사례 발표

임베디드 실시간 시스템 개발 교육 과정

정기훈*, 김도훈*, 박성호**, 강순주***

● 목 차 ●

1. 서 론

내장형 실시간 시스템은 제한된 응답시간을 보장하기 위한 운영 프로그램이 탑재된 것으로, 인터페이스 기기를 통해 물리적인 장치와 직접적으로 정보를 주고 받는다[5]. 즉, 대부분의 실시간 시스템은 센서나 기타 장치의 입력을 처리하고 평가한 후, 그 결과에 따라 허용된 시간 내에 엑츄에이터 등의 출력 장치를 움직인다. 이러한 실시간 시스템 개발의 복잡성은 현장의 엔지니어들에게 하드웨어 소프트웨어 동시 설계 능력, 시스템 소프트웨어에 대한 폭넓은 이해, 그리고 소프트웨어 공학론을 기반으로 한 시스템 설계 및 구현 능력 등 매우 전문적이고 체계적인 기술을 요구한다.

그러나 이러한 전문 인력 양성을 위한 체계적인 교육 과정을 구축하고 실용적인 교육을 하기는 매우 어려운 실정이다. 실시간 시스템은 기본적으로 타켓 시스템(예: 항공기, 발전소, 자동차 등)에 내장되어 동작하는 시스템 임에도 불구하고 그 타켓 시스템을 학교 실험실에서 접근하기가 용이하지 않고, 모형이 있다 하여도 다양한 종류를 구비하거나 재사용하기가 곤란하며, 하드웨어 소프트웨어 동시 설계, 실시간 운영체제 기반의 디바이스 드라이버 제작, 그리고 소프트웨어 공학적 설계 및 구현 방법론 등 관련 이론 및 기술들을 체계적으로 집약하여 교육하기가 쉽지 않기 때문이다.

본 논문에서는 그러한 문제를 해결할 수 있는 실시간 시스템 교과에 대한 실용적인 교육을 위한 실험 도구와 그것을 이용한 실습 과정을 제안하고자 한다.

2. 실시간 시스템 설계 방법론 학습을 위한 요구 사항

실시간 시스템을 제작하는 실습 과정은 다음과 같은 요구사항을 충족시켜야 한다.

- 재사용이 가능한 타켓 시스템 모델 구축 도구: 실제의 타켓 시스템 (예: 로봇, 자동차, 컨베이어 벨트 등)을 모사할 수 있는 도구가 있어야 하며, 이 도구는 다양한 타켓 시스템을 제작할 수 있도록 해체 및 재사용이 가능해야 한다.

* 경북대학교 전자공학과 석사과정
** 경북대학교 전자공학과 박사과정
*** 경북대학교 전자전기컴퓨터학부 정보통신전공 조교수

- 하드웨어, 소프트웨어 동시 설계:
  하드웨어, 소프트웨어 동시 작업이 필수인 실시간 시스템을 실습한다면, 되도록 하드웨어와 소프트웨어의 동시 설계 및 개발이 실습 과정에서 이루어져야 한다.
- 운영체제 및 디바이스 드라이버:
  실시간 시스템 실습인 만큼 실시간 운영체제를 바탕으로 디바이스 드라이버를 제작하고 응용 프로그램까지 구현할 수 있어야 하겠다.
- 응용 프로그램 개발 방법론 제시:
  응용 프로그램 개발 과정에서는 소프트웨어 공학적인 접근을 통해 사전에 문제를 분석하고, 프로그램의 구조를 디자인하여 안정성을 검증하는 과정을 거쳐야 하겠다.

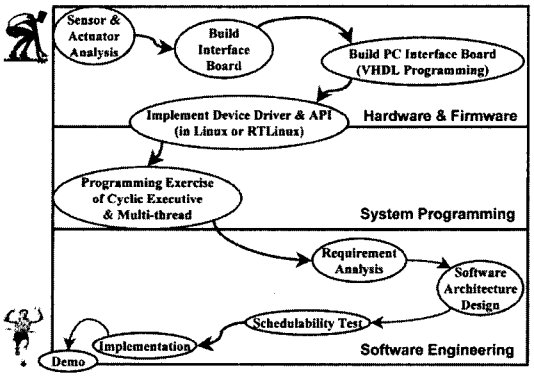
3. 교육 도구 개발의 목표

위와 같은 요구사항을 만족시키기 위해 다음과 같은 목표를 설정하여 실시간 시스템 교육 및 실험 도구를 개발하였다.

- 타켓 시스템 구축 도구:
  실시간 시스템의 특성에 맞춰 다양한 형태의 모델을 제작할 수 있고, 해체 및 재사용이 가능한 센서와 엑츄에이터가 포함된 Lego 블록[8]을 채택하였다. 학생들이 각자의 독창적인 아이디어로 다양한 타켓 모델을 Lego블럭으로 만들 수 있도록 한다.
- 하드웨어 제어 장치:
  FPGA 칩과 기타 TTL칩을 활용해 학생들이 직접 PC용 ISA 기반의 제어 보드와 Lego 센서와 엑츄에이터를 위한 I/O 인터페이스 보드를 제작하도록 한다. 특히 FPGA칩의 기능은 VHDL과 같은 HDL 언어로 구현하여 하드웨어와 소프트웨어의 동시 설계를 경험한다.
- 실시간 운영체제 및 디바이스 드라이버:
  소스가 공개되어 있으면서 실시간성이 우수한 RTLinux[10]를 시스템 운영체제로 사용하고, GNU 개발환경을 토대로 응용 소프트웨어를 개발한다. 또한 학생들은 이 운영체제에서 하드웨어를 제어하기 위한 디바이스 드라이버와 API를 경험하게 된다.
- 응용 프로그램 개발방법:
  학생들은 응용 프로그램 개발 시에 소프트웨어 공학적인 접근을 통해 실시간 시스템 개발에서 필수적인 알고리즘 검증 과정을 거치도록 한다.

4. 교육 과정 및 절차

교육 과정은 (그림 1)에서 보는 바와 같이 크게 3단계로 나눌 수 있다. 첫째 단계는 하드웨어와 펌웨어 제작을 경험하는 단계로 Lego 센서, 엑츄에이터용 I/O 인터페이스 보드와 PC용 ISA 또는 PCI 인터페이스 기반의 레고 제어 보드를 제작한다. 이때 레고 제어 보드는 FPGA칩을 사용하도록 하며, 이 칩의 기능은 VHDL로 작성하여 넣게 된다. 두 번째 단계는 만들어진 하드웨어를 이용하기 위한 디바이스 드라이버와 API를 제작하고 시스템 프로그래밍의 전반을 실습하는 단계이며, 마지막 단계에서는 Lego로 구체적인 실시간 시스템 응용 모델을 제작하고, 이전 과정에서 완성된 하드웨어와 디바이스 드라이버 및 API를 이용해 시스템을 운영할 실



(그림 1) 실시간 시스템 제작 실습 과정

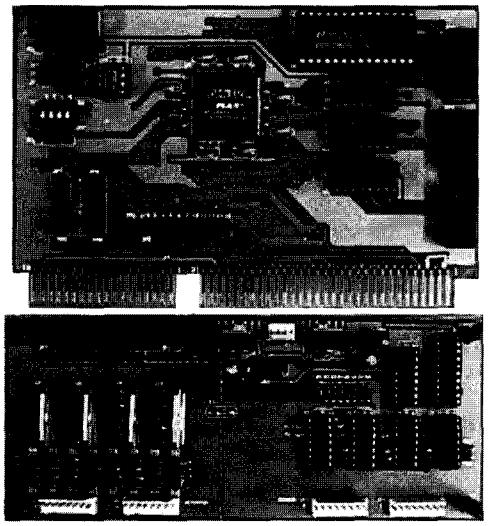
시간 응용 프로그램을 최종 제작하는 과정을 거친다. 특히 마지막 프로그램 제작 과정에서는 소프트웨어 공학론을 따라 프로그램의 구조를 디자인하고 그 작동 가능성을 검증하는 과정을 거쳐 체계적인 제작 능력을 기른다. 이러한 교육 과정은 실시간 시스템 제작에서 생길 수 있는 소프트웨어, 하드웨어 동시 설계의 기회를 학생들에게 부여하여 이 분야에 필요한 전반적인 지식과 경험을 가지는데 도움이 될 것이다. 이하의 본문은 위와 같은 실습 과정 중에 겪게 되는 과정들을 소개한다.

5. 하드웨어와 펌웨어 제작

실습에서 가장 먼저 시도하게 되는 일은 시스템의 기반이 될 하드웨어를 제작하는 것이다. 이 실습도구의 하드웨어는 센서 엑츄에이터 접속 회로와 그 디바이스를 제어할 컨트롤 회로로 구성된다. 학생들은 I/O 디바이스의 동작원리를 분석하고 그것을 위한 I/O 접속 회로를 직접 제작하는 일부터 실습을 시작하게 된다.

제작하는 I/O 인터페이스 보드는 Lego사에서 판매하는 Lego MindStorms[7] 세트에 들어있는 모든 종류의 입출력 센서와 엑츄에이터들을 제어하기 위한 것이다. 이러한 입출력 장치들과 I/O 인터페이스 보드의 기능은 다음과 같다.

- 센서[9] : 터치 센서, 온도 센서, 광 센서, 회전 센서
- 엑츄에이터 : 표준 모터, 기어 모터, 꼬마 전구, 부저
- 입력 장치 인터페이스 회로 :
  총 8개의 센서를 동시에 연결할 수 있다. 이 회로는 센서에 전원을 공급하고, 센서의 아날로그 출력을 레고 제어 보드에 있는 ADC칩에 전달해주는 일을 한다.
- 출력 장치 인터페이스 회로 :
  총 8개의 엑츄에이터들을 동시에 연결할 수 있다. 엑츄에이터의 구동 신호는 레고 제어 보드가 만든다.

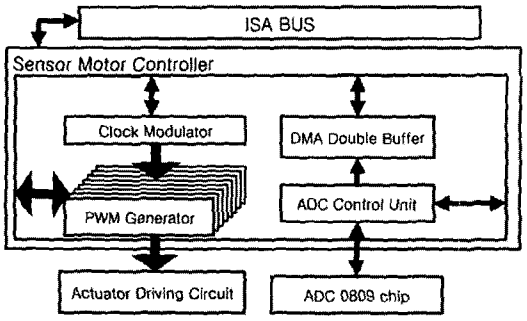


(그림 2) ISA 기반의 레고 제어 보드(上)와 센서 엑츄에이터 접속 보드(下)

위와 같은 I/O 인터페이스 보드는 말단의 디바이스들을 제어하기 위한 일을 할 뿐이며, 실질적인 운영은 ISA기반의 레고 제어 보드에서 하게 된다. 이를 위해 학생들은 ISA기반의 제어 보드를 제작해야 한다. 제어 보드는 센서의 입력 데이터를 관리하고 엑츄에이터의 동작을 제어하며, 프로그램의 명령에 따라 이들 디바이스의 동작 설정을 변경하고 관리하게 된다. 이 제어 보드는 메인 보드의 ISA 슬롯에 장착되어 동작하며, ISA 아키텍쳐를 따라 I/O R/W (Read, Write)와 DMA, 인터럽트를 사용한다[4].

이런 다양한 기능을 제작하기 위해 학생들은 VHDL[3]로 제어 회로를 작성하게 된다. 작성된 VHDL 소스 코드는 ALTERA사의 Flex10k FPGA칩으로 구현한다[7]. 또한 이 과정의 실습은 학생들이 직접 완전한 소스를 구현하는 것은 쉽지 않기 때문에 레퍼런스를 제공한다. (그림 3)은 제공되는 VHDL 레퍼런스 소스 코드의 기능 구성도다. 이러

한 레퍼런스 소스를 기반으로 학생들은 필요에 따라 좀 더 다양한 기능을 추가해보면서 VHDL을 이용한 하드웨어 소프트웨어 Co-design을 경험할 수 있다.



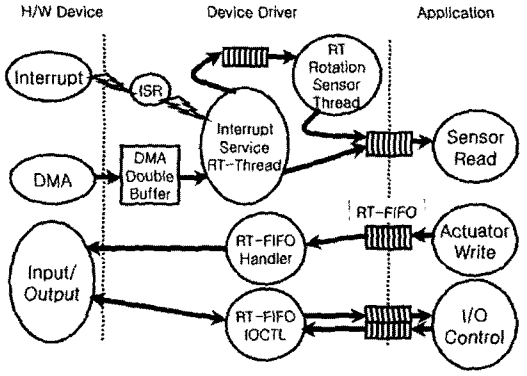
(그림 3) Flex10k 제어 칩의 구조

6. 디바이스 드라이버와 API 제작

하드웨어 디바이스 제작 후에는 이를 프로그램에서 이용하기 위한 디바이스 드라이버와 API를 제작한다. 학생들은 RTLinux에서 디바이스 드라이버를 제작하여 시스템 프로그래밍과 하위 레벨 프로그래밍을 경험하게 된다. (그림 4)는 레퍼런스로 제작된 ISA 기반 레고 제어 보드의 디바이스 드라이버 구조도다.

RTLinux에서는 실시간성이 필요한 작업들을 커널 모듈로 구현하게 되어 있는데, 이러한 구조에 따라 디바이스 드라이버에는 실시간 작업용 쓰레드들도 같이 포함하게 된다. 학생들은 실시간성이 중요하지 않은 GUI 등을 일반 리눅스용 프로그램으로 제작하고, 실시간성이 필요한 시스템 제어 작업들을 RTLinux 커널 모듈로 제작하게 된다.

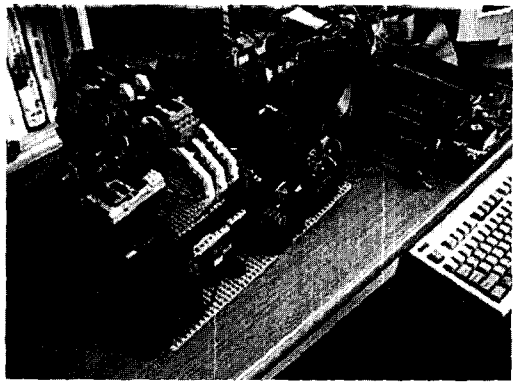
디바이스 드라이버 제작 후에는 유저 프로그램을 위한 API를 제작한다. 물론 학생들에게는 기본적인 레퍼런스 디바이스 드라이버와 API가 제공되며, 응용 사례에 따라 학생들이 임의로 직접 기능을 추가하게 된다. 이러한 부분에서 학생들은 각자의 타켓 시스템에 맞춰 디바이스 드라이버와 API를 최적화 시킬 수 있으며 그로부터 디바이스 드라이버나 시스템 프로그래밍의 기초를 체득할 수 있다.



(그림 4) 디바이스 드라이버의 기능 구조도

7. Lego 실시간 시스템 타켓 모델 제작

디바이스 드라이버와 API 작성이 완료되면 이를 기반으로 실시간 시스템 타켓 모델을 제작하게 된다. 어떤 시스템을 제작할 것인지는 학생들이 직접 창의적인 아이디어를 내어 결정하게 되며, Lego를 이용해 구체화하게 된다. (그림 5)는 실례로 제작된 것으로 공장 자동화 시스템의 로봇 팔과 컨베이어 벨트 시스템을 모델링한 것이다.



(그림 5) 로봇 팔과 컨베이어 벨트를 이용한 공장 자동화 시스템의 Lego 모델 사진

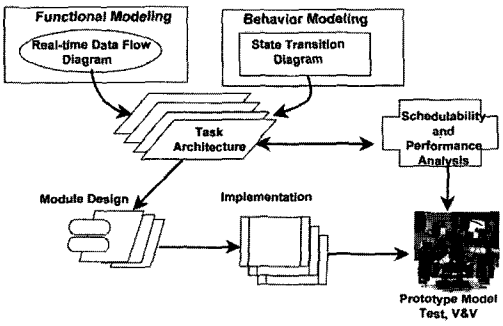
이 시스템은 컨베이어 벨트로 운반하는 Lego 블록이 합격인지 불합격인지를 파악하여, 불합격인 경우 로봇 팔로 집어서 버리는 일을 하며, 구체적으로 3개의 광 센서와 4개의 회전 센서, 1개의 표준 모터, 3개의 기어 모터를 사용한다.

3개의 광 센서는 벨트의 시작지점과 종료지점에 한 개씩 배치해 작업의 시작과 끝을 파악하며, 나머지 하나는 벨트의 중간 지점에서 이동하는 Lego 블록의 색깔을 감지해 합격, 불합격의 판단 정보를 제공하기 위해 사용된다. 4개의 회전 센서는 각각 모터의 회전 축에 연결되어 모터의 회전 방향과 회전 수를 측정하며, 이들을 통해 로봇 팔과 벨트의 움직임을 파악하고 제어한다.

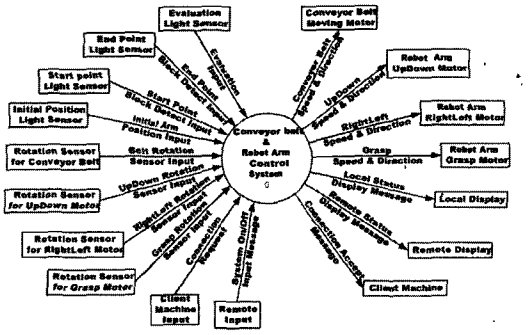
표준 모터는 컨베이어 벨트를 움직이기 위해 사용되며, 3개의 기어 모터는 로봇 팔을 움직이는 데 사용된다. 참고로 로봇 팔은 상하 좌우로 움직일 수 있으며, 손을 오므리고 펴는 것이 가능하다.

8. 소프트웨어 공학론을 기반으로 한 요구 분석 및 프로그램 구조 설계 과정

이 과정은 본격적인 실시간 시스템 응용 프로그래밍을 위한 작업으로, 앞의 사례와 같은 타켓 시스템의 환경과 요구사항 등을 소프트웨어 공학적인 차원에서 분석하고, 프로그램의 구조를 디자인하게 된다. 이 과정은 (그림 6)처럼 도시할 수 있다.



(그림 6) Software Engineering Process

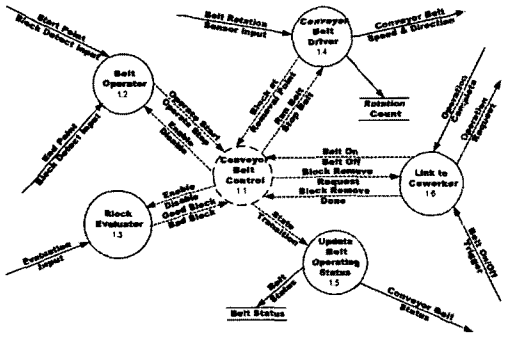


(그림 7) System Context Diagram

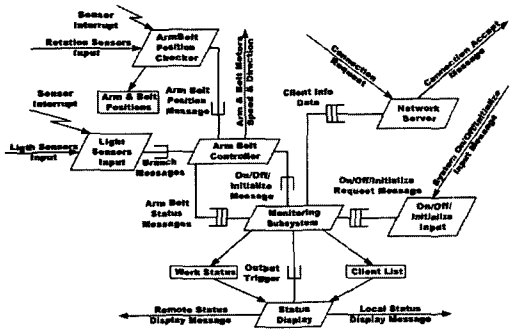
(그림 6)의 작업도는 경성 실시간 시스템 제작에 널리 사용되는 개발 기법인 CODARTS[2]를 따른 것으로, 타켓 시스템의 요구 사항을 분석하고 그것을 해결하기 위한 프로그램 구조를 디자인하며, 설계된 프로그램의 구조에서 스케쥴 가능성과 기타 문제점을 분석 및 보완하여 최종적으로 프로그램을 구현하는 과정으로 구성되어 있다. 그러면 앞서 소개된 로봇 팔과 컨베이어 벨트 시스템 디자인에서 이 과정을 따른 예를 살펴보자.

타켓 시스템의 요구사항을 분석하는 과정에서는 가장 먼저 Context Diagram 작성을 하게 된다. 이는 시스템의 입출력 환경이 어떠한 지를 파악하기 위한 것으로 (그림 7)이 실제 작성한 예이다. Context Diagram을 작성한 후엔 이를 기반으로 문제를 해결하기 위해 필요한 기능들을 파악하고 그것들의 구성 방식을 결정해야 한다. 로봇 팔과 컨베이어 벨트 시스템의 경우에는 로봇 팔의 제어, 컨베이어 벨트의 제어, 그리고 상태 모니터링의 세 가지 기능으로 나누었다. 분석 과정에서는 이 각각의 파트에 대한 데이터 및 제어 신호 흐름도를 작성해야 하며, (그림 8)은 그 중에서 컨베이어 벨트 제어부를 분석한 그림이다. (그림 9)는 분석 결과를 바탕으로 프로그램에서 필요한 것으로 파악된 기능 전체를 하나의 그림으로 나타낸 것이다. 이렇게 주어진 문제를 분석하고 프로그램의 구조를 디자인한 뒤에는 이 디자인이 실시간성을 보장하는 가를 확

인하는 과정을 가져야 한다. 이를 위해서 예상되는 작업의 절차를 분석하고, 각 태스크들의 우선순위와 작업 주기, 수행 시간 등을 파악하여 Schedulability 검증을 한다.

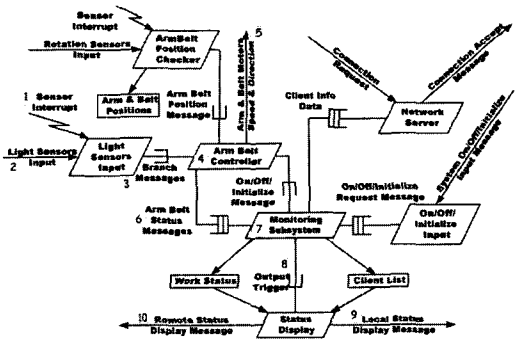


(그림 8) 컨베이어 벨트 제어부의 데이터 및 제어 신호 흐름도



(그림 9) Task Architecture Diagram

우리는 세 가지의 검증법으로 테스트 해봤다[2]. 특정 작업의 과정을 추적하면서 제한 시간을 만족하는 지 알아보는 이벤트 시퀀스 분석법, Rate-Monotonic 스케쥴러 적용 시를 가정해 Utilization Bound를 만족하는 지 확인하는 방법, 마지막으로 임의로 우선순위를 조정했을 때를 가정하여 스케쥴 가능성을 검증하는 방법을 적용해봤다. 이러한 검증을 위해서는 대상 작업을 선정해야 하는데, 여기서는 시작 지점에 블록이 놓여지고 이를 광 센서가 감지하여 작업이 시작되는 과정을 검증 대상으로 삼았다. 이 작업은 300msec 내에 완료되어야만 하는 시간 제약을 가지고 있다. 이 작업 과정을 표현한 Event Sequence Diagram은 (그림 10)이며, 이 작업 과정은 다음과 같다.



(그림 10) Event Sequence Diagram

1. 센서 입력을 알리는 인터럽트가 발생 (C_1)
2. 광센서로 블록이 있음을 감지 (C_2)
3. 작업 시작 메시지를 Arm Belt Controller에 전송 (C_3)
4. 작업 시작 상태로 전환함 (C_4)
5. 벨트 구동 모터를 동작 시킴 (C_5)
6. 작업 시작 상태 메시지 전송 (C_6)
7. Work Status 값을 갱신한다. (C_7)
8. Status Display 태스크에 출력 요청 (C_8)
9. 로컬 모니터로 출력 (C_9)
10. 원격 클라이언트에 상태보고 (C_{10})

위와 같은 과정에서 총 4가지 태스크가 관여하게 되며, Context Switching이 최소 4번이상 발생함을 알 수 있다. 메시지 전송시간(Cm)이나 Context Switching 소요시간(Cx)을 일정한 값으로 가정하고, 기타 태스크 소요시간을 정하고 수행 시간(Ce)을 계산하면 다음과 같다.

$$Ce = 3C_m+3C_x+C_1+C_2+C_4+C_5+C_7+C_9+C_{10} = 48 \text{ msec}$$

이외에 기타 작업의 수행에 의한 지연 시간까지

고려해야 하는데, 여기서는 ArmBelt Position Checker와 Network Server가 그것이다. 이 작업들이 300msec 이내에서 소요할 수 있는 작업 시간을 계산해보면 다음과 같다.

$C_a = 30*(C_{11}+2*C_x)+2*(C_{12}+2*C_x) = 102$ msec

따라서 대상 작업 수행과 관련해 걸릴 수 있는 총 예상 소요시간 Ct는..

$C_t = C_a+C_e = 102+48 = 150$ msec

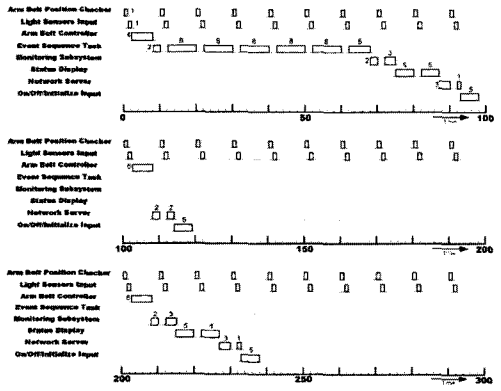
총 소요시간이 150msec로 300msec에 훨씬 못 미치므로 이 구조로 해당 작업을 수행하는 데는 문제가 없음을 알 수 있다.

두번째인 Rate Monotonic 스케쥴 방식에 대한 검증은 Utilization의 합산으로 파악되며, <표 1>을 참고로 계산해보면 0.585로 무한 개의 Task 운용시의 한계 Utilization인 0.69에 비해 작아서 문제가 없음을 알 수 있다.

<표 1> Task Parameters

Task	CPU Time	Period	Utilization	Priority (Case1)	Priority (Case2)
Arm Belt Position Checker	1	10	0.1	1	1
Light Sensors Input	1	10	0.1	2	2
Arm Belt Controller	6	100	0.06	3	3
Network Server	4	100	0.04	4	7
On/Off/Initialize Input	5	100	0.05	5	8
Monitoring Subsystem	5	200	0.025	6	5
Status Display	10	200	0.05	7	6
Event Sequence Task	48	300	0.16	8	4

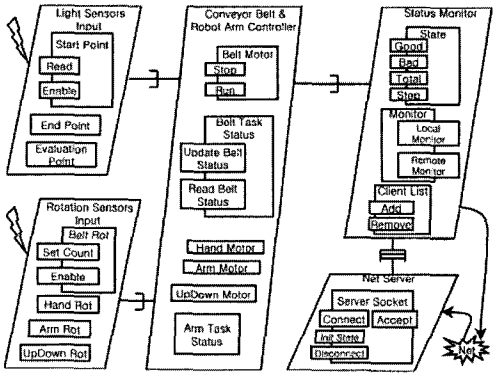
Case 1 : Rate monotonic priorities assigned
Case 2 : Non-rate monotonic priorities assigned



(그림 11) Timing Diagram

마지막은 임의의 우선 순위로 동작하였을 경우를 검증하는 방법으로 우선순위가 <표 1>의 Case 2처럼 주어질 때 동작에 문제가 없는 지를 Timing Diagram을 그려서 직접 확인하는 것이다. (그림 11)에 그 결과가 나타나 있으며 문제가 없다. 이러한 분석과 설계, 검증 과정을 통해 학생들은 체계적인 실시간 시스템 운영 프로그램 제작 과정을 경험하게 된다. 프로그램 구조 분석 및 검증을 하는 과정은 학생들에게는 가장 어려운 부분이자 가장 재미있는 경험 중의 하나가 될 것이다. 또한 이는 실용적인 실시간 시스템 실습으로써 의미 있는 교육이 될 것이다.

이렇게 검증을 마치면 실제 프로그램의 구조를 설계한다. 앞서 만든 Task Architecture Diagram과 Information Hiding Module을 합쳐서 완전한 프로그램의 구조를 만든다. 이 과정을 거치면 각 태스크 내에 정보 은닉 모듈들의 구조까지 포함된 Software Architecture Diagram 을 작성되게 되는데, (그림 12)가 그 예이다.



(그림 12) 정보은닉모듈을 포함한 태스크구조도

9. 응용 프로그램 구현 및 검증

분석, 설계와 검증이 완료되면 남은 일은 최종적으로 응용 프로그램을 작성하여 타켓 시스템에 적용하는 과정이다. 유저 프로그램은 RTLinux에서

GNU C를 이용하여 구현한다. 프로그램은 RTLinux의 특성에 맞춰 실시간성이 필요한 작업과 그렇지 않은 작업들을 구분하여 실시간 작업은 커널 모듈로 그렇지 않은 작업은 일반 유저 프로그램으로 작성하게 된다. 특히 GUI 부분은 자바 애플릿 등으로 원격 유저 인터페이스를 구축해본다. 물론 이러한 작업은 Software Architecture Diagram에 디자인된 구조를 바탕으로 진행하게 되지만, 프로그램의 구조는 작성하는 환경에 따라 일부를 변경할 수도 있다.

앞서 살펴본 사례의 경우, 실제 제작된 응용 프로그램에서는 앞서 구분된 5개의 태스크를 4개의 쓰레드로 구분하여 작동하도록 만들었다. 컨베이어 벨트와 로봇 팔을 제어하는 쓰레드, 상태 모니터링 쓰레드, 네트웍 서버 쓰레드는 구조도의 분류를 그대로 따랐다. 그러나, 센서입력 처리 태스크들의 경우 하나의 인터럽트를 두 부분이 공유하기 때문에 하나의 쓰레드로 합치게 되었다. 이때 쓰레드 내에서는 광 센서 처리 함수와 회전 센서 처리 함수로 구분하여 인터럽트 발생 상황에 따라 함수를 달리 호출하여 처리하도록 하였다.

이렇듯 앞에서 이론적으로 디자인한 구조를 실제로 적용하는 과정에서 실습자는 응용 시스템의 목표에 따라, 태스크를 실제로 구현하는 과정과 태스크들 간의 중요도에 따른 우선권 배분 문제, 하나의 자원을 공유하는 태스크간의 세마포나 뮤텍스를 사용하는 동기화 처리 등의 경험을 가질 수 있을 것이다. 또한 설계 당시의 스케줄 가능성 분석 자료를 바탕으로 하여 실제 구현 결과가 만족하는지도 검증하게 된다.

10. 결 론

본 연구진에 의해서 개발된 내장형 실시간 시스템 교육 및 실습을 위한 도구를 이용하면 앞에서 논한 바와 같이 실시간 시스템 개발에 필요한 모든 단계를 망라해볼 수 있어서 매우 효과적인 교육 체계를 마련할 수 있다. 다음은 본 도구와 교육체계를 적용함으로써 얻을 수 있는 장점들이다.

첫째, 하드웨어와 소프트웨어를 동시에 설계 및 구현할 수 있는 기회를 제공한다.

둘째, 재사용이 가능한 레고 블록을 이용하여 다양한 형태의 실시간 시스템 모형을 제작할 수 있다. 경우에 따라서는 실제 개발해야 하는 실시간 시스템의 프로토타입 모델용으로도 이용할 수 있다.

셋째, 실시간 이론을 실제 시스템에 적용해 보기 위한 도구로서 유용하게 활용할 수 있다. 예를 들어 우선 순위 역전 문제가 있는 상황을 만들고, 그것을 해결할 수 있는 방법을 적용하여 문제가 해결되는지를 확인할 수 있으며, 스케줄링 가능성을 분석하거나 여러가지 스케줄링 이론을 적용하여 그 결과를 분석하는 도구로 활용할 수도 있다.

이 도구는 현재 경북대학교 전자 전기 컴퓨터 학부 3학년 과정에서 내장형 시스템 실험(전자공학실험4) 과목과 대학원 전자공학과 실시간 시스템 설계 과목에서 실험 도구로 활용하고 있다.

참고문헌

[1] Alessandro Rubini, "LINUX Device Drivers, O'Reilly, 1998.

[2] Hassan Gomaa, "Software Design Methods for Concurrent and Real-Time Systems, Addison-Wesley, 1996.

[3] Peter J. Ashenden, "The Designer's Guide to VHDL, Morgan Kaufmann Pub, 1996.

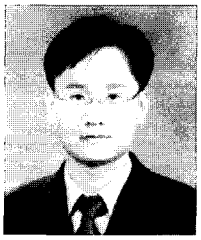
[4] Tom Shanley, Don Anderson, John Swindle, "ISA System Architecture, Addison-Wesley, 1995.

[5] Wolfgang A. Halang, Krzystof M. Sacha, "Real-Time Systems, World Scientific, 1992.

[6] 박성호, 이상문, 박동환, 강순주, "실시간 시스템 교육 및 실험 Tool Kit 개발, KISS`2000

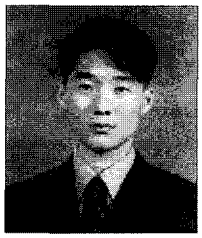
Spring Conf., 2000년 4월.
[7] http://mindstorms.lego.com/
[8] http://www.lego.com/
[9] http://www.plazaearth.com/usr/gasperi/ lego.htm
[10] http://www.rtlinux.org/

저자약력



정 기 훈

2001년 경북대학교 전자전기공학부 (공학사)
2001년-현재 경북대학교 전자공학과 석사과정
관심분야 : Real-Time System, System Software, IEEE1394
e-mail : jghlof@palgong.knu.ac.kr



김 도 훈

2001년 경북대학교 전자전기공학부 (공학사)
2001년-현재 경북대학교 전자공학과 석사과정
관심분야 : System Software, Real-Time System, RT-CORBA
e-mail : zamggan@palgong.knu.ac.kr

박 성 호

1999년 경북대학교 전자전기공학부 (공학사)
2001년 경북대학교 정보통신학과 (공학석사)
2001년-현재 경북대학교 전자공학과 박사과정
관심분야 : System Software, Real-Time System, Home Network, Software Engineering
e-mail : slblue@palgong.knu.ac.kr

강 순 주

1983년 경북대학교 전자공학과 (공학사)
1985년 한국과학기술원 전자계산학과 (공학석사)
1995년 한국과학기술원 전자계산학과 (공학박사)
1985년-1996년 한국원자력연구소 연구원, 핵인공지능 연구실 선임 연구원, 전산정보실 실장
1996년-현재 경북대학교 전자전기컴퓨터학부 정보통신전공 조교수
관심분야 : Real-Time System, Software Engineering, Knowledge-Based System
e-mail : sjkang@ee.knu.ac.kr