KHIX : 확장 및 재구성 가능한 임베디드 시스템 운영체제

백용규° 조진성
경희대학교 컴퓨터 공학과
ica28°@khu.ac.kr, chojs@khu.ac.kr

KHIX : A Scalable and Reconfigurable Embedded System Operating System

YongGyu Baek° JinSung Cho
Dept. of Computer Engineering Kyung Hee University

요 약

임베디드 시스템은 특정 목적을 수행하기 위해 설계된 시스템이며, 임베디드 운영체제는 실시간 운영체제, 범용 운영체제로 나뉜다. 실시간 운영체제는 각 운영체제에서 각각의 API를 제공하기 때문에 응용프로그램 작성 시 API를 새로 익혀야 되는 단점이 있다. 범용 운영체제는 사용자에게 익숙한 POSIX API를 제공 하지만 커널 이미지 크기가 커서 센서와 같은 메모리가 작은 운영체제에는 이식하지 못하는 단점이 있다. 본 논문에서는 이러한 단점을 보안하고 장점을 살리기 위해 컨포넌트화 하여 확장 및 재구성이 가능하도록 하고 POSIX 기반의 API를 제공하여 응용 프로그램 작성에 용의하도록 하는 KHIX 임베디드 시스템 운영체제를 설계 및 구현하고 고성능의 PXA255, 저성능의 ATmega128에 이식한 내용을 다룬다.

1. 서 론

임베디드 시스템은 우리가 쉽게 접할 수 있는 PDA, 휴대폰, TV, MP3 플래이어, 셋탑 박스, 디지털 카메라, 의료 기기 등 일상 생활과 매우 밀접한 관계를 갖고 있다. 이러한 임베디드 시스템이 점차 복잡해지고 응용분야가 넓어져 간단한 제어용 프로그램만으로 충분하지 않게 되었으며 점차 운영체제가 필요한 임베디드 시스템이 증가하였다.

임베디드 시스템의 큰 특징은 실시간 시스템이다. 실시간 시스템은 정해진 시간 내에 결과를 처리해야 하는 시스템이다. 이는 주어진 작업을 빨리 처리한다는 의미가 아니라 정해진 시간을 넘겨서는 안된다는 의미이다.

실시간 시스템은 크게 경성 실시간, 연성 실시간으로 나뉘어 진다. 경성 실시간은 정해진 시간 내에 작업의 결과가 절대적으로 처리되어야 하는 시스템으로 비행기의 비행제어 시스템, 핵발전소의 제어 시스템을 예를 들 수 있다. 연성 실시간은 정해진 시간 내에 작업의 결과가 않더라도 치명적인 결과를 초래하지 않고 시스템 오류가 되지 않는 시스템으로 라우터와 핸드폰을 예를 들 수 있다. 이러한 특징 때문에 임베디드 시스템 운영체제는 주로 실시간 운영체제를 사용한다.

실시간 운영체제는 컴퓨터의 자원을 효율적으로 관리하고 다양한 기능을 지원하는 범용 운영체제와는 다르게 마이크로프로세서가 내장된 제한된 하드웨어를 가지고 주변 상황을 고려하여 요구되는 기능을 효율적으로 수행하는 임베디드 시스템을 위해 작은 크기로 최적화된 운영체제이다.

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터지원 사업의 연구결과로 수행되었음 (IITA-2006-(C1090-0602-0002))

최근에 임베디드 시스템 운영체제 기술 변화는 각종 임베디드 시스템에 사용되는 하드웨어의 발전으로 인해 사용 할 수 있는 하드웨어 자원이 늘어나 VxWorks, pSOS, Tiny OS와 같은 기존 실시간 운영체제뿐만 아니라 임베디드 리눅스, WIN. CE와 같은 범용 운영체제를 많이 사용한다. <표1>은 실시간 운영체제와 범용 운영체제의 일반적인 차이점을 보여준다.

속성	실시간 운영체제	범용 운영체제
목적	단일	범용
스케줄링	우선 순위 기반	공평
자원할당	정적	동적
가상메모리	사용 않음	사용
파일 시스템	제한된 사용	사용
POSIX API	제한된 지원	지원
커널 이미지	작다	크다

<표1> 실시간 운영체제와 범용 운영체제 비교

실시간 운영체제는 제한된 하드웨어에서 수행되는 운영체제로 커널 이미지 크기가 작다. 따라서 센서와 같이 메모리 크기가 작은 시스템에도 커널 이식할 수 있다. 그러나 실시간 운영체제는 각 각의 API를 제공하기 때문에 응용 프로그램 작성 시 해당 API를 새로 익혀야 되는 단점이 있다. 반면 범용 운영체제는 POSIX 기반의 API를 제공하기 때문에 응용 프로그램 작성 시 기존의 리눅스 사용자라면 쉽게 응용 프로그램 작성이 가능하다. 또한 하드웨어의 발전으로 인해 하드웨어 자원이 늘어남에 따라 보다 다양한 시스템에 범용 운영체제의 커널 이식

이 가능하지만 커널 이미지가 커서 센서와 같이 메모리 크기가 작은 시스템에는 커널 이식이 불가능하다는 단점이 있다.

따라서 본 논문에서는 실시간 운영체제와 범용 운영체제의 단점을 보안하고 장점을 살리는 KHIX 임베디드 운영체제를 각 기능들을 컨포넌트화 시켜 쉽게 확장 및 재구성이 가능하도록 하고, POSIX 기반의 API를 제공해 응용 프로그램 작성에 용의하도록 설계 및 구현하여 고성능의 PXA255, 저성능의 ATmega128에 이식하였다.

본 논문은 2장 현재 국내 및 해외에서의 운영체제의 개발 상황, 3장 KHIX 임베디드 운영체제의 설계, 4장 KHIX 운영체제의 구현 및 PXA255, ATmega128에 이식, 5장 결론 및 향후 연구 과제로 구성되어진다.

2. 관련 연구

임베디드 시스템은 실시간성 특성 때문에 다중처리가 가능한 상용 실시간 운영체제가 주로 사용되었다. VxWorks[1], pSOS[2], Nucleus[3] 등 이러한 상용 운영체제는 실시간 처리 기능을 제공하는데 목적을 두고 신뢰성에 중점을 두며, 범용성이 아닌 주로 한가지 특수한 목적에 최적화된 운영체제이다. 이러한 상용 운영체제는 단순히 바이너리 코드만을 제공한다. 따라서 기능의 수정이 어렵다는 단점을 가지고 있다.

이로 인해서 국내, 해외 대학 및 연구기관에서는 핵심 기술 확보를 위해 일반적인 실시간 운영체제와 센서와 같은 저성능의 시스템에서 동작하는 센서용 운영체제가 개발 되어왔다.

일반적인 실시간 운영체제인 µC/OS-II[4]는 교육용 실시간 운영체제로 간단한 소스코드로 실시간 운영체제의 기본적인 기능들을 제공하고 있다. 또한 국내 대학에서 개발한 Velos[5]는 실시간 스케줄링 지원 뿐만 아니라 TCP/IP, 동적 프로그램 로딩지원, Microwindows 시스템 및 KVM을 지원하고 있다. 센서용 운영체제로는 전통적으로 TinyOS[6]와 최근 국내에서 개발된 나노Qplus[7]가 대표적인 센서용 운영체제이다. TinyOS는 UC 버클리에서 제작된 이벤트 발생 중심의 상태 변화 방식을 채택한 센서 네트워크용 운영체제로 C와 유사한 NesC를 통해 센터 네트워크용 응용 프로그램을 작성 할 수 있다. 나노 Qplus 운영체제는 한국전자통신연구원(ETRI)에서 개발된 센서 네트워크용 운영체제이다. 특징은 제한된 메모리 사용을 최소화하기 위한 멀티 스레드 간의 스택 공유, 저전력 파워소비 지원 등이 있다.

이러한 임베디드 운영체제는 독자적인 API를 가지고 있어서 사용자가 응용 프로그램을 작성할 경우 해당 API를 새로 익혀야 되는 단점이 있다. 따라서 최근의 운영체제 개발시에는 POSIX 기반의 API를 지원하는 추세로 가고 있다. [10][11]

또한 최근의 임베디드 시스템에 사용되는 하드웨어의 발전으로 인해 사용할 수 있는 하드웨어 자원량이 늘어나 인터넷 연결, 멀티미디어 처리 등의 기능이 요구됨에 따라 운영체제도 점차 범용 운영체제가 많이 사용되고 있다.

이러한 범용 운영체제로는 임베디드 리눅스[8]와 WinCE[9]가 있다. 임베디드 리눅스는 오픈소스모델로 인해 수많은 개발자들을 통해 개발이 되어졌다. 임베디드 시스템에서 실시간성 지원을 위해 기존의 커널의 수정을 통한 실시간 지원, sub커널을 통한 실시간 지원하고 있다. 또한 POSIX 기반의 API를 지원하고 GNU도구 사용으로 인해 응용 프로그램 작성에 용의하며, 다양양종류의 네트워킹, 파일 시스템, 프로토콜 지원하며 다양한 종류의 하드웨어를 지원하고 있다. 또한 여러 개발자들에 의해 개발이 되었기 때문에 기술성, 견고성, 보안기술이 뛰어나다. [12]

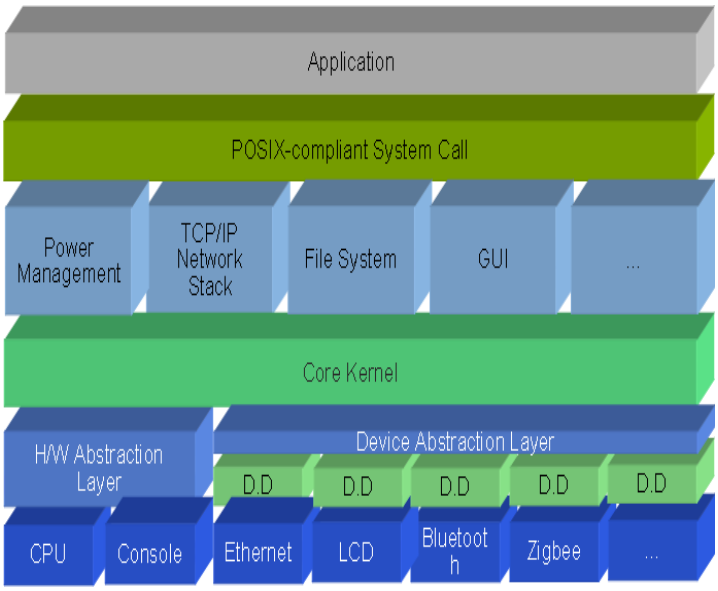
MS에서는 WinCE 3.0를 발표하면서 기존의 WinCE 2.0에서 실시간성 지원이 떨어진다는 점을 보안하였다. WinCE는 기존의 PC환경에서의 사용자들에게 이미 익숙해진 윈도우 인터페이스를 임베디드 환경에서도 동일하게 사용할 수 있도록 제공하고 있다.

이러한 범용 운영체제는 임베디드 시스템의 응용 범위가 넓어지게 되면서 많은 기능을 충족 시켜주면서 개발에 유리하기 때문에 최근에는 많은 분야에서 사용되고 있다.[13] 하지만 많은 기능을 제공하다 보니 커널의 이미지 크기가 커서 센서와 같이 메모리가 적은 시스템에서는 커널 이미지를 이식할 수 없다는 단점이 있다.

따라서 실시간 운영체제와 범용 운영체제의 단점을 보안하고 장점을 살리는 확장 및 재구성 가능한 KHIX 임베디드 시스템 운영체제를 설계, 구현 하게 되었다.

3. 연구 내용

KHIX는 임베디드 시스템을 위한 실시간 운영체제이다. 임베디드 시스템은 CPU 성능, 메모리, 소비 전력 등 제약사항이 많기 때문에 시스템 마다 목적에 맞는 구성을 필요로 한다. KHIX 임베디드 시스템 운영체제는 이러한 다양한 구성을 지원하고 최적의 성능을 보일 수 있도록 설계 되었다.



<그림1> KHIX 임베디드 시스템 운영체제 구성도

그림<1>은 KHIX 임베디드 시스템 운영체제의 구성도를 보여주고 있다. 각 기능들은 컨포넌트화 하여 확장에 용의하도록 하였고 HAL(Hardware Abstract Layer)를 통해 다른 프로세서에 이식하기 용의한 구조로 되어있다. 또한 POSIX 기반의 API를 제공해 응용 프로그램 작성에 용의하도록 하였다. 위의 그림에 따른 KHIX 임베디드 시스템 운영체제의 특징은 다음과 같다.

- 다중 쓰레딩 지원(Multi-threading)
- 실시간 스케줄링 지원
- POSIX 기반의 API 제공
- 확장성 지원
- 높은 이식성 지원
- 저전력 지원

3.1. HAL(Hardware Abstract Layer)

임베디드 시스템 하드웨어의 발전으로 임베디드 시스템에 사용되는 MCU가 다양해 졌다. 이로 인해서 새로운 MCU에 이식에 쉽게 되어야 되는 구조로 필요로 하였고 이를 위해 HAL을 정의하게 되었다. HAL은 모든 하드웨어 의존적인 부분을 모아서 이식성을 향상 시켰다.

HAL의 내용은 startup, context switching, UART, Timer, GPIO 등이다. 이로 인해 다른 프로세서에 이식할 경우 다른 기능들의 수정 없이 HAL의 수정만을 통해 다른 프로세서에 쉽게 이식이 가능하다. 뿐만 아니라 전통적인 시스템 개발 과정에서 하드웨어가 먼저 개발되고 이 후에 소프트웨어가 개발되는 구조에서 HAL을 통해 하드웨어와 소프트웨어를 병렬적으로 개발하는 것이 가능해졌다.

3.2. 다중 쓰레딩(Multi-Threading)

KHIX 임베디드 시스템 운영체제는 다중 쓰레딩 방식으로 동작한다. 각 쓰레드는 실시간성이 지원되는 쓰레드이며 POSIX 기반의 pthread API를 지원하게 된다.

3.3. 스케줄링

KHIX 임베디드 시스템 운영체제는 기본적으로 다중 쓰레딩 뿐만 아니라 실시간성이 요구되는 임베디드 시스템을 위하여 고정 우선순위 선점형 스케줄링을 지원한다. 또한 범용성을 위해 동일한 우선순위의 경우 FIFO와 Round-Robin 스케줄링을 지원하게 되며, Round-Robin 내의 시간 할당량을 조절 가능하다.

KHIX 임베디드 운영체제는 이러한 실시간성 보장 뿐만 아니라 기능의 수정 및 확장이 쉽게 가능한 구조로 되어 있어 다양한 임베디드 시스템의 요구를 반영할 수 있다.

3.4. POSIX API

임베디드 시스템에 사용되는 하드웨어의 발전으로 임베디드 시스템의 적용분야가 확산되면서, 새로운 시스템의 개발은 새로운 응용 프로그램의 개발을 필요로 하고 있다. 이전에는 임베디드 시스템에 사용되는 하드웨어의 성능이 시스템의 중요한 요소가 되었지만 최근에는 응용 프로그램의 개발이 임베디드 시스템의 성공 여부를 결정짓는 중요한 요소가 되고 있다. 따라서 응용 계층에 임베디드 운영체제의 호환성 지원을 위한 API 표준화 문제가 대두하게 되었다.

따라서 KHIX 임베디드 시스템 운영체제에서도 이러한 API 표준화를 위해서 POSIX 기반의 API를 지원하게 되었으며 다음과 같은 POSX 기반의 API를 지원하게 되었다.

3.4.1. 쓰레드(pthread)

KHIX 임베디드 시스템 운영체제는 다중 쓰레딩 방식으로 동작한다. 쓰레드 생성을 위한 pthread_create()와 쓰레드 종료를 위한 pthread_exit() 뿐만 아니라 POSIX에서 지정한 pthread API를 제공한다.

3.4.2. 디바이스 드라이버(Device Driver)

KHIX 임베디드 시스템 운영체제는 리눅스 및 유닉스 계열 운영체제와 유사한 디바이스 드라이버 등록 구조 및 인터페이스를 제공한다. 이로인해 응용 프로그램의 작성 시 기존의 리눅스에서 사용했던 코드를 재사용 할 경우에 크게 수정 할 필요 없이 바로 사용 가능하다.

디바이스 드라이버 동작	설명
open	장치 초기화
close	장치 해제
read	장치로부터 데이터 입력
write	장치로부터 데이터 출력
ioctl	입출력 제어

<표2> KHIX 임베디드 시스템 운영체제 드바이스 드라이버 인터페이스

위의 디바이스 드라이버 인터페이스를 통해서 LED, FND, LCD, Ethernet 등 장치에 대한 제어가 가능하다.

3.4.3. ITC(Inter-Thread-Communiation) 및 기타

KHIX 임베디드 시스템 운영체제에서의 ITC는 쓰레드와 쓰레드의 공동 작업을 위한 동기화 및 통신을 위하여 semaphore, Mutex, Message Queue를 지원하고 있다. 그 밖에 sleep, alarm, signal을 POSIX 기반 API를 지원하고 있다.

3.5 전력관리(Power Management)

임베디드 시스템에서 사용되는 기기들은 센서와 같이 배터리 중심으로 운영되는 경우가 많다. 따라서 전력 소

비를 줄이기 위한 방안이 많이 연구 되고 있다.
따라서 KHIX 임베디드 시스템 운영체제는 이러한 전력 소비를 줄이기 위한 기능을 위해 전력 관리 기능을 추가하였다. 전력 관리 기능은 크게 DVS(Dynamic Voltage Scaling)과 DPM(Dynamic Power Management) 2가지 기법이 존재 한다. DVS 기법을 위해 MCU의 클럭이 조절 가능하도록 하고, DPM 기법을 위해 MCU의 모드를 변경 가능하도록 한다. 이러한 기법을 가상 디바이스 드라이버 형태로 구현하여 응용 프로그램 작성 시 쉽게 사용 가능하도록 한다. 또한 이러한 DVS, DPM 기법을 스케줄링 레벨에서 특정 알고리즘을 적용하는 방안도 사용 가능하다.

3.6 네트워크, 파일 시스템, GUI

최근 임베디드 시스템에서는 인터넷 연결성, 멀티미디어가 강조 되고 있다. 기존의 단순 제어 기능에서 다양한 기능을 요구함에 따라 PC의 기능들이 임베디드 시스템에서도 사용되게 되었다. 이로인해 네트워크, 파일 시스템, GUI의 필요성이 생기게 되었다. 이는 임베디드 시스템 운영체제가 범용화 되었다는 것을 의미한다.
KHIX 임베디드 시스템 운영체제는 센서와 같은 저성능의 시스템 뿐만 아니라 이러한 멀티미디어 및 인터넷 연결이 가능한 고성능의 시스템을 고려해 네트워크, 파일 시스템, GUI에 대한 기능을 추가하여 구현 중에 있다.

3.7 미니 쉘(Mini Shell)

쉘은 사용자로부터 입력을 받을 수 있고 출력을 나타낸다. 현재는 KHIX의 각 기능들 구현 시 간단한 명령어를 통해 디버깅 용도로 사용되고 있지만 파일 시스템, 링커, 로더가 추가가 되면 기능이 추가 될 것으로 생각된다.

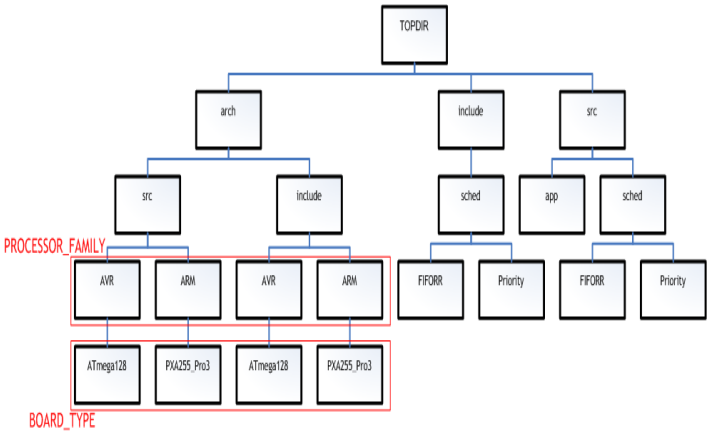
3.8 Configuration Tool

KHIX 임베디드 시스템 운영체제는 확장 및 재구성이 가능하며 필요한 기능들은 컨포넌트화 시켜 구현하면 쉽게 추가가 가능하다. 임베디드 시스템은 시스템을 구성할 경우 해당 시스템에 따라 그 구성이 달라진다. 따라서 필요로 하는 기능을 선택하여 커널 이미지를 생성하는 툴이 필요로 하게 된다. KHIX 임베디드 시스템 운영체제는 간단한 조작을 통해 필요로 하는 커널 이미지를 생성 할 수 있도록 지원하는 GUI로 작성된 툴을 지원해야 한다.

4. 구현

KHIX 임베디드 시스템 운영체제는 다양한 MCU에 이식이 용의하도록 HAL(Hardwayre Abstract Layer)를 제공하며 응용 프로그램 작성에 용의하도록 POSIX 기반의 API를 제공한다. 본 장에서는 실제 KHIX 임베디드 시스템 운영체제의 확장 및 재구성을 확인하기 위해서 구현된 내용을 살펴보고, 고성능의 PXA255[15], 저성능의 ATmega128[16]에 이식하는 내용을 다루며, 응용 프로그램의 작성 시 코드 POSIX 기반의 API 사용 및 코드 재사용성을 살펴본다.

4.1. KHIX 디렉토리 구조도



<그림2> KHIX 임베디드 시스템 운영체제 디렉토리 구조도

그림<2>는 KHIX 임베디드 시스템 운영체제의 디렉토리 구조도를 나타낸다. 빨간선으로 표시된 부분이 HAL Layer에 속한다. 다양한 MCU를 자원하기 위해 각 MCU별로 디렉토리를 분리해 구현하며 스케줄링은 현재는 FIFORR과 PriorityRR이 구현되어 있지만 다양한 종류의 스케줄링 방식이 구현 가능하다. 현재는 Configu 파일을 통해 다양한 기능들중 원하는 기능을 선택하여 커널 이미지 생성이 가능하다.
이와 같이 HAL을 통해 다양한 MUC를 지원하고 기능들의 컨포넌트화를 통해 추가 기능의 확장과 재구성이 쉽게 가능하다.

4.2. KHIX의 이식

실제로 KHIX의 확장 및 재구성성을 확인하기 위해 고성능의 PXA255, 저성능의 ATmega128를 선택하여 KHIX를 이식하였다. 여러 개의 쓰레드를 생성 후 실제 수행 여부를 확인 하였다. 다른 종류의 MCU로의 이식의 경우 단순히 HAL 부분을 수정하는 작업이 수행 되었다.

```
main..

|-------------------------------------------------------|
|                                                       |
|                  Welcome to KHIXOS                    |
|                                                       |
|-------------------------------------------------------|
pthread_create completed!!
(T1)Write = 0
(T1)Write = 1
```

<그림3> KHIX 임베디드 시스템 운영체제 수행 모습

<그림3>과 같이 KHIX 임베디드 시스템 운영체제가 고성능의 PXA255, 저성능의 ATmega128에서 정상 작동하는 것을 확인하였다. 따라서 단순히 HAL부분에 해당하는 내용만 수정하면 다른 시스템에 쉽게 이식되는 것을 확인 할 수 있었다.

4.3. 응용 프로그램 작성

KHIX 임베디드 시스템 운영체제에서의 응용 프로그램은 일반 리눅스 사용자라면 쉽게 작성이 가능하다.

```
pthread_t       tid1, tid2, tid3 ;
pthread_attr_t  attr1, attr2, attr3 ;
sem_t sem ;
int result ;

sem_init( &sem, 1 );

result = pthread_create( &tid1, &attr1, thread1, "Task1" );

if( result < 0 )
{
    printf("(%d)pthread_create failed.₩n", result);
}
…
…
void* thread1( void* pArg )
{
    int num = 1;
    Int fd;
    int i;

    fd = open( "LED", 1 );
    If( fd < 0 )
    {
        printf("GreenLED open failed!₩n");
        printf("Thread will be terminated..");
        pthread_exit(1);
        return NULL;
    }
    while( 1 )
    {
        sem_wait( &sem );
        for( i = 0; i < 10; ++i )
        {
            num ^= 0x1;
            printf("(T1)Write = %d₩n", num);
            write( fd, (void*)&num, sizeof(num) );
            usleep(300);
        }
        sem_post( &sem );
        sleep(3);
    }
    close( fd );
}

…
…
```

<표3> POSIX API를 지원하는 KHIX 임베디드 시스템 운영체제의 응용 프로그램

<표3>의 소스코드는 LED를 깜박거리는 테스트용 응용 프로그램의 소스코드 중 일부이며. 같은 소스코드를 PXA255, ATmega128에 이식하였다. 응용 프로그램의 소스코드에서 볼 수 있듯이 POSIX 기반의 API를 이용해 쓰레드를 생성하며, Semaphore, Mutex, Message Queue, sleet, usleet 및 기타 POSIX API 함수들의 사용이 가능하다.

또한 디바이스 드라이브의 인터페이스를 통해서 open(), close(), read(), write(), ioctl()의 함수들을 사용할 수 있다. 위의 응용 프로그램은 동일한 소스코드를 사용하여 고성능의 PXA255, 저성능의 ATmega128에서 정상 작동 하는 것을 확인하였다.

5. 결론 및 향후 연구 과제

임베디드 시스템은 특수한 목적을 위해 설계된 시스템이다. 최근에는 하드웨어의 발전으로 특수한 목적뿐만 아니라 인터넷 연결, 멀티미디어 처리 등 PC환경에서 사용하던 기능들이 임베디드 시스템에서 요구되어 최근의 임베디드 시스템은 기존의 실시간 운영체제의 사용에서 점차로 범용 운영체제를 사용하고 있다.

실시간 운영체제의 경우 각각의 API가 존재하기 때문에 응용 프로그램 작성 시 해당 API를 새롭게 익혀야 되는 단점이 있고, 범용 운영체제는 기능은 다양하고 POSIX API를 지원 하지만 커널 이미지의 크기가 커서 센서와 같은 메모리 크기가 작은 저성능의 시스템에는 커널 이식이 불가능하다.

본 논문에서의 KHIX 임베디드 시스템 운영체제는 확장 및 재구성이 가능한 운영체제로 다양한 종류의 MCU에 쉽게 이식이 가능하고 필요로 하는 기능을 재구성하기 쉽도록 설계되었으며 POSIX API를 지원해 응용 프로그램작성에 용의하도록 설계, 구현하여 고성능의 PXA255, 저성능의 ATmega128에 KHIX 임베디드 시스템 운영체제의 커널을 이식하였다.

향후 연구과제로는 점차로 범용화 되는 임베디드 시스템을 위해 인터넷 연결성, 멀티미디어 처리를 위해 다양한 네트워크 지원, 파일시스템, GUI 기능을 구현하고 사용자가 쉽게 시스템에 요구되는 기능들을 선택해 커널 이미지를 생성하는 Configuration Tool이 제작되어져야 될 것이다.

6. 참고문헌

[1] VxWorks, http://www.windriver.com
[2] pSOS, "pSOS System Concepts", 1996.
[3] Nucleus, http://www.mentor.com
[4] μC/OS-II. http://www.ucos-ii.com
[5] Velos, http://www.mdstec.com
[6] TinyOS, http://www.tinyos.net
[7] 나노 Qplus, http://www.qplus.or.kr
[8] Embedded Linux, http://embedded-linux.org
[9] Win CE, http://www.microsoft.com
[10] 김선자, 김흥남, 김채규, “임베디드 운영체제 표준화 동향”, 정보처리학회지 제9권 1호, 2002년 1월

[11] Institute for Elecrical and Electronic Engineers. In IEEE Std 1003.13-1998, IEEE Standard for Information Technology - Standardized Application Environment Profile-POSIX Realtime Application Support (AEP).
[12] 이형석, 정영준, “임베디드 운영체제 커널 기술 동향”, 전자통신동향분석 제21권 제1호 2006년 2월.
[13] 김정환, “전세계 임베디드 S/W 시장 동향”, 정보통신진흥연구원, 주간 기술동향 통권 1107호, 2003년 8월.
[14] MicroC/OS-II 실시간 커널 제2판, 성원호 역, Jean J.Labrosse 저, 에이콘 출판.
[15] Intel PXA255 Processor web page, http://www.intel.com/design/pca/products/pxa255/techdocs.htm
[16] ATmega128 Datasheet, http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf
[17] 박승민, “센서 네트워크 노드 플랫폼 및 운영체제 기술 동향”, 전자통신동향분석 제21권 제 1호