임베디드 리눅스를 이용한 산업용 인버터의 웹 기반 원격 관리

Web-Based Remote Management of Industrial Inverter using Embedded Linux

조 덕 연, 최 병 욱
(Duk-Yun Cho and Byoung-Wook Choi)

Abstract : Driven by the growth of the Internet and the increasing ubiquity of embedded computing systems, the embedded system is exploding in terms of a proliferation of products and the number of complex applications. Intelligent dedicated systems and appliances used in interface, monitoring, communications, and control applications increasingly demand the services of a sophisticated, state-of-the-art operating system. In the case of industrial controller, it is required to use a real-time operating system (RTOS) as a good building block to enable Internet connectivity. However, commercial RTOSes are increasingly less desirable due to their lack of standardization and their inability to keep pace with the rapid evolution of technology. In order to overcome these problems, we consider using embedded Linux and embedded web server. Availability of source code, reduced licensing, reliability, open source community support, as well as others, are key reasons for the use of embedded linux by embedded developers. In this paper, we develop embedded linux platform to control the industrial inverter with the Internet connecting feature. The method of web-based management is also proposed by using the embedded web server and Java applet. We show the feasibility of remote management for the commercial inverter controller with the proposed three-tier web-based remote management system.

Keywords : client pull, server push, Java applet, Modbus protocol, embedded Linux, and embedded web server

I. 서론

1980대 초 TCP/IP 발전과 더불어 인터넷 기술은 오늘날까지 급속히 발전하여, 여러 분야에 걸쳐 이용되고 있다. 특히 정보통신 분야에서 두드러지게 발전하고 있으며, 정보가전 분야에도 폭 넓게 이용되고 있다. 최근에는 산업 분야에서도 인터넷을 적용하는 사례가 늘고 있으며, 연구되고 있다[1-3, 6-7]. 이러한 인터넷의 발전은 특히 1990년대 월드 와이드 웹 (WWW)의 출현으로 급격히 이루어졌다.

기존의 산업용 장비들은 웹 서비스를 하기 위해서 주로 실시간 운영체제(RTOS)를 이용하거나 직접 TCP/IP 프로토콜을 작성하였다. 이러한 방법들은 몇 가지 단점을 갖고 있는데, 전자의 경우는 매우 비싼 상용 제품들이라는 점과 운영체제를 자유롭게 사용할 수 없다는 것이며, 후자의 경우는 작성한 프로그램의 신뢰성이 보장되지 않고, 개발 기간이 매우 길어진다는 점이다. 이로 인해 개발비가 저렴하며, 신뢰성이 높고, 네트워크 기능을 갖는 운영체제를 모색하게 된다. 그래서 개인용 컴퓨터(PC)의 운영체제인 리눅스를 이러한 특수한 목적에 맞추어진 시스템에 적용하기 시작하여, 2000년 5월에는 세계 50여개 산업계의 주요 업체들이 내장형 리눅스 컨소시엄을 만들어 현재까지 활동하고 있다[8]. 산업용 모터를 제어하는 인버터와 같은 장비들은 지역적으로 통신이 이루어져 원격지에서는 그것에 접근하기 힘들었고, 인터넷을 통해 접근하기 위해서는 별도의 접속 프로그램 등이 필요했다.

본 논문에서는 내장형 리눅스를 운영체제로 하고 웹 서비스를 위해 내장형 웹 서버를 사용하는 웹 기반 관리 시스템을 구현하고, 산업용 인버터를 원격지에서 웹 브라우저를 통해 모니터링하고 제어하는 방법을 제안하였다.

논문접수 : 2002. 10. 25., 채택확정 : 2003. 2. 28.
조덕연 : ㈜임베디드웹 (duke@embeddedweb.co.kr)
최병욱 : 선문대학교 기계제어공학부 (bwchoi@sunmoon.ac.kr)
※ 본 논문은 과학기술부/한국과학재단지정 선문대학교 공조기술 연구센터에서 지원하였습니다.

II. 웹 기반 관리

웹 기반 관리란 클라이언트의 웹 브라우저를 이용해 인터넷에 연결되어 있는 원격지의 서버를 통해 산업용 기기를 관리한다는 의미로 본 논문에서는 사용한다. 본 장에서는 이러한 것이 이루어질 수 있도록 하기 위한 방법들을 알아보고, 산업용 인버터에 적용할 방법을 제안한다.

1. 클라이언트 당기기

클라이언트 당기기(client pull) 메커니즘은 서버가 클라이언트에게 데이터를 보낼 때, "5초가 지나면 다시 읽어라" 또는 "8초 후에 다른 URL로 가라"라는 지시를 함께 보낸다[4]. 이렇게 하면 정해놓은 시간이 지나면, 클라이언트는 지시 받은 것을 수행한다. 즉, 원하는 시간마다 클라이언트가 서버에게 요청하여 새롭게 데이터들을 갱신할 수 있게 된다. 클라이언트 당기기의 사용은 다음과 같이 HTML 문서의 HEAD 부분에 META 태그를 삽입한다.

<META HTTP-EQUIV="Refresh" CONTENT="3; URL=http://www.embeddedweb.co.kr">

위의 예는 3초 후에, www.embeddedweb.co.kr 주소의 홈페이지로 이동하라는 의미를 갖는다. 즉, CONTENT는 다시 읽을 시간을 나타내는 것으로 초단위로 설정할 수 있다.

단, 0으로 값이 되면, 클라이언트 측 웹 브라우저에 모든 내용들이 로드 되는 순간에 다시 읽도록 요청하는 의미를 갖는다.

클라이언트 당기기는 간단하게 동적인 환경을 제공할 수 있지만, 클라이언트와의 연결은 일반적인 HTML 문서와 같이 끊어진다. 즉 수동적으로 클라이언트의 웹 브라우저 상에서 새로운 HTTP 연결을 요청하는 것과 같은 것이다. 이것은 클라이언트의 요청이 있을 때마다 서버에게 새로운 CGI 프로세스의 생성과 종료를 유발시켜, 서버 시스템에 부하를 일으키며 자원을 낭비하게 된다.

2. 서버 밀기

의의 클라이언트 당기기 메커니즘을 보완 할 수 있는 것으로 서버 밀기(Server Push)가 있다. 이것은 클라이언트의 요청이 없이도 서버가 능동적으로 클라이언트에게 데이터를 전송하는 방법을 제공한다. 특히 서버 밀기의 장점은 클라이언트의 요청에 대해 HTTP 연결을 유지하고 있다는 것이다. 그러면 서버는 원하는 시간에 데이터를 클라이언트에게 보낼 수 있게 되며, 새로운 CGI 프로세스를 생성시키지 않아 서버의 자원을 낭비하지 않는다. 여기에 두 가지의 장점을 덧붙이면, 하나는 쉽게 브라우저상의 "stop" 버튼을 서버와 클라이언트 간의 통해 연결을 끊을 수 있다는 것이고, 다른 하나는 서버 쪽에서 바뀌는 데이터 부분만을 클라이언트 쪽으로 새롭게 갱신할 수 있다는 것이다. 그렇게 되면 모든 자료를 전송할 필요가 없게 되므로 네트워크에 부하가 적어 빠른 응답을 할 수 가 있게 된다. 일반적인 HTTP 응답은 하나의 메시지에 하나의 슬라이스 데이터를 사용하지만, 서버 밀기는 MIME 메시지의 "multipart/mixed"를 통하여 하나의 메시지에 여러 슬라이스 데이터를 전송할 수 있는 방법을 이용한다. 이것은 다음과 같이 사용할 수 있다.

Content-type: multipart/mixed:boundary=ThisRandomString

-- ThisRandomString
Content-type: text/plain

Data for the first object.

-- ThisRandomString
Content-type: text/plain

Data for the second and last object.

-- This Random String--

위의 메시지는 두 개의 슬라이스 데이터를 갖고 있는데, 모두 "text/plain" 형태이다. 마지막 "ThisRandom String" 문자열의 끝에 두 개의 데쉬 "--"는 이 메시지가 끝이나 더 이상의 슬라이스 데이터가 없음을 나타낸다. 이처럼 하나의 메시지에 여러 슬라이스의 데이터를 넣기 위해 boundary에 "ThisRandomString" 이라는 매개변수를 이용하여 데이터 슬라이스를 나눈 것이다. 슬라이스의 시작은 두 개의 데쉬 "--" 다음에 boundary 매개변수를 쓰면 되고, 이 슬라이스의 끝은 다음 블록의 시작이 된다.

서버 밀기는 "multipart/mixed"의 확장인 "multipart/x-mixed-replace"를 사용할 수 있는데, "mixed"의 앞에 "x-"라고 붙은 것은 아직 실험적이(experimental)라는 의미이고, 끝에 "replace"라는 것은 슬라이스 데이터에서 새롭게 바뀐 부분만을 이전 슬라이스 데이터에 재배치하도록 한다. 그러나 서버 밀기에도 단점이 있다. 서버 쪽의 CGI 실행으로 부하가 걸리고, Netscape 브라우저 같은 특정 웹 브라우저에서만 이용 가능하기 때문에 약간의 제약이 있다.

이들 클라이언트 당기기와 서버 밀기는 CGI를 이용하여 동적인 웹 서비스를 제공할 수 있을 뿐만 아니라 메커니즘 또한 어렵지 않아서 많이 사용되어 왔다. 그러나 클라이언트 당기기와 마찬가지로 CGI의 실행이라는 특징으로 서버에 많은 부하를 가져오며, 프로그래밍 관점에서 CGI 코드 속에 HTML 문서를 넣는 방식은 디자인과 개발을 별도의 작업으로 분리시키지 못하기 때문에 유지, 보수에 많은 시간을 낭비를 초래하는 단점을 갖고 있다. 이를 개선하는 방법들로 HTML 문서에 프로그램 코드를 넣는 방식으로 server-side script이라 하며, ASP, JSP 등과 같은 것들이 있어 사용의 편의성을 제공하며, 여러 데이터 베이스들과 쉽게 연동할 수 있는 함수들과 웹 개발에 필요한 많은 자원들을 제공한다. 하지만 이들은 웹 서버에서 동작하는 것들로써 서버에 많은 부하를 가져오며, 기존 프로그래밍 언어와의 호환성이 좋지 않아 기존의 많은 라이브러리들을 재사용할 수 없게 된다. 따라서 이러한 server-side script는 리소스가 제한되어 있는 내장형 시스템에 적용하기가 힘들다.

3. 자바 애플릿

본 논문에서는 자바 애플릿(Java Applet)을 이용한 웹 기반 관리를 제안한다. Sun Microsystems에서 만든 자바 언어(Java Language)는 객체지향 언어로서 내장형 시스템을 위해 플랫폼에 구애 받지 않는 언어로 처음 개발되었다. 자바가 플랫폼에 구애 받지 않기 위해서는, 동작할 플랫폼에 자바 가상 머신(JVM, Java Virtual Machine)이 내장되어 있어야 한다. 하지만 이 자바 가상 머신을 내장형 시스템 보드에 올린다는 것은 리소스를 많이 필요로한다. 따라서 본 논문에서는 자바 애플릿을 이용한다. 이것은 서버 측에서 동작하는 것이 아니라 클라이언트에게 보내져서 클라이언트 환경에서 수행되기 때문에, 클라이언트 상에만 자바 가상 머신이 있으면 된다. 그러나 이 방법의 단점은 서버로부터 자바 애플릿 프로그램을 다운로드 받는데 시간이 필요하다는 것이다. 이러한 시간이 필요한 이유는 다운로드 받는 애플릿 프로그램이 자바 컴파일러로 컴파일된 자바 바이트 코드(byte code)로 HTML 문서에 비해 크기가 크기 때문이다. 그러나 최근 초고속 통신의 발달로 이러한 다운로드 시간은 크게 영향을 주지는 않는다. 그림 1에 CGI와 자바 애플릿의 동작 메커니즘에 대하여 간략히 도식화하였다. 좌측 client가 웹 서버에게 자료를 요청하면, 웹 데몬은 서버에서 CGI를 통해 명령을 처리한 후, 결과를 client에게 응답해준다. 반면에 우측 client는 웹 서버에게 자료를 요청하면, 웹 데몬은 자료를 처리할 수 있는 자바 애플릿을 client에게 전달해준다. 그러면 client는 자신이 갖고 있는 자바 가상 머신을 통해 전달 받은 자바 애플릿을 수행하여 결과를 얻는다.

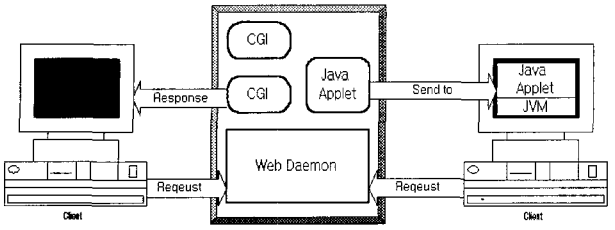

그림 1. CGI와 자바 애플릿의 동작 메커니즘.
Fig. 1. Mechanisms of CGI and Java Applet.

III. 웹 기반 관리 시스템 구현

본 장에서는 산업용 인버터를 관리하기 위한 시스템에 대해 살펴본다. 이 시스템 보드 구성과 보드를 운영하기 위한 내장형 리눅스, 그리고 웹 서비스를 하기 위한 내장형 웹 서버에 대해 알아보겠다.

1. 관리 시스템 보드 구성

관리 시스템 보드는 웹 기반 관리 시스템을 위한 것이기 때문에, 이더넷을 지원해야 한다. 그리고 산업용 인버터와의 통신을 위해서 RS-485 포트가 필요하다. 이러한 것들을 모두 충족하기 위한 시스템을 구현하기 위해 ARM7TDMI 코어를 내장한 삼성 S3C4530A를 CPU로 사용하였으며, 주변 장치로는 1개의 RS-232 포트, 1개의 이더넷 포트, 그리고 디버깅을 위한 JTAG 포트들이 있다. 그리고 2개의 RS-485 포트를 위해 UART 컨트롤러 (PC16652D)를 사용하였고, 보드의 상태 표시를 위한 LED와 2x16 문자 LCD 등을 함께 구성하였다. 시스템 메모리는 내장형 리눅스의 동작을 위해 SDRAM을 16MB, 커널과 루트파일시스템을 저장할 FLASH의 용량을 8MB로 하였으며, 그리고 시스템 부트로더를 위해 EEPROM 128KB를 사용하였다. 이렇게 구성된 시스템의 블록 다이어그램은 그림 2에 나타내었다.

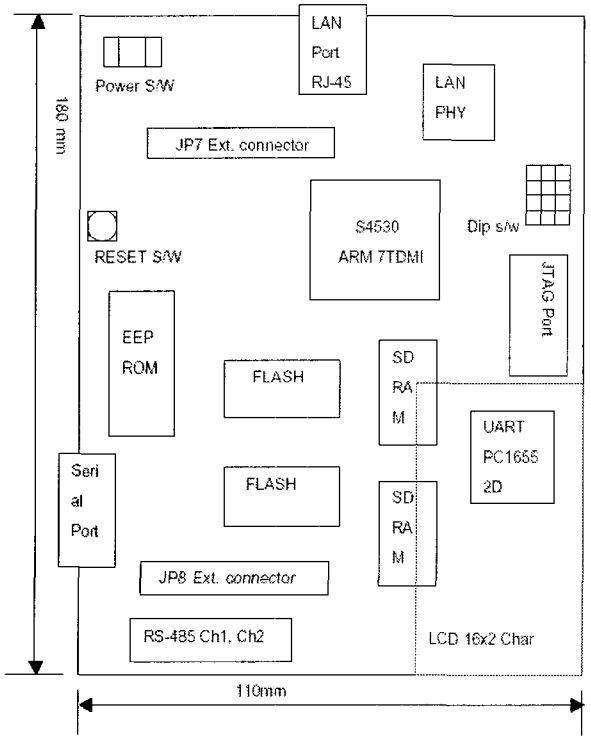

그림 2. 관리 시스템 보드의 블록 다이어그램.
Fig. 2. The Block Diagram of Management System Board.

2. 내장형 리눅스

리눅스는 기본적으로 32 비트 마이크로 프로세서인 i386에서 처음으로 개발되어 현재는 메모리 관리 유닛(MMU)이 있는 여러 마이크로프로세서 아키텍처에 포팅되어 사용되고 있다[9]. 그러나 본 연구에서 사용하는 S3C4530A는 메모리 관리 유닛이 없는 ARM7TDMI 코어 기반의 CPU이기 때문에 이러한 기존의 리눅스 커널을 사용할 수 없다. 따라서 본 논문에서는 이와 같이 메모리 관리 유닛이 없는 마이크로프로세서들을 위한 내장형 리눅스인 uClinux를 관리 시스템의 운영체제로 포팅하였다. 이 uClinux 커널은 일반 리눅스 커널과는 달리 가상 메모리 지원을 제거 하였으며, 멀티 테스킹을 위한 방법에도 차이가 있다[10].

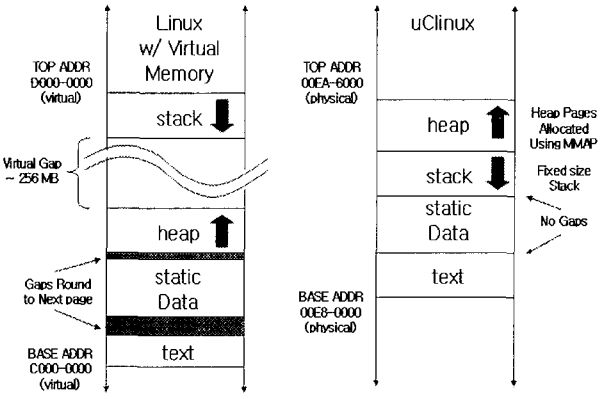

그림 3. 일반 리눅스와 uClinux의 메모리 맵.
Fig. 3. Memory Map for typical Linux and uClinux.

가상 메모리를 제거했다는 의미는 하드웨어적으로 이를 지원하지 못하기 때문에, 물리 메모리와 가상 메모리를 같은 주소로 사용한다는 것이다. 따라서 일반 리눅스 커널의 메모리 맵과는 다르게 uClinux 커널은 그림 3과 같이 heap과 stack의 위치가 바뀌어 있어, 고정된 stack을 사용한다. 멀티 테스킹을 위한 전형적인 리눅스의 fork 시스템 콜은 copy-on-write라는 기법을 통해 이루어졌다. 하지만 uClinux에서는 메모리 사용이 제한적이기 때문에 vfork 시스템 콜과 함께 쓰레드를 사용한다.

실제 관리 시스템 보드에 포팅된 uClinux 커널은 2.4.17 버전으로 삼성 S3C4530A에 내장된 이더넷 포트를 이용할 수 있도록 장치 드라이버를 구현하였으며, 내장된 시리얼 포트 역시 시스템 콘솔을 위해 장치 드라이버를 작성하였다. 그리고 산업용 인버터와의 통신을 위한 별도의 RS-485 장치 드라이버를 작성하였으며, 끝으로 2x16 문자 LCD 장치 드라이버를 구현하여 디스플레이 장치로 이용하였다.

본 논문에서 구현된 관리 시스템 보드는 산업용 인버터 장치와 RS-485 포트를 통해 통신하며, 외부 클라이언트 사용자와는 TCP/IP를 이용하여 통신이 이루어진다. 그래서

여기서는 이더넷 장치 드라이버와 RS-485 장치 드라이버에 대해 간단히 살펴본다.

리눅스 커널 부팅시 장치 드라이버를 설정하는 부분에서 이더넷 드라이버들은 ethif_probe() 함수를 통해 초기화가 이루어지는데, 여기서 samsungeth_probe()라는 초기화 함수를 호출한다. 이 함수는 메모리를 할당받아 장치를 위한 자료구조를 초기화하며, 실제 장치를 동작시키는 함수들을 자료구조에 연결시킨다. 그리고 이더넷 하드웨어 MAC 주소를 설정하여준다. 연결된 동작 함수들의 이름과 구현 내용을 정리하면 다음과 같다.

- *ss_net_open()*

이더넷 관련 인터럽트를 금지시키고, 물리계층과 이더넷 송수신 블록의 인터페이스인 MII(Media Independent Interface)를 초기화 해주며, 물리계층을 담당하는 PHY 칩을 설정한다. 이때, 외부에 연결되어 있는 네트워크의 특성을 검출하여 이에 따라 PHY 칩 설정을 해주도록 한다. 그리고 송수신을 위해 인터럽트 헨들러의 등록이 이루어지는데, 송신을 위해서는 MAC 인터럽트를 사용하며, 수신을 위해서는 DMA 인터럽트를 사용한다. 그리고 끝으로 MAC과 DMA를 사용할 수 있도록 자료구조를 초기화 해주며, 네트워크 하드웨어 주소를 설정하고, 네트워크 관련 인터럽트를 사용 가능하게 해준다.

- *ss_net_close()*

ss_net_open()을 통해 사용하던 네트워크 관련 인터럽트 자원을 반환함으로써, 이더넷 장치의 사용을 멈춘다.

- *ss_net_start_xmit()*

실제 하드웨어 송신 시작을 위한 함수로 OSI 상위 계층에서 만들어진 패킷을 소켓 버퍼(sk_buff)에서 프레임 버퍼로 가져온다. 그리고 이 패킷 송신을 위해 MAC 제어 레지스터를 설정한다. 끝으로 프레임 버퍼를 다음 패킷을 위해 다음으로 옮긴다.

- *ss_net_get_stat()*

네트워크 장치에 대한 통계 및 정보를 제공하는 함수로, ss_net_priv 구조체를 반환한다. 이 구조체에는 네트워크 장치의 통계를 위해 송수신한 패킷의 수나 패킷 전송시 에러와 같은 정보를 갖고 있는 net_device_stats 구조체를 포함하고 있다.

- *ss_set_multicast_list()*

이더넷 장치에 대한 멀티 캐스트 목록이나 플래그들이 바뀔 때 호출되는 함수이다. dev->flags를 검사하여 IFF_PROMISC 플래그를 통해 무차별 모드(promiscuous mode)에 들어갈 때를 판별하며, IFF_ALLMULTI 플래그를 통해 멀티 캐스트-라우팅의 허용 등을 판별하여 CAM 컨트롤 레지스터를 설정하여준다. 무차별 모드에서는 dev->mc_list와 상관 없이 모든 패킷을 수신하도록 하며, 멀티 캐스트-라우팅이 허용되면 모든 멀티 캐스트 패킷을 수신한다.

ss_net_open() 함수에서 설정한 네트워크 드라이버를 위한 인터럽트 헨들러들은 전송을 위해 MAX Tx 인터럽트를 사용하는 ss_mac_tx() 헨들러와 수신을 위해 DMA Rx 인터럽트를 사용하는 ss_bdma_rx() 헨들러가 있다.

이번에는 산업용 인버터와 통신을 위한 RS-485 장치 드라이버에 대한 설명을 하겠다. RS-485 장치는 직렬통신 장치로 본 논문에서는 S3C4530A 프로세서에 내장된 RS-232 장치인 "ttyS0"와 "ttyS1", 다음으로 PC16552D 컨트롤러를 "ttyS2"와 "ttyS3"로 정하였다. 그리고 주번호는 4로 RS-232 장치가 사용하는 TTY_MAJOR와 같이 하였으며, 부번호는 각각 66과 67로 할당하였다. 이와 같이 별도의 RS-485 장치를 RS-232 장치의 하위 장치로 구현하였다. 따라서 RS-485의 주요 함수들은 RS-232 장치 함수에서 호출 하도록 되어 있다. 즉 응용 프로그램에서 RS-485 장치로 접근하더라도 기본적으로 RS-232 장치 드라이버에서 부번호를 확인하여, RS-485 함수를 수행하도록 된다. 이렇게 하면 사용자나 개발자 모두 직관적으로 RS-232나 RS-485나 같은 드라이버로 인식하여 쉽게 접근할 수 있도록 하였다. 다음은 이렇게 구현된 RS-485의 주요 함수들이다.

- *PC16552D_rs_init()*

부팅시 직렬장치를 초기화 해주는 chr_dev_init() 함수에 의해 터미널과 관련된 장치들을 별도로 tty_init()함수에 의해 초기화 된다. 이때 RS-232 장치가 samsung_rs_init() 함수에 의해 초기화 되며, RS-485 장치를 함께 초기화하기 위해 PC16552D_rs_init() 함수가 실행된다. RS-232 장치 초기화 시에 터미널과 관련된 기본적인 자료구조는 설정되었기 때문에 여기서는 RS-485와 관련된 부분만 구현하였다. 먼저 인터럽트 헨들러를 등록하고, PC16552D 컨트롤러와 자료구조를 초기화 해주며, 끝으로 인터럽트를 사용 가능하게 한다.

- *PC16552D_rs_open()*

RS-232 장치의 samsung_rs_open() 함수에서 부번호가 66이나 67일때 호출되는 함수로, RS-485 장치의 사용을 표시하기 위해 카운트 값을 증가시키고, RS-485 장치를 위한 자료구조를 초기화 하고 터미널과 연결한다. 그리고 다시 PC16552D_startup_port()를 호출하여, 사용하려는 포트의 baudrate와 FIFO 레지스터를 설정한다. 이렇게 하면 장치를 사용할 준비가 완료된다.

RS-485 통신은 반이중(half-duplex) 방식으로 전이중(full-duplex)를 사용하는 RS-232 통신과는 다른 제어가 필요하다. 본 논문에서 구현한 RS-485 통신 장치들은 수신모드와 송신모드를 갖고 있으며, 이는 RTS 신호를 이용해 구별되도록 하였다. 기본적으로는 수신모드로 되어 있으며, 송신시에만 송신모드로 전한다. 송신모드는 RTS 신호를 활성화 시킴으로써 포트의 송신 제어권을 갖는다. 수신모드는 RTS 신호가 비 활성화 되어 있는 상태로 포트를 통해 송신은 불가능하며, 수신만을 할 수 있도록 하였다. 결국 응용프로그램에서 포트를 open 하면 모두 수신모드로 기본 설정된다.

- *PC16552D_rs_close()*

포트의 사용을 끝낼 때 호출되는 함수로, 포트 사용을 종료한다는 표시를 위해 카운트 값을 감소시킨다. 그리고 포트를 기본 초기값으로 설정한 후, 터미널과의 연결을 NULL로 해제시킨다. 물론 samsung_rs_write() 함수에 의해 호출된다.

- *PC16552D_rs_write()*

현재 구현된 RS-232나 RS-485 장치의 송신은 폴링을, 수신은 인터럽트를 이용한다. 응용프로그램의 write() 함수에 의해 호출되는 함수로 samsung_rs_write()를 통해 실행된다. 송신할 포트를 송신모드로 전환한 후, 실제 문자 전송 함수인 PC16552D_transmit_chars()를 통해 데이터를 전송한다. 전송을 완료한 후에는 반드시 포트를 수신모드로 전환해야 한다.

- *PC16552D_rs_interrupt_1()*
- *PC16552D_rs_interrupt_2()*

RS-485 장치의 인터럽트 핸들러로 ttyS2와 ttyS3를 통해 데이터가 입력되어 해당 인터럽트가 발생하였을 때 실행되는 함수들이다. 두 함수 모두 같은 기능을 하나 단지 포트의 차이 때문에 두 함수로 분리되었으며, 인터럽트 판별 레지스터를 확인하여 정상적인 인터럽트가 발생하면, PC16552D_receive_chars() 함수를 통해 PC16552D 컨트롤러의 버퍼에서 데이터를 읽어온다. 이때 통신 에러가 발생하였다면, 해당 플레그 변수들을 증가시킨다. 이 인터럽트 핸들러들은 RS-232와 상관없이 RS-485 장치 open시 등록한 인터럽트 핸들러이기 때문에 언제든지 해당 인터럽트가 발생하면 실행된다.

이렇게 구현된 이더넷 장치와 RS-485 장치의 장치 드라이버를 통해, 관리 시스템 보드를 TCP/IP 통신 및 모드버스 통신을 가능하게 하였다. 이번에는 이들 장치를 통해 실제 웹 서비스를 가능하게 하기 위한 웹 서버를 알아보겠다.

3. 내장형 웹 서버

웹 기반 관리 시스템을 구성하기 위해 가장 기본이 되는 것은 웹 서비스를 할 수 있는 웹 서버를 갖추는 것이다. 이 응용프로그램은 앞서 설명한 내장형 리눅스인 uClinux 위에서 동작 하게 될 것이다. Apache와 같이 대중적으로 많이 쓰이는 웹 서버는 많은 기능들을 포함하고 있지만, 그만큼의 메모리 자원을 필요로 한다[11]. 예를 들면 Apache는 클라이언트의 모든 요청에 대해 fork와 exec를 통해 서비스를 한다. 이때 메모리를 각각 할당 받아 사용하며, 프로세스들의 처리시에 문맥교환이 빈번히 발생하므로 속도가 느린 시스템에서는 많은 부하가 걸린다. 하지만 boa는 단일 테스크 서버로써, 단지 CGI 요청이 있을 때에만 fork를 실행하도록 되어있다[12]. 그래서 본 논문에서는 boa 웹 서버를 포팅하여 사용한다. 앞서 말했듯이 uClinux는 fork를 이용하지 않고 이를 vfork로 대신하여 멀티 테스킹를 구현한다. 따라서 boa 웹 서버 코드에서 이들을 수정하여 삼성 S3C4530A 프로세서에서 동작할 수 있도록 ARM7TDMI용으로 교차 컴파일하여 실행 가능한 형태의 파일을 생성하였다.

IV. 산업용 인버터에 적용

본 논문에서 산업용 인버터의 관리는 모터의 운동 명령 지령과 상태 모니터링을 의미한다. 운동 명령 지령에는 모터의 회전 방향, 속도를 제어하기 위한 입력 주파수가 있고, 상태 모니터링에는 출력 주파수, 부하 전류 및 전압, 그리고 오류 표시들이 있다. 본 장에서는 앞서 구현한 웹 기반 관리 시스템과 산업용 인버터간의 모드버스 프로토콜을 통해 연결하고, 원격지에 위치한 클라이언트의 웹 브라우저 상에서 산업용 인버터를 관리할 수 있도록 하는 사용자 인터페이스를 구현하여 적용한다.

1. 산업용 인버터와 연결

본 연구에서 사용되는 산업용 인버터는 RS-485 인터페이스를 통해 모드버스 프로토콜을 이용한다. 모드버스 프로토콜은 산업용 기기 공급자들이 사용하고 있는 사실상의 표준 프로토콜로써, 전송 바이트 포맷에 따라 ASCII 전송 모드와 RTU(Remote Terminal Unit) 전송 모드로 나뉘어 진다[5]. 여기서 적용한 산업용 인버터는 VVVF 형(Variable Voltage, Variable Frequency Type) 인버터로 삼상 200V 급이며 50~60Hz의 입력 주파수가 필요하다. 여기에 정격 출력 전류는 5.0A이고 정격 출력 주파수는 0.1~400Hz이다. 외부와의 통신 방법은 모드버스 RTU를 지원하기 때문에 본 연구에서는 RTU 전송 모드를 사용하여 웹 기반 관리 시스템과 연결한다. 그림 4는 모드버스 RTU 전송 모드의 메시지 프레임을 보여주고 있다.

Start Bits	Addr.	Func.	Data	Error Check	End Bits
T1-T2-T3-T4-	8 bits	8 bits	n x 8 bits	16 bits (CRC)	T1-T2-T3-T4

- Start : 3.5 이상의 입력 없음
- Address : 2 hexadecimal 문자(8 bits), 범위 0~247
- Function : 2 hexadecimal 문자(8 bits)
- Data : n개의 2 hexadecimal 문자(n x 8 bits), 범위 1~255
- Error Check : 2 bytes CRC, (16 bits)
- End : 3.5 문자 이상의 입력 없음

그림 4. 모드버스 RTU전송 모드의 메시지 프레임.
Fig. 4. Message Frame of Modbus on RTU Transfer Mode.

이렇게 연결된 웹 기반 관리 시스템과 산업용 인버터의 통신 모델은 그림 5와 같다.

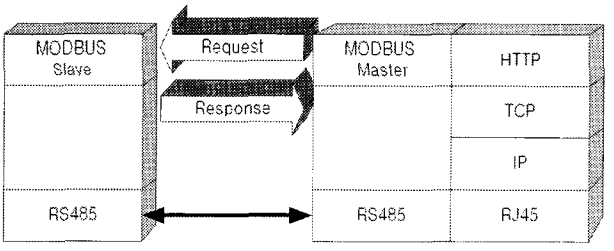

그림 5. 모드버스 통신 모델.
Fig. 5. Modbus Communication Model.

여기에서 RS-485 인터페이스는 멀티 드롭을 지원할 수 있다는 특징이 있으므로 여러 대의 인버터를 통해 다수의 모터를 제어할 수 있다.

2. 클라이언트 지원 사용자 인터페이스

원격지의 웹 브라우저에서 산업용 인버터를 접근하기 위한 사용자 인터페이스는 자바 애플릿을 이용하여 구현한다. 이 사용자 인터페이스에는 웹 기반 관리를 위한 모든 인터페이스 유닛들이 있어야 한다. 그림 6에 구현된 사용자 인터페이스가 나타나 있다.

사용자 인터페이스는 크게 명령 지령 부분과 상태 모니터링 두 부분으로 나뉘어 진다. 오른쪽의 "REMOTE CONTROL PAD" 부분은 모터의 운동 명령 지령을 내리는 부분이고, 왼쪽의 "MONITOR DISPLAY"부분은 인버터의 상태를 나타내주는 모니터링 부분이다. 운동 명령 지령부분에서는 다섯 가지 지령을 내릴 수 있는데, 모터의 회전 방향, 입력 주파수, 모니터링 스위치, 원격 제어 스위치 그리고 인버터 번호들이 그 예이다. 왼쪽의 모니터링 부분에는 출력 주파수 부분, 출력 전압 및 전류 부분 그리고 오류 표시 부분, 이렇게 세 부분으로 나뉘어져 인버터의 상태를 관찰 할 수 있도록 한다. 이때 출력 주파수, 출력 전압 및 전류 부분에는 시간의 흐름에 따른 변화량을 볼 수 있도록 시간 축 그래프가 지원되고 있다.

구현된 사용자 인터페이스는 클라이언트의 요청에 의해 서버 역할을 하는 웹 기반 관리 시스템의 boa 웹 서버로부터 자바 애플릿을 전송 받아 구성된다. 여기서 사용자 인터페이스가 동작하면 클라이언트와 서버 사이의 HTTP 접속은 종료되고, 새롭게 TCP/IP 소켓 접속이 이루어진다. 그러면 접속된 소켓을 통해 클라이언트의 명령 지령을 내릴 수 있고 서버측의 모니터링 결과들을 관찰 할 수 있게 된다. 그림 7의 클라이언트와 서버 연결 모델을 통하여 확인할 수 있다.

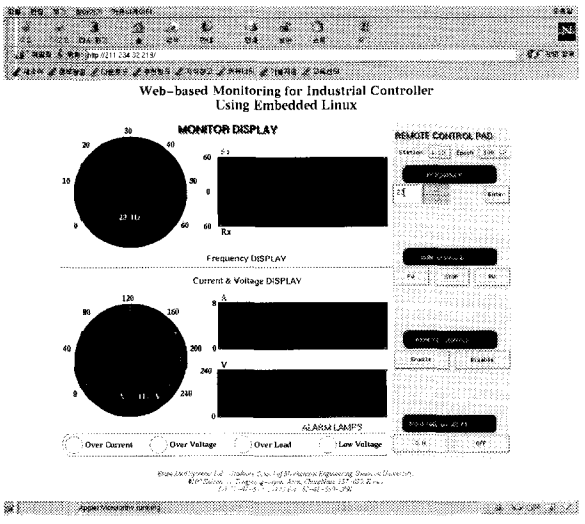

그림 6. 웹 기반 관리를 위한 사용자 인터페이스.

Fig. 6. User Interface for Web-based Managementon Client.

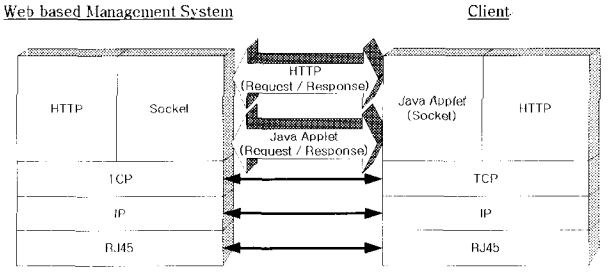

그림 7. 클라이언트와 서버 연결 모델.

Fig. 7. Client and Server Connection Model.

3. 산업용 인버터의 웹 기반 관리 시스템

구현된 웹 기반 관리 시스템의 전체 구성이 그림 8에 나타나 있다. 이것은 전형적인 three-tier architecture를 이루고 있다. 아래쪽의 산업용 인버터는 지령을 받아 처리되어지는 데이터 베이스 역할에 해당되며, 중간의 웹 기반 관리 시스템 보드는 클라이언트로부터의 요청에 응답해 주는 서버이며, TCP/IP 통신선을 넘어선 위쪽의 클라이언트는 사용자의 명령을 내려주고, 그 상태를 알려주는 클라이언

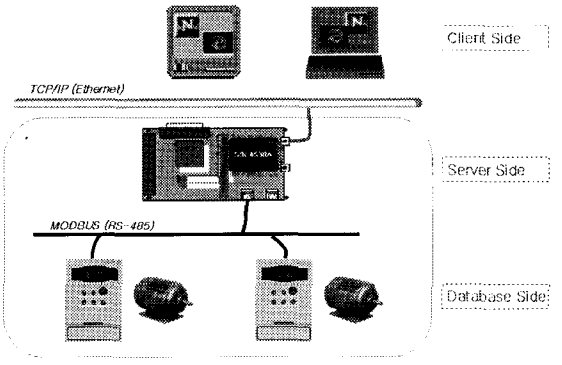

그림 8. 웹 기반 관리 시스템의 전체 구성도.

Fig. 8. Overview of the integrated Web-based Management System.

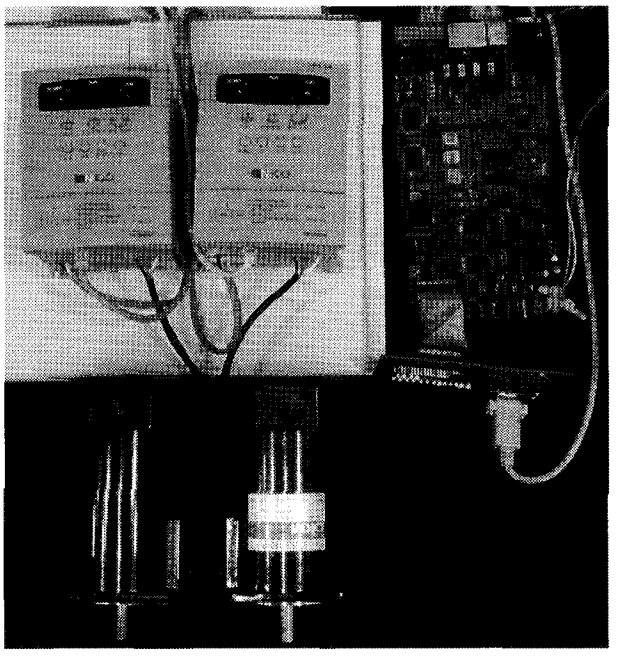

그림 9. 실험 중인 웹 기반 관리 시스템.

Fig. 9. The experimenting Web-based Management System.

트 인터페이스이다. 그림 9와 같이 구성된 전체 시스템을 원격지 웹 브라우저 상에서 지령을 내려 그 동작을 확인할 수 있었으며, 동작하는 상태를 계속해서 사용자 인터페이스를 통해 관찰 할 수 있었다.

V. 결론

본 논문에서는 원격지에 있는 산업용 인버터를 웹 브라우저를 통해 명령 지령을 내리고, 상태를 관찰할 수 있는 시스템을 구현하였다. 또한 웹 브라우저를 통한 관리 방법을 제안하여, 자바 애플릿을 이용한 사용자 인터페이스와 서버측의 통신 프로그램을 작성하였다. 이를 실제 실험을 통해 그 동작을 확인 할 수 있었다. 이는 앞으로 무인 공장이나 공장 자동화 등에 쓰일 수 있는 기반이 될 수 있다.

본 연구에서 제안된 방법을 통해 하나의 웹 기반 관리 시스템을 통해 여러 대의 인버터를 관리 할 수 있고, 자바 언어를 사용함으로써 클라이언트 플랫폼에 독립적으로 이용할 수 있다. 또한 TCP/IP 기반의 소켓 통신을 이용하여 차후 무선 인터넷 서비스에도 이용 가능하여 다양한 응용에 확장이 용이하다.

그러나 자바 언어를 사용하기 때문에 클라이언트에서 접속시에 약간의 시간 지연이 발생하며, TCP/IP의 특징으로 통신량에 따라 그 성능에 많은 영향을 끼친다. 또한 시스템에 사용되는 내장형 리눅스와 내장형 웹 서버의 보안문제가 안정화 되어 있지 않기 때문에 외부 침입에 대한 대처 방안이 개선되어야 할 것이다.

따라서 향후 연구에서는 웹 기반 관리 시스템을 내장한 산업용 인버터를 설계하고 구현하도록 하겠다. 나아가 UDP/IP를 통한 실시간 네트워크 모니터링 및 내장형 리눅스의 실시간 운영체제로의 모색이 함께 이루어질 것이다.

참고문헌

[1] R. Itschner, C. Pommerell, M. Rutishauser, "GLASS: Remote Monitroing of Embedded Systems in Power Engineering", *IEEE Internet Computing*, vol.2, no.3, pp. 46-52, May/June, 1998.

[2 L.Q.Kong, J.Malee and T.Korte, "A Simple Architecture for Real-Time Web Based Devices Control and Monitoring", ESCC 2001, no. 230, Chicago, 2001.

[3] C. D. Leidigh, "Web Based Management of Network Devices", ESCC 2001, no. 204, Chicago, 2001.

[4] B.W. Choi and D.Y. Cho, "Monitoring of Industrial Controller using Web Server on Embedded Linux Platform", ICCAS 2001, I-TE04-4, Jeju, Oct., 2001.

[5] MODICON Inc., *Modicon Modbus Protocol Reference Guide Rev.J*, PI-MBUS-300, pp. 1-72, June, 1996.

[6] 김인홍, 이정배, William Weinberg, "웹기반 내장형 응용", 정보처리, 제 5권, 제 4호, pp. 75-81, July, 1998.

[7] 최성종, "임베디드 시스템으로의 이식을 위한 TCP/IP 소프트웨어 분석", 정보기술연구소 논문집, 제 2집, pp. 46-56, July, 2000.

[8] Embedded Linux Consortium, http://www.embedded-linux.org

[9] The ARM Linux Project, http://www.arm.linux.org.uk

[10] Embedded Linux/Microcontroller Project, http://www.uclinux.org

[11] The Apache Software Foundation, http://www.apache.org

[12] Boa Webserver, http://www.boa.org

조 덕 연

1976년 4월 29일생. 2000년 선문대학교 전자공학과 졸업(공학사). 2002년 선문대학교 대학원 기계공학과 졸업(공학석사). 2001년~현재 ㈜임베디드웹 연구원. 관심분야는 실시간 운영체계, 내장형 시스템, 신경회로망.

최 병 욱

1963년 2월 13일생. 1986년 한국항공대학교 항공전자공학과 졸업(공학사). 1988년 한국과학기술원 전기 및 전자공학과 졸업(공학석사). 1992년 동대학 대학원 박사과정 졸업 (공학박사). 1992년-2000년 LG산전 연구소 엘리베이터 연구실장, 임베디드 시스템 연구팀장. 2000년~현재 선문대학교 기계 및 제어공학부 교수. 2001년~현재 ㈜임베디드웹 대표이사. 관심분야는 소프트웨어 엔지니어링, 내장형 시스템, 실시간 운영체계, 임베디드 리눅스, 필드버스 및 분산 제어 시스템.