임베디드 시스템을 위한 MPEG-4 동영상 플레이어 구현

김 수 한[†] · 이 명 원[††] · 장 성 태[†††]

요 약

임베디드 시스템 기술 발전으로 차세대 컴퓨터로서 웨어러블 컴퓨터가 등장하게 되었다. 아직까지 이러한 웨어러블 컴퓨터는 하드웨어와 소프트웨어 성능의 한계로 응용에 제한을 가지고 있다. 본 연구에서는 국내에서 개발한 임베디드 웨어러블 컴퓨터에서 멀티미디어 응용 서비스가 가능하도록 하는 MPEG-4 비디오 플레이어를 구현하였다. 본 논문은 ETRI에서 개발한 손목시계형 웨어러블 컴퓨터인 WPS(Wearable Personal Station) 상에서의 MPEG-4 동영상 플레이어 구현에 대해 기술한다.

키워드 : MPEG-4 동영상 플레이어, 웨어러블 PC, 임베디드 동영상 플레이어, 임베디드 멀티미디어, 임베디드 시스템, 임베디드 응용 SW

Implementation of a MPEG-4 Video Player for Embedded Systems

Soo-Han Kim[†] · Myeong-Won Lee[††] · Seong-Tae Jhang[†††]

ABSTRACT

A wearable computer has appeared as the next generation computer with the development of embedded system technique in our country, although it has not been applicable well since the hardware and software capabilities are limited for multimedia service. We have implemented a MPEG-4 video player to provide with such multimedia application service using the embedded systems. This paper illustrates the development of MPEG-4 video player operating on the wearable computer named WPS(Wearable Personal Station), the wristwatch PC developed at ETRI.

Key Words : MPEG-4 video player, Wearable PC, Embedded video player, Embedded multimedia, Embedded system, Embedded application software

1. 서 론

최근 각광받고 있는 임베디드 시스템은 특정 기능만을 동작시키는데 필요한 장치만으로 구성된 작은 컴퓨팅 시스템을 말한다. 임베디드 시스템은 제작될 때 시스템에서 제공하는 기능이 정해져 있다는 특징이 있다. 임베디드 시스템은 사용자들이 원하는 기능만을 사용할 수 있고, 휴대하기 편리하고 시스템에서 제공하는 기능을 간편한 방법으로 사용할 수 있다는 장점을 갖는다.

이러한 임베디드 시스템의 발전으로 웨어러블 PC, 즉 착용형 PC가 등장하였다. 웨어러블 PC는 사람 몸에 달거나 차고 다니면서 사용자들이 요구하는 기능을 처리하는 PC의 형태로서 최근 유비쿼터스 환경과 더불어서 차세대 PC로 개발되고 있다[1]. 국내에서도 ETRI(한국전자통신연구원)을 중심으로 해서 웨어러블 PC를 개발하고 있다. 웨어러블 PC 상에서의 응용 프로그램 개발도 활발히 전개되고 있으나 아직까지는 하드웨어적 한정된 자원 및 기능으로 응용 개발에 많은 제한이 따른다.

본 논문에서는 ETRI에서 개발된 손목시계형 웨어러블 PC인 WPS (Wearable Personal Station)에서 동작하는 MPEG-4 동영상 플레이어의 구현에 대해 기술한다.

2. 기존의 MPEG-4 동영상 플레이어

MPEG(Moving Picture Experts Group)은 국제표준화기구 ISO/IEC 산하의 동영상 연구그룹으로서 시간에 따라 연속적으로 변화하는 동영상에 대한 압축과 코드 표현을 통해 정보의 전송이 이루어질 수 있는 방법을 연구하고 있다[3, 7]. 미국의 AT&T, 영국의 BT, 일본의 NTT 등의 통신업체 및 후지쓰, 미쓰비시, 픽처텔, 비디오텔리컴 등 화상회의 장비 업체들이 MPEG에 소속되어 있다. MPEG-4는 MPEG-1 및 MPEG-2에서 다루어졌던 자연음 및 영상신호의 압축기술을 더욱 발전시키고, 컴퓨터가 생성하는 합성음 및 영상의 부호 표준을 비롯하여, 멀티미디어 객체를 4차원 가상현실에 배치하고 사용자가 상호 작용할 수 있도록 하는 부호 표준, 전송매체방식에 의존하지 않는 멀티미디어 전송규약 표준, 저작권 관리 및 보호 부호표준 등을 다루고 있다[4, 6, 8]. MPEG-4

† 정 회 원 : 코디콤 연구소
†† 정 회 원 : 수원대학교 컴퓨터학과 부교수
††† 정 회 원 : 수원대학교 IT 대학 컴퓨터학과 부교수
논문접수 : 2005년 12월 13일, 심사완료 : 2006년 12월 1일

는 휴대용 및 인터넷 영상전화, 영상회의, 인터액티브 TV, 인터넷 방송, 가상현실 멀티미디어 플레이어 등에서 많은 수요를 창출할 것으로 예상된다[9][10].

MPEG-4 동영상 플레이어는 멀티미디어용 비디오 압축 기술인 MPEG-4 압축 기술로 압축된 비디오 동영상 파일을 재생할 수 있도록 만들어주는 프로그램이다. MPEG-4 동영상 플레이어는 MPEG-4 동영상에 존재하는 비디오와 오디오 정보를 시스템 상에 존재하는 각각의 출력장치에 알맞게 변환한 후, 각각의 출력장치를 이용해서 사용자들에게 정보를 보내주는 역할을 하게 된다.

MPEG-4 동영상에 존재하는 비디오 정보와 음성 정보를 적절한 형태로 변환하는 작업은 MPEG-4 동영상 플레이어와 연관된 MPEG-4 코덱에서 이루어진다. MPEG-4 코덱은 멀티미디어 동영상 압축 기술로 압축되어있는 MPEG-4 동영상 파일의 정보를 추출할 수 있는 알고리즘을 가지고 있다. 이 알고리즘을 사용하여 동영상을 재생하기 위한 비디오 정보와 오디오 정보를 추출하게 된다. MPEG-4 동영상에서 추출된 비디오와 오디오 정보는 시스템 상에 존재하는 각각의 출력장치로 보내기 위해서 적절한 형태의 신호로 변환된다. 이렇게 변환된 신호들은 각각의 출력장치를 이용해서 사용자들에게 정보를 보내게 된다. (그림 1)은 본 논문에서 구현할 MPEG-4 동영상 플레이어의 전체적인 구조를 나타낸다.

리눅스를 기반으로 동작하는 대표적인 MPEG-4 동영상 플레이어로는 MPlayer와 Xine 등이 있다. MPlayer는 다양한 형태의 MPEG-4 코덱과 파일 포맷을 지원하는 리눅스 기반의 대표적인 MPEG-4 동영상 플레이어이다. MPEG-1/2/4, DivX 3/4/5, Windows Media 7/8/9, Quicktime 5/6, Vivo 1/2 등과 같은 많은 수의 코덱들을 지원하고, RealAudio/Video도 9까지 지원한다.

MPlayer 자체 코덱으로는 MMX/SSE(2)/3DNow (Ex)로 최적화된 자체 오디오 비디오 코덱을 내장하고 있으며, XAim이나 RealPlayer에서 사용되어지는 바이너리 형태의 코덱 플러그인의 지원과 함께 Win32의 코덱 DLL도 코덱으로 사용 가능하다. 기본적으로 VCD/DVD 재생, DVD 자막 뿐만 아니라, 텍스트 기반의 자막 포맷도 다수 지원해준다. 또한 MPlayer에 포함되어 있는 mencoder라는 프로그램을 사용하여 기존의 동영상 파일을 raw/divx/mpeg4 AVI(pcm/mp3 audio)형식으로 변환하는 기능을 제공하며 V4L 장치에서의 비디오 캡쳐도 지원한다.

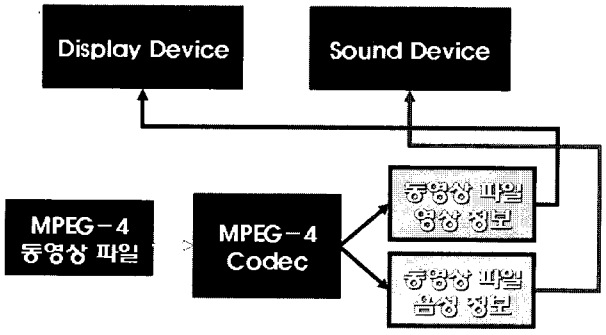
Xine은 리눅스 기반의 무료 MPEG-4 동영상 플레이어로 CD/DVD/VCD를 기본적으로 재생할 수 있으며, 로컬 디스크 드라이브로부터 AVI, MOV, WMV, MP3와 같은 멀티미디어 동영상 파일의 재생도 가능하다. 또한 인터넷을 통한 실시간 멀티미디어 동영상 파일 재생도 가능하다. Xine 플레이어의 장점으로는 xine-lib 라이브러리 형태로 제공하고 있어서 이를 이용하여 xine과 비슷한 MPEG-4 동영상 플레이어를 제작할 수 있다는 점이다.

3. 동영상 플레이어 개발 환경

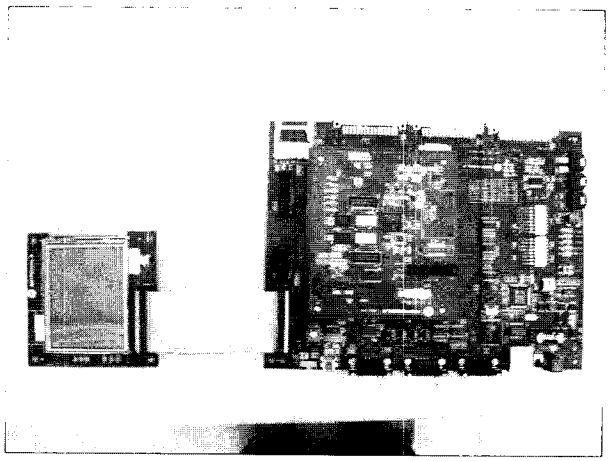
기존의 소프트웨어 개발과정은 소스 코딩을 하고 컴파일하고 실행하는 형태가 동일 PC 환경에서 이루어진다. 그러나 웨어러블 PC와 같은 임베디드 시스템에서의 소프트웨어 개발과정은 소스 코딩을 하고 컴파일 작업은 호스트 PC 라고 불리는 일반 PC 환경에서 이루어지고, 컴파일이 완료된 후 생성된 실행 파일은 크로스 케이블(Cross Cable)을 이용하여 실제로 동작하게 되는 타겟 시스템(Target System)에 전송된다. 즉, 실행 코드를 전송하여 타겟 시스템인 임베디드 시스템에서 실행시키게 되는 것이다. 이 때문에 임베디드 시스템의 소프트웨어 개발자는 타겟 시스템에 대한 지식이 필요하게 된다.

웨어러블 PC 기반의 MPEG-4 동영상 플레이어를 제작하기 위해서는 MPEG-4 동영상 플레이어의 소스 코딩 작업과 컴파일 작업을 하는 호스트 PC가 필요하다. 이러한 작업을 위한 호스트 PC는 Intel Pentium 2.6GHz, 512MB RAM, Linux Redhat 9.0, linux-2.4.20-8 리눅스 커널 버전의 사양을 갖는다[2].

웨어러블 PC기반의 MPEG-4 동영상 플레이어의 실행 환경인 타겟 시스템으로는 웨어러블 PC인 WPS 및 i.MX21 개발 보드가 사용되었다. WPS는 ETRI를 중심으로 개발되고 있는 웨어러블 PC로서 중앙처리장치로는 모토롤라(Motorola) 사에서 개발하고 있는 i.MX21 어플리케이션 프로세서를 사용하고 있다. WPS의 운영체제로는 리눅스 커널 버전인 linux-



(그림 1) MPEG-4 동영상 플레이어의 전체적인 구조



(그림 2) i.MX21 개발 보드

〈표 1〉 i.MX21 개발보드 및 WPS의 성능 비교

	i.MX21 개발보드	WPS
CPU	i.MX21	i.MX21
주메모리	SDRAM 64MB	SDRAM 64MB
시스템 설치	NOR FLASH 32MB	NOR FLASH 32MB
보조 저장 장치	NAND FLASH 32MB	NAND FLASH 32MB
LCD	26K Color TFT-LCD 1.2" 160X128	26K Color TFT-LCD 1.2" 160X128
CMOS Camera	VGA급(30만 화소)	VGA급(30만 화소)
무선 통신	Bluetooth v1.1 via UART4	Bluetooth v1.1 via UART4
Console	UART1(115200bps,8,N,1)	UART1(115200bps,8,N,1)
USB	USB OTG, USB Host	USB OTG
Sound	Microphone/Speaker	none(use Bluetooth Headset)
Buzzer	yes	yes
Vibrator	yes	yes
Battery	LI-Ion 3.7V/300mAh	LI-Ion 3.7V/300mAh
VGA Port	NTSC RCA Jack	none
Adaptor	5V/1A Power Jack	USB OTG 포트 이용

2.4.18-rmk4-mx2bsp를 사용하고 있으며, 리눅스 커널에는 WPS에 있는 장치들을 관리하기 위한 각종 디바이스 드라이버를 포함하고 있다. GUI로는 Trolltech사에서 개발하여 배포하고 있는 qt-embedded-2.3.7과 qt-embedded-2.3.7 상위에서 동작하는 오픈 소스 GUI 환경인 OPIE(Open Palmtop Integrated Environment)를 사용한다. (그림 2)는 모토롤라사에서 개발된 i.MX21 개발 보드의 실제 화면을 나타낸다.

WPS 참조 모델 시스템인 i.MX21 개발 보드는 WPS에서 동작하는 소프트웨어 개발에 필요한 모든 개발 환경을 편리하게 제공하기 위하여 사용되는 보드이다.

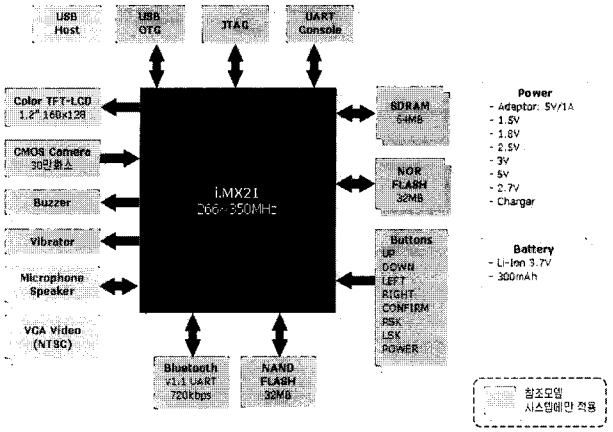
이것은 호스트 PC에서 개발된 소프트웨어를 WPS로 이식하기 전에 소프트웨어가 정상적으로 동작이 되는지 테스트를 하기 위해서 사용하는 보드이다. i.MX21 개발 보드는 WPS와 동일한 중앙처리장치인 i.MX21 어플리케이션 프로세서를 사용하고 있으며 주 메모리의 구성과 Flash 메모리의 구성은 WPS와 동일하다. i.MX21 개발 보드의 운영체제로는 WPS와 동일한 버전의 리눅스 커널이 사용되고 있다. MPEG-4 동영상 플레이어가 동작하게 될 i.MX21 개발보드와 웨어러블 PC인 WPS의 성능을 비교하면 (표 1)과 같다.

호스트 PC에서 WPS 및 i.MX21 개발 보드용 소프트웨어를 개발하기 위해서 제공되는 크로스 컴파일러(Cross Compiler)로는 arm-linux-gcc 3.2.3을 기반으로 하는 ARM Toolchain이 제공된다. ARM Toolchain은 arm용 gcc 컴파일러와 함께 컴파일할 때 사용되는 각종 라이브러리들과 타겟 시스템의 환경에 맞는 실행 파일을 만들기 위한 유틸리티가 들어있다. 이러한 ARM Toolchain을 호스트 PC에 설치하고 이를 사용하여 컴파일해서 WPS 및 i.MX21 개발 보드와 같은 타겟 시스템에서 동작하는 소프트웨어를 제작할 수 있다. (그림 3)은 I.MX21 개발 보드와 WPS 디바이스의 차이를 나타낸다. USB Host/Microphone speaker/VGA Video(NTSC)는 i.MX21 개발 보드에만 있는 장치이며 나머지 장치들은 WPS 및 i.MX21 개발 보드에 공통적으로 포함된다.

4. 웨어러블 PC 기반의 MPEG-4 동영상 플레이어 구현

4.1 MPEG-4 동영상 플레이어의 코덱 라이브러리

코덱 라이브러리에는 MPEG-4 인코딩 알고리즘을 이용해



(그림 3) i.MX21 개발보드 및 WPS의 블록 다이어그램

화상 정보를 인코딩하는 방법과 인코딩된 화상 정보를 디코딩하는 방법이 정의되어 있다. 웨어러블 PC 기반의 MPEG-4 동영상 플레이어를 구현하기 위해서는 우선적으로 MPEG-4 코덱 라이브러리가 웨어러블 PC에서 동작할 수 있도록 만들어 주어야 한다.

본 연구에서 사용되는 FFmpeg 코덱 라이브러리는 기능적으로 libavformat 부분과 libavcodec 부분으로 나누어진다. libavformat에는 MPEG-4 동영상 파일에 대한 구성과 포맷에 대한 정보를 관리할 수 있는 라이브러리 함수와 MPEG-4 동영상 파일에 대한 구성과 포맷에 관련된 자료 구조들이 정의되어 있다. AVFormatContext 구조체는 libavformat 부분에서 정의된 자료 구조들 중 가장 중요한 구조체로서 MPEG-4 코덱에 의해서 인코딩된 MPEG-4 동영상 파일의 각종 정보가 들어가 있다. MPEG-4 동영상 플레이어는 이 AVFormatContext 구조체를 사용하여 재생시킬 MPEG-4 동영상 파일에 대한 모든 정보를 얻을 수 있게 되는 것이다.

libavcodec에는 FFmpeg 코덱 라이브러리에서 지원하는 MPEG-4 코덱에 대한 관리 및 처리를 위한 라이브러리 함수들과 MPEG-4 코덱에 관련된 정보를 저장하기 위한 자료 구조들이 정의되어 있다. AVCodecContext 구조체는 libavcodec에서 정의되어 있는 자료 구조 중 하나이며 FFmpeg 코덱 라이브러리에서 지원하는 MPEG-4 코덱의 종류, MPEG-4 동영상 파일을 인코딩/디코딩하기 위한 각종 정보들을 저장하고 있다. MPEG-4 동영상 플레이어는 MPEG-4 동영상 파일을 디코딩하기 전에 AVCodecContext 구조체에 디코딩 작업을 수행할 수 있는 코덱 정보를 저장한다. MPEG-4 동영상 플레이어는 AVCodecContext 구조체에 저장된 코덱 정보에 따라 MPEG-4 동영상 파일의 디코딩 작업을 수행한다.

libavcodec 부분에 정의되어 있는 라이브러리 함수는 MPEG-4 동영상 플레이어에서 동작하는 실제적인 작업들을 포함하고 있으며 타겟 시스템에 종속적으로 동작한다. 예를 들어, 음성 정보의 인코딩/디코딩 작업을 수행할 경우에는 음성과 관련된 장치들의 동작에 대한 정의를 제공받아 효율적이고 안정적인 인코딩/디코딩 작업을 수행하게 된다. libavcodec의 라이브러리 함수 중 타겟 시스템의 동작에 관련

된 정의가 들어있는 부분을 웨어러블 PC 환경에 알맞게 변경해야 하며 이 작업은 MPEG-4 동영상 플레이어를 만들기 전에 우선적으로 해주어야 한다.

4.2 MPEG-4 동영상 플레이어의 비디오 출력

MPEG-4 동영상 플레이어의 역할은 MPEG-4 동영상 파일에 인코딩되어 있는 비디오/오디오 스트림 정보를 MPEG-4 코덱 라이브러리를 이용해 디코딩한 후에 시스템 출력 장치를 이용해서 출력하는 것이다. (그림 4)는 MPEG-4 동영상 플레이어가 화상/음성 정보를 출력하는 전반적인 실행 과정을 나타낸 것이다.

MPEG-4 동영상 플레이어의 비디오 출력 부분은 MPEG-4 동영상 파일에서 화상 정보를 얻는 부분과 얻은 화상 정보를 프레임버퍼를 이용하여 시스템 출력 장치에 출력하는 부분으로 나뉜다. 일반 PC의 경우에는 디스플레이 어댑터가 프레임버퍼에 해당되며 웨어러블 PC에서는 LCD 컨트롤러가 이에 해당된다. 프레임버퍼 드라이버에 의해 그래픽 하드웨어의 관리 및 제어 뿐만 아니라 그래픽 정보를 저장하기 위한 메모리 영역의 관리가 수행된다.

프레임버퍼를 이용하여 화면에 출력하는 응용프로그램에서는 /dev/fb 형태로 존재하는 디바이스 노드에 저수준 파일입출력 함수를 이용하여 디스플레이 장치에 출력할 데이터를 쓰면 프레임버퍼를 통하여 디스플레이 장치로 출력되는 것이다. 응용프로그램에서 프레임버퍼를 사용하여 디스플레이 장치로 데이터를 출력하는 방법은 (그림 5)와 같다.

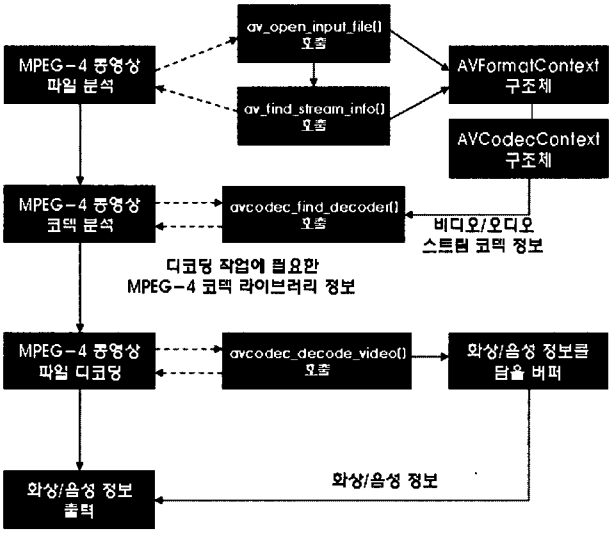
프레임버퍼를 이용하여 화면에 그래픽 데이터를 쓰는 방법에는 크게 두 가지가 존재한다. 저수준 파일입출력 함수인 write( ) 시스템 함수를 이용하는 방법과 mmap( ) 시스템 함수를 이용하는 방법이 있다. 저수준 파일입출력 함수인 write( ) 시스템 함수를 이용해서 그래픽 정보를 디스플레이 장치의 특정 위치에 출력하기 위해서는 lseek( ) 시스템 함수를 이용하여 그래픽 정보를 디스플레이 장치의 출력하고 싶은 특정 위치까지 파일 포인터를 옮긴 후에 write( ) 시스템 함수를 이용하여 그래픽 정보를 프레임버퍼에 출력한다. 이와는 달리 mmap( ) 시스템 함수는 프레임버퍼와 대응되는 메모리 영역을 할당하는 방법이다. 이것은 할당받은 메모리 영역에 그래픽 정보를 저장하면 이 그래픽 정보가 프레임버퍼를 통해서 디스플레이 장치에 출력되게 하는 방법이다.

본 논문에서 구현하는 웨어러블 PC기반의 MPEG-4 동영상 플레이어는 효율적인 그래픽 처리를 위하여 mmap() 시스템 함수를 사용하여 MPEG-4 동영상 파일에 있는 화상 정보가 시스템의 디스플레이 장치로 출력되도록 하였다. MPEG-4 동영상 플레이어의 비디오 출력의 구현을 위해서는 MPEG-4 동영상 파일에서 비디오 스트림을 찾은 후에 MPEG-4 코덱을 이용해서 디코딩하여 화상 정보를 얻는다. 이렇게 해서 얻은 화상 정보는 프레임버퍼를 이용하여 웨어러블 PC의 디스플레이 출력 장치에 출력되도록 한다. (그림 6)는 MPEG-4 동영상 플레이어의 비디오 출력 부분의 구현을 나

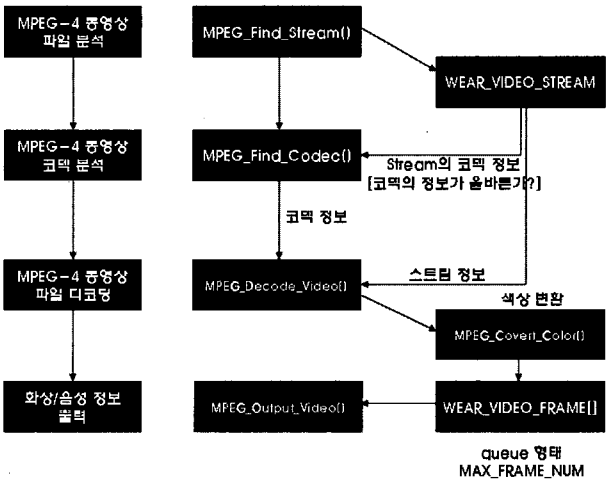
```
#define FBDEVFILE "/dev/fb"

int main()
{
   int fbfd = 0;
   .....
   .....
  // 프레임 버퍼 open...
  fbfd = open(FBDEVFILE, O_RDWR);
  if(fbfd < 0){
     perror("fbdev open");
     exit(1);
}
   // 프레임 버퍼 write...
   write(fbfd, "프레임버퍼에 Write할 내용",
         sizelen);
   ....
}
```

(그림 5) 리눅스에서 프레임버퍼를 사용한 화면 출력 프로그램



(그림 4) MPEG-4 동영상 플레이어의 전반적인 실행 과정



(그림 6) MPEG-4 동영상 플레이어의 비디오 출력 부분 구현

```
// MPEG-4 동영상 플레이어의 MPEG-4 동영상 파일 확인 작업

AVFormatContext *pFormatCtx;
const char      *filename;

// filename에 재생할 MPEG-4 동영상 파일 이름을 넣는다.

// MPEG-4 동영상 파일 open
if(av_open_input_file(&pFormatCtx, filename, NULL, 0, NULL)!=0){
// 오류 처리 부분
}
if(av_find_stream_info(pFormatCtx)<0){
// 오류 처리 부분
}
int     Loopcount, Videostream;

// Find the first video stream
videoStream=-1;
for(i=0; i<pFormatCtx->nb_streams; i++)
    if(pFormatCtx->streams[i]->codec.codec_type==CODEC_TYPE_VIDEO)
    {
        Videostream=i;
        break;
    }
if(Videostream==-1){
// Video Stream을 찾지 못했음
// 오류 처리 부분
}
```

(그림 7) MPEG__Find__Stream( ) 함수 구현 부분

타낸 것이다.

MPEG-4 동영상 파일에 있는 비디오 스트림을 찾는 작업은 FFmpeg 코덱 라이브러리의 libavformat 부분에 존재하는 라이브러리 함수와 AVFormatContext 구조체를 사용해서 수행한다(그림 7). 우선 av_open_input _file( ) 함수를 호출하여 MPEG-4 동영상 파일을 열어서 MPEG-4 동영상 파일에 있는 헤더 정보를 WEAR_VIDEO_STREAM 구조체에 있는 AVFormatContext 구조체에 저장해준다. WEAR_VIDEO_STREAM 구조체에는 MPEG-4 동영상 플레이어의 MPEG-4 동영상 파일의 재생을 쉽게 할 수 있도록 AVFormatContext 구조체와 MPEG-4 동영상 파일을 구성하는 비디오/오디오 스트림의 총 개수와 함께 비디오/오디오 스트림을 저장할 수 있는 스트림 데이터 버퍼가 있다. 이 구조체는 MPEG-4 동영상 플레이어에서 재생할 때 필요한 모든 정보를 총괄적으로 관리하는 구조체이며, 이 후 작업이 효율적으로 관리될 수 있도록 만들어준다.

MPEG-4 동영상 파일에서 비디오 스트림 검색 작업이 완료되면, 검색된 비디오 스트림 정보를 이용하여 화면에 출력할 화상 정보를 얻는 작업을 수행한다. 우선적으로 비디오 스트림 정보를 디코딩할 수 있는 코덱을 찾아야 한다. WEAR_VIDEO_STREAM 구조체의 비디오 스트림 데이터 정보에는 해당 스트림을 디코딩할 수 있는 기본적인 코덱 정보를 포함하고 있는 것이 특징이다. 이 정보는 libavcodec 부분에 정의되어 있는 AVCodecContext 구조체에 저장되며, MPEG-4 동영상 플레이어는 AVCodecContext 구조체의 정보를 이용해 디코딩 작업에 필요한 코덱을 찾는다.

MPEG-4 동영상 플레이어는 우선적으로 비디오 스트림에 있는 AVCodecContext 구조체를 비디오 스트림에서 가지고 온다. MPEG-4 동영상 플레이어는 avcodec _find_decoder( ) 함수를 호출하여 비디오 스트림을 디코딩할 수 있는 코덱을 찾는다. 이 작업이 완료되면, MPEG-4 동영상 플레이어는 MPEG-4 동영상 파일을 실제적으로 디코딩 할 수 있게 되는 것이다.

실제적인 디코딩 작업을 수행하기 전에 화상 정보를 저장할 수 있는 WEAR_VIDEO_FRAME라는 자료 구조를 정의하고 할당한다. WEAR_VIDEO_FRAME 구조체는 MPEG-4 코덱에 의해서 디코딩된 화상 정보를 저장하는 버퍼와 해당 화상 정보의 순서를 저장하는 변수로 구성되어 있다. 이것은 사용자가 원하는 화상부터 재생할 수 있는 기능을 지원하고, MPEG-4 동영상 파일이 재생될 때 끊김 현상이 발생하지 않도록 버퍼링 기능을 제공한다.

MPEG-4 동영상 플레이어는 libavcodec 부분에 정의되어 있는 라이브러리 함수인 avcodec_decode_video( ) 함수를 호출하여 비디오 스트림을 디코딩하여 화상 정보를 얻게된다. avcodec_decode_video( ) 함수는 MPEG-4 동영상 파일을 디코딩하여 Y:U:V 색상 방식으로 표현된 화상 정보를 AVFrame이라는 구조체에 저장하게 된다. (그림 8)은 위에서 설명한 일련의 과정을 나타낸 것이다.

MPEG-4 동영상 플레이어는 AVFrame 구조체에 저장된 화상 정보를 R:G:B 색상 방식으로 변환해서 WEAR_VIDEO_FRAME

```
// MPEG-4 코덱과 관련된 처리 부분

AVCodecContext *pCodecCtx;
// 비디오 스트림을 위한 AVCodecContext 구조체의 포인터를 얻음
pCodecCtx=&pFormatCtx->streams[videoStream]->codec;
AVCodec *pCodec;
// 비디오 스트림 처리를 위한 코덱을 찾는다.
pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
if(pCodec==NULL){
 // 알맞은 코덱을 발견하지 못했을 경우
 // 오류 처리 부분
}
// 찾은 비디오 코덱을 open한다.
if(avcodec_open(pCodecCtx, pCodec)<0){
  // 비디오 코덱을 open하지 못했을 경우에는...
 // 오류 처리 부분
}
AVFrame *pFrame;
pFrame=avcodec_alloc_frame();
....
r = avcodec_decode_video(pCodecCtx, pFrame, &frameFinished, rawData,
                         bytesRemaining);
....
```

(그림 8) MPEG-4 코덱 처리 부분

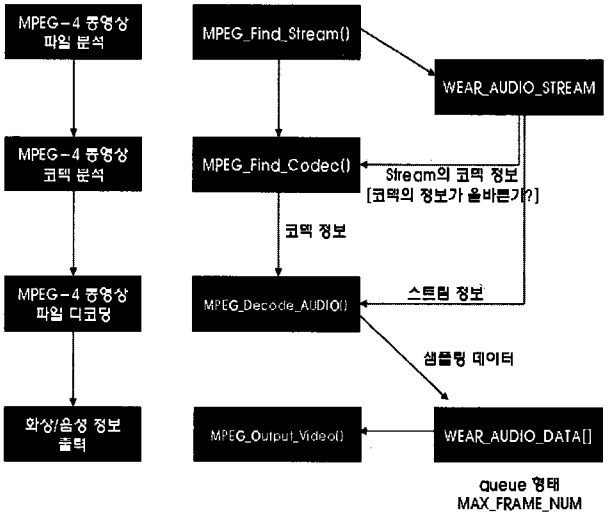
구조체의 화상 정보를 저장하는 버퍼에 저장되며, 이 화상 정보가 프레임버퍼를 이용하여 웨어러블 PC의 디스플레이 출력 장치에 표현되는 것이다.

4.3 MPEG-4 동영상 플레이어의 사운드 출력

웨어러블 PC의 사운드 처리는 DSP(Digital Signal Processor)에서 처리된다. DSP는 아날로그 신호를 A/D(아날로그/디지털) 변환기를 이용해서 얻어진 디지털 데이터에 대수적인 연산을 통해 필터링이나 스펙트럼 분석 등의 신호 처리를 하는 것을 말한다. 기본적으로 아날로그 신호의 실시간 디지털 처리를 목적으로 한다. 이러한 DSP는 웨어러블 PC에 공통프로세서(Coprocessor) 형태로 존재하며, 웨어러블 PC의 주 프로세서에는 DSP를 위한 명령어와 데이터들이 존재한다. 리눅스에서는 디바이스 노드를 이용하여 DSP를 관리하며, 응용프로그램에서는 이 디바이스 노드를 이용해서 DSP를 제어할 수 있다. 이것은 /dev 디렉토리에서 /dev/dsp 형태로 존재한다.

MPEG-4 동영상 플레이어에서 사운드 출력 부분의 구현은 MPEG-4 동영상 파일에 압축되어 있는 오디오 샘플링 데이터를 MPEG-4 코덱을 이용해서 추출하여 얻어진 오디오 샘플링 데이터를 DSP로 보내서 출력하는 것이다. 전체적으로 비디오 출력 부분의 구현과 상당히 유사하며, 출력 데이터를 위한 자료 구조 이외에는 동일한 자료 구조를 사용하기 때문에 사운드 출력 부분의 구현은 어렵지 않게 구현된다. (그림 9)는 사운드를 출력하는 과정을 나타낸 것이다.

MPEG-4 동영상 파일을 열어서 전체적인 MPEG-4 동영상 파일의 스트림 정보를 가지고 오는 것은 비디오 출력 부분의 설명과 같다. 이 때, MPEG-4 동영상 파일에서 재생 가능한 오디오 스트림 정보가 있는지 확인한다. (그림 10)은 MPEG-4 동영상 파일에서 비디오/오디오 스트림을 찾는 알고리즘을 나타내고 있다.



(그림 9) MPEG-4 동영상 플레이어의 오디오 출력 부분 구현

MPEG-4 동영상 파일에서 재생 가능한 오디오 스트림을 찾은 후에는, 이로부터 샘플링 데이터를 추출할 수 있는 MPEG-4 코덱을 찾아야 한다. 이 부분도 비디오 출력에서와 비슷하게 구현되며, (그림 11)은 샘플링 데이터 추출을 위한 MPEG-4 코덱을 찾는 알고리즘을 나타내고 있다.

avcodec_decode_audio() 함수는 동영상 파일의 스트림 정보에서 사운드 출력을 위한 샘플링 데이터를 얻는 함수이다. 이 함수를 통해 얻는 샘플링 데이터는 특별한 변환과정이 필요 없으며, 웨어러블 PC의 사운드 출력 장치인 DSP를 이용해서 사용자에게 출력된다.

(그림 12)는 본 논문에서 구현된 MPEG-4 동영상 플레이어에서 실제로 동영상 파일을 재생하는 모습이다.

본 논문에서 구현된 MPEG-4 동영상 플레이어는 한정된

```
// MPEG-4 동영상 파일에서 재생가능한 Video/Audio stream을 찾는다.
Videostream=-1;
Audiostream=-1;
for(i=0; i<pFormatCtx->nb_streams; i++){
    if(pFormatCtx->streams[i]->codec.codec_type==CODEC_TYPE_VIDEO)
    {
        Videostream=i;
        if(Videostream != -1 && Audiostream != -1)  break;
    }
    if(pFormatCtx->streams[i]->codec.codec_type==CODEC_TYPE_AUDIO)
    {
        Audiostream=i;
        if(Videostream != -1 && Audiostream != -1)  break;
    }
}
if(Videostream==-1){
// Video Stream을 찾지 못했음
// 오류 처리 부분
}
if(Audiostream==-1){
// Audio Stream을 찾지 못했음
// 오류 처리 부분
}
```

(그림 10) 비디오/오디오 스트림 분석

```
// MPEG-4 코덱과 관련된 처리 부분

AVCodecContext *pAudioCodecCtx;
// 오디오 스트림을 위한 AVCodecContext 구조체의 포인터를 얻음
pAudioCodecCtx = &pFormatCtx->streams[AudioStream]->codec;
AVCodec *pACodec;
// 오디오 스트림 처리를 위한 코덱을 찾는다.
pACodec = avcodec_find_decoder(pAudioCodecCtx->codec_id);
if(pACodec==NULL){
// 알맞은 코덱을 발견하지 못했을 경우
// 오류 처리 부분
}
// 찾은 오디오 코덱을 open한다.
if(avcodec_open(pAudioCodecCtx, pACodec)<0) {
  // 오디오 코덱을 open하지 못했을 경우에는
  // 오류 처리
}
...
r = avcodec_decode_audio(pAudioCodecCtx, sampledata, &gotsampledata,
                         rawData, ByteRemaining);
```

(그림 11) MPEG-4 오디오 처리 부분

(그림 12) MPEG-4 동영상 재생 화면

성능의 웨어러블 시스템에서 동작 가능하도록 MPEG-4 동영상 플레이어를 효율적으로 구현했다는 점에서 의미가 있다. 이것은 다른 일반 MPEG-4 동영상 플레이어와는 달리 리눅스 기반의 저사양 임베디드 시스템에서도 일반 컴퓨터에서와 동일한 기능으로 MPEG-4 동영상 파일을 재생할 수 있다.

5. 결 론

본 논문에서는 웨어러블 PC기반의 MPEG-4 동영상 플레이어 개발에 대해 기술하였다. 이 시스템의 기능은 웨어러블 PC에 부착되어 있는 디스플레이 출력 장치와 사운드 출력 장치를 이용하여 MPEG-4 압축 기술로 압축된 비디오

동영상의 영상 정보와 음성 정보를 출력해주는 것이다. MPEG-4 압축 기술로 압축된 비디오 동영상의 정보를 알맞게 출력하기 위해서는 출력에 알맞은 형태로 변환하기 위해서 MPEG-4 코덱이 필요하게 되는데, 본 논문에서는 리눅스용 MPEG-4 코덱으로 많이 사용되는 FFmpeg 코덱 라이브러리를 이용하였다.

기존의 MPEG-4 동영상 플레이어와는 달리 임베디드 시스템을 위한 전용 코덱으로 사용할 수 있도록 시스템의 효율성을 높여서 저사양의 시스템에서도 MPEG-4 동영상 파일이 충분히 재생할 수 있도록 한 것이 특징이다.

앞으로는 웨어러블 PC와 같은 모바일 단말기에서 서버의 멀티미디어 정보를 검색하여 재생시킬 수 있도록 MPEG-4 플레이어 기능을 확장할 계획이다.

참 고 문 헌

[1] Joanna Lumsden and Stephen Brewster, "A paradigm shift: alternative interaction techniques for use with mobile & wearable devices," Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research, IBM Press, pp.197-210, 2003.

[2] Intel XScale Core Developer's Manual

[3] ISO/IEC 14496-1, "Coding of Moving Pictures and Audio," ISO/IEC JTC1/SC29/WG11 N4699, Chapter 14-17, March 2002.

[4] ISO/IEC JTC1/SC29/WG11 N1683, "MPEG-4 Overview," April 1997

[5] Matthias Kalle Dalheimer, Programming with Qt, second ed., O'REILLY, 2002.

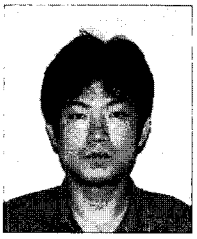
[6] ISO/IEC JTC1/SC29/WG11 N1683, "MPEG-4 Overview," April 1997

[7] ISO/IEC JTC1/SC29/WG11, Information technologyCoding of audio-visual objects Part 1: Systems, ISO/IEC 14496-1: 2001.1

[8] MPEG-4 DMIF Group, "Generic Coding of Audio-Visual Object : Part6 DMIF"

[9] Boughoufalah S., Dufourd J-C., Bouilhaguet F., "MPEG-Pro, an Authoring System for MPEG-4 with Temporal Constraints and Template Guided Editing," Proceedings of ICME2000, Vol.1, pp.175-178, 2000.

[10] Petros daras, Ioannis Kompatsiaris and Theodoros Raptis, "An MPEG-4 Tool for Composing 3D Scenes," IEEE Multimedia, pp.58-71. April-June 2004.



김 수 한

e-mail : gruf97@hanmail.net

2001년 수원대학교 컴퓨터학과 (학사)

2003년 수원대학교 대학원 컴퓨터학과 (석사)

2006년 수원대학교 대학원 컴퓨터학과 박사과정 수료

2007년~현 재 코디콤 연구소 재직 중

관심분야 : 컴퓨터구조, 다중 프로세서 시스템, 임베디드시스템 설계

이 명 원

e-mail : mwlee@suwon.ac.kr

1981년 서울대학교 (학사)

1984년 서울대학교 대학원 계산통계학과 전산전공 (석사)

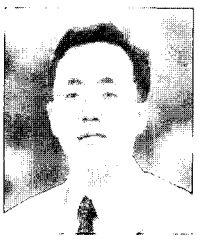
1990년 일본 동경대학(The U. of Tokyo) 대학원 정보과학과 (박사)

1990년~1993년 일본 Kubota Corp. 및 동경대학(The U. of Tokyo) 연구원

1993년~1996년 KT 멀티미디어연구소 선임연구원

1996년~현 재 수원대학교 IT대학 인터넷정보공학과 부교수

관심분야 : 컴퓨터그래픽스, 가상현실, 멀티미디어



장 성 태

e-mai : stjhang@suwon.ac.kr

1986년 서울대학교 컴퓨터공학과 (공학사)

1988년 서울대학교 대학원 컴퓨터공학과 (석사)

1994년 서울대학교 대학원 컴퓨터공학과 (박사)

1994년~현 재 수원대학교 IT 대학 컴퓨터학과 부교수

관심분야 : 다중 프로세서 시스템, 컴퓨터 구조, 병렬 처리, 캐쉬 구조, 메모리 모델