논문 2007-02-28

임베디드 리눅스 기반의 전자 칠판 시스템 개발

(Development of Electronic White-board Based on Embedded Linux)

서 창 준*
(Chang-Jun Seo)

Abstract : Recently, most embedded systems have the multi-functions mixed the hardware with the software. The existing sequence programming methods are not suitable to implement the embedded system with multi-functions. So, it can be overcome the limit of a facility implementation by introducing the operating system in system. Also, due to the requirement about the better convenient and comfortable meeting or lecture environment, the necessity of electronic white-board is getting higher. Specially, the education using multimedia information is much more desirable for various and improved lecture at the high school and the university. But the sequence program which have been managed in existing electronic white-board system has some difficulties to achieve the software-oriented systems which has to accomplish many functions. In this paper, we propose the method to implement a facility of electronic white-board through using the embedded linux with excellent performance. The embedded linux presents the powerful software environment for the implementation of an embedded system and makes the realization of many various functions easy because it follows kernel characteristics of linux. In this paper, we describe the details for the structure of hardware, kernel source and device driver of a developed electronic white-board.

Keywords : electronic white-board, embedded linux, embedded system, board level porting, device driver

I. 서 론

전자칠판 시스템이란, 칠판 화면에 기록된 내용을 용지로 즉시 출력할 수 있는 시스템을 말한다. 이는 기존 칠판이 가진 불편함, 즉 칠판에 판서되는 내용을 참석자들이 일일이 손으로 적어야 하는 어려움과 진행자의 이야기와 칠판의 내용을 동시에 습득하여야 하는 참석자들의 고충을 충분히 해결해 줄 수 있는 확실한 대안이라 하겠다. 이에 보다 편리하고 패적한 강의 및 회의 환경에 대한 요구에 의해 전자칠판은 그 필요성이 점점 더 높아지고 있으며 다양한 분야에서 사용자의 요구를 충족시키는 새로운 전자칠판 시스템의 개발이 필요하게 되었다.

기존의 전자칠판 시스템의 구성을 보면, 칠판 화면, 화면의 데이터를 스캐닝하는 CCD 센서, 화면의 데이터를 출력해 주는 전용 프린터, 그리고 외부 PC로의 데이터 전송을 위한 통신 부분 등으로 나누어져 있다. 이러한 구성 중에서 전용 프린터는 일반 용지가 아닌 전용의 감열용지를 써야하는 제약을 가지고 있다. 판서의 내용을 수정 보완하여 저장할 수 있는 기능을 가지고 있지 않는 단점이 있다. 따라서 보다 편리하고 패적한 강의 및 회의 환경을 만들기 위해 이러한 단점들을 해결한 새로운 전자칠판 시스템의 개발이 요구된다.

대부분의 전자칠판은 펌웨어 기반의 순차 프로그램 기법을 활용하여 개발된 것들이 사용되어지고 있다[1]. 전자칠판 사용자의 요구는 많아지고 고급화됨에 따라 이를 수용하는 데는 펌웨어 방식의 순

* 교신저자(Corresponding Author)
논문접수 : 2007. 12. 6., 채택확정 : 2008. 2. 16.
서창준 : 인제대학교 전자지능로봇공학과
※ 본 논문은 2006년도 인제연구장학재단 국외연수지원에 의한 연구결과임.

차 프로그램 기법은 한계를 갖는다. 이를 극복하기 위해서는 보다 고급의 개발환경 구축과 새로운 시스템이 개발되어져야 하는데 이를 위해서는 제어시스템에 운영체제의 도입이 필수적이다.

상용 운영체제를 사용할 경우 개발에 따른 라이센스 비용이 든다. 또한 전자칠판이 굳이 실시간 운영체제를 사용하지 않아도 되므로 안정성이 보장된 공개용 O/S를 사용함으로 인해 개발 비용과 시간을 단축시킬 수 있다.

본 논문에서는 공개용 임베디드 리눅스를 전자칠판의 운영체제로 이용하여 전자칠판 시스템을 개발하고자 한다. 리눅스 커널 자체가 태생적으로 모듈 형식의 코드로 이루어져 있어 쉽게 최소, 최적화가 가능하며, 오픈소스이면서도 개발을 위한 자체 툴과 라이브러리를 가지고 있어서 상용 운영체제에 못지않은 성능과 개발의 편의성을 지니고 있으므로 전자칠판 시스템의 구성과 성능향상에 적합한 선택이다. 임베디드 리눅스는 임베디드 시스템의 다양한 분야에서 이용되어 좋은 결과를 보이고 있다[2-5].

2장에서는 전자칠판 시스템의 구성을 살피고, 3장에서는 시스템을 구현하기 위해 커널 포팅 및 디바이스 드라이버를 작성에 대해 논하고, 4장에서는 개발된 시스템의 결과물을 살펴본 후, 마지막으로 결론 및 문제점을 서술한다.

II. 전자칠판 시스템의 구성

1. 시스템 개요

그림 1은 전자칠판 시스템의 구성을 블록도로 나타내고 있다. 우선 사용자는 칠판에 기록을 마친 후 조작부에 버튼을 눌러 칠판 내용을 저장하거나 외부로 획득을 시도한다. 조작부를 통하여 사용자 입력을 받으면 회전부는 칠판화면을 좌, 우로 회전시키고, 스캐닝부에서는 CCD 이미지 센서를 이용하여 칠판화면의 내용을 취득하고, 보조 기억장치로 스캔된 이미지를 저장하거나 출력부를 통하여 프린트하게 된다. 또한 RS-232C를 통해 외부 PC로 취득된 자료를 전송한다. 이러한 모든 동작들은 단일 보드컴퓨터로 구성되는 제어부에서 제어하고, 관리한다.

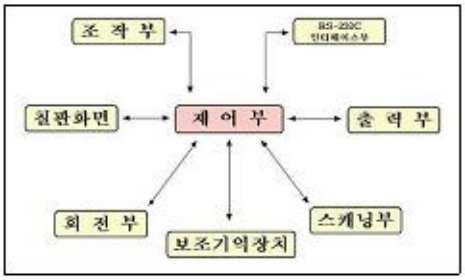

그림 1. 전자칠판 시스템 구성도
Fig. 1. Block diagram of electronic white-board system

그림 2는 시스템의 개발환경을 나타낸다. 개발 시스템은 호스트 시스템과 타겟 시스템으로 구성되는데 타겟 시스템은 단일보드컴퓨터와 스위치, 센서, 모터 등으로부터 입력되는 데이터를 처리하여 전송하는 입출력 인터페이스들(그림에서 영상처리부)로 구성된다. 자세한 내용은 다음과 같다.

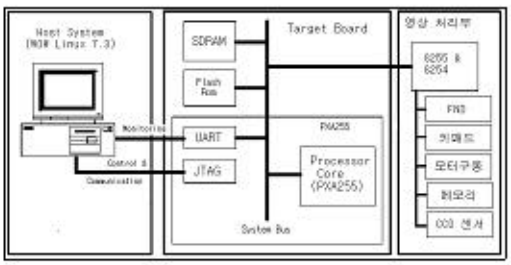

그림 2. 시스템 개발환경 구성도
Fig. 2. Block diagram for system development environment

2. 호스트 시스템(Host System)

호스트 시스템에는 교차 개발 플랫폼이 구성되어 타겟 시스템을 위한 바이너리 생성이 이루어진다.

3. 타겟 시스템(Target System)

프로세서는 Intel PXA255이고, 32bit 메모리 데이터 버스와 주변장치들이 통합된 17x17mm 크기의 256핀 PBGA 패키지를 타겟 보드로 사용하였다.

3.1 롬/플래시(ROM/FLASH)

롬/플래시는 Intel strata flash 16Mbyte을 사용하였다. 플래시는 루트 영역과 커널 이미지, 응용

소프트웨어 및 시스템 자원을 저장하는 매체로 사용된다.

3.2 SDRAM

그림 3처럼 SDRAM은 2Mbit x 4 Bank x 16bit SDRAM 2개를 데이터 버스에 연결하여 32MB를 구성하였다. 주 메모리 영역은 캐쉬와 시스템이 부팅되고 난 후에 리눅스가 상주하게 된다.

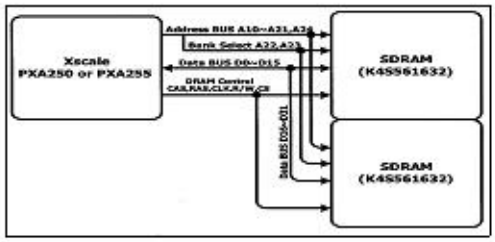

그림 3. SDRAM 메모리 인터페이스
Fig. 3. Interface of SDRAM memory

3.3 UART(Serial Port, 콘솔 인터페이스)

시스템 버스에 연결된 UART는 RS-232C를 사용하여 호스트와 통신을 한다. 이 콘솔 포트를 통해 전자칠판의 설정과 이미지들을 플래시에 저장시킬 수 있고, 하드웨어와 리눅스의 모든 과정을 모니터링 한다.

3.4 이미지 처리부

이미지 처리부는 칠판 스크린의 내용을 취득하기 위해 필요한 각종 주변 회로를 나타내는 총괄적인 명칭으로, 센서부, 회전부, 조작부, RS-232C 인터페이스부의 구성으로 이루어져 있다. 각 부분의 내용을 살펴보면 다음과 같다.

- CCD 이미지 센서(CCD Image Sensor)

칠판 스크린의 내용을 취득하는 부분이다. CCD 센서의 구동을 위하여 그림 4 와 같은 입력신호들이 필요하며, 타이머 8254(PPT)를 사용하여 설계하였다. 또한 CCD 센서의 출력 값은 판서된 내용의 흑백 데이터만 인지할 수 있도록 그림 5와 같이 기준 전압을 설정하여 0과 1의 데이터만 분리한다.

- RS-232C 인터페이스

콘솔용으로 FFUART를 사용하고, 적외선통신(IrDA)으로 STUART가 사용되며, 호스트 시스템과의 데이터 통신을 위해 Bluetooth UART를 사용하였다.

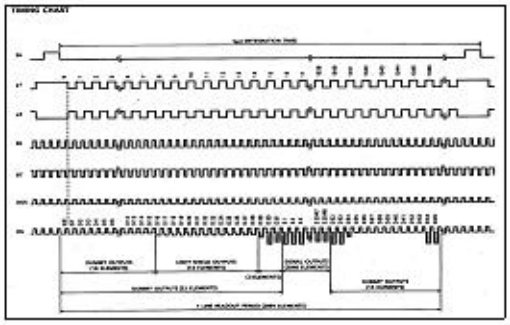

그림 4. CCD 센서 구동 펄스
Fig. 4. Driving pulse of CCD sensor

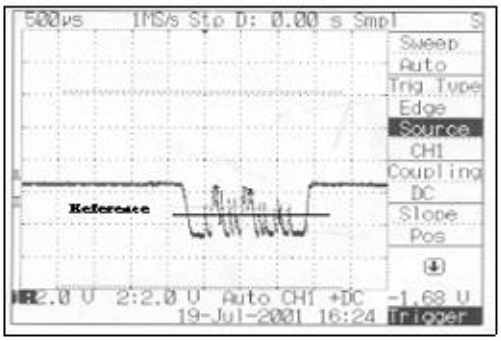

그림 5. CCD 센서 출력파형
Fig. 5. Output signal of CCD sensor

Ⅲ. 전자칠판 시스템의 구현

1. 시스템의 제원 및 개발환경

전자칠판을 제어하기 위해 개발된 시스템의 하드웨어 제원은 표 1과 같다

표 1. 하드웨어 구성요소
Table 1. Hardware components

Processor	Intel PXA255 400MHz
SDRAM	Samsung 32Mbyte
FLASH	INTEL strata flash 16Mbyte
Ethernet	CS8900A 10BaseT
Serial	3 Port
JTAG	1 Port
Connector	PC104+ connector

소프트웨어 개발 환경은 표 2와 같다

표 2. 소프트웨어 개발 환경
Table 2. Software development environment

타겟 시스템	리눅스 커널 2.4.18
파일 시스템	JFFS2, 램디스크
디바이스 드라이버	FND, 스테핑 모터, 키패드 8255, 8254
컴파일러	ARM 크로스 컴파일러 (GNU gcc compilers for C, C++)
호스트 시스템	와우 리눅스 7.3

2. 임베디드 리눅스 커널 포팅

타겟 보드에 사용된 임베디드 리눅스는 일반 리눅스 커널에 ARM과 PXA255 프로세서 패치를 적용시킨 후, 보드 레벨 포팅을 한 커널이다.

2.1 리눅스 커널 패치

타겟 보드에 사용된 임베디드 리눅스는 그림 6과 같이 일반 리눅스 커널 2.4.18에 ARM 패치인 2.4.18-rmk7과 PXA255 프로세서 패치인 2.4.18-rmk7-pxa1을 적용시킨 후, 보드 레벨 포팅을 한다.

```
linux/arch/arm/mach-pxa/xhyper255.c
static struct map_desc xhyper255_io_desc[] __initdata = {
/* virtual physical length domain r w c b */
#ifdef CONFIG_ARCH_XHYPER255A
{ 0xE8000000, 0x00000000, 0x01000000, DOMAIN_IO, 0, 1, 0, 0 }, // CS0: Intel Strata Flash 16M
#endif
{ 0xF0000000, 0x04000000, 0x00100000, DOMAIN_IO, 0, 1, 0, 0 }, // CS1 : CS8900A
#ifdef CONFIG_ARCH_XHYPER255A
{ 0xF0000000, 0x04700000, 0x00010000, DOMAIN_IO, 0, 1, 0, 0 }, // CS1 : 255A ADS7843/LCD
{ 0xF1000000, 0x14000000, 0x00800000, DOMAIN_IO, 1, 1, 0, 0 }, // CS5 : 8255 CHIP SELECT
{ 0xF1800000, 0x14800000, 0x00800000, DOMAIN_IO, 1, 1, 0, 0 }, // CS5 : 8254 CHIP SELECT
{ 0xF2000000, 0x15000000, 0x00800000, DOMAIN_IO, 1, 1, 0, 0 }, // CS5 : FND
{ 0xF2800000, 0x15800000, 0x00800000, DOMAIN_IO, 1, 1, 0, 0 }, // CS5 : STPPER MOTOR
{ 0xF3000000, 0x16000000, 0x00800000, DOMAIN_IO, 1, 1, 0, 0 }, // CS5 : KEY PAD
#endif LAST_DESC };
```

그림 6. 패치된 hyper255.c 소스 파일
Fig. 6. Patched source file hyper255.c

그림 7은 PXA255 프로세서의 가상주소를 물리주소로 대응시키는 과정을 보여준다. 메모리관리장치(MMU)와 정적 메모리 컨트롤러를 통해 각 디바이스의 신호가 전달된다. 리눅스 디렉토리의 내부 arch/arm/mach-pxa의 xhyper255.c에서 물리주소와 가상주소를 대응시킨다.

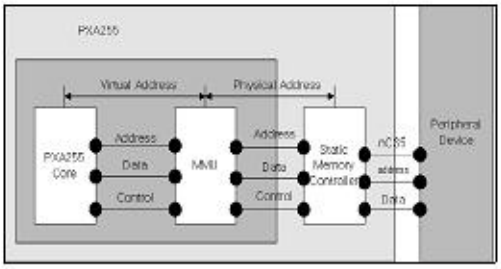

그림 7. 가상메모리 설정 블록 다이어그램
Fig. 7. Block diagram for assignment of virtual memory

2.2 커널 컴파일

환경설정과 소스코드의 의존관계를 설정한 후 컴파일러를 이용하여 커널 이미지를 생성하도록 커널을 컴파일 한다.

커널 이미지가 생성되면 tftp를 이용하여 타겟 보드에 포팅한다. 그림 8은 정상적으로 커널이 타겟 보드에 올려진 후 부팅된 화면이다.

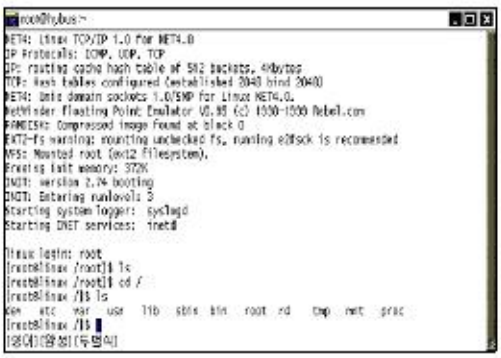

그림 8. 리눅스 커널의 부팅
Fig. 8. Booting of linux kernel

3. 디바이스 드라이버

본 논문에서 사용한 디바이스는 모두 문자 디바이스에 해당된다. 리눅스가 드라이버를 사용하기 전에 문자 디바이스는 register_chrdrv() 함수를 사용하여 등록되어져야 한다.

디바이스는 파일 구조체에 의해 내부적으로 구별되는데 커널은 드라이버의 함수를 수행하기 위해서 file_operation 구조체를 사용한다.

3.1 8255, 8254 디바이스 드라이버

병렬인터페이스 8255(PPI) 디바이스는 타겟 보

드의 PC104 커넥터에서 나오는 어드레스 신호와 데이터 신호를 확장하는 칩으로서 칠판 스크린의 데이터가 저장된 메모리로부터 데이터와 칠판 스크린의 회전 동작을 관측하는 센서로부터 얻은 데이터를 읽어 들이는 기능과 형광등의 ON/OFF를 처리하는 기능을 수행한다.

타이머 8254(PIT) 디바이스는 CCD 이미지 센서 구동펄스를 생성한다. 4MHz의 오실레이터에서 주파수를 분주하여 CCD 이미지 센서의 입력신호 Ø1, Ø2에 해당하는 2us의 신호와 출력신호 RS(reset), BT(boost) 신호를 만들기 위해 필요한 1us의 펄스를 생성한다. 그림 9는 PIT의 입력신호와 출력신호를 나타낸다.

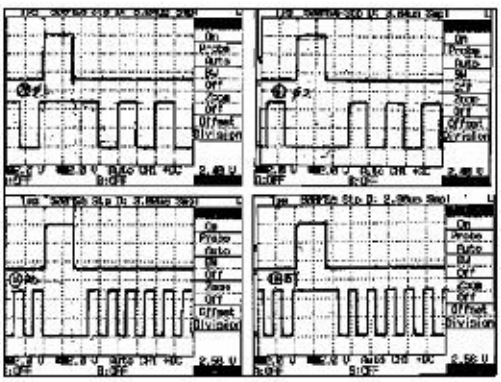

그림 9. 입력신호 Ø1, Ø2와 출력신호 RS, BT
Fig. 9. Input signal Ø1, Ø2 and output signal RS, BT

3.2 그 외의 디바이스 드라이버

그 외의 디바이스 드라이버는 다른 디바이스 드라이버의 동작원리와 같으며, 가상주소와 물리주소를 대응시킴으로서 구동된다.

표 3은 모터, 키패드, 7-세그먼트(FND)의 드라이버에 대한 물리주소와 가상주소이다.

표 3. 기타 디바이스 드라이버 정보
Table 3. Information for other device drivers

디바이스 드라이버	물리 주소	가상 주소
Stepping Motor	0xF2800000	0x15000000
Key Pad	0xF3000000	0x15800000
FND	0xF2000000	0x14800000

Ⅳ. 결 과

CCD 이미지 센서의 구동을 통해 칠판의 이미지 데이터를 취득하는 스케닝부, 이미지 저장을 위한 메모리부(보조기억장치), 칠판 스크린의 구동을 위한 회전부, 동작 상태를 나타내는 출력부(표시부)를 구현하였다.

그림 10은 전자칠판의 외형이다. 오른쪽 상단에 칠판의 동작을 조작하는 키패드와 FND로 구성된 조작부가 있음을 확인할 수 있다. 그림 11은 개발된 전자칠판의 메인보드의 모습이다. 외부 메모리와 주변회로가 설계되어 있으며 좌측에는 키패드, FND의 신호를 내어주는 케이블과 하측에 CCD 이미지 센서로부터 신호를 받는 케이블이 연결되어 있다. 그림 12는 칠판 뒷면에 부착되어 있는 CCD 이미지 센서를 나타내는데 이를 이용하여 칠판 이미지를 취득하였으며, 그에 따른 출력을 검증하였다.

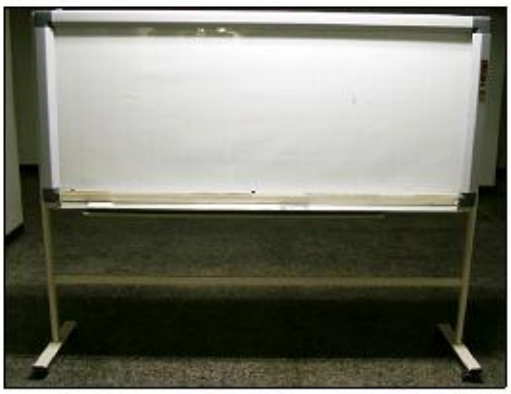

그림 10 전자칠판 외형
Fig. 10. External form of electronic white-board

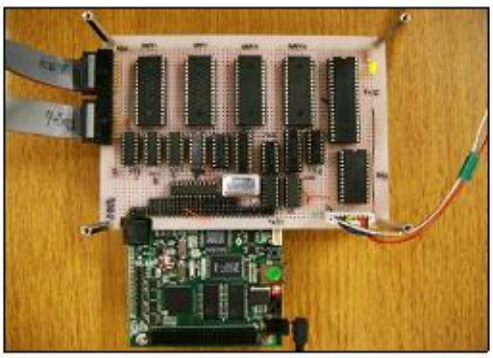

그림 11. 전자칠판 메인제어보드
Fig. 11. Main control board of electronic white-board

스크린 회전부는 실험을 통하여 정확한 동작여부를 검증하였고 BTUART를 이용하여 호스트 PC로의 데이터 전송도 확인할 수 있었다.

그림 12. CCD 센서
Fig. 12. CCD sensor

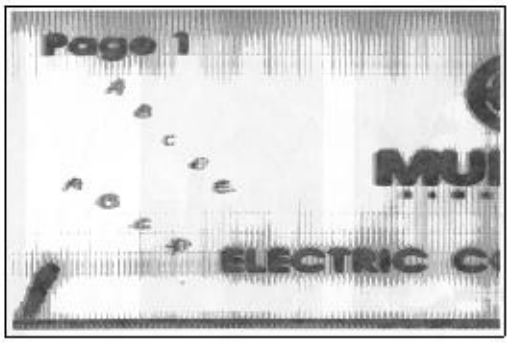

그림 13. 칠판으로부터 획득한 스크린 이미지
Fig. 13. Screen image obtaining from white-board

이 모든 일련의 실험을 통해 제어기가 정확하게 동작한다는 것을 확인할 수 있었다. 그림 13은 개발 시스템의 동작 결과를 보여주는 예로서 개발된 보드로부터 획득한 전자칠판의 스크린 이미지를 나타낸다.

V. 결 론

본 논문에서는 임베디드 리눅스 기반의 전자칠판을 개발하였다. 이를 위해 리눅스 2.4.18의 보드레벨 포팅을 하였으며, 전자칠판 기능 구현을 위한 디바이스 드라이버를 작성하였고, 칠판 스크린 이미지 전송프로그램을 작성하였다. 임베디드 리눅스를 전자칠판 시스템에 도입함으로써 칠판에 요구되는 다양한 기능을 보다 체계적이면서 용이하게 구현할 수 있었고, 향후 사용자의 여러 가지 요구에 대해 능동적이면서 어려움 없이 대처할 수 있는 장점을 갖는다. 그러나 획득한 이미지의 개선이 필요하며, 보다 편리한 사용을 위해 이동식 저장장치를 위한 디바이스 드라이버를 개발할 필요가 있으며 GPIO를 통한 프린터 포트 지원을 위한 연구가 진행되어야 한다.

참고문헌

[1] 이정민, 서창준, "임베디드 PC 기반 전자칠판 제어 시스템 개발", ICCAS, 2001
[2] 조덕현, 최병욱, "임베디드 리눅스를 이용한 산업용 인버터의 웹기반 원격관리", 제어 및 자동화 시스템공학회, 제9권 4호, 340-346쪽, 2003.
[3] 최병욱, 신은철, 이수영, "임베디드 리눅스를 이용한 웹기반 빌딩 자동화시스템", 제어 및 자동화 시스템공학회, 제10권 4호, 335-341쪽, 2004.
[4] 최병욱, 김현기, 신은철, "임베디드 리눅스 기반의 다중프로토콜 제어기 개발 및 빌딩 자동화시스템과의 연동 적용", 제어 및 자동화 시스템공학회, 제10권 5호, 428-433쪽, 2004.
[5] 신은철, 최병욱, "실시간 임베디드 리눅스를 이용한 이동로봇 플랫폼 구현", 제어 및 자동화 시스템공학회, 제12권 2호, 194-200쪽, 2006.
[6] Karim Yaghmour, Building Embedded Linux Systems, O'Reilly, 2004
[7] David E. Simon, An Embedded Software Primer, Addison-Wesley, 1999
[8] Matt Welsh, Matthias Kalle Dalheimer, Terry Dawson, Lar Kaufman, Running Linux, O'Reilly, 2004
[9] Craig hollabaugh, Embedded Linux : Hardware, Software, and Interfacing, Addison-Wesley, 2002
[10] Micheal F.Hordeski, Control system interface design and implementation using the personal computers, Prentice-Hall, 1992
[11] 우상철, 박철, Embedded Linux System 구조 및 설계 응용, Ohm사, 2003
[12] 문동섭 역, 임베디드 커뮤니케이션 소프트웨어 이해와 설계, 에이콘출판사, 2004
[13] 박영환, 임베디드 시스템 임베디드 리눅스, 사이텍미디어, 2002
[14] 박재호, 임베디드 리눅스, 한빛 미디어, 2002

저 자 소 개

서 창 준(Seo, Chang-Jun)

1989년 2월 : 경북대학교 전자공학과 학사
1991년 2월 : 한국과학기술원 전기 및 전자공학과 석사
1996년 2월 : 한국과학기술원 전기 및 전자공학과 박사
1996년~현재 : 인제대학교 전자지능로봇공학과 부교수
관심분야 : 임베디드 시스템, 실시간 운영체제, 제어시스템, 지능로봇
Email : elecscj@inje.ac.kr