리얼타임 임베디드 웹서버 시스템

윤종일* · 김은연* · 곽군평*

*국립 창원대학교 전기공학과

Real Time Embedded Web Server System

Jong-il Yoon* · Eun-yeon Kim* · Gun-pyong Kwak*

*Electric Engineering Department of Changwon National University

E-mail : web- angel@orgio.net

요 약

최근 multimedia와 network 분야의 발달로 embedded system에서도 multimedia 정보를 처리하거나 network를 접속하는 기능들이 포함되고 있다. 따라서 system이 더욱 복잡해지고 기존의 방법과는 다른 새로운 접근 방법이 필요하게 되었다. 일반 computer system에서 쓰이는 것과는 달리 대부분의 embedded system은 real-time이라는 특성을 만족시켜야 했기 때문에 나오게 된 것이 바로 real-time system이다. 본 논문에서는 real-time embedded web-server system을 구현하기 위해 strong-arm board에 Linux로 web-server를 구축하고 kernel을 재구성하였다.

ABSTRACT

Recently, most of the embedded system have the function of multimedia information processing and network connecting. This make the overall system more complicated and need a new approach. Moreover the embedded system must be real-time system, which are different with Linux is constructed and the kernel is redesigned on the strong-arm board in order to make the real-time embedded web-server system.

키워드

Embedded, Web-server, Real-Time, Linux, Strong-Arm

Ⅰ. 서 론

오늘날 많은 기기들이 인터넷에 연결되고 있다. 어떤 냉장고들은 인터넷에 연결되어 온도, 문의 개폐 여부, 현재 저장량과 같은 상태를 웹 브라우저를 통해 모니터 및 조절 할 수 있다. 그리고 오늘날의 인터넷 시대에 많은 telerobots이 있다. 'teletobot' 이란 network을 통해 원격에서 컨트롤할 수 있는 로봇을 말한다. 그리고 인터넷을 통해 원격에서 상태를 모니터할 수 있는 세탁기도 있다. 그리고 어떤 사람들은 집안의 기기들을 네트워크로 연결하여 집기들을 원격에서 조절하기를 바란다. 이처럼 기기들이 인터넷에 연결된다면 많은 서비스를 제공할 수 있다. 위에서 기술된 기기들의 대부분은 RS-232C와 같이 매우 단순한 통신 프로토콜만을 지원한다. 이런 단순한 통신 프로토콜은 직접 인터넷에 연결되기에는 충분치 않다. 따라서 이런 디바이스와 인터넷을 연결하기 위한 게이트웨이가 필요하다. 이런 문제에 대해 많은 해법이 제시되고 있지만 그들을 3가지로 분류할 수 있다.

첫 번째 방법은 전체 웹 서버를 기기 속으로 집어넣는 것이다. 하지만 대부분 임베디드 프로세서들은 웹 서버를 구동시키기에는 충분치 않은 연산능력과 메모리 크기를 갖고 있다. 따라서 이런 접근 방법은 지금 현실상 가격 및 효율이 떨어진다.

두 번째 방법은 게이트웨이로 PC를 사용하는 것이다. 현재 대부분의 기기들은 PC나 워크스테이션을 경유하여 인터넷에 연결된다. 그 기기는 최소한의 요구되는 정보를 RS-232C와 같은 통신 포트를 통해 PC에

보내고, PC에 있는 웹서버는 기기의 정보를 수집·분석하여 HTML페이지를 만들고 클라이언트에게 전송할 수 있다. 이런 접근 방법에서는 기기에 새로운 연산능력과 메모리 사이즈를 요구하지 않는 장점이 있지만 PC는 많은 경우 너무 비싸고 크기 때문에 가격이나 효율이 떨어진다.

세 번째 방법은 작고 싼 웹전용 임베디드 시스템을 개발하여 사용하는 것이다. 이런 임베디드 웹서버는 Ethernet 포트, 시리얼 및 패러럴 포트를 갖추고 있어야 하며 크기가 작고 가격이 저렴해야 한다. 또한 이러한 임베디드 시스템은 특성상 Real Time을 요구한다. 이러한 관점에서 보면 Linux가 임베디드 웹서버 시스템의 운영체제로서 가장 적당하다. Linux 소스코드는 GNU General Public License에 따라 누구나 아무런 대가 없이 이용 가능하고 비교적 안정적인 시스템일 뿐만 아니라 오픈소스이기 때문에 개발이 용이하다.

또한 임베디드 시스템은 전력 소비가 적어야 한다. 따라서 본 논문에서는 전력 소비가 비교적 적은 Strong-ARM Board에 Linux Kernel을 재구성하여 웹서버를 구축했다. 타겟을 제어할 제어기와 시리얼 통신으로 연결하고, 웹서버는 Ethernet를 통해 인터넷에 연결하여 전체 시스템을 구성했다.

II. 시스템의 구성

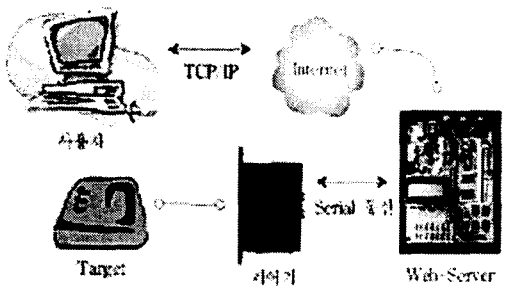

그림 1. 전체 시스템 구성도

그림 1은 전체 시스템의 구성이다. 사용자는 인터넷을 통해 원거리에서 웹서버에 접속하여 타겟의 상황을 모니터링하고 제어기를 통해 타겟을 컨트롤할 수 있다.

표 1은 Linux를 포팅시킬 module 구성을 나타내었다. CPU는 저전력이며 고성능을 자랑하는 인텔 스트롱암 SA1110을 사용하였고, flash 메모리는 인텔 스트라타 flash 메모리 16M와 32M의 SRAM을 사용하였다.

표 1. module specification

부 분	내 용	사 양
Module	CPU	인텔 SA1110BB
	SDRAM	32M
	Flash	16M
	Interface	LCD용 헤더핀
		JTAG용 헤더핀
		144pin SODIM
	Power	3.3VDC
		1.7VDC

표 2. Board Specification

부 분	내 용	사 양
Board	Ethernet	CS8900
	Serial	3 Port
	USB	Type B Port
	LCD	Character(16x2)
	Connector	JTAG
		GPIO
		SA1110 확장용
	Power	5VDC
		3.3VDC

표 2는 모듈을 탑재한 보드의 구성을 나타내었다. 개발환경으로 모니터링과 시리얼 다운로딩이 가능한 Serial Port가 3개가 구성되어 있고 network에 연동될 수 있도록 CS8900 Ethernet Controller를 사용하여 network 및 tftp 기능이 가능하도록 하였다. 또한 JTAG Port를 통하여 H/W Test 및 플레쉬에 데이터의 fusing할 수 있도록 하였다. 그리고 SA1110의 모든 기능을 사용 가능하도록 하기 위해 확장용 connector를 사용하였다. 또한 Charactor LCD와 GPIO S/W를 이용하여 디바이스 드라이버를 제작하여 테스트가 용이하도록 설계되어 있다.

III. zImage 만들기

Cross 개발 환경이란 host system에 target device용 Linux를 개발하기 위한 모든 환경을 말한다. 먼저 해당 CPU에 맞는 tool chain환경을 구축해야 한다. Tool chain이란 target device의 Software개발을 진행하기 위해 필요한 host system의 cross compile환경을 말한다. Tool-chain은 각종 source들을 Compile하고 Build하여 실행 Binary를 생성하는데 필요한 각종

Utility 및 Library의 모음이다. 기본적으로 Assembler, linker, C compiler, C library 등으로 구성되어 있다. 여기서는 GNU에서 제공하는 Tool-chain을 사용하며 다음과 같다.

- GNU GCC compilers for C, C++
- GNU binary utilities
  (assembler, linker various object file utilities)
- GNU C library

Srong-ARM에 사용하기 위한 ARM tool chain은 다음과 같은 사항으로 구성되어있다.

- binutils -arm-2.9.5.0
- gcc-arm-2.95.2
- libc-dev-arm-2.1.3
- cpp-arm-2.95.2
- g++-arm-2.95.2
- libstdc++2.10
- arm-2.95.2
- libstdc++2.10-dev-arm-2.95.2
- 그외 각종 library

zImage는 그림 2와 같은 과정을 통해서 생성한다.

1. Cross Compier 복사 및 압축 풀기
2. kernel source 복사 및 압축 풀기
3. Patch 및 link 하기

Kernel을 compile하는 과정에 옵션을 설정하는 과용량 정이 있다. 여기서 우리가 사용하고자하는 기능을 추가시키고 필요 없는 기능을 제거함으로서 zImage의

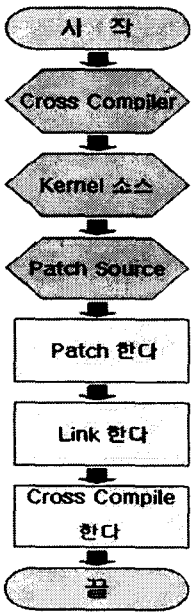

그림 2. zImage의 생성과정

을 줄이고 특정화된 시스템을 만들어 메모리의 낭비를 줄일 수 있다.

이렇게 생성된 zImage는 그림 3에서처럼 처음에는 자체로 실행 가능한 vmlinux로 되어있다. 그것을 압축하여 Piggy.o 만들고 거기에 기본적인 하드웨어 설정과 압축을 풀어 실행가능하게 하는 프로그램이 있는 boot-loader가 더해져 zImage를 구성한다.

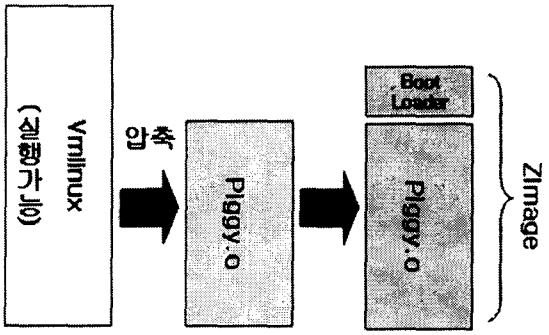

그림 3. zImage의 구조

IV. Web Server

Ethernet이란 1970년대에 Xerox에서 개발한 근거리의 packet교환 network을 말하는 것으로, IEEE 802.3 표준으로 정해져 있다. 현재는 가장 인기 있는 LAN 상에서 사용하는 기술이며, 10Mbps나 100Mbps로 data를 전송한다.

CSMA/CD(Carries Sense Multiple Access/Collision Detection)방식으로 작동하는데, 이것은 전송미디어에 다른 host에서 packet을 보내고 있을 경우 충돌(collision)이 있음을 보내고자 하는 host측에서 알게 되며, 얼마간의 간격을 두고 다시 보내려는 시도를 하게 된다는 것이다. 따라서, 전송속도가 100Mbps라고 하더라도 traffic이 많을 경우에는 이러한 속도가 나오지 못한다. 그래서 서버와의 접속이 끊기거나 장시간 중단된다면 내부적으로 짜여진 프로그램이 실행되어 제어기의 폭주를 방지해야 한다. 그리고 가능한 빨리 재접속을 할 수 있도록 해야 한다. Ethernet에서는 주소를 지정하는 방법으로 6byte의 card에 고유한 번호를 사용한다. 이렇게 할당된 address는 IP address와 함께, network하에서 system을 찾을 수 있도록 한다.

모든 TCP/IP 통신의 시작은 inetd 슈퍼서버를 통해 관리된다.

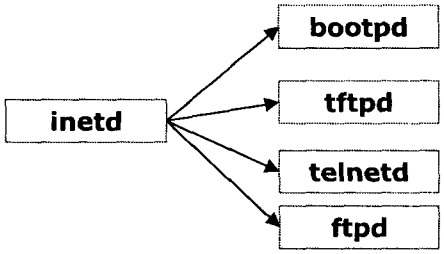

그림 4. 리눅스에서의 TCP/IP 통신

Inetd은 TCP/IP를 통하여 통신을 하고자 할 때 이를 관리하는 데몬이다. 그래서 TCP/IP와 관련된 프로토콜은 항상 inetd로 부터 각각의 해당되는 TCP/IP 관련 데몬을 호출한다.

socket을 통해서 application은 data를 network을 통해서 목적지로 전송할 수 있다.Socket에서 사용하는 data들은 socket buffer라는 형태로 만들어져서 linux에서 제공하는 protocol stack의 처리를 받게 되며, 최종적으로 network device driver에 도달하게 된다.

V. 실험 결과

일단 다른 보드에 리눅스를 포팅하기 위해서는 startkernel이란 펑션 이전까지의 작업과 ramdisk가 uncompressing되고 마운트 될 때까지의 과정이 리눅스 포팅에 중요하다. 여기까지 되면 리눅스가 올라가고 나머지는 일반 우리가 사용하는 리눅스라고 생각하고 작업하면 된다.

전체과정은 bootloader이후에서 startkernel전까지의 과정이다. 먼저 주요 수행과정을 보면 zImage의 압축을 푸는 과정과 cache사용 과정 프로세서 타입과 아키텍쳐 타입의 검증과정이 수행되고 page table을 생성하고 startkernel로 넘어간다.

부트로더에서 start_kernel까지의 수행 과정

1. zImage의 압축 푸는 과정.
2. Cache 사용 과정
3. Processor_type 검증 과정
4. Architecture_type 검증 과정
5. Page_table 생성 과정

그림 5의 상태에서 enter를 치면 그림 4와 같이 bootloader가 실행된다. bootloader는 flash 0 block에서 동작하여 하드웨어를 초기화, linux를 booting, kernel 및 ramdisk의 download 및 flash에 write하는 기능이 있다.

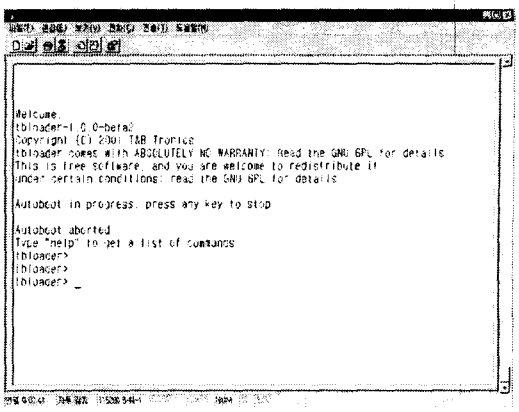

그림 4. bootloader가 실행된 화면

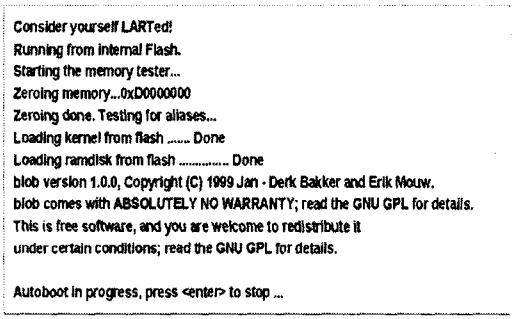

그림 5. booting 중인 화면

VI. 결 론

임베디드 물결이 국내 IT(정보통신)업계에 크게 일고 있다. 개인휴대단말기(PDA), 이동전화, 스크린폰, 인터넷TV, 셋톱박스, 정보가전제품과 각종 산업용 장비 등에 인터넷이 연결되어 실제 하드웨어를 구동하게 되었다. 일상생활의 어느 곳, 어떤 장비에도 컴퓨팅 요소가 장착되는 제3세대의 핵심기술로 앞으로도 무한한 발전 가능성을 지니고 있다. 이에 따라 국내외 IT 업체들은 임베디드 운영체제를 비롯한 소프트웨어와 애플리케이션 개발에 뛰어들고 있다.

그러나 아직 인터넷 속도가 안정되지 않아 산업 현장에서의 hard real time은 구현되기 어렵고 편리한 인터페이스 기능을 구현하는데 제약이 있어 아직 이러한 시스템을 실용화시키는 것은 시간이 필요할 것 같다. 본 논문에서는 리얼타임 임베디드 웹서버 시스템의 가능성을 검증하고 적용 방법을 모색해 보았으며 앞으로 보완해야할 점들에 대해서 알아보았다. 머지 않아 초고속 인터넷 시대가 오고 대용량의 마이크로 칩이 개발되면 본격적인 실용화가 이루어지리라 보며 경제적

인 파급효과도 엄청날 것으로 예상된다.

참고문헌

[1] Intel StrongARM SA-1110 Manual
[2] Sandra Loosemore외 3인, The GNU C Library Reference Manual
[3] 이연조, 임베디드 리눅스 프로그래밍
[4] 오재준, OS 제작의 정석
[5] Jean J. Labrosse MicroC/OS-II The Real-Time Kernel
[6] Kurt Qall 외 2인, Linux Programming Unleashed
[7] Richard Stones & Neil Matthew, Beginning Linux Programming
[8] Jeremy Bentham, TCP/IP LEAN

본 연구는 과학기술부 · 한국과학재단 지정 창원대학교 공작기계기술연구센터의 지원에 의한 것입니다.