임베디드 리눅스 시스템 설계 및 구현

박윤미[0], 성영락[†], 이철훈
충남대학교 컴퓨터공학과, [†] 국민대학교 전자정보통신공학부
(ympark[0], chlee)@.ce.cnu.ac.kr, [†] yeong@mail.kookmin.ac.kr

Design and Implementation of an Embedded Linux System

Yun-Mi Park[0], Yeong Rak Seong[†], and Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.
[†] School of Electrical Engineering, Kookmin Univ.

요 약

전세계적으로 Post PC 시장이 점점 확대되고 있다. 이 시장의 핵심 기술은 임베디드 리눅스로 우리 생활에서 쓰이는 각종 전자기기, 정보가전, 제어장치 등에 적용되고 있다. 최근에는 멀티미디어 처리와 같은 점차 복잡한 기능이 요구되면서 임베디드 운영체제를 사용하기 시작하였고, 이러한 운영체제로 리눅스가 주목받게 되었다. 이는 가격경쟁력뿐 아니라 각 H/W 에 따라 운영체제와 응용프로그램을 다양하게 변화시킬 수 있다. 본 논문에서는 안정적이고, 다양한 서비스를 제공해주며, 공개되어있는 리눅스를 이용해 임베디드 시스템을 구현한 내용에 대해 기술한다.

1. 서론

전기, 전자, 컴퓨터 기술이 발달하면서 이들 기술을 이용한 다양한 기기들이 우리의 생활 주변에 들어오게 되었다. 임베디드 시스템은 일상 생활에서 사용되고 있는 전자 가전제품 뿐만 아니라 핸드폰, PDA, 그밖의 항공 관제 시스템, 우주선 제어장치, 군사용 제어 장치 등에 사용되어 우리 생활에 도움을 주고 있다.

임베디드 시스템이란 미리 정해진 특정 기능을 수행하기 위해 컴퓨터의 하드웨어와 소프트웨어가 조합된 전자 제어 시스템을 말한다. 임베디드 시스템은 개발, 설치, 관리 등 모든 부분에서 소프트웨어의 비중이 하드웨어에 비하여 상대적으로 커지고 있는 추세이다. 그리고 기존의 시스템 소프트웨어들은 비교적 안정적이고 특정한 분야에서 상당히 최적화되어 있으나 크기가 너무 크고, 비공개된 소스로 인한 비싼 로열티로 인해 대중화와 상용화에 상당한 걸림돌이 되고 있다. 최근 가장 주목받고 있는 시스템 소프트웨어는 소스가 공개되어 있고, 매우 안정적이고, 다양한 서비스를 제공해 주는 리눅스이다. 본 논문에서는 이러한 리눅스의 장점을 이용하여 임베디드 시스템을 구현하였다.

본 논문은 2 장에서 임베디드 시스템 개요에 대해 설명하고, 3 장에서는 리눅스를 이용한 임베디드 시스템 설계 및 구현을, 마지막 4 장에서는 결론 및 향후 연구과제를 기술하였다.

2. 관련 연구

2.1 임베디드 시스템과 리눅스

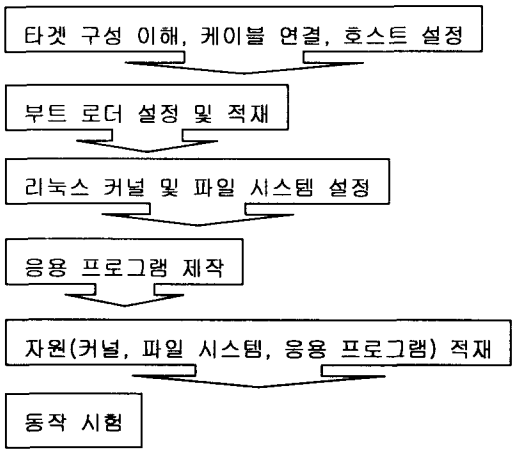
임베디드 시스템은 범용 시스템과 달리 특정한 작업만을 하도록 설계되어 있으며, 이전의 소규모 임베디드 시스템은 비교적 단순한 형태로 운영체제가 없이 순차적인 프로그램을 작성해서 실행되도록 하였고 단지 인터럽트가 발생되는 경우에만 순차적인 프로그램 실행 순서에서 잠시 벗어나는 정도였다. 따라서 운영체제를 사용한다는 것은 오히려 시스템 자원의 낭비만 초래하였다. 그러나 최근의 임베디드 시스템은 시스템 자체의 규모가 커지게 되고 네트워크나 멀티미디어가 시스템에 기본으로 자리잡으면서 순차적인 프로그램만으로는 운용이 매우 어렵게 되었다. 이런 문제점으로 인해 운영체제의 필요성이 대두되기 시작하였고 임베디드 시스템에 맞는 운영체제를 제작하거나 기존의 운영체제 가운데 하나를 선택하여 임베디드 시스템에서 동작하도록 하는 작업이 시작되었다.

임베디드 시스템에서 적용되는 운영체제인 리눅스는 운영체제 소스코드가 공개되어 있기 때문에 별도의 사용료를 지불하지 않아도 된다는 장점과 인터넷에 흘어져 있는 수 많은 공개된 개발 문서나 관련 소스와 같은 풍부한 자원 및 광범위한 하드웨어를 지원한다. 이로 인해 리눅스 기반의 임베디드 시스템이 주목받기 시작하였고, 현재 국내외 임베디드 시스템 분야에 폭 넓게 적용되고 있다.

3. 임베디드 시스템 설계 및 구현

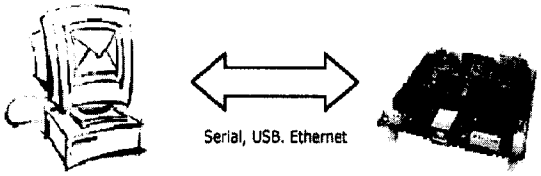
임베디드 시스템에 전원이 인가되면 부가적인

외부장비의 도움 없이 독립적으로 초기화를 마치고 예정된 작업을 수행하는 것이 일반적이지만 독립적인 임베디드 시스템으로 완성되기까지 개발 과정상 호스트라고 불리는 부가적인 외부 시스템에 의존하여 개발이 이루어진다. 임베디드 시스템은 특정한 목적을 위해서 간략하게 만들어졌기 때문에 개발작업에 필요한 환경구성이 불가능하다. 그래서 범용 시스템인 호스트상에서 환경을 구축하게 된다. 절차는 [그림 1]과 같다.



[그림 1] 임베디드 리눅스 개발 과정

3.1 구현 환경



[그림 2] 구현 환경

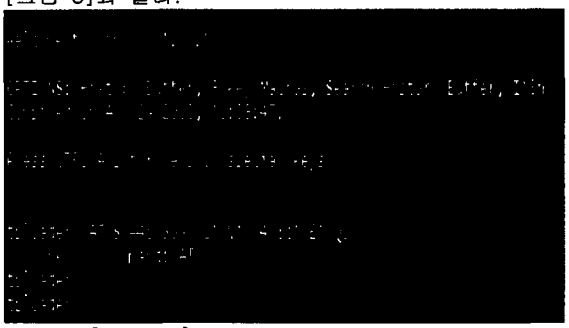
구현 환경은 [그림 2]와 같다. 호스트 시스템에서 부트로더, 커널, 파일 시스템을 구현한후, 그 결과물을 Hyper104 StrongARM(SA-1110) 타겟 보드로 다운로드하여 수행시키는 구조로 되어 있다.

호스트 개발 환경은 와우 리눅스 7.0 을 이용하였고, Hyper104 타겟 보드용 크로스 컴파일러를 사용하여 Hyper104 상에서 실행 가능하게 하였으며, 부트로더를 이용하여 다운로드를 수행하였다. 마지막으로 시리얼 케이블을 통해 타겟 보드에서 출력되는 메시지를 보거나 호스트에서 타겟 보드로 명령을 전달하기 위해 minicom 을 이용하였고, 임베디드 시스템이 실행되는 과정을 보고, 터치 스크린을 통해 결과 를 확인하였다.

3.2 구현 내용

3.2.1 부트 로더 설정 및 적재

Hyper104 타겟 보드에 맞추어 부트로더를 설정하고 전원 인가시에 동작하도록 플래시 메모리 저장 장치에 적재한다. 부트로더는 CPU clock, Memory timing, interrupt, UART 등 하드웨어를 초기화하고, 커널 이미지를 SDRAM 에 저장한 후 커널 이미지의 주소로 점프하여 리눅스를 부팅한다. 그리고 자원(kernel, file system, application)들을 시리얼이나 이더넷을 통해 Hyper104 타겟 보드의 SDRAM, 플래쉬 메모리 저장 장치에 다운로딩이 가능하도록 한다. 부트로더 환경은 [그림 3]과 같다.



[그림 3] minicom 상에서의 부트로더

3.2.2 커널 설정

리눅스 커널은 CPU, 메모리, 프로세스, IPC, 파일시스템, 네트워크 자원들을 관리하고 분배하는 기능 및 디바이스 드라이버에 의한 주변 장치 제어 기능들을 제공한다. 본 연구에서는 일반 리눅스 커널(2.4.5)에 ARM, SA1100, hyper104 patch 들을 적용시킨 커널 소스를 사용하였다. 이는 Hyper104 타겟 보드용 크로스 컴파일러를 이용해서 컴파일하고 커널 이미지를 생성한다.

3.2.3 파일 시스템

임베디드 시스템은 일반적으로 특정 목적을 위해 최대한 간결하게 설계되므로 하드디스크와 같은 대용량 저장 장치가 배제되어 있는 경우가 많다. 그러나 임베디드 시스템에서도 운영체제가 올바르게 동작하기 위해 하드디스크 역할을 대신할 수 있는 저장 매체가 있어야 한다. 이를 위해서 메모리의 일부를 디스크로 인식하는 램디스크로 구현된 파일 시스템과 개인정보 관리 시스템을 구현한 어플리케이션 실행파일과 라이브러리들을 적재 시킨 usr 파일시스템(저널링 파일시스템)을 각각 이미지로 생성한다.

3.3 다운로드

최종적인 결과물이 생성되면 부트로더를 이용해서 호스트에서 타겟보드로 다운로딩을 하고, 이는 타겟 보드의 SDRAM 에 적재된다. SDRAM 에 적재된 결과물을 타겟 보드의 플래시 메모리에 다운로드 함으로써 완전한 임베디드 시스템이 구현된다.

메모리에 각각 다운로드된 영역을 살펴보면 [표 1] 및 [표 2]와 같다.

[표 1] 플래시 메모리 (Rom 16M) 주소 영역

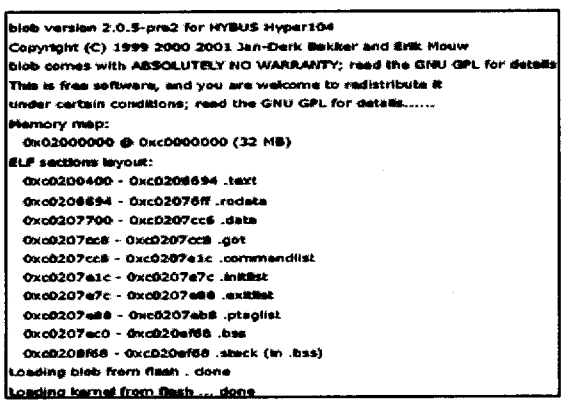
Physical Address	Usage
0x00000000~0x0001ffff	Boot loader
0x00020000~0x0003ffff	Boot script
0x0004 0000 ~ 0x000fffff	Not used
0x0010 0000 ~ 0x001fffff	Kernel
0x0020 0000 ~ 0x002fffff	Ramdisk
0x00300000~ 0x00ffffff	/usr

[표 2] SDRAM (32M) 주소 영역

Physical Address	Usage
0xc000 8000 ~	Linux Kernel
0xc050 0000 ~	Ramdisk

3.4 테스트 및 결과

본 논문에서는 StrongARM (SA-1100)을 탑재한 hyper104 타겟 보드상에 리눅스 커널을 이용하여 임베디드 시스템을 구동하였다. [그림 4]는 minicom을 이용하여 리눅스 커널의 부팅과정을 확인하는 과정을 보여준다.



[그림 4] 커널 부팅 과정

[그림 5]는 임베디드 시스템이 실제로 구동되어 터치 스크린을 이용하는 과정을 보여주고 있다.

[그림 5] 임베디드 시스템을 이용한 터치스크린

4. 결론 및 향후 연구 과제

본 논문에서는 컴퓨터 하드웨어와 소프트웨어가 결합해서 특정한 기능을 수행하는 리눅스 기반의 임베디드 시스템을 구현하였으며, 이를 터치 스크린 장치를 통해 확인해 보았다. 본 논문에서 구현된 리눅스는 공개된 소스로 인해 로열티의 부담없이 채택하여 안정적으로 사용할 수 있었다. 그러나 항공 관제 시스템, 우주선 제어장치, 군사용 제어 장치 등에 사용되는 임베디드 시스템에서 시간결정성이 중요한 이슈가 되었다. 대부분의 내장형 리눅스는 데스크탑에 사용하던 리눅스를 그대로 이식하는 경우가 많아 시간적인 요구사항을 만족시키지 못하고 있어 이에 대한 부분은 계속 연구되어야 할 것이다.

5. 참고 문헌

[1] Hybus homepage, http://www.hybus.net.
[2] CoreBell homepage, http://www.corebell.com.
[3] 이연조, *Embedded Linux Programming*, PC Book, 2002.
[4] 박영환, *Embedded System & Embedded Linux*, 사이텍미디어, 2002.
[5] Jean J. Labrosse, *uC/OS: The Real-Time Kernel*, R&D Publications, 1995.
[6] D. Comer, *Operating System Design VOL 1: The XINU Approach*, Prentice-Hall, 1988.