

Embedded Web Servers

nilsen elektronikk as

Jan Erik Nilsen

Jan 28. 2002

nilsen elektronikk as

Gml. Drammensvei 12B

N-1369 Stabekk

Norway

Tel: +47 67 58 31 62

Fax: +47 67 58 97 61

<http://www.nilsenelektronikk.no>

TABLE OF CONTENTS

1	<i>LICENSE AGREEMENT / DISCLAIMER</i>	3
2	<i>Brief about web servers</i>	4
3	<i>Welcome to VirtualPower</i>	5
3.1	Introduction	5
3.2	Getting started with the TINI board	6
3.2.1	Preparing	6
3.2.2	Communication between TINI and Application	7
3.3	Getting started with the SanPeople EtherPAD E88	8
3.3.1	Preparing	8
3.3.2	Communication between EtherPAD and Application	9
3.4	VirtualPower Print Screens	10
4	<i>A Lean Linux Web Server</i>	14
4.1	Introduction	14
4.2	CGI Example: Image viewer	14

1 LICENSE AGREEMENT / DISCLAIMER

Copyright © nilsen elektronikk as

The Embedded Web Server “**vpws**”, the Lean Linux Web Server “**ws**” and this documentation are properties of **nilsen elektronikk as**, Norway. The software is free; you can use it, redistribute it and/or modify it under the terms below. By using, changing or redistributing the software, you accept the conditions below:

1. You are not allowed to remove or modify this copyright notice and License paragraphs, even if parts of the software are used.
2. The improvements and/or extensions you make **shall** be available for the community under **this** license, source code included. Improvements or extensions, including adaptations to new architectures, **shall** be reported and transmitted to Nilsen Elektronikk AS.
3. You must cause the modified files to carry prominent notices stating that you changed the files, what you did and the date of changes.
4. You may **not** distribute this software under another license without explicit permission from Nilsen Elektronikk AS, Norway.
5. This software is free, and distributed in the hope that it will be useful, but **WITHOUT ANY WARRANTY**; without even the implied warranty of **MERCHANTABILITY** or **FITNESS FOR A PARTICULAR PURPOSE**. You **SHALL NOT** use this software unless you accept to carry all risk and cost of defects or limitations.

The software is provided «as is» and without any express or implied warranties, including, without limitation, the implied warranties of merchantability and fitness for a particular purpose. The software should not be used within Life Support Systems. Life Support Systems are equipment intended to support or sustain life and whose failure to perform properly used in accordance with instructions provided can be reasonably expected to result in significant personal injury or death.

2 Brief about web servers

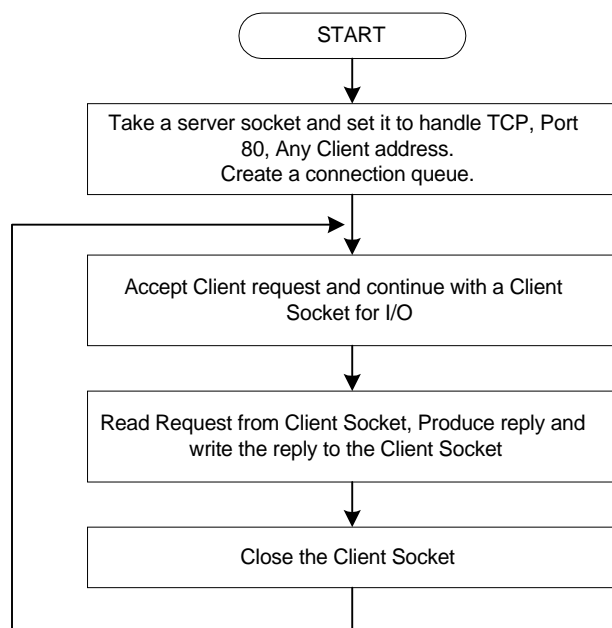


Figure 1. Web server flowchart

When the Web Browser requests a page, it opens a socket to the server. The server accepts the client request and reads page name, parameters etc.

After processing the request, a reply is made, either by retrieving a static HTML page, an image, a Java Applet or whatever or producing a dynamic one. The reply is written to the client socket.

Closing the client socket marks end of session. If the socket remains open, the Web Browser waits and waits until it times out.

The web server supports GET request. For an understanding of the request and reply formats, the following reading is highly recommended:

- “TCP/IP Lean Web Servers for Embedded Systems” by Jeremy Bentham. CMP Books 2000, ISBN 1-929629-11-7. The \$50 book is worth every dime.
- RFC1945 “Hypertext Transfer Protocol -- HTTP/1.0” <http://www.ietf.org/>

A typical request of method GETS. “\r\n” means CarriageReturnLineFeed:

```

GET /myhtmlpage.htm?parameter1=this&parameter2=that HTTP/1.1\r\n
Accept: application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,
image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*\r\n
Accept-Language: no\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)\r\n
Host: 192.168.1.38\r\n
Connection: Keep-Alive\r\n
\r\n

```

A typical reply is shown below. Content-type is `text/html` in this case. Other Content-types are `text/plain`, `image/gif`, `image/jpeg`, `image/x-bitmap` or `*/*`. The last option means “anything”:

```

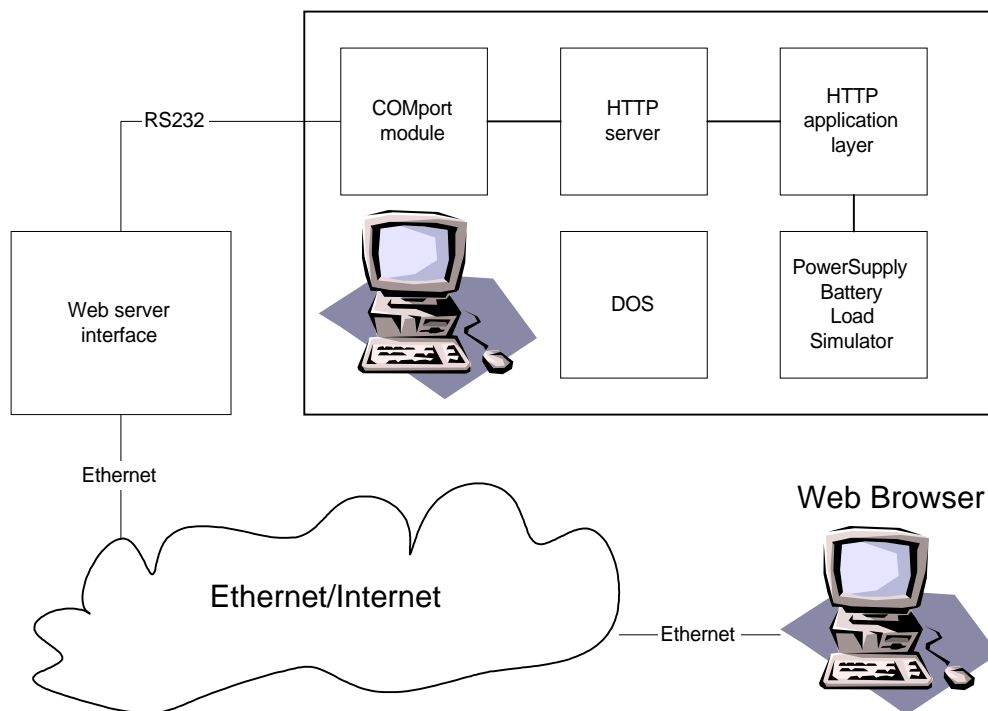
HTTP/1.0 200 OK\r\n
Content-type: text/html\r\n
\r\n
<Here comes the content. In this case the content type is HTML. Other content types
are plain text, Java, image or whatsoever>

```

3 Welcome to VirtualPower

3.1 Introduction

VirtualPower is a virtual company manufacturing Virtual Uninterruptible Power Supplies (VUPS). The system is virtual, not real. Instead of real hardware, a power simulator is used. There are no heavy batteries containing lead in sulphuric acid, no thick mains wires, no noisy power supplies, nothing but a PC and a Web Server Interface. The VUPS can be monitored and controlled using a Web Browser.



The Web Server Interface can be one of:

- TINI board with TINI Socket E10.
- SanPeople EtherPAD TCP+SNMP E88 with appropriate software.

3.2 Getting started with the TINI board

3.2.1 Preparing

1. Download Java from <http://java.sun.com> and TINI software from <http://www.ibutton.com/TINI/index.html>
2. Purchase and prepare a TINI Board. Define an IP address.
3. Connect the TINI Board to Ethernet and to a dedicated PC.
4. Download to the TINI Board and start SerialWebServer.tini
5. Copy vpws.exe to the PC and start the program. NB: The program is a DOS one. WinNT, Win2K or WinXP will not allow vpws.exe to run.
6. Start the Web Browser. Enter `http://` and the IP address.

The TINI Board runs Java code and costs about \$50, plus shipping and handling. The TINI Socket (an Euro-size motherboard for the TINI Board) costs another \$20. The prices are for small volumes down to one item. <http://www.ibutton.com/TINI>



After purchasing, software must be loaded into the TINI Board.

Loading the firmware is described in <http://www.junun.org/TINI/GettingStarted.jsp> and <http://www.ibutton.com/TINI/hardware/started.html> "II. Loading the Firmware Updates". However, there are problems. After consulting the TINI community, the problems were solved.

The first problem was to start JavaKit (A terminal for downloading firmware). Things had to reside in the right directories. In our case:

- c:\jdk1.3.1_01\jre\bin\win32com.dll
- c:\jdk1.3.1_01\jre\lib\javax.com.properties
- c:\jdk1.3.1_01\jre\lib\ext\comm.jar

The second problem was to start JavaKit the right way:

- \jdk1.3.1_01\jre\bin\javaw -cp \tini1.02c\bin\tini.jar JavaKit

The problem is that **javac** resides two places

- \jdk1.3.1_01\bin\ In the PATH
- \jdk1.3.1_01\jre\bin\ But this is the right one

The TINI as well as the Java community are very active, clever and helpful. Problems were solved in short time. This is the big advantage with open software. You never get the same support (if any) with proprietary suppliers, e.g. Microsoft, no matter how much you pay.

The TINI has shown to be very stable. Execution speed is impressive as well, taken in consideration that Java byte code is interpreted by a fast 8051-alike MCU.

TINIOS (TINI Operating System), an Unix-like OS was used. The SerialWebServer.tini application was downloaded by ftp and started with telnet. The environment was surprisingly convenient and free of problems.

Applications can be started by means of a start-up file, much like Unix. Applications can also replace the OS shell. See <http://www.ibutton.com/TINI> for details.

3.2.2 Communication between TINI and Application

See Figure 1. Web server flowchart, page 4, the box with “Read request from Client Socket, produce reply and Write the reply to the Client Socket”.

The request is written to the TINI Serial port. According to the rules, the request ends with an empty line, i.e. “\r\n\r\n”. The request is passed to the application until end is encountered.

Then the application produces reply and writes the reply to TINI. The reply can be of type text or binary, e.g. an image. It can be large and it could overflow the receive buffer if no precautions were taken. The following method has been chosen: Data is transferred in blocks of maximum 1024 bytes. A block is preceded with a header to announce the size of the block. The record format is:

```
<SOH>#<length>;<data>
  <SOH>      :- 0x01 (ASCII Start Of Header)
  <length>   :- ASCII decimal length of <data> [bytes]
  <data>     :- Data block
```

The last record is empty to mark the end. After transmitting a record, the application shall receive acknowledge, 0x06 (ASCII ACK), from TINI before transmitting the next record. Acknowledge timeout is about 3 seconds. TINI times out and closes the Client Socket if there is no reply activity in 5 seconds. The following example shows a transfer of 2000 bytes:

<u>Application transmits</u>	<u>TINI reads the record and transmits</u>
<SOH>#1024;<data 1024 bytes>	<ACK>
<SOH>#976;<data 976 bytes>	<ACK>
<SOH>#0;	<ACK>

SerialWebServer seeks for static files in the TINI directory “/webhome” before bothering the application. If found, it returns the file. Otherwise, the request is forwarded to the application.

3.3 Getting started with the SanPeople EtherPAD E88

1. Purchase a TCP+SNMP E88 product from SAN People or one of its distributors.
2. Replace the E88 software with the software found in **embwebsv.zip** directory **Sanpeople**. Please note that this software is not supported by SAN People. It was created for demonstration purposes only. Changes made to the original behaviour of the software will be described in section 3.3.2.
The E88 is a low cost, entry level product. Other products, with scripting capabilities, are available from SAN People. These products provide a platform to build customized solutions for serial-to-Ethernet problems.
3. General information on functionality and configuration issues concerning the E88 and other SAN People products can be obtained from SAN People’s website at <http://www.sanpeople.com>. The manual can be found under Support | Manuals | E88. (E88 TCP/IP SNMP Manual)
Note, however, that the document you are reading now is the only one describing the slightly different behaviour of the E88 software as implemented for this demonstration.

3.3.1 Preparing

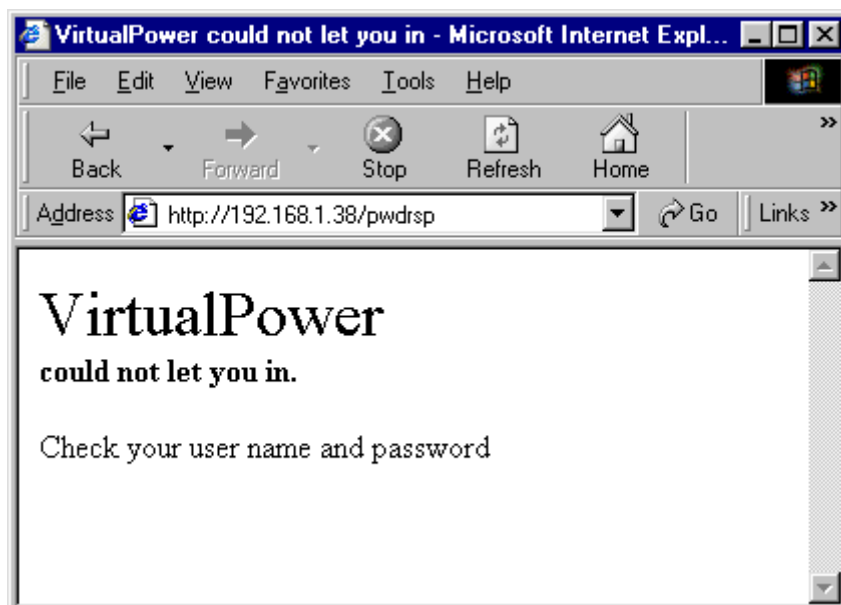
1. Software for the SAN People TCP+SNMP E88 can be found in **embwebsv.zip** directory **Sanpeople**. Upload it to the E88 as described in the product manual.
2. Configure the EtherPAD. Define an IP address.
3. Connect the EtherPAD to Ethernet and to a dedicated PC.
4. Copy vpws.exe to the PC and start the program. NB: The program is a DOS one. WinNT, Win2K or WinXP will not allow vpws.exe to run. vpws.exe must be started with option -e (e for EtherPAD)
5. Start the Web Browser. Enter **http://** and the IP address.

3.4 VirtualPower Print Screens

First you have to log in. The user name is “admin” and password is “1234”. NB: There is absolutely no security in this application. If you want higher security, please read RFC2617 “HTTP Authentication: Basic and Digest Access Authentication”. Thanks to the open “Request For Comments”, things are described.



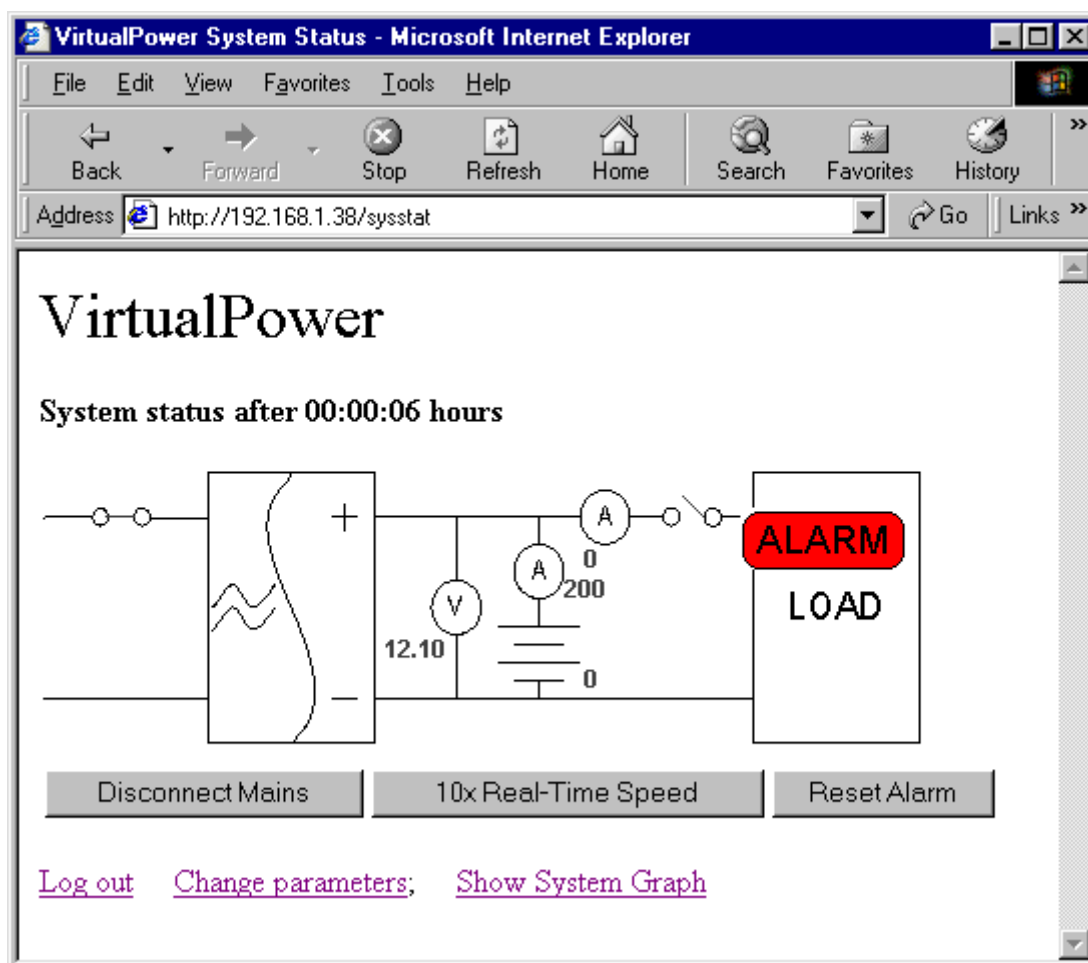
If your name or password is wrong, you end up here:

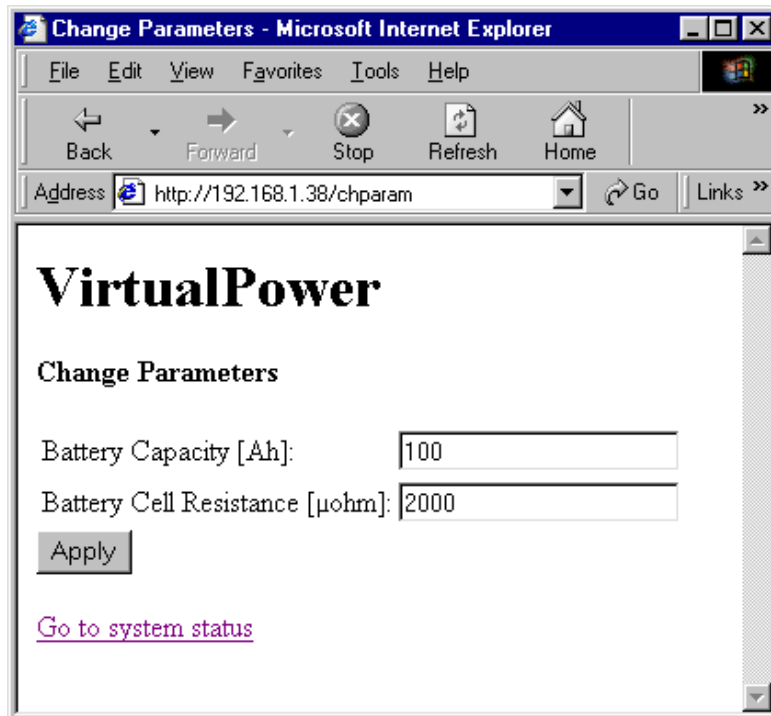


After a successful login, you are passed to the System Status page. The page below shows the situation 6 seconds after power on with an empty battery. A red alarm plate is flashing because the load is disconnected. The power supply delivers 53.5V, but due to the 200 Ampere current limitation, the voltage is low. After a while, the voltage increases to 50V and the load contactor connects. The alarm, however, remains flashing until the “Reset Alarm” button is pressed. When alarm is reset, the “Reset Alarm” button is removed.

There is also one button for the Mains contactor. The button text is “Disconnect Mains” while the contactor is on and “Connect Mains” while the contactor is off. Another button is for impatient system developers. With the “10x Real-Time Speed” activated, the time runs 10 times faster. The button text is then changed to “Real-Time Speed”.

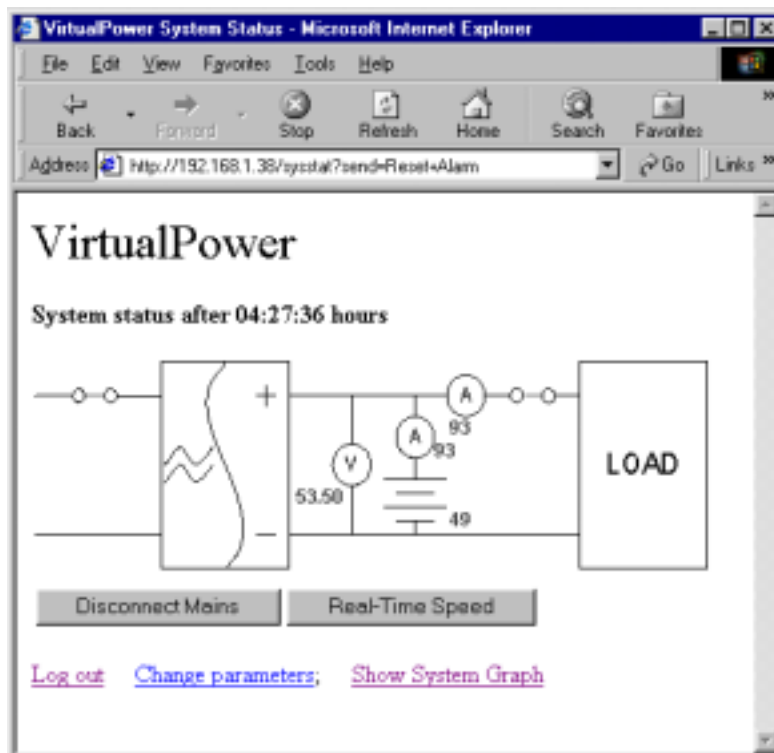
There are two Ampere meters and one Voltage meter. The value next to the battery symbol shows the battery charge in Ampere Hours. The load consumes 5kW while powered.





There is a form for changing parameters. This form can be used to change Battery capacity and cell resistance. The current values are the default values.

Be aware of the security aspect. If a cracker finds a way through the firewall, he may attach the power system.



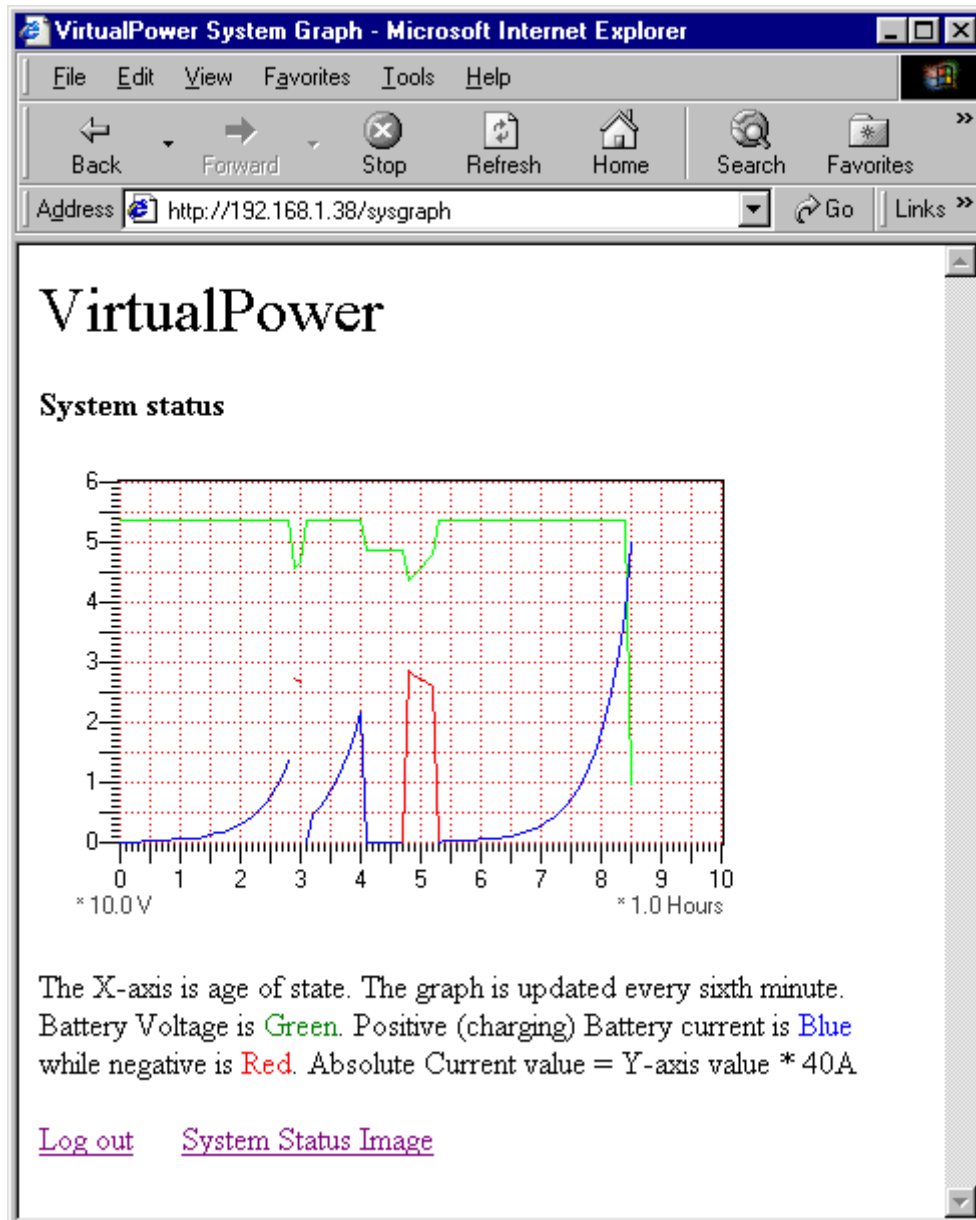
Now the battery is half charged. Everything is normal. The address box shows:

- **http://192.168.1.38,** the servers IP address
- **/sysstat,** the CGI name of the page
- **send=Reset+Alarm.** “send” is the parameter name when a button is pressed. “Reset+Alarm” is the parameter value. The ‘+’ is simply ASCII space. Thus, the last action was “Reset Alarm”.

The next page is a System Graph. From the graph we can see:

Hours ago What happens?

- 8:30 Power on with an empty battery
- 6:00 The battery was full charged
- 5:20 Mains failed. The load was kept powered about 30 minutes
- 4:48 The voltage dropped below 43.2V and the load was disconnected.
- 4:06 Mains back. Recharging starts.
- 3:06 Mains failed for 12 minutes. The load is not interrupted this time.



4 A Lean Linux Web Server

4.1 Introduction

The Lean Linux Web Server consists of 3 .c files and 2 .h files. Total size is less than 1200 lines, more than half of the lines are comments. Isn't it unbelievable? Two of the files are used in the VirtualPower example (webserv.c, webserv.h).

The mode of work can easily be understood by reading the code. Functions are well explained. The recommended reading, see page 4, is still highly recommended.

The Lean Linux Web Server features static as well as dynamic data. CGI is handled by external programs, which are called on demand. Reply parameters are passed with argc and argv.

```
int main(int argc, char *argv[])
```

The request arguments are parsed and transferred in different argv. Two formats are used:

- <argumentname>=<argumentvalue>
- <argumentname>

If the Lean Linux Web Server is started with the debug option, one of the passed arguments is "debug"

Responded data is transferred by use of shared memory.

4.2 CGI Example: Image viewer

One useful CGI program, cgi_im, has been made. It can be used to browse images. Digital cameras have one drawback, it doesn't provide real images, just Jpeg files. Images are seldom printed out and most view programs are inconvenient.

Using CGI solved the problem. Now we have the photo album on every PC connected to our home intranet.

The web server has a Home Directory for web stuff. It can be:
/home/intranet/html

This directory contains static pages as well as cgi programs. The image viewer seeks for image directories under the Home directory:

```
/home/intranet/html/images/summer2001  
/home/intranet/html/images/christmas2000  
/home/intranet/html/images/alicesbirthday  
/home/intranet/html/images/skiing
```

cgi_im can do two things:

- It can generate a “table of contents” with links to the different photo albums (sub directories filled with *.jpg files)
- It can show images from a chosen album

Table of contents

To generate a table of contents, **cgi_im** is called with the following arguments:

```
ptype=cat
dname=<name of common image parent directory>
```

(The second argument is “dname=images” in our case.)

cgi_im lists directories under the “image” directory, sorts the directory names and generates a html page with the following contents:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <title>Photo Album</title>
</head>
<body>
<p><font size="5">Select photo album:</font><p>
<a href="cgi_im?ptype=im&dname=image&dsub=alices_birthday&imno=0">alices_birthday
</a><br>
<a href="cgi_im?ptype=im&dname=image&dsub=christmas2000&imno=0">christmas2000
</a><br>
<a href="cgi_im?ptype=im&dname=image&dsub=skiing&imno=0">skiing
</a><br>
<a href="cgi_im?ptype=im&dname=image&dsub=summer2001&imno=0">summer2001
</a><br>
<p>&nbsp;
</body>
</html>
```

The result is:

Select photo album:

[alices_birthday](#)
[christmas2000](#)
[skiing](#)
[summer2001](#)

Show images from a chosen album

In the HTML code above, you can see 4 CGI examples. The first one is:

```
cgi_im?ptype=im&dname=image&dsub=alices_birthday&imno=0
```




The story ends with this image from a wonderful summer vacation.