International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015
ISSN 2091-2730

An Embedded Web Server Based Control and Data Logging System

Anoop T R

Mar Athanasius College of Engineering Kothamangalam, anooptr267@gmail.com, 9497171779

Abstract— An embedded system is designed for specific control functions within a larger system, often with real-time computing constraints But when networking technology is combined with it, the scope of embedded systems would be further more. Here design and implementation of embedded web server in LPC1769 is presented. That can be used as a control and data logging system in any embedded systems. Here in this design, a FAT file system is implemented first, then that file system is used for saving and accessing files to and from an SD card. By combining the web server and FAT file system operations the system can save processed data as files, access and manage these files from LAN or internet through web browsers, monitor embedded system status and control embedded system operations.

Keywords— Data Logging System, Embedded Web Server, Secure Digital (SD) Card, LPC1769, SPI (Serial Peripheral Interface), FAT file system, FatFs.

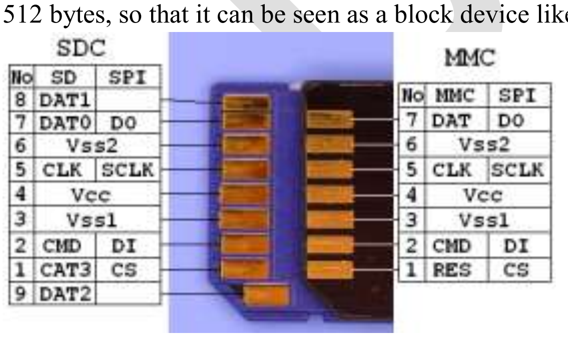
INTRODUCTION

The arrival of internet reduced the whole world communication boundary to that of a single village. After the "everybody in internet wave" now obliviously follows the "everything in the internet wave" .When the embedded device are provided with internet access, it is of no doubt that demand will rise due to the remote accessing capability of the devices[2].

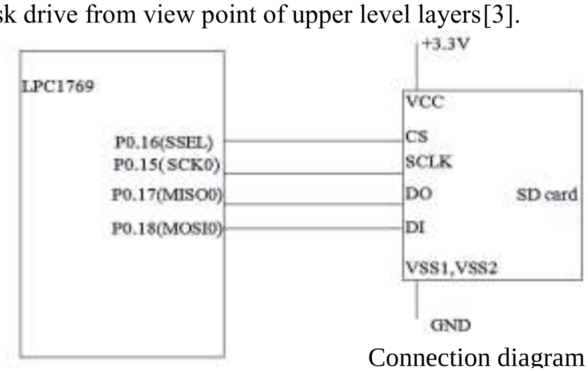
The paper includes complete implementation of an embedded HTTP Web Server and a FAT file system in a LPC1769, which are work together for work as a control and data logging system.The Secure Digital (SD) Card is a non-volatile memory card format developed by the SD Card Association for use in portable devices. It is based on flash memory technology and widely used in digital cameras, cell phones, e-book readers, tablet computers, notebook computers, media players, GPS receivers, and video game consoles. Ever since its adoption in the year 2000, the format has proven very popular and is considered the de-facto industry standard.

SECURE DIGITAL MEMORY CARD

The Secure Digital Memory Card (SDC below) is the de facto standard memory card for mobile equipments. The SDC was developped as upper-compatible to Multi Media Card (MMC below). SDC compliant equipments can also use MMCs in most case. There are also reduced size versions, such as RS-MMC, miniSD and microSD, with the same function. The MMC/SDC has a microcontroller in it. The flash memory controls (block size conversion, error correction and wearleveling - known as FTL) are completed inside of the memory card. The data is transferred between the memory card and the host controller as data blocks in unit of 512 bytes, so that it can be seen as a block device like a generic harddisk drive from view point of upper level layers[3].



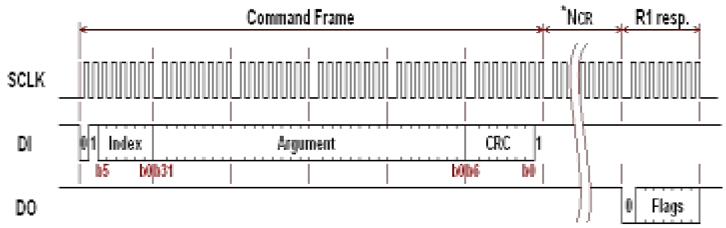
SDC/MMC contact surface



Connection diagram

COMMAND AND RESPONSE

In SPI mode, the data direction on the signal lines are fixed and the data is transferred in byte oriented serial communication. The command frame from host to card is a fixed length packet that shown below. The card is ready to receive a command frame when it drives DO high. After a command frame is sent to the card, a response to the command (R1, R2, R3 or R7) is sent back from the card. Because the data transfer is driven by serial clock generated by host controller, the host controller must continue to read data, send a 0xFF and get received byte, until a valid response is detected. The DI signal must be kept high during read transfer (send a 0xFF and get the received data). The response is sent back within command response time (NCR), 0 to 8 bytes for SDC, 1 to 8 bytes for MMC. The CS signal must be driven high to low prior to send a command frame and held it low during the transaction (command, response and data transfer if exist). The CRC feature is optional in SPI mode. CRC field in the command frame is not checked by the card.



SPI COMMAND SET

Each command is expressed in abbreviation like GO_IDLE_STATE or CMD<n>, <n> is the number of the command index and the value can be 0 to 63. Following table describes only commands that to be usually used for generic read/write and card initialization. For details on all commands, please refer to spec sheets from MMCA and SDCA.

Command Index	Argument	Response	Data	Abbreviation	Description
CMD0	None(0)	R1	No	GO_IDLE_STATE	Software reset.
CMD1	None(0)	R1	No	SEND_OP_COND	Initiate initialization process.
ACMD41(*1)	*2	R1	No	APP_SEND_OP_COND	For only SDC. Initiate initialization process.
CMD8	*3	R7	No	SEND_IF_COND	For only SDC V2. Check voltage range.
CMD9	None(0)	R1	Yes	SEND_CSD	Read CSD register.
CMD10	None(0)	R1	Yes	SEND_CID	Read CID register.
CMD12	None(0)	R1b	No	STOP_TRANSMISSION	Stop to read data.
CMD16	Block length[31:0]	R1	No	SET_BLOCKLEN	Change R/W block size.
CMD17	Address[31:0]	R1	Yes	READ_SINGLE_BLOCK	Read a block.
CMD18	Address[31:0]	R1	Yes	READ_MULTIPLE_BLOCK	Read multiple blocks.
CMD23	Number of blocks[15:0]	R1	No	SET_BLOCK_COUNT	For only MMC. Define number of blocks to transfer with next multi-block read/write command.
ACMD23(*1)	Number of blocks[22:0]	R1	No	SET_WR_BLOCK_ERASE_COUNT	For only SDC. Define number of blocks to pre-erase with next multi-block write command.
CMD24	Address[31:0]	R1	Yes	WRITE_BLOCK	Write a block.
CMD25	Address[31:0]	R1	Yes	WRITE_MULTIPLE_BLOCK	Write multiple blocks.
CMD55(*1)	None(0)	R1	No	APP_CMD	Leading command of ACMD<n> command.
CMD58	None(0)	R3	No	READ_OCR	Read OCR.

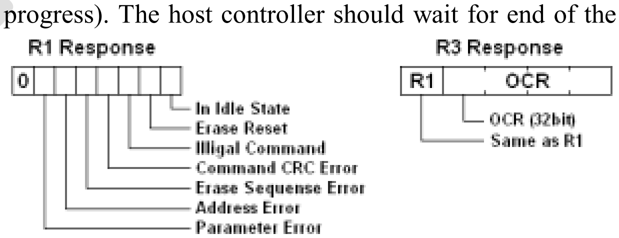
*1:ACMD<n> means a command sequense of CMD55-CMD<n>.

*2: Rsv(0)[31], HCS[30], Rsv(0)[29:0]

*3: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0]

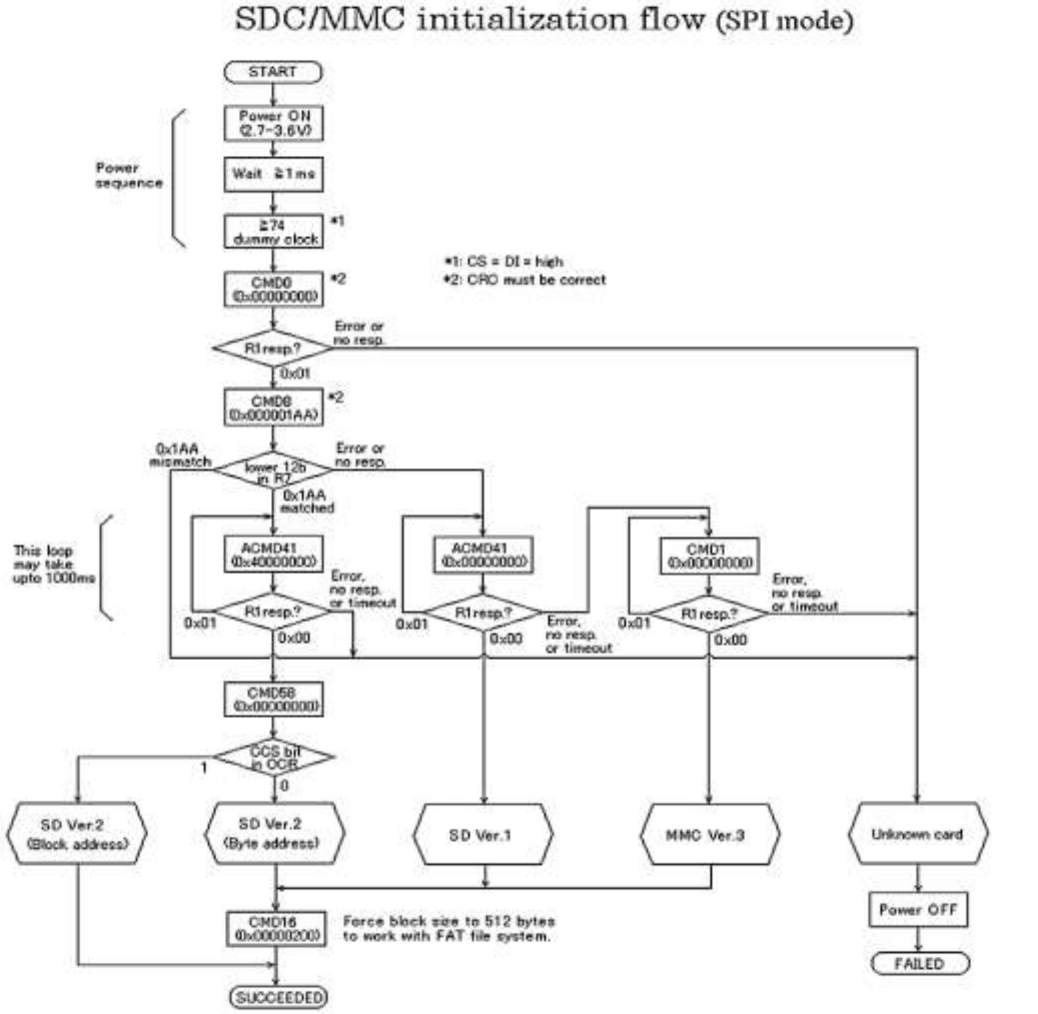
SPI RESPONSE

There are some command response formats, R1, R2, R3 and R7, depends on the command index. A byte of response, R1, is returned for most commands. The bit field of the R1 response is shown in right image, the value 0x00 means successful. When any error occured, corresponding status bit in the response will be set. The R3/R7 response (R1 + trailing 32-bit data) is for only CMD58 and CMD8.Some commands take a time longer than NCR and it responds R1b. It is an R1 response followed by busy flag (DO is driven to low as long as internal process is in progress). The host controller should wait for end of the process until DO goes high (a 0xFF is received).



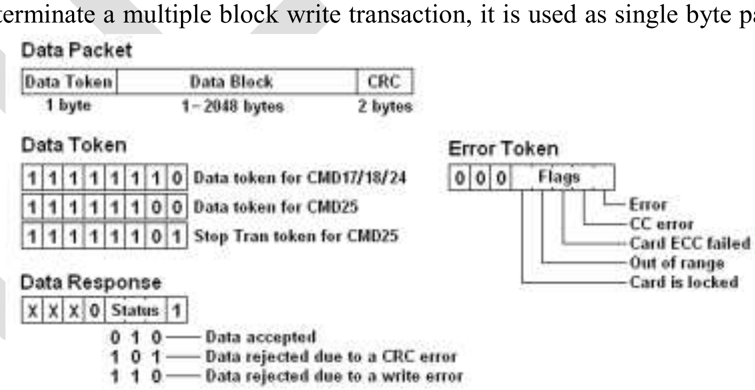
INITIALIZATION PROCEDURE FOR SPI MODE

After power on reset, MMC/SDC enters its native operating mode. To put it SPI mode, follwing procedure must be performed



DATA PACKET AND DATA RESPONSE

In a transaction with data transfer, one or more data blocks will be sent/received after command response. The data block is transferred as a data packet that consist of Token, Data Block and CRC. The format of the data packet is showin in right image and there are three data tokens. Stop Tran token is to terminate a multiple block write transaction, it is used as single byte packet without data block and CRC.



FAT FILE SYSTEM

- FAT stands for File Allocation Table
- The disk is divided into clusters, the unit used by the file allocation, and the FAT describes which clusters are used by which files.
- A FAT file system volume is composed of four basic regions,
    1. Reserved Region
    2. FAT Region

International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015
ISSN 2091-2730

3. Root Directory Region (doesn't exist on FAT32 volumes)
4. File and Directory Data Region

Contents	Boot Sector	FS Information Sector (FAT32 only)	More reserved sectors (optional)	File Allocation Table #1	File Allocation Table #2 ... (conditional)	Root Directory (FAT12/FAT16 only)	Data Region (for files and directories) ... (to end of partition or disk)
Size in sectors	(number of reserved sectors)			(number of FATs) * (sectors per FAT)		(number of root entries*32) / (bytes per sector)	(number of clusters) * (sectors per cluster)

FATFS - GENERIC FAT FILE SYSTEM MODULE

FatFs is a generic FAT file system module for small embedded systems. The FatFs module is written in compliance with ANSI C (C89) and completely separated from the disk I/O layer. Therefore it is independent of the platform. It can be incorporated into small microcontrollers with limited resource[4].

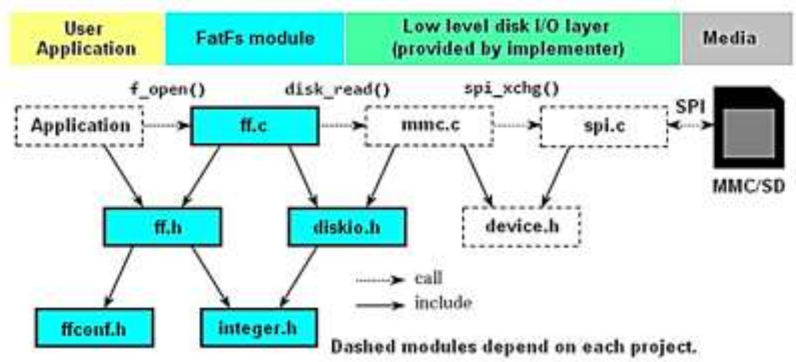
Features:

- Windows compatible FAT file system.
- Platform independent. Easy to port.
- Very small footprint for code and work area.
- Various configuration options:
- Multiple volumes (physical drives and partitions).
- Multiple ANSI/OEM code pages including DBCS.
- Long file name support in ANSI/OEM or Unicode.
- RTOS support for multi-task operation.
- Multiple sector size support upto 4KB.
- Read-only, minimized API, I/O buffer and etc...

FATFS MODULE APPLICATION INTERFACE

FatFs module provides following functions to the applications. In other words, this list describes what FatFs can do to access the FAT volumes.

- f_mount - Register/Unregister a work area
- f_open - Open/Create a file
- f_close - Close an open file
- f_read - Read file
- f_write - Write file
- f_lseek - Move read/write pointer, Expand file size
- f_truncate - Truncate file size
- f_sync - Flush cached data
- f_forward - Forward file data to the stream
- f_stat - Check existance of a file or sub-directory
- f_opendir - Open a directory
- f_closedir - Close an open directory
- f_readdir - Read a directory item
- f_mkdir - Create a sub-directory
- f_unlink - Remove a file or sub-directory
- f_chmod - Change attribute
- f_utime - Change timestamp
- f_rename - Rename/Move a file or sub-directory
- f_chdir - Change current directory
- f_chdrive - Change current drive
- f_getcwd - Retrieve the current directory
- f_getfree - Get free space on the volume
- f_getlabel - Get volume label
- f_setlabel - Set volume label
- f_mkfs - Create a file system on the drive
- f_fdisk - Divide a physical drive
- f_gets - Read a string
- f_putc - Write a character

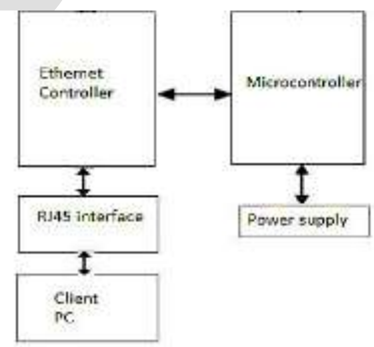


- f_puts - Write a string
- f_printf - Write a formatted string
- f_tell - Get current read/write pointer
- f_eof - Test for end-of-file on a file
- f_size - Get size of a file
- f_error - Test for an error on a file

EMBEDDED WEBSERVER

The implementation of embedded Internet technology is achieved by means of the embedded web server. It runs on embedded system with limiting computing resources to serve web documents including static and dynamic information about embedded system to web browser. We can connect any electronic device/equipment to web server and can obtain the real-time status information and control remote equipments without time and space restriction through web page released by embedded web server. Embedded server is a single chip implementation of the Ethernet networking standard. It consists of two primary elements communicating with each other: i) a server consisting of an ARM processor with an Ethernet controller and ii) a client computer which is connected to controller through this RJ45 interface. The client computer sends/receives data to/from the arm microcontroller using TCP packets. The client has to enter IP address to access this server. This request is taken by the operating system of the client and given to the LAN controller of the client system. The LAN controller sends the request to the router that processes and checks for the system connected to the network with the particular IP address. If the IP address entered is correct and matches to that of the server, a request is sent to the LAN controller of the server and a session is established and a TCP/IP connection is establishes and the server starts sending the web pages to the client through which we can remotely monitor and control the sensors and SD card content.

In this paper embedded systems and Internet technology are combined to form a new technology - the Embedded Internet Technology, which developed with the popularization of computer network technology in recent years. The heart of communication is TCP/IP protocol. Network communication is performed by the IEEE 802.3 Ethernet standard. It is the most modern technology of embedded systems. Since ARM processor has fast execution capability and Ethernet standard can provide internet access with reasonable speed, this system is suitable for enhancing security in industrial conditions by remotely monitoring and controlling various industrial appliances.

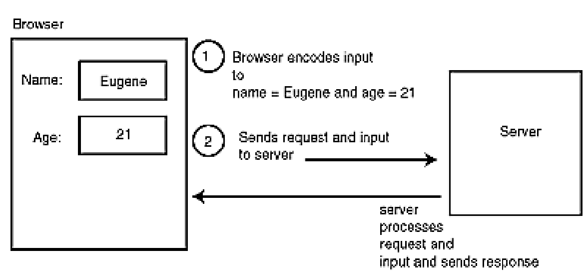


HTTP REQUEST METHODS

Two commonly used methods for a request-response between a client and server are:

1. GET - Requests data from a specified resource
2. POST- Submits data to be processed to a specified resource

post method is used in this work for sending http requests as follows



POST METHOD

PROPOSED SYSTEM

The objective of the project was to develop an embedded web server based control and data logging system using LPC1769

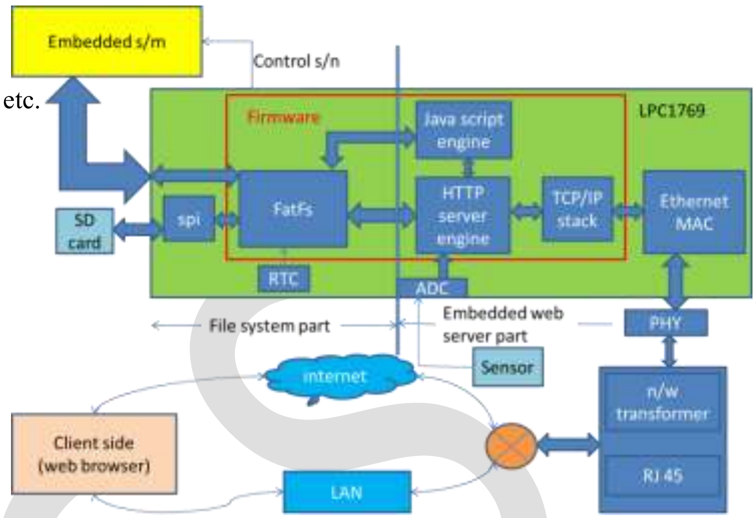
The main features are listed below:

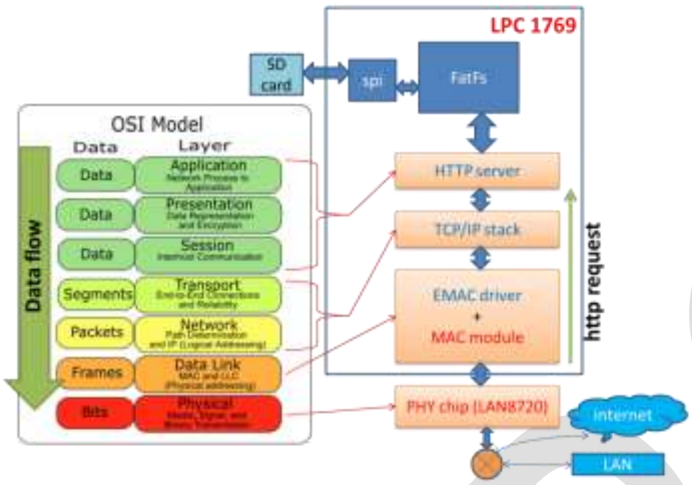
- For saving processed data as files using FAT file system
- Access and manage these files from LAN or internet through web browsers
- Monitor embedded system status by monitoring sensor outputs

- Control embedded system operations by commands from user through web browsers

The main highlights of proposed system are:

- HTML web page based user interface
- User name and password authentication
- Security levels like system log file, emergency shutdown etc.
- Direct file opening and file downloading in browser
- Files & folder list
- Embedded s/m operations and modules control
- Status monitoring through sensor data





For saving processed data as files, processed raw data is given to the FatFs and it will save the data as files in SD card.For access and manage files in the SD card from LAN or internet through web browsers file system module and embedded web server act together.For monitoring embedded system status, sensor output data is transferred through HTML web pages in real time.According to the control commands from the user through the web browser appropriate control signals will generated by the system for controlling the embedded system operations.

ACKNOWLEDGMENT

I am greatly indebted to my academic mentors, whose support has given me the confidence to make a study on the topic and present the Thesis work. I hereby express my heart full gratitude to Prof. Sunny Joseph, Head of the Department of Electronics and communication, for being a great source of inspiration. I will remain indebted to my guide, Dr. Mathew K, Asst. Professor in Electronics and communication Department, for his valuable guidance and help extended to me. His guidance enabled me to complete study of the topic in time and present it well. My deep sense of gratitude goes to all teachers of Electronics and communication department who gave valuable support and guidance. I express my gratitude to the college management and our principal Prof. Geetha B for providing us good library facilities and internet facilities that helped me a lot for the study of topic in detail.

CONCLUSION

An embedded system is designed for specific control functions within a larger system, often with real-time computing constraints But when networking technology is combined with it, the scope of embedded systems would be further more. Here design and implementation of embedded web server in LPC1769 is presented. That can be used as a control and data logging system in any embedded systems. Here in this design, a FAT file system is implemented first, then that file system is used for saving and accessing files to and from an SD card. By combining the web server and FAT file system operations the system can save processed data as files, access and manage these files from LAN or internet through web browsers, monitor embedded system status and control embedded system operations.

Proposed systems future scopes are:

- Can be used with any embedded systems
- Std. file system FAT is used
- Remote data monitoring

- Remote system status check
- Remote system control
- Small size
- Large storage area at cheap cost

REFERENCES:

[1] Design and Development of Air Temperature and Relative Humidity Monitoring System With AVR Processor Based Web Server,Mitar Simić ,NORTH Point Ltd, Member of the NORTH Group

[2] Design of ARM Based Data Acquisition & Control Using GSM & TCP/IP Network,Suraj Patinge,Yogesh Suryawanshi, Sandeep Kakde,Dept. of Electronics Engineering,Y. C. College of Engineering

[3] www.nxp.com

[4] elm-chan.org

[5] ISO 7730, 1984.

[6] K. Samalekas, E. Logaras, E. S. Manolakos, "Embedded Web Server for the AVR Butterfly Enabling Immediate Access to Wireless Sensor Node Readings", SENSAPPEAL 2009, LNICST 29, pp. 145–158, 2010.

[7] T.Tan, "Embedded ATMEL HTTP Server", Master thesis, Cornell University, 2004.

[8] S. Sunny, M. Roopa, "Data Acquisition and Control System Using Embedded Web Server", International Journal of Engineering Trends and Technology- Volume 3 Issue 3, 2012.

[9] I. Hariyale, V. Gulhane, "Development of an Embedded Web Server System for Controlling and Monitoring of Remote Devices Based on ARM and Win CE", International Journal of Recent Technology and Engineering (IJRTE), Volume 1, Issue 2, June 2012.

[10] V. Mishra, "AVR microcontroller based web server", Master thesis, Cochin University of Science and Technology, 2009.

[11] www.embeddedmarket.com

[12] V. B. R. Roy, S. Dessai, S. G. S. P. Yadav, "Design and Development of ARM Processor Based Web Server", International Journal of Recent Trends in Engineering, Vol. 1, No. 4, May 2009.

[13] I. Hariyale, V. A. Gulhane, "Development of an Embedded Web Server System for Controlling and Monitoring of Remote Devices", 2nd National Conference on Information and Communication Technology (NCICT), 2011