자동차 특성을 만족하는 실시간 스케줄링 알고리즘에 관한 연구

장승주*

A Study of Real-Time Scheduling Algorithms for Automotive System

Seung-Ju Jang*

요 약

본 논문은 자동차용 임베디드 시스템에 탑재되는 운영체제의 실시간 스케줄링 알고리즘에 대해서 개선된 알고리즘을 제안한다. 기존 OSEK OS의 스케줄링 알고리즘에서 큰 차이를 보이는 부분은 16개의 우선순위 대기 큐를 4개의 우선순위 대기 큐로 줄였다. 그리고 대기 큐에서 FIFO 알고리즘을 사용하지만 제안하는 스케줄링 알고리즘은 EDF 알고리즘을 사용하여 실시간성을 좀 더 높였다. 제안한 스케줄링 알고리즘에 대해서 간단한 실험을 수행하였다. 실험 결과 FIFO 알고리즘을 사용한 기존 OSEK OS의 경우 평균 26.29ms이고, 제안한 EDF 알고리즘을 사용한 경우 평균 26.13ms로 제안한 EDF 알고리즘을 사용한 경우가 0.16ms 정도 빠름을 알 수 있다. FIFO 알고리즘을 사용한 기존 OSEK OS의 스케줄링 알고리즘 보다 제안하는 EDF 스케줄링 알고리즘을 사용한 스케줄링 알고리즘이 우수함을 알 수 있다.

ABSTRACT

Recently, the automobile industry is going through drastic environmental changes. The relative importance of information technology rapidly developed so far is getting heavier as it is grafted to electrical and electronic areas among all other automobile-related technologies. In this paper, an improved algorithm from the real-time scheduling algorithm of operation system which is loaded in embedded system will be presented. The number of wait-queue of priority was reduced from 16 to 4 in the parts where wide differences were shown in scheduling algorithm of the existing OSEK OS. While the FIFO algorithm was used in wait-queue, the EDF algorithm was applied to the proposed scheduling algorithm, which more improved the real-timeness. Also a simple experiment on the proposed scheduling algorithm was conducted.

키워드

Automobile Embedded System, Real-time Scheduling, Performance Enhancement

* 동의대학교 컴퓨터공학과

접수일자 2009. 02. 05
심사완료일자 2009. 05. 25

Ⅰ. 서 론

최근 자동차 산업계는 급격한 환경적 변화를 겪고 있다. 기업 간의 기술 경쟁이 심화되고 있다. 한편 세계적으로 각국의 환경오염 규제가 강화되고 있으며, 차량 안전 기준도 점점 엄격해지고 있다. 이와는 별도로 최근까지 급격히 발전되어 온 IT기술이 자동차 관련 기술 분야들 중에서 전기 전자 분야에 접목되면서 비중이 높아지고 있다. 전장 부품의 적용은 자동차 안전 장치에서 부터 쾌적한 운행을 위한 장치에까지 적용되고 있다. 자동차 산업은 임베디드 소프트웨어 산업의 경쟁력에 의하여 운명이 결정될 가능성이 높다 [1-3].

본 논문에서 제안하는 자동차 특성을 만족하는 실시간 스케줄링 알고리즘은 대부분의 전장 부품(ECU : Electronic Control Unit)에서 사용이 된다. 자동차용 전장 부품에서 실시간 기능의 만족은 아주 중요하다.

자동차 임베디드 운영체제 중 널리 쓰이는 운영체제는 OSEK, Windows CE, 임베디드 리눅스 등이 대표적이다. 본 논문에서는 OSEK 운영체제의 실시간 스케줄링 알고리즘에 EDF 알고리즘을 추가하여 자동차 환경에 적합한 실시간성을 높였다. 기존의 OSEK 운영체제 스케줄링 알고리즘이 단순하게 FIFO 방식을 사용하는데 본 논문에서는 FIFO 방식 대신에 EDF방식을 사용하여 실시간 기능을 강화하였다. 제안한 스케줄링 알고리즘에 대해서 간단한 실험을 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서 본 연구와 관련 있는 운영체제를 설명하고, 3장에서는 실시간 스케줄링 알고리즘에 대해 기술하였다. 4장에서는 개선된 스케줄링 알고리즘을 설계한다. 5장에서는 개선된 스케줄링 알고리즘에 대하여 실험을 수행한다. 6장에서는 본 논문의 결론으로 구성한다.

Ⅱ. 관련연구

임베디드 시스템은 제어 장비, 컴퓨터만큼이나 오랜 역사를 가지고 있다. 특히, 통신 분야에서는 1960년대 후반 전기, 기계식 전화 교환기와 '내장 프로그램 제어' 시스템을 제어하는 데 이러한 시스템이 쓰였다. 그러나 컴파일러에 버그가 많고 쓸 만한 디버거가 없었기 때문에 소프트웨어는 항상 어셈블리 언어 또는 매크로 언어로 작성되었다.

1970년대 후반, 임베디드 시스템의 운영체제에서 표준화된 대량생산이 등장하였다. 그러나 주로 어셈블리 언어로 제작되었고 마이크로프로세서에서만 사용할 수 있었기 때문에 마이크로프로세서가 구식이 되면 운영체제도 구식이 되는 문제점을 가지고 있었다. C 언어가 나오면서 운영체제를 효율적이면서 안정적이고 포터블한 방법으로 작성하게 되었고, 이는 마이크로프로세서가 구식이 되어도 소프트웨어에 대한 투자를 보호할 수 있었다. 이런 장점 때문에 결국 C로 작성된 운영체제는 표준이 되고 오늘날까지도 남아 있다.

임베디드 OS는 실시간 운영체제를 포함하는 폭넓은 분야이다. 이러한 실시간 운영체제로 구성된 임베디드 시스템은 정확한 기능을 요구하는 시스템에 활용된다. RTOS는 속도가 빠른 운영체제라고 생각하지만 실행 속도와는 별 관련이 없다. 그러나 대부분의 범용 운영체제보다는 RTOS가 가볍기 때문에 동일한 조건이라면 훨씬 좋은 성능을 낼 수 있다. 운영체제의 각종 동작이 어떤 정해진 시간 안에 이뤄진다면, 그 시간이 아무리 길어도 실시간 운영체제라고 할 수 있다.

RTOS는 표준 하드웨어 환경이 정해져 있지 않다는 점이 윈도우나 유닉스 등이 범용 OS와 커다란 차이이며, 이는 곧 응용 프로그램뿐 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.

실시간 운영체제 활용에 있어서 임베디드 시스템에 대한 정확한 이해가 요구되고 이러한 적용분야가 확대되고 있다. 앞으로 미래에는 임베디드 시스템에 요구되는 하드웨어 기술이 더욱 첨단화될 것이고 임베디드 시스템이 범용화 되는 시대가 올 것이다. RTOS의 특수한 처리능력을 가진 운영체제에 기반을 둔 임베디드 시스템에 대한 연구와 적용 기술의 확보에 노력해야 할 것이다.

임베디드 시스템에서의 운영체제는 크게 실시간 운영체제와 비실시간 운영체제로 나눌 수가 있다. 현재까지 상용화된 운영체제에서 대표적인 실시간 운영체제는 아래 표 1.과 같다.

표 1. 실시간 운영체제와 홈페이지
Table 1. Real-time Operating System and Realted Home Page

운영체제	홈페이지 주소
VxWorks	www.windriver.com
pSOS	www.windriver.com
VRTX	www.mento.com
QNX	www.qnx.com
OSE	www.ose.com
Nucleus	www.atinudclus.com
MC/OSII	www.mcos-ii.com

표 1.에 소개된 실시간 운영체제는 공통적으로 특정 태스크를 중단시키고 다른 태스크를 수행할 수 있도록 하는 선점형(preempted) 멀티태스킹을 지원하며, POSIX API를 지원한다. 각 태스크들은 우선순위를 가지고 있어 높은 우선순위의 태스크가 먼저 실행된다. 실시간 운영체제는 보통 커널 모드와 사용자 모드가 있고 시스템 콜에 의해 각 모드에 대한 독립성을 보장한다. 또한, 통합개발환경과 디버깅 툴을 제공하여 개발자들이 쉽게 소프트웨어를 개발할 수 있도록 한다.

문제는 이들 실시간 운영체제들이 대부분 고가여서 시스템 개발 비용이 상승하고, 양산시 라이선스 비용이 제품원가에 차지하는 비중이 커진다는 점이다.

기존의 OSEK 운영체제에서 사용하고 있는 자동차용 알고리즘은 다음과 같다. OSEK OS 는 태스크의 수행 상태를 관리하거나 스케줄러를 통해 태스크의 수행 순서를 조정하는 역할을 한다. OSEK OS 의 태스크는 running, ready, waiting, suspended 의 네 가지 상태를 가질 수 있다. OSEK OS 는 선점형 실시간 운영체제이다. 이는 한 태스크가 수행 중이더라도, 언제든지 필요에 따라 다른 태스크가 현재 수행중인 태스크를 선점하고 즉시 수행될 수 있다는 뜻이다. 이를 위해 OSEK OS 는 각 태스크마다 별도의 메모리 영역을 할당하고 이 영역에 각 태스크의 상태(레지스터 값 등)를 저장하여 여러 태스크가 CPU 하나를 공유할 수 있도록 한다.

2.1 실시간 스케줄링 알고리즘

RTOS 시스템은 시스템 동작의 정확성이 논리적 정확성뿐만 아니라 시간적 정확성에서 좌우되는 시스템으로 정의 된다. 시스템의 수행 결과가 기능적으로 정확해야 할 뿐만 아니라, 결과가 도출되는 시간 역시 주어진 제약 조건을 만족해야 한다. RTOS 시스템은 주어진 시간의 제한에 따라 크게 3가지로 분류 할 수 있다.

첫째, Hard RTOS 시스템, 시스템이 주어진 종료 기한을 만족시키지 못한 경우에 막대한 재산적 손실이나 인명의 피해를 주는 시스템을 말한다.

둘째, Soft RTOS 시스템, 온라인 시스템과 같이 시간적인 제약 조건을 만족시키지 못하더라도 첫째의 경우처럼 치명적이지 않고 종료 기한을 넘겨 수행을 마쳐도 계산의 결과가 의미 있는 시스템을 말한다 [4-7].

마지막으로 Firm RTOS 시스템 첫째와 둘째의 중간 형태로 종료 기한을 넘겨 수행을 마치는 것은 무의미하지만 시간초과에 대한 손실이 치명적이지 않은 시스템을 말한다.

2.1.1 RM

비율 단조 스케줄링(rate-monotonic scheduling, 줄여서 RMS)은 실시간 시스템에 적용하기 위해 리우(Liu)와 래이랜드(Layland)가 제안한 스케줄링 정책이다. 비율 단조 분석(rate-monotonic analysis, 줄여서 RMA)는 RMS의 배경이 되는 이론으로서, 시간 당 CPU 사용률을 계산하여 프로세스들을 이상 없이 수행할 수 있는지를 알아보는 일을 말한다[4-6].

RMS는 프로세스에 부여하는 우선순위의 변동이 없기 때문에, 정적 스케줄링 정책이라고 할 수 있다. 그렇지만 상당히 효율적이라서 아직도 널리 사용되고 있다. RMS를 사용하는 운영체제는 일반적으로 선점형이고, 응답시간에 대해 결정적(deterministic)인 특징이 있다. RMA 이론에서는 시스템을 비교적 단순한 모델로 가정한다. 모든 프로세스는 단일 CPU에서 주기적으로 구동된다.

2.1.2 DM

DM 스케줄링 알고리즘은 Deadline에 따라 우선순위를 시스템 수행 전에 결정짓는 정적 스케줄링 알고리즘이다. EDF 알고리즘과 같이 Deadline이 짧은 태스크가 높은 우선순위를 갖는다.

2.1.3 EDF(Earliest Deadline First)

m개의 동일한 처리기(Processor)상에서 주기 태스크에 대해서 태스크 집합을 n개의 독립적인 주기 태스크

(ri)로 구성하고, 주기 태스크는 ri = (Ci, Pi)로 정의한다. 정의에서 Ci는 최악의 수행시간을 나타내고, Pi는 주기를 나타낸다고 가정한다.

모든 태스크는 동시에 시작하며 각 태스크의 상대적 마감시간(Relative deadline)은 주기와 같다고 가정한다. 이와 같이 가정을 하고 고정 우선순위의 대표적인 알고리즘들을 살펴보면, EDF 알고리즘은 마감시간이 가장 빠른 작업에게 가장 높은 우선순위를 부여하는 방식이다[48, 49, 50].

EDF는 효율적 구현이 가능하며 전체 선점횟수가 전체 작업 수의 2배를 넘지 않는다. 그러나 다중처리기 상에서 EDF의 스케줄 보장 이용률은 매우 낮을 수 있다. EDF-US의 경우 태스크들을 이용률이 높은 태스크 그룹과 이용률이 낮은 태스크 그룹으로 나눈다.

이용률이 m/(2m-1)보다 높은 태스크에 의해 생성된 작업에게는 최고의 우선순위를 부여하고, 이용률이 m/(2m-1)보다 낮은 태스크에 의해 생성된 작업에게는 EDF 알고리즘에 의해 우선순위를 부여한다[7-9].

이 알고리즘은 전체 이용률이 m/(2m-1) 이하인 모든 태스크 집합을 스케줄 할 수 있다. fpEDF 알고리즘은 이용률이 1/2을 넘는 태스크에 의해 생성된 작업 중 이용률이 가장 높은 태스크의 작업부터 최대 m-1개의 작업까지 최고의 우선순위를 부여하고, 나머지 작업들은 EDF 알고리즘에 의해 우선순위를 부여한다.

fpEDF 알고리즘은 전체 이용률이 (m+1)/2 이하인 모든 태스크 집합을 스케줄 할 수 있다.

EDF 알고리즘은 마감시간이 임박한 프로세스를 우선적으로 선택하여 CPU를 할당하게끔 하는 알고리즘으로 마감시간이 임박한 프로세스의 우선순위를 보다 덜한 프로세스의 우선순위 보다 높게 하여줌으로 실시간 프로세스간의 우선순위를 관리하고 이렇게 할당 받은 우선순위로 큐를 관리한다[10-14].

Ⅲ. 개선된 스케줄링 알고리즘 설계

본 논문은 OSEK 운영체제 스케줄링 알고리즘을 변형하여 자동차 환경에 적합한 실시간 스케줄링 알고리즘을 제안한다.

OSEK OS는 선점형 실시간 운영체제이다. 이는 한 태스크가 수행 중이더라도, 언제든지 필요에 따라 다른 태스크가 현재 수행 중인 태스크를 선점하고 즉시 수행될 수 있다는 뜻이다.

OSEK OS의 기능 중 태스크 관리 기능의 경우를 보면, 태스크에 대한 전반적인 관리를 해준다. 태스크의 실행, 대기, 종료 등을 제어 할 수 있는 스케줄링 알고리즘이 이 부분에 포함되어 있다.

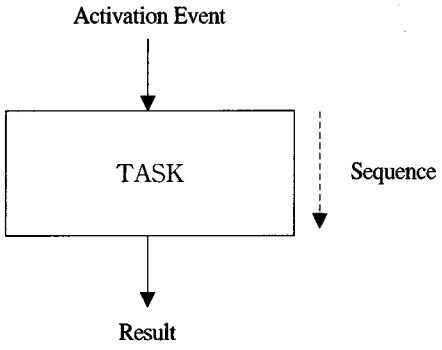

그림 1. 순차 실행
Fig. 1 Sequence Execution

그림 1.과 같이 태스크는 입력에 대한 연속적인 수행으로 출구로 결과를 전달하는 순차 실행 방식을 택하고 있다. 태스크 관리 기능은 태스크들이 어떠한 입력에 대해 순차적으로 실행을 하여 출구로 결과를 전달하게 한다.

태스크 내에서의 반복은 허용하나, 최악의 실행 시간을 외부에서 알 수 있을 정도의 반복이 허용된다. 무한 반복으로 인하여 최악의 실행 시간을 외부에서 알 수 없게 되는 것은 허용하지 않고 있다.

OSEK OS의 기존 스케줄러는 ready 상태에 있는 태스크들 중에 다음에 수행할 한 태스크를 선택하는 역할을 한다. OSEK OS 는 16 단계의 우선순위를 지원하는 FIFO 스케줄러를 사용한다. 각 태스크는 0 부터 15까지의 우선순위 단계를 가질 수 있으며, 우선순위 단계가 높은 태스크가 먼저 선택된다. 같은 우선순위 단계의 태스크가 여러 개 존재할 경우, 가장 오래 전에 ready 상태가 된 태스크가 선택된다.

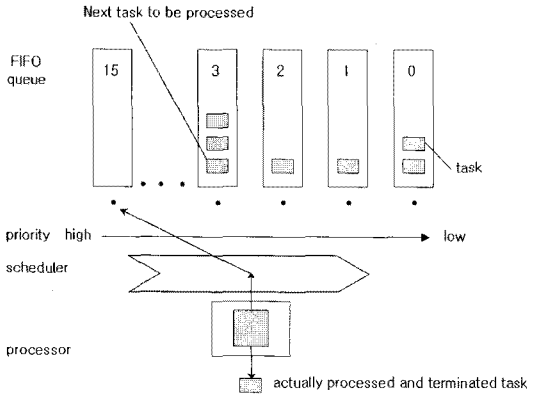

그림 2. 기존의 OSEK OS Scheduler
Fig. 2 OSEK OS Scheduler

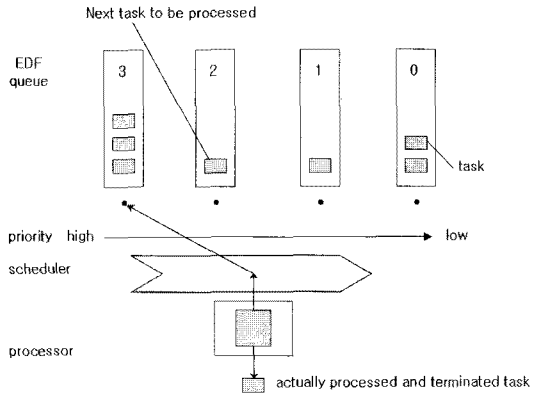

그림 3. 제안하는 스케줄러
Fig. 3 Proposed Scheduler

그림 2는 OSEK OS 의 스케줄러가 작동하는 방식을 보여주고 있다. 스케줄러는 16 단계의 FIFO 큐를 가지고 있으며, 각 큐에는 해당하는 우선순위의 ready 태스크가 존재한다. 스케줄러는 이들 큐를 순차적으로 검사하면서 수행할 태스크를 선택한다.

제안하는 스케줄러는 기존의 OSEK 스케줄러가 16 단계의 우선순위를 지원하는 것을 4 단계의 우선순위로 축소하였다. 16 단계의 우선순위는 선택의 폭을 넓게 하고는 있지만 대기 큐 공간을 많이 차지하고 있다.

제안하는 스케줄러는 대기 큐 내에서 EDF 알고리즘을 사용한다. EDF 알고리즘은 마감시간이 촉박한 태스크를 먼저 실행하는 스케줄러로서 보통 제한된 수의 우선순위들을 제공한다. 만약 우선순위의 단계가 많다면 새로운 태스크에게 필요한 순위를 사용할 수 있을 때까지 제공한다면 다른 태스크들의 순위를 재조정해야 한다. 최악의 경우에는 모든 태스크들의 순위가 재조정되어야 하고 엄청난 오버헤드를 발생시킬 수 있다. 이런 문제점을 방지하기 위해 보통 EDF 알고리즘을 사용하는 일반적인 시스템에서는 제한된 우선순위를 제공한다.

본 논문에서도 EDF 알고리즘을 사용함으로 우선순위의 단계를 줄일 필요가 있어서 16 단계의 우선순위를 4단계로 줄여서 사용한다.

제안하는 스케줄러는 그림 3.과 같이 기존의 스케줄러와 동일하게 대기 큐가 각각의 우선순위를 나타내지만 모든 대기 큐에 FIFO 스케줄러가 아닌 EDF 알고리즘을 사용한다면 좋은 성능을 보일 것으로 예상이 된다. 모든 우선순위 대기 큐에 EDF 알고리즘을 적용하는 것이 아니라 중/하위의 우선순위 대기 큐에 적용하는 것이 좋은 성능을 보일 것이다.

Ⅳ. 실험

기존의 OSEK OS의 스케줄링 알고리즘은 16 단계의 우선순위를 갖고 있고, 각 우선순위마다 하나의 대기 큐를 가지고 있다. 0 단계부터 15 단계까지의 우선순위 중 단계가 높을수록 우선순위가 높다. 대기 큐에 태스크가 여러 개 존재할 경우, 가장 오래 전에 대기 큐에 들어온 태스크가 먼저 실행이 되는 FIFO 스케줄러를 사용한다.

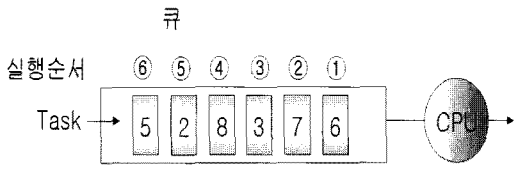

그림 4. 기존 OSEK OS에서 대기 큐 태스크 처리 순서
Fig. 4 Ready Queue Task Processing in OSEK OS

그림 4는 기존 OSEK OS에서 대기 큐에 있는 태스크들을 처리하는 순서를 보여준다. 먼저 들어온 태스크를 먼저 처리하는 일반적인 큐 자료구조를 사용한다.

제안하는 스케줄링 알고리즘은 4 단계의 우선순위를 갖고 있고, 각 우선순위마다 하나의 대기 큐를 가진다. 0 단계부터 3 단계까지의 우선순위 중 기존 OSEK OS와 동일하게 단계가 높을수록 우선순위가 높다. 대기 큐에 태스크가 여러 개 존재할 경우, 마감시간이 임박한 즉, 빨리 처리해야 하는 태스크를 먼저 실행하는 EDF 스케줄러를 사용한다.

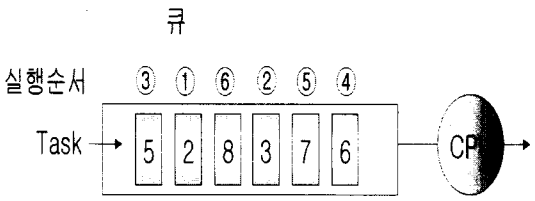

그림 5. 제안하는 대기 큐 태스크 처리 순서
Fig. 5 Proposed Ready Queue Processing

그림 5는 제안하는 스케줄링 알고리즘의 대기 큐에 있는 태스크들을 처리하는 순서를 보여준다. 마감시간이 짧은 중간의 태스크를 먼저처리 할 수 있게 하고 있다.

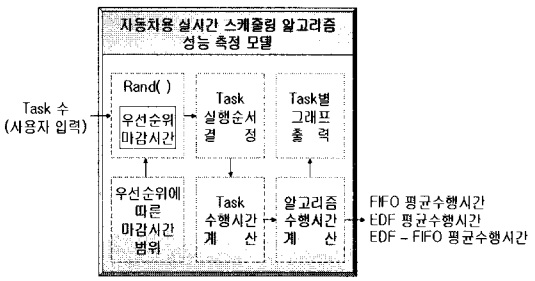

그림 6. '자동차용 실시간 스케줄링 알고리즘 성능 측정 모델' 동작 과정
Fig. 6 Performance Evaluation Model for Automobile Real-Time Scheduling Algorithm

그림 6은 '자동차용 실시간 스케줄링 알고리즘 성능 측정 모델'의 동작 과정을 보여준다. 사용자가 Task 수를 입력하면 Rand() 함수로 Task의 우선순위를 임의로 결정을 한다. 우선순위가 결정된 Task는 성능 측정 모델에 정의되어 있는 우선순위에 따른 마감시간 범위 내에서 마감시간을 임의로 결정한다.

Task의 우선순위와 마감시간이 결정되면 스케줄링 알고리즘을 통하여 Task의 실행순서가 결정되어 Task들의 수행시간을 계산한다. 계산된 결과를 그래프로 출력 하고, 사용자에게 알고리즘별 평균수행 시간을 보여준다.

실험을 통한 성능 측정을 위하여 엑셀의 VBA(Visual Basic for Application) 프로그램을 이용한다. 아래 그림 7은 엑셀의 VBA를 이용한 성능 측정 프로그램의 초기화면이다.

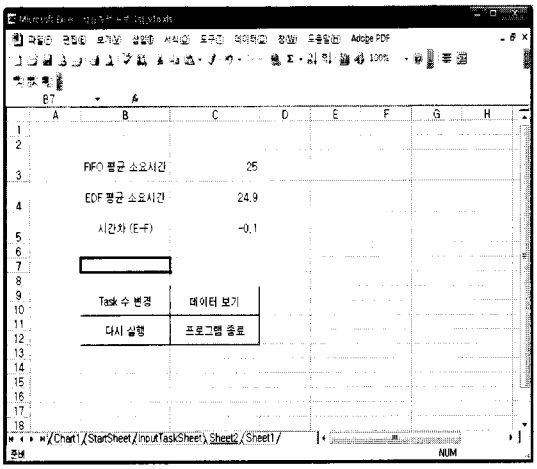

그림 7. 성능측정 결과 시트
Fig. 7 Performance Test Result Sheet

그림 7은 성능 측정 결과 시트이다. 'FIFO 평균 소요시간' 항목은 대기 큐가 FIFO 알고리즘을 사용하였을 때, 평균적으로 Task가 수행하는데 소요되는 시간을 나타낸다. 'EDF 평균 소요시간' 항목은 대기 큐에 본 논문에서 제안한 EDF 알고리즘을 사용하였을 때, 평균적으로 Task가 수행하는데 소요되는 시간을 나타낸다.

그림 7에 의하면 FIFO 알고리즘을 사용한 기존의 OSEK 실시간 스케줄링 알고리즘은 25ms의 시간이 평균적으로 소요되고, EDF 알고리즘을 사용한 제안한 실시간 스케줄링 알고리즘은 24.9ms의 시간이 평균적으로 소요된다.

표 2. '자동차용 실시간 스케줄링 알고리즘 성능 측정 모델'을 이용한 테스트 결과(1번씩 수행)
Table 2. Test Result using Performance Evaluation Model for Automobile Real-Time Scheduling Algorithm(one time execution)

(단위 : ms)

순번	FIFO (OSEK OS 스케줄링 알고리즘)	EDF	시간차 (EDF - FIFO)
1	25.8	25.775	-0.025
2	26.35	26.45	0.1
3	26.15	26	-0.15
4	25.15	25.575	0.425
5	26.95	26.425	-0.525
6	26.7	26.525	-0.175
7	25.85	25.45	-0.4
8	25.825	25.45	-0.375
9	26.4	25.8	-0.6
10	26.925	26.75	-0.715
평균	26.21	26.02	-0.24

표 2는 '자동차용 실시간 스케줄링 알고리즘 성능 측정 모델'을 이용하여 FIFO 알고리즘(OSEK OS 스케줄링 알고리즘)을 사용하고 있는 기존의 실시간 스케줄링 알고리즘(FIFO라 표기 되어 있음)과 본 논문에서 제안한 EDF 알고리즘을 사용하는 실시간 스케줄링 알고리즘(EDF라 표기 되어 있음)의 성능을 테스트한 결과이다.

V. 결론

본 논문은 임베디드 시스템 특히 자동차용 임베디드 시스템의 운영체제로 사용되는 OSEK OS에서 사용하는 실시간 스케줄링 알고리즘의 성능 개선에 대해 제안하였다. 제안한 알고리즘은 기존의 자동차용 운영체제에서 사용하는 알고리즘을 개선한 알고리즘이다. OSEK OS에서 사용하는 기존 알고리즘에서 우선순위 큐의 수와 각 큐에서 동작하는 FIFO 알고리즘을 EDF 알고리즘으로 개선시켰다. FIFO 알고리즘은 실시간 시스템에 사용하기엔 부적절한 알고리즘이다. 실시간 시스템에서는 주어진 시간 내에 일을 처리하여 결과를 출력해주어야 한다. 주어진 시간 내에 결과를 출력하는 Hard Real-Time 시스템에 사용하기에는 EDF 알고리즘이 적합하다. 본 논문에서 제안하는 알고리즘은 실시간성을 요구하는 자동차 시스템에 적합하도록 EDF 알고리즘을 대기-큐 부분에 설계하였다. 설계된 내용을 바탕으로 개선된 실시간 스케줄링 알고리즘은 기존의 실시간 스케줄링 알고리즘 보다 성능이 뛰어남을 간단한 '자동차용 실시간 스케줄링 알고리즘 성능 측정 모델' 프로그램을 통해 알아보았다.

'자동차용 실시간 스케줄링 알고리즘 성능 측정 모델' 실험 결과 제안하는 실시간 스케줄링 알고리즘이 26.13ms로 기존의 실시간 스케줄링 알고리즘의 26.29ms보다 0.16ms만큼 빨리 처리 하여 성능이 좋음을 알 수 있다.

참고문헌

[1] 장승주, 권오훈, "자동차용 임베디드 운영체제 기술 동향", 주간기술동향 통권 1311호, 2007.8.

[2] 홍성수, 박지용, 유우석, "OSEK와 AUTOSAR를 중심으로 본 차량용 OS와 미들웨어 기술 동향", 한국정밀공학회지 제21권 제00호, 2006.

[3] OSEK, "OSEK/VDX Time-Triggered Operation System", OSEK, Jul. 2001.

[4] OSEK, "OSEK/VDX Operation System", OSEK, Feb. 2005.

[5] 조문행, "시간 결정성을 보장하는 실시간 태스크 스케줄링", 충남대학교, 2006.11

[6] 박윤미, "실시간 운영체제를 위한 태스크 스케줄링의 설계 및 구현", 한국정보과학회, 2003

[7] 임베디드 시스템을 위한 소규모 Real-Time OS 설계, 이광명, 영남대학교 석사학위논문, 2003

[8] 김지훈, "임베디드 시스템을 위한 실시간 스케줄링 알고리즘 기법", 영남대학교 석사학위논문 ,2005

[9] 박현선, "MicroC/OS-II를 위한 실시간 스케줄링 구현", 단국대학교 석사학위논문, 2002

[10] 이두원, "실시간 운영체제와 임베디드 시스템", 데이터베이스월드, 2000.01

[11] 유병석, "임베디드 시스템을 위한 표준형 RTOS 개

발", Jounal of Korean Electronics, 2005
[12] 한대만, 이종대, 전병욱, 구용완, "임베디드 시스템을 위한 실시간 스케줄링 알고리즘 연구", 한국인터넷정보학회 제7권 제2호, pp.203-206, 2006.
[13] J.Y.T Leung, M. L. Merrill, "A Note on Preemptive Scheduling of Periodic Real-Time Tasks", Information Processing Letters, pp.115, Nov, 1980.
[14] 박영환, "임베디드 스시템 & Embedded 리눅스", 사이텍미디어, 2002
[15] 박윤미 외, "임베디드 리눅스 시스템 설계 및 구현", 정보과학회 2003년 추계 학술대회, 제30권, 제1호, Apr. 2003.

저자소개

장승주(Seung-Ju Jang)

1985년 부산대학교 계산통계학 (전산학) 학사
1991년 부산대학교 계산통계학 (전산학) 석사
1996년 부산대학교 컴퓨터공학과 박사
1987년~1996년 한국전자통신연구원 시스템 S/W 연구실
1993년~1996년 부산대학교 시간강사
2000년~2002년 University of Missouri at Kansas City, visiting professor
1996년~현재 동의대학교 컴퓨터공학과 교수
※관심분야 : 운영체제, 임베디드 시스템 운영체제, 분산시스템, 시스템 보안