임베디드 리눅스 기반의 사용자 영상인식시스템 구현

박창희* · 강진석** · 고석만** · 김장형****

The Implementation of User Image Recognition based on Embedded Linux

Chang-Hee Park* · Jin-Suk Kang** · Suk-Man Ko*** · Jang-Hyung Kim****

요 약

본 논문에서는 CIS(CMOS Image Senser)와 GPS 모듈이 장착된 임베디드 시스템에 리눅스를 포팅하여, 리눅스 커널 상에 카메라와 GPS 모듈을 인식시켜 GPS 모듈로부터 GGA(Global positioning system fix data) 문장을 획득하고 위치 정보를 CIS로부터 정지영상을 얻을 때 수신되는 위치 정보를 정지영상에 포함하는 것을 목적으로 한다. 임베디드 시스템을 위한 하드웨어를 구성하고 카메라 설치가 가능한 보드를 장착해서 리눅스 부트로더와 커널을 포팅한 후 CIS(CMOS Image Sensor) 제어 디바이스 드라이버와 GPS 모듈 디바이스 드라이버를 커널에 작동가능하게 구현한다. GPS 모듈로부터 현재 위치의 위도와 경도 값을 문자열 형태로 획득하고, CIS로부터 초당 17 프레임의 영상을 획득하여, 한 프레임을 정지영상으로 저장한다. 정지영상에 위치정보를 추가시켜 JPEG 압축을 하고 결과를 얻어오는 임베디드 영상처리 시스템을 구현하였다.

ABSTRACT

In this paper, we propose a system that the Linux is ported in embedded system with peripheral devices of CIS(CMOS Image Sensor) and GPS module. The system acquires GGA sentence from GPS module by recognizing camera and GPS is used module in Linux kernel. And then the received location information is used to include still image acquired through CIS According to this paper, We compose hardware for embedded system, attach board (including camera), port Linux BootLoader and Kernel. And, then we realize that it insert kernel in CIS control device driver and GPS module device driver.

키워드

Embedded Image Processing, LBS(Location Based Services), GPS, CIS(CMOS Image Sensor)

Ⅰ. 서 론

이동통신 기술의 발달과 더불어 다양한 통신망에서 이동단말의 위치를 계산하는 기술이 활발하게 연구되고 있으며, 무선뿐만 아니라 유선 통신망과 GPS(Global positioning system fix data) 음영지역인 실내, 빌딩 숲 속에 있는 사용자의 위치를 파악하는 기술이 개발되고 있다. 이렇게 사용자의 위치를 수집하는 방법이 다양하게 연구되고, 그 수요가 증가함에 따라 많은 국내외 민간기업, 공공기관과 지자체 등에서는 위치정보를 가공하여

* 제주산업정보대학 인터넷비지니스학과
** 인천대학교 컴퓨터공학과
*** 제주산업정보대학 인터넷비지니스학과
**** 교신저자 : 제주대학교 통신컴퓨터공학부

접수일자 : 2007. 1. 23

민간과 공공부분에 서비스할 다양한 콘텐츠를 제작하고 있다.

이와 같이, 위치기반서비스(LBS: Location Based Service)는 위치확인 기술을 이용한 이용자의 위치를 파악과 이와 관련된 어플리케이션을 부가하는 서비스이다[1]. 현재 차량용 네비게이션이나 휴대폰 등에 LBS의 연구가 활발하게 진행되고 있고 인프라 구축 수준도 높아지고 있다.

그에 반해 LBS가 영상처리분야에서는 많은 연구가 이루어지지 않고 있다. 디지털 카메라나 캠코더 등을 이용하여 영상을 획득할 때 위치 정보를 영상에 추가한다면 이를 이용한 다양한 분야로의 응용이 가능할 것이다.

본 논문에서는 일반적으로 intel chip을 사용하는 Windows 기반의 컴퓨터가 아닌 ARM(Advanced anced RISC Machine) CPU를 사용하는 임베디드 리눅스 기반의 시스템에서 영상처리를 연구한다[2].

연구의 흐름은 범용 임베디드 시스템에 Linux 커널을 포팅하고, 카메라와 GPS 모듈을 장착하여 정지영상에 위치정보가 추가된 정지영상을 임베디드 리눅스에서 구현하고 이 영상을 파일로 저장하여 다양한 분야에서 활용이 가능하게 한다.

임베디드 시스템을 위한 기존의 여러 상용 운영체제가 존재하지만 새로운 기능이 요구될 때 핵심 커널을 수정하기가 용이 하지 않다. 그러나 임베디드 시스템에 리눅스를 사용할 경우 지금까지 입증된 리눅스 운영체제의 우수성을 그대로 사용할 수 있고, 필요시 소스를 수정해 사용할 수 있는 여러 이점이 있다[9].

Ⅱ. 이론적 고찰

2.1 임베디드 리눅스의 부트 로더(Boot Loader)

임베디드 시스템을 개발하는데 있어서 부트 로더는 커널이 시작되기 전 커널의 전반적인 운영환경을 제공한다.

일반적으로 부트 로더는 커널이 동작하기 위해서 필수적으로 필요로 하는 시스템 하드웨어의 요소를 초기화 시키는 기능과 타겟 시스템의 메모리에 커널을 적재시키는 기능을 수행하며, 커널의 운영환경을 위해 필수적으로 필요한 하드웨어 요소로서 메인 메모리 초기화, 타이머 초기화, 인터럽트 초기화, 시리얼 인터페이스 초기화, 시스템 버스(로컬 버스, PCI 버스) 초기화 등의 기능이 포함된다. 호스트 컴퓨터에서 개발한 커널을 타겟 보드의 메모리로 다운로드 시키기 위한 방법으로는 네트워크를 이용한 커널 다운로드 방법, 시리얼 인터페이스를 이용한 커널 다운로드 방법, 플래쉬 메모리를 이용한 커널 다운로드 방법과 그 이외의 다른 방법들이 존재한다.

일반적으로 임베디드 시스템에서 하드웨어의 초기화 기능과 호스트 컴퓨터로부터 타겟 보드에 커널의 다운로드 기능은 부트 로더에서 수행된다.

2.2 임베디드 리눅스의 커널(Kernel)

운영체제의 가장 핵심적인 기능은 컴퓨터에 들어있는 여러 자원들을 관리하는 것이다. 프로세서와 메모리, 디스크, 네트워크, 인터럽트 등 프로그램이 사용하는 모든 자원을 관리하며, 자원을 필요로 하는 프로세스에게 적절하게 배분하고, 또한 자원을 잘못 사용하는 것을 막는다.

나아가 프로그램이 좀 더 복잡한 작업을 하는데 필요한 기능들을 제공한다. 이러한 운영체제의 가장 핵심적인 부분을 커널이라고 부른다. 운영체제는 커널 위에 사용자와의 인터페이스를 제공하는 쉘(shell)이 더해져서 만들어진다.

쉘은 컴퓨터를 사용할 때 사용자가 직접 접하는 부분이다. 사용자는 키보드나 마우스 등의 입력 장치를 통해 쉘에 명령을 내리면 쉘은 필요한 작업을 커널에 요청하여 실행함으로써 시스템이 동작하게 된다. 리눅스 커널은 프로세스 관리, 메모리 관리, 프로세스 간 통신, 파일 시스템, 네트워크 등의 운영체제의 기본 기능과 함께 방대한 양의 디바이스 드라이버로 이루어져 있다. 그림 1은 이런 리눅스 커널의 구조를 잘 표현하고 있다.

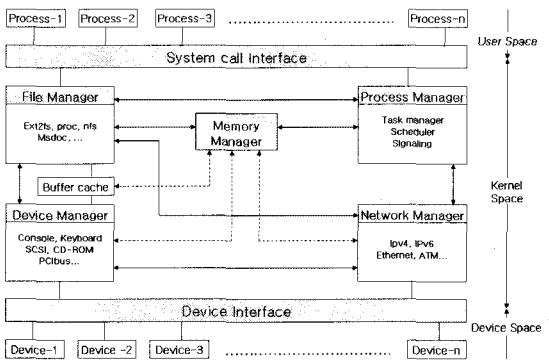


그림 1. Linux 커널의 기본 구조
Fig. 1. The Structure of Linux Kernel

위의 그림에서 보이는 것처럼 커널의 역할은 다음과 같이 구분할 수 있다.

○프로세스 관리(Process management)
-프로세스 생성 및 소멸
-프로세스 간 통신(Signal, Pipe, LPC, 세마포어 등)
-CPU 스케줄링 동기화
-제한된 자원에 대한 다중프로세서의 효율적인 관리기법 제공

○메모리 관리(Memory Management)
-가상 메모리 관리 기법 제공
-메모리 하드웨어의 효율적인 관리

○파일 시스템 관리(File System Management)
-가상 파일시스템에 의한 여러 파일 시스템 타입 지원
-디스크의 물리적 구조를 논리적 구조로 표현 하는 기법
-파일, 디렉터리 관리

○디바이스 관리(Device Management)
-입출력 요청 검증
-입출력 요청 작업의 스케줄링
-주변장치와 메모리간의 자료 전송
-입출력 제어기 관리
-인터럽트 요청 및 처리

○네트워크 관리(Network Management)
-통신 프로토콜 구현

Ⅲ. 시스템 라이브러리

3.1 호스트 컴퓨터와 타겟 보드 구축

임베디드 시스템 안에서 직접 개발을 한다는 것은 쉬운 일이 아니다. 그렇기 때문에 부트 로더, 커널이미지, 램디스크, 사용자영역 설정 등 기타 여러 응용 프로그램을 개발하기 위한 작업 공간을 제공하는 컴퓨터가 필요하다 . 이러한 컴퓨터를 호스트 컴퓨터라 하며, 호스트 컴퓨터에서 개발된 소프트웨어로 구동되는 시스템을 타겟 보드라 한다[3].

개발환경을 구축하기 위해 가장 먼저 해야 할 일은 타겟 보드 개발에 필요한 호스트 컴퓨터의 환경(LINUX)을 구축하는 일이다. 타겟 보드가 어떠한 운영체제를 선택하느냐에 따라 호스트 환경에서 각종 유틸리티를 운용하는데 테크닉이 필요하게 된다. 이렇듯 호스트 컴퓨터에서 타겟 컴퓨터를 위한 각종 소프트웨어를 작성하고 컴파일하기 때문에 타겟 컴퓨터가 물리적으로 존재하지 않거나 소프트웨어의 형식이 가상적으로만 존재하더라도 타겟 컴퓨터에 사용 될 소프트웨어를 미리 만들어 볼 수 있다.

그림 2는 개발 호스트 구축 환경의 예이다. 개발 호스트로 LINUX를 설치하고 크로스 컴파일 개발환경을 구축한다. 가장 초기에 하드웨어적으로 타겟 컴퓨터에 접근하기 위해 JTAG를 사용한다. 하지만 JTAG의 처리 속도는 매우 느리므로 개발 보드에서 부트 로더가 동작 한 이후에 실행 이미지를 다운로드하기 위해서는 이더넷을 이용한 네트워크상에서의 다운로드를 이행한다. RS232는 부트스트랩 로더가 시동할 때 필요한 모니터링 명령을 내리거나 부팅 도중이나 부팅이 끝난 후 응용프로그램이 실행하는데 필요한 콘솔 입·출력을 화면에 나타낼 경우 사용한다.

Ethernet의 NFS(Network File System)는 리눅스 커널이 동작한 후에 타겟 보드에는 대용량 보조 기억 장치가 없으므로 호스트와 파일을 공유하여 보조 기억 장치로 사용하기 위해서 필요하다. 또한 호스트 컴퓨터에 있는 파일시스템을 타겟 컴퓨터에서 원격 마운트로 사용하거나 TFTP(Trivial File Transfer Protocol), FTP(File Transfer Protocol)를 사용하여 커널 이미지나 램디스크, 각종 응용프로그램을 다운로드 할 경우 사용되고 텔넷으로 원격 접속을 하기 위해서도 사용된다. 이때 사용되는 것이 Bootp(Bootstrap Protocol)와 TFTP이다. Bootp를 설치함으로써 호스트 컴퓨터에서 타겟 보드에 포팅 된 리눅스 시스템에 유용한 IP 주소를 유동적으로 할당할 수 있다.

네트워킹에 기본이 되는 IP를 먼저 할당함으로써 tftp가 부트 로더에서 사용가능하게 된다.

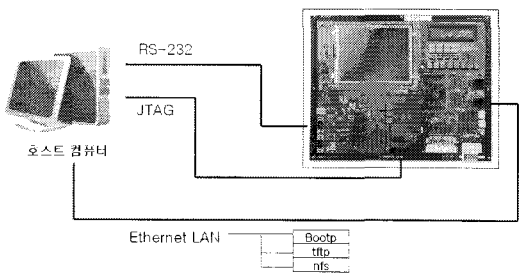


그림 2. 개발 호스트 구축 환경
Fig. 2. Development Host Constructing

임베디드 시스템에서 하드웨어는 크게 마이크로프로세서, 메모리, 주변장치로 구분할 수 있다. 마이크로프로세서는 임베디드 시스템을 구성하는 가장 중요한 하드웨어 구성요소로 최근에 나온 64비트 마이크로프로세서도 있지만 임베디드 시스템에서는 주로 8, 16비트 마이크로프로세서를 사용하며, 최근에는 32비트 마이크로프로세서를 많이 사용한다.

임베디드 시스템의 메모리에는 램(RAM)과 롬(ROM)이 있으며, 커널 이미지나 램디스크 이미지와 같이 비휘발성 데이터들은 롬에 저장한다. 표 1은 타겟 보드의 하드웨어 사양을 보여 주고 있다[4].

표 1. 타겟보드 하드웨어 사양
Table. 1. Hardware Specifications with Targetboard

CPU		Intel Xscale PXA255(over 400MHz)
Display	LCD	LG Philips LB064V02 640*480 TFT LCD
	TEXT LCD	DATA IMAGE CM202S1LY-K2 Character LCD
	7-Segment	
	LED	SMD Type
Memory	SDRAM	Micron MT48LC8M16A2(64MB*2)
	FLASH	Intel StrataFlash 28F128J3A(32MB*2)
External Interface		3.3V Compact Flash Type
		PS/2*2
		20-Pin JTAG
		USB client port
		4*4 Keypad
		PCMCIA slot
Serial port		9-Pin RS-232*2
Ethernet		Cirrus Logic CS8900A

3.2 크로스 컴파일러 구축

본 논문에서 사용하는 프로그램들은 x86계열이 아닌 ARM 보드에서 동작을 요하기 때문에 크로스 개발 환경을 구축해야 한다. 크로스 개발 환경은 그림3과 같이 호스트 컴퓨터와 타겟 보드로 구성되는데 호스트 컴퓨터는 컴파일, 링크, 디버깅과 응용프로그램 작성에 이용되고, 타겟 보드는 호스트 컴퓨터에서 크로스 컴파일 된 응용프로그램을 적재하여 실행시키기 위해 이용된다.

따라서 호스트 컴퓨터에서 제작된 소스 코드를 실행환경이 다른 타겟 보드에서 작동하기 위해서는 호스트 컴퓨터에 설치된 컴파일러가 아닌 ARM 용으로 컴파일을 해주는 컴파일러, 링커, 라이브러리 등이 필요하다.

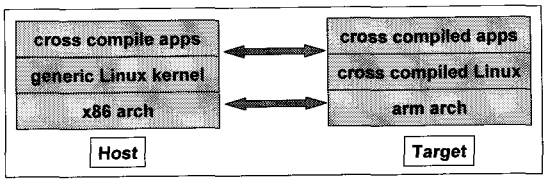


그림 3. 호스트 컴퓨터와 타겟 컴퓨터와의 관계
Fig. 3. On the Relationship of Host Computer and Target Computer

Ⅳ. 시스템 모듈 구현 및 고찰

4.1 GPS 모듈 설계

XScale에는 세 개의 UART(Universal Asynchr- onous Receiver Transmitter)를 가지고 있다. 본 논문에 사용한 타겟 보드는 이 세 가지 모두 사용하도록 디자인 되어 있다. 그 중에서 BTUART를 통해 GPS 모듈과 통신하여 유효한 정보를 얻을 수 있다. 먼저 공통의 BTUART (Bluetooth UART)를 사용하므로 XScale GPIO를 어드레스 디코더에 입력하여 각각의 장치로부터 데이터를 받도록 설계를 한다[3].

그림 4는 각각의 장치에 BTUART 의 Rx, Tx를 연결해주기 위한 회로도이며, 어드레스 디코더에서 필요한 GPS_OE 신호를 만들어 준다.

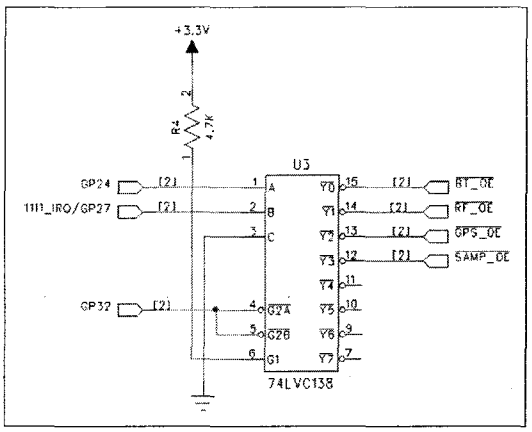


그림 4. BTUART의 Rx, Tx를 연결해주기 위한 회로도
Fig. 4. BTUART Rx, Tx Circuit Design

따라서 XScale GPIO 24, 27, 32를 mmap를 출력으로 설정한 후 GP24, GP27에 회로 설계한 신호를 인가하여 BTUART의 Tx, Rx를 GPS 모듈로 연결하고 데이터를 수

신하면 1초마다 갱신된 NMEA 프로토콜 데이터가 수신된다. 이때 수신된 문장을 분석하면 현재위치 정보를 알 수 있다. 본 논문에 사용된 GPS 모듈은 다음과 같다.

- SiRF Star II chipset
- 프로토콜 지원
- COM A : NMEA Protocols : GGA, GSA, GSV, RMC
- 4,800 to 115,200 bps
- 좌표 갱신율 1/sec

4.2 커널(Kernel)

부트 로더의 수정과는 달리 커널 수정은 패치파일이라는 특정한 파일을 이용해서 하게 되며, 패치파일은 원본 소스에 변경 부분의 정보를 가지고 있어서 이 파일을 특별한 명령어로 패치하면 원본 소스가 변경된다.

리눅스 커널 소스는 많은 CPU 칩들에 적용될 수 있도록 작성되었기 때문에, 사용하려고 하는 CPU에 맞는 패치파일들을 찾아서 패치 해야 하고, 타겟 보드에는 인텔의 xscale칩을 메인 CPU로 사용하고 있으므로 이 칩에 맞도록 패치를 해야 한다.

그 후의 프로세스는 첫째, CIS(CMOS Image Sensor)를 사용하기 위해 CIS는 사이클론 FPGA(Field-Programmable Gate Array)에 만들어진 드라이브 코어를 사용한다. 디바이스 드라이버는 코어를 사용하여 화상 데이터를 가져온다. 디바이스 드라이버는 커널과 제어를 필요로 하는 하드웨어 사이의 계층이다. 이 디바이스 드라이버는 커널을 매우 단순화 한 유용한 추상적 개념으로 커널이 시스템과 직접 대화하게 하는 대신에 잘 정의 된 인터페이스를 제공하여 작업을 개별적인 디바이스 드라이버에게 맡기는 형식이다. 디바이스 드라이버들은 커널에 정적으로 링크되어서 사용되기도 하며, 때로는 동적으로 운영체제가 가동 중인 상태에서 커널과 링크 되거나 제거 될 수 있다.

이를 위해 리눅스에서는 모듈기능을 제공하여 디바이스의 동적 적재 및 제거를 가능하게 한다. 그림 5는 디바이스 드라이버가 어느 계층에 존재하는지 잘 보여준다[5].

둘째, CIS제어 디바이스 드라이버를 만들고 커널에 삽입하여 타겟 보드에서 정지영상을 얻어오는 과정은 필요한 헤더 파일을 포함한다.

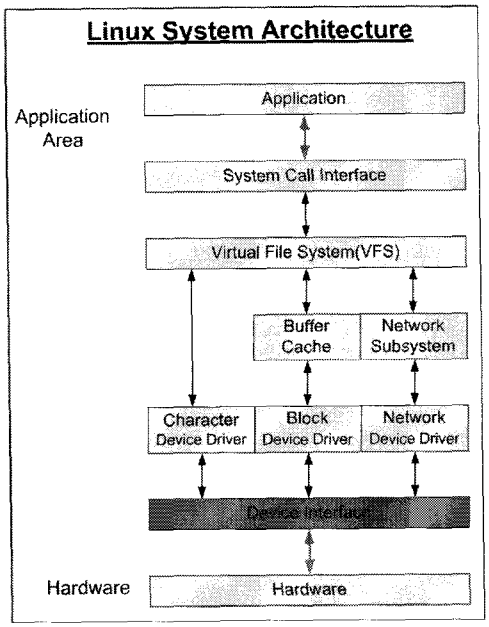


그림 5. 리눅스 시스템 구조에서 디바이스 드라이버
Fig. 5. Devices Driver of Linux System Structure

사이클론에 접근하기 위한 가상 번지를 설정하여 인터럽트를 GPIO27번 핀을 통해서 받게 되며, 인터럽트는 CIS에서 한 프레임의 영상이 만들어졌을 경우에 low로 떨어진다. 디바이스 드라이버에서는 이 핀이 low로 떨어지는 시점에서 일정 인터럽트 루틴으로 들어가게 되며, 이 루틴에서 프레임의 픽셀 값을 가져온다. 픽셀 값을 다 가져왔으면 SIGUSR1(사용자 정의) 시그널을 응용프로그램으로 전송한다. 이 시그널을 받은 응용프로그램은 한 프레임 영상을 화면에 뿌리게 된다.

셋째, 인터럽트가 걸린 경우 인터럽트 루틴이 수행된다. 먼저 인터럽트가 걸리지 않도록 설정하고, 픽셀 값을 저장하고 있는 번지에서 픽셀 데이터를 프레임 크기만큼 읽어 온 후 SIGUSR1 시그널을 전송한다. 인터럽트가 다시 걸리게 하려면 응용프로그램에서 pid값을 write함수를 이용하여 디바이스 드라이버로 넘겨주면 다음의 gpio_write함수가 실행되면서 인터럽트가 걸리게 된다.

넷째, 응용프로그램에서 write함수를 이용하여 응용프로그램의 pid 값을 쓰게 되면 수행되는 루틴으로 인터럽트가 걸리도록 설정하고 있다. 즉 영상을 계속 수신하려면 pid값을 계속 write함수를 이용하여 디바이스 드라이브로 넘겨주면 된다.

다음의 그림 6은 실제 영상의 픽셀 값을 구하여 그 값들을 응용프로그램으로 넘겨주는 루틴이다.

```
static int gpio_open(struct inode *inode, struct file *filp)
{
        int res;
        unsigned int gafr;

        //gafr = 0x3; //gpio16
        gafr = (0x3 << 22);//gpio27
        GAFR0_U &= ~gafr;

        GPDR(IRQ_TO_GPIO_2_80(IRQ_CMOS))
        &= ~GPIO_bit(IRQ_TO_GPIO_2_80(IRQ_CMOS));
        set_GPIO_IRQ_edge(IRQ_TO_GPIO_2_80(IRQ_CMOS),
         GPIO_FALLING_EDGE);

        res = request_irq(IRQ_CMOS,&cmos_interrupt,SA_INTERRUPT,
                          "CMOS",NULL);

        if(res < 0)
                printk(KERN_ERR "%s: Request for IRQ %d failed\n",
                        __FUNCTION__,IRQ_CMOS);
        else {
                enable_irq(IRQ_CMOS);
        }
        pFlag = (unsigned int *)FLAG_ADDRESS;
        // frameinfo=*pFlag;
        pSRAM = (unsigned int *)SRAM_ADDRESS;
        MOD_INC_USE_COUNT;
        return 0;
}
```

그림 6 . 픽셀 값을 구하는 루틴
Fig. 6. Pixel Routine

그리고 응용프로그램에서 디바이스 드라이버를 실행할 경우에 대한 부분을 설정하여야 한다. 디바이스 드라이버를 닫을 경우(close), 디바이스 드라이버를 삽입(insmod) 할 경우와 디바이스 드라이버를 제거(rmmod) 할 경우에 대한 루틴을 만들고 컴파일을 하여 디바이스 드라이버를 만든다.

마지막으로 CIS 디바이스 드라이버를 사용하기 위한 응용프로그램을 만든다. 응용프로그램은 qt라이브러리를 가지고 제작한다. 타겟 보드의 개발환경은 qtopia가 포팅 되어 있다[6]. 디바이스 드라이브에서 받은 픽셀 정보는 arm JPEG 라이브러리를 이용하여 정지영상을 저장한다. 응용프로그램을 작성하고 컴파일 하여 이를 저널링 플래시 파일 시스템(Journalling Flash File System)으로 만든다. JFFS는 디스크가 없는 임베디드 장치에서 플래시 메모리를 이용한 전원 · 파손 등에 안전한 파일 시스템이다. 컴파일 과정을 거친 응용프로그램을 JFFS 이미지로 만들고 타겟 보드의 부트 로더를 이용하여 다운로드해서 실행하면 원하는 영상을 얻을 수 있다[3].

4.3 정지영상에 위치 정보 적용

타겟 보드에서는 위치정보와 초당 17프레임의 영상을 얻을 수 있으며, GPS 모듈을 통해 1초마다 NMEA 문장을 수신한다. 여러 문장 중 GPGGA 문장의 위도와 경도 값을 얻어오고 그 값을 문자열 변수에 저장한다.

그리고 TextLCD에 표시한다. CIS에서 수신된 영상은 픽셀 값을 받아서 BMP 파일로 만들고 타겟 보드의 CF 메모리에 저장한다.

위치 정보를 정지영상에 포함하기 위해서 저장된 영상을 다시 불러오고 거기에 위도와 경도 값을 받은 변수의 값을 얻어온다. 불러온 정지영상의 윈도우 프레임의 DC값을 얻어서 정지영상 위에 위도 와 경도 값을 변수로부터 받아서 추가한다. 그리고 armJPEG 라이브러리를 이용하여 JPEG 파일을 만든 후 CF 메모리에 저장한다. 그림 7은 GPS 모듈에서 수신한 위도와 경도 값의 문자열을 정지영상에 추가하는 소스의 일부분이다.

```
QColor text("white");
QPixmap *pix = new QPixmap("input.bmp");
QPainter p;
p.begin(pix);
p.drawText(pb->rect(), AlignRigntBottom, "*data");
//위치정보
p.end();
pix->save("output.jpg"); //정지영상 저장
```

그림 7. 정지 영상에 위치 정보 추가 루틴
Fig. 7. Location Information Add Routine of Still Image

4.4 시스템 구조

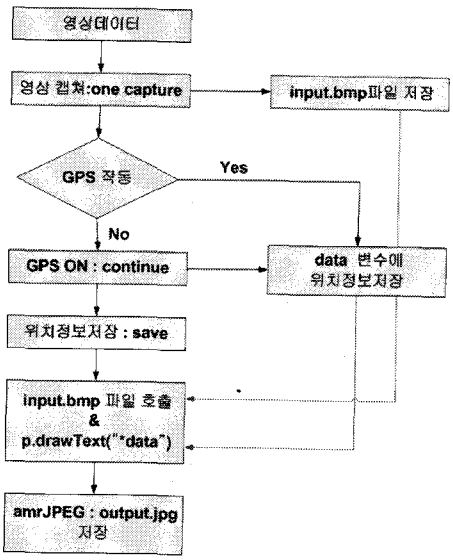


그림 8. 시스템 전체 흐름도
Fig. 8. System Flowchat

그림 8은 본 논문에서 설계한 시스템 전체의 흐름도이다. 타겟 보드에 GPS모듈이 장착된 CIS를 연결하고 타겟 보드에서 구현된 응용프로그램을 실행키면 정지영상과 위치정보를 획득할 수 있다.

먼저 GPS모듈을 작동시키기 위해 GPS ON 버튼을 활성화 시킨 후 카메라로부터 들어오는 영상을 캡처한다. 캡처된 영상은 타겟 보드의 CF 메모리에 저장된다. 위치정보를 사진에 적용하기 위해 저장된 사진에 현재 수신된 위도와 경도 값을 적용하고 armJPEG 라이브러리를 이용해서 위치정보가 적용된 JPEG 파일을 얻을 수 있다.

4.5 구현 및 고찰

실험에 사용된 장비는 Intel Xscale PXA-255(400 MHz)가 장착된 임베디드 장비를 사용하였다.

XScal에는 세 개의 UART를 가지고 있으며, 그 중에서 BTUART를 통하여 GPS 모듈과 통신을 하여 유효한 정보를 얻었다. BTUART를 사용하므로 XScale의 XScale GPIO를 어드레스 디코더에 입력하여 각각의 장치로부터 데이터를 받도록 설계하였다. 실험에 사용된 GPS는 SiRF StarⅡ chipset을 장착하여 NMEA프로토콜과 GGA, GSA, GSV, RMC를 지원하며, 좌표 갱신은 1초에 한 번씩 이루어진다.

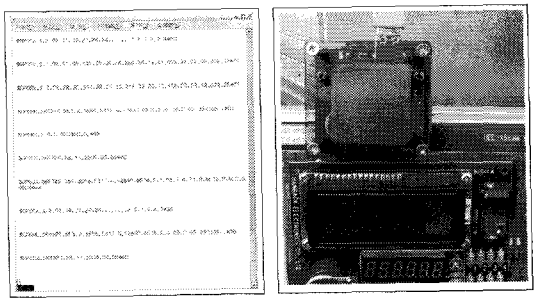
영상을 획득하기 위해 사이클론 FPGA와 CMOS Image Sensor를 사용하였으며, CIS카메라는 초당 17 프레임의 영상을 획득한다.

위치정보의 획득은 GPS 모듈을 BTUART에 연결하고 위성으로부터 데이터를 수신하면 1초마다 갱신된 NMEA 문장이 수신된다.

다음의 그림 9는 호스트 컴퓨터를 통하여 NMEA 문장이 수신되는 값과 정지영상에 사용하기 위해 GPGGA 문장 중에서 경도와 위도 값을 획득한 화면이다.

위치정보 값 중에서 위도와 경도 값을 획득한 이 값을 정지영상에 적용하기 위해 정지영상을 획득한다. CIS를 사용하기 위해 부트 로더와 커널을 타겟 보드에 맞게 수정을 하여 포팅 하고 디바이스 드라이버를 구현하여 타겟 보드가 부팅 될 때 적재되게 설정하였다. CIS제어를 위해 응용프로그램을 프로그래밍 한다.

개발환경은 호스트 컴퓨터의 qt-embedded-2.3.2 환경에서 프로그래밍 하고, 타겟 보드에는 Qtopia가 포팅 되어 있다. 응용프로그램은 디바이스 드라이버에서 받은 픽셀정보를 받아서 arm용 jpeg라이브러리를 이용하여 jpg파일로 저장한다.



a) NMEA 문장 수신 b) 위도와 경도 값 출력

그림 9. 위치 정보 획득
Fig. 9. The Value of Location Information

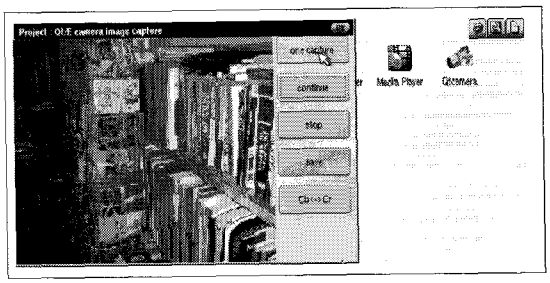


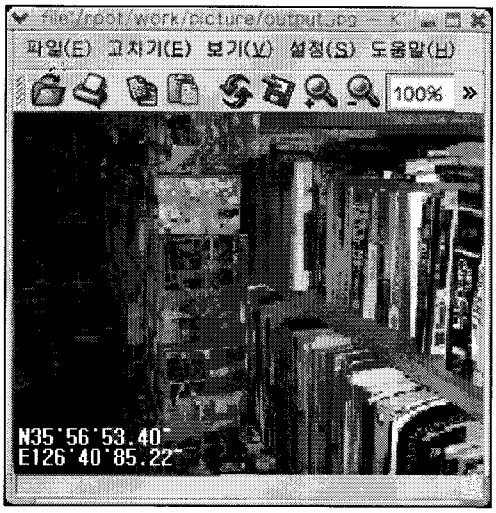
그림 10. qt/embedded camera image capture 응용 프로그램
Fig. 10. qt/embedded camera image capture Application Program

그림 10은 qt/embedded 개발환경에서 프로그래밍 한 파일을 타겟 보드의 Qtopia를 통해 구현된 CIS 영상을 얻어오는 화면이며, 여기에서 구현된 응용프로그램의 기능은 다음 5가지이다.

-one capture : 입력 받은 영상에서 정지영상을 캡처한다.
-continue : GPS 모듈 작동하게 한다.
-stop : GPS 모듈 작동을 멈추게 한다.
-save : 정지영상에 위치정보 값을 적용하고 CF 메모리에 저장한다.
-Cb<->Cr : 영상의 흑백모드 와 컬러모드로 전환하는 기능이다.

그림 11은 CIS로부터 수신된 정지영상을 저장한 그림 a)와 저장된 정지영상에 위치정보 값을 적용한 후에 얻은 그림 b)의 정지영상이다.

a) CIS로부터 직접 저장



b) 저장된 파일에 위치정보 적용

그림 11. 위치 정보가 적용된 정지영상 획득
Fig. 11. Still Image with Location Information

위의 그림에서 화질이 조금 차이가 있는 것은 a) 그림에 armJPEG 압축을 하기 때문에 b)의 화질의 질이 다소 떨어지는 결과를 얻었다.

V. 결 론

Windows 기반의 intel chip을 사용하는 X86 컴퓨터에서 영상처리와 GPS 모듈 사용은 어렵지 않게 구현이 가능하다. 본 논문에서는 arm CPU를 사용하는 임베디드 리눅스 기반의 시스템에서 구현을 하였다는 점에 중점을 두었다.

임베디드 시스템에서 GPS 모듈을 이용하여 위치정보를 획득하고, CIS를 이용하여 정지영상을 획득하였다. 획득한 정지 영상에 GPS 모듈을 이용하여 얻은 위치정보를 arm JPEG 라이브러리를 이용하여 영상에 위치정보를 추가하여 결과를 보였다. GPS 모듈의 위치 정보 수신율이 1초 간격으로 위치정보를 수신하기 때문에 위치의 정확성은 다른 어느 것 보다도 뛰어나다. 임베디스 시스템 기반에서 설계를 하였기 때문에 디지털 카메라, PDA, 핸드 핼드 PC등에 쉽게 적용이 가능하다. 앞으로의 연구 방향은 국내에서 많이 사용되고, 도면 데이터 교환을 위해 개발된 DXF 포맷의 국립지리원 전자지도[11]를 이용하여 지리정보 시스템을 구축하고, 이를 이용하여, 정지영상에 위도, 경도의 좌표가 출력이 되는 부분을 정확한 지명이 출력되도록 지리정보 시스템을 구축하는 것이다. 그리고 수신된 영상에 비해 결과적으로 얻은 이미지의 화질이 다소 떨어지는데 arm용 JPEG 압축 알고리즘의 개선을 통하여 화질의 성능 향상시켜야 될 것이며, 실제 디지털 카메라에 GPS 모듈을 연결하여 구현될 수 있도록 연구를 할 것이다.

참고문헌

[1] Pillip J. Koopman, Jr "Embedded System Design Issues", Proc. of the Internationl Conf. on Computer Design(ICCD96)

[2] Wookey, Chris Rutter, Jeff Sutherland, Paul Webb, "The GNU Toolchain for ARM targets HOWTO", http://www.alephl.co.uk/armlinux/docs/tollchain

[3] 임베디드 리눅스 시스템 empos II (주)한백전자

[4] Hanback Electronics (http://www.hanback.co.kr/) prentice hall, 165- 192

[5] 유영창, "리눅스 디바이스 드라이버," 한빛미디어
[6] Patric Ward, "QT programming for linux and windows 2000,"
[7] Intel PXA255 Processor Developer's Manual
[8] ARM Architecture Reference Mannual
[9] National Marine Electronics Association(http://www.nmea.org)
[10] 강진석, 2005, "모바일 시스템에 적합한 위치기반 서비스에 관한 연구", 제주대학교 공학박사학위논문
[11] 윤재관, 한기준, 2002, "LBS(Location Based Service)를 위한 기술개발 동향", 대한전자공학회 전자공학회지, Vol.29, No.12.
[12] 국토지정보원(http://www.ngi .go.kr)
[13] 한국천문연구원 GPS 연구 그룹(http://www.gps.re.kr)
[14] http://www.korone.net/

저자소개

박 창 희(Chang-Hee Park)

1988년 일본 Kinki University경영공학과 (이공학사)
1993년 일본 Kinki University(상학석사)
2003년 제주대학교 대학원 정보공학과 박사과정 수료
1995년~현재 제주산업정보대학 인터넷비즈니스과 교수
※관심분야 : 멀티미디어시스템, 영상처리

강 진 석(Jin-Suk Kang)

1999년 제주대학교 정보공학과(공학사)
2001년 제주대학교 대학원 정보공학과(공학석사)
2005년 제주대학교 대학원 정보공학과(공학박사)
2004년 8월~2006년 2월 군산대학교 BK교수
2006년 9월~현재 인천대학교 연구교수
※관심분야 : 멀티미디어시스템, 영상처리

고 석 만(Suk-Man Ko)

1988년 광주대학교(경영학사)
1992년 동국대학교 대학원 전자계산학과(경영학석사)
2002년 조선대학교 대학원 컴퓨터공학과 (공학박사)
1993년~현재 제주산업정보대학 인터넷비즈니스과 교수
※관심분야 : 멀티미디영상처리, 인터넷비즈니스

김 장 형(Jang-Hyung Kim)

1981년 홍익대학교 정밀기계공학과 (공학사)
1983년 연세대학교 대학원 기계공학과 (공학석사)
1990년 홍익대학교 대학원 기계공학과 (공학박사)
1998년~2000년 제주대학교 전자계산소장
1984년~현재 제주대학교 통신컴퓨터공학부 교수
※관심분야 : CAD/CAM, 멀티미디어, 인공지능