IMPLEMENTATION OF PIC18F87J10 MICROCONTROLLER USING EMBEDDED WEBSERVER

M. Kesavan[1] S. S. Saravanakumar[2] K.Rajendran[3], P.Anbarasu[4]

[1]PG and Research Department of Electronics, Government Arts College, Paramakudi, Tamilnadu, India
**Corresponding Author:E- Mail ID: mksashwin@gmail.com ORCID ID: https//orcid.org/ 0000-0002-0714-9778*
[2]Department of Physics, Sethupathy Government Arts College, Ramanathapuram, Tamilnadu, India
Email ID: saravanakumarssmsc@gmail.com
[3]Department of Electronics, LRG Government Arts College for Women, Titupur, Tamilnadu. India
Email ID: krishrazen@gamil.com
[4]PG and Research Department of Electronics, Dr. Kalaignar Karunanithi government Arts college, Kulithalai, Tamilnadu.
Email ID: Pon_anbarasu@rediffmail.com

Abstract:

The intention of this research was to design and build an embedded HTTP server using a PIC18f87j10 microcontroller chip. The web server required the implementation of the interface with Ethernet as well as several internet protocols such as TCP/IP and ARP. This embedded web server is able to serve small, static web pages as well as perform certain useful laboratory lab functions such as displaying the current temperature read by the PIC18f87j10 microcontroller from a thermometer, on the webpage. While the capabilities of the embedded web server are nowhere near that of a regular server computer, its small size and relatively low cost makes it more practical for some applications.

Keywords: Embedded web server, PIC18f87j10, EEPROM, TCP/IP ARP, Ethernet and RJ45

1. Introduction

The internet is a versatile, convenient and efficient means of communication in the 21st century (1).Through web page released by embedded web server users can obtain the real-time status information and control remote equipments without time and space restriction. In this paper embedded systems and Internet technology both are combined to form a new technology-the Embedded Internet Technology, which is developed with the popularization of computer network technology in recent years. This technology could function in the hardware and software as long as they are connected. Only by using web browser through the Ethernet and TCP/IP protocol users can get access information of remote devices (3). Protocols such as TCP/IP, UDP, DHCP and ICMP form the backbone of internet communications a large bulk of which consists of Hyper Text Transfer Protocol (HTTP) traffic for the World Wide Web. A HTTP or web server is a server process running at a web site which sends out web pages in response to HTTP requests from remote browsers (2). While high performance 32 bit desktop computers are used for serving websites, much smaller and 16 bit microcontrollers, though not as powerful in terms of processing power can do the job as well. This report details the workings of the embedded web server built for the project. PIC microcontroller is both being versatile and adequate in terms of capability was chosen for this project. Building a HTTP server involved implementing several protocols, namely, UDP, TCP/IP, DHCP and ARP. ICMP was also implemented for testing. The web server was implemented with no problems and worked. The server was able to send a DHCP request for an IP address from a router and served the required webpage on the browser when the IP address of the web server was entered. While the TCP stack is not fully RFC compliant, it is adequate for the purposes of this project. The webpage itself was stored in the flash memory of the AT-Mega32 but future improvements could include adding an external EEPROM to support larger web pages (4). Ethernet provides services corresponding to Layers 1 and 2 of the OSI reference model as shown in figure1. It is standardized as IEEE 802.3 It specifies bus topology with connecting cable with both station and the actual network medium. Ethernet interface consists of MAC controller and PHY interface. The software running on the embedded web server follows the same layered structure as used in the TCP/IP protocol suite. The TCP/IP protocol suite allows computers of all sizes, running different operating systems, to communicate with each other. The TCP/IP protocol suite is a combination of different protocols at various layers. Every layer acts independently from each other. The Link Layer normally includes the device driver in the operating system and the corresponding network interface in the computer. The network layer controls the communication between hosts on the Ethernet. MAC layer is responsible for data packaging, closing, sending and receiving. embedded with MAC but does not provide physical interface. Soit uses ENC28J60 chip to provide Ethernet access channels. ENC28J60 is a low-cost Ethernet LAN controller for embedded applications. Its highly integrated design eliminates the need for costly external components required by other Ethernet controllers. The Address Resolution Protocol (ARP) at network layer translates IP addresses to Ethernet MAC addresses. Internet Protocol (IP) delivers packets to Transmission Control Protocol (TCP), UDP, and Internet Control Message Protocol (ICMP), the ICMP answers to PING requests. TCP/UDP delivers data to the applications. HTTP runs on the top of TCP/IP protocol. It is set of the rules for transferring files like text, image, sound and other multimedia file on the World Wide Web. When Web the applications can communicate with the transport layer through buffer switch data and variables with control information. As soon as a Web user opens their Web browser, the user is indirectly making use of HTTP. When you are set up with direct access to the Internet, your computer is provided with a copy of the TCP/IP program. Implementation of web server using PIC18f87j10 for intelligent monitoring is a new method to monitor a in-dustrial appliances which designed here for the prototype. By using PIC18f87j10 Processor, the embedded system becomes highly precise and gives better performance over traditional 8/16-bit Microcontrollers. The system can also communicate with PC through Serial Port. It supports online-supervision and control not only within Private Network (LAN) but also in Public Network (Internet). The whole system has low-cost, good openness and portability, and is easy to maintain and upgrade. It is possible to interface different kind of sensors with these modules and make various applications. So it can monitor embedded system operation state through Internet, achieving network monitoring purposes. Hence for our future we make the system for domestic environment monitoring, for Ocean environment monitoring, Educational Institution etc.

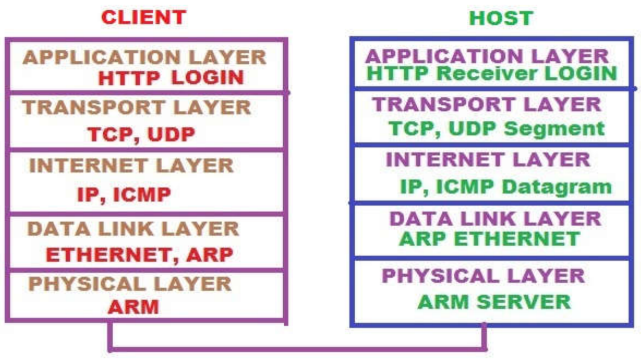

Figure1. Communications in the OSI model

2. Performance of the Circuit Diagram

In figure 2 show the complete schematic diagram for the implementation of the proposed project “EMBEDDED BASED WEBSERVER” which is primarily targeted the applications like remote monitoring and control. Care has been taken in selecting the components with respect to the cost, package, hardware complexity etc to bring optimization in all the parameter during the design. Also the hardware and software code for the web server are designed in such a way to meet any applications requiring remote Internet connectivity.

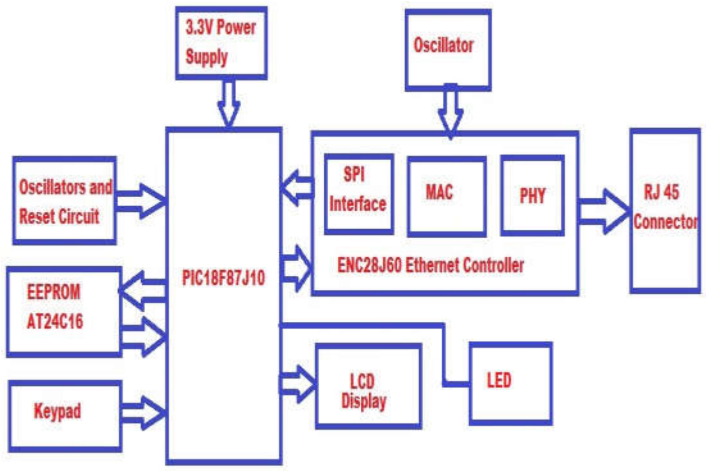

Figure 2. The complete schematic diagram of Embedded based web server

PIC18f87j10 micro controller can be operated in any of the ten different oscillator configuration modes such as LP, XT, HSPLL, RC, and RCIO etc. The XT Type of oscillator is selected in which the user have to connect high frequency crystal/resonator externally to the processor. A 25MHz crystal oscillator is connected between the oscillator pins OSC1 and OSC2 respectively and the same pins are connected with two 22pf ceramic capacitor. The capacitor values are selected to produce acceptable oscillator operations, also the performance of oscillator over the expected voltage and temperature is also tested for this applications. The MCLR pin of the processor will provide a method for triggering an external Reset of the device. Holding the pin low generates a Reset. These devices have a noise filter in the MCLR Reset path, which detects and ignores small pulses. The MCLR pin is not driven low by any internal Resets, including the WDT. The MCLR input can be disabled with the MCLRE configuration bit. When MCLR is disabled, the pin becomes a digital input. Since this pin is low active, normally it should be maintained high using the pull-up resistor. A capacitor of 1 Microfarad between the MCLR pin and the VDD is used to allow the timing of the pulse generated when the reset key is pressed. The other main part of the system design is the Ethernet controller interfaced to the processor using the SPI serial interface. The main by the processor to interact with the Ethernet controller is a SDI, SDO and SCK pin, which is used for data, input, data output and clock synchronization respectively. It is designed to serve as an Ethernet network interface for any controller equipped with SPI The ENC28J60 meets all of the IEEE 802.3 specifications. It incorporates a number of packet filtering schemes to limit incoming packets. It also provides an internal DMA module for fast data throughput and hardware assisted checksum calculation, which is used in various network protocols. Communication with the host controller is implemented via an interrupt pin and the SPI, with clock rates of up to 20 MHz. Two dedicated pins are used for LED link and network activity. The ENC28J60 is designed to operate at 25 MHz with a crystal connected to the OSC1 and OSC2 pins. The ENC28J60 design requires the use of a parallel cut crystal. Use of a series cut crystal may give a frequency out of the crystal manufacturer specifications.

The internal analog circuitry in the PHY module requires that an external 2.32 kΩ, 1% resistor be attached from RBIAS to ground. The resistor influences the TPOUT+/- signal amplitude. On the TPIN+/TPIN- and TPOUT+/TPOUT- pins, 1:1 center-taped pulse transformers rated for Ethernet operations are required. When the Ethernet module is enabled, current is continually sunk through both TPOUT pins. When the PHY is actively transmitting, a differential voltage is created on the Ethernet cable by varying the relative current sunk by TPOUT+ compared to TPOUT-.Each VDD and VSS pin pair should have a 0.1 μF ceramic bypass capacitor (not shown in the schematic) placed as close to the pins as possible. The LEDA and LEDB pins support automatic polarity detection on Reset. The LEDs can be connected such that the pin must source current to turn the LED on, or alternately connected such that the pin must sink current to turn the LED on. Upon system Reset, the ENC28J60 will detect how the LED is connected and begin driving the LED to the default state configured by the PHLCON register. If the LED polarity is changed while the ENC28J60 is operating, the new polarity will not be detected until the next system Reset occurs. An EEPROM of ATMEL AT24C16 of 16Kb NV Memory is interfaced to the processor using the I2C interface which is used to hold the data of the Web page. The MPFC tool will convert any image (Web Page) in to C optimized data structure and this will be stored in the EEPROM. It is the user convenience to keep this constant web page data either in the Internal flash memory or in the External data memory such as EEPROM. The Low active write pin of the EEPROM is connected to VDD since this features is not required in this system design. The power source is connected between the VDD and VSS line of the memory chip and the address line of it is also connected to ground since additional identical chip are not connected in cascade. Low active switches are interfaced to the processor using the same pin is also connected to VDD using the resistors in order to maintain the input pin in to known high level. These switches are used to feed the user requirement such as transmitting the data etc. as well as to know the present status of the processor execution.

2.1 TCP (Transmission Control Protocol): TCP traffic accounts for more than 90% of the internet traffic. It is a interactive connection protocol which deals primarily with end to end reliability, the flow of data in the internet, as well as error checking, retransmission and sequencing. As with UDP, socket calls are used to determine the type of service required and in this case, the well known port 80 is used to indicate a HTTP request. HTTP traffic is sent via TCP and therefore it is the most essential protocol for this project. Functions such as the 3-way handshake synchronization, TCP close connection, checksum, data retransmission and data sequencing were implemented in this project. Many of the other complex protocol functions such as traffic management and multiples connectivity were not implemented since they were redundant for the purposes of this project. TCP has a protocol number of 6 in the IP Protocol ID field (8). The first step in establishing a TCP connection is a 3-way handshake and data transfer which is Show in below figure 3. Client sends a SYN request (SYN flag = 1), Host replies with a SYN and an ACK (SYN, ACK =1), Client sends an ACK (ACK=1) and Connection is established

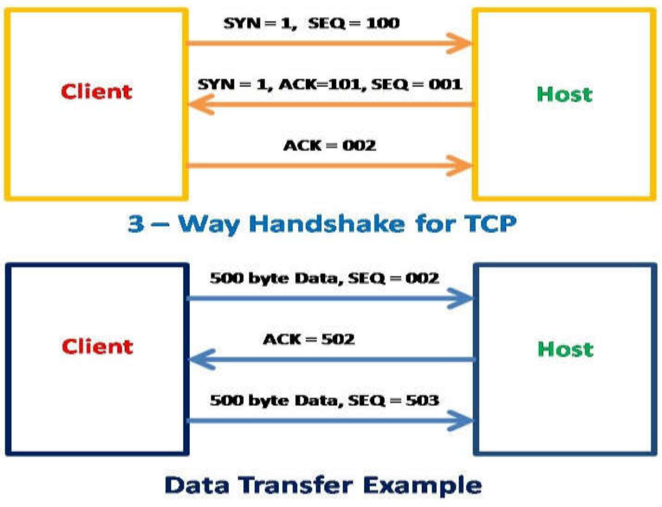

Figure 3 TCP/IP connection is a 3-way handshake and data transfer

After establishing a connection, the host proceeds to send the client data. The steps in which the data is sent is shown below: The host sends the data to the client with a starting sequence number. The client responds by replying with an acknowledgement number which is the sum of the number of bytes in the data received and the sequence number. If the client does not receive all of the data sent, the Host TCP will resend the lost bytes starting from the client's acknowledgement number and if the ACK in the above diagram is 402 instead of 502, the Host TCP will resend the last 100 bytes of the initial data. If an ACK is not received after a certain amount of time, the host will resend the original data and continue to do so until an ACK is received. There are several ways to inform the client that all the data has been sent and the method used in this web server was simply to set the FIN flag when sending the last packet(9). A packet capture program, Ethereal was used to view the packets sent out by the web server. This program was highly essential to checking and debugging the web server. Before the DHCP implementation was done, testing was performed by connecting the web server directly to the Ethernet port of the computer via a cross cable and using Ethereal to check that the DHCP implementation was correct. Once DHCP was implemented, I was able to connect the web server to a router and from there, debug using Ethereal. The web server was mostly tested on a Microsoft router but it was also tested to work seamlessly with a Linksys router. Since the DHCP implementation was as close to the RFC specifications as possible, the web server should be able to work with any DHCP enabled router. When the IP address in this case 192.5.168.5.186 of the web server was keyed in and the temperature reporting function was also working. The TCP data retransmission protocol was also tested by setting the received ACK to always be a certain number less than the expected ACK and the web server was able to resend this number of "lost bits" in the next packets. In addition, the total amount of flash memory used by this project was less than 50% of the available flash memory of the microcontroller which implies that there are 16 kilobytes of memory available for storing additional web pages and images.

3. SOFTWARE DEVELOPMENT TOOLS

MPLAB is windows based integrated development environment (IDE) for the microchip technology with Incorporated PIC microcontroller families. MPLAB allows us to write, debug and optimize the PIC application for firmware product designs. MPLAB also supports the MPLAB-ICE and PIC MASTER EMULATOR, PIC START PLUS, PROMATE all programmers and other microchip or third party development system tools. The organization of MPLAB tools function help make pull down means and customizable quick easy to find and use. MPLAB tools allows us to assemble, compile and link source code and debug the executable logic by watching program flows with the simulator, or in real time with the MPLAB-ICE emulator. By using simulators we can, make timing measurement, view variable in watch window and program firmware with PIC START PLUS or PROMATE. MPLAB is an easy to learn and use integrated development environment. The IDE and use provides firmware development engineer the flexibility to develop and debug for microchip PIC microcontroller families. The MPLAB IDE allows you to create and edit source code providing you with a full-featured text editor. Further the source code can easily debug with the aid of built result window that displays the errors found by the compiler, assembler and linker when generating executable files. A project manager allows to group source files precompiled object files, libraries and linker script files into project format. The MPLAB IDE also provides feature rich simulator and emulator environments to debug the logic of executables, some of ifs features are a variety of windows allows to view the contents of all data program memory locations. Source code program memory and absolute listing windows allows to view the source code it's assembly level equivalent separately and together. The ability to step through execution, or apply break, trace, standard or complex trigger points. The MPLAB IDE integrate several tools to provide a complete development environment. **MPLAB project manager;** Use the project manager to create to a project and work with the specific files to the project. When using a project, source code is rebuilt and downloaded to the simulator or emulator with a singe mouse click. **MPLAB EDITOR:** Use the MPLAB editor to create and edit text files such as source files, code and linker script files. **MPLAB SIM SIMULATOR:** The software simulator models the instruction execution and I/O of the PIC microcontrollers. **MPLAB ICE EMULATOR:** The MPLAB ICE emulator uses hardware to emulate PIC micros in real time either with or without a target system. **MPLAB IDE DEBUGGER:** MPLAB ICD is a programmer for PIC18F87J10 family as well as in circuit debugger. It program hex files into PIC18F87J10 and offers basic debugging features like real time code execution, stepping and break points.

3.1. PIC C30 CROSS COMPILER

The main advantage of using cross compiler is to reduce the time needed to develop the program. By using a cross compiler, we are letting the computer “figure out” the assembly code needed to perform the desired function. Another important advantage is that the higher level code is (generally) more portable. In the event that we have written an assembly program to work with one processor and later on decide to switch to a different processor, the higher-level code will still have to be modified but not as severally as if we had written it directly in assembly code. There is no tool that will magically solve all the problems, and cross compilers are no exception. Compare the amount of memory needed to create HEX file by writing the code directly in the assembly. The amount of program memory (ROM) required when compiling C code is conveniently displayed in the build results window after compiling the project. It can be understood that program memory required has been reduced. Besides a potential increase in code size, there are other issues to be aware of when program timing is critical, than the assembly code is used to see how many clock cycles any port of the program will take to run. This means writing the code in C and then completing to an. As file to see how the program is implemented along the some lines, the MPLAB simulator will not be available to help debug the C code. Finally as with any other software the cross compiler is dot foolproof normally software bugs are known and documented by the manufactures, with the bugs required in the later versions of the software. The best way to handle this issue to check with the company periodically to see if any new problem have been discovered. Despite the drawbacks a cross compiler presents, it is a very powerful tool. As with any other tool, the better we under stand how it works the more useful it becomes for us.

3.2. MPFC Utility

The purpose of using this utility software is to convert the web page into certain file format and whenever the remote system is calling this web server by it unique IP address the PIC micro controller will fetch this web page information from the internal or external memory to the requested person. The MPFS image can be stored in on-chip program memory or an external serial EEPROM. MPFS follows a special format to store multiple files in given storage media, which is summarized in the below figure4.

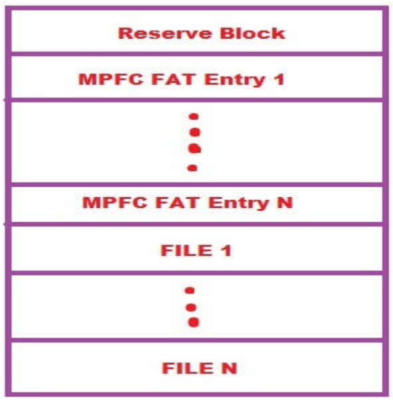

Figure 4 MPFS special format to store multiple files in given storage media

The length of "Reserved Block" is defined by MPFS_RESERVE_BLOCK. The reserved block can be used by the main application to store simple configuration values. MPFS storage begins with one or more MPFS FAT (File Allocation Table) entries, followed by one or more file data. The FAT entry describes the file name, location and its status. The format for the FAT entry is shown in the below table 1.

FLAG	**ADRESS**	**FILE NAME**
8- Bit	16 or 24 Bits	8-Byte + 3-Byte Format
DATA Variable Length	**EOF** 8- Bits	**FFFF** h **or FFFFFF** h 16 or 24 Bits

The *Flag* indicates whether the current entry is in use, deleted, or at the end of the FAT. Each FAT entry contains either a 16-bit or 24-bit address value. The address length is determined by the type of memory used, as well as the memory size model. If internal program memory is used, and the Microchip TCP/IP Stack project is compiled with a small memory model, 16-bit addresses are used. If internal program memory and the large memory model are selected, a 24-bit address is used. The 16-bit addressing scheme is always used for external EEPROM devices, regardless of the memory size model. This implies a maximum MPFS image size of 64 Kbytes for these devices. MPFS uses "short" file names of the "8 + 3" format (8 bytes for the actual file name and 3 bytes for the extension, or NNNNNNNN.EEE). The 16-bit address gives the start of the first file data block. All file names are stored in upper case to make file name comparisons easier. The address in each FAT entry points in turn to a data block that contains the actual file data. The data block format is shown in Figure 5. The block is terminated with a special 8-bit flag called EOF (End Of File), followed by FFFFh (for 16-bit addressing), or FFFFFFh (24-bit addressing). If the data portion of the block contains an EOF character, it is stuffed with the special escape character, DLE (Data Link Escape). Any occurrence of DLE itself is also stuffed with DLE. This application software for the special PC-based program (MPFS.exe) is used to build MPFS image from a set of files. Depending on where the MPFS will ultimately be stored, the user has the option to generate either a 'C' data file or binary file representing the MPFS image. The complete command line syntax for MPFS.exe is

```
mpfs [/?] [/c] [/b] [/r<Block>]
<InputDir> <OutputFile>
```

where

/? displays command line help

/c generates a 'C' data file

/b generates a binary data file (default output)

/r reserves a block of memory at beginning of the file

(valid only in Binary Output mode, with a default value of 32 bytes)

<InputDir> is the directory containing the files for creating the image

<OutputFile> is the output file name For example, the command

mpfs /c <Your WebPage Dir> mypages.c generates the MPFS image as a 'C' data file, mypages.c, from the contents of the directory "Your Web Pages". In contrast, the command mpfs <Your WebPage Dir> mypages.bin generates a binary file of the image with a 32-byte reserved block (both binary format and the 32-byte block are defaults), while mpfs /r128 <Your WebPage Dir> mypages.bin generates the same file with a 128-byte reserved block. Before the MPFS image is built, the user must create all of the Web pages and related files and save in a single directory. If the file extension is "htm", the Image Builder strips all carriage return and linefeed characters to reduce the overall size of the MPFS image. If the MPFS image is to be stored in internal program memory, the generated 'C' data file must be linked with the "Web server' project. If the image is to be stored in an external serial EEPROM, the binary file must be downloaded there. The MPFS Image Builder does not check for size limitations. If the binary data format is selected, verify that the total image size does not exceed the available MPFS storage space.

The file "MPFS.c" in the application source will implements the routines required to access MPFS storage. We do not need to understand the details of MPFS in order to use HTTP. All access to MPFS is performed by HTTP, without any help from the main application. The current version of the MPFS library does not allow for the addition or deletion of individual files to an existing image; all of the files comprising a single MPFS image are added at one time. Any changes require the creation of a new image file. If internal program memory is used for MPFS storage, MPFS_USE_PGRM must be defined. Similarly, if external data EEPROM is used for MPFS storage, MPFS_USE_EEPROM must be defined. Only one of these definitions can be present in "StackTsk.h"; a compile-time check makes certain that only one option is selected. Depending on the type of memory device used, its page buffer size will vary. The default setting of the page buffer size (as defined by MPFS_WRITE_PAGE_SIZE in the header file ("MPFS.h")) is 64 bytes. If a different buffer size is required, change the value of MPFS_WRITE_PAGE_SIZE appropriately.

4. Conclusion

In this project our task is to implement an embedded web server based on PIC18f87j10 microcontroller which can be accessed by any client over internet. The hosted website on server can be transferred successfully when requested from any other system connected to the server. This paper completes the implementation of the application layer HTTP protocol. On this basis, the design of the embedded Web server is completed, and the test runs successfully. Through the web browser, it is indeed possible to remotely log in to the web server and control the peripheral devices on the server side, realizing the dynamic interaction of the network. The server has the characteristics of high transmission rate, high reliability, easy access, and the like, and has broad application prospects.

Disclosure Statement

No potential conflict of interest was reported by the authors

Acknowledgements

This project has been made possible with the support and guidance of Dr. M. Kannan and Dr. K. Venmathi. In addition to providing valuable information and help in Embedded C programming.

References

1) Zhao Hai, "Embedded Internet – an information technology revolution of 21st century," Beijing: Tsinghua University Press, 2002, pp. 198–225.
2) Wenqi Zhang, ShugangXie. The ARM embedded common steel module and integrated system design examples[M] Beijing, China electronic industry press. October 2008.
3) KyasaShobha Rani , "Design of On-line Interactive Data Acquisition and Control System for Embedded Applications" et al, *International Journal of Research in Computer and Communication technology, IJRCCT, ISSN 2278-5841, Volume 1, Issue 6, November 2012.*
4) G.Sunil Kumar, T.Swapna, "Design an Embedded Web Server for Monitoring and Controlling Systems or Devices", *International Journal of Engineering Trends and Technology (IJETT) – Volume X , Issue Y- Month 2013.*
5) Soumya Sunny P, Roopa .M, "Data Acquisition and Control System Using Embedded Web Server", *International Journal of Engineering Trends and Technology- Volume3, Issue3- 2012.*
6) P. Stephen Nischay, S. Latha, "Design of DACS(Data Acquisition and Control System) using Linux", *International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, Volume 2, Issue 11, November 2012).*
7) V. Billy Rakesh Roy, SanketDessai, and S. G. Shiva Prasad Yadav,"Design and Development of ARM Processor Based Web Server*", International Journal of Recent Trends in Engineering, Vol. 1, No. 4, May 2009.*
8) M. Manoj Kumar, G. SrinivasaRaju, "Design and Implementation of the Lab Remote Monitoring and Controlling System Based on Embedded Web Technology*", International Journal of Scientific and Research Publications, Volume 3, Issue 3, March 2013,1 ISSN 2250-3153.*
9) Jie Xiao, FenShiZeng, "Design and Implementation of Embedded Web Server," *The 7th International Conference on Computer Science & Education (ICCSE 2012) July 14-17, 2012.Melbourne, Australia.*
10) Shouqian Yu, Weihai Chen, Li Li, Jianglei Qin, "Development of ARM-based Embedded System for Robot Applications," Beijing University of Aeronautics and Astronautics ,Beijing 100083, China.
11) Manivannan M, Kumaresan: Design of Online Interactive Data Acquisition and Control System for Embedded Real Time Applications *Proc. OF ICETECT 2011.*
12) Pokharkar P. et al., "Design and Implementation of ARM Based Embedded Web Server using Linux" *Journal of Harmonized Research in Engineering, 2(1),2014,36-41,ISSN2347-7393.*