임베디드 LINUX 시스템 기반 USB 카메라 인터페이스 구현

Implementation of an USB Camera Interface Based on Embedded Linux System

송 성 희*, 김 정 현*, 김 태 효*

Sung-Hee Song*, Jeong-Hyeon Kim*, Tae-Hyo Kim*

요 약

최근 국내외에서 임베디드 시스템 구현에 관한 많은 관심과 개발 경쟁이 한층 심화되고 있다. 지금까지 실시간으로 임베디드 영상획득 및 처리시스템을 구축하는 데에는 현실적 제한이 많았다. 따라서 본 논문에서는 임베디드 LINUX 시스템에 저가의 USB2.0카메라를 이용하여 USB Camera 인터페이스 시스템을 구현하였다. Host2.0 TDI 보드의 디바이스 드라이버를 커널에 탑재하여 USB카메라에서 들어오는 영상 신호를 X-hyper255B로 입력하게 된다. 커널의 디바이스 관리에서 Video4Linux를 이용하여 USB카메라에 대한 정보의 초기 설정이 필요하다. 이렇게 구현한 시스템에서 영상을 획득하고 영상 신호처리를 하게 된다. 처리된 영상 데이터는 네트워크 파일 시스템(NFS)으로 패킷화되어 인터넷으로 전송되고, 인터넷이 접속된 클라이언트 컴퓨터에서 전송된 영상정보를 모니터링 할 수 있음을 확인하였다.

Abstract

In recent, implementation of the embedded system is gradually in the spotlight of world-wide by information technology(IT) engineers. By this time, an implementation of real time system is limited on image acquisition and processing system in practical.

In this paper, the USB camera interface system based on the embedded linux OS is implemented using USB 2.0 camera with low cost. This system can obtain image signals into the memory via X-hyper255B processor from USB camera. It is need to initialize USB camera by the Video4Linux for the kernel device driver. From the system image capturing and image processing can be performed.

It is confirmed that the image data can be transformed to packet of Network File System(NFS) and connected to the internetwork, then the data can be monitored from the client computer connected to the internetwork.

Key words : Embedded Linux System, USB2.0 Camera Interface, Image processing, X-Hyper 255B Processor

I. 서 론

최근에는 시스템 온 칩과 임베디드 시스템(Embedded System)에 대한 많은 관심과 개발로 가전기기를 포함한 모든 제품에 ARM 프로세서, 시스템 온 칩을 적용하여 임베디드 시스템화를 추진하고 있는 추세이다[1][2]. 실로 많은 사람들이 이용하고 있는 휴대폰, PDA 그리고 사이버 아파트의 홈 관리 시스템, 항공관제 시스템, 군사용 제어 장치, 모바일 Post PC 시스템 등 많은 기술들이 이미 내부적으로 임베디드 시스템 기술이 우리 생활과 아주 밀접하게 관련되어 도움을 주고 있다[3][4]. 그러나 현재까지의 임베디드 시스템으로서는 광 대역 신호인 영상신호를 실시간으로 입력하고 처리하는 데는 문제점이 있다. 기존의 USB1.1은 low speed 모드로 1.5Mbps, full

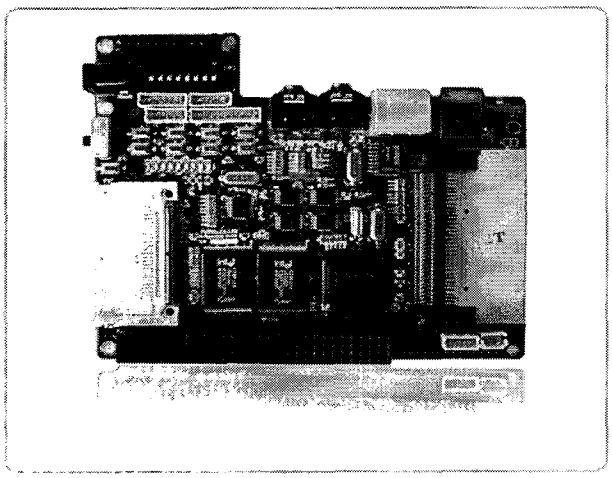
*경남대학교 정보통신공학부
접수 일자 : 2005. 10. 10 수정 완료: 2005. 10. 26
논문 번호 : 2005-4-1

speed 모드로 12Mbps를 지원한다. USB1.1로 영상을 받았을 때 320×240 해상도에서 최대 초당 15프레임 이하이다. 이는 저 해상도이고, 초당 프레임수가 적기 때문에 실시간으로 영상을 처리하는 데는 문제가 있었다.

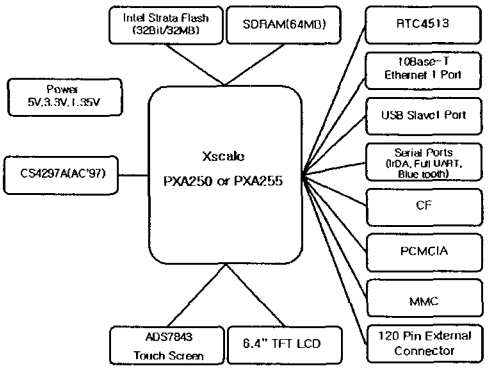
이러한 문제점을 해결하기 위해서 본 논문에서는 USB2.0을 지원하는 임베디드 시스템 기반의 영상 시스템을 구축하였다. 2005년 초, USB2.0을 지원하는 임베디드 시스템이 출시되었고, USB2.0 카메라에서 영상 처리를 위한 시도와 개발이 이루어지고 있다. USB2.0은 low speed 모드와 full speed 모드를 지원하면서, 40배 빠른 최대 480Mbps를 지원하는 high speed 모드가 제공된다. 얻어진 영상은 640×480의 해상도를 가지고 최대 초당 30프레임 이상이다. 임베디드 보드에서 USB2.0을 지원받기 위해서 USB Host2.0 TDI 보드를 장착하고, USB Host2.0 TDI 보드의 디바이스 드라이버를 커널에 탑재한다. QT/Embedded 프로그램으로 카메라에 대한 초기화 설정을 통해 카메라로부터 영상을 획득한다. 이렇게 입력되는 영상신호를 처리한 후 NFS(Network File System)를 이용하여, 호스트 컴퓨터에서 영상 처리한 결과를 화면을 통하여 모니터링 할 수 있다. 본 시스템의 성능을 확인하기 위한 응용 사례에서는 이동물체를 검출하는 알고리듬을 적용하여 그 처리결과를 화면 디스플레이를 통하여 확인하고자 한다.

II. 임베디드 LINUX 시스템

본 논문에서는 그림 1과 같이 hybus사의 X- Hyper255B 보드를 사용하였다[5].



(a) Photograph of X-Hyper255B



(b) Block diagram

그림 1. X-Hyper255B 보드의 구성

Fig. 1. Board configuration of X-Hyper255B

그림 2는 X-Hyper255B 보드의 구체적인 사양을 보여준다.

구 분	항 목	사 양
H/W	Processor	32bit Intel XScale PXA255 (Up to 400MHz)
	Memory	64MB SDRAM Flash Memory 32MB (Intel StrataFlash)
	LAN	10 Base-T Ethernet controller CS8900A
	Audio	AC'97(CS4297)
	Display	6.4 inches TFT LCD (included Touch screen Panel)
	Touch screen	Touch screen Controller ADS7843
	RTC	RTC4513
	IrDA	HSDL3600
	External Interface	Ethernet 1 Port Serial 2 Port JTAG Port USB Slave Port PCMCIA 1 Slot CF 1 Slot MMC 1 Slot 120PIN Connector for GPIO, Address and Data BUS

그림 2. X-Hyper255B의 사양

Fig. 2. Specification of X-Hyper255B

본 논문에서는 USB Host 2.0 지원을 받기 위하여 USB Host2.0 TDI 보드를 설치 하였고 이 보드에 USB카메라를 장착하였다. 다음 그림 3은 USB Host2.0 TDI 보드이다.

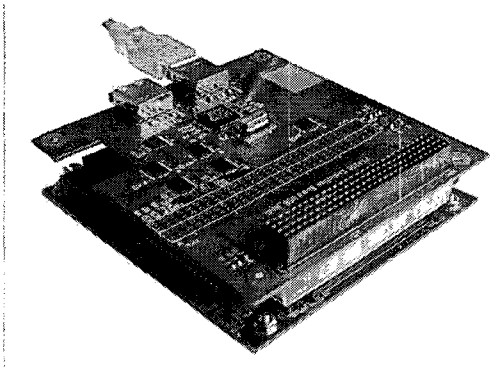


그림 3. USB Host2.0의 TDI 보드

Fig. 3. TDI board of USB Host2.0

Intel XScale PXA255 프로세서는 USB를 Slave 기능만

제공하고, USB Host2.0 TDI 보드는 USB의 Host 기능을 추가하기 위해 개발되었다. USB Host2.0 TDI 보드는 X-Hyper255B, TKU III 보드에 확장하여 사용 가능하다. X-Hyper255B 보드에 USB Host2.0 TDI 보드를 설치하였기 때문에 USB 2.0 기능을 지원하므로 고속으로 데이터 처리를 할 수 있다. 그림 4는 전체 시스템의 구성을 나타낸다.

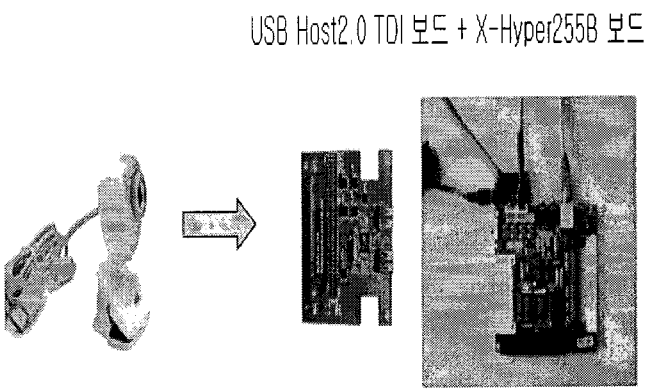


그림 4. 임베디드 시스템의 구조
Fig. 4. Structure of embedded system

USB 카메라를 통하여 디지털 영상신호가 임베디드 보드에 전송된다. 임베디드 보드에서는 영상을 처리하고 그 결과 데이터를 호스트 컴퓨터에 전송해 준다. 따라서 그림 4와 같은 시스템 구성을 통하여 다 채널 영상감시 및 모니터링도 가능하게 된다.

III. USB 카메라 인터페이스

본 장에서는 X-hyper255B와 USB카메라를 연결해 주기 위한 인터페이스를 구현 하고자 한다. 하드웨어적으로 볼 때, X-hyper255B와 USB카메라 사이에는 USB Host2.0 TDI 보드가 연결 되어 있고, 소프트웨어적으로 볼 때는 이 USB Host2.0 TDI 보드의 디바이스 드라이버를 커널에 탑재하여 USB카메라에서 들어오는 영상 신호를 X-hyper255B로 받아 온다. 이렇게 인터페이스를 구현하는데는 커널, Device Drive, Vide Capture 등이 중요한 역할을 한다.

3.1 커널 매칭

커널이란 운영체제의 핵심을 이루는 요소로서 컴퓨터내의 자원을 사용자 프로그램이 사용할 수 있도록 관리하는 프로그램이다. 이런 커널의 역할은 사용자가 작동시키는 응용 프로그램과 하드웨어간의 조정자 역할을 맡는다. 동시에 수행되는 여러 응용 프로그램(엄밀히 말하자면 프로세스들과 쓰레드들이다.)을 위해 메모리 관리를 해 주며, 컴퓨터 자원을 배분하는 역할을 해 준다.

그림 5와 같이 커널은 프로세스 관리, 메모리 관리, 파일 시스템 관리, 디바이스 관리, 네트워크 관리의 5개의 기능 블록으로 구분할 수 있다. 본 논문에서는 기능 블록들 중에서 디바이스 관리 블록에 중점을 두었다.

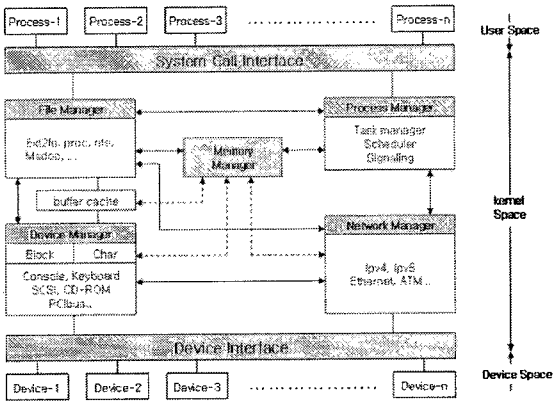


그림 5. Linux 커널의 전체적인 블록도
Fig. 5. block diagram of Linux kernel

디바이스 관리는 디바이스의 입출력 인식(I/O notIfIcation)과 입출력 요청 작업의 스케쥴링(scheduling), 주변장치와 메모리 간의 직접적인 자료 전송(DMA), 디바이스 컨트롤러 관리, 인터럽터 요구 및 핸들링 등의 기능을 수행한다[6].

3.2 Video4Linux API Spec

본 논문에서는 디바이스 관리(Device Manager)에서 Video4Linux를 이용하였다.

표 1. Video4Linux
Table 1. Video4Linux

장치(Device) 파일명	마이너 번호의 범위	기능
/dev/video	0-63	비디오 capture를 위한 인터페이스(Interface)
/dev/radio	64-127	AM/FM 라디오 장비
/dev/vtx	192-223	Teletext 인터페이스(Interface) 칩(Chips)
/dev/vbi	224-239	Raw VBI 데이터(Intercast/Teletext)

Video4Linux는 표 1과 같이 Linux에서 Video 장치(Device)들을 제어하고 사용하기 위한 API들의 모임이다. 비디오 장치들로는 튜너를 가지고 있는 비디오 카드나 PC Web Cam과 같은 장치들을 들 수 있다. 튜너를 가지고 있는 장치들을 제어해서 텔레비젼이나 AM/FM라디

오, TeleText등의 방송을 시청할 수 있다. Video4Linux는 커널에 기본적으로 포함되어 있으므로 따로 설치할 필요가 없다. 이 Video4Linux를 이용해서 X윈도우 상에서 TV를 볼 수 있는 프로그램으로는 xawtv나 xwintv등이 있다[7].

Video4Linux는 다음의 장치(Device) 파일들을 제공한다. 이것들은 보통 "/dev/bttv"로서 알려져 있는 캐릭터형의 장치(Device)이며, 많은 사람들을 위해서 /dev/bttv는 '/dev/video0'에의 기호 연결(Symlink)이 되어 있다.

3.3 Device Driver

디바이스 드라이버는 응용 프로그램이 하드웨어(즉 디바이스)를 제어할 수 있도록 인터페이스를 제공해주는 코드 프로그래머로 하여금 하드웨어에 독립적인 프로그램을 작성하도록 도와준다. 표준화된 system call을 이용하여 디바이스와 정보를 주고받을 수 있도록 하는 역할을 하며 UNIX의 기본적인 특징 중 하나인 추상화 장치를 다루는데 사용한다는 것이다. 모든 하드웨어 장치들은 보통 파일처럼 보이며, 파일을 다루는데 쓰이는 표준 system call과 같이 open, read, write, close 한다. 이러한 디바이스 드라이버는 크게 3가지로 분류가 되며, 각각 문자 디바이스 드라이버, 블록 디바이스, 네트워크 디바이스 드라이버가 있다[8]. 디바이스 드라이버들은 커널에 정적으로 링크 되어서 사용되기도 하며, 때로는 동적으로 운영체제가 가동 중인 상태에서 커널과 링크 되거나 제거 될 수 있으며, 이를 위해 Linux에서는 모듈기능을 제공하여 디바이스 동적 적재, 제거를 가능하게 한다. 다음 그림 6은 디바이스 드라이버 계층도를 나타낸다.

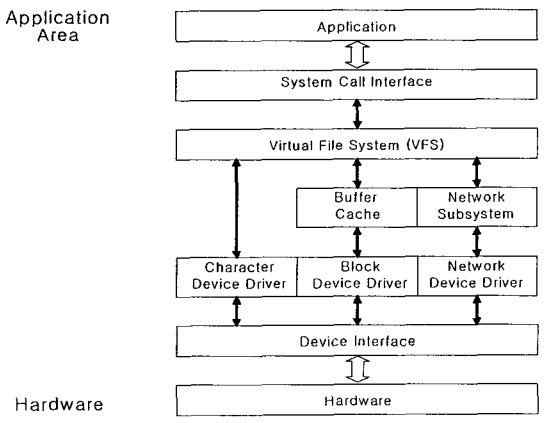


그림 6. 디바이스 드라이버 계층도

Fig. 6. layer diagram of device driver

위 그림 6과 같이 어플리케이션에서는 system call을 통하여 Virtual File System 안의 장치 디바이스를 읽어 디바이스 드라이버에게 명령을 내리며 디바이스 드라이버는 명령에 맞는 함수를 호출하여 하드웨어를 제어 하게 된다.

모듈은 Linux 시스템이 부팅 된 후에 동적으로 load, unload 할 수 있는 커널의 구성 요소이다. 이런 특징으로 커널을 다시 컴파일하거나 시스템을 재부팅하지 않고도 커널의 일부분을 교체할 수 있다. 우리가 익히 들어온 디바이스 드라이버라든지, 파일시스템, 그리고 네트워크 프로토콜 등도 모두 모듈로 만들어져 커널에 포함된 것이다.

본 논문에서는 Device Interface 계층에서 USB 디바이스 드라이버인 OTG파일을 디바이스 드라이브 메모리에 탑재하였다. 이 OTG 파일은 USB 디바이스 드라이버로서 임베디드 Linux 시스템이 USB카메라를 인식할 수 있게 한다. 호환 카메라로부터 카메라 설정을 한 후 이미지 처리 환경을 만들어 주면 영상 신호를 입력 받을 수 있게 된다.

3.4 NFS (Network File System)

임베디드 시스템에서 실시간으로 영상 처리를 할 경우 메모리 사이즈를 고려해야 한다. 1초에 10프레임 이상으로 들어오는 동영상을 저장할 수 있는 메모리 크기에 한계성이 있다. 또한 호스트 컴퓨터에서 실시간으로 모니터링하는 데에도 한계가 있다. 처리한 영상을 클라이언트에게 전송하는 절차 및 고속처리에 있어서도 제한이 많다. 이러한 문제를 해결하기 위해서 본 논문에서는 NFS(Network File System)을 사용하여 해결하였다. NFS는 Linux 머신에서 윈도우 파티션을 마운트하여 사용하듯 NFS서버의 특정 디렉토리를 클라이언트에서 마운트하여 자신의 영역인 것처럼 사용한다. NFS에 접근한 시스템이 있으면 NFS포트에서는 인증을 하고 파일을 주고받는 포트는 새로운 포트를 할당하는 일을 한다. 웹서버나 ftp서버와는 다르게 데몬 프로세스가 특정 네트워크 포트를 점유하면서 동작하여 접속을 대기하고 있는 것이 아니라, portmapper는 111번 포트를 점유하면서 데몬 상태로 대기하고 있다가 NFS 서비스를 요구하는 접속이 들어오면 nfsd(또는 rpc.nfsd)에게 포트 번호를 바꾸어 접속을 연결시켜 준다.

구 분	서 버	클라이언트
호스트명	Linux PC	PXA255
용 도	NFS Server	NFS Client
공유 디렉토리	/nfs	/mnt/nfs
I P	203.253.180.101	203.253.180.102

그림 7. NFS 구성

Fig. 7. NFS configuration

그림 7에서 보는 것과 같이 Linx PC에서는 /nfs라는 디렉토리를 공유시키고, PXA255에서는 nfs라는 명령어로

/mnt/nfs 디렉토리를 마운팅한다[9]. 이렇게 함으로써 LinuxPC에서 /nfs폴더에 hello.c를 저장하면 별다른 설정 없이 PXA255에서는 PXA2555 컴퓨터의 다른 디렉토리처럼 /mnt/nfs폴더에서 hello.c 파일을 쓸 수 있다.

Ⅳ. 영상획득 처리 구현

아래의 내용은 그림 8과 같이 기본 카메라 설정 및 영상 처리 초기 설정을 순서도로 표현한 Capture 순서도이다.

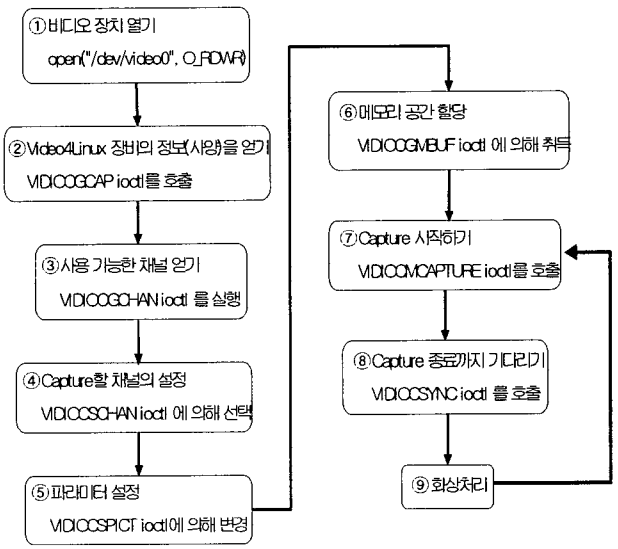


그림 8. 영상획득 순서도
Fig. 8. order diagram of image acqusition

구축한 시스템에 USB카메라의 디바이스 드라이버를 커널에 추가시키고, 임베디드 Linux의 자원을 이용하여 USB카메라의 영상 정보를 추출한다. 비디오 화면을 캡쳐하는 순서는 보통 다음의 순서를 따른다. 다음 그림 9는 효율적인 영상 디스플레이 순서도를 나타낸다. bttv 드라이버와 같이, struct video_mbuf의 멤버 frames가 두개 이상인 경우, capture 처리와 화상 처리를 각각 다른 프레임에 대해서 실시하는 것으로, 이것들을 동시에 실시할 수가 있다. frame 수가 두 개인 경우를 예를 들어 설명한다.

(1) 프레임 #0에 대해서, capture 처리 개시의 ioctl를 호출한다 (VIDIOCMCAPTURE ioctl).
(2) 프레임 #1 에 대해서, capture처리 개시의 ioctl을 호출한다.
(3) 프레임 #0 에 대해서, capture처리가 종료하는 것을 기다린다 (VIDIOCSYNC ioctl).
(4) 프레임 #0 에 대해서, 화상 처리를 한다.
(5) 프레임 #0 에 대해서, capture 처리 개시의 ioctl을 호출한다.
(6) 프레임 #1 에 대해서, capture 처리가 종료하는 것을 기다린다.
(7) 프레임 #1 에 대해서, 화상 처리를 한다.
(8) (2)로 돌아온다.

capture한 버퍼내의 영상데이터를 원하는 목적으로 처리한다. 이 과정에서 화면에 영상을 표시할 수 있다.

본 논문에서는 QT프로그램을 이용하여 화면 디스플레이 및 영상 처리 알고리즘을 적용하였다.

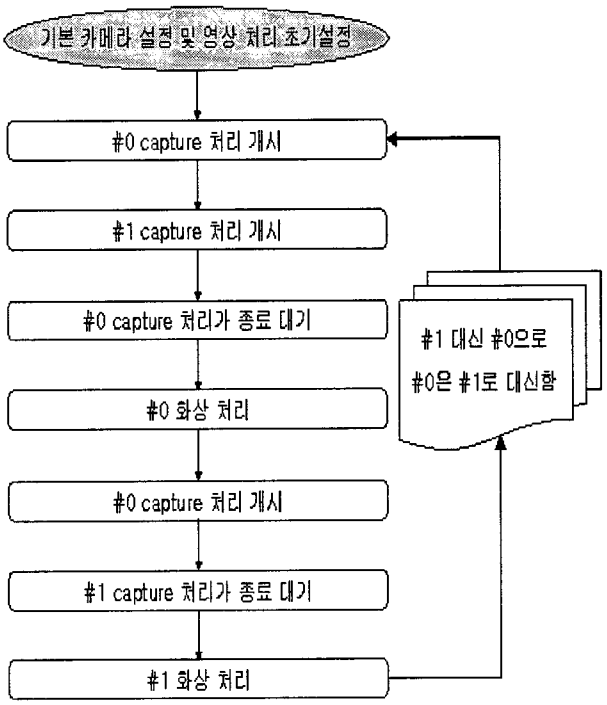


그림 9. 효율적인 영상 디스플레이 순서도
Fig. 9. order diagram of effectiove image display

Ⅴ. 실험 및 결과

임베디드 시스템을 기반으로 USB카메라를 구현하는 연구들은 제약점이 많았다. 앞에서 거론한 것처럼 임베디드 Linux를 이용하여 장비 개발시 개발 환경에 접근하기는 상당히 어렵고, 커널을 비롯하여 마이크로 프로세스(micro process), 네트워크, 각종 툴(tool) 등을 다루어야 하는 내용이 방대함으로 단 시일 내에 전문적인 개발자 양성에 문제가 있다. 임베디드 시스템에서 USB2.0을 지원하는 USB Host2.0 TDI 보드의 경우에는 2005년 초에 상품화 출시되었고, qte-3.3.3도 역시 비슷한 시기에 나왔다. 그러나 본 논문에서는 QT/Embedded(qte-3.3.3)를 이용하여 USB2.0 카메라 인터페이스를 구현하였다. X-Window시리즈를 통한 많은 시도와 개발이 이루어지고 있지만, QT/Embedded는 오픈 소스이기 때문에 소스코드 자체가 공개되어 있고 이의 적용에 따른 로열티 자

체가 없기 때문에 부담 없이 쉽게 자체 개발 아키텍처에 적용할 수 있다. X-윈도우와 비교해 볼때 QT/Embedded는 Linux 커널 자체가 태생적으로 모듈 형식의 코드로 이루어져 있어 쉽게 최소, 최적화가 가능하다[10]. 그리고 실 시간(real time)을 지원하여 지금까지 상용 OS에 열세였던 Real Time 성을 크게 확보하고, 보다 다양한 분야에 적용을 할 수 있다. 이런 장점들 때문에 본 논문에서는 QT/Embedded를 채택했다. 본 장에서는 임베디드 리눅스 환경에서 QT/Embedded를 이용한 Video Capture 실험을 보여 주고 있다.

5.1 적용 사례 구성도

그림 10은 영상 획득 및 영상 처리를 위한 임베디드 시스템의 전체 구성도이다.

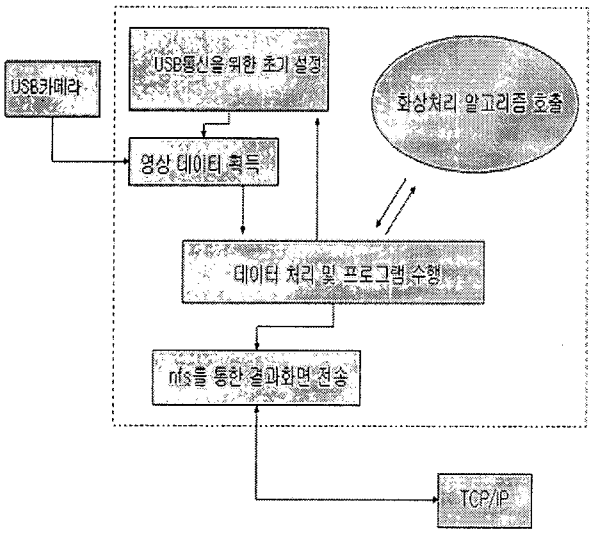


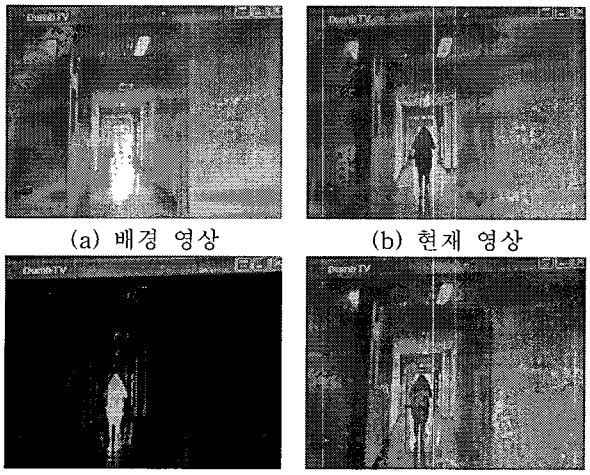
그림 10. X-Hyper255B 보드를 이용한 영상처리 구성도

Fig. 10. image processing diagram using X-Hyper255B

X-Hyper255B 프로세서에서는 USB2.0 카메라에 대한 초기 설정을 하고, USB카메라로부터 영상데이터(디지털 신호)를 획득한다. 이렇게 획득한 데이터를 프로세서에서 화상처리 알고리즘에 따라 정보처리하고 그 결과를 인터네트워크를 통하여 전송할 수 있다.

5.2 적용 사례(객체 추출)

본 논문에서는 그림 11과 같이 움직임 물체 인식을 실험하였다. 잡음이 제거되고 남은 영역에서, 화소 값이 255인 화소 부분이 움직이는 물체로 인식된다. 입력 영상의 밝기 변화에 따른 잘못된 영역추출이나 확대 현상을 줄이기 위해 적응형 임계값 설정[11]을 하였다. 화면 디스플레이를 통하여 카메라 인터페이스 구현이 제대로 작동함을 확인할 수 있다.



(a) 배경 영상 (b) 현재 영상

(c) 차 영상 (d) 이동물체 검출

그림 11. 이동물체 검출 사례

Fig. 11. example of moving object detection

본 논문에서 구현한 시스템에서는 초당 4~5 프레임 정도 화면이 출력 되었다. 이런 결과에 대한 원인으로는 QT/Embedded와 스레드 처리의 낮은 활용도를 들 수 있다.

이 문제를 해결하기 위한 방안으로는 첫째, C코드로 구현한 프로그램과 QT/Embedded로 구현한 프로그램을 각각 비교 분석하는 방법을 생각할 수 있다. 즉 QT/Embedded 프로그램이 OS 자원의 대부분 차지하게 됨으로서 처리 속도가 저하되는 결과를 분석하고, 둘째는 GCC(GNU General Public License)를 이용하여 Frame 버퍼에 영상 데이터를 입력 하였을 때, 속도를 측정하는 방법을 모색하는 것이라 사료된다. 원인 분석 및 충분한 자원 활용을 하는 경우에 초당 20프레임 이상 처리가 가능할 것으로 예상된다.

Ⅵ. 결 론

본 논문에서는 임베디드 Linux 시스템을 기반으로 USB 카메라 인터페이스를 구현하였다. 기존의 PC기반의 영상 감시 시스템은 영상 입력 카메라, 영상처리 보드, PC로 구성되어졌기 때문에 고가이면서도, 그 부피와 활용 면에서 제한이 많았지만, 임베디드 기반의 시스템은 활용 범위와 가격 및 성능 면에서 PC기반 시스템보다 우수함을 알 수 있었다. 그리고 응용 프로그램의 설계 및 제작시 USB카메라에 대한 설정만으로도 영상 획득 및 영상 처리가 가능하여, 기존의 번거롭고 까다로운 설계 과정을 줄일 수 있음을 확인할 수 있었다.

본 논문에서는 임베디드 보드에 1개의 USB 카메라를 설치하여 USB 카메라 인터페이스를 구현하였지만, 향후

연구 과제로서 고해상도 및 고속처리의 다 채널 USB 카메라 인터페이스 구현이 이루어져야 할 것이다.

참 고 문 헌

[1] Jim Turley, "Embedded System by the Numbers", Embedded System Programming, May, 1995.

[2] Don Thomas & Rolf Ernsr (eds.), Proceedings : Fourth International Workshop on Hardware/Software Co-design, IEEE Computer Society, Los Alamitos CA, (1996)

[3] 윤상진, "임베디드의 현재와 미래, 왜 임베디드 시스템인가", 마이크로 소프트웨어, 12월호, pp.239-249. 2000,

[4] 박재호, "IT Expert, 임베디드 리눅스", 한빛미디어, 2003.

[5] HyBus, " 임베디드 리눅스 시스템 ", 2003.

[6] Bernard Cole, "Architectures Overlap Applications", Electronic Engineering Times, March 20, pp.64-65, (1995)

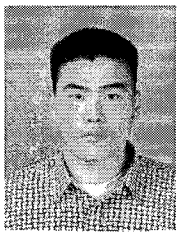
[7] Raj Kamal, "Embedded Systems, Architecture, Programming and Design", Mc-Graw Hill, 2004.

[8] Randal S. Janka, "Specification and Design Methodology for Real-Time Embedded Systems, CMP Books, Nov. 2001

[9] Steve B. Farber, "ARM System-on-chip Architecture 2nd Edition, Addison Ewsley & Benjamin Cummings, 2002

[10] 휴인스 기술 연구소, "Intel PXA255와 임베디드 리눅스 응용", 홍릉과학 출판사, 2004.

[11] Paolo Remagnino, Graeme A. Jones, et al, "Video-Based Surveillance Systems Computer Vision and Distributed Processing", Kluwer Academic Pub., 2002

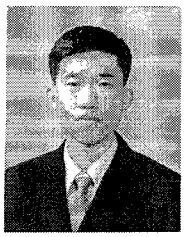


송 성 희(Sung-Hee Song)

2002년 2월 경남대학교 정보통신공학부 졸업(공학사)

2005년 8월 경남대학교 대학원졸업(공학석사)

관심분야 : 임베디드시스템



김 정 현(Jeong-Hyeon Kim)

1999년 2월 경남대학교 전자공학과 졸업(공학사)

2004년 2월 경남대학교 대학원 석사과정 졸업(공학석사)

2004년 3월-현재 경남대학교 대학원박사과정

관심분야 : 영상신호처리, 임베디드시스템

김 태 효(Tae-Hyo, Kim)

1977년 영남대학교 전자공학과 (공학사)

1980년 영남대학교 대학원 전자공학과 (공학석사)

1988년 영남대학교 대학원 전자공학과 (공학박사)

1990년 12월-1991년 1월 미구펜실바니아 대학 Post Doc

1983. 3 - 현재 경남대학교 정보통신공학부 교수

관심분야 : 컴퓨터비전, 영상계측, 영상처리