전력전자학술대회 논문집 2002. 7. 3 ~ 7. 6

ARM 프로세서와 LINUX를 이용한 마이크로 웹서버 구현

이동훈, 한경호
단국대학교 전기공학과

Implemantation of Micro-Web Server Using ARM Processor and Linux

DongHoon Lee, Kyongho Han
Department Of Electrical Engineering. DanKook University

ABSTRACT

In this paper, we proposed the micro web-server implementation on Strong ARM processor with embedded Linux. The parallel port connecting parallel I/O is controlled via HPPT protocol and web browser program. HTTP protocol is ported into Linux and the micro web server program and port control program are installed on-board memory using CGI to be accessed by web browser, such as Internet Explore and Netscape. 8bit LED and DIP switches are connected to the processor port and the switch input status is monitored and the LED output is controlled from remote hosts vie internet. The result of the proposed embedded micro-web server can be used in automation systems, remote distributed control via internet using web browser.

1. 서 론

전기, 전자, 컴퓨터 기술들이 발달하면서 이를 이용한 많은 기기들이 생활 주변에 들어오게 되었고 이러한 기기와 기기간의 통신환경이 인터넷으로 변화되어 가고 있다. 많은 사람들이 이러한 통신기기들이 더 강력한 기능에 사용하기 쉽고 크기는 더 작아지길 원하고 있으며 이러한 요구를 수용해 줄 수 있는 분야중의 가장 유력한 것이 바로 Embedded System이다.

Embedded System이란 내장형 시스템으로서 특정 기기나 시스템에 H/W, S/W가 내장되어 정해진 기능을 실행하는 시스템으로서 다양한 용도로 활용되는 일반적인 컴퓨터 시스템과 구별하여 명명한 것으로 다양한 시스템을 포함하게 된다.

Embedded System은 일반적으로 S/W가 H/W에 밀접하게 연결되어 개선이 용이하지 않은 경우가 많아 최근에는 S/W가 여러 계층(Booter, OS, 응용프로그램등)으로 분화하여 각기 다운로드 가능한 구조의 시스템으로 진화하게 되었다.

본 논문에서는 ARM 프로세서용 Booter와 Embedded Linux OS를 이용하여 웹서버를 구현하고 CGI 프로그래밍으로 입, 출력을 제어해보고자 한다.

2. Embedded Linux 기반의 웹서버를 이용한 입, 출력 제어

2.1 개발환경

2.1.1 타겟보드 사양

타겟 보드로는 HyBus사에서 제작한 Intel StrongARM SA1110 Processor (32Bit 206MHz), 32Mbyte SDRAM, 16Mbyte 플래쉬를 내장한 보드를 사용하였다.

2.1.2 Host 개발환경

PC를 개발 호스트로 사용하여 gcc를 컴파일러로 사용하였다.

① Wow Linux Paran 7.2 (커널 ARM Patch), GNU Tool, Toolchain, Minicom, Jtag프로그램 J플래쉬

② Windows 2000 Pro, Code Maker, WinNT용J 플래쉬, Hyper Terminal

2.2 StrongARM SA1110 Processor

32비트 방식의 1클럭당 1개의 명령을 수행하는 RISC방식의 프로세서이며 ARM V4의 코어를 사용하고 있다. 400mW의 낮은 소비전력과 매우 작은 패키지 크기를 갖추어서 PDA와 같은 소형화 기기에 유리하며 3.3V I/O입출력, 클럭발생기 내장,

전원 제어 모드, MMU(메모리 관리 유닛), 명령캐쉬(16kb), 데이터 캐쉬(8k) 기능, 28개의 인터럽트 처리 가능한 GPIO등의 기능을 갖추고 있다.

이렇게 SA1110이라는 것은 MCU와 그 외 주변 장치들이 하나의 패키지 형태로 집적되어 있는 시스템이므로 아주 작은 마이크로 보드로 보아도 무방하다.

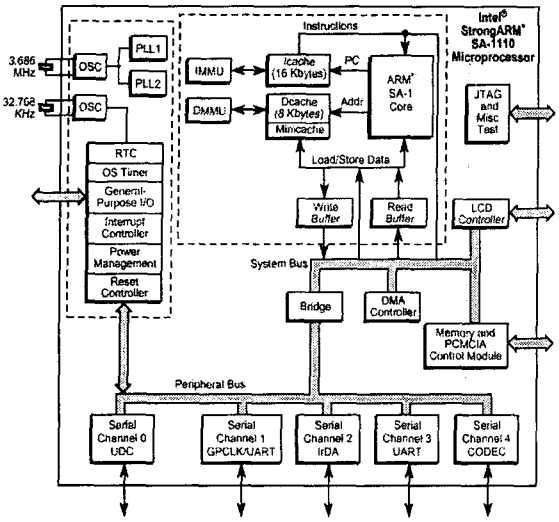

그림 1 SA1110 블록다이어그램

Fig.1 SA1110 Block Diagram

2.3 Embedded Linux

Embedded Linux는 최근 Mobile Networking이 대두되면서 기능이 강력하고 사용자나 개발자의 요구에 가장 발전가능성이 높은 O/S로 인정받고 있다.

Embedded Linux는 네트워크 기능이 매우 뛰어나고 Open Source와 Community가 방대하다는 장점을 가지고 있을 뿐 아니라 로열티가 없고 거의 모든 프로세서 및 플랫폼을 지원하며 안정적이고 다양한 응용에 따른 성장가능성이 높다. 그러나 개발환경이 비교적 어렵고 개발 인력이 부족하며 완전히 검증되지 않았다는 점과 특정 OS에 비해서 Multimedia구현 능력이 떨어진다는 단점이 있다.

2.4 Board Level Porting

포팅은 크게 CPU Level Porting과 Board Level Porting으로 나눌 수 있다. CPU Level Porting이란 Cross개발환경과 Linux커널에 있는 Arch부분을 특정 CPU에 맞도록 일체의 H/W에 관련된 리눅스 커널을 다시 만드는 작업이라고 할 수 있고 해당 CPU에 관계되는 Arch 부분이 일반 리눅스에 포함되었거나 Patch가 존재하는 경우는 Cross개발환경을 구축하고 해당보드 특성과 관련된 부분만 수정하여 사용하면 된다.

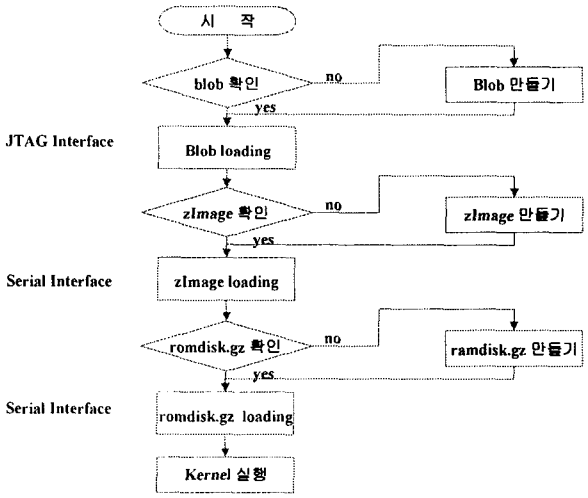

그림 2 리눅스 Loading 과정

fig.2 Linux Loading Flow

2.4.1 Cross개발환경 구축

Cross개발환경이란 Host System에 Target 디바이스용 리눅스를 개발하기 위한 모든 환경을 말한다. 우선 소프트웨어 개발을 진행하기 위해 필요한 Host System의 Cross Compile환경을 구축하기 위해 각종 소스들을 컴파일하고 Build하여 실행 Binary를 생성하는데 필요한 각종 Utility 및 Library의 모음인 ToolChain을 설치, 컴파일 한다. GNU에서 제공하는 ToolChain의 내용은 GNU GCC Compilers For C, C++, GNU Binary Utilities, GNU C Library를 포함한다.

2.4.2 Bootloader Porting

일반적으로 Embedded System에서는 bios에 해당하는 Bootloader가 사용되며 그 기능은 H/W(MCU, SDRAM, 플래쉬, UART)등을 초기화하며 리눅스를 부팅(커널이미지를 SDRAM에 저장한 후 커널 이미지의 주소로 점프)하고 커널과 파일 시스템을 다운로딩하고 플래쉬 메모리에 쓰는 기능 등을 수행한다. ARM에 보통 쓰이는 Bootloader는 BLOB이 있으며 본 연구에서도 BLOB-2.0.5-pre2를 보드에 맞게 포팅하여 사용한다. BLOB을 컴파일 하기 위해서는 우선 ToolChain이 설치되어 있어야 하며 이를 플래쉬 메모리에 직접 쓰기 위하여 JTAG을 사용한다. JTAG프로그램으로 J플래쉬를 사용하였으며 이를 이용하여 플래쉬 메모리에 쓰고 부팅할 수 있다.

표 1. 플래시 메모리 맵
Table 1. 플래쉬 Memory Map

User	0x01000000 0x00480000
Configuration	0x00380000
파일 시스템	0x00180000
커널	0x00080000
Boot Param	0x00040000
Bootloader	0x00000000

2.4.3 커널 컴파일 및 포팅

커널은 버젼2.4.5를 사용했으며 ARM 패치, SA1110 Patch를 적용시킨 것을 사용한다. 커널 컴파일은 일반 리눅스에서의 컴파일과 크게 다르지 않으며 구성을 하고 나서 의존도를 검사하고 이미지 파일을 만들 수 있다. 만들어진 이미지를 BLOB을 이용하여 SDRAM에 다운로드 받고 이를 BLOB의 플래쉬명령을 사용하여 플래쉬의 0x00080000 번지에 올린다. 이로서 커널을 부팅시킬 수 있다.

2.4.4 파일 시스템

Embedded Linux에서 우리는 램 디스크, Jffs, Jffs2, Cramfs, Ramfs등을 사용할 수 있다. 램 디스크는 메모리의 일부를 디스크로 인식시킨 것이며 가장 일반적으로 램 디스크를 Root 파일 시스템으로 사용한다. 램 디스크는 Ram에서 동작하기 때문에, 속도가 빠를 뿐만 아니라 Gzip압축을 사용하여 용량을 줄일 수 있다는 장점이 있다. Cramfs(Cram a 파일 시스템 onto a small ROM)는 일반적으로 Rom에 구현되는 파일시스템으로 Read-Only이며 Zlib Routines를 사용하여 공간을 절약할 수 있고 안전하게 Data를 보존할 수 있다. Jffs(Journalling 플래쉬 File System)는 플래쉬에서 구현되는 파일시스템으로 보통 램 디스크와 같이 구현이 되며, 저장이 되어야하는 내용들을 Jffs를 구현하여 사용한다. 이러한 파일시스템을 이용하여 만들어진 이미지를 BLOB을 이용하여 다운로드, 플래쉬에 저장하면 리눅스로 부팅할 수 있게 된다. 본 연구에서는 램 디스크에 WebServer인 GoAhead WebServer를 컴파일해서 추가하고 포팅한다..

2.5 디바이스 드라이버

GPIO를 사용하기 위하여 디바이스 드라이버를 작성하여 모듈화 하여야 한다. 디바이스 드라이버는 크게 응용프로그램이 직접 읽고 쓰는 문자 디바이스와 커널이 사용하는 블록 디바이스, 커널 네트워크 프로토콜에서 사용하는 네트워크 디바이스 등으로 나눌 수 있으며 만든 디바이스 드라이버를 컴파일하여 생성되는 오브젝트 파일이 GPIO Character 디바이스 드라이버이며 타겟보드에 Module을 실장하고 제어하고자하는 응용 프로그램에서 모듈을 열어 읽고 쓴 다음 모듈을 닫고 사용할 수 있다.

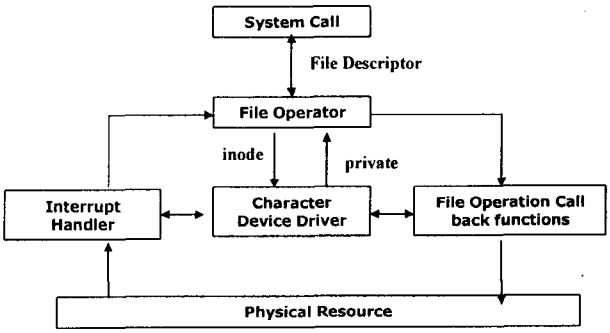

그림 3 디바이스 드라이버의 구조
Fig.3 디바이스 드라이버 structure

2.6 Web Programming

범용 Browser에서 HTML을 이용하여 하드웨어를 컨트롤하기 위해서 C언어를 이용한 CGI를 사용한다.

2.6.1 CGI(Common Gateway Interface)

프로그래밍을 위하여 가장 강력하고 일반화되어있는 C언어를 이용하여 프로그래밍하고 이를 ARM-LINUX-GCC로 CGI컴파일 하여 웹페이지(HTML)에서 프로그램을 불러와 사용할 수 있도록 하여 GPIO 입, 출력을 제어할 수 있다. 포팅한 Webserver인 GoAhead에서 CGI를 지원하고 있으며 Host System에서 CGI로 컴파일 할 수 있다.

2.6.2 HTML

웹페이지(HTML)에서 입력을 받아 컴파일한

CGI 프로그램을 실행하고 이 프로그램이 GPIO를 제어하면 결과를 다시 웹페이지(HTML)로 출력 할 수 있다.

3. 결 론

본 연구는, Embedded System의 OS중 여러 면에서 장점을 가지고 있어 최근 대두되고 있는 Linux를 StrongARM MCU에 포팅하여 Embedded Web Server를 구현하고 Web Server를 기반으로 GPIO 입, 출력을 위한 디바이스 드라이버를 제작하고 CGI프로그래밍을 이용하여 GPIO 입, 출력을 제어하려고 하였다. Web 기반의 하드웨어 제어가 가능하므로 이 시스템을 통해 원격 디스플레이, 데이터검출, 건물이나 사무자동화 등과 같은 수많은 전기, 전자 제품에 응용될 수 있을 것이라고 기대된다. 또한 S/W를 계층화하여 OS를 사용하는 시스템이므로 개발 후 S/W의 업그레이드도 손쉽게 할 수 있다고 기대되며, 더 나아가 Linux OS의 이점을 살려 윈도우용 응용프로그램 개발로 사용자에게 더 쉬운 인터페이스를 제공할 수 있을 것이다.

참 고 문 헌

[1] Intel StrongARM SA-1110 Microprocessor, "Developer's Manual", 2000, June.

[2] Alessandro Rubini, "Linux 디바이스 드라이버s", 2000, January.

[3] Michael Barr, "Programming Embedded System : In C And C++", 1999, June.

[4] Jeremy Bentham, "TCP/IP LEAN : Web Servers For Embedded Systems", pp. 207-268, 2000.

[5] Warren Gay, "C 사용자를 위한 리눅스 프로그래밍", pp. 87-109, 2000, April.

[6] Richard Stones & Neil Matthew, "Beginnning Linux Programming", pp. 955-1156, 2000, April.

[7] Neil Matthew, Richard Stones외 13인 공저, "Professional Linux Programming", 2001, June.

[8] Stepben Asbury, Jason Mathews, Selena Sol, With Kevin Greer, "CGI How-To : The Definitive CGI Scripting Problem-Solver", pp. 445-482, 1996.

[9] Ed Tittel, Mark Gaither, Sebastian Hassinger, & Mike Erwin, "CGI Bible", 1997, April.