임베디드 멀티미디어 시스템 구현을 위한 기술 분석

조 수 현*

1. 서 론

1.1 임베디드 시스템 개요

오늘날 IT 기술의 발전으로 초고속 네트워크를 기반으로 하는 멀티미디어 서비스와 멀티미디어 가전, MP3 재생기, 이동 멀티미디어 재생기(PMP) 등을 중심으로 사용자의 요구에 즉각적인 응답이 이루어지는 형태의 서비스를 제공하고 있다. 다시 말해 멀티미디어 콘텐츠를 이동기기 내 프로그램에 의해서 처리가 이루어지고 이를 위해서 여러 가지 기술들이 적용되고 있는 것이다.

임베디드 시스템(embedded system)은 제한된 자원을 갖고 특정한 목적을 갖는 작업을 처리하는 시스템을 의미한다. 즉 일상생활에서 사용되는 전자기기, 제어장치 등과 같이 단순히 회로로만 구성된 것이 아닌 마이크로프로세서가 내장되어 있고 그 마이크로프로세서를 구동하여 특정한 기능을 수행하도록 프로그램이 내장되어 있는 시스템을 뜻한다. 적용 분야로는 TV, 냉장고, 로봇, 텔레매틱스, 핸드폰, PDA, 항공관제, 군사용 제어 장치 등 매우 광범위하다[1]. 그림 1은 임베디드 시스템이 적용된 분야를 나타낸다.

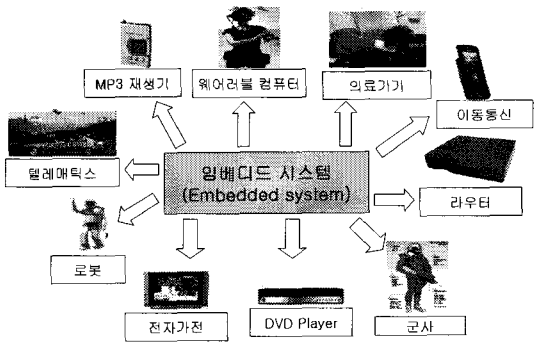


그림 1. 임베디드 시스템 적용 예

임베디드 소프트웨어는 임베디드 시스템에 내장되어 있는 소프트웨어 기술을 말한다. 즉 하드웨어를 제어하기 위한 임베디드 시스템에 들어가는 명령어 집합으로 하드웨어를 제어하고 하드웨어 운영에 필요한 정보를 사용자에게 보여주고 사용자로부터 받는 역할을 한다. 센서와 통신 장비를 통해 들어오는 외부 자극을 처리하는 사건 구동 방식으로 동작하는 특성이 있으며 작은 메모리와 느린 CPU에서 동작하도록 구현한다[2].

임베디드 소프트웨어는 임베디드 운영체제, 특정 응용에 적용하기 위한 미들웨어, 서비스 개발을 위한 라이브러리, 응용 소프트웨어를 개발하기 위한 개발 도구, 응용 소프트웨어/서비스 동작시 시스템을 분석하기 위한 도구, 임베디드 시스템을 구성하기 위한 구성 환경 등을 포함한다. 이와 같이 멀티미디어 콘텐츠를 저장, 관리, 재생 가능한

※ 교신저자(Corresponding Author) : 조수현, 주소 : 경북 구미시 양호동 1번지(730-918), 전화 : 054)478-7897, FAX : 054)478-7539, E-mail : shcho@kumoh.ac.kr
* 금오공과대학교 컴퓨터공학부 계약교수

임베디드 시스템을 위해서 많은 임베디드 소프트웨어 기술이 절실히 요구되고 있다. 또한 정보통신부에서도 최근 u-IT839 정책에서 임베디드 소프트웨어를 9대 신성장동력에 대한 공통기반기술로 정의하고 있다. 그만큼 IT/비IT 제품의 부가가치 향상에 직접적인 요소라는 점에서 그 중요성은 날로 증가하고 있다.

1.2 임베디드 시스템과 PC와의 차이점

PC(personal computer)는 CPU를 중심으로 다양한 응용 프로그램을 구동하기 위해서 많은 양의 메모리와 대용량의 보조 기억장치, 그래픽 작업을 위한 고성능 그래픽 카드와 인터넷을 위한 랜 카드 등 추가적인 장치가 필요하다. 또한 하드웨어를 구성하고 이용함에 있어 필수적으로 운영체제 및 응용 프로그램은 상대적으로 규모가 커지게 된다.

PC의 목적은 여러 분야에서 사용할 수 있고 다양한 기능을 이용하여 보다 더 많은 효과를 얻는 것이다. 하지만 다양한 기능을 가진 PC를 단순한 분야에 사용하다 보면 사용하고자 하는 기능에 비해 시스템 자체가 상당히 비대해지며 그에 따른 소요 비용이 증가한다. 따라서 단순한 기능을 포함하고 비용절감 측면을 고려한다면 임베디드 시스템이 필요할 것이다[3].

임베디드 시스템이라고 하여 PC와 전혀 다른 것은 아니다. 복잡하게 구현된 대형의 임베디드 시스템에서도 PC와 같은 소프트웨어 계층을 가지고 있다. 하지만 소형의 경우는 다르다. 그림 2의 임베디드 시스템 구조와 같이 PC의 CPU에는 포함되어 있지 않은 내부 메모리 영역과 각종 기기와 통신을 위한 직렬 통신, 특수 기능 포트(port)를 이용한 다양한 외부 제어기능 그리고 아날로그와 디지털 신호를 입출력할 수 있는 장치

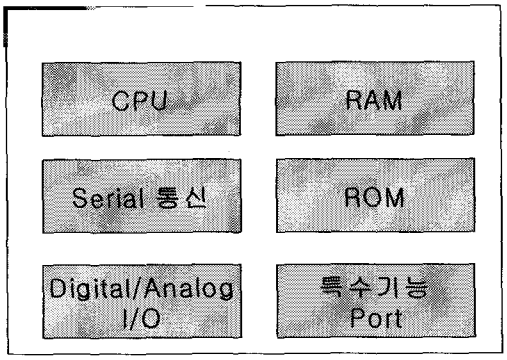


그림 2. 임베디드 시스템 구조

까지 PC에서 구성하였던 여러 장치들을 하나의 원칩 프로세서로 구성하여 주변회로에 대한 간략화와 값싼 비용을 이용하여 소규모의 장치에 활용할 수 있도록 되어 있다[4].

본 논문에서는 임베디드 시스템에서 멀티미디어 콘텐츠들에 대해 저장, 관리, 재생 등을 효과적으로 처리하기 위해 운영체제 차원에서 필요한 기술들을 살펴볼 것이다.

2. 임베디드 소프트웨어 특징

임베디드 시스템 내 소프트웨어는 개별적으로 동작하지 않는다. 이는 하드웨어와 밀접한 연관이 있기 때문이다. CPU가 동일하더라도 환경이 조금만 달라지면 동작하지 않는다. 비용을 줄이기 위해 임베디드 시스템의 하드웨어가 항상 최적화된 부품만 포함하고 있기 때문이다. 본 장에서는 일반적인 임베디드 소프트웨어의 특징에 대하여 살펴본다.

2.1 실시간성

일반적으로 범용 시스템의 처리는 주어진 자원을 최대한 효과적으로 활용하여 가능한 빠르게

수행하는 것을 목적으로 한다. 이는 절대적인 처리 기한이 없이 주어진 시간 안에 되도록 많은 일을 수행하도록 설계되었음을 의미한다.

임베디드 시스템에서의 실시간성은 외부 자극에 대해 시간에 맞추어 예측 가능한 방식으로 반응해야 한다. 계산이 정밀해야 한다는 조건 이외에 결과를 산출하는 시간에 제한(deadline)을 둔다는 점에서 범용 시스템과 다르다. 시간은 고정된 값이 아니라 구축하는 시스템마다 큰 차이를 보일 수 있다. 실시간성의 종류는 표 1과 같으며 실행 제한 시간을 어겨서는 안되는 경성 실시간(hard real time)과 어느 정도 허용되는 연성 실시간(soft real time)이 있다.

그림 3은 연성 실시간과 경성 실시간 시스템을 비교한 그림이다. 연성 실시간은 외부 자극에 대한 반응이 어느 정도 지체될 경우에도 실패하지 않는 특성이 있다. 그렇다고 해서 무조건적인 지체를 허용하는 것은 아니라 불가피한 상황에서 벌어지는 지체만 허용한다. 대표적인 예는 항공권 발권 시스템이다. 사람이 기다릴 수 있을 만큼 반응이 늦어져도 인명에 피해를 주지 않지만 고객이 상당히 불만을 가질 것이다.

표 1. 실시간성 종류

종류 / 특성	경성 실시간	연성 실시간
하드웨어	특수한 전용 하드웨어	일반 하드웨어 + 일부 특수한 하드웨어
자극에 대한 반응정도	오차를 허용 하지 않음	오차를 어느 정도 허용함
자극에 대한 반응이 실패한 경우	인명과 금전적인 피해를 줌	사용자 불만이 커짐
응용분야	항공기, 의료기기, 산업용 등	비디오/오디오 재생기, 게임 등

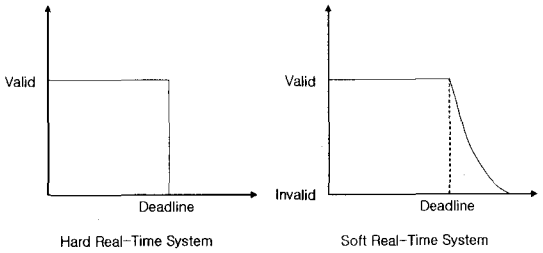


그림 3. 경성 실시간과 연성 실시간 시스템 비교

경성 실시간은 시간에 대한 조건이 있다는 것은 연성 실시간과 같다. 하지만 외부 자극에 대해 정해진 시간 내에 완벽하게 반응을 보이지 않을 경우 응용 프로그램 자체가 실패하는 특성이 있다. 이럴 경우 큰 인명피해 등이 야기될 수 있다는 것이 중요한 점이다. 항상 주어진 시간 내에 처리할 수 있다는 가정 자체가 중요하다.

대표적인 예로 오폭을 막기 위해 한 치의 오차도 허용하지 않는 미사일 제어시스템, 수술에 사용되는 의료기기에 탑재하는 제어시스템이다. 실제 실시간 응용 프로그램은 일반적으로 경성 실시간과 연성 실시간을 모두 포함하고 있다. 어떤 부분은 경성 실시간으로 동작해야 하고 어떤 부분은 연성 실시간으로 동작해도 무방하기 때문이다. 즉 설계하고자 하는 임베디드 시스템이 어떤 시간 제약성을 요구하는지를 미리 분석하여 적용하는 것이 중요하다.

2.2 안정성

임베디드 시스템은 열악한 환경에서 동작하는 경우가 많다. 가령 기온이 영하로 떨어졌다고 전화기가 동작하지 않거나 여름에 자동차 시동이 걸리지 않고 엔진 제어가 불가능 하다면 문제가 발생할 수 있다. 이 보다 극적인 신뢰성을 요구하는 장비가 있는데, 미사일 유도시스템, 레이더 시스템과 같은 군사용 장비와 MRI, 라식 수술과 같

은 의료 장비는 조금만 잘못되면 인명 피해가 발생할 수 있으므로 더욱 정밀하고 안정성을 보장해야 한다.

2.3 가격

범용 시스템 내의 소프트웨어는 최신 기술을 모두 적용하는 동시에 확장성까지 고려해야 하므로 구매당시에는 사용하지 않는 많은 기능들을 포함하려 한다. 이는 구매자들이 기능 외 가격에도 민감하기 때문이다. 반면 임베디드 시스템은 시장에 대량으로 공급해야 하므로, 가격에 상당히 민감하다. 소비자에게 필요한 기능 이외에 것은 제외하고 성능을 떨어뜨리거나 기능을 단순하게 만들 필요가 있는 것이다.

2.4 크기

일반적으로 임베디드 소프트웨어는 크기가 매우 작다. 예를 들면 어린이용 게임기에 들어가는 소프트웨어는 16-64 킬로바이트면 충분하다. 메모리 크기는 임베디드 시스템에 탑재한 마이크로프로세서와 연관이 있다. 8비트 CPU로 제어할 수 있는 메모리 공간에는 한계가 있기 때문에 최근에 디지털 정보기기는 강력한 32비트 CPU를 탑재하고 있으므로 큰 소프트웨어를 운영할 수 있다.

2.5 특수성

임베디드 소프트웨어를 범용으로 만든 경우는 찾아보기 어렵다. 물론 상황에 따라 몇 가지 작업을 수행하게 작성할 수는 있지만 PC와 같이 다용도로 사용할 수 있는 소프트웨어는 만들지 않는다. 특히 하드웨어를 제어할 목적인 경우에는 특수 작업만을 위해 소프트웨어를 만드는 경향이 두드러진다. 하지만 최근 PDA, PMP 등의 멀티미디어를 처리하는 임베디드 시스템에서는 다양한 요구사항을 위해 특수성을 벗어나 범용성을 추구하는 경향이 많아지고 있다.

2.6 성능

임베디드 소프트웨어는 주어진 하드웨어 성능을 최고로 이끌어내야 하므로 높은 성능을 발휘하게 작성해야 한다. 주의할 점은 성능이 좋다고 해서 소프트웨어 자체가 무작정 빨라야 할 필요는 없다. 일반인을 상대로 하는 제품일 경우에는 하드웨어와 보조를 맞춰 요구사항을 정확히 충족시킬 수 있을 수준으로 성능 목표를 정하는 경우가 대부분이다.

2.7 오류 처리

사용자 개입이 적은 임베디드 소프트웨어는 오류 처리에 대해서 고려해야 한다. 전문가가 사용하는 임베디드 소프트웨어도 있지만 시스템의 오동작에 적절히 대응할 수 없는 비전문가용 임베디드 소프트웨어도 많기 때문이다. 실시간 시스템에서 실시간성이 진가를 발휘할 때는 비상사태에서 적절히 대응해 오류를 처리할 때이므로 실시간 시스템에 탑재하는 오류 처리는 더욱 정밀하고 복잡해져야 한다.

2.8 사용자 인터페이스

기계를 제어하는 데 필요한 최소 정보만 보여주고 최소 입력만 받을 수 있으면 되기 때문에 임베디드 소프트웨어는 대부분 사용자 인터페이스가 제한적이다. 물론 비행기를 제어하는 소프트웨어는 현재 일어나는 모든 상황을 운영자에게 보여줄 수 있어야 하므로 사용자 인터페이스가 상당히 복잡하다. 최근 기술 발전으로 디지털 정

보기기를 위한 임베디드 장치는 LCD 화면과 필기체 인식 소프트웨어를 탑재하기도 한다. 앞으로는 임베디드 장치에 탑재한 사용자 인터페이스도 PC처럼 화려해지고 사용자 중심으로 발전할 것이다.

3. 임베디드 소프트웨어

3.1 실시간 운영체제(real time operating systems)

초기의 임베디드 시스템은 비교적 단순해서 운영체제가 없이 사람이 순차적인 프로그램을 작성해서 수행하도록 했고 중간에 인터럽트(interrupt)가 발생되는 경우에만 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 실시간 이라는 요소를 만족해야 했다. 그래서 실시간 운영체제가 임베디드 시스템에 도입된 것이다.

일반적으로 실시간 운영체제(이하 RTOS)는 임베디드 시스템에서 동작하면서 외부 입력에 대해 정해진 시간 내에 반응할 수 있도록 기능을 제공하는 실행환경을 의미한다. 또한 구현 측면의 RTOS는 임베디드 시스템에서 동작하면서 여러 개의 실행 태스크(task)를 생성할 수 있는 API를 제공하는 환경이라 할 수 있다. 표 2는 실시간성을 제공하고 있는 운영체제의 종류를 나타내며 그 중에서 전세계적으로 많이 사용되고 있는 것이 VxWORKs이다. 그림 4는 VxWORKs의 구조를 나타낸다. VxWORKs는 마이크로 커널 기반의

표 2. 실시간 운영체제 종류

종류/구분	무료 RTOS	상용 RTOS
실시간 운영체제	- MicroC/OS-II - eCOS - RTLinux	- pSOS - QNX - VRTX - VxWORKs - Nucleus PLUS - Windows CE

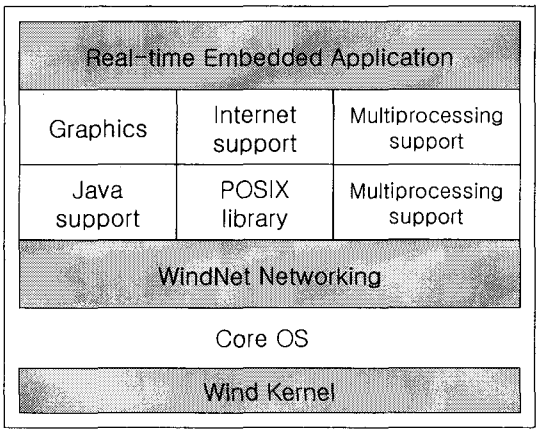


그림 4. VxWORKs 구조

클라이언트/서버 구조의 운영체제로서 개발환경과 네트워크 환경을 제공하며 Wind 마이크로 커널이란 이름으로 멀티태스킹, 스케줄링, 태스크간 통신/동기화, 메모리 관리 기능이 포함된 실시간 커널이다.

이와 같이 RTOS는 태스크 간에 메시지를 전달하고 인터럽트 핸들러와 태스크 간에 이벤트를 전달하고 여러 개의 태스크가 I/O 장치 사용을 중재하는데 필요한 실행환경을 생성한다. RTOS의 조건을 만족하기 위해서는 시스템으로 들어오는 입력에 대해 예측 가능한 시간 내에 동작하도록 만들어야 한다[5].

과거에는 일반 운영체제에 하드웨어 부분 관련 시간적 제약 조건만을 고려하면 되었지만 사용자의 요구사항과 처리 내용이 증가되면서 태스크라

는 개념이 포함되면서 점차 복잡한 구조를 요구하게 된 것이다. 단순한 임베디드 시스템은 운영체제와 응용 프로그램이 하나로 합쳐진 펌웨어 방식으로 작성한 소프트웨어를 내장하므로 태스크와 스케줄러에 대한 복잡한 요구 사항이 필요 없다. 하지만 복잡한 기능을 수행하는 임베디드 시스템을 위해서는 RTOS가 다음과 같은 요구사항을 충족시킬 수 있어야 한다[6-7].

- 우선순위가 높은 태스크는 언제나 우선순위가 낮은 태스크에 앞서 수행된다.
- 우선순위 뒤바뀜 현상(priority inversion)은 지정 시간 내 풀려야 한다.
- 커널은 태스크 동작을 방해해서는 안된다.

위의 요구사항을 만족하기 위해서 가장 고려가 되어야 할 부분은 태스크를 관리하는 스케줄러이다. 즉 RTOS의 핵심은 스케줄러인 것이다. 스케줄러는 다음번에 어떤 태스크를 실행해야 할지 결정하는 코드이다. 일반적으로 시스템 호출을 실행할 때 마다 시스템은 현재 실행하고 있는 태스크보다 우선순위가 높은 다른 태스크로 실행권을 넘겨야 할지 결정하기 위해 스케줄러를 동작시킨다. 그림 5와 같이 태스크들은 3가지 상태중 하나의 위치에 놓여진다. 따라서 스케줄러는 어떤 태스크를 다음에 실행시킬지를 태스크에 할당된 우선순위를 보고 준비 상태의 태스크들 중에서 가장 우선순위가 높은 태스크를 실행시키고 다른 태스크들은 준비상태로 전환시킨다.

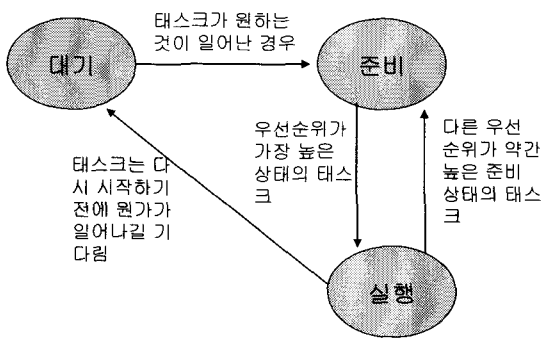


그림 5. 태스크 상태

범용 시스템의 스케줄러는 프로세스의 우선순위를 동적으로 바꾸며 동일한 우선순위 프로세스에 대해서는 시분할 방식으로 선점이 가능하게 만들어져 있다. 즉 비선점형 커널 구조를 채택하고 있어 특정 프로세스가 시스템 호출이나 인터럽트를 통해 커널 내부로 진입한 경우 작업이 끝나기 전까지 CPU를 우선순위가 더 높은 프로세스에 양보할 방법이 없는 것이다.

하지만 RTOS는 태스크가 특정 시간 조건에 맞춰 동작하지 않으면 시스템이 실패할 가능성이 그 만큼 높아진다. 따라서 범용 운영체제에서 제공하는 일반적인 스케줄러 특성만으로는 실시간성을 보장할 수 없으므로 보다 정교한 스케줄러를 탑재할 필요가 있다. 다음은 RTOS가 되기 위해 필요한 기능들을 정리한 내용이다.

첫 번째, RTOS는 태스크 간 스케줄을 위해 고정-우선순위 선점(preemption) 기능을 제공해야 한다. 즉 태스크 우선순위가 상대적으로 바뀌거나 역전될 경우 실시간 특성을 만족 시킬 수 없으므로 태스크나 프로세스에 항상 고정적인 우선순위를 부여할 수 있어야 한다. 그래서 우선순위가 높은 태스크가 낮은 태스크보다 항상 우선시 될 수 있다.

두 번째, 태스크 동기화를 위한 우선순위 상속이나 우선순위 한도 제한과 같은 기능을 제공해야 한다. 만약 이런 기능이 없을 경우에는 우선순위가 낮은 태스크가 우선순위가 높은 태스크에 필요한 자원을 잡고 있을 가능성이 있는데 이를 우선순위 뒤바뀜 현상이라 한다.

예를 들면 우선순위가 가장 낮은 태스크 A와 우선순위가 가장 높은 B, 우선순위가 중간인 C가

있을 때, 태스크 B가 깨어나 작업을 하려고 보니 필요한 자원을 태스크 A가 갖고 있다. 태스크 A가 끝나기 전까지 태스크 B가 자원을 확보할 수 없으므로 태스크 C로 제어권이 넘어간다. 분명히 태스크 B에게 우선권이 부여해야 하는데 순위가 낮은 태스크 A로 인해 태스크 C가 우선순위가 높은 태스크 B를 새치기하는 현상이 발생하게 되는 것이다.

가령 비행기 내에서 엔진화재 감지 장치보다 화장실 흡연 감지 장치가 먼저 동작한다면 어떻게 되겠는지 상상해보면 알 수 있을 것이다. 이 문제의 해결방법은 태스크 A에 태스크 B와 동일한 우선순위를 상속한다고 가정하면, 태스크 B는 우선순위가 태스크 C보다 높아진 태스크 A가 실행하고 놓아준 자원을 원활히 사용해서 태스크 C에 앞설 수 있게 하는 것이다.

세 번째, 커널이 중첩 인터럽트를 지원하면서 인터럽트 서비스에 대한 고정 최대 대기시간을 유지할 수 있어야 한다. 인터럽트 서비스 루틴이 우선순위가 높은 다른 태스크 시간을 무작정 빼앗아 쓸 수 없도록 서비스를 수행함에 있어 한계 시간을 엄수하도록 인터럽트 서비스 루틴을 정밀하게 작성해야 한다. 왜냐하면 인터럽트 서비스 루틴은 우선순위가 낮은 태스크도 얼마든지 호출할 수 있으므로 여기에서 지연 현상이 일어날 경우 우선순위가 높은 태스크가 서비스 종료를 무작정 기다릴 수밖에 없기 때문이다.

끝으로, 커널에 진입했을 때 중요한 태스크가 커널이 확보한 제어권을 필요에 따라 선점할 수 있게 수행 시간을 특정 대기 시간 이내에 되도록 최소화해야 한다. 기타 상세한 스케줄링 기법과 RTOS에 관한 내용은 [8-9]를 참고하길 바란다.

앞으로 임베디드 시스템에서 실시간 운영체제의 숫자가 점차 늘어날 것이다. Handheld PC에서 채용한 Windows-CE는 많은 가전제품에 적용되고 있다. 즉, 무수한 제품에 눈에 보이지 않는 컴퓨터가 들어갈 것이며 가전제품들 사이에 네트워크 기능을 이용해서 제어하는 홈 오토메이션(home automation)시스템도 가능할 것이다. 따라서 실시간 운영체제의 숫자도 현재와는 달리 매우 많이 늘어날 것으로 예상된다.

3.2 파일 시스템

오늘날 멀티미디어 서버나 멀티미디어 데이터를 저장하고 재생하는 DVR(digital video recorder), PVR(personal video recorder), 홈 미디어 서버 등이 시장에서 중요한 위치를 차지하고 있다. 또한 캠코더와 같은 휴대용 영상 기기가 자기 테이프 대신 하드 디스크나 대용량 플래시 메모리(flash memory)와 같은 임의 접근이 가능한 저장장치를 점차 많이 사용하는 추세이다.

하지만 플래시 메모리는 기존 하드 디스크와는 다른 동작 특성을 가지고 있기 때문에 하드 디스크를 직접 대체하기는 어렵다. 즉 응용 프로그램이 하드 디스크와 동일한 개념의 저장장치로 손쉽게 사용하기 위해서는 별도의 하드웨어 또는 소프트웨어가 필수적이다. 특히, 플래시 메모리의 여러 가지 제약 사항을 극복하는 복잡한 알고리즘을 구현하기 위해서는 소프트웨어의 역할이 매우 중요하다. 그러나 국내에서 플래시 메모리 기반 임베디드 소프트웨어 개발에 대한 투자는 매우 미흡한 수준이다.

따라서 플래시 메모리 기반 소프트웨어 중에서 장치들에 대용량의 멀티미디어 정보를 신뢰성이 있으면서도 높은 성능으로 저장하고 재생할 수 있는 파일 시스템이 더욱 필요한 것이다[10].

현재 널리 사용되고 있는 파일 시스템은 리눅스의 표준 파일 시스템인 EXT2와 저널링 파일 시스템들인 EXT3, JFS, XFS, ReiserFS 등이 있

다. 하지만 기존 파일 시스템들은 주로 서버 환경에서 최적화가 되도록 파일 시스템이 설계 및 구현되어 있으며 파일 시스템과 주변 유틸리티 코드를 포함하여 수 만에서 수 십만 라인의 소스 코드로 구현된 매우 복잡한 소프트웨어들이다. 따라서 임베디드 시스템에서는 적절한 유지보수가 어려울 뿐만 아니라 실행 바이너리 저장을 위한 플래시 메모리 공간을 많이 차지함으로써 제품의 비용 상승을 야기시켜 개인용 임베디드 멀티미디어 시스템에는 적합하지 않는다. 이를 위해 임베디드 시스템에서는 파일 저장을 위해 NAND 또는 NOR 형태의 플래시 메모리를 이용한 JFFS2 (Journalling Flash File System 2)[11,12] 등이 개발되어 있고 마이크로소프트에서도 플래시 파일 시스템을 제시하고 있다. 그림 6은 JFFS의 구조를 나타낸다.

다음은 그 밖에 플래시 기반 파일 시스템에서 필요한 기능을 요약한 것이다[13].

- 고성능 FTL(flash translation layer) 상에서 호환성과 신뢰성을 보장하는 플래시 메모리 파일 시스템 개발
- 기존 파일 시스템과 유사한 계층 구조 및 저장 공간 배치 구조를 적용하여 호환성 보장
- 저널링에 기반한 쓰기를 통한 파일 시스템 자동 복구 기능 구현

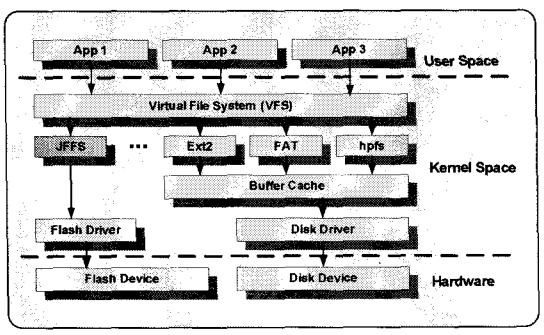


그림 6. JFFS 구조

- 파일 시스템 단편화를 최소화하는 클러스터 할당 정책에 따라 플래시 메모리에 최적화
- 대용량 플래시 메모리에서도 빠르고 메모리 사용량이 적은 주소 사상 기술 개발
-오버헤드가 적은 자유 블록 확보 정책 등 개발

3.3 기타

임베디드 시스템에서는 멀티미디어와 같은 데이터를 저장, 관리, 재생하기 위해 플래시 메모리를 사용하고 있다. 따라서 기타 플래시 메모리 관련 임베디드 소프트웨어 개발이 필요하며 다음과 같이 정리할 수 있다[13].

- 다양한 임베디드 시스템 환경에 최적화할 수 있는 고성능, 저전력, 고신뢰성 FTL 개발
- 플래시 메모리 드라이버 개발
- 대용량 플래시 메모리를 접근하는 고성능, 저전력 컨트롤러 지원 소프트웨어 개발
- 성능향상을 위한 메모리 관리, 저전력이 가능한 소프트웨어 개발
- 플래시 메모리 기반 가상메모리 시스템 개발
- 임베디드 DBMS 개발
- 플래시 메모리의 쓰기 횟수나 소거 횟수 감소를 고려한 버퍼 교체 알고리즘과 버퍼 관리자 개발
- 메타 데이터 시스템 개발 등

4. 결 론

초기의 임베디드 시스템은 비교적 단순하여 운영체제 없이 사람이 순차적인 프로그램을 작성해서 수행하도록 했다. 하지만 최근 들어 대용량 멀티미디어 정보를 저장, 관리, 재생 등의 복합적인 처리를 요구하는 임베디드 시스템이 늘어나면서 그와 관련된 기술들이 필요하게 되었다. 대표적인

것이 운영체제의 개념이며 임베디드 시스템의 특성상 실시간 이라는 요소를 만족해야 했다. 따라서 실시간 운영체제가 임베디드 시스템에 도입된 것이다. 그밖에 범용 시스템에서 프로세스 생성 및 관리, 메모리 관리, 파일 시스템 등이 필히 요구되는 기능이듯이 멀티미디어를 처리할 임베디드 시스템에서도 핵심 기술로 고려되어야 한다. 본 논문에서는 임베디드 멀티미디어 시스템을 위해 요구되는 여러 기술들을 살펴보았고 결국 해결책은 관련된 임베디드 소프트웨어가 개발되어야 함을 알 수 있다. 이를 위해 많은 임베디드 소프웨어 국산화가 하루속히 이루어지길 기대해본다.

참 고 문 헌

[1] David Corman and James Paunicka, "Industrial Challenges in the Composition of Embedded Systems," *LNCS* 4888, pp. 97-110, 2007.

[2] 박재호, *IT EXPERT 임베디드 리눅스*, 한빛미디어, 2002.

[3] Janos Sztipanovits and Gabor Karsai, "Embedded Software: Challenges and Opportunities," *LNCS* 2211, pp. 403-415, 2001.

[4] 김상헌, 정재영, 이동명, *Embedded Linux 기반의 로봇 설계 & 제작*, 영진닷컴, 2005.

[5] 성원호, *임베디드 시스템 펌웨어 분석*, 에이콘출판, 2002.

[6] 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.

[7] Liu. and Jane W.S., *Real-Time systems*, Prentice-Hall, pp. 26-34, 2000.

[8] J. A. Stankovic, M. Spuri, K. Ramamritham and G. Buttazzo, *Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms*, Kluwer Academic Publishers, 0-7923-8269-2, 1998.

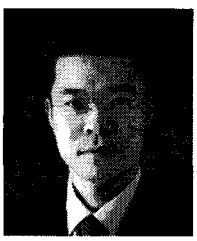
[9] H. Chetto and M.Chetto, "Some Results of the Earliest Deadline Scheduling Algorithm," *IEEE Trans on Sofrware Eng.*, pp. 1216-1269, 1989.

[10] 이민석, "임베디드 멀티미디어 시스템을 위한 파일 시스템의 설계 및 구현," *한국데이타베이스학회 논문지: 정보기술과 데이타베이스* 제12권 제1호, pp. 125-140, 2005.

[11] D. Woodhouse, "JFFS: The Journalling Flash File System," *Preceeding of Ottawa Linux Symposium*, 2001.

[12] Aleph One, "YAFFS: Yet another flash filing system," http://www.aleph1.co.kr/yaffs/, 2002.

[13] 곽종철, 민상렬, 박장석, "Flash Memory 기반 임베디드 멀티미디어 소프트웨어 기술 개발," *정보과학회지* 제25권 제6호, pp. 5-10, 2007.



조 수 현

- 2000년 금오공과대학교 컴퓨터공학과(공학사)
- 2002년 금오공과대학교 컴퓨터공학과(공학석사)
- 2006년 금오공과대학교 컴퓨터공학과(공학박사)
- 2006년~현재 금오공과대학교 컴퓨터공학부 계약교수
- 관심분야 : 임베디드 시스템, USN, 그리드 컴퓨팅 등