Web-Based Monitoring and Control for BAS Using Multi-Protocol Converter with Embedded Linux

Byoung Wook CHOI†, Member, Kyoung Chul KOH††, Nonmember, and Soo Yeong YI†††, Member

SUMMARY In this paper, a Web-based management system for the building network is described. We developed a multi-protocol converter based on SoC and embedded Linux. It requires an appropriate operating system for handling protocols and an advanced development environment very similar to embedded linux. The multi-protocol converter integrates control networks of RS-485 and LonWorks devices to BAS through TCP/IP protocol or a client with Java applet. The system consists of three-tier architecture, such as BAS or clients, a multi-protocol converter, and control devices. In order to compare the feasibility of system architecture, it was applied to a small BAS system. By using UML, we modeled a Web-based control system with a unified TCP/IP socket communication and the system architecture. The developed system includes the inverter motor control system with modbus protocol for the RS485 network. The experiment results show that the multi-protocol converter using embedded Linux is a flexible and effective way to build a Web-based monitoring and control system.

Key words: remote monitoring and control, Web-based control, embedded system, LonWorks network, BAS

1. Introduction

Since the current personal computer (PC) on the market appears to have a disadvantage in terms of size, weight, and complexity, a smaller and more expedient post-PC has emerged by a recent advance in information technology allowing for information acquisition to occur more freely and apparently. With the advent of the post-PC, a terminology called embedded systems became popular in the new millennium [1]. From the viewpoint of information technology, embedded systems can be defined as electronic control systems designed to carry out a fixed specific function in which the computer system is installed. Recently, driven by the growth of the Internet and the increasing ubiquity of embedded systems, embedded systems are now seen everywhere. In particular, the embedded operating system is mission-critical when building Internet appliances. The role of embedded operating systems is to manage increasingly complex system resources and virtualize the hardware and enable use of off-the-shelf function libraries, drivers, etc. So far, the real-time operating system is widely used to implement a real-time system since it meets developer’s requirements. However, disadvantages of real-time operating systems include a lack of standards, expensive development, and license costs that becomes critical since the embedded Linux is available in embedded systems design; embedded Linux is able to overcome their disadvantages [2]. In Japan, TRON is a de facto standard in embedded industries. However, most embedded systems have become very complex now and they need a more powerful operating system. The characteristics of Linux are suitable for embedded systems [3]. Linuxdevices.com expects that embedded Linux will be the most widely used embedded operating system by embedded engineers in the future [4], [5].

In this paper, we develop a multi-protocol converter to interconnect control devices and the sensors network with BAS in a unified way, in which the CPU is System on a Chip (SOC) and the embedded Linux is running. The control nodes are implemented with LonWorks, which include air conditioner, pump, temperature sensor, the luminous intensity sensor, 7-segment and LED display. And it supports RS485 communication for industrial controllers. An industrial controller is an inverter driving an AC motor. The modbus protocol is implemented for the control node connected with RS485 in a multi-protocol converter to keep standard protocol implementation. Such a network is used to control devices as well as motor controller.

In order to integrate the multi-protocol converter to the conventional BAS, we defined TCP/IP protocol. Therefore, the multi-protocol converter easily transfers data acquired from LonWorks and the RS485 network to BAS through TCP/IP protocol so that it is a flexible architecture for building a distributed network which has a different field data protocol. Comparing it with a typical BAS system which is implemented with a centralized interface, it is implemented with standard TCP/IP protocol and a multi-protocol converter connects most of the device network. Therefore, this system meets interoperability requirements.

Another architecture using the multi-protocol converter is a Web-based system without the conventional BAS as shown in Fig. 1. In order to develop this system, we also do porting embedded Web server on multi-protocol. In this case, the system will be a three-tier client-server architecture. Recently, Web-based monitoring and control systems are widely used by Internet technologies for transparent access of control devices [6]–[8], in which the Web-server is implemented on a Workstation or PC server. However, we designed an embedded Web-server based on embedded Linux. Furthermore, it encompasses multi-protocols which...
are used in industrial and building automation fields. In order to implement a Web-based control system, a Java applet instead of Common Gateway Interface (CGI) is used, in which Java socket programming and Graphic User Interface (GUI) for users are implemented.

Figure 1 shows that the multi-protocol converter is able to interconnect BAS and can be used for Web-based control system without the conventional BAS. In order to show the feasibility of these architectures, we modeled them by using UML [9] and performed experiments with them. Using component diagrams about BAS architectures is to show the flexibility, expandability, and interoperability in building network systems with multiple protocols. The experimental results show that the proposed multi-protocol converter using embedded Linux is a flexible and effective way to build a Web-based monitoring and control system.

2. Development of a Multi-Protocol Converter

2.1 Multi-Protocol Converter

For useful management for a control network, a multi-protocol converter is implemented. Table 1 shows specifications for it. It consists of two CPUs. One is S3C4530A based on ARM7TDMI core that is used for embedded Linux and controlling peripherals [10], [11]. The other is a Neuron chip supporting LonWorks protocol [12]. The first part of ARM-based SoC is designed to interface TCP/IP, RS485, RS232C, and LonWorks protocols. The LonWorks suggested by Echelon is a protocol supporting OSI 7-layer. The LonWorks is proper to control and monitor network devices just like LAN in the building network system. Figure 2 shows the device layout and components of the developed multi-protocol converter.

In Fig. 2, the multi-protocol converter consists of two CPUs, Neuron Chip and S3C4530. The hardware interface is required in order to transfer data between two CPUs. It can be implemented with parallel communication of data bits or shared memory of DPRAM. We adapt DPRAM of CY7C136 to transfer data in which the control data is collected and converted into TCP/IP for BAS or the Web-server. RS485 communication is also implemented with UART embedded in S3C4530A. And we implemented Ethernet controller with LAN controller embedded in S3C4530A. The communication mechanism is examined in Sect. 4.

2.2 Embedded Linux Porting

The Linux kernel was originally intended to support architectures 80386 found in the PC which includes Memory Management Units (MMU). Recently porting efforts to support various architectures is done so that it had to be modified to adjust to the other architecture including non-MMU architectures. The S3C4530 using ARM7TDMI is a non-MMU SoC so that we cannot use the Linux kernel which is oriented to MMU CPUs. uClinux is a derivative of Linux kernel intended for non-MMU CPUs. In uClinux, multitasking is tricky and source code for the kernel have been rewritten to tighten-up and slim-down the code base. This all means that the uClinux kernel is much smaller than the original Linux kernel, while retaining the main advantages of the linux operating system. And uClinux maintains a 1-to-1 virtual-to-physical mapping [13].

In this paper we choose uClinux 2.4 which is based on Linux kernel 2.4 since we use S3C4530, a non-MMU CPU. The porting is starting with modifying the hardware dependent portion of uClinux kernel so that it supports hardware control. This work is done in startup code as shown in Fig. 3. It is usually implemented with assembly language for doing the followings:

- defining the entry point
- setting up system configuration & PLL
- initializing the serial ports

Table 1

<table>
<thead>
<tr>
<th>CPUs</th>
<th>S3C4530A, CY7C53150</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td></td>
</tr>
<tr>
<td>FLASH</td>
<td>8MB + 32KB</td>
</tr>
<tr>
<td>EEPROM</td>
<td>128KB</td>
</tr>
<tr>
<td>SDRAM</td>
<td>16MB + 32KB</td>
</tr>
<tr>
<td>DPRAM</td>
<td>2KB</td>
</tr>
<tr>
<td>Peripherals</td>
<td></td>
</tr>
<tr>
<td>2 Serial Ports</td>
<td></td>
</tr>
<tr>
<td>2 RS485 Channels</td>
<td></td>
</tr>
<tr>
<td>10/100Mbps Ethernet</td>
<td></td>
</tr>
<tr>
<td>LonWorks Transceiver</td>
<td></td>
</tr>
</tbody>
</table>
Fig. 3 Booting sequence of embedded Linux.

- initializing the memory map
- copying ROM to RAM and setting up initial stack
- setting up the architecture type
- jumping to Linux (start_

The high-level porting is related with Linux kernel and implemented with C language. The entry point of C language is start_kernel called by startup code. The main functions are depicted in Fig. 3 and we modified hardware dependent portion of functions. The following is a main function:

- setting up the processor and the machine information
- parsing boot command-line options
- initializing traps (exception vectors)
- initializing the interrupt
- initializing the software interrupt
- initializing console
- enabling interrupt

After initializing the high level functions of kernel, next is loading the device drivers. This work is done in another process called init. The device drivers are called in do_initcalls so we should implement device drivers for example, character device, block device and network drivers. And finally, the function of deciding what/where to mount and load ramdisks, etc is done in prepare_namespce. Once we have ported embedded Linux, we are able to use Linux functions and utilities such as shell commands, and network protocols so that it is a useful platform for embedded systems.

3. Local Network Devices

3.1 LonWorks Control Devices

LonWorks is a communication protocol which is adopted as a standard protocol in building automation systems and a control network using Neuron Chip. Neuron Chip is a VLSI integrated with a microprocessor from MAC layer of this LonTalk protocols until application layer. In this paper, we implement node devices of LonWorks using Neuron chip as a distributed control network. The node controller is developed with LonWorks hardware and interface circuit in order to implement various control objects. In Fig. 4, there is interface circuit handling peripherals with memory-mapped I/O or general purpose I/O. Therefore, the control nodes of Fig. 5 are implemented by using a LonWorks control module while adding interface logics. In LonWorks, lots of transceivers are supported such as twisted pair, power line, and RF, etc. The control node is implemented with twisted pair of FTT-10. The communication rate is 76.8 kbps.

Figure 5 shows a developed control network of LonWorks, in which the control nodes are air conditioner control (AIRC), pump control (PUMP), temperature sensing (TEMP), luminous intensity sensor (LUX), and display nodes such as 7-segment (SEG) and LEDs (LED). These are typical examples of control device construction which are connected in bus topology so that it can be extended.

3.2 Software Structure of LonWorks Network

The program of LonWorks is done with Neuron C based on an event-driven program. The data types of the network and I/O port are available [12]. In order to communicate with LonWorks devices in the network, the network variable should be connected to each other and network binding does this. Binding is defined as a logical connecting variable placed at distributed control nodes so that the communication between nodes is accomplished. In this paper, we use network variables, and both variables declared at input and output nodes should be of the same data type.

The binding example is described in Fig. 6. The output network variables in the multi-protocol converter are defined for segment data and air-conditioner control. The output variable of vairc is a command to control air conditioner such as power on/off, target temperature, and the direction of wind blind. The other output variable of vseg is used to send segment data to SEG node. The input network variables in multi-protocol converter are vpump, vtemp, vlux, and vled to receive sensing or status data of nodes. The nodes named
as TEMP and LUX are designed to acquire the temperature and illumination with A/D converter.

3.3 Control Network Based on RS485

The RS485 communication is widely adopted in industrial fields to implement multi-drop communication. In this experiment, we use modbus protocol to communicate motor controllers which are connected to a RS485 multi-drop network. Modbus protocol is a message type protocol independent of a connected network and it is a de facto standard in industrial fields. The message is used to query or respond to other controllers and defined in a uniform manner. In modbus protocol, the master-slave method is adopted so that one master control is able to initialize the network and then other controllers, i.e. the slave is to respond to requests of the master or act according to the master’s command. Therefore, the multi-protocol converter will be a master and motor controllers become slaves as shown in Fig. 1.

4. Mechanism of Data Transmission

4.1 Socket Communication

The communication between BAS and the multi-protocol converter is done through TCP/IP socket so that the command from BAS is transferred to control devices or the data gathered from control devices are transferred to BAS. Therefore, the program named Server in multi-protocol converter is designed to gather data received from each protocol and convert it to socket data to utilize TCP/IP.

The Server program includes functions of RS485_driver and Lon_driver. The RS485_driver module has a role of open function for the RS485 device driver implemented in embedded Linux and functions needed to handle modbus protocol to handle motor control function. Lon_driver is used to communicate LonWorks device; however, it is not directly handling LonWorks devices but storing data or getting data to/from DPRAM. In Server function module, the decoding of data received from the client through TCP/IP is achieved and the decoded data is sent to the appropriate node.

We examine a TCP/IP packet on how to transfer command to control network of LonWorks devices. The packet is defined as Fig. 7. In a receive packet, the data field of addr are used to describe the protocol type of control. The cmd is used for command and the additional command information is included in datal and datah. In the case of a sending packet, pump, temp, lux, and led have the status for each device.

4.2 Data Transmission of DPRAM

The data transfer between two CPUs is performed in DPRAM with an interrupt signal as shown in Fig. 8. In order to trigger an interrupt signal, intention signal is written at 0x7f2 after writing data such as cmd, addr, datal, and datah onto DPRAM. Then CY7C53150 chip read DPRAM according to interrupt signal. The received data is sent to the appropriate node through LonWorks protocol.

The communication from LonWorks to Server program is achieved by a similar method. The Neuron chip in multi-protocol converter collects data from LonWorks network and stores them periodically onto DPRAM. Then Server program reads data from DPRAM.

5. Integration of a Conventional BAS with Control Devices

5.1 System Integration

We carried out an experiment to prove the possibility and excellence of a multi-protocol controller by applying it to a conventional BAS. The experiment environment is shown in Fig. 9. Two types of networks are there, one is motor...
controller with RS485 network and the other is LonWorks network. In the conventional BAS, Man-Machine Interface (MMI) server and DB server are required for the user interface and management of control devices. The multi-protocol converter is used to interface network devices and the conventional BAS in a unified way. In order to integrate the BAS with various control devices, RS485 device driver, LonWorks program, and socket program are performed in a multi-protocol converter.

The data communication is only accomplished through TCP/IP just like a gateway since the multi-protocol converter directly handles low-level devices for various protocols. Therefore, this yields an open system and has a feature for flexible system configuration independent of specific protocols.

The software of the conventional BAS is described in Fig. 9, where there are DB server, MMI server and applications for example socket client programs. The socket client program is required to communicate with the Server program of the multi-protocol control. In the case of the human interface, the application software is registered in the MMI server so that one can easily monitor the status of control node and control them with graphical objects. The MMI is depicted in Fig. 10.

5.2 Comparison of Architectures

The conventional BAS is needed for various types of protocol converters to interface various kinds of protocols as shown in Fig. 11. This requires the high-cost facilities to expand the system. In order to cope with this problem, we propose the simple structure using a multi-protocol converter through the standard TCP/IP. As shown in Fig. 11, the conventional BAS is centralized and requires additional devices to integrate various protocols such as Direct Digital Controller (DDC), converter, and PCLTA LonWorks adapter. Furthermore, software for handling protocols should be implemented in a central control server. This causes a problem in the sense of standardization and expandibility. Furthermore, the central control server requires high computing power and high-cost interfacing devices such as DDC, converter and LonWorks adapter.

To the contrary, the system described in Fig. 12 has standard communication protocol. The BAS can be expanded without considering the control devices. And the multi-protocol converter is using an embedded Linux, which is appropriate for building any kind of protocol. If we use the useful features of embedded linux, we can easily handle network. Furthermore, we are able to implement Web-based BAS without central control station by using embedded Web sever on embedded Linux. The experimental result of Web-
based BAS will be described in the next section.

6. Development of a Web-Based BAS

6.1 Configuration of a Web-Based BAS

The modeled architectures of Figs. 11 and 12 are suitable for a large-scale BAS. However, in a small building, a small BAS is required to control each devices. In order to cope with this requirement, we designed a Web-based BAS by using Java applet and embedded Web-server on embedded Linux [14], [15].

To our knowledge, little architectural work has been done to show the suitability of embedded Linux for building BAS. The component diagram of Fig. 13 shows the flexibility of a Web-based BAS in terms of a multi-protocol converter. In the multi-protocol converter, Boa Web Server to be used for Web service and Network Server for socket communication between the client and Lon Device Server for LonWorks communication are performed. A Java applet is initially placed at the multi-protocol converter, which is loaded to the client after connecting Boa Web server from the Web browser that is running in the client. The objective of the Java applet is to build a GUI and perform socket communication between client and Network Server running in a multi-protocol converter.

This is applicable to a small building and home and if it has a display such as TFT LCD it would be a useful home automation terminal.

6.2 Java Applet

For a Web-based system, the script language such as Active Server Page, the PHP and the Java Script Program might be used. However, the server script language is heavy for use in embedded systems so that CGI can be used. CGI is based on client pop and server push mechanism and all kinds of program languages are available for CGI. The problem of CGI is in the running environment. The application program for CGI is running in embedded systems so that it is not appropriate in the system having limited resources.

Therefore, we adapt a Java applet. A Java applet does not require script language and the running environment is in the client rather than in server so that it does not have resource limitation. The applet is downloaded from server to a client and runs in the client so that there is no load in server side that is embedded systems. Figure 14 shows running mechanism of Java applet. HTML document containing the applet is required in embedded systems. Once the browser opens HTML document from the client, the Java applet is uploaded to the client and it will be running in the Client with JVM.

As mentioned in the literature [15], the applet has a life cycle. Figure 15 shows the structure of the designed applet program. After loading to the client, init() function is invoked, where the initialization of variables is performed. And then start() function is executed to make GUI as shown in Fig. 16, connecting the Java socket to Network Server program as shown in Fig. 13 so that the communication is defined. Once the connection is performed, the user is able to control network devices by using GUI. When one clicks objects, mouseclicked() event occurs and invokes mouseClickedEvent() to send the command to multi-protocol converter through Java socket with TCP/IP. All of the graphic objects are defined with event handlers. Therefore, we can easily handle low-level network devices without any application programs and the conventional BAS. The destroy() function is performed when program terminates which sends termination message to Network Server and close client socket.

![Fig. 13](image1.jpg) Component diagram of a proposed Web-based BAS.

![Fig. 14](image2.jpg) Running mechanism of a Java applet.

![Fig. 15](image3.jpg) Structure of the designed Java applet program.
7. Conclusions

In this study we designed a useful Web-based BAS. A multi-protocol converter and the network devices are implemented. In the multi-protocol converter, S3C4530A is used as a main CPU and embedded linux and embedded Web server are also implemented on this to build a useful platform for a Web-based BAS. In order to integrate various protocols, network interface circuits, Neuron Chip, RS485 and Ethernet, are also implemented. The data communication between S3C4530 and Neuron Chip is performed by DPRAM. The network devices such as motor controllers and control nodes based on LonWorks devices are also implemented.

Two types of BAS systems are experimented to verify the suitability of the multi-protocol converter and they are compared with each other in terms of architecture modeling based on UML. For the Web-based BAS, a Java applet is designed in which the Java socket program and GUI for users are performed. The software programs required to develop two types of architectures are also discussed.

In this work we have showed that the method is a viable solution to construct, modify, and expand the BAS through the Internet while the centralized conventional BAS is not used. The three-tier architecture of the BAS is useful and expandable in terms of multi-protocol converter and embedded Linux. The work done here can be a useful guide to develop system-level design for multi-protocol environments and it can be successfully applied to designing embedded systems which require a unified and flexible way in the design process.

References

Soo Yeong Yi received the M.S. and Ph.D. degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology in 1990 and 1994, respectively. During 1995–1999, he stayed in Human Robot Research Center in Korea Institute Science and Technology as a senior researcher. He also was a Post Doctorial Researcher in Department of Computer Science, University of Southern California, Los Angeles in 1997. He is now with Division of Electronics and Information Engineering in Chonbuk National University, Korea. His primary research interest is in the area of mobile robot, walking robot, and intelligent control theory.

E-mail: suylee@moak.chonbuk.ac.kr