임베디드 리눅스 시스템을 이용한 향상된 기능의 라우터 구현

*김진태, 박종구
성균관대학교 정보통신공학부
e-mail : *kim94@lycos.co.kr, pjk@yurim.skku.ac.kr*

Implementation of Improved Functional Router Using Embedded Linux System

*Jin-Tae Kim, Jong-Koo Park
School of Information and Communication Engineering
Sungkyunkwan University

Abstract

By adding user interface to the usual router, an improved functional router is implemented in this paper. The proposed router is developed based on the SA1110 processor, and the system contains 1 ethernet port, 2 PCMCIA slots, and 1 serial communication port. The Emdedded Linux is adopted as an operating system, and application programs are implemented by using QT/Embedded.

I. 서론

일반적으로 PC는 다목적 시스템인데 비해 임베디드 시스템은 흔히 내장형 시스템이라고 하며 정해진 용도에 맞추어서 최적화 되어 있다. 임베디드 시스템은 PC에서와는 다른 개발 플랫폼을 사용하기 때문에 많은 애로사항이 발생 한다. 현재의 임베디드 환경에 사용 되어지는 마이크로프로세서의 종류도 다양하며 소프트웨어 플랫폼 역시 무수히 많은 임베디드 OS가 사용된다.

본 논문에서는 기존에 개발 되어서 사용되고 있는 임베디드 OS 중에서 공개 소프트웨어인 임베디드 리눅스 시스템을 이용하여 향상된 기능의 라우터를 구현한다. 임베디드 리눅스를 사용하는 장점으로는 네트워크 호환성이 뛰어나며, 크기가 작으며, 쉽게 재구성화가 가능하다는 점이다. 본 논문에서는 구현과정에 대한 자세한 기술보다는 전체적인 작업과 사용자 인터페이스 부분에 초점을 맞추어서 기술한다.

II. 관련 연구

2.1 StrongArm

StrongArm SA1110은 다양한 제어 응용 장치를 에 사용될 수 있도록 SA-1코어를 중심으로 주변 여러 장치를 통합한 것이다. 이런 이유로 StrongArm 프로세서는 통신과 네트워크 장비에 많이 사용되며 특히 핸드헬드(hand held) 장비에 널리 사용된다. SA-1110은 범용적인 목적으로, 단일 칩 내에 MMU, read 버퍼, write 버퍼, 미니캐시, 8K 바이트 write-back 데이터 캐시 및 16K 바이트의 명령 캐시를 가진 32비트 RISC 형 마이크로프로세서이다. SA-1110은 ARM V4 아키텍쳐 프로세서 패밀리와 소프트웨어적으로 호환이 되며, ARM을 지원하는 칩들, 예를 들어 I/O, 메모리 및 비디오 칩들을 사용 가능하다[1].

2.2 리눅스

리눅스는 윈도우에서 동작하는 응용프로그램이 아닌 독자적인 32비트 또는 64비트의 운영체제 프로그램으로 인텔계열 CPU와 알파, 스팍, PPC 등 다양한 아키텍쳐에서 잘 작동하는 운영체제이며 특히 강력한 네트워크 기능을 가지고 있다[2].

리눅스는 초기에 PC나 서버급 시스템에 포팅이 되어 사용되다가 최근에는 임베디드 시스템으로 그 관심사가 높아짐에 따라, 임베디드 시스템에 리눅스를 포팅하여 실제 제품으로 선보이고 있다.

2.3 무선랜(IEEE802.11)

무선랜은(WLAN)은 기존의 유선랜(LAN)의 허브(hub)에서 클라이언트(client)까지 유선 대신 전파나 빛을 이용하여 네트웍을 구축하는 방식으로서, 일반적으로 30~150m 거리 내에서 무선으로 1-54Mbps 속도로 데이터를 고속으로 전송하는 네트웍을 가르킨다. 전세계적으로 인정된 비 허가 주파수 대역인 적외선(IR: Infrared), 2.4GHz ISM(Industrial. Scientific and Medical) Band와 5GHz UNII(Unlicensed National Information Infrastructure) Band를 사용한다.

무선랜(WLAN)의 구성요소는 기존의 NIC역할을 하는 무선카드, 무선 환경에서 LAN 허브 역할을 하는 AP(Access Point) 그리고 분산된 무선랜(WLAN) 세그먼트(segment)를 연결시켜주는 무선 브리지(wireless bridge)로 구성되어 있다. 그리고 현재 무선랜(WLAN) 기술에서 IEEE 802.11a, 2.4GHz 대역에 11Mbps의 전송속도를 갖으며 CCK(Complementary Code Keying) 변조와 기존 802.11 DSSS(Direct Sequence Spread Spectrum) 방식을 사용하는 장비와 하위 호환성을 유지하는 802.11b, 그리고 2.2GHz 대역에 22Mbps의 전송속도를 제공하며 802.11b와 호환성을 갖는 802.11g가 있다. 이는 1999년9월 IEEE 802.11b 고속 물리계층 표준발표 및 Wi-Fi인증부여로 시장을 주도하기 시작 하였다. IEEE802.11b Wi-Fi가 사실상 업계 표준으로 등장하면서 경쟁심화에 따른 포트 당 단가의 저렴화와 노트북 사용자의 이동성에 대한 욕구 증대 그리고 새로운 무선 네트워크를 이용한 새로운 비즈니스 모델의 출현 등으로 시장이 급격히 성장하고 있다.

2.4 QT/Embedded

마이크로 소프트윈도우, 애플 맥 OS, X에서 동작하는 교차 플랫폼지원 라이브러리로 명성을 높인 트롤테크가 임베디드를 목표로 만든 버전이 QT/Embedded 이다. 여러 가지로 좋은 특성으로 인해 Gtk+와 더불어 특히 유닉스 개발자들 사이에 널리 퍼져 있으며, 최근에는 임베디드 개발자들에게 급속도로 침투하고 있다. QT/Embedded는 하부에 X 없이도 프레임 버퍼를 직접 제어하는 방식으로 QT/X11 라이브러리가 지원하는 기능을 그대로 사용 할 수 있으며, 동일한 리눅스 환경이라면 원시 코드 레벨에서 다시 컴파일 하는 작업으로 응용프로그램 이식을 쉽게 끝낼 수 있다. QT/Embedded가 채택한 라이센스 모델은 GPL(GNU General Public License)과 상용 라이센스이므로, 사용자가 요구에 맞춰 선택해 사용할 수 있다. 개발 시점에서 GPL(GNU General Public License)을 따른 공개 버전을 사용할 수 있지만, 제품을 양산할 경우에는 개발자 라이센스 비용을 물어야 한다는 점에서 다른 윈도우 시스템에 비해 불리하다. 하지만 그 대신 유료 기술지원을 받을 수 있다[3].

III. 시스템 개발 환경

3.1 교차 개발 환경 구축

일반적으로 임베디드 시스템을 개발하기 위해서는 시스템의 특성상 큰 용량의 저장장치를 갖지 못하기 때문에 교차 개발환경을 구축 하여 시스템을 개발한다. 본 논문에서는 개발 시스템 적합한 ARM용 크로스 컴파일러를 사용한다[4].

표 1. 교차 개발 환경 도구 모음

Cross compiler	
binutils-arm-2.9.5.0.37	bin과 object 파일 만드는데 사용
gcc-arm-2.95.2	GNU c 컴파일러
cpp-arm-2.95.2	Macro substitution 사용하기 위함
g++-arm-2.95.2	GNU c++ 컴파일러
libc6-dev-arm-2.95.2	standard library
libstdc++2.10-arm-2.95.2	추가적인 run time library
libcdc++2.10-dev-arm-2.95.2	c++프로그램을 위한 headers and static library files

크로스 컴파일러란 동일한 환경에서 동작되는 컴파일러와 이 컴파일러에서 생성된 실행파일을 동일한 환경에서 수행한다면 이때의 컴파일러를 네이티브(native) 컴파일러라고 하고, 이와 반대로 컴파일러가 동작하는 시스템과 컴파일러에 의하여 생성된 실행파일이 동작하는 시스템이 다를 때 이 컴파일러를 크로스(cross)컴파일러라 한다. PC와 같은 구조를 갖는 임베디드 리눅스 시스템을 만든다면 PC의 네이티브 컴파일러에서 만들어진 실행파일이 그대로 실행되기 때문에 크로스

컴파일러의 설치가 필요 없지만, 임베디드 시스템 대부분이 저렴한 가격대를 요구하거나 또는 특수한 기능을 수행하는 구조를 가지고 있기 때문에 이러한 경우는 아주 드문 경우라고 할 수 있다.

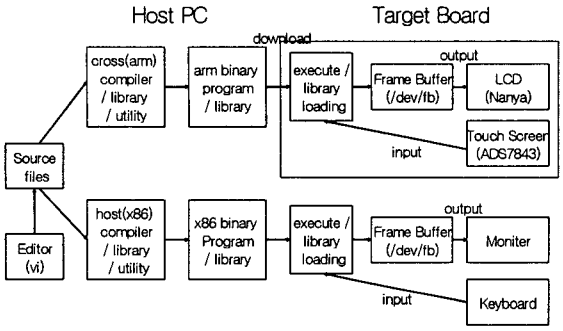

그림 1. 크로스 컴파일러의 구조

그럼에도 불구하구 개발 프로세서를 빠르게 진행하기 위해서 익숙한 PC구조를 사용하는 경우가 많아지는 추세이다.

3.2 애플리케이션

임베디드 리눅스에서 동작하는 애플리케이션의 개발은 QT를 사용하며 Trolltech (www.trolltech.com) 홈페이지에서 다운 받을 수 있다. 다운받은 파일 QT/X-11과 QT/Embedded를 설치한다. Qt/Embedded는 X-Window 없이 Linux 커널이 제공하는 프래임 버퍼(Frame Buffer)를 이용하는 방식을 사용한다. 애플리케이션 작성을 하고나면 먼저 호스트 컴퓨터(x86)에서 동작을 체크 하고, QT/Embedded를 Arm 용으로 컴파일 하여 타겟 시스템에 전송하고 테스트 한다.

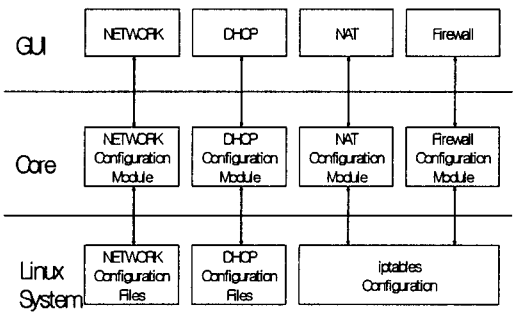

그림 2. 애플리케이션 구조

3.3 커널 수정

커널을 설치한 후에 하드웨어 관련 부분을 수정하여야 한다. 수정을 가한 부분은 arch, include/asm, driver 디렉토리 내에 파일들이다. 커널소스 구성시에는 불필요한 파일들을 포함하지 않도록 타겟 시스템의 기능만을 지원하는 부분을 선별적으로 선택하여 커널 이미지를 작성해야 한다.

3.4 파일 시스템

3.4.1 Cramfs

Cramfs(Cramf a filesystem onto a small ROM)는 간단한 작은 파일 시스템으로 구현 되어져 있으며, 일반적으로 ROM에 구현되는 파일 시스템으로 읽기 전용(Read-Only)이다. 특징은 zlib Routines을 사용하기 때문에 압축이 되어 공간을 절약 할 수 있고 읽기 전용이기 때문에 안전하게 데이터를 보존할 수 있다. 그리고 Cramfs를 ROOT filesystem으로 사용하면 Ramdisk가 차지하던 RAM의 크기를 줄일 수 있다.

3.4.2 Jffs2 (Journalling Flash File System)

Jffs2는 Jffs를기반으로 레드햇에서 개발한 임베디드 시스템을 위한 신형 저널링 파일 시스템이다. 리눅스 커널 2.0에서 동작하는 Jffs와는 달리 리눅스 커널 2.4에서 동작한다. 취약한 가비지 컬렉션 문제를 해결하기 위해 가비지 컬렉션 스레드를 따로 둬 부분적으로 사용하는 블록을 다른 곳으로 옮겨 유효한 블록을 확보하는 기능을 탑재 하고 있으며, 갑작스런 전원공급 차단에 대비하기 위해 쓰는 작업을 완벽하게 성고하기 직전까지는 어떤 파일도 건드리지 않는 방법을 사용하고 있다. 이런 가비지 컬렉션을 하는 도중에 연속적으로 읽고 쓰는 과정을 통해 플래쉬 메모리 한 곳에 몰린 파일을 여러 곳으로 분산시킴으로써 닳기 균등화(wear-leveling)상태를 유지한다. 또한 실행 중에 zlib을 사용한 데이터 압축기능과 속이 빈파일(space file) 저장 기능을 가지므로 작은 기억장치를 탑재한 임베디드 시스템에 최적화되어 있다고 볼 수 있다.

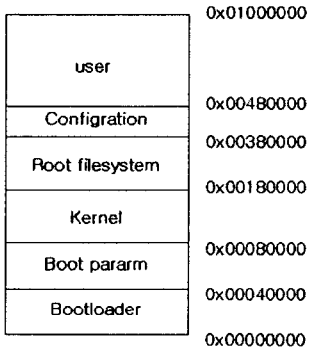

그림 3. 16Mb Flash ROM의 Memory Map

Ⅳ. 구현

구현에 사용된 시스템은 StrongArm 계열의 SA1110 MCU, 16Mb flash Rom, 32Mb SDRAM, PCMCIAI,II, Touch Screen 등으로 구성되어 있다.

작성된 애플리케이션 프로그램을 ARM용으로 컴파일 후 파일시스템 이미지를 작성하고 작성된 이미지와 수정된 커널 이미지를 시스템에 적재한다. 시스템의 동작은 호스트에서 클라이언트 시스템 간에는 무선으로 연결이 되며 호스트 시스템에서 동적으로 클라이언트 시스템에게 IP주소를 할당하는 방법과 터치스크린을 이용하여 할당하는 방법이 있으며, 직접적으로 LCD를 통하여 IP 주소 할당 사항 등을 모니터링 할 수 있다.

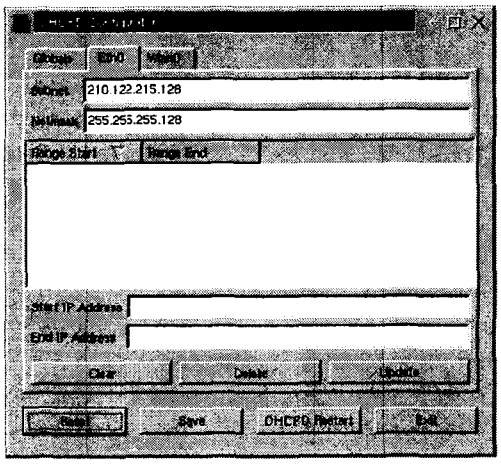

그림 4. 구현된 애플리케이션

```
[root@root pcmcia]$ping www.yahoo.com
PING www.yahoo.akadns.net (66.218.71.87): 56 data bytes
64 bytes from 66.218.71.87: icmp_seq=0 ttl=54 time=316.8 ms
64 bytes from 66.218.71.87: icmp_seq=1 ttl=54 time=462.5 ms
64 bytes from 66.218.71.87: icmp_seq=2 ttl=54 time=471.5 ms
--- www.yahoo.akadns.net ping statistics ---
4 packets transmitted, 3 packets received, 25% packet loss
round-trip min/avg/max = 316.8/416.9/471.5 ms
```

그림 5. 클라이언트에서 외부로 연결되는 모습

Ⅴ. 결론 및 향후 연구 방향

타겟 시스템에 임베디드 리눅스를 이식하기 위해서는 시스템의 특성과 구성 요소들의 이해가 상당히 중요하며, 이를 위해 우선적인 프로세서의 이해가 선행되어야 한다. 본 논문에서는 최근에 활발히 연구 되고 있는 임베디드 시스템과 빠르게 보급 되고 있는 무선랜을 이용하여, 기존에 개발되어 사용 되고 있는 라우터를 임베디드 운영체제의 한 종류인 임베디드 리눅스를 이용하여 구현하였다. 특히 LCD, Touch Screen, QT 애플리케이션을 이용하여 사용자 인터페이스 부분의 구현에 목적을 두었다.

참고문헌

[1] Intel Corporations, Intel StrongArm SA-1110 Microprocessor Develper's Manual, June, 2000.

[2] 박재호, IT Expert 임베디드 리눅스, 한빛 미디어, 2002.

[3] www.trolltech.com

[4] 마이크 루키디스, 앤디 오람 저, 이기동 역 GNU 소프트웨어로 프로그래밍 하기, 한빛미디어, 2000.

[5] 이연조, 임베디드 리눅스 프로그램밍, PC book, 2002.

[6] 주민규 외, "내장형 리눅스를 이용한 라우터의 설계 및 구현", 정보처리학회논문지, 제8-A권, 제4호, pp. 339-344, 2001.

[7] www.kelp.or.kr

[8] www.kesl.org

[9] www.falinux.com