특집

임베디드 OS 기술 동향

노 학 종* 김 강 진* 김 문 회**

목 차

1. 서 론

임베디드 OS는 컴퓨터 상에서 컴퓨터의 자원을 효율적으로 관리하고 다양한 기능을 지원하는 범용 OS와는 달리 마이크로프로세서가 내장된 제한된 하드웨어를 가지고 주변 상황을 고려하며 요구되는 기능을 효율적으로 수행해야 하는 임베디드 시스템을 위해 작은 크기로 최적화된 운영체제다. 〈표 1〉은 임베디드 OS와 범용 OS 간의 일반적인 차이점을 보여주고 있다.

<표 1> 임베디드 OS와 범용 OS

속성	임베디드 OS	범용 OS
가상 메모리	사용 않음	사용
응용	단일 목적	범용
스케줄링	우선 순위 기반	fairness
IO 처리	응용 프로그램	디바이스드라이버 사용
태스크 간 대용량 데이터 전달	포인터만을 전달하는 등 속도 향상 기법 사용	pipe, mailbox 등의 시스템 서비스 사용
자원 할당	정적	동적(run-time)
파일 시스템또는 디스크	사용 않거나제한된 사용	사용

* 건국대학교컴퓨터정보통신공학과 석사과정
** 건국대학교 컴퓨터공학부 교수

임베디드 OS를 필요로 하는 임베디드 시스템의 종류는 매우 다양하다. 예를 들면 PDA, 휴대폰과 같은 정보 단말 기기, 그리고 냉장고, 디지털 TV와 같은 정보 가전 기기로부터 산업용 로봇, 공장 자동화와 같은 산업용 제어기기, 그리고 자동차 엔진 제어, 항공기 운행 제어, 고속전철 신호제어와 같은 오동작 시 인명피해까지 예상되는 안전 동작을 필수로 하는 교통 제어 시스템까지 여러 규모의 다른 특성을 가진 임베디드 시스템이 있다. 이러한 다양한 유형의 임베디드 시스템은 그 특성과 목적에 따라 서로 다른 종류의 임베디드 OS를 사용하여 왔다.

우선 임베디드 OS를 필요로 하는 임베디드 시스템들이 일반적으로 갖는 특성을 살펴보면 다음과 같다.

- 임베디드 시스템은 운용환경과 긴밀하게 상호작용(interaction)하며 그 기능을 수행한다. 이를 위하여 실시간 동작(Real-Time Operation)을 필요로 하는 경우가 대부분이다.
- 설계 시에 크기, 무게, 저전력, 험악한 운용환경, 생산가격, 신뢰성, 안전 동작 등 고려하여야 할 제약사항이 매우 많다.

임베디드 OS는 위와 같은 임베디드 시스템의 요

구사항을 효율적으로 충족시키기 위하여 다음과 같은 속성을 갖추어야 한다.

- 예측성(predictability) : 오동작 시 막대한 피해가 예상되는 응용에서는 시스템 동작의 응답시간이나 그 영향이 예측 가능하여야 오동작을 미연에 방지하고 오동작이 예상될 시 예외처리를 할 수가 있다.
- 성능(performance) : 제한된 자원이나 열악한 운용 환경에서 성능을 가능한 한 최적화하는 것이 바람직하다.
- 조립성(configurability) : 자원이 제한된 상황에서 오직 필요한 기능만을 갖추는 것이 바람직하다.
- 적응성(adaptability) : 동적으로 변화하는 주변 환경에 신속하게 대처할 수 있는 능력이 필요하다.

이외에도 유연성(flexibility), 신뢰성(reliability), 보안성(security and privacy), 안전성(safety) 등 필요에 따라 요구되는 여러 속성이 있다.

본 고에서는 다음 장에서 임베디드 OS의 일반적인 구성요소, 특징, 분류, 적용 분야 등 일반적인 사항들을 살펴보고, 3장에서는 임베디드 OS가 갖추어야 할 기능들을 간략하게 기술한다. 4장에서는 기존의 임베디드 OS들을 살펴보고 그 중 일부를 비교하고, 5장에서 앞으로 임베디드 OS들이 갖출 것으로 예상되는 기능들을 기술하고 본 고의 끝을 맺는다.

2. 임베디드 OS의 적용분야 및 분류

2.1 임베디드 OS의 적용분야

기존의 임베디드 시스템은 간단한 스케줄러를 기반으로 특정 기능 수행을 위한 임베디드 프로그램만으로도 기능을 수행하였지만, 근래에는 임베디드 시스템의 성능이 증대되면서 임베디드 OS라는 운영체제의 개념을 도입하게 됐다. 이러한 임베디드 시스템의 응용분야가 확장되고 복잡한 기능들이 요구됨에 따라 임베디드 OS도 목적에 따라 지속적으로 발전되고 사용이 증가하고 있다.

임베디드 OS가 사용되는 임베디드 시스템들을 살펴보면, 위성과 같은 우주항공 분야와 미사일, 레이더와 같은 군사용 제어장치나 자동차, 교통관리 시스템, 주차 관리 시스템, 보안 시스템, 로봇, 원격 시스템, 공장 자동화와 같은 산업용에도 사용되며 홈-시큐리티, 가전제품 원격제어, 사이버 아파트 등의 홈-오토메이션에도 사용된다. 특히 우리 생활공간에서 쉽게 접할 수 있는 TV, 셋톱박스, 전자 밥솥, 세탁기, 오디오 등의 가전제품과 이동전화기나 PDA와 같은 개인정보 단말기에도 사용되고 있다. 이렇듯 우리 일상 생활에 임베디드 시스템이 일반화되어 있고, 또한 임베디드 OS도 우리가 미처 깨닫지 못할 정도로 여러 범주에서 사용되고 있다.

임베디드 OS가 적용되는 분야는 목적에 따라 자동차용 실시간 운영체제, 항공기용 실시간 운영체제, 네트워크 라우터/스위치용 실시간 운영체제, 센서 네트워크용 실시간 운영체제, 모바일 장비용 실시간 운영체제 및 비 실시간 운영체제 등으로 분류할 수 있다[2]. 따라서 임베디드 OS도 적용되는 분야에 따라 특화되어 개발되고 있다. 자동차모바일 장비용 실시간 운영체제 등은 복잡한 기능을 지원해 주어야 하기 때문에 멀티쓰레드 기반 임베디드 OS가 적용되어야 한다. 또한 실시간성도 요구된다. 특히 항공기용 실시간 운영체제를 사용하는 임베디드 시스템은 실시간성을 요구한다. 만일 실시간성이 보장되지 않는 임베디드 OS를 사용할 경우에는 시스템은 물론 인명 피해까지 초래할 수 있기 때문이다. 이렇듯 일부를 제외한 임베디드 시스템들이 특성 상 실시간성을 요구하기 때문에, 대부분의 임베디드 OS가 RTOS(Real-Time Operation System)의 속성을 가진다.

그렇다면 왜 임베디드 OS를 사용해야 하는 것일까? 먼저 가격을 이유로 들 수 있다. 임베디드 시스템은 범용 컴퓨터와 달리 프로세서의 성능이나 메모리 용량을 기준으로 좋은 평가를 받는 것은 아니다. 단지 특정한 용도에 필요한 최소한의 기능만 적절히 수행하면 된다. 따라서 최소한의 자원만을 사용하는 시스템 구성을 통해 최저의 생산비용을 유지해야만 한다. 즉 임베디드 시스템마다 그 특성에 따라 다른 최적의 운영체제가 필요하기 때문이다. 하지만 일부 수요가 적은 임베디드 시스템의 경우는 범용 운영체제를 이용해서 개발하기도 한다. 임베디드 OS를 사용하는 또 다른 이유는 인공위성이나 미사일 제어 등과 같이 오류에 견고해야 하는 시스템이나 실시간 시스템을 위해 최적화된 운영체제가 필요할 경우다[3,4]. 이러한 경우에는 다양한 목적의 범용 운영체제보다는 RTOS (Real Time Operating System)와 같은 임베디드 OS를 사용하는 것이 바람직하다.

또 다른 이유는 응용 소프트웨어 개발의 용이성에 있다. 일반적으로 임베디드 OS는 작고 빠른 그리고 우선 순위 기반의 스케쥴링 기능이 포함된 실시간 커널, 응용 소프트웨어의 개발 기간 단축을 위한 라이브러리 또는 미들웨어, 그리고 IDE(Integrated Development Environment), 원격 디버거, 시스템 이벤트 모니터링 도구 등의 응용 소프트웨어 개발 도구로 구성되어 있다. 라이브러리 또는 미들웨어, 그리고 개발 도구의 제공은 응용 소프트웨어 개발 생산성의 향상에 지대한 영향을 미친다.

2.2 임베디드 OS의 분류

임베디드 OS는 보는 관점에 따라 여러 방법으로 분류할 수 있으나 〈표 2〉와 같이 보편적으로 태스크 실행 주체에 따라 쓰레드(thread) 기반 임베디드 OS와 프로세스(process) 기반 임베디드 OS로 구분한다.

프로세스 기반 운영체제는 응용 프로그램과 운영체제 커널이 다른 주소 공간에서 실행된다. 따라서 응용 프로그램의 프로세스는 운영체제 커널이 관리하는 자원을 직접 사용할 수 없으며 커널을 통해서만 가능하다. 즉, 응용 프로그램에서 오류가 발생하더라도 운영체제에는 심각한 손상이 발생하지 않는다. 쓰레드 기반 운영체제는 응용 프로그램과 운영체제 커널이 동일한 주소 공간을 공유함으로써, 문맥 교환 시간(Context Witching Time)이 빠르고 커널 자원을 제한 없이 사용할 수 있다. 따라서 실시간(Real-Time)성이 요구되는 전문적인 시스템에서 많이 사용한다. 그러나 쓰레드 기반 운영체제는 응용 프로그램의 오류가 운영체제 커널에도 그대로 영향을 미치기 때문에 시스템 전체를 손상시킬 수 있는 단점이 있다[1].

〈표 2〉에는 현재 많이 사용되고 있는 임베디드 OS들이 열거되어 있으며 이들의 대부분이 공통적으로 갖고 있는 특징은 다음과 같다.

- 실시간 POSIX API 표준 준수
- Modularity 및 Scalability 지원
- Context Switching Time, Interrupt Latency, Semaphore Get/Release Latency의 극소화를 통한 성능 개선
- 비 선점형(Non-Preemptive) 부분의 최적화
- 우선 순위 기반의 스케쥴 가능한 Interrupt 처리 기능
- 실시간 스케쥴링 지원
- 간단한 메모리 관리(Memory Management) 기능

등 또한 임베디드 OS는 실시간성 지원이 이루어진 형태에 따라 다음과 같은 4부류로 구분하기도 한다.

- 작고 빠른 독자적인 커널 기반 임베디드 OS : 빠른 Context Switch, 외부 Interrupt에 대한

〈표 2〉 임베디드 OS의 분류

Multi Thread 임베디드		Multi Process 임베디드	
VxWorks	세계 시장에서 점유율이 가장 높다. 많은 종류의 마이크로 프로세서를 지원하며 대부분의 상용 Chip에 대한Device Driver도 모두 지원한다	LunxOS	LinuxWorks 사에서 개발, 판매하고 있는 Embedded Linux RTOS. Unix와 호환 가능하며 OS의 사이즈가 크고, 복잡하고 규모가 큰 Real-Time Application 개발에 적합
VRTX	국내 시장 점유율이 가장 높았던Mentor Graphics사의 RTOS로서 현재 판매량이 감소하고 있다	QNX	UNIX와 호환가능하며 비상업용으로는Real-Time Platform Package를 무료다운로드 받을 수 있다
pSOS	VxWorks와 함께 세계시장 점유율 상위 랭크	OS-9	Microware사에서 개발, 판매. 국내보다는 세계시장에서 높은 인지도와 시장 점유율
Nucleus Plus	Full source Code 제공하며 휴대폰 단말기와 PDA등 50 여종의 제품에서 사용	RTLinux	Finite State Machine Labs사에서 개발, 판매하는 Embedded Linux
SuperTask	Open Source Code, No Royality	Windows CE.NET	Microsoft에서 판매하는 임베디드 Windows OS
microC/OS	학교중심으로 많이 사용되며 널리 알려진 임베디드 OS		

빠른 반응, 우선 순위 기반 스케쥴링, 실시간 메시지 기능 등을 통하여 커널에 의한 실행시간 오버헤드를 줄여 커널 실행을 빠르게 수행하나 시간 제약, 자원 할당 등을 직접적으로 처리하지는 않는다. QNX, VxWorks, pSOS, LynxOS 등 대부분의 상용 RTOS들과 eCos가 이 부류에 속한다.

- 범용 OS에 실시간성 지원 기능 확장한 임베디드 OS : 스케쥴링, interrupt 처리, IPC, 파일 시스템, I/O 지원 기능, 메모리 관리 부분 등이 수정되어야 한다. 범용 OS가 갖는 다양한 기능과 좋은 소프트웨어 개발 환경을 갖고 있으나 연성 실시간 임베디드 응용에만 사용할 수 있다. RT-Linux, Chorus 등이 이 부류에 속한다.
- 연구용 임베디드 OS : 특정 실시간 프로세스/객체 모델에 기반을 두고 있으며 응용 수준의 예측성 제공, 결함허용성 지원, 분산 응용 지원 기능을 갖추고 있다. MARS, SPRING, MARUTI, TMO-Linux 등이 이 부류에 속한다.
- 윈도우 기반 임베디드 OS : Windows CE.NET, XP Embedded 등이 이 부류에 속한다.

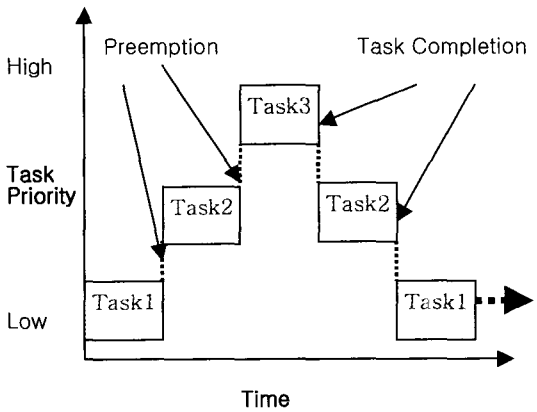
3. 임베디드 OS에 요구되는 기능

임베디드 시스템의 적용 범위가 확산됨으로 인해 임베디드 OS의 신뢰성이 점차 중요시 되고 있다. 다음은 임베디드 OS(실시간/비 실시간)가 신뢰성을 갖기 위해 요구되는 기능들에 대한 설명이다.

- ○ 실시간 처리 : 임베디드 시스템에서는 대부분 제한된 시간 이내에 요구사항을 처리하여야 하는 응답성이 요구된다. 이러한 응답성을 만족시키기 위해서는 태스크(Task)의 실시간 스케줄링 및 비동기적 이벤트의 효율적 처리가 요구된다. 즉 태스크 수행 시 최악의 경우의 지연시간(Worst-Case Latency)을 보장하는 예측 가능한 실시간 스케줄링 메커니즘을 지원해야 한다. 또한 비동기적 이벤트(Asynchronous Event)에 대해 최대 지연시간을 보장함으로써 외부사건에 대한 시스템 응답성을 만족하여 주어야 한다.
- ○ 스케쥴러 : 임베디드 OS는 태스크(Task)라 불리는 프로그램 수행 단위를 가지게 된다. 보통 복수 개의 태스크가 동시에 수행되게 되며 OS 내부의 스케쥴러에 의해서 다음 번에 수행되어야 할 태스크를 선택하게 된다. 이때 사용되는 임베디드 OS의 스케쥴링으로는 다양한 알고리즘이 나와 있지만, 구현상의 문제 등으로 대부분의 임베디드 OS에서는 (그림 1)과 같은 우선 순위에 기반을 둔 스케쥴링 알고리즘을 사용한다. 우선 순위 기반 스케쥴링 방식

으로는 우선 순위가 설계 시에 정적으로 결정되는 Rate Monotonic 방식과 같은 정적 방식과 EDF(Earliest Deadline First) 방식, Least Laxity 방식과 같은 우선 순위가 실행 중에 동적으로 변화하는 동적 방식이 있다.

- 태스크 통신 : 멀티 태스킹 지원시 태스크 간 통신이나 동기화 메커니즘이 제공되어 서로 통신할 수 있어야 한다. 이는 OS에서 지원하는 여러 기능들에 의해 수행이 된다. 대표적인 방법으로는 시스템 전역의 변수를 선언하여 사용하는 방법이 있으나 이 경우에는 변수 자체가 임계지역(Critical Section)이 되므로 Interrupt Enable/Disable이나 세마포어 등으로 상호배제(Mutual Exclusion)를 해야 한다. 그밖에 큐, 파이프, 메일박스, 공유 메모리 등을 사용하는 방법이 있으며 이들은 커널에 의해 자동으로 상호배제(Mutual Exclusion)가 이루어져야 한다.



(그림 1) 선점형 우선순위 스케쥴링

- 선점형 커널(Preemptive Kernel) : 선점형 커널(Preemptive Kernel)이란 시스템의 응답이 매우 중요한 경우에 사용하는 것으로, 커널이 작업을 수행 중인 경우에도 다른 태스크가 커널의 제어권을 가져올 수 있는 경우를 말한다. 즉, 선점형 커널은 우선순위가 높은 태스크로의 전환이 확실하다. 만약 우선 순위가 높은 태스크가 준비 상태일 때 커널 관련 태스크가 실행 중이라도 그 순간 Interrupt가 되고, 우선 순위가 높은 태스크로 Context Switch가 이루어진다.
- 사용자 개발도구 지원 : 임베디드 시스템의 특성을 고려해 볼 때, 특정 기능을 수행하는 어플리케이션을 제외하고는 제한된 자원을 가지고 있는 경우가 대부분이다. 따라서 사용자가 임베디드 시스템 상에서 직접 응용 소프트웨어를 개발하거나 새로운 기능을 추가하는 것이란 매우 어려운 일이다. 이 경우에는 사용자가 별도의 개발 시스템을 이용하여 응용 소프트웨어를 개발하고 임베디드 시스템에 이미지를 설치/구동 시킬 수 있도록 편리한 응용 개발 환경과 개발 도구를 지원하여야 한다.

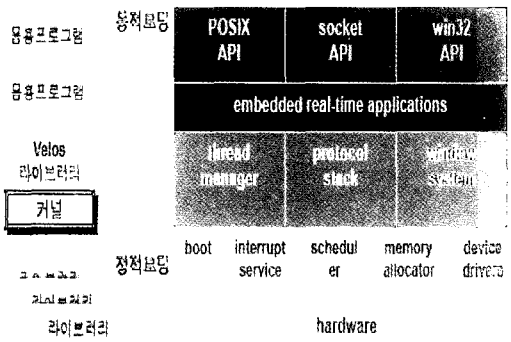
4. 기존의 임베디드 OS

본 장에서는 임베디드 OS를 실시간성 지원 여부에 따라 실시간 임베디드 OS와 비 실시간 임베디드 OS로 나누어 기존 임베디드 OS들의 특징을 간략히 살펴 본다. <표 2>에 열거한 임베디드 OS들을 모두 포함하고 있으며 그외 다수의 임베디드 OS가 기술되어 있으며 소수의 임베디드 OS만 커널의 구조를 포함하였다. 뒤에 나오는 <표 3>은 본 장에 기술한 임베디드 OS 별로 주 응용 분야 및 특징을 나타내고 있으며 <표 4>는 실시간 OS 들 중의 일부분만 프로그래밍 언어, 타겟 프로세서, 호스트에서 사용하는 OS, 그리고 API를 비교한 것이다.

4.1 실시간 임베디드 OS

- Velos[5] : MDS테크놀러지의 'Velos'는 국내 최초의 상용 RTSO로서의 의미가 크다. POSIX 기반의 시스템 API와 표준 C/C++ 지

원으로 어플리케이션 개발이 매우 용이할 뿐만 아니라, 기존 오픈소스 프로그램의 포팅을 쉽게 할 수 있다는 장점이 있다. 또한 실시간 성능을 필요로 하거나 빠른 부팅이 요구되는 분야에 최적화되어 있다. Velos의 주력 분야는 멀티미디어 단말기기, 산업용 컨트롤러, 로봇과 무인 항공기 등이며, 현재 ARM 기반의 프로세서와 Velos 기반의 지능형 빌딩 컨트롤 시스템(시설 관리 시스템), 출입통제 시스템 등 전통적인 임베디드 분야에 두각을 나타내고 있다. (그림 2)는 Velos의 커널 구조를 보여주고 있다.



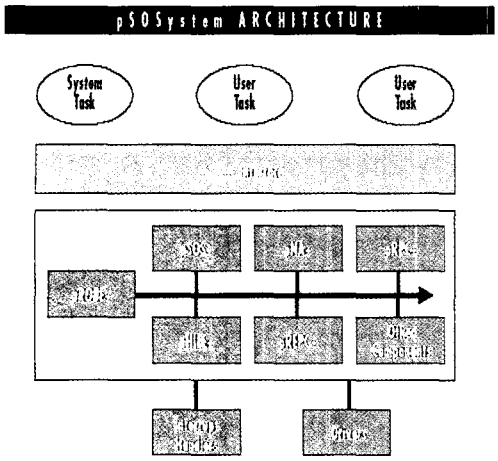
(그림 2) Velos 커널 구조 [5]

● Palm OS[7] : PDA 분야에서는 대표적인 OS로 Palm OS가 있다. PIMS나 PDA 사용자 편의성, 그래픽 등에서는 편리하나, OS 자체의 제약이 너무 많아서 멀티미디어 등에 매우 취약한 단점이 있다. 또한 데이터나 프로그램 등도 Palm OS 용으로 변환해서 넣어야 하는 등의 단점도 있다. 현재 지속적으로 개선되고 있어 앞으로 멀티미디어 분야에서도 강한 면모를 보여 줄 수 있는 것으로 판단된다. 외국에서는 Pocket PC와 Palm OS의 점유율이 비슷하지만, 국내에서는 Pocket PC의 점유율이 압도적으로 높다.

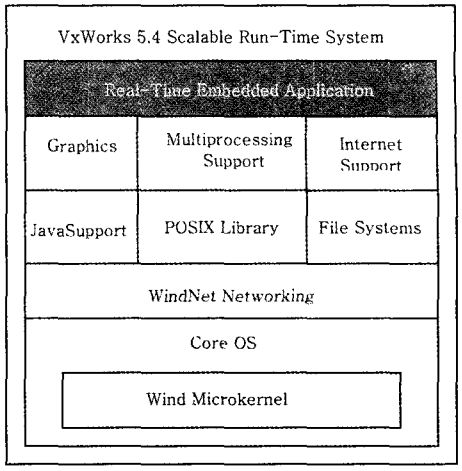
◎ VRTX : VRTX는 Mentor 사에서 개발한 RTOS로 VRTXoc와 VRTXsa 2가지의 커널을 가지고 있다. VRTXoc는 간단한 기능을 가지는 임베디드 시스템에 적합하고, VRTXsa는 메모리 보호를 필요로 하는 임베디드 시스템에서 사용한다. VRTX는 통신장비, 네트워크 장비, 모바일 기기 및 산업용 모니터링 시스템 등 여러 분야에서 사용하고 있다. VRTX는 POSIX 인터페이스와 I/O 인터페이스 라이브러리를 제공함으로써, MS-DOS 파일 시스템을 제공한다. 이 외에는 기존의 RTOS와 크게 다른 점은 없다.

◎ pSOS[9] : ISI에서 1980년대에 임베디드 프로세서를 위하여 모듈화되고 고성능, 신뢰성, 사용 편이성에 주안점을 두고 개발한 pSOS는 우리나라의 여러 업체가 채택해서 사용하고 있는 RTOS로 삼성전자가 실시간 멀티태스킹 커널인 pSOS+ 개발에 참여해 라이센스를 갖고 있어 잘 알려져 있다. 삼성전자의 휴대폰에 사용되어 왔으며, 각 모듈들이 헤더, 데이터, CRC Check-Sum으로 이루어져 모듈의 안정성과 높은 보안성을 제공함으로써, 각종 통신장비와 네트워크 장비에서 주로 사용되고 있다. pSOS는 커널을 중심으로 해서 여러 개의 소프트웨어 컴포넌트들로 구성되어 있다. 이들 소프트웨어 컴포넌트들은 각각의 독립적인 모듈로 되어 있다.

pSOS는 선점형 스케쥴링 방식의 멀티태스킹 RTOS로서, 각 태스크들은 우선 순위를 가지고 있다. 즉, 우선 순위가 높은 태스크들의 작업 수행이 먼저 이루어지며, 같은 우선 순위를 가지는 경우는 스케쥴링 방식이 라운드로빈(Round-Robin)방식으로 바뀐다. (그림 3)은 pSOS의 커널 구조를 보여주고 있다.



(그림 3) pSOS 커널 구조 [9]

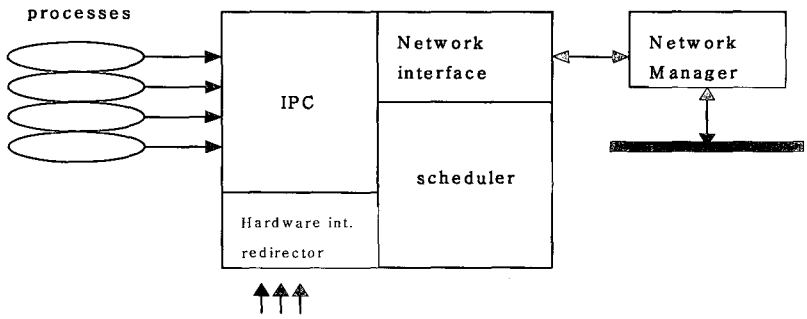


(그림 4) VxWorks의 구조 [9]

● VxWorks[9] : 윈드리버 사의 RTOS인 VxWorks는 pSOS와 유사한 점이 많다. pSOS가 통합개발환경으로 pRISM+를 제공한다면 VxWorks는 개발환경으로 Tornado를 제공한다. VxWorks는 RTOS 중 세계 시장 점유율이 제일 높으며 그 이유가 Tornado에 있다고 할 만큼 Tornado는 높은 평가를 받고 있다. VxWorks의 커널인 마이크로 커널은 선점형 멀티태스킹 방식이며 태스크의 우선순위는 256 단계로 나누어지며, 스케쥴링 방식도 높은 우선순위를 가지는 태스크가 먼저 실행하는 방식을 지원한다. 만일 같은 우선순위를 가진다면 라운드로빈 방식의 스케쥴링을 이용하는 것도 pSOS와 유사하다. 그리고 pSOS가 여러 개의 컴포넌트로 되어 있다고 하면 VxWorks는 200개 가량의 모듈들로 구성되어 있어 개발자는 이들 중 필요한 모듈만을 사용해서 시스템에 맞는 운영체제를 구성할 수 있다. VxWorks는 다른 RTOS들이 지원하는 거의 모든 서비스를 지원하고 있다. 태스크간의 통신을 위해 세마포어와 메시지 큐, 공유 메모리, 소켓, 시그널 등을 제공하고, 표준 TCP/IP 네트워킹과 ROM이나 로컬 디스크, 네트워크로 부팅이 가능하게 되어 있다. (그림 4)는 VxWorks의 내부 구성을 보여주고 있다.

● QNX[10] : X86 CPU 플랫폼에서 20년 이상 실시간 운영체제 경험을 바탕으로 둔 QNX는 마이크로커널의 특징을 발전시켜 개발하여 왔으며, QNX Neutrino라는 마이크로커널을 코어로 내장시키고 나머지 부분은 프로세스의 형태로 모듈화하여 SMP 서버와 같은 대형 시스템에서부터 작은 임베디드 시스템까지 광범위하게 사용 가능하다. 현재 사용하는 대부분의 MCU를 지원하고 있고, 네트워크 및 높은 실시간성의 성능을 가지고 있는 것이 특징이다. 텔레매틱스 기반 기술의 세계적인 선두 주자인 QNX OS는 해외에서 높은 점유율을 가지고 있으며 그 커널 구조는 (그림 5)에서 보여주고 있다.

● Windows CE.NET[21] : Wiindows CE.NET은 1996년 MS 사에서 개발한 Windows CE에서 비롯된 것이며, MS 사에서 제작된 기존의 윈도우 인터페이스에 모바일 네트워크 기능을 강화해 정보 가전 기기 및 PDA 등에 주로 사용되고 있다. 적외선 통신, 웹 브라우징 기능



(그림 5) QNX의 커널 구조[1]

이외에도 편리한 개발환경을 가지고 있으며 UI가 기존의 윈도우와 매우 유사하게 제작돼 있어 대부분의 사용자들은 별도의 학습 없이도 바로 사용이 가능하다. 또한 PC 용의 윈도우 OS나 각종 오피스 프로그램들과 완벽한 호환이 가능해 기존의 데스크톱이나 노트북에서 가능하던 각종 어플리케이션들을 PDA에서도 그대로 사용할 수 있다는 이점이 있다. 그러나 제한적인 하드웨어를 가진 임베디드 시스템에 너무 많은 기능을 구현하려 했기 때문에 오히려 실행 속도가 느린 단점이 있다.

- OS-9[11] : Microware 사에서 개발, 판매하는 RTOS로서, 국내 보다는 세계시장에서 높은 인지도와 시장 점유율을 가지고 있다. OS-9은 현재 대부분의 상용 MCU를 지원하고 있고 개발 도구와 소프트웨어 컴포넌트 타입으로 구성되어 있다. 이 소프트웨어 컴포넌트들로는 OS Kernel, Networking, Graphics, Power Management 등이 있다.
- LynxOS[12] : LinuxWorks 사에서 개발, 판매하고 있는 Embedded Linux RTOS로 UNIX와 호환이 가능하나 OS 크기가 크다는 단점이 있다. LynxOS는 복잡하고 규모가 큰 실시간 응용 개발에 적합하며, 항공기 실시간 시스템을 주요 타겟으로 개발된 경성 실시간 시스템용 임베디드 OS다.

○ ThreadX[13] : Express Logic사에서 개발/판매하고 있으며, 국내에는 잘 알려지지 않은 제품이다. 이 제품을 구입하면 Source Code가 제공되며, Royalty가 없다는 장점이 있다. ThreadX는 현재 대부분의 상용 MCU를 지원하고 있고, 특이한 사항으로는 타 임베디드 OS와 호환될 수 있는 API 라이브러리를 제공한다는 것이다.

4.2 비 실시간 임베디드 OS

○ Qplus[23] : 한국전자통신연구원(ETRI)에서 개발한 'Qplus'는 리눅스를 기반으로 하고 있으며, 'ESTO' 등의 개발도구를 ECLIPS 플랫폼에 플러그-인 형태로 내장하는 연구가 한창 진행되고 있다. Qplus의 주력분야는 디지털-홈, 텔레매틱스, 휴대용 단말(차세대 PC, 이동통신)이다. 특히 디지털 홈 분야에서는 홈 게이트웨이, IP 셋톱박스의 상용화 단계에 있으며, 그 이외에도 텔레매틱스 단말, 로봇, 차세대 PC 등의 기기에 시제품 수준의 개발이 진행되고 있다. Qplus는 낮은 로열티로 개발비용의 부담은 덜 수 있으나, 업체들의 기대감에 비해 안정성 및 지속적인 기술지원에 대한 확신을 주지 못하고 있다. 따라서 문제점 보완 및 서비스를 지원하는 수행 업체 또는 기관의 필요성이 요구되고 있다.

● 미지리눅스[22] : 미지리서치의 미지리눅스는 리눅스를 기반으로 하고 있으며, 스마트 폰에 장착된 리눅스 기반의 상용 OS로 자체 개발한 브라우저와 이메일 클라이언트, PIMS, 미디어 재생기, 문서 편집기 등 다양한 응용 소프트웨어를 포함하고 있다. 현재는 시스템의 안정성과 UI 성능을 강화하는 작업을 진행 중이다. 미지리눅스는 스마트 폰, 텔레매틱스, 홈 네트워크 등의 시장에 주력하고 있으며, 응용에 따라 "미지 스마트폰 에디션", "미지 텔레매틱스 에디션", "미지 디지털 홈 에디션" 등의 플랫폼 개발환경을 제공하고 있다.

● Symbian (Epoc OS)[6] : Symbian은 영국에서 개발된 것으로, 주로 핸드폰 시장을 주력하여 만들어진 임베디드 OS다. 이 OS는 현재 유럽시장 전역에 걸쳐서 많은 점유율을 가지고 있으며, 대표적인 것으로 Nokia, Fujitsu, Sony Ericsson, Motorola 등에서 Symbian을 채택하여 사용하고 있다. Symbian은 원래 i386만을 지원했지만, 최근 Strong ARM 등 ARM 계열로 주요 타겟을 확장하고 있다.

● Windows XP Embedded[21] : MS사에서 개발된 기존의 범용 OS(Windows 계열)의 대부분 기능을 임베디드 OS로 가져온 것이다. PC용의Windows OS나 각종 오피스 프로그램들과 완벽한 호환이 가능할 뿐만 아니라, 기능면에서도 Windows CE.NET과 매우 흡사하다. 주로 정보가전기기, 셋톱박스 및 네트워크 장비를 타겟으로 개발되었다.

4.3 공개 소스 기반의 임베디드 OS

● Nucleus[8] : Accelerated Technology사의 'Nucleus'는 저작권 없음(Royalty Free)이라는 정책 때문에 순식간에 퍼지고 있는 OS다. 현재 국내의 삼성이나 현대에서도 Nucleus의 무저작권 정책 때문에 이를 자사의 ARM 기반의 CPU에 포팅/지원하고 있는 실정이다. Nuclues가 포팅되어 있는 CPU는 매우 많으며, ARM, MIPS, PPC, M68K, SH 등 마이크로프로세서를 비롯하여, Analog, TI 그리고 Hitachi의 DSP도 지원하고 있다. 게다가 등록만 하면 제한된 버전이라도 데모 버전을 받아서 쓸 수 있다는 장점이 있다.

● 트론 (TRON, ITRON, uITRON)[15] : 트론(TRON)은 1984년 일본에서 만들어졌으며, 일본 전자업계를 중심으로 발전했다. 트론(TRON: The Real-ime Operating system Nucleus)은 원래 전화나 가전 등 주변의 모든 도구가 인터넷에 연결되는 것을 가정해 개발된 OS다. 성능이 그다지 높지 않더라도 신속히 작동하며, PC의 OS처럼 작동에 시간이 걸리거나 이따금 정지하는 법이 없어서 안전성이 탁월하다는 평가를 받고 있다. 리눅스처럼 트론도 '개방(Open Source)'을 기본으로 하고 있어 누구라도 자유롭게, 그리고 무료로 그 성과를 이용할 수 있기 때문에 주목 받고 있다. 그러나 Open Source를 지향한다고 하지만, 사양과 커널 만 공개할 뿐, 실제 트론에 기반한 소프트웨어를 개발할 때 필수적인 미들웨어 등은 공개하고 있지 않다.

● RT-Linux[14, 18] : 기존의 리눅스는 Time-Slice에 의한 스케줄링 방식을 가지고 있다. 따라서, 지금 수행하고 있는 프로세스보다 다음에 수행될 프로세스가 우선순위가 높아도 바로 지금 수행되는 프로세스가 중단되는 것이 아니고 Time-Out이 되어야만 현재의 프로세스가 CPU를 놓아준다. 이런 이유로 인해서 기존의 리눅스는 실시간성 지원 기능이 떨어지는 것이다. RT-Linux는 기존의 리눅스에 비해 실시간성이 향상된 것은 사실이지만

기존의 리눅스 커널에다 새로이 실시간 커널을 추가한 것으로 자원이 제약된 임베디드 용으로는 OS의 덩치가 큰 것이 단점이다. (RT-Linux의 경우는 상용과 비상용 즉, Open-Source와 Non Open-Source 두 가지 버전이 존재한다)

- SuperTask[16] : US Software 사에서 개발, 판매하는 RTOS다. Nucleus와 마찬가지로 Source Code를 공개하고 있으며, Royalty가 없다. 현재 국내에서는 자세히 알려지지 않은 임베디드 OS다.
- Micro C/OS (uC/OS)[17] : 최근에 학교를 중심으로 많이 사용하면서 널리 알려진 RTOS다. Jean J. Labrosser라는 사람이 개발하여 배포한 작은 크기의 RTOS며, 책을 구입하면 부록에 Source Code를 제공하는 형태로 판매되는 OS다. 역시 Royalty가 없으며, 꾸준한 Upgrade를 통하여 많은 종류의 프로세서를 지원하고 있다. 국내에도 널리 알려진 임베디드 OS며, 간단한 기능을 가진 기기에 쓰일 수 있다.
- 임베디드 리눅스[14, 18] : 임베디드 리눅스는 소스 공개와 아울러 안정성, 신뢰성 및 성능의 확보에 따라 활용도가 급격히 증대되고 있으며, 레드햇, 몬타비스타, 리니오 등이 임베디드 리눅스 개발 및 기술 지원 사업에 주력하고 있다. 그러나 임베디드 리눅스는 열악한 개발환경에 따른 개발기간에 대한 부담과 라운드-로빈 방식의 스케쥴링을 하므로 실시간 시스템에는 적합하지 않다는 문제점을 가지고 있다.
- eCos[19] : 레드햇 사에서 만든 공개 소스 기반의 실시간 임베디드 OS다. 다양한 프로세서에서 이식이 되고 있으며, 운영체제 차원에서 전력소비를 고려하여 설계되어 있어서 손쉽게 전력 정책을 구현할 수 있다. eCos는 임베디드 OS의 표준화 방안 중 하나인 uITRON와 POSIX를 위한 API를 지원하고 있다.

4.4 기타

- ⓟ L4[20] : 독일 Liedtke에 의해서 개발된 제 2세대 마이크로 커널이다. L4는 엄밀히 볼 때 임베디드 OS는 아니지만, 실시간성을 비롯하여 임베디드 OS에서 요구되는 기능을 대부분 수행할 수 있다. L4 커널은 임의의 위치에서 중단 가능하며, 인터럽트가 발생했을 때 우선순위에 의해서 처리할 수 있는 메커니즘을 제시하고 있다. 특히 높은 우선 순위의 쓰레드가 수행 가능한 상태가 되었을 때 최소 20밀리 세컨드 이내에 서비스를 받을 수 있다. 즉 최소한의 실시간성을 보장한다고 할 수 있다.
- ⓒ OSEK[24] : OSEK는 유럽에서 표준화된 대표적인 자동차 산업용 실시간 임베디드 OS다. 이 OS는 실시간 응용의 분산 결함허용 기능을 지원하기 위해서 다음과 같은 서비스를 제공한다.
  - Task Management : 태스크 종료와 활성화, 태스크의 상태 및 테스크 스위치를 관리
  - Synchronization : 태스크 동기화를 위한 Event Management와 Resource Management를 제공
  - Interrupt Management : 인터럽트 처리를 위한 서비스
  - Alarms : Relative/Absolute Time과 Static/Dynamic Time을 제공
  - Error Treatment : 다양한 오류로부터 안전을 유지할 수 있는 메커니즘

또한 스케쥴링 정책에 있어서는 선점 스케쥴링, 비선점 스케쥴링 또는 선점/비 선점 스케쥴링이 혼합된 스케쥴링 방식을 사용할 수 있다.

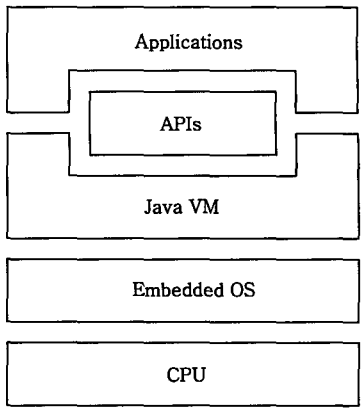
5. 임베디드 OS의 발전 방향

최근 임베디드 OS는 OS 이미지의 경량화, 저전력, GUI 및 임베디드 미들웨어 내장 등의 기술 개발에 초점을 맞추고 있다. 이를 증명하듯, Windows CE.NET와 OS-9과 같은 임베디드 OS는 Power Management 기술을 포함시킴으로써, 전력 소비에 취약점을 가진 Mobile 시스템 시장에서 좋은 평가를 얻고 있다.

상용 임베디드 OS의 경우는 Time-to-Market에 초점을 두어, 임베디드 시스템에 OS를 쉽고 최적으로 포팅하기 위한 수단으로 BSP(Board Support Package) 기술을 포함하고 있다. BSP 기능을 지원하는 대표적인 임베디드 OS로는 Windows CE.NET, pSOS, VxWorks 등이 있다. 즉, StrongArm, Xscale, MIPS, Hitachi, ARM 등의 프로세서를 기반으로 하는 임베디드 시스템에 최적의 OS를 구성하기 위한 Package를 제공하는 것이다. 이를 사용함으로써 개발자가 여러 차례의 시행착오를 겪지 않고도 최적의 OS를 구성할 수 있다.

또한 임베디드 OS의 기술 과제 중 하나로 임베디드용 미들웨어를 들 수 있다. 임베디드 미들웨어란 임베디드 OS와는 독립적으로 응용 소프트웨어들을 연결하여 이들이 서로 데이터를 교환할 수 있게 해 주는 소프트웨어로서, 예를 들면, Embedded Java, TMOSM, GOPI, MULTI-ORB 등이 있다.

한 예로 Java 환경은 Programming Language와 VM(Virtual Machine)인 Run-Time 환경으로 나뉘어 진다. 즉, 임베디드 시스템에서 동작하는 구조는 (그림 6)과 같다. 이 Java 환경을 사용함으로써 얻을 수 있는 이점으로는 향상된 생산성, 거대한 개발자 동호 모임, Networking, Security, Write Once and Run Anywhere 등이 있으며 셋톱박스, 홈 네트워킹 등의 분야에서 표준기술로 핵심적인 역할을 하고 있다.



(그림 6) Embedded Java

현재 진행되고 있는 연구의 추세와 CE Linux Forum 등의 표준화 활동을 살펴 보면 앞으로 임베디드 OS는 다음과 같은 기능을 포함할 것으로 예상된다.

- 분산 적응형 미들웨어에 대한 지원 기능이 강화될 것으로 보인다. 이를 위하여는 자원 예약, 동적 스케쥴링, 재구성, 환경 인식(Environment Awareness) 기능 등이 OS에 보강되어야 한다.
- 센서 네트워크의 노드와 같은 아주 작은 기기에 대한 저전력 지원 기능, 통신 경로를 자유로이 설정할 수 있는 기능 등이 보강될 것이다.
- 개발 환경의 사용자 편의성이 점점 더 향상될 것이고 기능도 다양화 할 것이다.
- 임베디드 OS가 더 모듈화하고 컴포넌트화하여 필요에 따라 필요 기능만 조합하여 사용할 수 있도록 조립성(Configurability)이 높아질 것이다.

6. 결 론

본 고에서는 임베디드 OS의 개념과 필요 요구에 따른 특성들을 살펴보았다. 앞서 소개한 바와 같이 임베디드 OS는 우리 생활공간 도처에서 사용되고 있고 또한 그 존재의 가치와 필요성에 대해서도 간

접적으로나마 느끼고 있다. 앞으로도 더욱 복잡하고 견고한 임베디드 시스템을 필요로 할 것이고 이에 적합한 실시간 임베디드 OS도 요구될 것이다. 따라서 타 임베디드 소프트웨어 분야와 더불어 임베디드 OS에 대한 지속적인 연구와 관심이 필요하다.

<표 3> 임베디드 OS 제품별 주 응용 분야 및 특징

	Real-Time	Multi-Thread	상용화 여부	주 응용분야	특 징
Qplus	△	○	○	디지털홈,텔레매틱스,휴대용단말	개발비용부담없으나 지속적기술지원여부불확실
Velos	○	○	○	멀티미디어단말기기, 산업용 컨트롤러, 로봇과 무인항공기	국내 최초의 상용 RTOS
미지 리눅스	△	○	○	스마트 폰, 텔레매틱스, 홈네트워크	리눅스 기반, 자체개발 애플리케이션 다수 존재
Symbian	△	○	○	핸드폰	유럽전역에 걸쳐 많은 점유율
Palm OS		○	○	PDA	멀티미디어에 매우 취약
VRTX	○	○	○	통신장비,네크워장비, 자동차엔진제어시스템	신뢰성 높은 OS
pSOS	○	○	○	휴대폰	우리나라의 여러 업체가 채택
VxWorks	○	○	○	RTOS가 지원하는 모든 서비스	실시간성이 매우 뛰어남
QNX	○	X	○	자동차	매우 강력하고 안정적
WindowsCE .NET	○	X	○	가전제품, PDA,	국내에서 가장 활발히 개발
Windows XP Embeded	X	X	○	가전제품, 셋톱박스	네트워크, 서버에 적합
OS-9	○	X	○	RTOS가 지원하는 모든 서비스	세계 시장에서 높은 인지도
LynxOS	○	X	○	항공기 실시간 시스템	복잡하고 규모가 큰 애플리케이션에 적합
ThreadX	○	○	○	가전제품, 셋톱박스	제품구입시 소스코드 공개, 로열티 없음
Nucleus	○	○	X	가전제품, PDA, 셋톱박스	등록시 제한된 데모버전 사용가능
임베디드 리눅스	X	X	X	가전제품, 셋톱박스	소스코드 공개
트론(TRON)	○	○	X	전화, 가전제품	스펙과 커널만 오픈, 미들웨어 등은 유료
RT-Linux	○	X	△	기존리눅스에 리얼타임 기능 향상	OS의 사이즈가 큼
SuperTask	○	○	X	-	소스코드 오픈, 로열티 없음
uC/OS II	○	○	X	간단한 기능을 가진 가전기기	소스코드 공개
eCos	○	○	X	RTOS가 지원하는 모든 서비스	소스코드 공개

〈표 4〉 실시간 임베디드 OS

	Manufac-turer	Programming Languages	Target processors	Host operating system	API
VxWorks	Wind River Systems	C, C++	Intel:386, 486, 586; PowerPC; StrongARM; NEC	Win95/98, WinNT	POSIX
VRTX	Mentor Graphics Corporation	C, C++	Intel:386, 486, 586; PowerPC; ARM; Motorola	WinNT, Sun Solaris	자체 개발
pSOS	Integrated Systems	C, C++	Intel:386, 486, 586; PowerPC; ARM; Motorola, MIPS	WinNT, Sun Solaris, SunOS, HP-UX	POSIX
Nucleus Plus	Accelerated Technology	C, C++, Assembler	Intel:386,486,586; PowerPC; Motorola, NEC	Win95/98, WinNT	자체 개발
QNX-Neutrino	QNX	C, C++, Assembler, Java, ADA, etc	Intel:386, 486,586; MIPS, Socket7 Processor	Win95/98, WinNT, Solaris, Linux, Unix	POSIX
OS-9	Microware	C, C++, Assembler, Java	Intel:386, 486, 586; PowerPC; ARM; Hitachi, Motorola	Win95/98, WinNT	자체 개발
LynxOS	LynxRTS	C, Assembler	Intel:386, 486, 586; Motorola; Hitachi; NEC	Sun Solaris, SunOS, RS6000, LynxOS	POSIX
RTLinux	New Mexico Tech	C, Assembler	Intel:386, 486, 586;	Linux	POSIX
Micro C/OS-II	Micrium Inc.	C, C++	Intel:386, 486,5 86; ARM; Mitsubishi; Hitachi; Motorola	Win95/98, WinNT	POSIX
Windows CE	Microsoft	C, C++, Assembler, Java	Intel:386, 486, 586; PowerPC	Win95/98, WinNT	자체 개발

참고문헌

[1] 김선자, 김홍남, 김채규, "정보가전용 임베디드 운영체제 기술", 전자공학회지 제29권 10호 pp72-81 ('2001)

[2] 김대영, 도윤미, 김주홍, 조혜영, 성종우, "실시간 운영체제 연구개발 동향", 전자공학회지 제29권 9호 pp78-86 ('2002)

[3] 김재호, 김현준, "임베디드 운영체제", http://www.zdnet.co.kr/

[4] R. P. Dick, G. Laksshminarayana, A. Raghunathan and N. K. Jha, "Power Analysis of Embedded Operation System, In Proceeding of the Design Automation Conference, June 2000

[5] Velos Home-page, http://www.velos.co.kr/

[6] Symbian Home-page, http://www.symbian.com/

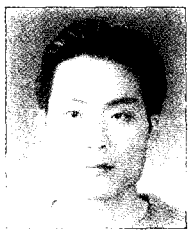
[7] Palm OS Home-page, http://www. palmsource.com/

[8] Nucleus Home-page, http://www.mentor.com/
[9] pSOS & VxWorks Home-page, http://www.windriver.com/
[10] QNX Home-page, http://www. qnx.com/
[11] OS-9 Home-page, http://www.microware.com/
[12] LynxOS Home-page, http://www. lynuxworks. com/
[13] ThreadX Home-page, http://www. expresslogic.com/
[14] Embedded Linux Home-page, http://www.embedded-linux.org/
[15] TRON, ITRON, uITRON Home-page, http://www.tron.org/
[16] SuperTask Home-page, http://www.ussw.com/
[17] MicroC/OS Home-page, http://www. ucos-ii.com/
[18] Redhat Home-page, http://www.redhat. com/
[19] eCos Home-page, http://ecos.sourceware. org/
[20] L4 Reference Web-site, http://l4hq.org/
[21] Microsoft Home-page, http://www. microsoft.com/
[22] Mizi Home-page, http://www.mizi.com
[23] QplusHome-page, http://embenix.com/ qplus/
[24] OSEK Home-page, http://www. osek-vdx.org/

저자약력

노 학 종

1999년 서남대학교 전산통계학과졸업 (학사)
2000년~2003년 ㈜협우인포테크 연구원
2003년 - 현재 건국대학교 컴퓨터정보통신학과(석사과정)
관심분야 : RTOS, Embedded System



김 강 진

2002년 단국대학교 전자계산학과 졸업 (학사)
2003년 - 현재 건국대학교 컴퓨터정보통신학과(석사과정)
관심분야 : RTOS, Embedded System

김 문 회

1979년 서울대학교 전기공학과 학사
1981년 서울대학교 석사
1985년 University of South Florida, Computer Science and Engineering, MSCS
1991년 University of California, Berkeley, Computer Science, Ph.D
1991년 - 현재 건국대학교 컴퓨터공학부 교수
관심분야 : 객체지향 분산실시간시스템, 임베디드시스템, 결함허용시스템, 소프트웨어공학