Implementation of Embedded Web Server for Mobile Robot System

Noor Azurati Ahmad @ Salleh
Centre for Advanced Software Engineering,
Universiti Teknologi Malaysia *City Campu*s
Jalan Semarak, 54100 Kuala Lumpur. MALAYSIA.

Rosbi Mamat
Faculty of Electrical Engineering,
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor. MALAYSIA.

***Abstract*— With the rapid growth of the Internet, the interest for connecting small devices such as sensors or embedded system appliances into an existing network infrastructure has increased. Web browser is used by remote operator to control and monitor the devices via Internet and these applications are widely utilized in tele-operation, space exploration and tele-training systems. Such devices often have very limited CPU and memory resources and may not be able to run a complete TCP/IP protocol suite. This paper described an implementation of embedded Web server for a mobile robot using minimal TCP/IP stack. The minimal TCP/IP stack has to minimize the standard TCP/IP, in terms of code size and resources to be embedded into minimal systems such as small mobile robots. The performance of the embedded Web server for tele-control of a mobile robot was evaluated and presented.**

Keywords—minimal TCP/IP stack, mobile robot, embedded system, embedded Web server,

I. INTRODUCTION

Web-based embedded systems started with the Cambridge coffee pot Webcam project that appeared in 1991 by using a standard browser. By early 1998 – 1999, there were many solutions based on embedded PCs. In year 2000 many researchers focused on developing standalone embedded Web server that has its own IP address and connected to the Internet independently without PC. Currently, only a limited number of Internet devices are available in the commercial marketplace but much documentation has been written describing the key features and problem areas in the development of embedded Web servers. Nevertheless, some companies and electronic industries have developed an add-on network interface, or a single chip solution for embedding network connectivity into devices.

The use of the Internet as link of telecommunications in Telerobotics enable the control of robots in places where previously was economically unviable. Using Web interface, people can interact with the robot in real time, they can control its movement, observe the state of the different magnitudes and receive the images and sound that robot captures from the environment.

In order to be able to communicate over the Internet, an implementation of the TCP/IP protocol stack is needed. Since many mobile robots are often required to be physically small and inexpensive; typically implemented with 8-bit or 16-bit microcontrollers with few kilobytes of memories, an implementation of the Internet protocols will have to deal with having limited computing resources and memory. The TCP/IP implementation should be sufficiently small in terms of code size and resource demands to be used in minimal systems such as small mobile robots.

This paper described an implementation of embedded Web server for a mobile robot using minimal TCP/IP stack. The embedded Web server was implemented using an Intel i386EX microprocessor board with PC/104 interface. The server stores the available information including the network protocols such as HTTP, TCP/IP, UDP and ICMP, and the application codes. The end user can control the robot by sending data through the Web browser. The HTML codes from embedded Web server are sent out in response to HTTP requests from remote browsers.

II. RELATED WORKS

A. Telerobotics using Web server

Previous Web server systems make used a PC to store all the network protocols starting from the physical layer towards application layer. Those projects used separate microcontroller and microprocessor for processing the robot instruction and the TCP/IP protocol.

Malinowski and Wilamowski [1] reviewed several client-server systems to control the robot movement over the World Wide Web, which were developed at Bradley University in years 1998 to 2001. Robot manipulators performed robust real time manual control over the Internet connection characterized by varying bandwidth and latency. A Web server was used to provide the client application to the operator while the client used custom TCP/IP protocol to connect to the server. Sensors and a video camera provide the feedback to the client.

Different controllers were used in tele-operation projects. For example, the "Lego Robot" was controlled with a parallel port and with a stationary camera located off side. While another project, the Florida Robot [2] utilized a Motorola HC11 microprocessor on the board that implemented motor controllers and collected data from sensors. In this project, the microprocessor connects to the server computer via a serial link.

The Mercury Project [3] and The Telerobotics Experiment via Internet project [4], for instance, were developed to control the robotic arm via Internet. In these projects the robots used by the telerobotics experiment, were attached to the desktop PC that acts as a server.

1-4244-1355-9/07/$25.00 @2007 IEEE

In the above projects, the main concern is how to control the robot through Internet, not how big the system will it be. Therefore, the traditional Web server has no constraints with limited processing power and memory.

In general, a robot is a portable device and more reliable if it can be stand alone device to perform as robotic embedded Web server. Many robot designers use small embedded PC boards for supporting main PC functions including network, I/O ports and data storage.

RW1[1] used Hitachi microcontroller which was connected to a PC/104 computer to control the Magellan robot. The PC/104 provided a microcontroller based upon the Intel 386, 486 or Pentium processor and connects to the Internet using a wireless network card. The use of a PC/104 compliant microcontroller offers a remarkable power and simplicity to the robot control architecture.

The legOS project used Dunkles's TCP/IP stack [5], uIP and embedded in Lego's RCX Mindstorm platform powerd by a h8300 Hitachi microcontroller. The code size of uIP TCP/IP stack is an order of a magnitude smaller than generic TCP/IP. It was used to take care of the packet transmission using LNP integrity packets. The LNP protocol is a quite simple like UDP protocol, and specifically designed for LegOS. This system stored the uIP TCP/IP stack in the embedded PC.

As a summary, many of previous works used standard TCP/IP to enable the Internet connectivity for robot system and they used serial and parallel ports as medium to connect the robot and the PC. Many Web server systems were developed using high end onboard computer such as laptop and PC/104 that can run Java and support WLAN functions.

In this paper, the Web server was developed using PC/104 that does not has those high end capabilities so that we can identify the actual minimum hardware requirements for maintaining the Internet robotic system.

The LegOS project seems to be similar to our project due to the usage of the lean TCP/IP, uIP [6]. The major different between LegOS project and our work is that the Hitachi H8 microcontroller runs LegOS operating system while our system used embedded DOS operating systems which is pported by PC/104.

III. EMBEDDED WEB SERVER IMPLEMENTATION

Implementing the embedded Web server on small mobile robot requires very small TCP/IP to suite with the size of memory in the embedded microprocessor. Versatile TCP/IP stack facilities such as an SMTP (Simple Mail Transfer Protocol) client to send emails or a POP3 (Post Office Protocol 3) client to fetch emails from a server are not required. The main requirement of Web server is that it must accept a request string in hypertext transfer protocol (HTTP) format. The HTTP protocol essentially works by exchanging text messages followed by transfer of web data across TCP connection.

A. Designing the Minimal TCP/IP Stack

There are exists numerous tiny TCP/IP implementations. One of the most notable implementations is the iPic web server [7], which was implemented on a PIC 12C509A. TCP/IP stack is 256 bytes of code. The files for the web server are stored on an EEPROM chip. The source code for the TCP/IP stack is however, is not available.

LwIP, the small TCP/IP stack, was developed by Dunkles and has a small code size. It was designed for embedded systems with little RAM. A simple tests have shown that it can operate in an environment with very little RAM available. Operating system dependencies is moved into a separate module to simplify porting of lwIP to other operating systems. It consists of several modules and apart from the modules implementing the TCP/IP protocols (IP, ICMP, UDP, and TCP). A number of supported modules are implemented. The supported modules consists of the operating system emulation layer, the buffer and memory management subsystems, network interface functions and functions for computing the Internet checksum. LwIP includes an abstract API which is very similar to the BSD socket API. LwIP was designed so that it is possible to run lwIP without the API presented, thus it saved the memory usage. However, Dunkles's work did not include any performance analysis of the proxy based scheme. Furthermore, performance testing of the lwIP with respect to memory consumption and execution time were not conducted.

The uIP stack is specifically designed for very small systems such as the RCX mobile robot. The stack has very small code size and low RAM usage, configurable at compile time. It uses 23 bytes of RAM for each TCP connection and 2 bytes of RAM for each listening TCP port. Even though it has well documented source code, procedure in building its software and hardware systems were not completely explained.

Another well-known small TCP/IP stack is the TCP/IP Lean [8]. TCP/IP Lean was designed by Bentham and the source code was well organized in his book. This book explained how to develop a protocol from the beginning. He also provides techniques to minimize the TCP/IP in designing miniature Web server. His objective is to create a miniature Web server in C that is potentially useful, in that it can monitor and control real world devices connected to the system's I/O line. The Web server used only 256 bytes of read only memory (ROM) for TCP stack. He used microcontroller PIC16C76 in his project. This microcontroller has 8K words of ROM, 368 bytes of RAM and (flash) programmable memory.

The PICDEM.net/Packet Whacker firmware constructed by Fred Eady [9,10]. The PIC flash memory filing system was programmed successfully to allow the PICDEM.net [11] and Packet Whacker [12] combination to be an embedded Web server.

Base on the studies, it is found that TCP/IP lean and Eady's design were suitable for adoption because of the following reasons:

1. TCP/IP lean provides well-documented TCP/IP development and the source code is available with the book.
2. It is free and simple to understand the development of network protocol from scratch.
3. Eady's work shows some examples of writing TCP/IP programming by using EDTP Packet Whacker which is suitable for small embedded Web server.

B. Software Implementation

The development of minimal TCP/IP software involves designer to analyze almost the four layers (Application,

Transport, Network Access and Physical) of TCP/IP stack model. Each Internet appliance is assigned an IP address so that the networks can identify which node the packet is addressed to and where it is coming from. The protocol field determines the type of upper layer service required by the data packet. The upper layer protocols such as UDP, TCP and ICMP are encapsulated into IP packets.

I. IP Processing

IP packets received by the network interface have to be filtered into ICMP, TCP, or UDP packets. It then depends on the 8 bit value inside protocol field. A value of 1 is for ICMP, number 6 indicates TCP and 17 is for UDP.

Previously, the system was tested in local area network that based on the Eady's design. However, the system needs some modification upon the incoming packet was received. Conceptually, the Ethernet frame brings the data from physical layer and then the data will be sorted into different packet groups. But before that, the source IP address should be filtered to determine either that packet requires IP routing or not. The routing algorithm for embedded Web server is uncomplicated. If the packet arrives from the lower layer, the source IP address (client IP) must be checked to see it is on the same subnetwork (i.e. on the same LAN) with the destination IP address (Web server IP). If it is in the same LAN, the packets will be sent back directly to the client IP; if not, the packets will be sent to a gateway or router, which will forward them to the actual destination.

To design routing process for this system, four components are needed; destination IP address, source IP address, subnet mask and router IP address. Source IP address will be ANDed with subnet mask and the results will determine where the packets should go next. This operation eliminates the host IP address so that the rest of the address field can be compared.

When the incoming datagram is received, the embedded Web server will take the following steps [13].

- If ARP request was received, send back the ARP response to client.
- If a nonbroadcast datagram was received, check the client's IP address and take one of the following steps.
  a. If the client's IP address of the datagram is on the local area network, the datagram does not require routing.
  b. If the client's IP is on a different subnet, forward the datagram to the gateway.

For example, the embedded Web server has the network information as follows,

subnet mask	:	255.255.255.0
IP address	:	161.139.16.2
Gateway	:	161.139.16.250

Once the system received the datagram, the operation should check the client's IP address whether the IP is on its LAN or others. For instance, if the client's IP address is 161.139.16.156, and the logical AND with subnet mask value, 255.255.255.0, will obtain 161.139.16.0. This means that the datagram (ARP reply or ICMP) can be sent directly to the client. Otherwise, if the system received a datagram with source IP is 218.111.94.13 (A Digital Subscriber Line**,** DSL), the system will detect that the client's IP was not on the same LAN and requires forwarding the datagram to the router. Figure 1 illustrates the example of routing process.

To forward the datagram from embedded Web server to router, the device needs to send the datagram using router's MAC address as Ethernet destination address. The 48-bit Ethernet address corresponding to router's IP can be obtained using ARP. It is important to realize that the destination IP address does not change until the packets is received by the recipient. In this case, the destination IP address is the final destination with IP 218.111.94.13 but the link layer address is the 48 bit Ethernet address of gateway1's Ethernet interface. The link layer destination address always contains the link layer address of the next node.

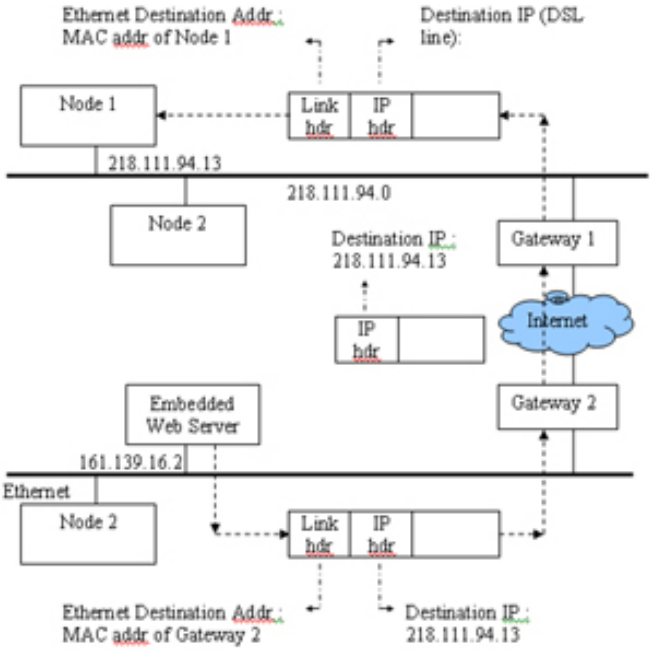

Figure 1. Routing packet

When Gateway 1 receives the datagram, it realizes that the destination IP address is not destined to itself, so it forwards the datagram. Its routing table is searched and the default entry is used. All the routing decisions are based on the destination IP address. The software design does not consider the routing mechanisms or routing policy implementation, but only to address the datagram to go to the router if the destination IP address was not on the same subnet.

In the network layer, IP header is pointed to IP destination address, 218.111.94.13 (client's IP address). On the recipient network, the datagram is received through Gateway 2 as shown in Figure 1. Gateway 2 then realizes that the datagram is destined to one of the nodes on its network. The datagram finally received by the client and the next packets will be traveled using the same way as the first datagram was sent.

During routing process, the router may receive datagrams where the destination IP address does not match its own and send datagrams using a source IP address that is equally alien.

Hence, the router must filter incoming datagrams based on their Ethernet addresses rather than their IP addresses. Any datagram sent to the router's Ethernet address that does not match its own IP address is considered a potential candidate for routing. In this case packets will require multiple hops to make the journey due to the source and destination is not on the same network.

II. TCP Implementation

The toughest part in implementing the network communication protocol is TCP protocol. TCP is a connection oriented protocol which means, a connection must be established between two ends, and in this case we call them as a client and server. In contrast, UDP send the data to the IP layer, does not guarantee either they reach to their destination or not. The TCP provides reliable transmission of data, stream data transfer, efficient flow control, sequencing, error checking and retransmission. Therefore it requires acknowledgement message (ACK) of transmitted and received data across the link.

The TELNET program is the best program for testing TCP packets data movement to the embedded Web server. The PC/104 and Packet Whacker combination will echo the data back to the TELNET session and close the session. By using the TELNET application, the TELNET client must negotiate with TELNET server before establishing a connection between them. An ephemeral port number is applied to get around the negotiation data stream of the well-known TELNET Port 23. Ordinary TELNET port was not used because its negotiation part concerns about some impractical tasks to do with miniature embedded Web server such as terminal parameters and what each side of the TELNET session is expected to do or react to.

III. HTTP design and analysis

The HTTP protocol consists of two different groups of data; the set of requests from browsers to servers, and the set of responses return from servers to browsers.

These groups involve in the replacement of text messages throughout the data transmission between the client and server. This protocol was implemented by using the simplest request of HTTP text messages. Fundamentally, when the clients attempt to access the Web page for the first time, they must enter the IP address of the Web server, for example:

http://10.1.1.13

The browser locates the IP address and opens a TCP connection to server port 80, then uses HTTP protocol to send a request consist of a single GET line:

GET / HTTP/1.0

The GET method is one of the HTTP commands that is used to fetch a Web document. The server replies with a response line containing the HTTP version, status code and description, such as HTTP/1.0 200 OK.

If the request succeeded, the Web page will be sent. The user can control two major application systems, the LED system and Robot system. There are seven different instructions including ON, OFF in LED system, and FORWARD, REVERSE, RIGHT, LEFT and STOP, in Robot system.

Figure 2 shows all commands involved in this research. These commands is transformed to the logical value to execute the system application depends on what the user wishes. For instance, if the FORWARD button was pushed on the displayed Web page; the software system will find which system is called and obtain the value of the instruction. The client sends a GET request together with the significant command such as

```
GET /robot.egi?INSTRUCTION=FORWARD
```

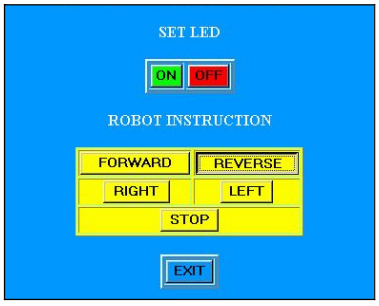

Figure 2. The commands to control the system application

In this case, the robot system has been called and the value is 'FORWARD'. HTTP response to this request by sending out command to the robot and LCD, the robot will move ahead from where it is, while LCD will display the command words. The easiest way to implement the embedded interfaces is that by finding a unique character of the command for a system, purposely to distinguish it from others, and match it with command value of the HTTP request. We used 'F' to identify forward command, 'R' for reverse, 'R' for right, 'L' for left, 'S' for stop, 'N' to turn on the LED and 'F' to turn off the LED. The details of HTTP request and HTTP response for right and left command are shown in Figure 3.

```
No. Time    Delta time  Source  Destination  Protocol Info
122 165.8439 0.009629 10.1.1.12  10.1.1.13    HTTP
                                  GET /robot.egi?INSTRUCTION=RIGHT
123 166.0323 0.188365 10.1.1.13  10.1.1.12    HTTP
                                  HTTP/1.1200OK(text/html)

No. Time    Delta time  Source  Destination  Protocol Info
147 202.3164 0.001643 10.1.1.12  10.1.1.13    HTTP
                                GET /robot.egi?INSTRUCTION=LEFT
148 202.5128 0.196366 10.1.1.13  10.1.1.12    HTTP
                                  HTTP/1.1200OK(text/html)
```

Figure 3. HTTP request to control the system applications

Once the software system found the two suite characters of each command, the application will be executed immediately. This system can serve only one client at a time; hence if other clients would like to browse the Web page, they have to wait until the previous client terminates the communication. The client can control every single instruction as revealed.

IV. SYSTEM PERFORMANCE

Video streaming allows the reception of images and sounds in a continuous ways through the network [14]. Unlike other formats of audio and video in which it is necessary to download a complete file before displaying it, streaming technology allows initiating the visualization while the file is downloading. Teleoperated systems remotely need data or images from the control object, as well as, the transmission of commands through a communication link, that is based upon connection through Internet protocol. The Internet presents a width of heterogeneous band with variable rates of transmission that vary from 10 Kbps (wireless connection) to more than 10Mbps in local net, depending on the Internet connection and the traffic.

With the limitation of bandwidth, applications in real time for video capture present serious restrictions. To overcome these restrictions it is necessary to use data compression and connection to a great speed Internet. Typical rates for video transmission with compression need 20 Kbps (Real Video) and without compression, of 100 Kbps (sequence of images JPEG) with 5 frames/sec [14]. Another limitation is the delay inherent to the protocol TCP, because the packages sent are not necessarily in the same order of the packages received by the client that is not desirable for applications in real time. For application such as audio and video transmission, it does not matter much if the packets take 20 ms or 30 ms to be delivered, as long as the transit time is constant. Having some packets taking 20 ms and other taking 30 ms will give an uneven quality to the sound or image [15].

To determine whether the system can be used for the video transmission via Internet, an example of monochromatic image (black and white) with 53x53 pixels will be used in this analysis. This scheme using 1 bit per pixel, it is also called a 1-bit image. The reason for using this scheme is because of the low bit rate that the embedded Web server can support while transferring data through live network environment. The acceptable frame rate to be at least five frames per second on an embedded Web server to allow a stop-action affect that can be interpreted easily by casual observer. The bandwidth needed to transfer this image by using a 5 frames a second streaming scheme is 14 Kbps [15].

Ethereal software [16] was used for evaluating the performance of embedded Web server. This programs that run on client machine can capture all the data, including HTTP requests and HTTP responses that flow over Internet. By analyzing these data, the performance of embedded Web server can be measured in certain metrics. Performance metrics include items such as request latency (how long it took for an individual response to come back from the server) and throughput (how many responses a server can generate per second).

From Figure 4, it can be seen that throughput is the highest with 1800 bytes per second (14.4 Kbps). The smallest number of throughput is 950 bytes per second. An average throughput of these transmissions is around 1400 bytes per second. In video streaming case study, the total bandwidth available for video is 14 Kbps. This shows that the developed Web server can support the minimum bandwidth of video transmission speed. At the lowest network traffic load, the maximum video streaming speed that can be supported is 14 Kbps at 5 frames/sec. At other time to support video streaming, a much lower rate or less than 5 frame/sec will have to be used.

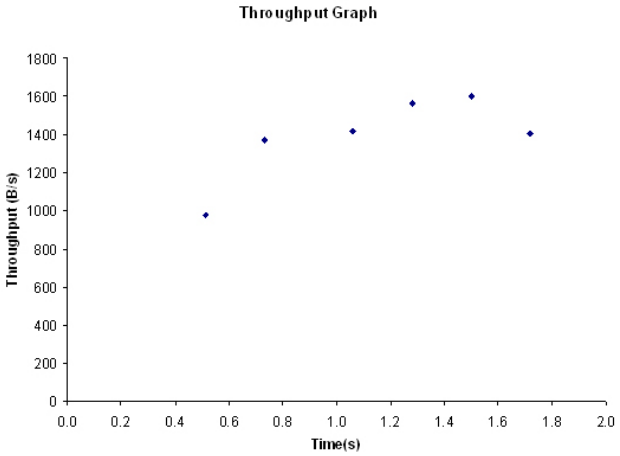

Figure 4. Traffic analysis

This system gave good results for remote robot's control through its communication protocol, which requesting from 2 to 3 K bytes of data. However the feedback through video demands larger bandwidth for applications in real time which is about 64 Kbps (minimum) to make the graphic feedback, through video on-line possible.

Latency is basically the delay or time consumed sending a packet from a node and receiving a response from the recipient. This value is also referred to as the round trip time (RTT). Round trip time (RTT) is an important metric in determining the behavior of a TCP connection.

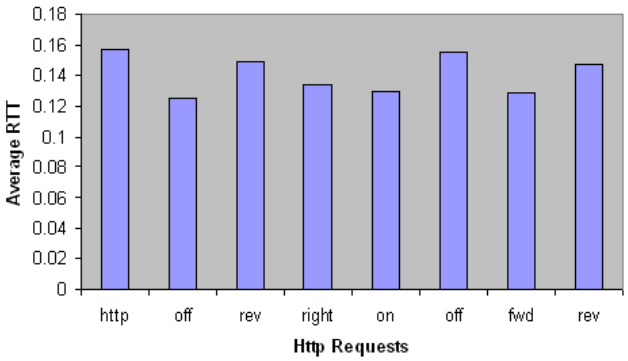

Figure 5. Average RTT for every command

As mentioned above, it is important for video transmission to have a constant transit time to reduce the amount of jitter. Figure 5 illustrates that all commands has almost similar RTT value. It is about 80 percent of the packets be delivered with a delay range of 0.145 sec to 0.15 sec. This could give a good quality to the image transmission via Internet.

V. DISCUSSION

Software design of embedded Web server required consideration of certain parameters such as power consumption, code efficiency in terms of execution time, code size and memory utilization. Results on software and hardware implementation of our embedded Web server are as follows:

A. Power consumption

In autonomous robot application, it is important to prolong the battery life so that the robot can operate in longer time without having to recharge the battery. Thus, a single microprocessor board PC/104 was used for processing both embedded Web server and robot application. This could reduce the power consumption of the system. Previous projects used separate microprocessor for robot system and Web server. Therefore, the power consumption is much higher than using single processor board.

B. Code efficiency in terms of execution time

This has not been analyzed in the current system. But the measurement of Web server performance indicates this indirectly.

C. Code size

From the file.exe file, the total number of code size can be obtained. In designing embedded Web server, the main constraint is to minimize the number of code size for TCP/IP stack. In fact, the number of code size for this system (embedded Web server and robot system) is 48 kilobytes. The functional space of 1 Mbytes FLASH memory in AIM104 is 768 kilobytes because the ROM-DOS has used 256 kilobytes to run its utilities software. Hence, there is 720 kilobytes remain to run the extension of robot application software.

D. Memory utilization

The minimal RAM utilization is very important in developing the small system which has limited memory resources. The number of RAM usage was shown in file.map linker file. In the DATA portion, it stated that the embedded Web server needs only 7220 bytes of RAM to run.

Overall, the miniature Web server has been successfully designed with optimizations of the factors mentioned above. The software system of this project requires very minimal resources in term of RAM and ROM, and this is suitable for small embedded system to enable Web server functionality.

ACKNOWLEDGMENT

A number of people have assisted in this research endeavors over the years and collectively contributed in making this paper, especially Dr Rosbi and my colleagues.

REFERENCE

[1] Malinowski, A. and Wilamowski, B. Controlling Robots via Internet. *1st International Conference on Information Technology in Mechatronics*, Istanbul, Turkey. 2001. 101-107.

[2] Ramos, J.G, Mirisola, G.B, Faria, G. and Bruciapaglia, H. Internet Based Solutions in the Development and Operation of an Unmanned Robotic Airship, *Proceeding of the IEEE*. Vol. 91. 2003.

[3] Goldberg, K., Gentner, S., Sutter, K., and Wiegley, J. The Mercury Project: *A Feasibility Study for Internet Robots*. UC Berkeley and University of Southern California. 1998.

[4] Hu, Hosheng, Yu, Lixiang, Tsui, P.W and Zhou, Quan. Internet-based Robotic Systems for Teleoperation Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK. *International Journal of Assembly Automation,* Vol. 21, No. 2. 1-10.

[5] Dunkles, A. *Design and Implementing of the LwIP TCP/IP Stack*. Master Thesis. Swedish Institute of Computer Science. February 20, 2001.

[6] Christ, O. *TCP/IP Enabled LegOS*. Student research project. University of Applied Sciences Hamburg; 2002.

[7] Shrikumar, H. IPic - a match head sized web-server. Web page. 2000-11-24.Available: http://www-ccs.cs.umass.edu/~shri/iPic.html

[8] Bentham, J. *TCP/IP Lean, Web Servers for Embedded Systems*. 2nd Ed. Lawrence,Kansas, USA.: CMP Books. 2002.

[9] Eady, F. Introducing the Packet Whacker. *Part 1: Hitching a Ride on the PICDEM.NET*. Circuit Cellar Online: Feature Article. October 2001.

[10] Eady, F. Introducing the Packet Whacker. *Part 2: Setting a Course with Code*. Circuit Cellar Online: Feature Article. November 2001.

[11] Microchip Technology Inc. *PICDEM.net Internet/Ethernet Demonstration Board DM163004*. Chandler. 2001.

[12] Edinger, J., EDTP Electronics.Homepage

[13] Stevens, W. Richard. *TCP/IP Illustrated, Volume 1: The protocols*. Indianapolis, IN: Addison-Wesley. 1994.

[14] Tanenbaum, Andrew S. *Computer Networks*. 3rd Ed. Vrije Universiteit, Amsterdam, The Netherlands.: Prentice-Hall International, Inc. 1996.

[15] Ferwon A., Lu W., *Optimizing for Video and Telerobotic Control on Palm OS PDAs*. Ryerson University, Toronto, ON Canada. Network-Centric Applied Research Group.

[16] Sharpe, R. and Warnicke, E. *Ethereal User's Guide: V1.1 for Ethereal 0.8.19*. : User's Guide. 2001.