ARM CPU를 이용한 리눅스기반 독립형 Vision 처리 모듈 개발

이석*, 문승빈*
*세종대학교 컴퓨터공학과
e-mail : sbmoon@sejong.ac.kr

Development of Stand-Alone Vision Processing Module Based on Linux OS in ARM CPU

Seok Lee* and Seungbin Moon*
*Dept. of Computer Engineering, Sejong University

요 약

현재 Embedded system 에서 많은 기업체들이 리눅스를 채용하고 있고, 이러한 임베디드 리눅스는 실시간 운영체제가 필요한 로봇제어기에서부터 PDA, set-top box 등 여러 분야에 걸쳐 응용되고 있다. 본 논문에서는 StrongARM SA-1110 CPU 을 이용하여 만들어진 임베디드 시스템에 리눅스를 사용하여 독립형 비전모듈을 개발한 내용을 기술한다. 또한, WinCE 를 사용하여 개발된 비전모듈과의 성능을 비교하여 리눅스를 이용한 독립형 비전모듈을 평가하고, 머신비전 분야에서의 리눅스 응용 가능성을 제시하였다.

1. 서론

리눅스 OS 는 현재 다양한 CPU 에 포팅되어 사용되고 있으며, 최근에는 임베디드 시스템에서의 Linux 응용이 많은 관심사로 제기되고 있다.

리눅스를 응용하여 내장형 장비를 개발하거나 연구되는 사례는 크게 두 가지로 나누면 고전적인 유닉스의 시분할 방식을 그대로 유지하는 경우와 리얼타임 기능을 가지는 RT-Linux[5]등을 이용하는 경우로 나누어지고 있다. 전자의 경우는 PDA 가 대표적인 예로서 현재 많은 개발이 이루어지고 있으며, GUI 환경이 WinCE 에 버금가는 상태에 이르고 있다. 후자의 경우는 VoIP[2], Router[3], 로봇등과 같은 장비에서 사용되고 있다.

최근에 개발되는 장비들의 성향은 Microsoft 사의 Windows 의 GUI 가 가지는 장점을 적용시키는 사례가 많다. 이러한 경향은 특히 리눅스를 이용한 PDA 에서 두드러지게 나타나고 있다.

본 논문에서는 리눅스를 이용하여 개발된 장비들의 사례를 바탕으로 독립형 비전처리 모듈에 적용하여 WinCE 가 내장된 독립형 비전처리 모듈과의 성능을 비교하여 가능성을 제시하였다. 또한 내장형 장비에서 GUI 를 이용하는 추세에 맞추어 비전 처리모듈의 어플리케이션의 제작한 경험을 기술하고자 한다.

본 논문의 구성은 다음과 같다. 2 절에서는 개발환경에 있어서의 하드웨어 구성과 각 기능에 대해서 기술한다. 3 절에서는 내장형 장비에 있어서 리눅스 커널을 포팅하는 방법 및 카메라로부터 영상획득을 위한 디바이스 드라이버 그리고 획득한 영상을 표시하기 위한 어플리케이션의 개발에 대한 내용을 기술한다. 4 장에서는 리눅스를 이용한 독립형 비젼모듈과 WinCE 를 이용한 것과의 비교 분석 및 결론을 기술한다.

2. Hardware Platform

하드웨어의 구성은 그림 1 에서 비전처리모듈의 실물 사진을 보여주고 있고, 그림 2 에서는 하드웨어 구성도를 나타내고 있다. 그림 2 에서 볼 수 있듯이, CPU 는 StrongARM SA-1110 32bit RISC 를 이용하였고, 메모리는 SDRAM 32MB, Flash Memory 16MB 를 이용하였으며, 호스트 장비와의 통신을 위해서 LAN Adapter, 8IN/16OUT PIO, RS-232C 를 사용하였고, 디스플레이를 위해서 MQ200 chipset 을 사용하였다. 이 모델은 Assabet[6]를 기반으로 하여 만들어졌다.

RS-232C 는 flash 메모리에 있는 부트로더가 수행

시 host PC 와의 인터페이스 역할을 한다. 즉, RS-232C 를 통하여 커널 이미지와 램 디스크 이미지등을 메모리에 올려 놓거나 메모리에 올려진 이미지들을 flash 메모리에 쓰는 역할을 수행하게 하였다.

해당 영상을 CRT 에 디스플레이 하기위해서 MQ200 Graphic chipset 을 사용하였다. 이 제품은 저전력, 고성능을 그 장점으로 하고 있으며, 2D Graphic Accelerator 기능을 가지고 있다.[7]

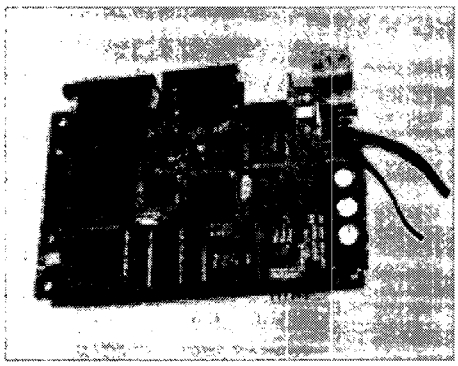

그림 1. **RABBIT Board**

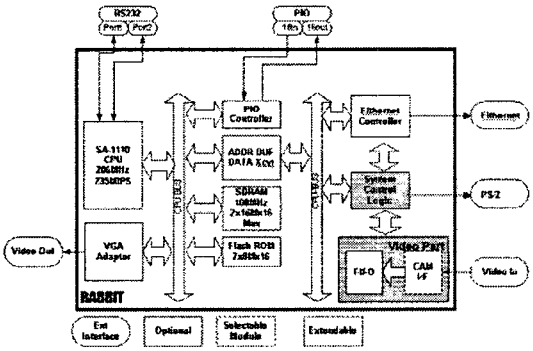

그림 2. **Schematic Diagram**

Flash 메모리는 비휘발성 메모리로 보조기억장치로서 사용된다. 부트로더, 커널이미지, 램 디스크 이미지 및 파일 시스템 모듈을 이용하여 리눅스에 필요한 라이브러리들과 설정파일들을 저장하는데 사용하였다. 그 구성은 아래의 표 1 에서와 같이 하였다.[3]

size	offset	name
0x00040000	0x00000000	Rabbit Boot Loader
0x000c0000	0x00040000	Kernel Image
0x00300000	0x00100000	Ramdisk Image
0x00c00000	0x00400000	User Space

표 1. **Flash Memory 구성**

3. Linux Kernel 및 Application

3-1. Linux Kernel

리눅스 커널은 여러 프로세스에서 이식 작업이 성공적으로 진행되었지만 특정한 내장형 시스템의 경우에 있어서는 커널소스 및 해당 디바이스 드라이버를 수정해야 하는 작업이 필요하다.[3] 독립형 비전모듈을 개발하는데 있어서 커널소스에서 RS-232C 통신, 메모리의 구성, I/O mapping, 네트워크 드라이버 및 flash 메모리를 파일시스템으로 사용하기 위한 모듈을 수정하였다.

수정된 커널을 컴파일을 한 후 생성된 zImage 를 boot loader 의 명령어를 통해서 RAM 의 영역에 전송하게 된다. 여기에서 사용된 전송방법은 RS-232C 를 통해서 base-64 로 인코딩된 데이터를 boot loader 가 받고 이를 다시 base64 로 디코딩하여 메모리에 전송을 시키는 방법을 이용하였다.

압축된 커널 이미지는 zImage 라고 하고 그 크기가 500KB 에서 600KB 정도의 수준이었다. 커널의 이미지를 RS-232C 를 이용하여 RAM 에 저장시켜야 하므로 115200bps 의 속도로 570KB 를 전송하는데 약 1 분 정도가 소요되었다.

ARM Linux[4]는 커널로 들어가기 전에 boot loader 에게 세 가지 요구사항을 만족시키도록 하고 있다. 하나는 Memory Management Unit(MMU)을 disable 시키고 register r0 의 값을 0x00 으로 설정하고 register r1 의 값에 Architecture ID 를 설정한다. 이 ID 는 커널이 각 Architecture 마다 해당 설정을 할 수 있게 해준다.

마지막으로 decompressor 가 압축된 커널이미지를 풀고 kernel entry point 로 점프하게 되면 커널이 수행된다.

3-2. Video Input Device Driver 의 설계

디바이스 드라이버는 크게 Block Device Driver, Character Device Driver, Network Device Driver 로 분류할 수 있다. 비디오 입력장치는 스트림 기반의 장비이므로 Character Device Driver 로 하는 것이 적합하다.[10]

리눅스에서는 이미 비디오 입력 장치에 대한 디바이스가 많이 존재하며, 또한 OOP(Object Oriented Programming)의 추상객체와 같이 비디오 입력 장치 드라이버에 대한 추상 드라이버가 제공되고 있다. 이와 같은 기법을 Module Stacking 라 한다.

리눅스에서는 Video Input 을 위한 추상 드라이버를 Video4Linux[11]라고 하고, 여기서 개발된 디바이스 드라이버 또한 이것에 기반으로 하여 만들어 졌다.

SA1110 은 Bus DMA 가 지원되지 않으므로 가능한 한 연속적인 데이터를 가져오기 위해서 또한 FIFO 의 호환성을 고려하기 위해서 Variable Latency I/O[1]를 이용하였다.

인터럽트가 발생되는 주기는 16ms 이며 초당 60 회가 발생하고 한번에 읽어오는 이미지 데이타의 크기는 640x240 이다. 인터럽트 발생시에 데이터를 커널의 메모리에 복사하는데 6ms 의 시간이 소요되었다. 인터럽트의 발생시 수행시간을 6ms 로 보장하기 위해서

이미지가 저장되는 FIFO 영역을 커널의 가상메모리 영역으로 맵핑할 때, 캐쉬 비트를 설정하여 데이타를 고속으로 읽을 수 있게 했다. 개발과정에서 테스트를 해본 결과 캐쉬비트를 설정하지 않고 이미지 데이타를 읽어 왔을 때 걸리는 시간은 12-14ms 가 소요되었다. Interrupt Service Routine(ISR)에서는 데이터를 복사하기 위한 일을 이외의 다른 작업을 하는 것은 적합하지 못하였다.

카메라로부터 데이터를 획득하는 절차는 아래의 그림 3 과 같다.

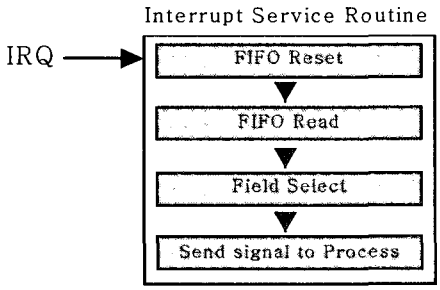

그림 3. Interrupt Service Routine

매초당 인터럽트가 발생하면 인터럽트 루틴은 FIFO_READ 레지스터를 읽는 동작으로 읽을 위치를 FIFO 의 처음으로 돌려놓는다. FIFO Read 구간에서는 Burst 모드로 FIFO 에 있는 데이타를 고속으로 읽고 그 내용을 커널의 메모리에 저장시킨다. Field Select 에서는 640x240 의 이미지가 odd 인지 even 인지에 대한 것을 결정한다. 이는 GPI17 를 이용하여, 카메라로부터 들어온 영상 이미지가 even 이면 1, odd 이면 0 으로 설정되게 하였고 이를 통해서 읽어온 이미지의 필드를 결정한다. 마지막으로 읽기 작업이 마치면 대기하고 있는 사용자 프로세스에게 유닉스 시그널(SIGIO)을 이용해서 알려준다. 디바이스 드라이버에서 사용자 프로세스의 등록은 디바이스 드라이버에 관계된 장치 파일을 열었을 때 커널의 전역 변수인 current 로부터 얻을 수 있었다. current 는 struct task_struct 의 포인터로 현재 프로세스의 대한 정보를 가지고 있다.

유닉스 계열의 특징 중 하나는 장치도 파일로서 다룰 수 있다는 점이다. 디바이스 드라이버는 이것을 가능하게 해주는 교량적 역할을 한다. 해당장비에 대해서도 역시 파일 인터페이스를 통하여 사용자 프로그램이 카메라로부터의 데이터를 획득할 수 있게 했다.

일반적으로 파일을 제어하는 함수는 open, read, write, ioctl, close 등이 있다. 카메라는 read 는 할 수 있어도 write 가 필요없는 장비이므로 write 는 지원하지 않았다.

이 장비에서 쓰이는 StrongARM SA-1110 은 우리가 사용하는 PC 의 CPU 보다 그 성능의 차가 많다. 이것은 드라이버로부터의 데이터 획득의 방법이 read 만으로는 그 성능을 보장 할 수가 없음을 의미한다. 이러한 점을 보완하기 위해서 Memory Mapped I/O 를 지원했다. 개발된 내장형 장비에서 read 함수만으로 구성해서 read 함수가 호출되었을 때, 640x240 의 크기를 읽는데 대략 4ms 가 소요되었다.

전반적인 디바이스 드라이버의 구조에 대한 도식은 아래의 그림 4 와 같다.

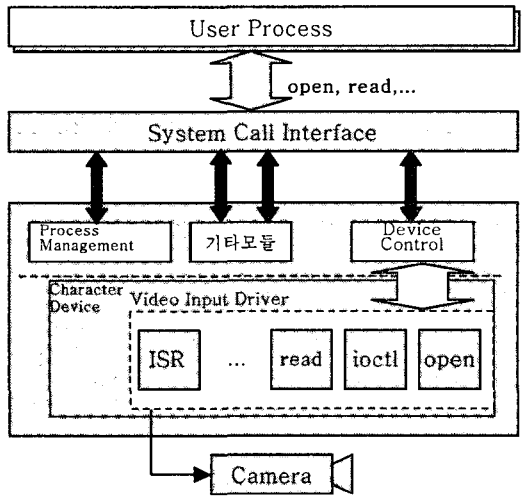

그림 4. Video input device driver 구조도

3-4. GUI Toolkit

최근 경향은 Embedded 장비라 하더라도 GUI 화면 구성이 필수로 들어간다. Linux 는 X-Window 를 사용하므로 Xlib 를 기반으로 하여 GUI 를 구성할 수 도 있다. 하지만 이는 용량이 크고 작성하기에 많은 노력이 필요하며 개발기간도 길어진다. 이러한 것을 보완하고자 Toolkit 들이 많이 등장했으며, Linux platform 에서는 주로 GTK(The GIMP Toolkit)와 QT 가 사용되고 있다.

대부분의 GUI toolkit 들은 많은 기능을 제공하기 때문에 그 크기가 크다. 하지만, Embedded 장비들은 기억용량이 작으므로 그것에 맞추어서 GUI 라이브러리의 크기 또한 작아야 한다.

Embedded QT[8]는 TrollTech 사에서 만든 GUI Toolkit 이다. QT 는 Cross-platform C++GUI application framework 이다. QT 가 지원하는 운영체제들은 Microsoft 사의 Windows 95, 98, NT, 2000 이 있으며, Unix 계열에서는 Linux, Sun Solaris, HP-UX, Digital Unix, IBM AIX, SGI IRIX, 기타 여러 vendor 들, Embedded 로는 Linux platform 에서의 frame buffer 를 지원한다.

본 개발장비에서는 frame buffer 를 사용하는 Embedded QT 를 사용하여 GUI 를 구성하였다. Embedded-QT 는 그 기능에 따라서 최소 800KB 에서 7.5MB 까지 라이브러리의 크기를 정할 수 있다.

Embedded-QT 는 Window Manager 가 없어도 application 을 실행시킬 수 있다는 점과 기존의 소스와도 호환을 가진다는 점이 장점이다. 하지만, 아직까

지는 소수의 I/O 장치에 대해서만 구현이 되어 있으므로 지원되지 않는 장치에 대해서는 직접 수정하거나 구현해야 하는 경우도 발생한다.

3-5. Vision Application 의 구현

Vision Application 의 구현은 크게 두 개의 부분으로 나눌 수 있다. 디바이스 드라이버와 통신을 하여 해당 이미지 버퍼 및 영상의 제어에 해당하는 부분과 입력받은 영상을 화면에 출력하는 부분으로 나눌 수 있다.

디바이스 드라이버에 대한 액세스는 일반적으로 유닉스 파일 인터페이스로 구성된다.

유닉스에서 파일의 입/출력은 system call 인 read 와 write 을 통해 이루어지고 그 파일에 대한 제어는 fcntl 이나 ioctl 등을 통해서 이루어진다. 여기서 사용되는 비전 모듈에서는 매 초마다 odd field 및 even field 에 해당하는 인터럽트를 합쳐 60 번이 발생하여 초당 30 프레임을 만든다. 이렇게 생성된 영상 데이터를 매번 read 함수를 호출하여 구현한다는 것은 간접비용이 크다. 이러한 간접비용을 줄이기 위해서 Memory Mapped I/O 를 이용하여 영상데이타를 직접 access 할 수 있게 구현하였다.

Memory Mapped I/O 는 대규모의 디스크 입출력을 수행해야만 할 경우 자료를 커널의 내부 버퍼로 복사하고, 이어 동일한 자료를 사용자 프로세스에 포함된 자료 구조에 다시 복사해야 하는 성능상의 간접비용의 발생을 디스크의 파일을 바로 프로세스의 영역에 맵핑함으로써 파일의 접근 속도를 증가시키는 기법이다.[3]

영상 입력 장치 파일은 실제 디스크가 아니므로 인터럽트 발생시 이미지를 커널에 저장하게 되는 버퍼를 디스크라고 간주하고 이에 대해서 맵핑을 한다.

Memory Mapping 된 영역은 이제 커널과 사용자 프로세스가 이 영역을 공유하게 된다. 리눅스를 기반으로 해서 만들어진 몇 몇 이미지 캡쳐 드라이버는 사용자 프로세스에게 이미지를 인터럽트기간에 이미지를 받았는지에 대한 여부가 없어서 응용 프로그램이 이를 폴링하는 경우가 있었다. 이러한 경우는 이미지를 정확히 받을 수 있는 인터페이스를 제공하지 못한다. 간단히 이미지 디스플레이에는 적합할지 모르나 비전 모듈에서는 적합하지 못하였다. 본 개발에 있어서 응용프로그램은 디바이스 드라이버가 인터럽트 루틴이 끝나는 시점에서 발생시켜주는 시그널을 받을 수 있게 하였다. 즉, POSIX 기반의 sigaction() 함수를 이용하여 인터럽트 핸들러가 이미지에 대한 처리를 수행하게 하였다.

응용 프로그램은 두 가지로 개발을 하였다. 하나는 GUI 를 사용하지 않고 8bit grayscale 모드에서 수행되는 것과 다른 하나는 GUI 를 사용하여 16bit 모드에서 수행되는 것이다. 둘 모두 프레임버퍼를 이용하였다. 8bit 모드에서는 640x480 의 이미지를 보여주는데 좋은 성능을 보였지만 16bit 모드에서는 만족할 만한 성능을 보여주지 못했다. 이는 리눅스의 성능보다는 MQ200 드라이버의 지원이 부족함에 기인했다.

4. 결론

해당 RABBIT 보드는 WinCE 와 리눅스 두 임베디드 운영체제가 각각 설치되어 있어서 좋은 비교가 될 수 있었으며, ISR 및 영상 Display 에 소요되는 시간을 비교하여 이를 표 2 에 나타내었다. 여기에서 사용된 값은 1 시간 동안의 평균 수행시간을 측정하였다.

O/S	ISR 단계 수행시간	Display 수행시간
WinCE	6.5ms	70ms
Linux	6ms	140ms

표 2. Linux 와 WinCE 의 비교 (1 시간 평균)

Linux 에서는 Display 로 Embedded QT 를 이용하였고, 디스플레이모드는 16bit color 상태에서 수행하였다.

ISR 단계에서는 수행시간이 비슷하였지만, Display 에서 많은 차이를 보였다. 이는 아직까지 Embedded-QT 에서는 MQ200 의 2D Accelerator 를 지원하고 있지 않는 것으로 WinCE 와 비교가 성립되지 못하였기 때문이다. 차후 Graphic Accelerator 를 추가하여 비교할 예정이다.

본 논문에서 소개된 독립형 비전모듈은 공장 자동화를 위한 비전처리를 위해 개발된 것이다. 해당 장비가 WinCE 와 Linux 두 운영체제를 각각 설치하여 비교하였을 때, 그 성능은 서로 비슷하였음을 확인 할 수 있었다. 좀더 발전적인 응용을 위해, 리눅스 커널은 네트워크 프로토콜을 완벽하게 지원하므로 여러 독립형 비전모듈로 분산환경을 구성하여 운용할 수도 있고, RT-Linux 를 응용하여 비전처리 뿐만 아니라 실시간 기계의 제어에도 응용할 수 있다.

참고문헌

[1] Intel Crop. SA-1110 Microprocessor Developer's Manual,June 2000. Order no:278240-003

[2] 이명근, 이상정, 조성범, 임재용, "실시간처리 리눅스 기반 VoIP 시스템 설계 및 구현", 정보과학회 2001 년 동계학술발표논문집, pp.345-351, 2001.

[3] 주민규, 최경희, 김종수, 문종욱, 정기현, "내장형 리눅스를 이용한 라우터의 설계 및 구현" 동계정보처리학회논문지(A), pp.339-344, Dec. 2001.

[4] ARM Linux, http://www.arm.linux.org.uk

[5] RT-Linux, http://fsmlabs.com/community/

[6] ARM Linux on Assabet board, http://www-2.cs.cmu.edu/~wearable/software/assabet.html

[7] LCD/CRT 2D Graphics Subsystem Data Book, May 26, 2000. Revision 0.08, http://www.mediaq.com

[8] QT Embedded Free Edition, http://www.trolltech.com/

[9] Daniel P. Bovet and Marco Cesati: Understanding Linux Kernel. O'Reilly, Jan 2001.

[10] Alessandro Rubini and Jonathan Corber: Linux Device Driver 2nd. O'Reilly, Jun 2001.

[11] Vieo4Linux, http://www.video4linux.net/