2003년도 대한전기학회 하계학술대회 논문집 2003.7.21-23

내장형 리눅스를 이용한 웹 기반 원격 제어 시스템 구현

이석원, 이진우
우송대학교 컴퓨터전자정보공학부

Implementation of Web-Based Remote Control System Using Embedded Linux

Seok-Won Lee, Jin-Woo Lee
School of Computer, Electronic and Communication Engineering, Woosong University

Abstract - In this study, we explain the process and the technique required for implementing web-based remote control system based on embedded processor, SA1110, and embedded Linux. At first, cross-compile environment for target system should be constructed on host computer, and bootloader in charge of Linux booting may be loaded on the target system. Then, kernel image is compiled and loaded on the target system. Embedded Linux porting process is completed when ram disk image is generated. Finally, we load wed-server and device driver on the ram disk and make web-page for remote control using CGI. Through the above process, we implement web-based remote control system and present the result.

1. 서 론

요즘은 정보화 시대 또는 네트워크 시대라고 불리면서 다수의 장비들과 시스템들이 이를 수용하기를 원하고 있다. 즉, 영상, 음성, 이미지 등과 같은 대용량의 데이터가 갈수록 늘어나면서 전송이나 저장에 대한 안정성, 신속성, 그리고 편리성 때문에 네트워크 기능은 여러 분야에 응용화가 요구되어지고 있다. 네트워크 기능을 사용하기 위해서는 PC의 역할이 중요하지만 대부분의 장비들은 네트워크 기능을 요구하는 점 외에는 각각의 쓰임과 특성이 달라서 PC만을 이용하는 방식은 여러 가지 요구를 수용하기가 쉽지 않다. 성능에서는 일반 PC와 차이가 없고 다양한 요구를 수용할 수 있는 장비의 개발이 필요해지면서 Embedded 시스템에 대한 연구와 개발이 활발히 이루어지고 있다.

본 연구에서는 다양한 요구중 하나인 웹 상에서의 제어를 Embedded 시스템을 통해서 구현해 보았다. Embedded board에 Linux를 포팅하고 웹 서버를 올려서 웹페이지 상에서 직접 제어를 하기 위한 리눅스 포팅, 그리고 웹을 이용한 제어의 과정을 설명하고, 그 결과를 제시하였다. 그림 1에 전체적인 개발 프로세스를 도시하였다.

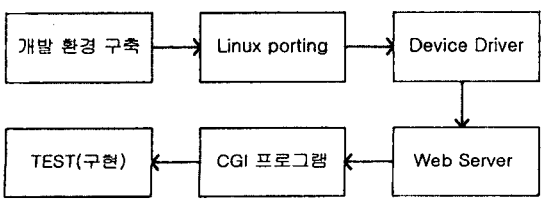

그림 1. 개발 프로세스

2. 임베디드 리눅스 포팅

2.1 임베디드 시스템

임베디드 시스템이라고 하면 어렵고 복잡한 장치라고 생각하기 쉬운데, 복잡한 시스템도 있지만 주변에서 쉽게 볼 수 있는 간단한 장치도 있다. 임베디드 시스템의 가장 큰 특징이라면 이처럼 다양한 스펙트럼을 보여 줄 수 있다는 점이다. 임베디드 시스템은 상위 시스템의 구성 요소를 이루거나 사람의 개입 없이 동작하도록 기대되는 하드웨어와 소프트웨어이다[1]. 전형적인 임베디드 시스템은 전원이 켜져 동작하기 시작하면서 전원이 꺼질 때까지 멈추지 않는 특수한 용도로 사용하는 일부 응용 소프트웨어를 포함하는 단일 보드 마이크로컴퓨터이다. 임베디드 시스템은 운영 체제를 포함하거나 단일 프로그램으로 작성할 수 있을 만큼 단순할 수 있다. 시스템에 필요 없을 경우 키보드, 모니터, 직렬 통신, 대용량 기억장치와 같은 일반적인 주변 장치나 사용자 인터페이스용 소프트웨어를 지원하지 않는다. 그리고, 실시간성을 요구할 때도 있다.

2.2 Linux Porting

리눅스라는 운영 체제는 유닉스와 거의 유사한 운영 체제로서 유닉스의 장점을 이용하여 필요로 하는 기능을 추가한 운영 체제이다. 임베디드 시스템 구현에 사용되어지는 Embedded Linux는 일반적인 PC상에서의 Linux와는 다소 차이가 있다. 본 연구에서 사용할 Embedded Linux는 기본적으로 특정 임베디드 애플리케이션에 맞도록 리눅스 커널의 크기와 성능을 최소, 최적화 시켜 만들어낸 커널을 의미하며 여기에 C 라이브러리나 유틸리티 등이 필요에 따라 추가되거나 교체되어 하나의 시스템을 만드는 형태가 된다. 커널의 최소, 최적화시 커널의 내부 구조의 변화도 가능하며 이를 극단화시키면 일반적인 상용 운영 체제의 성능을 뛰어 넘는 것도 가능하다.

Embedded Linux를 사용하는 이유는 누구나 제약 없이 자유롭게 사용할 수 있는 운영체제이며 소스가 오픈되어 있어 소스를 변형, 개발, 재배포 할 수 있는 특성을 가지고 있기 때문이다. 장점으로는 한 대의 시스템에서 여러 사용자들이 사용할 수 있는 다중 사용자 환경 시스템이나, 여러 가상 작업 공간을 사용자에게 제공하여 한 대의 컴퓨터 내에서 여러 개의 화면을 통하여 다수 작업들을 동시에 수행할 수 있는 다중 작업 및 가상 터미널 환경을 제공하는 점이다. 또, 네트워크 기능을 갖춘 서버로 사용할 수 있도록 인터넷에서 쓰이는 프로토콜을 대부분 지원하므로 인터넷을 위한 각종 데몬 프로그램을 동작시킬 수 있다. 웹 서버, 이메일 서버, DHCP 서버, DNS 서버, FTP 서버와 같이 인터넷을 운영하는데 있어 필수 불가결한 각종 소프트웨어를 임베디드 리눅스에서 운영할 수 있다. 또한, NFS나 AFS와 같은 유닉스를 위한 네트워크 파일 시스템은 물론이고, 삼바를 사용해 윈도우 시스템과도 손쉽게 데이터를 주고 받을 수 있다. MS-Window 운영 체제에서는 하드웨어를 제조한 업체에서 제공한 드라이버를 사용하여야 하지만, 리눅스에서

는 하드웨어를 제공한 업체의 드라이버가 아니더라도 하드웨어가 사용하고 있는 칩만 동일하다면 하나의 드라이버로 모든 제조업체의 하드웨어를 사용할 수 있다. 이러한 리눅스의 특성을 바탕으로 Strong ARM 보드(ez board)에 Linux를 이용하여 시스템을 구현하였다[2-5]. 개발 환경의 구성은 그림 2와 같다.

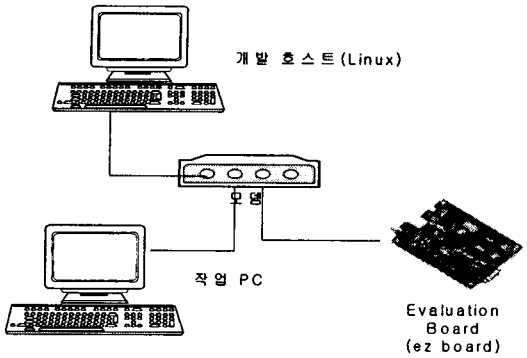

그림 2. 개발 환경 구성도

2.2.1 Cross Compiler 환경

리눅스를 호스트 컴퓨터에서 타겟 컴퓨터로 이식하기 위해서는 작업 환경 구축이 필요하다. 일반적으로 컴파일러는 자신의 시스템에 맞는 바이너리 코드를 만든다. 예를 들어, x86의 시스템에서 gcc를 사용하여 컴파일을 하게 되면 x86의 바이너리 코드가 생긴다. 저장할 수 있는 디스크 공간이 매우 부족하기 때문이다. 그러므로, 타겟용 커널 및 응용 프로그램을 개발하기 위해서 호스트 시스템에 타겟용 크로스 컴파일 환경을 구축한다. x86머신에서 ARM용 바이너리 코드를 만들어 주는 것이 크로스 컴파일러이다.

2.2.2 boot loader

부트로더(boot-loader)는 알기 쉽게 말하면 인텔 관련 보드(x86)에서 말하는 BIOS와 LILO(Linux Loader)를 결합한 것이라고 하면 이해하기가 쉽다. Embedded 장비에서의 부트로더 기능은 하드웨어에 대한 초기화(CPU 속도, 메모리, 인터럽트, UART)를 해주게 되며, 리눅스를 부팅할 수 있게 해주는 기능이 있다. 또한, Kernel 이미지나 Ram disk 이미지를 다운로딩 할 수 있는 기능을 갖고 있다[6]. BLOB(Boot Loader OBject) 실행순서는 그림 3과 같다.

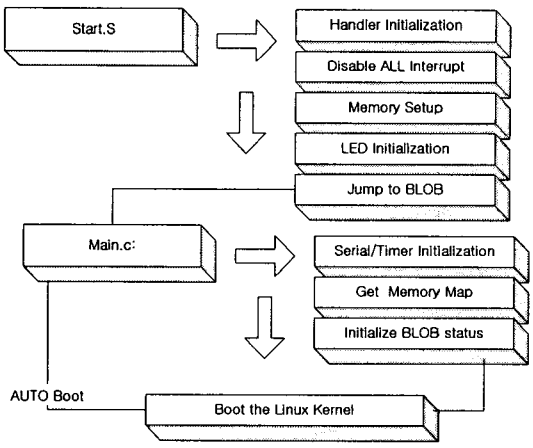

그림 3. BLOB 실행 순서

2.2.3 Kernel Image

커널은 운영 체제의 가장 핵심적인 부분으로서 프로세서와 시스템 메모리를 관리하여 시스템을 효율적이고 원활히 동작시키는 기능을 수행한다. Embedded용 커널은 일반 Linux에서의 커널을 타겟의 특성에 맞게 고쳐 주어야 한다. ARM용 타겟 보드에 커널을 올리기 위해서는 이런 환경에 맞는 커널을 구해야 한다. C 프로그램 작성을 위해서 ARM에 관련된 헤더 파일을 작성해야 하고 부트로더에서 리눅스 커널의 헤더 파일을 참조하며 최종적으로 리눅스 커널을 타겟에 포팅 한다[6]. 커널 이미지의 압축 과정은 그림 4와 같다.

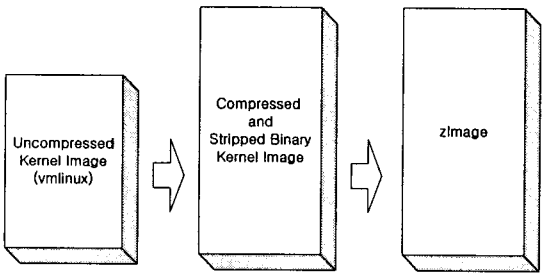

그림 4. 압축된 커널 이미지

2.2.4 Ram disk Image

램디스크는 특별한 물리적 장치를 지칭하는 것이 아니라, 메모리의 일부분을 디스크 드라이브로 인식시켜 이를 하드디스크처럼 사용하기 위한 것이다. 보드에는 하드디스크가 없으므로 이 램디스크에 의해 부팅이 된다. 램디스크에는 커널이 완전히 로드 된 후, 시스템의 구동에 필요한 init 유틸리티, 바이너리 파일들이 있어야 부팅이 된다. 여기서 램디스크 위에 구성되어 있는 유틸리티들과 라이브러리들을 모아 압축한 것이 램디스크 이미지이다. 램은 전원이 꺼지면 기억된 내용이 지워지므로 램디스크 이미지를 특정한 위치에 저장하여야 한다.

3. 시스템 구현

3.1 웹 서버 제작

웹으로 제어를 하기 위해서는 보드 상에서 웹 서버를 구현해야 한다. 보아 웹 서버는 아파치와 같은 강력한 성능을 제공하지는 않지만 임베디드 기기에서 웹 서비스를 제공하기 위한 용도로 적절한 매우 작은 크기의 서버 프로그램이다. 웹 서버에 추가로 들어오는 서버의 요청에 대해 하위 프로세스를 생성하지 않고 내부적으로 모든 연결을 다중화 한다. 크기가 매우 작기 때문에 기능의 제한은 있으나 기본적인 HTML 문서의 전달을 하는 HTTP 프로토콜과 CGI를 기본적으로 갖추고 있다[7].

3.2 디바이스 드라이버 제작

웹 서버를 통한 제어를 위해서는 디바이스 드라이버를 제작해야 한다. 제어하고자 하는 대상에 대한 디바이스 드라이버를 작성해서 ARM용으로 컴파일하고 이를 타겟 보드에 insmod(모듈 등록)시킨다[8][9].

3.3 CGI 구현

웹브라우저에서 HTML로 여러 가지 정보를 처리하지만, 그 기능만으로 모든 정보 처리를 할 수 없다. 이것을 보충하기 위한 외부 프로그램과 웹서버(HTTP Server) 간의 연결 역할을 하기 위한 규약이 CGI로서, 웹 브라우

저와 웹 서버간의 정보 전달을 위한 중간 역할을 제공한다. 사용자는 웹 특정 서비스를 요청하고 필요에 따라 웹 브라우저를 이용하여 데이터를 작성한다. 웹 클라이언트는 사용자에게 데이터를 입력할 수 있도록 하고 입력된 데이터를 인코딩하여 웹 서버에게 전달하며, 웹 서버로부터 넘어 온 서비스 결과를 사용자에게 보여준다. 사용자(웹 클라이언트)의 요청에 따라 해당 CGI 프로그램을 구동하고 사용자로부터 전달된 데이터를 CGI 프로그램에게 전달한다. CGI 프로그램은 웹 서버에 의해 실행되고 웹 서버로부터 전달받은 데이터는 해당 데이터를 해석하여 요청에 맞는 작업을 수행한다. 해당 작업을 수행시 필요에 따라 웹 서버에 존재하는 자원(파일, 데이터 베이스, 시스템 서비스)으로의 접근 결과는 웹 서버에게 일반적으로 HTML로 반환된다. 실행 파일을 .cgi 확장자로 고치고 이 파일을 컴파일해서 웹브라우저에서 실행시킬 수 있다[10]. CGI 프로그램 동작 원리는 그림 5와 같다.

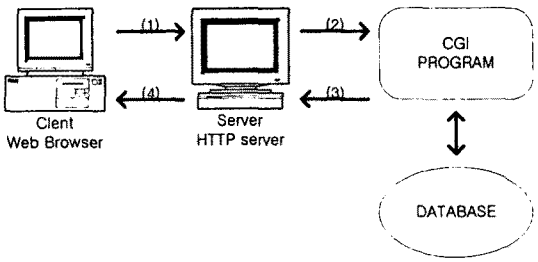

그림 5. CGI 프로그램 동작 원리

3.4 TEST

웹 상에서 CGI를 이용하여 보드에 연결된 LED Device를 제어하였다. 구현한 웹 페이지는 그림 6과 같다.

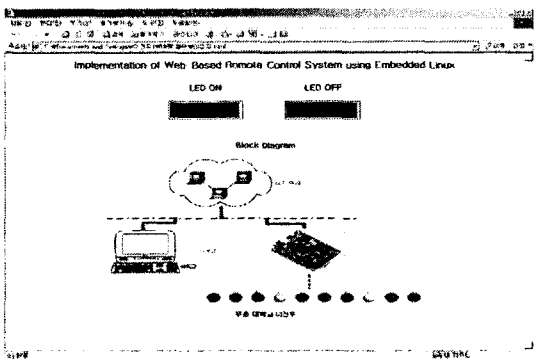

그림 6. 웹 기반 인터페이스

4. 결 론

시대가 발전함에 따라 시스템에 대한 요구 사항도 다양해지고 있어서 Embedded 시스템에 대한 계속적인 연구가 필요하다. 본 연구에서는 기기들의 제어를 웹에서 구현하고, 그 과정 및 결과를 제시하였다. 시스템의 구성은 StrongARM(ez board) 보드에 Linux를 사용하였고, Device Driver는 Linux의 gcc 컴파일러를 이용해서 구현했으며, 웹 서버로는 보아 웹 서버를 이용해서 CGI 프로그램으로 웹에서 제어를 하였다. 구현된 내용은 간단한 것이지만 이를 응용해서 많은 가정용 기기나 산업용 기기에 이용할 수 있을 것이다.

[참 고 문 헌]

[1] 박재호, *임베디드 리눅스*, 한빛미디어, 2002.
[2] 차현준, "네트워킹 디바이스를 위한 임베디드 리눅스 시스템의 설계 및 구현", 포항공과대학교 석사학위 논문, 2000.
[3] http://falinux.co.kr
[4] Intel ®StrongARM * SA-1110 Microprocessor Data Sheet
[5] CS8900A Product Data Sheet
[6] 권수호, *Linux Programming Bible*, 글로벌, 2002.
[7] http://www.boa.org
[8] Alessandro Rubini, *Linux Device Drivers*, 한빛미디어, 1998.
[9] Cameron Newham & Bill Rosenblatt, *Learning the bash Shell-2nd Edition*, 한빛미디어, 1998.
[10] http://kelp.or.kr