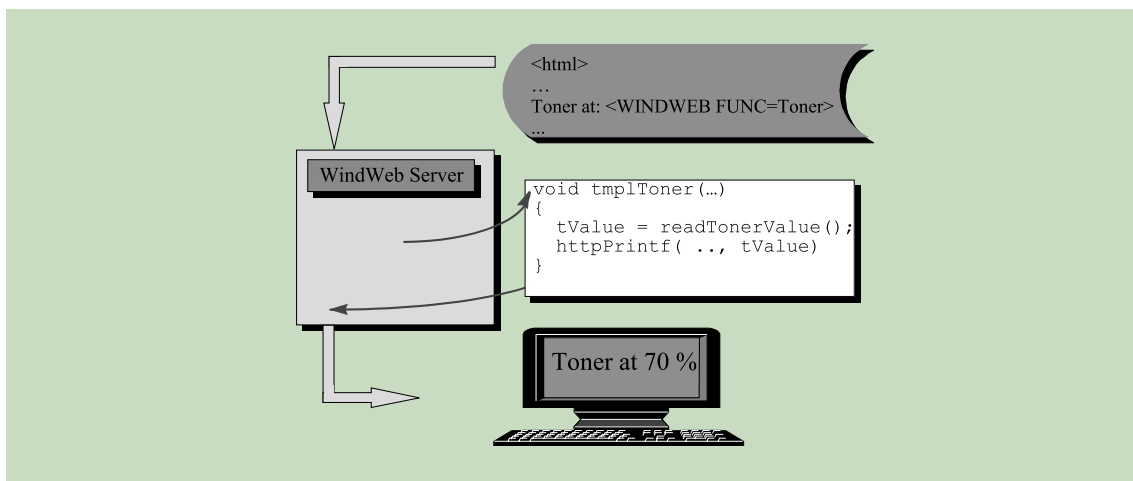


Figure 1.

INTERNET EMBEDDED

page from the application. This lets the defaults change in response to current conditions. When the browser returns the form, the server parses the information contained in the form and sends it to the function named in each of the form tags.

Unless the EID is to be used purely for serving HTML pages, such as a dedicated Web appliance, the Web server component will be a comparatively small part of the overall design. This means that the server must take up no more space than necessary to cope with the expected demand. The embedded systems designer has to decide on a workable trade-off between server resources, response times and the number of expected simultaneous users. The content and the type of HTTP will also influence the decision.

Frequently, embedded HTTP server will be used to access and alter configuration information. For example, network routers, communications controllers and industrial controllers may use an embedded Web server to let a network manager configure the devices occasionally. Other protocols, such as SNMP or custom protocols may be used to relay alarms or continuous real-time data.

Because the HTTP server's expected workload is expected to be low, the single-tasking mode of the Wind Web Server has the lowest overhead. In this mode, requests are handled sequentially. On a high-performance system, this will not introduce a performance penalty for small numbers of clients: the network delay will often be higher than the processing latency for a simple request.

If the response to a request is expected to take a long time and another request is received shortly afterwards, there is possibility that the browser making the second request will time out before it is acknowledged. If this is likely to be the case, it may be better to configure the server in multitasking mode so that multiple requests can be handled in parallel. This will be the best approach for situations where there is likely to be a large number of simultaneous users or where the pages contain a large number of linked elements such as graphics or audio.

To provide the embedded systems designer with a high degree of control over the Web server's environ-

ment, tasks and resources are not dynamically allocated in single- or multitasking mode. At start-up, the server configures and initialises a pre-determined number of tasks. Each task is activated when a new request comes in. This method ensures that precious memory is not allocated to the Web software on a dynamic basis and helps maintain high-speed response for Internet clients.

Authentication is likely to be a key component of any Web-enabled EID. Most applications will need fine-grain control over user access rights, with access granted on a page-by-page basis. Unless the EID is designed to be a publicly accessible instrument, it should only transmit HTML pages to users with a password or operating from a known domain or user group.

Using off-the-shelf software, such as the Wind Web Server, embedded systems designers can take advantage of the Internet to deliver information to clients anywhere on the network. At the same time, robust authentication procedures ensure that only users with the correct passwords can gain access to sensitive data. Coupled with any standard browser, the Web Server can bring a graphical front panel to even the most deeply embedded application, with minimal impact on the target system.

Alex Wilson graduated from Imperial College of Science and Technology, London, with a BSc (ENG) in Electrical Engineering. He went on to work for British Aerospace in Bracknell working in the Quality Engineering Department. He worked there for 4 years writing real time code for VME Based Laser Gyro Test Equipment. He then worked for Motorola Computer Group for five years working with the 68K and Power PC VME products. This included support for Motorola's VMEexec Real Time Operating System, as well as supporting other Real-Time Operating Systems on Motorola Boards. Alex Wilson joined Wind River Systems in 1996 supporting the Tornado Development Environment. This includes support for most 32-bit Embedded Microprocessors as well as Unix and Windows development tools. More recently I have been working on Tornado for Java including the Wind Web Server, Java VM and TrueFFS (Flash File System).

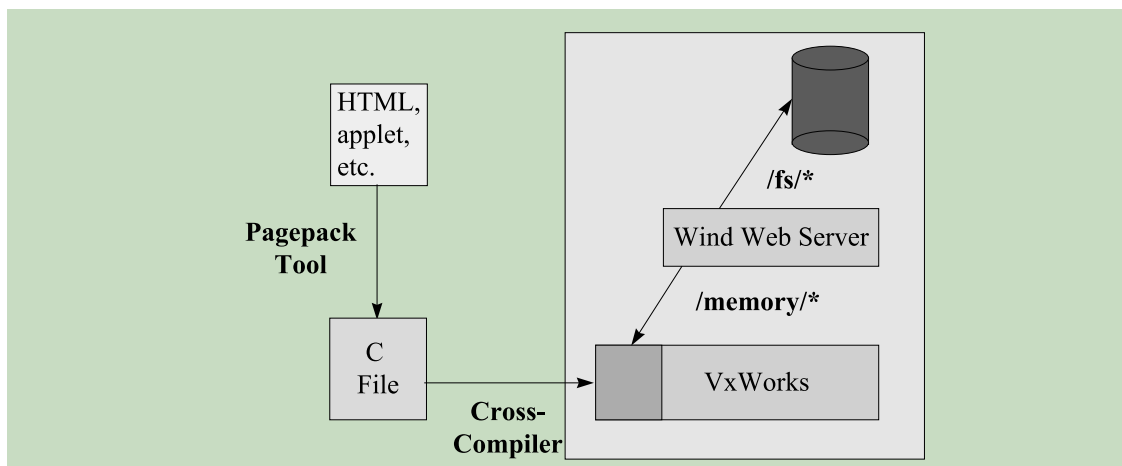


Figure 2.