특 집

임베디드 시스템 연구 동향

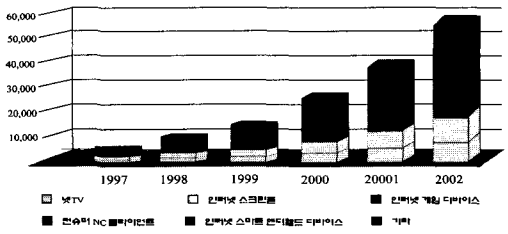
이 정 배*, 이 두 원**

● 목 차 ●

1. 임베디드 시스템의 개념 및 연구 동향

지난 10여년 동안 PC시장은 대형 PC업체들은 단순 조립업체로 전락하고 마이크로소프트, 오라클 등 소프트업체가 시장 및 기술 발전을 좌지우지해 왔다. PC시장의 예에서 보아왔던 것처럼 앞으로의 정보가전 산업은 단순한 시스템조립업체에 의한 상품의 대량생산이 제품 경쟁력의 제고를 위한 방법이 아니라 얼마나 우수하고 다양한 서비스를 제공하는 소프트웨어를 개발하는 데 있다. 이런 시대를 준비하기 위해서는 소프트웨어 기술, 디지털TV나 PDA, 게임기 같은 임베디드 시스템용 실시간운영체제(Real Time Operating System)기술을 확보하고 있어야 한다. 전화국과 일반가정, 사무실을 유선이 아닌 무선으로 연결하여 음성, ISDN, Fax, 데이터 서비스 등을 제공하는 서비스 시스템 개발을 위하여 관계되는 장비간의 인터페이스를 정의하고, 구현하는 실질적인 적용분야다.

요소기술들은 일부 개발이 완료되어 있기 때문에 기 개발 모듈에 대해서는 성능의 보완과 향상을, 미개발 모듈의 추가 개발을 통해 상업화를 목표로 있으며, 향후 발전 시켜 고가용성(High Availability)와 Low-Cost를 만족하는 임베디드 모바일 Linux기반형 시스템으로 발전되고 있는 추세이다. 임베디드 시장의 인터넷응용분야 전망을 IDC자료를 통하여 보면 그림 1에서 보는 바와 같다.



(그림 1) 임베디드 시장의 인터넷응용분야 전망

■ 주요 수요처 : 크게 구분지어 3분야로 나뉘어진다. 텔레컴분야(사설통신-LAN스위치, 라우터, 사설전화교환기, ADSL, 케이블모뎀 ..., 공중통신-대용량 교환기), 정보가전 분야(디지털TV, 셋탑박스, 디지털가전, 휴대폰, PDA, HPC ...), 산업전자 분야(자동차, 산업용로봇, 공정자동화, 의료, 국방, 운송, 공정제어)

■ 사용될 실시간 임베디드 Linux 운영체제 시스템의 특징은 다음과 같다.

* 부산외국어대학교 컴퓨터공학과
** (주)두올정보기술 대표이사

- PowerPC MMU 지원
- Fixed Address Database Support 0xC1000000
- POSIX, UNIX Compatibility
- Broad CPCI Platform 지원
- Hard real time 기능 만족

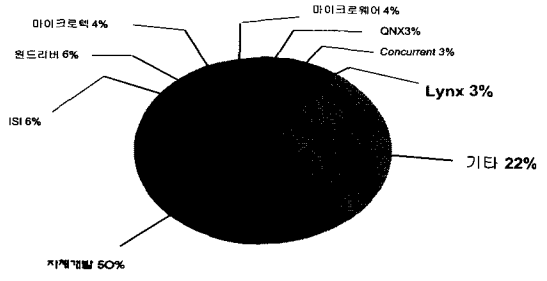
1.1 세계적 기술현황 및 전망

운영체제 개발이 핵심 기술로 인식하고 자체적인 OS 개발에 주력, 운영체제는 Real-Time OS인 VxWorks가 현재 많이 사용되고 있으나, 저가형 임베디드 Linux의 등장으로 시장의 변혁이 예상된다.

☞ 해외시장규모 : 세계적으로 자국 또는 지역별로 자체 임베디드 OS를 확보해나가는 추세이다. ePOC 중심의 유럽, xTRON중심의 일본이 그 대표적인 사례로, 1세대 컴퓨터분야에서 미국에게 내줬던 자리를 빼앗기지 않기 위해 심혈을 기울이고 있는 것이다. 현재 세계적으로 상용 임베디드 OS사용이 40%정도이고, 점점 더 상용 RTOS를 사용하는 것이 일반화되고 있어 국산 임베디드 개발과 본 제품의 개발은 해외로의 로열티 유출을 차단하는 방패역할을 할 것이다. 현재 해외시장의 규모는 대략 매년 50억불정도이며, 매년 30%씩 성장하고 있는 추세이다.

1.2 국내 기술현황 및 전망

소프트웨어 기술은 응용 프로그램 개발에 주력하고 있으나 핵심 소프트웨어인 운영체제 분야 연구가 전혀 되고 있지 않다.



(그림 2) 국내시장현황

☞ 국내시장규모 : 연간 500억시장이며, 이중 50%에 해당되는 부분을 대부분 제조 및 생산을 담당하는 업체에서 직접 개발하여 사용하고 있는 상황이며, 이의 개발을 위하여 6개월 이상을 소요하기에 시간적인 손실을 가져오고 있는 현황이다.

1.2.1 임베디드 시장 개요

1년동안 전세계에서 생산되는 컴퓨터 칩의 단지 10 ~ 20% 만이 데스크톱이나 랩톱 컴퓨터를 만드는데 사용되고, 나머지의 20억개에 달하는 컴퓨터 칩이 임베디드 시스템 속으로 들어간다는 사실은 임베디드 시스템의 발전 가능성을 보여준다. 수많은 임베디드 시스템에 대한 사용자들의 요구가 복잡해지고 기대 수준이 높아지면서 이러한 요구를 충족시킬 수 있는 고성능의 운영체제가 필요하게 되었고, 그 대안으로 주목 받고 있는 것이 지금까지 이야기한 임베디드 실시간 운영체제이다.

<표 1> 정보가전 응용 제품 분야 시장 예측
(단위:USD, 0.1 million dallar)

	Desktop Operating System	Smart Phone	SetTop Box	Medium-Large Scale Computer	Net Top
2000	290	410	468	3,905	47
2001	639	629	666	9,764	135
2002	1,055	967	1,214	-	384
2003	1,741	1,490	1,881	-	1,096
2004	2,872	2,298	2,844	-	3,126
Source	KSA	MIC	Cahner Group	MIC	IDC, LG-EDS

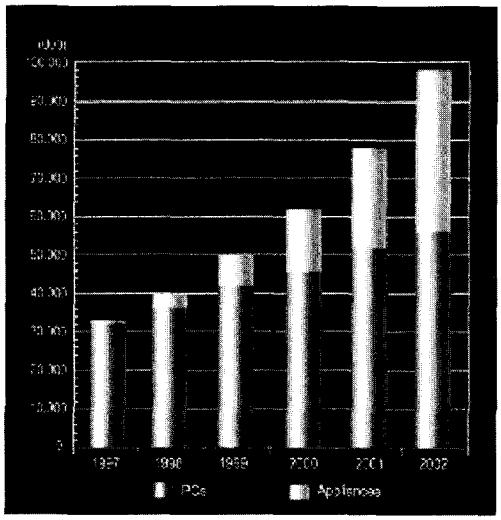
국내 임베디드 산업은 지난 99년부터 본격적으로 성장하기 시작해 올해에는 참여 회사들의 증가와 수요 증대 등에 힘입어 큰 폭의 성장세를 보이고 있다.

정보통신부 등의 자료에 따르면 임베디드 시장 규모(하드웨어 매출 포함)은 데스크톱 운영체제 290억원, 스마트폰 410억원, 셋톱박스 468억원, 중대형컴퓨터 3천905억원, Net Top 47억원 등 총 5천120억원에 달하는 것으로 추산됐다. 이중 임베디드

시스템 관련은 스마트폰과 셋톱박스 등 878억원에 달하는 것으로 추산됐다.

그러나 하드웨어를 제외한 리눅스 관련 매출액만 집계할 때 총 시장규모는 지난해 100억원 가량이었던 것이 올해에는 1천억원을 넘어설 것으로 전망된다.

이 임베디드 소프트웨어 시장은 크게 보아 판매 목적이 아니라 특수한 목적으로 만들어지는 소위 In-house 임베디드 소프트웨어와 상용 임베디드 소프트웨어로 나눌 수 있고 아직까지 상용 제품의 비율이 그리 크지 않다. 그러나 보다 나은 서비스를 보다 빠르게 제공하여야 하는 치열한 경쟁의 시대에 점차 상용 제품의 사용이 늘어나고 있는 실정이다.



(그림 3) 임베디드 시스템 시장 예측

임베디드 소프트웨어 시장의 전체 규모는 조사기관에 따라 다르나, Vessels, Arnold & Henderson사의 조사에 의하면 96년 한해 동안 약 25억불(약 3조 7,500억원)에 이르는 것으로 알려지고 있다. 이중 상용 임베디드 소프트웨어를 약 8%정도 규모로 2억불 정도로 보고 있다. 그러나 2001년에는 전체적인 규모는 약 52억불, 상용제품은 11억불로 년간 각각 21%, 100%이상으로 매우 급성장할 것으로 내다보고 있다.

1.3 실시간/임베디드 시스템의 개요

실시간 운영체제가 왜 나오게 되었는지는 현재 어디에서 쓰이고 있는지 알면 될 것이다. 실시간 운영체제가 쓰이고 있는 경우는 크게 2가지로서 임베디드 시스템과 다중처리기이다.

다중처리기는 여러 개(multi)의 처리기들이 한 시스템에 들어가서 동시에 작업을 수행하는 것이다.(multiprocessing과는 다른 개념) 이때 여러 개의 처리기들을 관리하기 위해서 기존의 운영체제가 아닌 다중처리기용 운영체제가 필요하다.

임베디드 시스템이라는 것은 일반적인 컴퓨터 시스템과는 달리 특정한 작업만을 하도록 설계되며 초기의 임베디드 시스템은 비교적 단순해서 운영체제가 필요 없이 인간이 순차적인 프로그램을 작성해서 수행하도록 했고 중간에 인터럽트가 발생되는 경우에만 그 순차적인 프로그램에서 잠시 벗어나는 정도였다. 그러나 최근 들어 멀티미디어 정보를 처리해야 하는 임베디드 시스템이 늘어나면서 그 시스템이 해야 할 일들도 많아지고 복잡해 졌기 때문에 순차적인 프로그램 작성이 매우 어렵게 되었다. 따라서 임베디드 시스템에서 운영체제의 개념이 필요하게 되었으며 임베디드 시스템의 특성상 실시간이라는 요소를 만족해야 했다. 따라서 실시간 운영체제가 임베디드 시스템에 도입된 것이다.

참고적으로 실시간 시스템에 대해 설명하자면 정해진 시간 제약을 해결할 수 있는 시스템을 말한다. 이 말은 주어진 시간 내에 어떠한 일을 반드시 처리해야 한다는 말이다. 즉 단지 빨리 처리해야 하는 것이 아니고 정해진 시간을 넘겨서는 안된다는 것이다. 실시간에서 가장 대표적인 분야로는 군사용을 들 수 있다.

우리가 쓰는 컴퓨터에서는 에디터가 어느 때는 좀 빨리 어느 때는 좀 늦게 실행된다고 해서 크게 문제가 될 건 없다. 그러나 임베디드 시스템에서 주어진 입력중에는 시간 내에 처리하지 못하면 문제가 될 수 있기 때문에 실시간 요소를(보통 soft

real-time)만족시켜야 한다.

1.4 일반 시스템과의 차이점

실시간 운영체제라고 하면 일반적으로 말하는 운영체제와는 막연히 무언가가 다르다고 생각한다. 그러나 실시간 운영체제라고 해서 특별히 다른 것은 없다. 일반적인 운영체제와 마찬가지로 여러 가지 task를 동시에(가상적인 의미에서) 수행하는 것이다. 대부분의 실시간 운영체제 매뉴얼을 살펴보면 다음과 같이 task scheduling, task communication, task synchronization, memory management, interrupt service, I/O driver, timer 등의 기능을 하며 이러한 것은 일반적인 운영체제에서도 필요한 기능이다.

말 그대로 실시간이기 때문에 실시간을 처리하는 운영체제라고 생각할 수 있겠다. 그렇다면 무언가 다른 것이 있을 것이며 또 어떻게 처리해야 실시간을 만족할 것인가?

여러 가지 차이점이 있겠으나 대체로 효율성, 속도(좀더 엄밀히 말하면 시간 제약), 공평성(fairness) 등을 들 수 있겠다.

일반적인 운영체제에서는 자원(예를 들어, 메모리, 하드 디스크 등등)을 얼마나 효율적으로 낭비없이 쓸 것인가에 초점이 맞춰져 있다. 그러나 실시간 운영체제에서는 효율적인 것도 중요하지만 그 속도에 좀 더 신경을 쓴다.

예를 하나 들어보자. 보통 C 프로그램을 작성할 때에 메모리를 할당하기 위해서는 malloc이라는 함수를 쓰게 된다. 이 함수는 malloc을 할 수 있는 영역에서 메모리를 찾게 된다. 만약 메모리가 당장 쓸 수 없는 경우에는 기다리는 일도 발생된다. 그러나 실시간 운영체제에서 메모리를 할당하는 경우에는 malloc을 쓰지 않고 실시간 운영체제에서 제공하는 메모리 할당 함수를 쓴다. 이 함수가 malloc과 차이점이 있다면 정해진 구역(이미 처음에 메모리를 할당하기 위한 메모리 풀을 선언하게 된다)에서 메모리를 할당하려고 시도하고 만약에 당장 할당할 메모리가 없다면 무작정 기다리는 것이 아니라 현재 할당할 수 없음을 알리게 된다. 이는 무작정 메모리 할당을 기다리다가 정해진 시간 제약을 만족하지 못해서 실시간으로 실행하지 못하게 됨을 막기 위한 것이다.

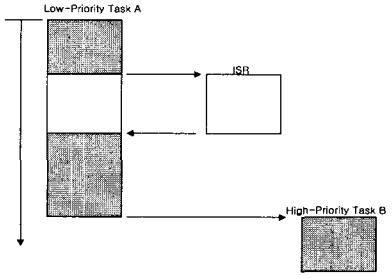
이 두 가지 구현 방법을 살펴보면 일반 운영체제에서는 메모리라는 자원을 좀 더 효율적으로 쓰려는 의도가 있는 것이고 실시간 운영체제에서는 약간의 메모리 낭비를 감수하고서라도 시간 제약을 맞추려는 의도가 있는 것이다.

또한 UNIX와 같이 여러 명의 사용자가 쓰는 경우에는 각 사용자들이 실행하는 프로그램이 task로서 수행이 되고 대개의 경우에는 우선순위에 많은 차이를 두지 않는다. 이것은 그 task들 사이에 공평성을 유지하려 하기 때문이다. 그러나 실시간 운영체제에서 task는 대개 우선순위가 차이가 있도록 하며 이때 task사이의 공평성은 고려하지 않는 것이다.

이상에서 본 것처럼 메모리를 할당하거나 task를 scheduling하는 작업은 두가지 운영체제에서 모두 필요한 작업이므로 작업 자체는 공통적인 것으로 볼 수 있다.

한편, 구현을 함에 있어서는 원하는 목표가 다르기 때문에 차이가 나타난다.

결론적으로, 원래 실시간 시스템과 일반 시스템의 차이가 해야 할 기능은 같지만 주어진 시간 안에 처리할 수 있는가 아닌가라고 할 수 있듯이 운영체제에서도 역시 결국 하는 작업은 같지만 어떻게 처리하는가가 차이라고 할 수 있을 것이다.



(그림 4) 실시간 운영체제 커널 구조(Nonpreemptive kernel)

1.5 Embedded Linux 운영체제의 개념

무선 인터넷 정보가전 시스템용 입출력 디바이스를 개발하여 포팅하게 될 운영체제는 높은 신뢰성을 가지며, 임베디드 시스템을 위한 운영체제이다. 이 임베디드 Linux는 X86 및 UNIX 호환, POSIX 준수의 멀티 프로세스와 멀티 스레드를 제공하는 임베디드 운영체제이다.

임베디드 Linux는 스케일 측정, 풍부한 재입력, 공간확보, ROM화 등의 기능이 가능하며, 커널은 Hard 실시간 응답과 고성능 시스템 로딩 및 다양한 서비스를 제공하기 위해서 설계되었다. 임베디드 애플리케이션을 구성할 수 있는 신뢰할 만한 기본 요소를 아래와 같이 제공을 한다.

1.5.1 강 실시간(Hard Real-Time) 기능

임베디드 Linux는 시스템 설계자들에게 신뢰성, 개방성을 제공하기 위해서 의사결정, 짧은 Blocking time, 우선순위의 변동을 방지하기 위한 설계 특징을 통하여 실시간 응답성을 만족시킨다. 임베디드 Linux 스케줄링에서 임베디드 Linux 스케줄러는 실행에 대한 사용자와 시스템의 쓰레드(thread)를 통하여 Hard Real-Time 우선순위를 구성하고 있으며, 공평한 시간 공유를 위한 우선권을 선택하지 않고 실시간 설계자가 구성한 최우선 순위대로 실행 스케줄링을 채택한다.

임베디드 Linux 우선순위 탐색 커널 쓰레드를 통하여 다음과 같은 상호작용을 관리한다. 커널 쓰레드 대부분의 운영체제의 입출력 구조는 ISRs (Interrupt Service Routines)의 중심에서 장치코드를 가진 실시간 이벤트를 다루고 있으며 이러한 ISR이 실행되는 동안 낮은 우선순위의 하드웨어뿐만 아니라 모든 소프트웨어 우선권을 선점한다.

1.5.2 신뢰성 있는 프로세스 환경

임베디드 Linux는 복잡한 멀티 스레드와 멀티 프로세스 애플리케이션을 위한 고수준의 신뢰성을 제공한다. POSIX 프로세스 모델은 주소 공간의 소유권을 가진 싱글 및 멀티 스레드 프로그램의 실행을 지원하며, 프로그램은 서로 보호, 분리되고 다른 프로그램과 상호관계를 가지며 메모리 구조를 고려한다. 그리고 MMU(Memory Management Units) 프로세스는 물리적인 주소와 논리적인 주소를 상호 맵핑하는 고수준의 신뢰성을 가진 주소 체계를 지원한다.

임베디드의 개념에 리눅스의 장점이 접목되면서 임베디드 리눅스의 성장 가능성은 높이 평가되고 있다. 예를 든다면, 오늘날 많은 기기들이 인터넷에 연결되어 있다. 어떤 냉장고들은 인터넷에 연결되어 온도, 현재 저장량 등과 같은 상태를 웹브라우저를 통해 모니터링 또는 조절이 가능하다. 이러한 오늘날의 인터넷 세태는 Telerobots이라는 것을 만들어 냈는데 이는 네트워크를 통해 원격 콘트롤이 가능한 Robots을 지칭한다. 냉장고 뿐만이 아니라 세탁기 등 원격으로 집안 기기들을 네트워크를 연결해 콘트롤 할 수도 있다. 위의 내용들을 토대로 살펴보면 임베디드 리눅스의 연구개발은 다음과 같은 분야에서 이루어지리라 전망된다.

- 첫째, 저전력 시스템 개발

소형 휴대 정보 단말기의 개발에 있어 효율적인 전력 관리를 통한 배터리 사용 시간의 연장은 중요한 경쟁력 향상의 요소이다. 리눅스 커널에 이용되고 있는 진보된 전력 관리 기술을 연구 개발하고 이를 소형 시스템에 구현하는 일이 필요할 것이다.

- 둘째, 효율적인 메모리 관리 기술

소형 시스템은 제한된 메모리 지원을 갖고 있으며, 이런 제한된 메모리 지원 위에 다양한 기능을 갖는 소프트웨어를 지원하여 기능성을 향상시키는 기술이 필요할 것이며, 리눅스에 이용된 메모리 및 프로세스 관리 기술을 연구 개발하여 이를 소형 시스템에 구현하여야 할 것이다. 특히 실행 후 잔류 메모리를 감소시키는 연구가 필요할 것이다.

- 셋째, 유저 인터페이스 연구 및 개발

유저 인터페이스를 필요로 하지 않는 임베디드

시스템도 있지만, PDA를 포함한 많은 임베디드 시스템들은 간단한 텍스트 기반에서부터 훨씬 복잡한 방식까지 유저 인터페이스를 필요로 한다. 데스크 탑에 쓰이는 유저 인터페이스는 현재까지 키보드, 마우스, 그래픽 등이 사용되어 왔으나 PDA의 경우는 훨씬 작은 플랫폼이란 제한성으로 키패드, 필기체 인식, 음성인식 등이 연구 개발되어 왔다. 최근에는 이 외에도 한 손만으로 사용할 수 있는 방향성이나 무게성에 기반을 둔 유저 인터페이스에 대한 개발도 이루어지고 있다. 사용자의 편의성과 전달의 정확성을 중시하는 새로운 개념의 유저 인터페이스에 대한 개발은 리눅스 뿐만 아니라 대부분의 임베디드 시스템에 연구되어져야 할 부분일 것이다.

● 넷째, 안정적 시스템 소프트웨어 기술 개발

데스크탑 컴퓨터에 쓰이는 리눅스는 다른 운영체제에 비해 훨씬 향상된 안정성을 갖고 있으며, 이는 특히 통신 구성요소에 있어서 더욱 그러하다. 이런 리눅스의 안정성을 소형 시스템에도 그대로 유지할 수 있는 기술을 개발할 것으로 전망된다.

다섯째, 실시간성

휴대 정보 단말기를 멀티미디어 플레이어로도 사용하기 위해서 가장 필요한 요소 중의 하나는 운영체제의 실시간성이다. 완화된 실시간성은 기존의 리눅스에 이미 POSIX 연실시간 스케줄링으로 구현되어져 있으며 엄격한 실시간성을 커널에 구현하는 연구는 최근에 이루어지고 있다. 리눅스에 실시간 동작을 원하는 시스템을 위한 연구 기관으로는 미국의 뉴멕시코 대학과 캔사스 대학을 꼽을 수 있다. 실시간 리눅스는 멀티미디어 단말 뿐만 아니라 네트워크 라우터 장비에도 적용되고 있다. 따라서 실시간성에 대한 연구가 요구되어진다.

● 여섯번째, 부분 시스템으로서의 임베디드 시스템

PDA와 같은 임베디드 시스템 및 이동 단말은 작고 가벼워야 하므로 컴퓨팅 파워 및 메모리 사이즈에 있어서 제한적일 수 밖에 없으며 하드 디스크 같은 대용량 저장 장치를 장착하기도 어렵다. 물론 휴대 시에는, 단말 자체가 독립적인 완전한 시스템으로 작동하여야 하기 때문에 이 부분에 대한 연구개발이 요구되어진다.

이렇듯 독립적인 시스템만으로 PDA가 작동하는 데에서 더 나아가 다른 PC나 워크스테이션에 있는 대용량 저장 용량 장치나 다른 자원을 이용할 수 있다면 더 나은 서비스를 제공할 수도 있을 것이다. 예를 들어 어떤 멀티미디어 단말이 이동 중에는 플래쉬 메모리 사이즈가 작으므로 불과 수 곡의 MP3 파일만을 연주할 수 있으나, 집에서는 PC에 있는 MP3 파일 디렉토리를 NFS 마운트하여 자신의 파일 시스템으로 사용할 수 있게 만든다면 수백 개의 MP3 파일까지도 연주할 수 있다.

2. 국내외 임베디드 S/W의 산업현황

2.1 임베디드 운영체제(RTOS)

실시간 운영체제에 대해 이야기 하기 전에 실시간에 대한 정의부터 하고 넘어가기로 한다. 실시간 시스템은 기존의 컴퓨터 시스템과 달리 시스템 동작의 정확성이 논리적 정확성 뿐만 아니라 시간적 정확성에서도 좌우되는 시스템을 말한다. 이러한 실시간 시스템의 전형적인 예로서 제어시스템을 들 수 있다. 제어시스템은 감지장치(sensor)로부터 입력을 받아들여 이를 정해진 시간 내에 처리하여 작동장치(actuator)로 출력하며 극히 작은 시간적 오차를 허용한다. 실시간 시스템의 응용분야로는 핵발전소의 제어, 공정제어, 병원의 감시장치, 항공기 제어, 무기 체계, 우주선의 운항 및 유도 등의 분야를 들 수 있다. 실시간 시스템에 존재하는 시간 제약 조건은 종료시한(deadline)으로 주어진다. 종료시한은 그것의 엄격성에 있어 세 가지로 분류될 수 있다. 첫째로, 경성(hard) 종료시한은 시스템이 주어진 종료시한을 만족시키지 못한 경우에 막대한 재산적 손실이나 인명의 피해를 주는 경우를 말한

다. 둘째로, 연성(soft) 종료시한은 시간제약 조건을 만족시키지 못하더라도 경성의 경우처럼 치명적이지 않고 종료시한을 넘겨 수행을 마쳐도 계산의 결과가 의미가 있는 경우를 말한다. 연성 종료시한을 갖는 대표적인 시스템은 온라인 트랜잭션 시스템을 들 수 있다. 마지막으로 준경성(firm) 종료시한은 경성과 연성의 중간 형태로 종료시한을 넘겨 수행을 마치는 것은 무의미한 경우를 의미하며 손실이 치명적이지 않은 경우를 말한다.

실시간 시스템이 종료시한을 만족시키기 위해서는 고속의 계산을 요구하게 되지만, 고속의 계산이 실시간 시스템의 요구조건을 만족시키는 것은 아니다. 일반적으로 고속의 계산은 시스템의 평균 응답시간을 최소화하지만, 실시간 시스템에서 요구되는 예측가능성을 보장하지는 않는다. 예측가능성이란 시스템의 명세에 정의된 고장이나 작업 부하 조건에서 태스크의 종료시한 만족을 보장하는 것을 의미한다.

앞의 내용을 간단히 언급하면 실시간 시스템은 "시스템의 수행 결과가 기능적으로 정확해야 할 뿐만이 아니라, 결과가 도출되는 시간 역시 주어진 제약 조건을 만족시켜야 하는 시스템"이라고 간략하게 정의할 수 있다. 실시간 시스템이 시간적 제약 조건을 만족시키지 못한 경우에는 작게는 시스템 오동작, 크게는 인명 손상과 같은 재앙을 유발하게 된다. 이와 같은 요구 조건 때문에, 시스템의 수행은 예측 가능해야 한다. 실시간 운영체계는 이러한 실시간 시스템의 개발, 운영에 사용되는 운영체계이다.

실시간 운영체계는 그 특성상 MS-DOS나 Windows95 등과 같은 범용 운영체계로 쓰이지는 않고, 내장 제어 시스템과 같은 특수 목적으로 사용되는 경우가 대부분이다. 실시간 운영체계가 가져야 할 몇 가지 특징을 생각하면 다음과 같다.

첫째, 다중 쓰레드를 지원하고, 선점가능해야 한다. 이는 실시간 시스템이 예측 가능해야 하기 때문인데, Windows 3.11과 같은 상호 동작 가능한 운영체계 하에서는 한 태스크가 무한히 CPU를 점유하는 것이 가능하여, 시스템의 성능을 예측하기 어렵게 만든다 .

둘째, 쓰레드간의 우선 순위를 보장하여야 한다. 이는 쓰레드의 데드라인을 만족시킬 수 있는 방법을 마련하기 위해 필요하다.

셋째, 쓰레드간의 동기화를 지원해야 한다. 쓰레드들 사이에 자원의 공유와 커뮤니케이션이 필요하므로, 이들간의 동기를 맞추어 줄 수 있는 방법이 마련되어야 한다.

넷째, 운영체계의 행동이 명확해야 한다. 여기서 행동이라는 것은 시간적인 측면을 의미하는 것인데, 여기에는 인터럽트 지연 시간, 시스템 콜의 처리 시간, 그리고 운영체계와 드라이버가 인터럽트를 마스크하는 시간 등이 포함된다. 따라서 실시간 운영체계 개발자들은 다음과 같은 사항들을 명시해야 한다. 즉, 시스템의 인터럽트 레벨, 디바이스 드라이버의 IRQ 레벨, 그리고 이의 최대 처리 시간 등이 포함되어야 한다.

실시간 시스템이 적용되는 분야를 예로 들면 자동 제어 시스템 등을 생각할 수 있다. 어떤 공장에서 로봇이 컨베이어 벨트에 의해 운반되는 물건을 집어서 이동시킨다고 하자. 로봇이 시기 적절하게 물건을 집어 내지 못한다면, 기능적으로는 올바르게 수행 했으나(물건을 집을 위치로 이동함) 시간 제약을 지키지 못하게 되어 결과적으로 정상적으로 수행을 행하였다고 볼 수 없게 되는 것이다(물건을 집어 오지 못함).

최근 운영체제는 컴퓨터의 성능을 향상시키는데 많은 도움이 되고 있다. 그러나, 실시간 시스템에서의 성능은 단지 평균 실행시간에 의해서만 측정되지는 않는다.

실시간 시스템의 제약조건은 시간 제약 조건 외에도 자원, 우선순위 또는 선행관계, 태스크간 통신

및 동기화 제약 조건이 있을 수 있다. 실시간 시스템을 구성하기 위해서 필요한 하드웨어와 소프트웨어는 다음과 같은 특성을 갖는다. 우선 하드웨어는 고신뢰성을 제공하기 위해 결함허용성을 지원해야 하고 확장성과 유연성, 코드의 ROM화, 그리고 기존의 일반 부품을 사용하여야 한다. 소프트웨어 부분은 실시간 운영체제 또는 실시간 실행체제 등이 요구되며 이들은 태스크의 스케쥴링, 태스크간 통신 및 동기화, 인터럽트 처리, 실시간 시계관리등의 기능을 수행한다. 실시간 시스템의 주요 연구분야로는 명세와 검증, 설계방법론, 프로그래밍 언어, 스케쥴링 알고리즘, 자원관리를 위한 운영체제의 기능, 실시간 통신 구조, 결함 허용성등을 들 수 있다.

흔히 임베디드 소프트웨어 시장으로 불리는 이 시장의 주요 제품은 크게 보아 실시간 운영체제, Tool(Compiler, Debugger, Board Support Package Development Tool) 로 나눌 수는 있으나, 대부분의 실시간 운영체제 공급자들은 자신들의 실시간 운영체제 제품에 맞는 도구들을 함께 공급하고 있으므로 두 부분을 현실적으로 나누어 생각하기 다소 어려운 점이 있다. 따라서 이 글에서도 역시 매출액의 크기 등 대부분의 경우에 실시간 운영체제와 Tool을 모두 포함하여 대표적으로 실시간 운영체제라 칭한다.

2.2 임베디드 소프트웨어 시장

임베디드 소프트웨어 시장은 크게 보아 판매목적이 아니라 특수한 목적으로 만들어지는 소위 In-house 임베디드 소프트웨어와 상용 임베디드 소프트웨어로 나눌 수 있고 아직까지 상용 제품의 비율이 그리 크지 않다. 그러나 보다 나은 서비스를 보다 빠르게 제공하여야 하는 치열한 경쟁의 시대에 점차 상용 제품의 사용이 늘어나고 있는 실정이다.

1997년도 한해 동안의 상용 실시간 운영체제 시장 규모는 대략 3억 6,400만불정도 규모로 알려져 있으며 이중 VxWorks(Windriver사), pSOS(ISI사), VRTX (Mento Grapics), OS-9(Microware Systems사), QNX (QSSL사) 그리고 Microsoft사의 WinCE, Sun사의 임베디드 JAVA등이 주요한 제품으로 망라되고 있다.

각 제품마다 특장점들이 있기 때문에 이들을 일률적으로 재단하여 비교 평가하기는 어려우나, 대략 크게 Cross 개발 환경을 제공하는 것과 Native 개발환경을 제공하는 두 가지로 구분하여 바라보는 것이 일반적이다.

Cross개발환경은 UNIX, Win95/NT에 개발 환경(이처럼 개발환경이 구축된 장비를 Host라 부른다)을 구축하고 이곳에서 특정 Target보드에 맞는 실시간 운영체제 및 응용 프로그램 이미지를 Cross컴파일, 즉 이기종간의 컴파일을 수행하는 방식을 말한다. Host의 플랫폼은 SUN, HP UNIX가 강세였으나 최근에는 NT 및 Win95도 많이 채택되고 있는 편이다. 이와는 반대로 Native개발환경은 흔히 Self-Host라고도 불리는 데, 이는 일반적인 데스크탑환경과 같이 Host와 Target보드의 운영체제가 같은 경우를 말한다. 이러한 Native 개발환경은 X86 CPU에서는 상당히 활성화 되어 있다. Cross업체는 개발툴에, 그리고 Native업체는 개발툴보다는 Production License (Runtime Fee)에 비즈니스의 초점을 두는 경향이 있다.

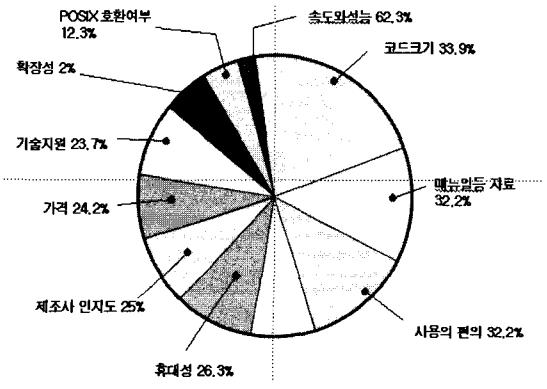
전통적인 실시간 운영체제 공급자를 중심으로 살펴보면, Cross 환경 쪽에서는 VxWorks, LynxOS, pSOS, VRTX 등이 세계적으로 가장 널리 알려져 있고 실제로 국내에서도 이들 벤더들은 매우 활발하게 움직이고 있다. 국내 시장현황과 국외 시장현황은 다소 다른 점이 있지만 특수한 기능들을 강화하여 시장을 선점하는 운영체제의 등장이 주목되고 있다. 특히 최근의 LynxOS는 Telecommunication시장을 주요 고객으로 하는 HA(고가용성, High Availability) 솔루션 등을 내 놓아 사용자들이 급증하고 있는 추세이다.

Native환경 쪽에서는 LynxOS, QNX, OS-9등이 세계적으로 널리 알려져 있으나 아직 이 분야의 국내 시장현황은 외국에 비해 다소 저조한 편이다.

상기에서 언급되지 않은 수많은 상용 실시간 운영체제가 시장에서 활동을 하고 있으나 너무 많아서 일일이 언급하기가 어려우며 대부분 이들은 Source Code를 제공하여 오히려 In-house 실시간 운영체제를 더욱 많이 만들어 내는 데 기여하고 있다.

실시간 운영체제는 표준 하드웨어환경이 정해져 있지 않다는 점이 윈도우나 유닉스 등 범용 운영체제와 커다란 차이이며, 이는 곧 응용프로그램뿐 아니라 운영체제 자체에 대해서도 사용자가 환경을 설정해야 하고 때에 따라서는 포팅 노력이 요구된다는 뜻이다. 그리고 시스템의 하드웨어 자원, 특히 메모리 사용에 대해서 많은 제약이 따른다. 일반 운영체제에서 작동하는 응용프로그램이 몇 백 KB 메모리를 더 사용한다고 해서 무슨 큰 문제가 되는 것은 아니다. 하지만 임베디드 시스템에서는 메모리를 더 사용하면 그만큼 제품 단가가 올라가고, 특히 모빌 환경에서는 제품의 크기와 전지의 사용시간에도 직접적인 영향을 주게 된다. 임베디드 분야의 개발자 실력이 0에서 100점까지 확연하게 드러나는 이유가 바로 여기에 있다.

현재 시장에선 임베디드와 실시간 운영체제에 대한 확실한 구분 없이 사용되고 있다. 정확히 말



(그림 5) 상용 RTOS와 리얼타임커널 선택 기준

하면 임베디드 운영체제는 실시간 운영체제를 포괄하는 폭넓은 분야이다.

유닉스나 윈도우와 같은 일반 운영체제는 몇 가지 표준 때문에 정해진 하드웨서 돌아간다. 리눅스를 제외하고는 사용자가 운영체제를 건드릴 수가 없기 때문에 현재로서는 리눅스는 임베디드 운영체제일 수 있어도, 실시간 운영체제는 아니다. 흔히 실시간 운영체제라면 속도가 빠른 운영체제라고 생각하지만 실행 속도와는 별 관련이 없다. 다시 말해 운영체제의 각종 동작이 어떤 정해진시간 안에 이뤄진다면, 그 시간이 아무리 길어도 실시간 운영체제라고 말할 수 있다. 실시간 운영체제는 공정제어나 의료정보시스템 등 어떤 상황에서도 정확한 처리를 해줘야 하는 데 적합하다.

위의 그래프는 상용 실시간 운영체제와 리얼타임커널을 선택하는 가장 중요한 기준으로 1997년도 임베디드 시스템 Programming 7월호에 게재된 내용이며, 대부분의 경우에 가장 타당하게 적용하는 내용을 가리키고 있다.

2.3 전통적인 실시간 운영체제 시장의 주요 이슈와 경향

전통적인 실시간 운영체제 시장은 그 자체의 극심한 경쟁과 놀라운 기술 혁신과 달리, 매우 보수적인 경향이 있다. 다시 말해 실시간 운영체제는 기본적으로 데스크 탑 세계에서 증명된 기술들만을 가져가는 경향이 있다.

전통적인 실시간 운영체제는 주소 경성 실시간(내장형, Hard 실시간)시스템에서 사용되어지기 때문에 안정성은 필수적인 요소로 된다. 이것이 다소 상용 실시간 운영체제를 데스크탑 영역에서 이미 증명된 기술만을 사용하는, 보수적인 경향을 띄게 만들고 있다. 그래서 최근 실제로 화두로 던져지고 있는 것들 중 상당수는 이미 데스크탑영역에서 이루어졌던 것들이다.

POSIX지원, GUI, Internet Solution, JAVA등이 바

로 그것이다. 현재 주요한 상용 실시간 운영체제는 POSIX표준에 대해 이미 지원하고 있거나 현재 개발 중이다. Cross쪽에서는 개발 환경의 GUI화가 이슈이며, VRTX의 Spectra와 LynxOS의 PosixWorks가 가장 먼저 출시되었었고, 최근 Tornado라 불리우는 VxWorks의 GUI가 최근 출시되었다.

Native쪽에서는 대부분 UNIX세계의 표준인 X를 지원하면서도 독자적인 Window System을 가지고 있고, GUI개발툴을 같이 제공하고 있다. QNX사의 Photon이라는 microGUI가 매우 작은 크기로 안정적인 GUI환경을 제공하고 있는 것으로 알려져 있으며, OS-9의 경우에는 Third Party에서 제공하고 있는 것으로 알려져 있다.

인터넷과 관련하여서는 대부분의 실시간 운영체제가 기본적으로 TCP/IP를 지원하고 있으며 나아가서 NFS, SLIP/PPP, Web Server, Web Browser, Email 등등의 개발툴들도 지원하는 방향으로 적극 선회하고 있다. 대부분의 업체에서 특화된, 인터넷 관련 개발툴을 경쟁적으로 내놓고 있다.

JAVA와 관련하여서도 대부분의 상용 실시간 운영체제 업체들이 나름대로의 정책을 내놓고 있다. LynxOS나 QNX와 같이 아예 Enterprise JVM을 올린 업체도 있고, VRTX와 같이 임베디드 시스템에 JVM을 올리는 것은 현실적인 요구도 별로 없을 것이고 또한 과도한 하드웨어 자원을 추가로 필요로 하므로 어울리지 않다고 판단하고 단순히 JAVA의 언어 특성에 주목하여 Cross 개발환경에서 특정 Target보드 위에서 VRTX와 함께 돌아가는 기계어를 만들어 주는 JAVA Cross Compiler를 개발하는 정책을 발표한 곳도 있다. 대부분은 임베디드 JVM을 올리는 방향으로 가닥을 잡고 있으며 현재 개발 중이다.

한편, 기존 UNIX 세계와 Windows 세계와의 관계 설정을 어떻게 하느냐 하는 것이 중요한 관건으로 등장하고 있다.

전체적으로는 UNIX 세계의 표준인 POSIX를 기본으로 지원하는 방향으로 나아가고는 있지만, Win32 API에 대한 지원 방안도 사실상 다각도로 연구되고 있는 실정이다. 게다가 최근 Microsoft사에서 Win32 API를 완벽하게 지원하고 있는 윈도우 패밀리로서, WinCE를 출시한 지난 해 이후 더욱 핫 이슈가 되고 있다. QNX와 같이 단순히 윈도우용 터미널을 제공하거나 혹은 그 반대의 NT터미널 기능을 구현한 경우 외에는 아직 구체적인 정책이 발표된 것은 없지만 상당히 다각도로 이 점을 고려하고 있는 것으로 알려지고 있다.

또한 상용 실시간 운영체제의 특정 CPU 및 특정 상용보드에 대한 지원 여부는 영업현장에서 매우 중요한 이슈이지만, 프로세서 업체와 상용보드 업체, 그리고 실시간 운영체제 업체들의 정책 및 각각의 시장 상황이 얽혀서 매우 복잡하여 한마디로 말하기가 어려운 상황이다.

임베디드 32-bit 처리기 시장에서 널리 알려져 있는 주요 프로세서를 살펴보면, 모토로라의 68k 및 PPC, 실리콘 그래픽스사의 MIPS, 히다찌사의 SuperH, 인텔계열의 x86, i960 그리고 ARM 등이 있다. 사실 너무 많은 칩이 있어 이를 상용 실시간 운영체제 업체에서 모두 지원한다는 것은 사실상 불가능하며, 결국 주요한 칩에 대해서만 지원하고 그 외의 칩에 대해서는 특수한 경우에 제한적으로 지원하는 방식으로 진행되고 있다.

대부분이 모토로라 68k, PPC 그리고 x86,ARM에 대해서는 기본적으로 지원하고 있다. X86만을 지원하였던 QNX도 최근 Cross Compiler 전문 업체인 Networks사와의 전략적으로 결합하여 PPC, MIPS 등을 지원하게 되었다.

흔히 임베디드 시장은 모토로라 및 ARM등의 RISC 영역으로 치부되어 왔었는데, 최근 인텔의 적극적인 공세로 CISC칩의 사용도 증가하고 있는 편이다.

이러한 칩 이슈는 결국 컴파일러 이슈와도 연관이 있다. 크게는 독자적인 컴파일러를 가질 것이냐

아니면 GCC 같은 널리 사용되고 있는 컴파일러를 사용할 것이냐 하는 선택을 강요받게 된다. 어떤 것이 올바르다고 말할 수는 없으나, LynxOS와 VxWorks는 GCC를 ISI는 Diab을, VRTX는 독자적인 Microtec Compiler를 가지고 있다. ISI가 Diab을 M&A하였고 지금은 VxWorks에 합병되었고, 또한 Microtec이 Ready System사의 VRTX부분을 M&A한 것에서 알 수 있듯이 필요시 상호간의 M&A도 매우 활발하게 이루어 지고 있다.

게다가 상용보드의 지원 여부도 매우 중요한 이슈인데, 결국 칩업체, 보드 업체, 실시간 운영체제 업체들간의 전략적인 동맹 관계 및 M&A가 매우 중요하게 고려되어져야 할 것이다.

2.4 새로운 시장의 출현과 새로운 경쟁자들

실시간 운영체제 시장의 급격한 변화는 특히 데스크탑 솔루션의 임베디드화, 그리고 휴대형 인터넷 정보 단말기 같은 새로운 개념의 양산되는 신개념 제품이 시장에 소개되고 있는 것으로부터 연유한다. 이 시장에서 영원한 강자도 승자도 존재하지 않는다고 이야기 할 정도로 급격하게 신기술이 도입되고 다양한 종류의 전략적 동맹관계, 그리고 인수합병 개념의 제품이 나타나고 있는 실정이다.

이러한 새로운 개념의 양산시장은 주로 마이크로소프트사가 주도하고 있는데, Microsoft는 WinCE라는 임베디드 운영체제를 새로이 내놓고 WinCE의 주력 공략시장을 자동차용 Car PC, Palm PC, Handheld PC 등으로 크게 3가지로 구분하고 각각에 대해 별도의 개발툴을 제공하는 계획을 가지고 있다.

MS 및 Sun사의 진입과 새로운 정보 단말기의 등장으로 상용 실시간 운영체제 시장은 새로운 단계로 접어들었다고 볼 수 있다. 즉 이제까지는 제어시스템 등에 깊숙히 박혀 있는 블랙박스 같은 기능을 하였으나 이제는 비록 각양각색을 띠고 있다 하더라도 사용자가 직접 보고 제어하는 방식으로 겉모습을 나타내게 된 것이다. 이러한 연유로 하여 최근 언론에서도 정보가전 운영체제, 혹은 정보 단말 운영체제 등으로 부르며 열띤 취재 경쟁을 보이고 있다.

흔히 정보가전시장이라 불리우는 이 시장은 반도체 등의 초소형 기술의 발전과 더불어 인터넷과 휴대형 통신기기, 그리고 방송매체를 결합하는 새로운 개념의 제품들이 주도하고 있다.

휴대폰과 전화를 결합한 SmartPhone에서 알 수 있듯이 이 시장의 요구는 매우 한정된 하드웨어 자원으로 고기능을 고속으로 처리하고 또한 지속적인 Upgrade도 가능하여야 한다는 것이므로 이는 자연스럽게 상용 실시간 운영체제를 요구하게 된다.

그럼에도 이 시장은 기존 시장과는 매우 다른 요구, 특히 최종 사용자가 GUI를 기본으로 요구하고 Windows 등의 데스크 탑 환경과의 호환을 원하는 등의 새로운 요구에 신속하게 응대하여야 하기 때문에 막대한 마케팅 능력과 자금력을 가진 WinCE 혹은 임베디드 JAVA 등의 새로운 경쟁자들이 매우 강세를 보이고 있으며, 최근에는 Linux가 여러 정보 단말기에 사용되면서부터 아주 강력한 임베디드 운영체제의 시장을 차지할 것으로 전망되고 있다. 따라서 전통적인 상용 실시간 운영체제업체는 더욱 치열한 경쟁에 직면하고 있는 실정이다.

2.5 임베디드 시스템의 개요

임베디드 시스템의 운영체계에서 표준화된 대량생산이 등장한 것은 70년대 후반이며, 이들 대다수는 어셈블리 언어로 제작되었고, 개발 대상으로 하는 마이크로프로세서에서만 사용할 수 있었다. 그러므로 마이크로프로세서가 구식이 되면, 그 운영체계도 구식이 되었다.

C언어가 나타난 그때부터 운영체계를 효율적, 안정적이고 포터블(Portable)한 방법으로 작성할 수 있었다. 이는 현재의 마이크로프로세서가 구식이 되었을 때, 그때까지 들인 소프트웨어에 대한 투자를 보호할 수 있었기 때문이다. 이는 마케팅 관점

에서 무척 반가운 이야기로 들렸으며, 결국 C로 작성된 운영체계는 표준이 되었고 오늘날까지도 남아있다. 다시 말해서 소프트웨어의 재사용이 이루어지게 되었고 오늘날까지 이어지고 있는 것이다.

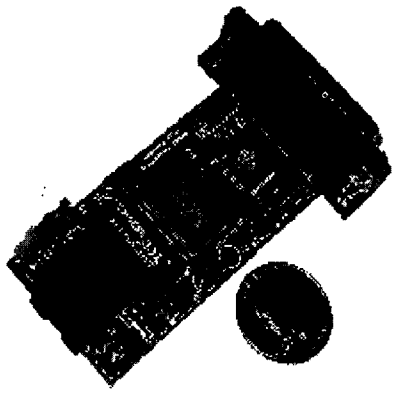
그럼 이러한 역사를 토대로 임베디드 시스템이 무엇인가를 알아보도록 한다. 데이터베이스나 응용 프로그램의 경우는 특별히 내장됐다는 표현을 쓰지 않지만, 엘리베이터나 텔레비전 등의 경우에는 내장된 시스템이라는 용어를 쓰고 있는데, 이런 분야에서 임베디드 시스템(Embedded System)이라는 말이 거론될 수 있다.

우리 주변에는 인공지능 혹은 퍼지라는 기능을 채택한 전자제품이 많이 나오고 있다. 퍼지 세탁기, 인공지능 전기밥솥 등이 대표적인 제품이라고 할 수 있는데, 단지 세탁을 하고 밥을 하거나 얼음을 얼리는 등의 기능이라면 특별히 인공지능이란 표현을 쓰지 않아도 될 것이다. 여기에는 세탁물의 종류에 따라, 혹은 밥을 지을 때 들어가는 재료에 따라, 냉장실 또는 냉동실에 들어가는 재료에 따라 기기가 자동/수동으로 반응하기 때문에 인공지능이란 표현을 쓴 것이다.

그런데 여기서 중요한 것은 이런 모든 기능을 회로만으로 구성해서 구현한다는 것은 사실상 불가능하다는 점이다. 적당한 제어용 CPU가 있고, 또 그 기기의 기능에 맞는 프로그램이 탑재돼 있어 그 프로그램을 통해야만 기능을 구현할 수 있는 것이다. 바로 이런 것이 임베디드 운영체계라 할 수 있다.

임베디드 시스템에 대한 예를 일상에서만 찾아보았지만, 실제로 임베디드 시스템이 요구되는 곳은 무궁무진하다. 특히 공장자동화나 가정 자동화와 같이 자동화 분야에서는 필수적인 요소로 부각되고 있다. 즉 임베디드 시스템이란 기계 또는 전자 장치의 두뇌 역할을 하는 마이크로프로세서를 장착해 설계함으로써 효과적인 제어를 할 수 있도록 하는 시스템을 의미한다.

임베디드 시스템이 바로 이런 것이다라는 것을 보여줄 만한 예로서 바로 작은 웹서버를 만들수 있다는 가능성을 보여준 사이트가 있다. (http://www.picoweb.net)



(그림 6) 임베디드 초소형 서버 picoweb

리눅스의 슬림사이즈를 구현한 uClinux/Coldfire Project은 Motorola Coldfire 프로세스 계열로 구성된 리눅스 기반 시스템을 구성한 것이다. 커널은 Micro-controller linux(uClinux)로 만들었고, GNU/Linux 유틸리티를 Coldfire에 포팅하였는데, 이런 기반의 프로젝트들이 요사이 많이 등장하는 모습을 볼 수 있다. 그만큼 성능대비 가격에 대한 내용을 고려한 것이라고 볼 수 있겠다.

현재 지원되는 내용은 Linux 커널 버전 2.0.38을 기반으로 하였고 Ethernet, PPP 등과 같은 네트워크 환경도 비교적 안정적으로 지원하고 있으며, IP-masquerading과 Dial-on-demand의 동작도 가능하다. 또한 NFS, SMB 파일 시스템 마운트도 가능하고, 다른 시스템으로 부터 Coldfire 바이너리를 실행할 수도 있다.

3. 결 론

국내 임베디드 소프트웨어 산업 현황을 살펴보면 휴대폰을 기반으로 하는 스마트폰 분야와 PC

산업을 바탕으로 발전해온 PDA 분야가 대표적이라고 할 수 있다. 이를 토대로 실시간 운영체제의 국산화 및 여러 응용 제품들이 출시되고 있는 상황이다. 배터리 사용시간, 다양한 주변기기 접속, 무게, 디자인 등이 가장 중요한 쟁점이 될 수 있으며, 또한 무선모뎀을 이용한 통신 속도 개선에 관한 연구도 중요한 분야라고 할 수 있다. 그 외 인터넷 단말기, 셋탑박스 등의 산업이 활성화 될 전망이다. 결과적으로 무선통신이 결합된 이동통신형 임베디드 시스템 산업의 경쟁력이 치열할 것으로 예상된다.

그 외 자바 미들웨어 분야에서도 임베디드 자바 미들웨어의 연구 개발 활성화가 요구된다. 경량화, 실시간 서비스, 고성능 지원, 개발의 용이함이라는 각기 상반된 특성을 동시에 요구하는 특성이 있다.

그리고 임베디드 시스템 산업을 목표로 하는 브라우저 산업 역시 2005년에는 12억 6천만 명이 무선을 사용하게 될 것으로 전망되기 때문에 매우 중요한 연구 개발 분야라고 할 수 있다. 우선 무선 단말기용 무선 브라우저가 필요하다. 이는 압축 기술의 발달과 대역폭 증가, 전송 기술의 발전 등으로 성능이 지속적으로 향상되고 있는 실정을 감안해서 국가적으로 연구 개발을 서둘러야 한다.

그 외에도 디지털 정보가전 단말 기술 분야를 주시해야 하는데, 그 이유는 게임기 , 디지털 TV 및 PC간의 경쟁이 치열해 질 것으로 예상되기 때문이다. 제어기와 정보단말의 통합: 응용별 특화된 정보단말 보급이 보편화될 것이며 궁극적으로는 멀티미디어 대응 정보가전 단말기가 그 최종 발전목표가 될 것이다.

결론적으로 임베디드 소프트웨어 산업 현황을 보면 전 세계적으로 이러한 분야의 투자와 연구가 집중화되고 있는 추세이다. 임베디드 소프트웨어 국내 산업은 아직 그 기본 인프라가 취약한 반면, 대기업 및 벤처 기업들의 응용 제품에 관한 연구 및 개발은 상대적으로 매우 활발한 상황이라고 할 수 있다. 임베디드 소프트웨어 산업의 육성 방안을 나열하면 아래에서 보는 바와 같다.

- 임베디드 소프트웨어 인력 양성 체계를 형성하여 임베디드 소프트웨어 연구 및 개발 인력을 체계적으로 육성하는 것이 필요하다. 우선적으로 임베디드 소프트웨어 교육 기관의 설립 및 교육프로그램 구축이 요구된다.
  - 대기업/벤처기업/사설학원/대학을 통한 인력양성
  - ESRC/IDEC 등과 같은 전문연구기관을 통한 인력 양성
- 임베디드 소프트웨어와 관련된 교과과정을 대학 교육 커리큘럼에 반영하는 것이 필요하다. 교육 커리큘럼은 로드-맵을 구축하여 체계적인 전문가 양성을 할 수 있도록 하여야 한다.
  - 하드웨어, 소프트웨어 동시-디자인 능력 양성에 역점을 둔 교육
  - 물리/수학/알고리즘/System Programming 등 기초과목 강화에 기반을 둠
  - 커리큘럼 로드-맵(Road-Map)이 필요
  - 교육부 혹은 ESRC 등과 같은 전문연구기관과 긴밀한 협조가 요구됨.
- 임베디드 소프트웨어 개발 전 과정을 위한 교육 및 실험용 툴-킷 개발이 요구된다. 즉, 효율적인 임베디드 소프트웨어 전문가 양성에 필요한 첨단 실험용 도구들의 연구 개발이 매우 중요하다.
- 실시간 운영체제 표준화 단체 육성 및 지원이 필요하다. 예를 들면, 무선통신모듈은 핵심기술을 전량 수입 상태이다. 그러므로 이를 위해 무선통신 모듈기술을 표준화하고 테스트베드를 제공하여 국내 자체적으로 기술 확보에 치중하는 것이 요구된다.
  - 국내 표준화 단체 지원 및 국제 표준화 단체와 연계 필요
  - 실시간 운영체제 및 개발도구 연구 개발 지원

- 보안성이나 QoS를 위한 지원 같은 첨단 기술의 필요
- 임베디드 리눅스의 대부분이 실시간성의 지원 강화 요망
- 저전력 시스템 연구 분야 지원 필요
- 통신모듈기술의 표준화

• ESRC, IDEC 등 임베디드 시스템 연구 단체 육성 및 지원
- 체계적인 임베디드 시스템 개발 능력 배양 필요
- 단기적인 연구과제 및 장기 적인 기초연구과제 지원 필요

• 특정한 경쟁력있는 임베디드 소프트웨어 컴포넌트 개발과 관련된 연구 지원이 요구된다. 실시간 미들웨어, 셋탑박스 Api, PDA 개발용 Api 개발 등이 이러한 연구 분야의 한 축이 될 것이다.

참고문헌

[1] Edward A. Lee, "What's Ahead for Embedded Software?", IEEE Computer, September 2000, pp. 18-26

[2] Wolfgang Fleisch. Applying Use Cases for the Requirements Validation of Component-Based Real-Time Software. In Proceedings of 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'99), IEEE Computer Society Press, 1999

[3] Clemens Szyperski. Component Software-Beyond Object-Oriented Programming, Addison-Wesley, 1997.

[4] Sun's Java Community Process Real-Time Expert Group, "Proposed Java Real-Time API, v0.2," Technical report, Java Software Division of Sun Microsystems, December 1998. http://www.sdct.itl.nist.gov/carnahan/real-Time/Sun/api

[5] The Real-Time for Java Expert Group, "Real-Time Specification for Java, v0.8.1," Technical report, RTJEG, September 1999. http://www.rtj.org.

[6] J Consortium, "Draft International J Consortium Specification," Technical Report, J Consortium, September 1999. http://www.j- consortium.org/.

[7] K. H. Krause and W. Hartmann, "RT Java Proposal," Technical Report, A&D GT 1, 1999. http://www.j-consortium.org/rtjw

[8] K. Nilsen, "Adding Real-Time Capabilities to Java," Communications of the ACM, 41(6), June 1998. page 49-56

[9] G. Hilderink, "A new Java thread model for concurrent programming of real-Time systems," Real-Time Magazine, January 1998, pp. 30-35

[10] G. Back, P. Tullmann, L. Stoller, W. Hsieh, and J. Lepreau. Java Operating Systems: Design an implementation. Technical report, Department of Computer Science, University of Utah, August 1998, www.cs.utah.edu/projects/flux.

[11] H. McGhan and M. O'Connor, picoJava: a direct execution engine for Java bytecode, IEEE Computer, 31(2), October, 1998

[12] G. Bollella, B. Brosgol, P. Dibble, et al., "The Real-Time Specification for Java", The Real-Time for Java Expert Group, December 1999.

[13] Java Community Process, "J2ME Connected, Limited Device Configuration," JSR-000030, May, 2000

[14] Java Community Process, "Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices," A White Paper, May, 2000.

[15] Java Community Process, "The K Virtual Machine," A White Paper, June 1999.

[16] A. Andersen, G. S. Blair, G. Coulson, F. Eliassen, "A Reflective Component-Based Middleware in Python," NORUT IT, IPC8, September 1999.

[17] R. Hayton et al., "FlexiNet Architecture Report," ANSA Phase III report, February 1999

[18] G. Coulson, "A Configurable Multimedia Middleware Platform," IEEE Multimedia, Vol. 6, No. 1, January-March 1999.

[19] D. Isovic, M. Lindgerg and I. Crnkovic, "System Development with Real-Time Components", Malardalen University, Sweden, http://www.mrtc.mdh.se

[20] D. Isovic, M. Lindgerg, "Real-Time Components", Malardalen University, Sweden, http://www.mrtc.mdh.se

[21] "임베디드 기술보고서", 두올정보기술, 2001. 5

[22] N.Audsley, A. Burns, M. Richardson, and A. Wellings, "Hard Real-Time Scheduling: The Deadline-Monotonic Approach", Proceedings of IEEE Workshop on Real-Time Operating Systems and Software" , pp. 133-137, 1991

[23] 정보처리 제5권 4호 (1998.7) 신현식, 김태응

[24] ESFT의 Embedded System (http://www.esft.fr)

[25] Oregon Micro Systems (http://www.omsmotion.com)

[26] Real-Time Linux (http://www.realtimelinux.org)

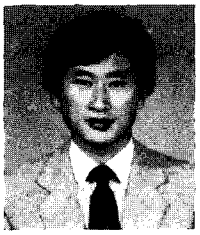
[27] Lynx Real-Time Systems Inc. (http://www.lynx.com)

[28] Embedded System (http://www.embedded.com)

[29] RTOS' 99 자료집

[30] 국내 Real-Time Linux Forum Site(http://rtlinux.to)

저자약력



이 정 배

1981년 경북대학교 전산공학과(공학사)
1983년 경북대학교 전산공학과(공학석사)
1995년 한양대학교 전산공학과(공학박사)
1982년-1991년 한국전자통신연구원 선임연구원
1996년-1997년 U.C.Irvine 객원교수 Dept.of Electrical & Computer Eng.
1991년-현재 부산외국어대학교 컴퓨터공학과 부교수
관심분야 : 분산 UNIX 시스템, 원격 영상감시 및 제어,멀티미디어 서버

이 두 원

1991년 숭실대학교 전자계산학과 (공학사)
1995년 일본 정보처리전문가협회 네트웍스페셜리스트
1991년-1996년 (주)교보정보통신 연구원
1997년-1998년 (주)블루버드코리아 기술이사
1998년-현재 (주)두올정보기술 대표이사
관심분야 : 실시간 OS, 임베디드시스템, 인터넷정보가전
e-mail : dwlee@doall.co.kr