홍채인식 시스템을 위한 임베디드 시스템의 설계 및 구현

Design and Implementation of Embedded LINUX-Based System for Iris Recognition System

임철수
서경대학교 컴퓨터공학과
박병섭
인하공업전문대학 컴퓨터정보공학부

중심어 : 홍채인식 시스템, 임베디드 리눅스

요 약

본 논문에서는 홍채인식 상용제품에 적용하기 위한 임베디드 시스템의 운영체제로서 리눅스를 대상으로 하여 홍채인식 시스템을 위한 UI 보드를 개발하고, 구현된 보드에 리눅스를 포팅v하였다. 이를 위하여 구현된 UI 보드에 맞게 LINUX 전체시스템을 설계하고 부트로더, 커널, 제어 프로그램 등을 이에 맞도록 정합시켰다. 실험 결과, 홍채인식시스템은 정상적으로 동작하고, TCP/IP로 연결된 UI 보드는 타 시스템과 성공적으로 데이터를 실시간 송•수신함으로써 제품의 전체 기능상 신뢰성 있는 사용자 인터페이스 기능을 구현할 수 있었다.

Cheol-Su Lim (cslim@skuniv.ac.kr)
Dept. of Computer Engineering, Seokyeong University
Byoung-Seob Park (bspark@inhatc.ac.kr)
School of Computing & Information systems,
Inha Technical College

Keyword : Iris Recognition System, Embedded Linux

Abstract

In this paper, we implemented embedded LINUX-based UI(User Interface) board which can be applied to Human Iris recognition product. For this purpose, we also analyzed and designed LINUX operating system and adapted boot loader, kernel, control program modules according to the developed H/W architectures. As the experimental results shows that Iris recognition system is operable and embedded LINUX-based UI board which is connected to heterogeneous system by TCP/IP protocol can both successfully send and receive data, this UI board has been able to obtain high performance.

I. 서론

최근 전자상거래 활성화와 9.11 테러, 빈번히 발생하는 금융사고 등에 힘입어 생체인식기술에 대한 사회적 관심이 높아지고 있다. 생체인식이란 인간의 신체적/행동적 특징을 이용하여 개인의 신원을 확인/인증하는 자동화 기술로서 이러한 개인 특성은 절도나 누출될 수 없으며 변경되거나 분실할 위험성도 없으므로, 이러한 기법을 사용할 경우 신원확인 및 정보보안이 완벽하게 구축될 수 있다[1].

신뢰성 있는 생체인식 시스템은 빌딩의 출입자 관리 시스템에 이용될 뿐 아니라, 은행의 금고와 같은 고도의 주의가 요구되는 경우, 회사나 공항의 보안 시스템, 전자상거래 인증, 현금카드 비밀번호 대체, 신용카드 결재 등 일반적인 활용 범위를 가지려는 추세이며, 보다 적용이 용이하면서 더욱 높은 신뢰도를 갖게 하려는 많은 연구가 이루어지고 있다[1],[2].

생체인식 시스템을 구현하는데 있어서 주로 사용되는 인간의 특징으로서는 홍채(Iris), 지문(Fingerprint), 얼굴(Face), 장문(Palmprint), 손모양(Hand geometry), 열상(Thermal image), 음성(Voice), 필체(Signature), 혈관(Vein), 타이핑(Typing, keystroke dynamics), 망막(Retina) 그리고 유전자 형질 등이 있다[3],[4].

한편 컴퓨터, 통신 및 전자기술들이 급속도로 발전하면서 통신기술을 이용한 다양한 제품 및 서비스들이 일상생활 주변에 널리 사용되고 있다. 또한, 기존의 PC이외에 일상생활

접수번호 : #030811-001
접수일자 : 2003년 8월 11일, 심사완료일 : 2003년 9월 16일

* 교신저자 : 임철수, e-mail : cslim@skuniv.ac.kr

에서 사용되고 있는 백색가전 제품(TV, 냉장고, 세탁기, 전자레인지 등) 뿐만 아니라, 이동전화기, PDA 그리고 홈 오토메이션, 교통, 주차관리시스템, 그리고 각종 산업기기 등과 같은 다양한 분야에 다양한 제품들을 주변에서 쉽게 접할 수 있는 실정이다. 이러한 기술적, 시대적 배경을 바탕으로 임베디드 시스템(Embedded System) 이란 상기 제품들과 같이 특정 제품에서 특정 기능을 수행하기 위한 전자 및 컴퓨터 시스템을 의미한다[5]. 아울러, 각종 가전제품 및 제어장치가 전기·전자 회로로만 구성된 것이 아니라, 마이크로 프로세서가 내장되어 있어 특정한 기능을 수행하도록 설계된 시스템을 총칭하여 임베디드 시스템이라고 한다. 이에 따라 기존에 단순한 목적으로 개발된 제어 프로그램은 일부 특수한 목적에만 맞추어서 개발되었으므로 오늘날과 같은 다양하고 복잡한 정보통신 제품 및 서비스를 구현하기에는 한계성을 드러내게 됨으로써, 이를 만족할 수 있는 요구가 나오기 시작하였다. 이와 같이 임베디드 시스템의 요구사항이 복잡해지고 다양해짐에 따라 이를 효과적으로 수용할 수 있는 저가격·고성능 하드웨어가 필요하고, 임베디드 시스템 개발을 단축하기 위하여 응용 프로그램을 쉽게 작성할 수 있는 환경이 필요하다.

이에 따라 본 논문에서는 생체인식 보안 분야중 최고의 정확도 및 신뢰성을 보장할 수 있는 홍채인식 방식을 기술하고, 홍채인식기 제품에 적용할 임베디드 리눅스 기반 하드웨어 UI(User Interface) 보드 설계 기술 및 개발된 전용보드에 리눅스를 탑재하여 DSP보드와 어떻게 홍채 특징 데이터를 주고 받는지에 대한 개발 사례를 중심으로 각 주요 구성 요소별 개발 내용을 기술하고자 한다. 이에 따라 2장에서는 홍채인식 절차에 대해 기술하고, 3장에서는 전체 시스템 구조를 설명한다. 다음으로는 각 구성요소별 핵심 기능 및 개발 내용을 설명하며, 마지막으로는 전체 하드웨어 UI 보드와 DSP 보드와의 통신 실험결과 등을 제시하고자 한다.

본 연구에서와 같은 홍채인식 시스템용 리눅스 기반의 임베디드 시스템은 최초의 시제품 개발이고, 현재 이와 우사한 LG 전자의 PC 버젼 제품은 상용화 되어 있다. 본 연구개발의 대상이 되는 홍채인식용 타겟(Target) 보드 시스템인 타이눅스 박스(Tynux Box)는 팜팜테크㈜에서 개발한 임베디드 리눅스인 타이눅스(Tynux)를 기본 운영체제(OS)로 한다. 타이눅스 박스는 국내에서 최초로 개발된 임베디드 리눅스 시스템 개발도구로 PDA, 웹 패드, 전자책 등의 포스트-PC(Post-PC) 제품과 무선 PDA, IMT-2000단말기 등의 제품 개발에 레퍼런스 보드(Reference Board)로 활용될 수 있다[3].

II. 홍채인식 시스템

1. 홍채인식 처리절차

인식 홍채(Iris)는 사람 눈의 동공과 흰 부위 사이에 존재하는 영역이다. 이 홍채에 생겨 있는 긴 띠 모양의 망(빗살무늬의 인대), 코라지를 한 듯한 붉은 색의 섬유질, 속눈썹 모양의 돌기, 꾸불꾸불한 혈관계, 링 모양의 원들, 동공을 둘러싸는 코로나모양의 인대, 홍채 고유의 색, 얼룩점 등이 각 사람마다 다른 생물학적 특성을 가지며, 생후 어느 정도의 기간이 흐른 뒤에는 홍채의 특징이 변하지 않는다(그림 1 참조). 이러한 홍채의 특징은 안과학자들로부터 눈의 지문이라 불리며 1961년과 1965년에 학계에 보고되었다[3],[4]. 이렇게 각 사람들이 다르게 가지는 홍채의 특징을 추출하고, 그 정보를 이용하여 개개인을 인식하는 것을 홍채인식이라 말한다.

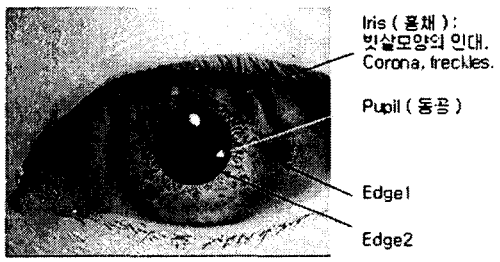

그림 1. 홍채 영상

홍채인식 시스템은 그림 2와 같은 인식처리절차를 가진다. 홍채인식 시스템 사용자에 대한 320 x 240의 영상을 획득하여 전처리 과정에서 홍채인식에 불필요한 영상을 분리 제거하고, 정규화 및 극좌표계로의 변환을 수행한다. 전처리를 마친 후, 홍채영상으로 웨이블렛 변환을 이용하여 홍채특징을 추출하고, 특징값은 미리 등록되어진 특징값들과 비교하여 본인여부를 판단하여, 신원을 확인한다.

2. 홍채특징 추출

카메라를 통하여 사람의 홍채 영상을 획득할 때, 카메라를 향한 눈의 자세가 일정하지 않은 경우 극좌표 변환과 정규화를 거친 홍채 영상은 수평 방향으로의 위치 이동이 발생하게 된다. 반면에 정교하게 정규화 된다면 수직 방향으로의 위치 이동은 없다. 이러한 위치 이동이 있는 경우에는 웨이블렛 변환된 각각의 서브 밴드에서 그 값이 일정하지 않다는 단점이

있다.

홍채패턴만을 추출하여 464×64의 크기로 정규화된 영상은 웨이블렛 변환을 이용하여 특징값을 추출하게 된다. 웨이블렛(Wavelet) 변환이라 함은 위와 같은 변환 과정에서 기저함수로

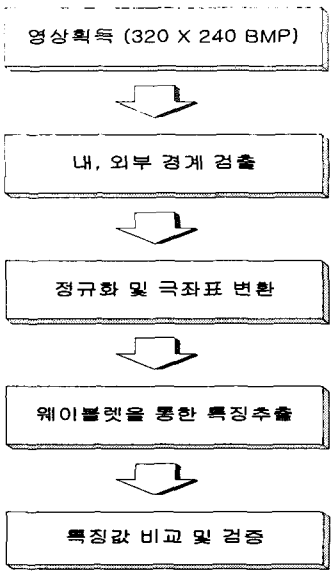

그림 2. 홍채인식 처리절차도

웨이블렛 함수를 사용하는 것이다. 그 기저함수의 종류에 따라 웨이블렛 변환의 종류가 정해지게 되는데 여기서 사용한 것은 Haar 웨이블렛이다. 이 과정을 통해서 영상 정보의 크기는 1/4로 줄인 상태에서 원 영상의 정보를 대부분 유지되는 LL 부분을 얻게 되는 것이다. LL 부분과 나머지 세 부분의 값을 갖고 원 영상을 복원할 수 있다.

3. 홍채 특징벡터 비교 검증

이진 벡터로 생성된 두 특징벡터의 비교 검증을 위해서는 다음 식과 같은 해밍 거리(HD)를 이용하는 방법이 있다.

$$HD = \frac{1}{N}\sum_{i=1}^{N} A_i (XOR) B_i$$

즉, 각 차원별로 할당된 비트값을 비교하여 일치하면 0, 다르면 1을 반환하여 이를 총 차원수로 나누어 최종결과를 나타내게 된다. 이는 이진 코드로 형성된 특징벡터의 일치 정도를 판별하는데 있어서 간단하고도 유용한 방법이다. 해밍 거리를 이용하게 되면 동일한 데이터의 경우 모든 비트의 비교 결과가 0이 되게 되므로 0에 가까운 결과일수록 본인의 데이터임을 알 수 있다. 타인의 데이터 경우 그 일치정도가 확률상 0.5를 나타내게 되므로 타인의 데이터와의 비교시 0.5근처에 값이 집중해 있음을 알 수 있다. 이에 따라 0과 0.5 사이에 적절한 경계치를 설정하면 본인과 타인의 데이터를 구분짓는 경계가 되는 것이다. 특징벡터의 데이터 일치율도 전체 데이터 차원에 대해서 일치하는 비트의 수를 비율로 나타낸 것으로 동일 데이터에 대해서는 100, 타인의 데이터에 대해서는 50에 가까이 분포하여 검증 결과를 보여주게 되는 것으로 해밍 거리와 같은 원리이다.

III. 홍채인식용 임베디드 시스템

리눅스가 임베디드 시스템의 운영체제로서 최적이라는 평가를 받는 이유는 다음과 같이 간단히 제시할 수 있다. 리눅스는 크기가 작고, 맞춤화가 쉽게 가능하고, 인터넷과의 최적화된 호환성을 제공할 뿐만 아니라, 공개 소프트웨어로서 무료라는 점이다[6],[7]. 이에 따라 리눅스를 이용한 모듈화 설계방식이 임베디드 시스템과 잘 맞아 떨어진다는 평가를 받으면서 임베디드 리눅스 어플리케이션 개발 영역은 급격히 확대되고 있는 추세라 할 수 있다.

1. 시스템 구조

현재 개발하고 있는 홍채인식 시스템을 위한 UI보드 부분은 UI보드 자체의 하드웨어적인 부분과 UI보드 구동을 위한 소프트웨어적인 부분으로 구성되어 있다. 먼저 하드웨어적인 부분을 보면 그림 3에서 볼 수 있듯이 메인 CPU로는 StrongARM CPU를 사용하고 그 외에 통신을 담당하는 부분과 메모리, 플래시 메모리, 사운드를 담당하는 부분, GPIO (General Purpose I/O)를 이용하여 외부의 장치를 제어하는 부분으로 구성되어 있다[8].

다음으로 소프트웨어적인 부분을 보면 부트로더, 커널, 제어 프로그램, 응용 프로그램으로 구성되어 있다. 제한된 메모리 용량, 프로세서의 성능, 그리고 특정 작업의 수행으로 인해, 임베디드 리눅스는 일반 리눅스와는 다른 여러 특징을 갖는다. 본 연구의 대상이 되는 임베디드 시스템에서 사용되어지는 램(RAM)과 저장장치(Flash memory, ROM)는 일반 리눅스가 타겟으로 삼는 PC 보다 훨씬 적은 용량이므로 OS의 전체 크기, 부팅시의 메모리 사용흔적(foot print), 그리고 런-타임때의 메모리 사용을 줄여야 한다. 이러한 메모리 사용과

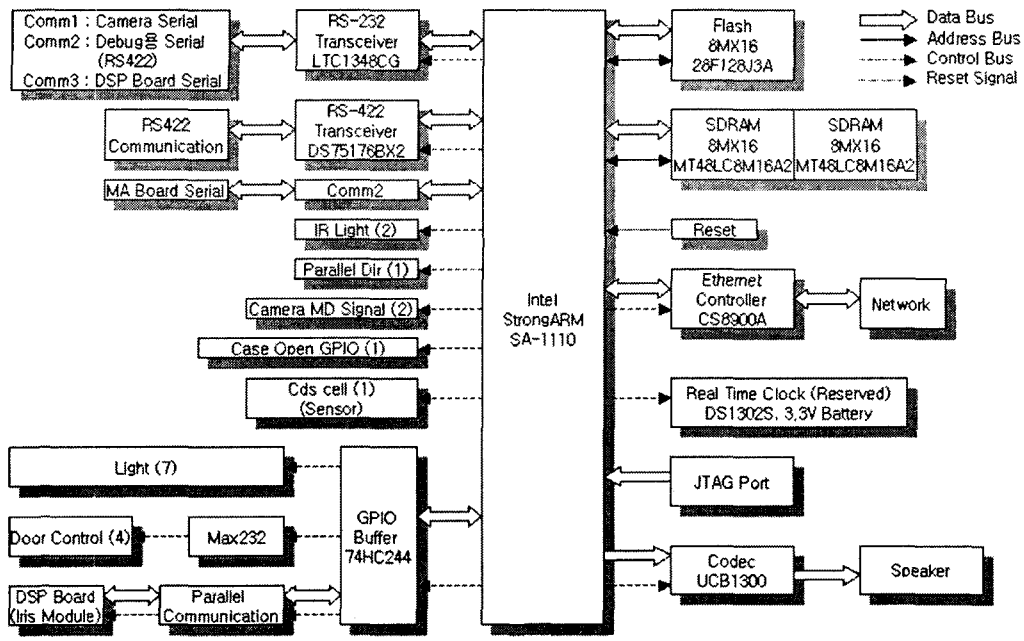

그림 3. 홍채인식 시스템의 UI 하드웨어 구조

전체 코드 크기의 축소, 그리고 DSP 보드와의 통신 기능 구현을 위해 필요하지 않은 기능은 커널에서 삭제하여 커널을 효율화 하였다. 또한 전체적인 구조상 그림 3에서 제시된 하드웨어 보드 구조에 맞게 커널, 부트로더, 제어프로그램을 수정하고 재설정하였다. 즉, 임베디드 시스템에서 리눅스 운영체제를 이용하여 시스템을 구동하려면 위의 소프트웨어적인 부분을 하드웨어 구성요소에 맞게 수정해야만 한다.

IV. 구현된 UI 시스템 구성요소

1. 부트로더 (Boot Loader)

부트로더의 역할은 먼저 임베디드 시스템의 플래시 메모리나 HDD와 같은 외부 저장장치에 저장되어있는 커널을 메모리(여기서는 SDRAM)에 탑재하는 것이다. 임베디드 시스템을 포함한 대부분의 시스템(PC 포함)은 외부저장장치에 있는 운영체제를 사용할 수 없으므로, 운영체제를 CPU가 접근할 수 있는 메모리로 옮겨야 하는데 그 부분을 부트로더가 담당한다. 그리고, 부가된 기능으로서 외부 네트워크를 이용하여 여러 프로그램들을 시스템의 플래시 메모리나 주 메모리에 적재할 수 있게 한다.

부트로더에서 시스템의 플래시 메모리에 있는 커널을 주메모리에 로드하기 위해서는 플래시 메모리에서 커널을 읽어와서 주메모리에 다시 그 읽어들인 커널을 써야한다. 여기서 부트로더는 플래시 메모리의 특성과 주메모리의 특성 에 맞춰서 수정되어 있어야 하는데, 만약 그 특성이 틀리면 잘못된 값을 읽어와서 잘못된 메모리 주소에 쓰게 될 수도 있으므로, 여기서는 시스템의 코어클럭(core clock) 설정을 변경시켰다. 부트로더가 구동될 때 먼저 하는 일은 CPU 세팅, 플래시 메모리, 메인 메모리 설정, 그리고 그 외의 다른주변 장치들을 설정하는 일이다. StrongARM에는 PPCR(Power Manager PLL Configuration Register)이 있는데 이 부분에서 191.7MHz로 맞추었다. 이 부분은 CPU에서 제공하는 클럭을 설정하는 부분이고, 여기서 설정된 값은 시스템 전반에 영향을 미치게 된다. 메인 메모리에 클럭이 들어갈 때 여기에서 값을 참고하여 들어가게 되는 것이다. 이 값이 맞지 않으면 메인 메모리에 정확한 데이터가 들어가지 않는다.

메모리에 대한 세팅이 끝나면, 다음에는 메시지 입출력에 관한 부분을 수정해야 한다. 타이눅스에서 제공하는 부트로더는 시리얼포트 2, 3번으로만 메시지를 입출력하게 되어 있다. 하지만, UI보드의 경우에는 StrongARM에서 제공하는 3개의 시리얼포트를 모두 사용해야 하므로 다 사용할 수 있도록 수정하였다. 부트로더의 "serial.c" 파일을 다음과 같이 수정하여 3개의 시리얼라인이 작동되도록(enable) 하였다.

이에 따라, 수정 후 부트로더를 컴파일하고 JTAG를 이용하여 시스템에 올리면, 제대로 동작하는 것을 확인 할 수 있다.

표 1. 부트로더의 "serial.c" 파일 수정부분

```
...
#elif defined USE_SERIAL2
  CR3 = Ser3UTCR2;
  Ser2UTCR3 = (CR3 & (~UTCR3_TXE));
  while(Ser2UTSR1 & UTSR1_TBY);

  Ser2UTCR3 = 0x00;
  Ser2UTSR0 = 0xff;
  Ser2UTCR0 = ( UTCR0_1StpBit | UTCR0_8BitData );
  Ser2UTCR1 = 0;
  Ser2UTCR2 = (u32)baudrate;
  Ser2UTCR3 = ( UTCR3_RXE | UTCR3_TXE );
...
void SerialOutputByte(const char c)
#elif defined USE_SERIAL2
  /* wait for room in the tx FIFO */
  while((Ser2UTSR1 & UTSR1_TNF) == 0);
  Ser2UTDR = c;
...
void SerialOutputRawByte(const char c)
#elif defined USE_SERIAL2
  /* wait for room in the tx FIFO */
  while((Ser2UTSR1 & UTSR1_TNF) == 0);
  Ser2UTDR = c;
...
int SerialInputByte(char *c)
#elif defined USE_SERIAL2
if(Ser2UTSR1 & UTSR1_RNE)
int err = Ser2UTSR1 & (UTSR1_PRE | UTSR1_FRE |
UTSR1_ROR);
*c = (char)Ser2UTDR;
...
```

2. 커널(Kernel)

본 개발의 레퍼런스 보드인 타이눅스 2.4.6은 x86용 커널 2.4.6을 ARM용으로 수정한 것이다. 커널은 보통 x86 PC용으로 제작된 것을 ARM-Linux라는 조직에서 ARM core에 맞추어서 수정한 것이 ARM-Linux이고, 그것을 레퍼런스 보드인 타이눅스에 맞게 수정한 것이 타이눅스 2.4.6이다. 이 커널은 LAN이나 사운드, 시리얼, GPIO등의 여러 장치들을 레퍼런스 보드에 맞추어서 드라이버들을 수정했다. 본 개발시에는 이 커널을 UI보드에 맞추어서 다음과 같은 부분을 수정·구현 하였다.

- 현재 커널에서 하드웨어의 상태를 GPIO를 통하여 LED에 표시하게 되어있는 부분을 삭제하여 UI보드가 모든 GPIO를 자유로 사용할 수 있도록 하였다.
- 응용프로그램이 GPIO를 자유로 제어할 수 있도록 제어 드라이버를 제작하였다.
- 본 임베디드 시스템 특성상, 플래시 파일시스템을 지원할 수 있도록 저널링 플래시 파일시스템2(JFFS2)를 사용하였다.
- 현재 타이눅스에서는 시리얼 포트를 1, 3번밖에 쓰지 못하도록 되어있다. 보통 시리얼포트 2번은 IR포트(적외선포트)로 쓰게 되어있는데, 이 부분을 풀어서 UI보드가 시리얼포트 1, 2, 3번 모두를 자유로이 사용할 수 있도록 하였다.
- UI보드에 맞춰서 커널 전반적인 세팅을 해준다.

2.1. GPIO 제어기

타이눅스 커널에서는 하드웨어의 상태를 GPIO를 통하여 연결되어있는 LED에 표시하도록 되어있다. 이 부분이 GPIO 0번에서 3번까지를 사용하므로 이 부분을 사용하지 못하도록 해당부분을 주석 처리한 후, UI 보드가 GPIO를 자유로 세팅할 수 있도록 관련 제어 드라이버를 제작했다.

프로그램을 작성했으면 커널이 그 드라이버를 인식하게 하기 위하여 세팅해야 한다. 수정할 파일은 /drivers/char 디렉토리에 있는 Makefile과 Config.in, 그리고 /init 디렉토리에 있는 main.c이다. 여기서 수정한 사항들은 커널 세팅때 GPIO 설정부분에서 체크해줌으로 커널 컴파일시에 제어드라이버를 포함시킬 수 있다. main.c 파일은 커널에서 제공하는 주번호, 부번호를 등록시키는 부분이다. 커널은 각 장치들마다 주번호와 부번호를 두어서 구별하고 관리하고 있다. 주번호는 각 장치들의 메인번호가 된다. HDD나 시리얼포트, LAN카드 등의 장치들을 구별할 때는 주번호를 사용해서 구별한다. 부번호는 각 장치들에 해당하는 번호로, 예를 들어서 HDD가 3개 있을 경우에는 주번호에 각각 3개의 부번호를 할당해서 HDD를 구별하게 된다. GPIO 제어드라이버도 역시나 제어드라이버에 속하기 때문에 커널이 분류할 수 있도록 주번호와 부번호를 부여해야 한다.

2.2. JFFS2 (저널링 플래시 파일시스템2)

임베디드 시스템 특성상 정보가전제품은 물론 생체인식 제

품들도 간편하게 조작하고 쉽게 데이터를 보관할 수 있는 저장매체를 요구하고 있다. 따라서, 본 개발에서도 전원이 꺼진 상태에서도 데이터 저장이 가능한 플래시 메모리를 사용하였다[9],[10]. 타이눅스 2.4.6에서 제공하는 초기 파일시스템은 EXT2이다. EXT2 파일시스템은 플래시 메모리에 쓰기가 불가능한 파일시스템이므로 우리는 EXT2 파일시스템 외에 플래시에 읽고 쓰기가 가능한 파일시스템을 하나 더 설정해야 한다. 현재 UI 보드의 플래시 형태는 다음과 같다[4].

표 2. 플래시 메모리 구성

항목	범위
Boot Loader	0x00000000 ~ 0x00001FFF
Boot Script	0x00002000 ~ 0x00003FFF
Kernel Image	0x00100000 ~ 0x001FFFFF
Root Image	0x00200000 ~ 0x003FFFFF
User Partition	0x00400000 ~ 0x005FFFFF
Data Partition	0x00600000 ~ 0x00FFFFFF

표 2의 내용에 따르면 JFFS2를 적용할 구간은 Data Partition이다. 그래서 커널에 위의 구간을 적용할 수 있도록 인식시켜줘야 한다. 커널의 /drivers/mtd/maps 디렉토리에 있는 sa1100-flash.c 파일을 수정한다.

2.3. 시리얼 포트

현재 타이눅스 2.4.6에서는 시리얼포트 3번을 통하여 커널 메시지를 입출력하고 포트 1번은 여유분으로 자유로 사용할 수 있도록 설정하고 있다. 여기서 우리는 포트 2번 또한 사용할 수 있도록 하기위해 커널의 /arch /arm /mach - sa1100 디렉토리에 있는 cerf.c 파일을 아래 표 3과 같이 수정하였다.

표 3. 커널의 "cerf.c" 파일 수정부분

```
...
static void __init cerf_map_io(void)
sa1100_map_io();
iotable_init(cerf_io_desc);
sa1100_register_uart(0, 3);
Ser1SDCR0 |= SDCR0_SUS;

/* 변경된 부분 -- */
sa1100_register_uart(1, 1);
Ser2UTCR4 &= ~UTCR4_HSE;
Ser2HSCR0 &= ~HSCR0_ITR;
sa1100_register_uart(2, 2);
...
```

2.4. 커널 세팅과 커널 컴파일

커널 컴파일전에 하는 커널 세팅은 다음과 같은 3가지 방법이 있다.

- make config : 각 커널 세팅 항목들 각각을 Yes/No의 대답으로 설정하는 방법이다. 커널 세팅 항목이 100여개가 넘으며, 지금은 거의 사용하지 않는다.
- make menuconfig : 커널 세팅 항목들이 주제별로 메뉴식으로 구성되어있어서 사용자가 각 메뉴를 선택하며 해당 항목의 커널 항목을 세팅할 수 있다. 현재 가장 많이 사용하는 세팅 방법이다.
- make xconfig : 방식은 menuconfig와 비슷하나, 리눅스의 Xwindow 시스템에서 동작하는 방법이다. 그래픽으로 표현되어서 보기에는 편한 반면, 시스템 자원을 많이 차지하는 단점이 있다.

따라서, 본 논문에서는 두 번째 make menuconfig 방식을 써서 UI보드의 시스템 구성에 맞게 다음과 같은 방법으로 커널을 세팅하였다.

O General setup 선택
① Default kernel command string 선택
② console=ttySA2, 115200 입력 (여기서 커널 메시지를 시리얼포트 2번으로 입출력하게 한다.)
③ Timer and CPU usage LEDs 선택 제거

O Networking options 선택
① IP : kernel level autoconfiguration 체크 (IP를 자동으로 설정기능)
② IP : DHCP support 체크 (만약 Bootp 프로토콜을 사용한다면 이 부분을 체크)

O Character devices 선택
① SA1100 serial port support 체크
② Console on SA1100 serial port 체크
③ Default SA1100 serial baud rate 선택
④ UI보드의 시리얼포트 속도를 115,200으로 조정
⑤ UCB1200 support 체크
⑥ UCB1200 audio support 체크, 나머지는 체크제거
⑦ SA1100 Real Time Clock 체크

O File systems 선택
① Journalling Flash File System version 2 설정
② JFFS2 debuffing verbosity 선택

3. 제어프로그램 및 사용자(usr) 이미지

커널이 구동될 때 참조하는 제어프로그램은 타이눅스에서 함께 제공하는 Root 파일시스템과 함께 제공된다. 초기 제어 프로그램의 설정상태는 시리얼포트 3번에서 커널메시지 입출력, 외부파일시스템 인식 불가로 되어있다. 이러한 부분을 UI 보드에 맞춰서 수정하였다. UI보드에서 사용하는 제어프로그램 설정은 시리얼포트 2번에서 커널메시지 입출력과 커널에서 설정한 Data partition에 JFFS2(Journaling Flash File System ver. 2)를 적용하기 위해서는 Root 파일시스템 이미지를 수정해야 한다. 타이눅스에서 제공하는 Root 파일시스템 이미지는 다양한 이름으로 되어있지만 여기서는 root.gz라고 하도록 한다. Root 파일시스템 이미지는 GZ형식의 압축파일로 되어있다. 압축은 커널이 부팅될 때 풀어진다. 즉, 시스템에는 압축된 상태로 올라가게 된다.

먼저 mkdir rootdir이라고 해서 Root 파일시스템 이미지를 받아낼 수 있는 공간을 확보한 다음에 gzip - d root.gz로 Root 파일시스템 이미지의 압축을 해제하면 root라는 파일이 생성된다. 이 파일을 우리가 만든 rootdir이라는 디렉토리에 연결해야 한다.

% mount root rootdir - o loop

다음으로 JFFS2를 인식하도록 Root 파일시스템의 /etc/rc.d 디렉토리에 rc.sysinit 파일을 수정하면, 커널은 매번 부팅할 때마다 usrdir 디렉토리를 만들고 거기에 우리가 설정한 Data partition을 JFFS2로 인식시키게 된다. 마지막으로는 보드에서 터미널을 통하여 프로그램을 실행시킬 수 있도록 터미널 설정을 해준다. 역시 Root 파일시스템의 /etc 디렉토리에 inittab 파일을 수정하면, 커널이 모두 올라가고 시리얼통신 포트 2번을 통하여 프롬프트가 뜨면서 보드와 통신할 수 있게 된다.

V. 실험 및 결과분석

홍채인식 시스템을 구현하기 위해서는 홍채이미지 캡처, 카메라 제어, 영상처리부, 그리고 홍채인식과 등록 등의 제어를 담당하는 모듈들이 필요하다. 이러한 모듈을 구현하기 위해 영상처리 DSP보드 모듈과 UI, 등록 PC와의 통신, 주변장치의 제어를 담당하는 StrongARM이 탑재된 메인 보드로 구성하였다.

홍채 영상처리 DSP보드 구성을 위한 DSP칩은 TI사의 C6711 DSP칩을 채택하고 CPLD를 이용해 제어 모듈을 FPGA에 코딩했으며 카메라는 22배줌 자동초점 CCD카메라를 사용하였다. 또한 메인 보드는 홍채 영상 보드와의 통신을 비롯 사용자 인터페이스를 위해 사운드, LCD 출력, 등록기와 통신 그리고 사용자 입력을 위한 키패드와의 통신을 담당하도록 구현하였다. 이를 위해 StrongARM이 탑재된 보드를 제작해서 기존의 리눅스 커널에 GPIO를 이용하여 병렬(Parallel) 통신을 지원하도록 하고, 시리얼 레지스터를 이용해 통신과 홍채 영상처리 DSP보드의 통신을 통한 간접제어가 가능하도록 본 논문에서 설명한 바와 같이 디바이스 드라이버를 수정 · 보완하였다.

실험을 위하여 UI보드를 홍채인식용 출입관리서버와 TCP/IP로 연결하였고, 네트워크를 통한 통신 기능을 확인하기 위하여 NFS(Network File System)를 이용한 네트워크 테스트를 실시하였다(그림 4 참조). UI 보드는 PC대비 상대적으로 적은 용량을 갖고 있으므로 개발에 많은 제약이 따른다. 그래서 보통 프로그램 개발시에는 리눅스가 설치된 다른 PC에서 개발하고 NFS로 UI보드와 연결하여 UI보드의 임베디드 리눅스 환경에서 실행시킨다. NFS로 연결된 부분은 PC와 UI 보드의 내용이 동일하다. 즉, UI보드의 임베디드 리눅스 시스템이 NFS로 연결된 PC의 리눅스 시스템으로 디렉토리를 공유하는 것이다.

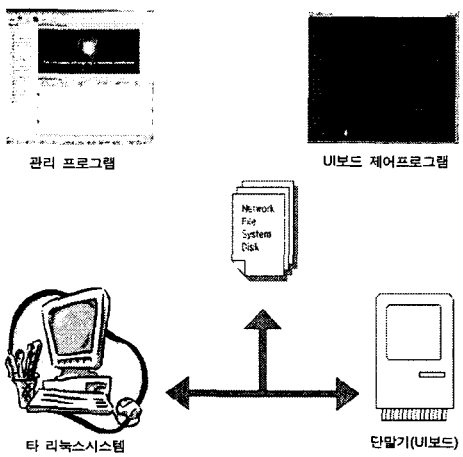

그림 4. UI보드와 타 리눅스 시스템과의 데이터 통신

이에 따라, 본 실험 결과 TCP/IP 방식으로 1:1 연결된 PC와 UI 보드상의 두 디렉토리의 내용은 서로 동일하며 PC나 UI보드에 추가, 삭제, 수정이 일어나면 PC, UI보드 양쪽에서 동시에 서로 데이터를 주고 받음을 확인할 수 있었다.

VI. 결론

본 논문에서는 홍채인식 상용제품에 적용할 임베디드 리눅스 기반 UI 보드를 설계 및 포팅하고, 구현된 보드가 이기종 시스템과 성공적으로 통신하는 것을 실험으로 증명하였다. 임베디드 리눅스가 임베디드 시스템의 운영체제로서 더욱 더 활성화되기 위해서는 리눅스 커널, 장치 구동기, 윈도우에 기반한 개발환경의 일반화는 물론, 이를 기반으로 하는 실시간 운영체제 기능이 보완•개발 되어야할 것으로 예상된다. 더욱이 임베디드 시스템의 응용분야가 이동통신, 인터넷이나 소형 가전 및 산업분야로 널리 확장되면서 임베디드 시스템 설계시 성능, 비용 및 QoS(서비스 품질)를 충족시켜주기 위해서는 본 논문에서 제시된 기술들 외에도 플래시 파일관리 기술이나 SOC(System On Chip) 설계방법을 이용한 전력량 감축기술 등이 상용 제품에서는 시급히 요구되고 있다고 할 수 있다.

참 고 문 헌

[1] 한영인, "생체인식을 위한 적응적 방향 결정 방법", 2002. 11, 한국정보보호학회, 2002.

[2] 이근, "생체인식 정보관리 및 보안표준(X9.84)", 2002.11, 한국정보보호학회, 2002.

[3] 이상원, 전명근, "홍채인식에 의한 개인 확인 및 인증 알고리즘", 제 13회 한국자동 제어학술 회의, pp. 1133-1136, 1998년 10월.

[4] 조성원, "위치이동에 무관한 웨이블렛 변환을 이용한 홍채인식방법", 국내특허출원번호 10-2002-0085266, 2002. 12.

[5] 김재훈, "임베디드 리눅스 기술 및 시장동향 조사", 정보통신학술연구과제, 2001년 2월.

[6] 김명규 외 2인, "리눅스를 이용한 임베디드 시스템의 기술동향", ETRI 주간기술 동향, 930호, 2000년 2월.

[7] Embedded Linux Basics, 와우리눅스㈜ 교육자료, 2002.

[8] 타이눅스박스 사용자매뉴얼 Ver 1.1, 팜팜테크(주), 2001.

[9] 김정기, 박승민, 김채규, "임베디드 플래시 파일 시스템", 정보처리학회지, 제 9권 1호, pp. 43-49, 2002년 1월.

[10] M.L. Chang, P.C.H. Lee, "Managing Flash Memory in Personal Communication Devices," Proc. of IEEE Symp. on Consumer Electronics, pp. 177-182, 1997.

임 철 수(Cheol-Su Lim) 정회원

1985년 2월 : 서울대학교 계산통계학과 (이학사)
1988년 8월 : Indiana University 컴퓨터과학과(공학석사)
1995년 8월 : 서강대학교 컴퓨터공학과 (공학박사)

1994년 8월 ~ 1997년 2월 : (주)신세기통신 근무
1997년 3월 ~ 현재 : 서경대학교 컴퓨터 공학과 교수
<관심분야> : 차세대인터넷, 멀티미디어시스템

박 병 섭(Byoung-Seob Park) 종신회원

1989년 2월 : 충북대 컴퓨터공학과 (공학사)
1992년 2월 : 서강대학교 전자계산학과 (공학석사)
1997년 2월 : 서강대학교 전자계산학과 (공학박사)
1997년 4월 ~ 2000년 2월 : 국방과학연구소 선임연구원
2000년 3월 ~ 2002년 8월 : 우석대학교 컴퓨터교육과 교수
2002년 9월 ~ 현재 : 인하공업전문대학 컴퓨터정보공학부 교수
<관심분야> : 3G/4G 이동통신, 차세대인터넷, 이동인터넷, Mobile-IP