韓國 컴퓨터情報學會 論文誌
第11卷 第4號, 2006. 9.

2006-11-4-4-1

임베디드 리눅스를 이용한 위치기반 관광안내 단말기의 설계 및 구현

김 정 원*, 전 봉 기*

A Design and Implementation of LBS-based Tour Guide System using Embedded Linux

Jeongwon Kim*, Bonggi Jun*

요 약

본 논문에서는 임베디드 리눅스를 이용하여 위치기반 관광정보안내 단말기를 설계 및 구현하였다. 구현된 단말기는 GPS 인터페이스를 통해 단말의 현재 위치를 파악하고 해당 관광지의 정보를 Qt-embedded 윈도우상에서 디스플레이한다. 논문의 단말기는 타겟보드상에서 부트로더와 임베디드 리눅스 커널 및 각종 디바이스 드라이버가 구현되었으며 GUI로는 임베디드 윈도 매니저인 Qtopia를 이용하여 애플리케이션을 개발하였다. 구현 결과 무료 운영체제를 이용한 모바일 관광안내 단말기로서 기능을 충분히 수행할 수 있음을 확인하였고, 향후 연구과제로는 통신 기능을 추가하여 단말기의 관광 정보를 실시간에 업데이트할 수 있는 기능을 구현하는 것이다.

Abstract

In this paper, we has implemented a tour guide terminal using the embedded linux. The implemented terminal can locate its position using GPS interface and display the sightseeing information of the current location. With view of this implementation results, we confirmed the possibility of embedded linux terminal as a tour guide PDA. The realtime update of sightseeing information using a mobile communication functionality will be included in our future works.

▸ Keyword : 임베디드 리눅스(Embedded Linux), 위치기반시스템(LBS), 큐티 임베디드(Qt/Embedded)

• 제1저자 : 김정원
• 접수일 : 2006.07.28, 심사일 : 2006.08.17, 심사완료일 : 2006.09.20
* 신라대학교 컴퓨터정보공학부 조교수

Ⅰ. 서 론

최근 PDA, 핸드폰, GPS, 무선통신의 발달로 이동통신과 무선단말기를 결합한 지도 검색과 위치 파악 및 추적서비스 분야가 새로운 이머징마켓으로 등장하고 있고, 이를 실현하기 위한 무선단말기 개발과 위치기반서비스(LBS: Location-Based Service) 기술 개발이 새로운 이머징 기술로 등장하고 있다.

또한, 내장형 리눅스는 포스트 PC 시대에서 최적의 운영체제가 될 것으로 기대되면서 다수의 회사 및 연구기관에 의해 개발 및 구현되고 있다[1]. 서버 시장에서 리눅스는 비교적 성공적인 시장 진입을 하고 있지만 전자수첩, 무선 전화기 등 단순 단말기와 PDA, 스마트 폰 등의 정보 단말기의 중요한 구분 요인이 되는 소프트웨어의 유연성, 즉 새로운 응용 프로그램을 수행할 수 있는 능력을 가지고 있지만 작은 이동형 단말기에 필요한 모든 기능을 수용할 수 있는 운영체제인가 라는 질문에 아직은 부족한 점이 많은 것이 사실이다[12]. 즉, 리눅스는 포스트 PC 단말기와 같은 내장형 시스템에 적합한 운영 체제로 인식되고 있지만 이동형 단말기의 커널로서 가져야 할 특성을 완벽하게 만족시키지는 못하고 있다. 이러한 요구사항에는 먼저 커널의 측면에서 전력 관리 기능, 메모리 관리 기능, 플래쉬 파일 시스템, XIP(Execute In Place), 각종 장치 드라이버 지원 등이 있다[2]. 미들웨어 측면에서는 GUI, 보안 관련 솔루션, 원격 데이터베이스 참조 솔루션, 싱크 관련 솔루션 등이 있고 응용 프로그래밍 측면에서는 기존의 마이크로소프트웨어사 문서와의 호환성을 가지는 응용 프로그램들, 다양한 언어 지원, 멀티미디어 플레이어, 각종 플러그 인을 지원하는 브라우저, 개인 정보 관리, 게임들이 있다.

한편, 위치기반서비스(LBS)는 GPS 칩을 내장한 휴대폰이나 PDA 단말기를 휴대한 사용자가 자신의 위치를 기반으로 다양한 응용 서비스를 제공받을 수 있다. 현재 위치기반서비스를 위한 국내 여건은 GPS수신기, 휴대폰이나 PDA, 이동통신사의 무선통신 기술, 상세한 전자지도의 CP(Contents Provider)가 모두 충족되어 있어 LBS를 위한 시장성과 필요한 기술 요소를 모두 충족하고 있다[4,5]. 현재 위치기반서비스를 위한 요소기술들을 모두 충족하고 있지만, 위치기반서비스는 물류 택배와 같은 큰 규모의 사업장에서만 개발되어 사용되고 있다. 이러한 이유는 PDA의 가격이 비싸고, 위치기반서비스 개발 기술이 보편화되지 않았기 때문이다[6,7,8]. 논문에서는 고가의 외국산 PDA를 사용하지 않고, 무료 운영체제인 리눅스를 탑재한 위치기반서비스를 위한 전용 모바일 단말기를 개발하고, 또한 국내 기술로 PDA를 위한 위치기반서비스 엔진을 개발하였다.

본 논문에서는 임베디드 리눅스를 탑재하여 위치기반 관광 안내서비스를 할 수 있는 단말기를 개발하고 Qt[9] 기반의 애플리케이션을 개발하였다. 논문의 구성은 다음과 같다. 2장에서는 구현된 단말기를 소개하고, 3장에서는 임베디드 리눅스 커널, 부트로더, USB, 키패드등 대표적인 디바이스 드라이버 구현결과를 소개하며, 4장에서는 클라이언트 프로그램의 구현 내용을 기술하며 5장에서 결론을 제시한다.

Ⅱ. 단말기의 설계 및 구현

표 1. 주요 사양
Table 1. Specification

디바이스	주요 사양
CPU	Intel PXA255(400MHz)
메모리	Flash 64MB, RAM 128MB
디스플레이	TFT LCD 3.5"
통신장치	Ethernet 2port, Serial, Bluetooth, PCMCIA, USB 2.0
저장장치	CF, HDD
입력장치	10 Keypad, Touch

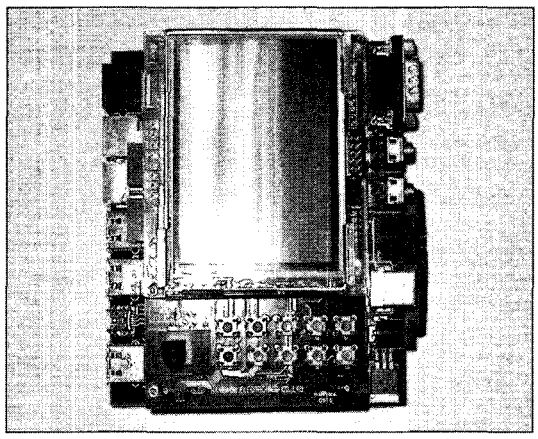

그림 1. 개발된 타겟보드
Fig 1. Target board

그림 1은 개발된 타겟보드이다. 초기 모델은 하나의 보드에 각종 디바이스를 통합한 시험 모델이었으나 최종 개발된 프로토타입은 스택형 방식으로 각종 디바이스를 포함하고 있다. 표 1은 단말기의 주요 사양이다. CPU로는 인텔 PXA255로서 400MHz의 클럭 스피드를 가지고 있어 이미지, 동영상을 단말기에 디스플레이할 수 있을 정도의 충분한 성능을 가지고 있다. 플래시 메모리는 64MB로서 부트로더, 커널, 루트파일시스템 그리고 사용자 정의형 파일 시스템을 포함하기 충분한 공간이며 메인메모리를 128MB의 램을 장착하고 있어 동영상 등의 디스플레이를 위한 충분한 메모리 공간을 제공한다. LCD는 3.5인치의 비교적 소규모 패널 크기이며 일반적인 PDA의 화면크기와 비슷하며, 통신 인터페이스로는 이더넷이 2포트, PCMCIA 인터페이스가 있어 무선랜이 가능하며, 시리얼, 블루투스와 같은 인터페이스가 있어 다양한 응용이 가능하다. 저장 장치로는 CF와 HDD가 동시에 지원된다. HDD를 지원하는 이유는 CF가 아직은 HDD에 비하여 가격이 비싸기 때문이다. 본 구현에서도 HDD 장착형 단말기를 구현하여 저가형 단말기 구현 기술을 확보하였다. 입력 장치로는 터치 스크린과 10개의 키패드가 있어 기존 상용 PDA와 비슷한 기능을 제공할 수 있다.

Ⅲ. 임베디드 리눅스 포팅

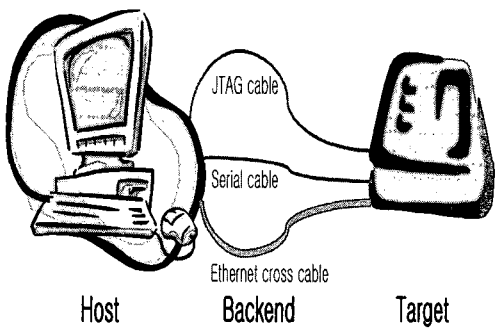

그림 2. 개발 환경
Fig 2. Development environment

3장에서는 개발된 단말기에 임베디드 리눅스 커널 및 부트로더, 그리고 각종 디바이스 드라이버의 포팅 내용을 기술한다. 그림 2는 임베디드 리눅스 커널을 포팅하기 위한 개발 환경이고 그림 3은 보드를 디버깅하기 위한 주요 환경이다. 개발보드는 컴파일 환경을 가질 수 없는 최소한의 자원만 가지고 있으므로 x86과 같은 호스트 컴퓨터에서 커널, 파일시스템, 사용자 파일시스템,그리고 부트로더를 개발한다. 주요 개발 환경으로는 다음과 같다.

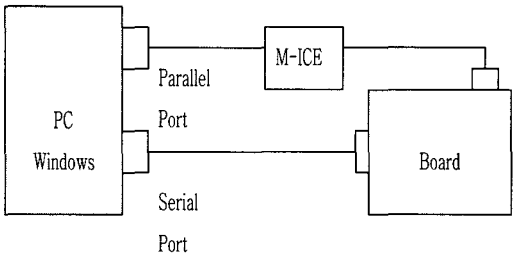

그림 3. 디버깅 환경
Fig 3. Debugging environment

- Linux kernel 2.4.18
- gcc 3.3.x를 비롯한 툴 체인
- JTAG을 통한 보드 디버깅 및 부트로더 다운
- 시리얼 케이블을 통한 모니터링
- 이더넷 케이블을 통한 이미지 다운로드
- REALVIEW를 통한 보드 디버깅
- Multi-ICE를 통한 보드 디버깅

(1) 부트로더 개발

개발 보드에는 u-boot가 포팅되었다. 원래 u-boot는 PPC용으로 개발된 부트로더이지만 ARM 용으로 확대된 범용 부트로더이다. 간단하게 실행 흐름을 설명한다.

① startup(cpu/arm920t/start.S) - cpu초기화, dram초기화, 그 후에 부트로더의 ram으로의 relocation, C코드에서의 main인 start_armboot()를 호출함.
② c쪽의start코드(lib_arm/board.c) - dram초기화 이후의 flash, uart등의 주변 디바이스 초기화
③ 메인루프(command line shell: common/main.c) - 유닉스 shell과 같은 명령어 처리 루틴. autoboot delay 동안 enter키 입력을 기다리는데 그동안 키 입력이 안되면 바로 boot command(설정파일을 보면 나옴)를 실행시킴.
④ app의 실행 혹은 리눅스 등의 OS로 부팅(lib_arm/armlinux.c) - 헤더가 붙은 커널의 헤더를 읽어서 압축을 풀고 리눅스로 제어권을 넘겨주는 코드

다음은 부트로더에서 메모리 및 주변회로, 스택, 램의 크기, 시작주소를 지정하는 부분을 설명한다.

- 메모리, 주변 회로 구성

CS0: Amd29LV800BB, 1MB, 16 bit모드 - boot flash CS1: Intel strata flash E28F128J3A x 2, 32비트 모드 - data저장용 flash

CS3: cs8900 ethernet driver

CS6: 삼성 K4S561632C-TC75 32MB x 2, 32비트 - SDRAM

위의 내용과 같이 CS0는 부트 플래시에 CS1은 데이터 저장용 플래시에 연결되어 있다. 또한 CS3는 이더넷, CS6는 SDRAM에 연결되게 구성하였다.

- 스택 지정

```
#define CONFIG_STACKSIZE (128*1024) /* regular stack */
#ifdef CONFIG_USE_IRQ
#define CONFIG_STACKSIZE_IRQ (4*1024) /* IRQ stack */
#define CONFIG_STACKSIZE_FIQ (4*1024) /* FIQ stack */ #endif
```

스택의 크기를 지정한다.

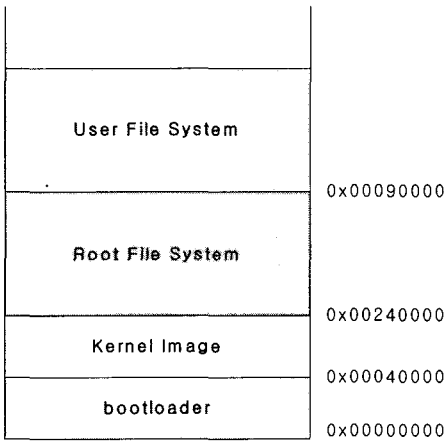

그림 4. 메모리 맵
Fig4. Memory map

- SDRAM의 크기와 시작 주소를 지정한다.

```
#define CONFIG_NR_DRAM_BANKS 1 /* we have 1 bank of DRAM */
#define PHYS_SDRAM_1 0x30000000 /* SDRAM Bank #1 */
#define PHYS_SDRAM_1_SIZE 0x08000000 /* 64 MB */
#define PHYS_FLASH_1 0x00000000 /* Flash Bank #1 */
#define CFG_FLASH_BASE PHYS_FLASH_
```

(2) 메모리 맵

flash에서 부팅하기 위해서 u-boot, 커널 이미지, 램디스크가 저장될 위치, 즉 주소를 지정해야한다. 그림 4는 본 논문에서 사용한 플래시 메모리 맵이다. 부트로더는 0x00000000, 커널이미지는 0x00040000부터 저장되고, 램디스크 이미지는 0x00240000에 저장되면 나머지 공간은 사용자 영역으로 할당된다.

(3) 루트 파일 시스템 개발

본 논문에서 루트 파일 시스템은 하드디스크에 존재하지 않고 플래시메모리에 저장되어 있다가 부팅시 램에 설치되기 때문에 메모리의 일정 부분을 할당하여 하드 디스크처럼 사용가능하도록 하고 있는데 이것을 램디스크(ramdisk)라고 한다. 이 램 디스크는 램 자체가 휘발성이므로 전원을 차단하면 모든 데이터가 사라지며, 램의 특성상 실행이 빠르며 압축을 이용하므로 용량을 효율적으로 사용할 수 있는 장점이 있다. 본 논문에서는 몬타비스타 하드햇 리눅스의 램디스크 사용하여 램디스크를 구성하였다. 다음은 주요 개발 절차이다.

① 16메가바이트 크기의 루프백 디바이스를 생성한다.

```
rm -f bigRamdisk
dd if=/dev/zero of=bigRamdisk bs=1024
   count=16384
```

② ext2 파일시스템을 생성한다.

```
echo "make bigramdisk file system"
mke2fs ./bigRamdisk
```

③ 생성된 파일시스템을 개발용 루트파일시스템에 마운트한다.

```
echo "mounting bigramdisk to newramdisk"
/bin/mount -t ext2 -o loop bigRamdisk
   /mnt/newRamdisk
```

④ 개발용 루트파일시템의 소스를 새롭게 생성된 파일시스템에 복사한다.

```
echo "copy orignal ramdsk to new ramdisk"
    cp -rf /mnt/ramdisk/* /mnt/newRamdisk/
```

⑤ 생성된 파일시스템을 언마운트한다.

```
echo "unmounting new ramdisk"
```

```
umount /mnt/newRamdisk
⑥ 생성된 파일시스템을 gzip으로 압축한다.
echo "gzip new ramdisk"
gzip bigRamdisk
⑦ mkimage 툴로 u-boot 용 램디스크로 만든다.
echo "making new uimage new ramdisk"
./mkimage -A arm -O linux -T ramdisk -C gzip \
        -n 'Ext2 Ramdisk Filesystem' \
        -a 0x12000000 -e 0x12000000 \
        -d ./bigRamdisk.gz ./uramdisk
⑧ /tftpboot 에 복사하여 타겟보드의 플래시메모리에
   복사한다.
cp ./uramdisk /tftpboot
```

(4) USB 인터페이스를 통한 HDD 제어

본 논문에서는 단말기의 컨텐츠 내용을 편리하게 저장하기 위해 USB 인터페이스를 통하여 HDD 제공한다. 따라서, USB 장치를 인식하기 위한 디바이스 드라이버 개발이 필수적이다. 그림 5는 본 구현에서 사용한 UMS(USB Management System)의 타이밍 그림이다.

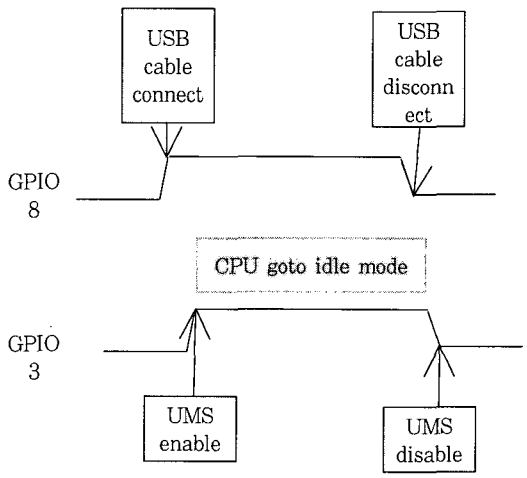

그림 5. UMS 타이밍
Fig 5. USM timing

그림 5에서 보듯이 GPIO 8번이 High가 되면 USB 케이블이 연결되었으므로 GPIO 3을 High 로 설정하여 UMS를 enalbe 시킨다. 이때부터 하드디스크는 연결된 PC에서 제어하게 된다. USB 케이블이 비연결되면 GPIO 8 이 Low 상태가 되는데 이때 GPIO 3을 Low 상태로 만들어 UMS를 disable 시킨다. UMS가 enable 상태인 동안 PMP는 idle 상태가 된다. 참고로 s3c24a0에서 idle 상태는 cpu로의 클럭만 정지되고 나머지 장치로의 클럭은 정상 입력되는 모드이다. UMS는 usb 케이블의 연결 및 비연결동작을 감지해서 커널로 보내면 커널이 HDD 마운트와 언마운트를 제어하며, CPU idle 모드로의 결정은 커널이 UMS 모듈로 통보한다.

표 2. UMS와 App 사이의 메시지
Table 2. Messages between UMS and App

Message ID	동작
X_COMMAND_PID	App 가 UMS 모듈로 pid를 전송하는데 사용
X_COMMAND_MOUNTED	App 가 UMS 모듈로 HDD가 마운트되었음을 통보하는데 사용
X_COMMAND_UNMOUNTED	App 가 UMS 모듈로 HDD가 언마운트되었음을 통보하는데 사용
기타	

그림 6 에서 보듯이 UMS 는 usb 케이블의 연결 및 비연결동작을 감지해서 App로 보내면 App가 HDD 마운트와 언마운트를 제어하며, CPU idle 모드로의 결정은 App가 UMS 모듈로 통보한다. 표 2는 UMS와 App가 signal() 함수로 메시지 통신을 할 때 사용되는 메시지 종류들이다.

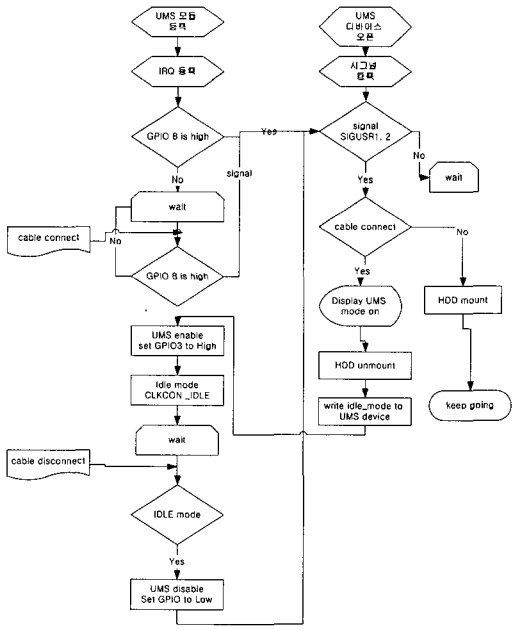

그림 6. UMS와 App의 상호 동작 프로세스
Fig 6. Flow chart between UMS and App

그림 7은 UMS 디바이스 드라이버의 핵심인 ISR(Interrupt service routine)의 코드이다. 라인 3~6은 인터럽트 마스킹에 대한 처리를 수행한다. UMS는 INT 8번을 사용하므로 8번을 제외한 나머지 인터럽트는 클리어 시켜 UMS가 신속하게 응답할 수 있도록 하였다. UMS ISR의 전반적인 흐름은 파워 절약 모드일 때의 동작과 파워 절약 모드가 아닐 때의 동작을 나누어서 처리하도록 하였다. 라인 7~16은 파워 절약 모드로 진입하기 위한 것이며 라인 17~27은 외부 인터럽트에서 UMS가 wakeup하기 위한 코드이다. 그리고 28~30 라인은 인터럽트를 복구 시키는 동작을 수행한다. 라인 11에서는 이미 UMS가 동작 중이면 바로 빠져나가고 그렇지 않으면 라인 13에서 파워 절약 모드로 진입하기 위한 GPDAT를 설정하고 UMS_Idle_Mode() 호출하여 파워 절약 모드를 위한 백업 작업을 수행한다. 라인 18에서는 파워 절약 모드를 빠져나오기 위해서 GPDAT를 복구 시키고 클록을 인가시키기 위해 클록 아이들 플래그를 해제시킨다. 라인 24~26까지는 INT 8번을 클리어 시켜 더 이상의 추가 INT가 현재 작업을 방해하지 못하도록 설정해 준다.

```
static void  ums_isr(int irq, void *dev_id, struct pt_regs *regs)
{
1  int i = 0;
2  printk("Entering ums_isr(EINT08)...₩n");

3  EINTMASK |= (BIT_EINTPEND_EINT8);
4  INTMSK |= (BIT_EINT7_10);
5  EINTPEND = BIT_EINTPEND_EINT8;
6  ClearPending(BIT_EINT7_10);

7  if (!nTrytoEnterPowerSavingMode)
8  {
9    sys_kill(nox_pid, UMS_INSERTED);
10   for (i=0; i < 100000; i++);
11   if ( mp_mounted == 1 )
12     goto out;

     // Try to enter power saving mode
13   GPDAT=GPDAT | (1<<3);
     // set ATA_EN to high-level
14   for(i=0;i<100;i++);
15     nTrytoEnterPowerSavingMode = 1;
16     UMS_Idle_Mode();
   }
    else
   {
     // Wake up from power saving mode by external interrupt
17   for (i=0; i < 100000; i++);
18   GPDAT=GPDAT & ~(1<<3);
     // set ATA_EN to low-level
20   for (i=0; i < 100000; i++);
21   CLKCON &= ~(CLKCON_IDLE);
     // return to normal mode
22   nTrytoEnterPowerSavingMode = 0;
23   printk("Wake-up from power savin mode by external interrupt(EINT08)...₩n");

24   EINTMASK &= ~(BIT_EINTPEND_EINT8);
25   INTMSK &= ~(BIT_EINT7_10);
26   sys_kill(nox_pid, UMS_REMOVED);
27   return;
     }
28    out:
29     EINTMASK &= ~(BIT_EINTPEND_EINT8);
30     INTMSK &= ~(BIT_EINT7_10);
   } // end of ISR
```

그림 10. UMS 디바이스 드라이버 ISR
Fig 7. ISR of UMS device driver

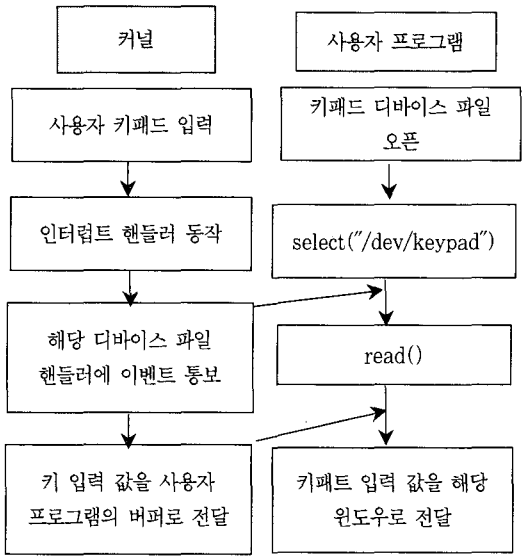

그림 11. 키 패드 동작 흐름도
Fig 8. Flow chart of keypad operation

(5) 키패드 디바이스 드라이버

그림 8은 키 패드 디바이스 드라이버의 동작 원리를 나타내고 있다. 커널에서는 사용자의 키패드 입력이 발생하면 인터럽트 핸들러가 동작을 하고 키패드 디바이스 파일 핸들러에 이벤트를 통보한다. 그리고 키 입력 값을 사용자 프로그램의 버퍼로 전달하게 된다. 사용자 프로그램에서는 키패드 디바이스 파일을 오픈하면 select() 시스템 콜로 키패드 입력의 발생 여부를 폴링하여 발생되면 read() 시스템 콜에 의해 입력값을 커널로부터 획득하게 된다. 주요 동작 절차는 다음과 같다.

Linux # mknod keypad c 233 0
:/dev 밑에 keypad 란 디바이스 파일이 생성됨.

(6) 사용자 파일 시스템

개발된 단말기에는 64MB의 플래시 메모리가 장착되어 있다. 부트로더, 커널, 그리고 루트 파일시스템이 탑재되고 남은 공간은 사용자 정의 파일시스템을 구축하여 GUI(Qt)를 위한 공간과 사용자가 디스크처럼 사용할 수 있는 공간을 제공한다. 이 공간은 ramdisk /etc/fstab 파일에서 지정되어 있는데 /mnt/mtd에 마운트되어 램 디스크의 /mnt/mtd 아래에서 수행한 작업들은 시스템 종료후에도 지워지지 않고 그대로 남아 있게 된다. 다음은 본 연구에서 사용한 파일시스템인 jff2를 구축하기 위한 절차이다.

① jffs2를 제작하기 위한 디렉토리 작성
② 작성한 디렉토리에 Qt를 위한 프로그램을 저장한다.
③ mkfs.jffs2 유틸리티로 jffs2 파일 시스템을 생성한다. 이것은 가상 파일 시스템으로 호스트 PC의 파일형태로 만들어 진다.
예: mkfs.jff2 -r jffs2-make -o jffs2 —eraseblock =0x40000 —pad=0x1800000
④ 생성된 파일을 플래시 메모리에 다운로딩한다.

IV. Qt/Embedded 기반의 관광안내 프로그램

임베디드 리눅스 환경에서 애플리케이션 개발은 GUI 툴킷의 선택이 매우 중요하다. 이유는 각각의 임베디드 기기마다 환경이 다르기 때문이다. 그러므로 기기에 맞는 GUI 툴킷을 선택해야 하는 것이다. 대부분 공개 소프트웨어로 제공되는데 Qt/Embedded, Microwindows, MiniGUI, Tiny-X 등이 이용되며 본 연구에서는 윈도즈, 리눅스 등 다양한 플랫폼에서 MS 윈도우즈에 능가하는 GUI 기능을 제공하는 TrollTech사의 Qt/Embedded 및 Qtopia 윈도우 매니저를 이용하는 관광안내 프로그램을 개발하였다. 특히 Qtopia는 리눅스 플랫폼상에서 동작하는 윈도우 매니저로 다양한 기능을 제공한다.

그림 9는 Qtopia 용 GUI 프로그램 개발 과정이다. 일반적으로 Linux용 Qt 의 경우 Qt/X11, Qt/E, 그리고 Qtopia가 필요하다. 우선 Host PC에서 위 세 개의 패키지를 설치하고 가상 프레임 버퍼를 이용하여 Qtopia를 에뮬레이션한다. 이 가상 버퍼상에서 계측용 애플리케이션을 개발하게 된다. 애플리케이션 개발이 완료된 후 타겟 보드용 Qt/E 라이버러리를 컴파일하여 실제 타겟보드에 포팅하게 된다. 본 논문에서 사용한 Qt 버전은 다음과 같다.

- Qt/X11 2.3.2
- Qt/Embedded 2.3.7
- Qtopia 1.7.0
- tmake 1.13

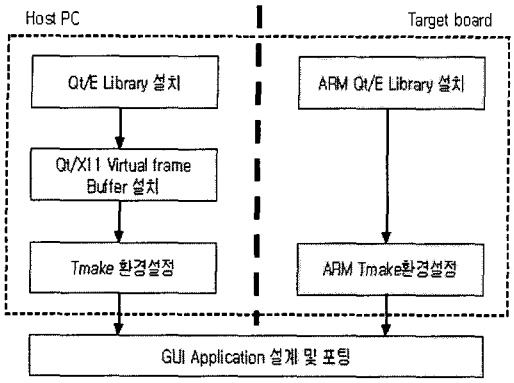

그림 12. Qt/E 개발 과정
Fig 9. Qt/Embedded development process

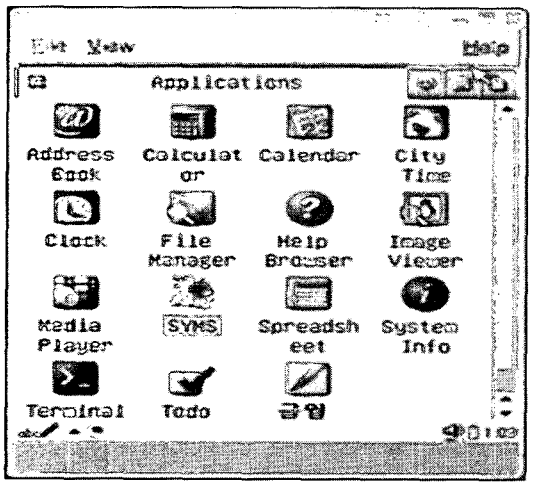

그림 10. 단말기 초기화면(Qtopia)
fig 10. Qtipia starting display

그림 10은 Qtopia가 실행된 화면이고 그림 11은 관광안내 애플리케이션이다. 구현된 인터페이스에는 관광지의 이미지, 텍스트 설명, 그리고 비디오 안내 등의 기능을 가지고 있다.

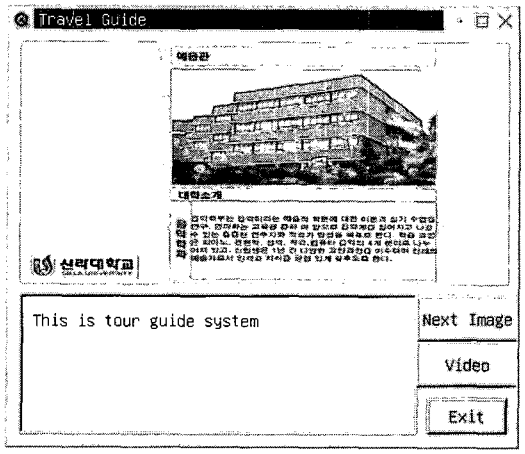

그림 11. 관광안내 애플리케이션
Fig 11. Tour guide application

V. 결론

본 논문에서는 임베디드 리눅스를 이용한 관광안내 단말기를 설계 및 구현하였다. 구현된 단말기는 리눅스 운영체제를 탑재하고 있는 PDA와 비슷한 인터페이스를 가지고 있으며 Qtopia가 구동되고 있어 사용자에게 WinCE 와 필적할 만한 편리한 인터페이스를 제공할 수 있다. 향후 연구과제로는 무선 통신 기능을 추가하여 관광정보를 실시간으로 업데이트하며 RFID와 연계한 관광안내 단말기를 제작하는 것이다.

참고문헌

[1] Embedded Linux tops developers' 2002 wishlist, LinuxDevices.com, 2001. 17.

[2] Stephen Balaco, Whitepaper: Linux's Future in the Embedded Market, LinuxDevices.com, 2001.6.

[3] Rick Lehrbaum, "Using Linux in Embedded and Real-time Systems," LinuxDevices.com, 2000.2.

[4] 박영준, 리눅스 플랫폼의 임베디드 프로그래밍, 프로그램 세계, 2000.7.
[5] John Lombardo, Embedded Linux, New Riders, 2002.
[6] http://linux-embedded.com/software.php3.
[7] BlueCat Linux Users Guide - Release 4.0, Lynux-Works, 2001.
[8] http://www.peeweelinux.org/Documentation/PeeWeeLinux.html.
[9] http://www.trolltech.com/
[10] "Presentation: The State of Embedded Linux," LinuxDevice.com Homepage, http://www.linuxdevices.com/articles/AT2113794413.html, July, 2000.
[11] 이민석, 모바일 기기를 위한 임베디드 리눅스, pp.112-119, 정보처리학회지, 제9권1호, 2002.
[12] Prabhat k. andleigh, Kiran thakrar, Multimedia Systems Design, pp.112, Prentice Hall PTR, 1996.

저 자 소 개

김 정 원

1995년 부산대학교 전자계산학과 (학사)
1997년 부산대학교 대학원 전자계산학과 (석사)
2000년 부산대학교 대학원 전자계산학과 (박사)
2000년~2001년 기술신용보증기금 기술평가역(차장)
2002년~현재 신라대학교 컴퓨터정보공학부 교수
관심분야 : 내장형시스템, 멀티미디어, 운영체제

전 봉 기

1991년 부산대학교 컴퓨터공학과 (학사)
1993년 부산대학교 컴퓨터공학과 (석사)
2003년 부산대학교 컴퓨터공학과 (공학박사)
1993년~1998년 한국통신연구소 전임연구원
2000년~2001년 동명대학교 정보공학부 전임강사(기간제)
2003년 ~현재 : 신라대학교 컴퓨터정보공학부 조교수
〈관심분야〉 데이터베이스, 공간 데이터베이스, 이동체 데이터베이스