CHAPTER 5

REAL-TIME EMBEDDED LINUX BASED DAQ SYSTEM

5.1 INTRODUCTION

Industrial automation is one of the mounting fields that forge many technologies for its growth which makes the automation simple and user friendly. Embedded system based design makes many applications user friendly and surmounts all the disadvantages that subsist in the erstwhile design like CPU Utilization, Memory and power supply. Industrial automation engrosses data acquisition and control system units for a meticulous application.

The proposed system uses the ARM processor that performs like a general purpose computer which collects the data and takes the obligatory action depending on the specific application. The Central Processing Unit (CPU) of the system is S3C2440 16/32-bit microprocessor based target board which contains all the peripherals like RAM, ROM, secondary memory and Ethernet controller with high performance ratio (Hemanth Kumar & Manjunath Iakkannavar 2012). This system overcomes the disadvantages that exist in other design and uses underlying protocols for data communication over the computer networks since the target board itself act as an embedded Web server. The main advantage of the system is lesser computational memory and larger storage memory for the Real-time control of the industrial appliances. Overall the system is a single stand alone multitasking system that performs various tasks like data acquisition, supervise and control.
Using the development of embedded technology, embedded DAQ and remote monitoring technology in production and data monitoring application had become a new trend. Based on ARM-Linux embedded system, combined with open-source software, the design and implementation of embedded DAQ and remote monitor control system are discussed (Xiangyang Li et al 2012). This system realizes the DAQ, data transmission, data storage and remote control functions.

The results show that the scheme is feasible and the Real-time property of data monitoring and remote control system was improved, so it will have broad application prospects. Hacking embedded Linux kernel is both simple and extremely non-trivial. It is simple but the user should have some experience with kernel programming. The Linux Board Support Package (BSP) guide is provided for every board, but there are a few differences in kernel mode programming between vanilla and embedded Linux (Xiang-Hai Li & Jian-Hui Liu 2010), which the user may find useful if doing small patches to already working embedded Linux kernel:

Memory allocation is different, although this fact is hidden from the user by an API. It is still important to understand the mechanism behind this API in order to use it efficiently.

- It is not uncommon to find embedded systems without PCI bus. The SoC devices can be connected directly to a CPU local bus and for external devices; many vendors use their proprietary buses.

Particular kernel may contain many non-standard modifications added by the HW vendor for multiple reasons like performance improvements and workaround for HW bugs. One of the typical examples is a fast path between two network devices which does not go through the Linux network
stack, which is a sort of optimizations that many vendors are doing in order to squeeze more performance from slow embedded processors. BSP sources and HW documentation are used for the application development.

5.2 EMBEDDED DATA ACQUISITION SYSTEM BASED ON LINUX

Embedded DAQ systems are usually developed using the microcontrollers and processors. While developing an application using embedded processor as a CPU, it needs an OS to operate in a multitasking environment. The DAQ system comprises DAQ, processing and transmission units. These units are controlled by the Real-time OS ported on the embedded processor. The commonly used Real-time OS for embedded applications are WinCE, Embedded Linux, µCOS and PSOS (Mason 2002). Embedded Linux systems have an added advantage that the OS is available at free of cost and the user can able to add application module to the kernel that makes the development efficient. There are many vendors to produce products that can support the Linux. This makes the application developer to produce an efficient product at low cost with the available resources from the vendor.

5.2.1 Real-time Systems

Real-time systems are evaluated on the basis of latency and jitter. An embedded system is a kind of special computer system which is application-centric, based on computer technology, scalable on software and hardware, suitable for application system with strict requirements on the functionality, reliability, cost, size and power consumption. With the development of industrial modernization and embedded technology, it has been widely applied in that field (Kreimer 2002).
To meet the needs of these new conditions, designing a new monitor and control system to adapt modern production becomes a top priority. Based on S3C2440A ARM-core processor and ported Linux OS, between database stations and server using CIS communication mode, between remote user and the server using BIS communications mode, details about the design and implementation of an embedded system integrated functions of DAQ, data display and remote monitor control are discussed. The developer needs TCP/IP protocol stack to implement the Embedded Web server (Robson et al 2006).

5.2.1.1 Introduction to TCP/IP protocol suite

The Linux OS supports the TCP/IP protocol, so it can transmit data directly without a PC (Nakul Padhye 2012). This pattern also has many other advantages, such as speeding up the development of the system, facilitating the works of the future expansion, improving the timeliness and accuracy of the meteorological observation and archiving the commands of automatic meteorological observation. User can monitor and control remote temperature and video information. The platform used is Linux and ARM9 processor. The big advantage of this Web server is embedding a PC based Web server into the ARM platform without losing any of its features. It consists of application programs written in C for accessing data through the serial port and updating the web page, porting of Linux 2.6.3x Kernel with application program on ARM9 board and booting it from the RAM.

5.2.2 Software Development Using RTOS

The implementation realizes a Linux platform based DAQ and Control System that helps the user to monitor and control the data with a standard Web browser. The user will be monitoring the data collected by the sensors like temperature, light intensity and smoke which can be remotely
controlled. It consists of application programs written in C and C++ in µVision Keil and QT Integrated Development Environments (IDE) for accessing data through the serial port and updating on the server and web page simultaneously. Porting of Linux Kernel with application program on ARM9 board and all other supporting file systems will be done by using Domain Name Wired (DNW) tool through serial interface.

For the embedded software engineers, starting embedded Linux application development can present a major hurdle. Conceptually, things are very different to develop using an RTOS or bare metal programming. Linux shows its mainframe heritage. To further complicate matters, embedded Linux application development normally requires development tools that run under Linux. There are various choices for open source development tools: both command line and IDE-based. Although this wide choice is very powerful, it represents a fierce challenge as one attempts to find a learning path. The purpose is to provide a unified, coordinated path for embedded developers starting out in embedded Linux programming.

One of the major differences between Linux and a standard embedded OS is the omnipresent command line “shell”. The shell changes the way of embedded software so that it is no longer a binary, which transfers over to a target and then executes. Instead, it can be a series of programs present within a file system. These programs can be started, stopped, configured and monitored from the command line. The command line therefore provides an excellent tool for experimentation, diagnostics and debugging during development, testing and even whilst in the field.

5.2.3 Data Communication Using TCP/IP Protocol

Embedded systems are increasingly using socket-based communications to communicate with the outside world (Robson et al 2006).
A primary reason for choosing embedded Linux is the free, mature and built-in TCP/IP communications stack and tools.

- The fundamentals of how sockets are handled in C, to send and receive TCP/IP requests.
- An example of how to send HTTP GET and POST requests to interrogate devices and send information to central servers.
- Embedded Linux Web servers, from tiny servers such as thttpd and micro_httpd to Apache. Handling HTTP requests in C using CGI.
- Creating an embedded Web browser for an embedded system that allows users to configure an embedded application.
- A practical example using TCP/IP to retrieve temperature values from a number of small embedded devices, then sending the aggregated data to a remote server on the internet.

The key foundation of ARM Trust Zone is the introduction of a “secure world” and “non-secure world” operating mode into the Trust Zone enabled processor cores. This secure world and non-secure world mode split is an orthogonal concept to the privileged/unprivileged mode split already found in earlier ARM cores (Mo Guan & Minghai Gu 2012). On a typical ARM Trust Zone core, secure world and non-secure world versions of all privileged and unprivileged processor modes coexist. A number of System Control Coprocessor (CP15) registers, including all registers relevant to virtual memory, exist in separate banked secure and non-secure world versions.

Security critical processor core status bits (interrupt flags) and System Control Coprocessor registers are either totally inaccessible to non-
secure world or access permissions are strictly under the control of secure world. For the purpose of interfacing between secure and non-secure world a special Secure Monitor Mode together with a Secure Monitor Call instruction exists.

Depending on the register settings of the processor core, IRQ and FIQ-type interrupts are routed to Secure Monitor Mode handlers. Apart from the extensions to the processor core itself, the AMBA AXI bus in a Trust Zone enabled system to carry extra signals to indicate the originating world for any bus cycles. Trust Zone aware System-on-Chip (SoC) peripherals can interpret those extra signals to restrict access to secure world only. In conjunction with the ability to reroute external aborts to Secure Monitor Mode handlers, a secure world executive can closely monitor any non-secure world attempts to access secure world peripherals. To summarize an ARM Trust Zone CPU core can be seen as two virtual CPU cores with different privileges and a strictly controlled communication interface.

5.2.3.1 Application programming interface (API)

ARM has published its own Trust Zone software API specification. Unfortunately this API specification only defines an interface for applications wants to interact with Trust Zone protected “services” through some kind of service manager. The mentioned API specification does not cover most aspects of the backend “service provider API” interface of the service manager. Together with Trusted Logic, ARM has developed its own closed-source Trust Zone software stack, complementing the Trust Zone hardware extensions. An independent approach purely based on an open-source software component is focussed. As a consequence, the term ARM Trust Zone is used to refer to the hardware specific aspects of Trust Zone technology only. All design and prototype ideas are implemented at IAIK on
a Trust Zone aware prototype ARMv6 processor based on the ARM1176JZF-S core.

Open Kernel Labs has developed an implementation of the L4 microkernel with support for ARMv5 and ARMv6 based platforms. Their L4 implementation utilizes hardware isolation mechanisms available on the ARMv5 and ARMv6 platforms. At the time of writing, the OKL4 source tree contains rudimentary support for Trust Zone specific features found an ARMv6 platform. At the time of writing, the user is not aware of any ARM Trust Zone specific support code in the mainstream Linux source tree.

A number of virtualisation style approaches have been integrated into mainstream Linux kernel sources: User-Mode-Linux (UML) is an approach, which allows an adapted Linux “guest” kernel to run as an unprivileged process under the control of a regular “host” Linux kernel. KVM is an alternative approach on X86 kernels with hardware virtual-station extensions. The KVM device driver allows user space applications, to act as supervisors, taking advantage of the processor’s hardware virtualisation extensions.

5.3 SOFTWARE IMPLEMENTATION OF THE SYSTEM

A software development process based on the embedded OS includes: Establishment of cross compiler, the transplantation of Boot loader, transplantation of embedded Linux and root file system. Later the embedded application is ported onto the system. To begin with, system cross-compiler environment using ARM-Linux-GCC-4.3.3 is established. Supervising Boot loader is used as Boot loader. For compiling the kernel on the ARM9 board, cross compiler tools are used to generate executable code for embedded system. The user should configure the kernel 2.6.29 according to the requirements and its compilation to obtain the required image. The ARM-
Linux-GCC cross compiler is employed to generate the executable for the Master system (Jian Ge et al 2010).

5.3.1 **Boot Loader**

Embedded Software development process based on OS includes: the establishment of cross-compiler, the creation of a root file system, the transplant of boot loader, the porting of Embedded Linux and the development of the embedded Web server. ARM Linux GCC cross compiler is used. The function of the boot loader is to initialize the hardware devices and establish memory mapping tables. Thus establishing appropriate hardware and software environment provides an interface to send commands to target board and prepare for the final call to the OS kernel. Domain Name Wired tool (DNW) is used to send the Z-images of the files to the ARM9 board.

The serial interface is used with a baud rate of 115200bps. Linux can port to a wide range of hardware platforms and can run in most of the architecture. Linux has a comprehensive set of editing, debugging and other development tools, graphical interface, a powerful network supporting and rich applications. The data collected by the sensors is first converted from analog to digital form by ADC IC (0809) and the digital data is transferred to the microcontroller.

5.3.2 **Serial Communication Using ARM9**

The microcontroller receives the data and arranges the data in a predetermined format as per user specification. This format of data is transferred to ARM9 controller through UART. The ARM controller distinguishes the individual sensor’s data and places the corresponding values over Web server for monitoring. In the process of controlling, the user wishes
to control sensor data sends a corresponding command signal. The user interface detects the command signal and passes the same. Upon receiving the command signal the micro controller which is in serial interrupt mode, stops sending the sensor data. This in turn processes the corresponding command signal. Thus, after the successful control operation the default process of DAQ will resume. The HTML Web interface is used to monitor and control data. This part is written using Linux sockets. It makes the communication fast and simple compared to standard Web server (Stephen Nischay & Latha 2012).

5.4 REAL-TIME LINUX DATA ACQUISITION SYSTEM

Real-time distributed industrial control systems are flattering one of the vital areas in embedded control applications. Web based control systems are employed for many applications like home automation, industrial automation and medical field. As they can afford a GUI interface to various client server applications which increases the performance and reduces the human errors, occur in the monitored system (Patel Hiren & Patel Dipak 2012). Remote access is performed by using the communication protocols that are used on either sides of the communicating device.

Microcontroller and Microprocessor based systems have been developed for the industrial automation. But most of the system development needs a secondary storage device that stores the acquired data and publishes it as a Web page on the Internet via Web server. Due to the progression of semiconductor techniques, a whole system is developed in a single board that can produce the identical results as though the application is developed for the general purpose system (Goswami et al 2005). In the proposed system, the target board is programmed to be acting as a general purpose computer that work in a multitasking environment to do the various tasks like DAQ and
Web server for remote access of the peripherals and respond to the clients independent of the location (Monmasson et al 2011).

The design and implementation of a DAQ system were carried out with an onboard Ethernet interface allowing remote device control and DAQ through Internet. The DAQ unit of the system can be an important element for building systems that shows a very promising prospect for building manufacturing systems and for industrial applications. The main advantage of the system is memory and power management as the whole system is developed by a single target board. A study has been launched to identify the limitation of the present acquisition system with respect to the long pulse operation. During the study, Advanced RISC Machines (ARM) technologies have been developed in order to realize the continuous DAQ and Real-time data transmission during a long-pulse discharge. This device offers a flexible, portable, multichannel, continuous simultaneous signal acquisition with programmable gain and isolation amplifiers.

5.4.1 Real-time Kernel

RT Core is a POSIX 1003.13 PE51 type Real-time kernel, looks like a multithreaded POSIX process with its own internal scheduler. RTCore can run a secondary OS as a thread, using a small virtual machine to keep the secondary system from disabling interrupts. This is a peculiar model: a UNIX process with a UNIX OS as a thread, but it provides a useful avenue to modularity. RTCore with Linux (RTLinux) and RTCore with BSD UNIX act as the secondary kernel.

Real-time applications run as Real-time threads and signal handlers either within the address space of RTCore or within the address spaces of processors belonging to the secondary kernel. Real-time threads are scheduled by the RTCore scheduler without reference to the process scheduler in the
secondary OS. It is the idle thread for the Real-time system. The virtual machine virtualizes the interrupt controller so the secondary kernel can preserve internal synchronization without interfering with the Real-time processing. Performance is adequate to allow standard PC and single board computers to replace DSPs in many applications.

Figure 5.1 RTLinux run time model

Figure 5.1 illustrates the RTLinux Run Time model. Unlike Linux, RTLinux provides the hard Real-time capability. It has hybrid kernel architecture with a small Real-time kernel coexisting with the Linux kernel running as the lowest priority task. This combination allows RTLinux to provide highly optimized, time-shared services in parallel with the Real-time, predictable and low-latency execution. Besides this unique feature, RTLinux is freely available to the public. As more development tools are geared towards RTLinux, it will become a dominant player in the embedded market. RTLinux is a typical dual-kernel, one is the Linux kernel which provides features like general purpose OS and another one is RTLinux kernel which supports the hard Real-time capability.

The Real-time OS is embedded in the processor for its working under the multitasking environment. Some of the Real-time OS mostly used
in the embedded system are \( \mu \text{COS}, \) Embedded Linux and PSOS (Hua Fang et al 2011). The most widely used OS for any embedded development is the Embedded Linux. The programmers prefer this OS for their development because it is open source and it needs small memory for running on a processor. The main advantage of the Linux kernel is it can be configured according to the application.

The embedded Linux OS is developed for the networked embedded systems that have the TCP/IP protocol stack for data communication over the network (Nakul Padhye 2012). The number of tasks is loaded in the memory and they are given priority depending upon the importance of the task. The interrupts are used to stop the lower priority task and run the higher priority tasks. The user can define the task's importance with the help of the registers in the memory. The OS schedules these tasks for the execution in the processor. The operation of an RTLinux is represented in Figure 5.2.

![Figure 5.2 Real-time Linux architecture](image-url)
5.4.2 Real-time Performance

In general, Real-time performance can be grouped into two broad categories: hard Real-time and soft Real-time (Kreimer 2002). If the system does not respond to an interrupt within a fixed and predictable amount of time, disastrous things can happen. Soft Real-time, on the other hand, doesn’t have any terrible consequences associated with a late answer but relies on deterministic timing of interrupt handlers to achieve top performance. It is important to remember that the actual performance numbers are important. Real-time characteristics are actually more about deterministic behaviour than raw speed.

The bare Linux kernel is not Real-time. No guarantees are provided for deterministic interrupt handling times and the numbers can vary greatly depending on the system load when the interrupt occurs. RTLinux was designed to add hard Real-time characteristics to the Linux kernel. First released in February 1997 by Michael Barabanov and Victor Yodaiken of the New Mexico Institute of Technology, RTLinux is not a separate and modified version of the Linux kernel. After considering this possibility, it was deemed to be a maintenance and compatibility nightmare. Instead RTLinux was designed as a small, independent Real-time kernel and a set of replacements for the Linux kernel routines that enable and disable interrupts. RTLinux also supplies an API that allows the user to create tasks in the Real-time kernel and communicate with processes in the Linux kernel via FIFOs or shared memory.

When RTLinux is included and the Linux kernel calls the CLI or STI to disable or enable interrupts respectively, the request gets routed to an RTLinux routine that remembers the current Linux interrupt state. When interrupts occur they are always handled by the RTLinux kernel and are passed on to the Linux kernel if they were not handled and interrupts are
enabled. This allows RTLinux to react to every interrupt without the Linux kernel getting in the way. This also gets around the problem of the Linux kernel not being able to handle nested interrupts since RTLinux always identifies the interrupt first and can ensure that the kernel is never re-entered.

RTLinux is not only the player in this arena. Real-time Application Interface (RTAI) takes a similar approach in extending the existing Linux kernel with Real-time features like periodic timers that are not present in RTLinux but do not have the wide popularity of RTLinux. There are some problems with the features present in both RTLinux and RTAI. First of all it is clear that the features were bolted on as an afterthought instead of being an intrinsic part of the design. Although RTLinux does an admirable job of accomplishing its goals it tends to be useful in certain classes of applications such as DAQ rather than being more generally applicable. Another issue is that RTLinux does nothing to address the problem of priority inversion, which can be the cause of serious bugs that are hard to reproduce and track down.

Every Real-time embedded system should be run by the RTOS (Puchr & Ettler 2012). In the proposed work Real-time Linux OS is ported on ARM9 processor. Generally all ARM9 processors have the portability with any kind of higher end RTOSs. The embedded Web server application is developed and ported onto ARM9 with this setup. This single ARM board has been acting as a DAQ unit, control unit, embedded Web server and self-diagnosis. All processes are allocated with essential resources and associated with reliable scheduling algorithms and internet protocols followed by ARM processor. This miniaturized setup reduces the complexity and size of the system.

It contains an OS and Web pages to run the application with a large memory space for server functionality. When the configured IP address is entered into the Web browser, the predefined HTML Web pages get displayed
through which the user can remotely monitor and control the sensor and the device status respectively.

5.5 EMBEDDED LINUX ADVANTAGES

1. **Low cost:** The users don’t need to spend time and money to obtain licenses since Linux and much of its software come with the General Public License (GNU). The user can start to work immediately without worrying that this software may stop working anytime because the free trial version expires. Additionally, there are large repositories from which the user can freely download high quality software for almost any task the user can think of.

2. **Stability:** Linux doesn’t need to be rebooted periodically to maintain performance levels. It doesn’t freeze up or slow down over time due to memory leaks. Continuous up-times of hundreds of days (up to a year or more) are not uncommon.

3. **Performance:** Linux provides persistent high performance on workstations and on networks. It can handle unusually large numbers of users simultaneously and can make old computers sufficiently responsive to be useful again.

4. **Network Friendliness:** Linux was developed by a group of programmers over the Internet and has strong support for network functionality. Client and server systems can be easily set up on any computer running Linux. It can perform tasks such as network backups faster and more reliably than alternative systems.
5. **Flexibility:** Linux can be used for high performance server applications, desktop applications and embedded systems. The user can save disk space by only installing the components needed for a particular use. The user can restrict the use of specific computers by installing, for example only selected office applications instead of the whole suite.

6. **Compatibility:** It runs all common UNIX software packages and can process all common file formats.

7. **Choice:** The large number of Linux distributions gives a choice. Each distribution is developed and supported by a different organization. The user can pick the one which is best; the core functionalities are the same. Most software runs on most distributions.

8. **Fast and Easy Installation:** Most Linux distributions come with user friendly installation and setup programs. Popular Linux distributions come with tools that make installation of additional software very user friendly as well.

9. **Full use of Hard disk:** Linux continues work well even when the hard disk is almost full.

10. **Multitasking:** Linux is designed to do many things at the same time; e.g., a large printing job in the background will not slow down the user’s other work.

11. **Security:** Linux is one of the most secured OS. “Walls” and flexible file access permission systems prevent access by unwanted visitors or viruses. Linux users have an option to select and safely download software, free of charge from online repositories containing thousands of high quality
packages. No purchase transactions requiring credit card numbers or other sensitive personal information are necessary.

12. **Open Source:** If the user develops software that requires knowledge or modification of the OS code, Linux’s source code is at the fingertips. Most Linux applications are open source as well.

Today, the combination of inexpensive computers and free high-quality Linux OS and software provide incredibly low-cost solutions for both basic home office use and high-performance business and science applications. The available choices of Linux distributions and Linux software may be overwhelming at first, but if the user knows where to look, it shouldn’t take long to find best online guidance.

**5.6 REAL-TIME TASK ALLOCATION**

Run-time strategy for allocating application tasks to embedded multiprocessor SoC platforms where communication happens via the network-on-chip. This allows the system to better respond to Real-time changes and to adapt dynamically to different user needs. Several algorithms are proposed for solving the task allocation problem while minimizing the communication energy consumption and network contention. It has been overcome by using Real-time Linux OS for power saving by reliable Real-time scheduling algorithms.

It is developed and operated in parallel to an embedded processor and detects any attack that causes the embedded processor to deviate from its originally programmed behaviour. Different characteristics are explored that can be used for monitoring and quantifying trade-offs between these approaches. This hash-based monitoring pattern can detect attacks within one
instruction cycle at lower memory requirements than traditional approaches that use control flow information.

5.7 EMBEDDED LINUX BASED WEB SERVER IMPLEMENTATION USING S3C2440A PROCESSOR

A DAQ system is connected to Web clients via the Internet through the Web browser. The DAQ system needs to relay on the acquired information to send the data to the requesting clients. Embedded processor converts the analog data which acts as a Web server’s database and they are updated every particular time period (Robson et al 2008). Whenever the client wants to access data, it sends the request to the server. This request is taken from the router and then connected to the internet. The Web processes the request made and finally connects to the desired Web server, access the requested data and sends the data to the client.

Embedded Web server acts as an intermediate between the user and the application for accessing and controlling data. The target board is configured as the embedded Web server using the programming code written in C and HTML. This code is very useful in the interaction of the client with the server. Application running on the embedded target board is updated and the information about the updated data is sent to the requesting clients through the standard communication protocols through the browser. The server allows the client to access the DAQ through the username and password. This feature is added to enhance the security of the implemented DAQ system.

The server running the application code compares the user name and password and if it is same to the typed content at the client side, the server allows the user to access the data acquired by the ARM target board. The well-known commands in the HTTP protocol are GET and POST. When a client request for a particular information GET command is used by the
protocol and to put any data in the server, POST command is used by the HTTP protocol. For a system to be acting as a Web server the user need to configure the IP address and the protocol stack for networking is available in the embedded Linux which is used for this application. The HTML Web page is generated by the server that is shown at the client side. The embedded Web server architecture operation using HTTP protocol is illustrated in Figure 5.3.

**Figure 5.3 Embedded Web server architecture**

Many number of tasks like DAQ, process and data communication are carried out over the network (Daming Liu et al 2010). DAQ is implemented with the help of both hardware and software. Most of the embedded systems employ sensors to convert the physical parameters to electrical signals and make the process by ADC that is available in the CPU. DAQ unit consists of sensors like temperature sensor LM35, smoke sensor and the information about the peripherals is monitored and the data is collected in the ARM based embedded board (Li et al 2010).
The Overall System Architecture is shown in Figure 5.4.

![Figure 5.4 Overall system architecture](image)

The MINI2440 target board is a single board computer based on Samsung S3C2440 microprocessor has inbuilt ADC, Web server and Ethernet controller. Use of ADC for analogue data processing by connecting the sensor at the analogue input on the board and Web server updates the Web page at stipulated time which is to be shown at the client side. The board has inbuilt DM9000 network chip and an Ethernet controller perform the communication of packets over the network that send and receive the data for client-server interaction.

The overall system consists of the three major units. One is the DAQ unit, control unit and other is the data communication unit that is similar to an industrial environment. The analog data are acquired by the sensors that convert the external physical parameters to a voltage level that can be processed by the control unit. The control or CPU is an ARM6 processor which controls all other units. The peripheral components like DC motor and
cooling fan status can be viewed in the browser and it can be ON/OFF manually. In the automated mode, DC motor is in ON condition but when the temperature rises beyond the particular limit, DC motor stops and cooling fan starts running. The security of the system is enhanced by providing the smoke detection using the smoke detector. Using the standard web browsers the client can login and view the status as well as control all the peripherals connected to the system via the browser (Qin Bin et al 2012). Data communication is carried using the Ethernet controller present in the target board.

The application code for the proposed DAQ system is developed using the embedded C language and Hyper Text Markup Language (HTML). The ADC conversion process is carried out in the target board that runs the C code for the same. HTML language is used for client-server interaction between the target board and the client via the browser. The OS used for the implementation is embedded Linux which is widely used OS in the embedded applications.

5.7.1 Linux Porting

The boot loader and Qtopia are provided along with the target board. It is loaded using the memory card or the NAND flash. The compiler for the development is installed in the Linux environment like Ubuntu and the application code is compiled with the help of the GCC compiler based on Linux environment. The compiler produces the bin file and this is to be copied in the memory card which can be executed on the target board.

The steps for RTLlinux porting are discussed as follows:

1. USB driver has to be installed and the cable has to be connected to system for copying.

2. Run DNW.exe in tools. Configure to 11500 baud rate. The user can use 2 kinds of serial communications.
3. This helps PC information and board information to be observed in the PC itself. For PC, the CMD window uses USB and for board, Hyper Terminal RS-232 is used.

4. Using DNW download elements in the NAND Flash and Format NAND flash by pressing F.

5. Transfer .bin file for having information about what and how to read from file in Linux Kernel.


5.8 EXPERIMENTAL RESULTS

The implemented system works in two modes, manual mode and auto mode. In case of manual mode the peripherals connected to the target board are controlled by the client through the browser. The client has to login using the Username and Password. The Web page is shown at the client side by embedded Web server where the client has to send commands for controlling the peripherals. The login session shown at the client side is represented in Figure 5.5.

Figure 5.5 Login page
Using proper authentication, the user can login. The overall hardware setup of the proposed work and the information monitored on the PC screen are shown in Figure 5.6.

**Figure 5.6 Overall systems**

In automated mode, depending on the temperature the DC motor and cooling fan is ON/OFF automatically. Figure 5.7 shows the auto mode where the DC motor is ON and cooling fan is OFF. Figure 5.8 illustrates the manual mode where the clients can ON/OFF the DC motor and cooling fan.

**Figure 5.7 Auto mode: DC motor is in ON condition**
Figure 5.8 Manual Mode: DC motor and cooling fan are in ON condition

At the time of any smoke or fire emission, smoke sensor’s alert message is displayed on the client side making the DC motor to OFF and cooling fan and buzzer to ON as shown in Figure 5.9. ON state condition of the cooling fan and the buzzer is based on exceeding the reference level of heat and smoke.

Figure 5.9 Smoke detection alert and cooling fan ON
5.9 PERFORMANCE EVALUATION

The performance of the proposed system is compared with the existing DAQ systems designed using ARM Cortex processor based and ARM7 processor based methods as follows:

Table 5.1 Performance comparison of proposed system with existing systems

<table>
<thead>
<tr>
<th>Features</th>
<th>ARM Cortex Processor Based Method (Patel Hiren &amp; Patel Dipak 2012)</th>
<th>ARM7 Processor Based Method (Shaik 2011)</th>
<th>Proposed Method using ARM9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Machine Interface</td>
<td>Moderately Complex</td>
<td>Moderately Complex</td>
<td>Easy</td>
</tr>
<tr>
<td>Programmable Real-time Unit</td>
<td>Available</td>
<td>Available</td>
<td>Available</td>
</tr>
<tr>
<td>Applications</td>
<td>Medium end applications</td>
<td>Medium end applications</td>
<td>High end applications</td>
</tr>
<tr>
<td>Supply Current</td>
<td>65mA typical, 120mA maximum</td>
<td>240 mA typical (Fully active)</td>
<td>500 mA Maximum</td>
</tr>
<tr>
<td>OS Support</td>
<td>RTX-deterministic Real-time OS and μC/OS-III</td>
<td>μC/OS-II and RTX-Real-time OS</td>
<td>Windows CE 5 and 6, Linux and Android</td>
</tr>
<tr>
<td>Software development cost</td>
<td>Less</td>
<td>Moderate</td>
<td>Less</td>
</tr>
<tr>
<td>Memory</td>
<td>64 KB SRAM, 512 KB flash</td>
<td>256 KB Single Cycle Flash, 64 KB Single Cycle SRAM</td>
<td>64 MB SDRAM, Up to 1GB flash, 256 Byte EEPROM</td>
</tr>
<tr>
<td>Board Support Package</td>
<td>Not available</td>
<td>Not available</td>
<td>Available</td>
</tr>
<tr>
<td>Process/Task function</td>
<td>Multitasking process restricted by computational memory</td>
<td>Multitasking process restricted by computational memory</td>
<td>Multitasking process</td>
</tr>
</tbody>
</table>
The proposed system is targeted on high-end industrial applications and it has enhanced Thumb (16-bit) Instruction set, Debug extensions, 32x8 Multiplier and Embedded ICE macro cell. The power consumption of the proposed method is 0.3A without LCD and 1A with LCD. This method has a lesser computational memory and larger storage memory required for Real-time control of industrial applications. It consists of RAM, ROM and Ethernet controller with a high performance ratio compared to the other system. In the other methods secondary storage device is required to store the acquired data and it needs a separate web server to publish the acquired data as a Web page. In the proposed system the target board act as a Web server and it has more internal memory which eliminates the complexity in the previous methods. The proposed method has both manual mode control and auto mode control without user interface. The information can be stored in the SD card or flash memory for post analysis.

ARM embedded Linux based DAQ can be used with industrial equipment and medical instruments. A remote user can monitor and control the equipment with a simple but enhanced and more powerful user interface without additional hardware. Rapid development in the field of industrial process control demands for high data accuracy and reliability. This embedded ARM system can adapt to the requirements as a whole. In embedded database, the collected data can be stored on the server and can be used for further processing. As ARM core is a fully accustomed and provided with USB host device, video processing can also be implemented at ease.

5.10 SUMMARY

This application creates a primary data acquisition through low power high end processor. Since low power processor ARM9-S3C2440A is being used, power consumption will be reduced. ARM9-S3C2440A consists of 8 ADC channels of 10 bit, as a result this processor has been used to design
low power hardware interface. It enhances the performance of the formal techniques of the monitoring and control systems. The Embedded Linux based DAQ system provides the multitasking application development. One of the applications developed, performs the various tasks like DAQ, control, monitoring and network communication.

The implementation shows the existing methods which need a complex hardware for design can be replaced by the single board for obtaining the same result as the user gets when implemented through the general purpose computer. This application is designed with high secured access to the Web server through user authentication. This system can be enhanced by the online database system that displays the information about the acquired data during the specified periods of the user at the browser side. The information can be stored in the SD card, for which the slot is provided on the target board.