Proceedings of the 11th
KACC, October 1996

실시간 운영체제를 이용한 로봇제어기 소프트웨어의 구현 및 성능분석

An Implementation and Performance Analysis for Robot Control Software under Real-Time Operating Systems

°손 승 우, 이 기 동

영남대학교 전산공학과 (Tel: 810-2584; Fax: 816-1976; E-mail: rkd@white.yeungnam.ac.kr)

Abstracts: Robot control software is a hard real-time system that must output the planned trajectory points within an explicit short time period. In this paper, we present a design and implementation method for robot control software using commercial real-time operating systems, RTKernel 4.5. Therefore, various robot motions, efficient user interface, and system failure check are easily implemented by using multitasking function, intertask communication mechanism, and real-time runtime libraries of RTKernel. The performance analysis of commercial real-time operating system for robot control is presented based on Timed Petri net(TPN) and we can use these results to design an optimal system.

Keywords: real-time, robot control s/w, Petri net, performance analysis, multitasking

1. 서론

로봇 제어기 소프트웨어는 일정한 샘플링 시간 내에 계획된 궤적 값을 제어기로 내주어야 하며 고장진단과 사용자 인터페이스 등의 다양한 기능들의 지원을 필요로 하는 경성 실시간 시스템(hard real-time system)이다. 이러한 로봇 제어기 소프트웨어의 개발에 있어서 과거에는 제어기의 운영체제를 직접 설계하고 코딩하였으므로 상당히 많은 시간과 노력이 요구되었을 뿐만 아니라, 많은 경우 구현된 운영체제가 멀티태스킹을 지원하지 않으므로 여러 가지 사건의 실시간 처리와 다양한 모션을 구현하기 위해서는 하드웨어 인터럽트를 빈번하게 이용해야 했다. 이것은 로봇 제어를 위한 응용 프로그램을 작성하는 작업을 힘들게 할 뿐만 아니라, 실시간 시스템의 특징을 갖고 있는 로봇 제어기의 기능을 여러 면에서 제약하는 요인이 되었다. 특히 최근 들어 마이크로 프로세서의 성능이 크게 향상되면서 로봇 제어기의 다양한 기능들을 하드웨어로 구현하기보다는 점차적으로 소프트웨어로 구현하고 있다. 이러한 요소들로 인하여 로봇 제어 소프트웨어의 구조 또한 그 복잡도가 점점 더 커지고 있다.

이러한 문제를 극복하기 위해 실제 실시간 시스템인 로봇 제어기 소프트웨어에 적합하거나 범용 제어 시스템용의 연구용 실시간 운영체제들이 많이 설계되고 있다[1,5,8]. 본 논문에서는 실시간 시스템의 구축을 용이하게 위해 상용의 PC용 실시간 운영체제인 RTKernel을 이용하여 로봇 제어기 소프트웨어를 디자인, 구현하는 방법을 제안한다. 특히, 로봇의 다양한 궤적계획과 효과적인 사용자 인터페이스, 시스템의 고장진단 및 복구 등을 실시간으로 처리하기 위하여 RTKernel의 멀티태스킹기법과 태스크간 동기화 기법을 적절히 응용하여 보다 효과적인 로봇 제어기 소프트웨어를 구성한다. 그리고 사용한 상용의 실시간 운영체제인 RTKernel을 시간 페트리 넷(Timed Petri net)을 이용한 모델링을 통하여 성능분석을 제시한다. 또한 이러한 성능분석의 결과를 바탕으로 최적의 시스템을 구성하는데 적용한다.

2. 실시간 시스템과 실시간 운영체제

2.1 실시간 시스템

일반적으로 실시간 시스템이란 시스템에 주어진 입력에 의해 발생하는 출력에 대해 입력부터 출력까지에 소요된 시간-반응(response time)이 일정한 제한치 보다 작을 것이 요구되는 시스템을 말한다. 좀 더 좁은 의미로 말하면, 반응시간이 제한시간을 초과하는 경우 정상적으로 동작하지 않는 시스템을 지칭한다[3, 7]. 실시간 시스템은 정기적으로 발생되는 데이터를 실시간으로 처리하거나, 비정기적으로 발생하는 사건(event)에 대한 신속한 대응을 목표로 한다[3]. 로봇 제어기 시스템 역시 실시간 시스템으로 로봇 손끝의 위치나 속도를 제어하기 위해서 정기적으로 각 축 모터의 엔코더 값을 읽어서 궤적계획의 결과인 원하는 값과 비교, 제어 알고리즘에 따라 모터를 제어해야 한다.

이러한 실시간 시스템을 디자인하고 구현하기 위해서는 다음 사항들에 대한 세심한 고려가 있어야 한다[3].

1) 적절한 하드웨어와 소프트웨어의 선택 또는 조합
2) 특별한 운영체제를 설계할 것인지, 상용화된 운영체제를 이용할 것인 지의 결정
3) 시스템 개발을 위한 적절한 개발언어의 결정
4) 세심한 디자인과 엄격한 테스트로 시스템의 신뢰도(reliability)와 에러저항력(fault tolerance)의 극대화
5) 반응시간(response time)의 예상과 측정, 반응시간의 단축

2.2 상용의 운영체제

실시간 운영체제는 실시간 시스템의 구축이 용이하도록 다음과 같은 기능을 갖춘 운영체제이다. 실시간 시스템 설계자는 이러한 기능을 갖춘 실시간 운영체제를 직접 디자인하든지 상용화된 것을 구입하여 사용하여야 한다[3].

1) 멀티태스킹(multitasking)
2) 선점형 스케쥴링(preemptive scheduling)
3) 태스크간의 통신 및 동기화
4) 태스크와 인터럽트와의 통신

이러한 특징을 가진 상용의 실시간 운영체제는 운용 기반 운영체제에 따라 크게 PC용과 유닉스용으로 나뉜다. 많이 쓰이는

것으로는 OS-9, pSOS+, VMEExec, VRTX, VxWorks 등이 있으며 대부분 유닉스 운영체제아래 운용되는 것들이다. 기반 운영체제의 특징으로 인하여 유닉스용 실시간 운영체제는 보다 다양한 기능에 완벽한 멀티태스킹이 지원된다. 반면 PC용 실시간 운영체제는 유닉스용보다 스케쥴링능력이나 제공되는 루틴은 적지만 구현될 시스템의 규모가 적어지고 실제 사용자가 쉽게 적용할 수 있는 PC환경이라는 장점이 있다. 이러한 PC용 상용 실시간 운영체제로는 FlexOS, iRMX Ⅲ, QNX, RTKernel, RTXC 등이 있다. 본 논문에서는 이중에서 On-Time Informatik사의 RTKernel을 로봇 제어기 시스템의 운영체제로 사용하였다.

RTKernel 4.5는 MS-DOS 운영체제아래 운용되며, 커널은 C와 어셈블러로 구성되어 있다. RTKernel 4.5의 주요 특징을 보면 다음과 같다[4].

1) 협동형(cooperative)과 선점형(pre-emptive) 스케쥴링
2) 사건/인터럽트-구동 스케쥴링(우선 순위기반)
3) 태스크간 동기화 - 세마포어, 메일박스(메시지 큐 또는 파이프), 메시지 패싱(동기성 데이터 전달)
4) 화면, 키보드, 프린터 등의 입출력 장치 및 IPX 통신 드라이버 지원
5) MS-DOS의 재진입 문제 해결

3. 기존의 연구

로봇 제어 시스템의 실시간성을 만족시키기 위해서 많은 경우 제어 시스템에 적합한 실시간 커널을 직접 설계하고 있다[1,5,8]. 그리고 이러한 커널들은 대부분 객체 지향적으로 설계하여 개발의 용이성과 개발후 시스템의 유지, 보수의 편리성을 도모하고 있다[5,8]. 그런데, 이러한 커널들은 실시간 운영체제의 설계에 중점을 두어서 실제 다양한 제어시스템의 특성을 모두 만족하기에는 부하가 크다. 또한 기존의 연구에서 나타난 로봇 제어 시스템에서는 주로 새로운 커널 또는 프로그래밍환경을 설계한 것으로 설계한 커널 또는 환경의 성능 내지 가용용성(applicability) 검증을 위한 아주 기본적인 태스크들만을 정의하였다. 실제 로봇 제어 소프트웨어에서는 궤적계획을 하는 태스크와 매 샘플링 주기마다 데이터를 제어기로 내주는 태스크가 가장 기본적이다. 그러나 이러한 태스크들만으로는 로봇의 보다 지능적인 제어를 요구하는 산업현장의 요구를 충족시킬 수 없다. 따라서 실제 로봇제어에 필요한 다양한 태스크들을 실시간적으로 구성하는 연구가 필요하며 부가적으로 로봇 제어용 실시간 운영체제의 성능분석도 병행하여 최적의 시스템을 구현하여야 한다.

4. 로봇 제어기 소프트웨어의 구성 및 구현

본 논문에서는 로봇 제어기 소프트웨어의 구성 태스크로 다음과 같이 11개를 정의하였다.

1) kin - 로봇의 기구학을 해결하는 태스크
2) inv_kin - 로봇의 역기구학을 해결하는 태스크
3) io - 로봇 제어기의 입력원으로 쓰일 수 있는 키보드, 티칭박스, 마우스 등의 입출력을 처리하는 태스크
4) servo - 매 샘플링 주기마다 로봇 제어기로 계획된 궤적값을 내보내는 태스크
5) clk - 시스템의 전반적인 클락 및 시간을 처리하는 태스크
6) dis - 각종 메시지를 화면에 디스플레이 하는 태스크
7) failchk - 로봇 제어기의 고장을 매 샘플링 주기마다 점검하여 적절히 대응하는 태스크
8) react - 사용자의 react 명령을 받아서 처리하는 태스크
9) load - 현재 CPU의 부하상태를 체크하는 태스크
10) log - 계획된 궤적 값들을 디스크에 기록하는 태스크
11) trj - 이동 명령에 따른 궤적계획을 하는 태스크

이중에서 비교적 우선 순위가 높아야 할 태스크로는 servo, failchk, io가 있다. 태스크들이다. Failchk 태스크는 그 중요도로 보아 servo 태스크보다 우선 순위가 높아야 한다. Failchk 태스크는 비주기적으로 발생하며 발생시 즉시 응답해야 한다. 여기서는 매주기(100ms)마다 오류점검을 하는 방식을 택하였다. 사용자 인터페이스를 위한 dis 태스크는 다른 태스크들에 비해 우선 순위를 낮게 주어도 무리가 없다. React 태스크는 주어지는 명령에 따라 태스크의 우선 순위가 동적으로 변화할 수 있다. 이러한 11개의 태스크들의 우선 순위와 실행주기를 정리하면 표 1과 같다.

표 1. 태스크의 정의 (0 : 가장 낮은 우선 순위)
Table 1. Definition of tasks (0 : lowest priority)

태스크	priority	실행주기	태스크	priority	실행주기
clk	3	1sec	load	2	400ms
dis	2		log	1	16ms
failchk	5	100ms	react	1	
inv_kin	2		servo	4	16ms
io	3	80ms	trj	3	
kin	2				

앞에서 정의한 태스크들을 주어진 RTKernel의 동기화 도구인 메일박스와 메시지 패싱을 이용하여 그림 1과 같이 구성하였다. Trj 태스크는 계획된 궤적 값을 메일박스를 통하여 servo와 log 태스크에 전달하며, trj와 react 태스크는 io 태스크로부터 요청 메시지가 발생하면 실행한다. 그리고 kin과 inv_kin 태스크는 trj 태스크의 요청 메시지가 발생하면 실행한다. Dis 태스크 역시 다른 태스크들에 의한 화면 출력 요청 메시지가 발생하면 실행하며, 나머지 실행주기를 가진 태스크들(clk, failchk, io, load, log, servo)은 각각 샘플링 시간에 맞게 깨어나서 실행한다.

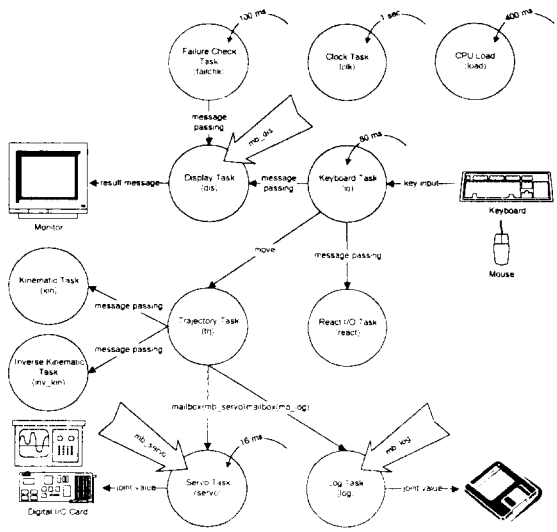

그림 1. 태스크간 통신 및 동기
Figure 1. Intertask communication and synchronization

5. 성능평가

실시간 시스템은 그 특성으로 인하여 시스템의 정확한 검증이 반드시 수반되어야 한다. 이것은 safe-critical한 경성 실시간 시스템에서 더욱 절실히 요구된다. 이러한 실시간 시스템이 시스템의 요구사항을 만족하는지를 검증하는 방법으로는 크게 대상 시스템의 예상되는 기능을 가능한 시험 시나리오를 통하여 시뮬레이션 하는 것과 직접 시스템의 성능을 모니터 하는 방법이 있다. 시뮬레이션과 실험을 통한 접근방법은 시스템 개발자에게 어느 정도의 확률적으로는 시스템이 동작한다는 확신을 제공한다. 해석적 또는 큐잉 이론적인 접근 역시 이러한 목표와 비슷하다. 그러나 시스템의 해석을 이용하면 해석결과를 개발과정에 쉽게 도입할 수 있다. 게다가 시스템을 해석한 결과가 어느 정도의 오류 확률보다 작다면, 실제 시스템도 그것의 요구사항을 만족한다고 말할 수 있다[7].

확률적 접근방법은 심지어 최악의 경우에서도 정확히 동작할 수 있는 시스템의 검증 또는 수단이 된다. 이러한 기법들은 최악의 도착 패턴, 최악의 실행시간, 최악의 오류가정 등에 대한 정보를 필요로 한다. 이러한 최악의 시간에 근거한 접근방법은 실시간 시스템의 safety-critical한 부분의 정확성을 검증하는데 필수적이다[7].

5.1 시스템 모델링

로봇 제어기 소프트웨어는 태스크간에 발생되는 이산사건에 의해 상태가 변화하므로 이산 사건 시스템이다. 본 논문에서는 사용한 운영체제의 성능을 여러 가지 이산형 모델링 기법 중에서 페트리 넷(Petri net: PN)을 이용하여 분석한다.

PN은 병행적(Concurrent), 비동기적(Asynchronous), 분산적(Distributed)으로 사건(event)이 발생하는 시스템을 표현하기 위한 수학적 모델이다. PN의 모델화 대상이 되는 시스템은, 상태가 사건의 발생에 의해 변화하는 사건 구동형(event driven) 시스템이므로 RTKernel의 사건 구동형 스케쥴러를 쉽게 모델링할 수 있다. 또한 PN에 의한 모델화에 있어서 특징적인 측면은 상태의 분산성과 사건 발생의 국부성에 있다. 전체 시스템의 상태는 국부적으로 정의되는 상태의 집합으로 표현되며, 사상은 미리 정해진 조건을 만족할 때, 혹은 미리 정해진 상태에 있을 때 발생 가능하다. 따라서 발생조건을 공유하지 않는 복수의 사건은 서로 독립하여 발생하며, 이 독립성에 의해 병행성과 비동기성의 표현이 가능해진다. 이러한 PN의 독립성은 본 논문에서 정의한 다수의 태스크들을 쉽게 모델링할 수 있다[9].

시간 페트리 넷(Timed Petri Net: TPN)은 넷의 place나 transition에 시간 개념을 도입한 것이다. 본 논문에서 정의한 주기적 태스크는 TPN을 이용하여 그림 2와 같이 모델링할 수 있다[6].

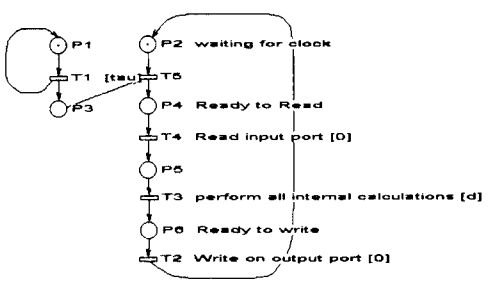

그림 2. 간단한 페트리 넷 모델
Figure 2. A simple Petri net model

그림의 좌측의 transition에 나타나있는 τ로 이 태스크의 실행주기를 표현한다. 그리고 우측의 각 transition에서 실제 태스크가 실행하는 시간을 적용하면 된다. 예를 들면, Servo 태스크의 실행주기는 그림 2에서 t1의 transition 시간인 τ를 실행주기인 16ms로 하면 된다.

이외에도 각 태스크의 실행 우선 순위는 그림 3과 같이 각 태스크의 transition에 우선 순위를 부여하여 모델링할 수 있다. 그림 3에서 t1, t10, t11은 우선 순위가 가장 높아 항상 먼저 실행된다. 그리고 왼쪽의 Servo 태스크의 우선 순위는 2이므로 t2, t3, t4, t5 모두 2로 할당하고, t6, t7, t8, t9는 1로 할당하였다. 따라서 예를 들어 t3과 t7이 동시에 활성화되면 우선 순위가 높은 t7이 먼저 실행된다.

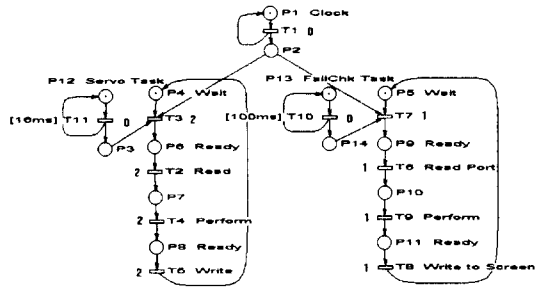

그림 3. 태스크의 우선 순위 모델링
Figure 3. Modeling Task's Priority

그리고 페트리 넷의 최근의 확장으로 큐(queue)를 모델링할 수 있다. 즉 큐가 필요한 곳의 place의 용량을 설정하면 된다. 각 태스크간의 동기화 도구로 사용한 메일박스는 이러한 place의 큐로서 그림 4와 같이 모델링할 수 있다. 그림에서 메일박스 p12의 용량은 15로 설정하였다.

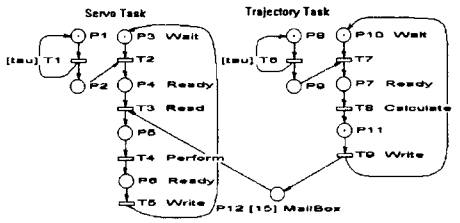

그림 4. 메일박스 모델링
Figure 4. Modeling Mailbox

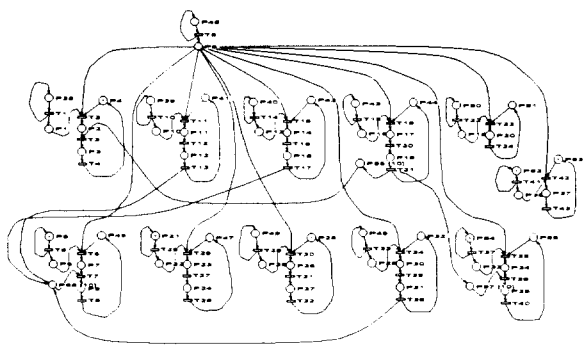

그림 5. 로봇 제어기 소프트웨어의 페트리 넷 모델
Figure 5. Petri net model for robot control software

본 논문에서는 각 태스크의 실행시간, 우선 순위, 태스크간의 통신도구 등을 잘 모델링 할 수 있는 시뮬레이션 도구로 Visual SimNet 1.34[2]를 사용하였다. 이것을 사용하여 정의한 태스크를

모델링 하면 그림 5와 같다. 11개의 태스크를 모델링한 것외에도 전체적인 시스템의 모델링을 위하여 전역 클락을 모델링 하였다. 전역 클락은 RTKernel에서 제공되는 소프트웨어 클락 틱이다.

5.2 태스크들의 예상실행시간 및 실행시간 분포도

실제 시스템을 근사적으로 모델링하기 위하여 RTKernel에서 제공되는 라이브러리를 사용하여 각 태스크의 실행시간을 측정하였다. 측정한 각 태스크들의 실행시간 분포는 표2와 같다.

실행시간이 고정적이지 않은 태스크들은 최악의 시간을 사용하였다. 실행주기가 있고 실행이 간단한 태스크의 실행시간은 상수 분포로 하였고, inv_kin, kin, trj 등의 태스크들은 지수 분포를 사용하였다. 그리고 react 태스크를 깨우는 사건도 지수분포를 사용하여 모델링하였다. 이렇게 설정한 시간분포함수를 그림 5의 해당하는 각 transition에 적용하였다.

표 2. 태스크의 실행시간분포
Table 2. Execution time distribution of tasks

태스크명	실행시간	태스크명	실행시간
clk	1	load	1
dis	1~3	log	2~3
failchk	1	react	2~3
inv_kin	5~6	servo	5~6
io	1~2	trj	5~6
kin	2~3		

5.3 실험결과

정의한 각 태스크들의 서비스 거리 및 큐의 용량을 측정하였다. 표 3은 정의한 태스크 중 react 태스크를 제외한 실행상 그 중요도가 높은, 즉 우선 순위가 높은 태스크에 대한 서비스 거리 분포특성을 나타낸 것이다. 표 4는 태스크간의 통신수단으로 사용한 메일박스의 평균 용량을 나타낸 것이다.

모델링한 태스크 중에서 주기적 태스크인 clock, failchk, io, load, servo 태스크들의 결과를 살펴보자. 5개의 주기적 태스크 중 서비스를 제때 받은 것은 clock, failchk, load 태스크들이다. io 태스크와 servo 태스크는 서비스를 제때 받지 못할 확률이 약간씩 있다. 특히 servo 태스크는 약간의 miss 확률이라도 로봇의 동작에 영향을 미칠 수도 있다. servo 태스크는 failchk 태스크보다는 우선 순위 반드시 낮지만 실행할 당시에는 가장 높은 우선 순위를 보장해야만 로봇의 모션에 이상이 없다. 이렇게 하기 위해서는 동적으로 우선 순위를 조정하거나, 자원 세마포어를 사용하여 시스템을 설계, 모델링해야 한다. 그러나 io 태스크의 miss 확률은 견딜 수 있다.

표 3. 결과 I (서비스 거리)
Table 3. Result I (service distance)

태스크명	mean	deviation	minimum	maximum
failchk	100	0	100	100
io	80	0.0999	80	81
trj	13.4	5.29	0	17
servo	16	0.0999	16	17
clock	1000	0	1000	1000
load	400	0	400	400

비주기적인 태스크들은 비록 서비스률이 낮지만 실제 로봇의 동작에는 무관한 것들이므로 시스템에 큰 장애를 주지 않는다. 특이한 것은 load나 clk 같은 태스크들은 낮은 우선 순위에도 불구하고 제때에 서비스를 받는다는 것이다. 이것은 두 태스크 모두 주기적인 태스크이기는 하나 실행주기가 servo 태스크에 비해 상대적으로 길기 때문이다.

표 4. 결과 II (평균 큐의 길이)
Table 4. Result II (mean queue length)

메일박스명	mean	deviation	minimum	maximum
servo	13.6	1.96	0	15
display	0.0169	0.131	0	2

servo 메일박스는 평균 길이가 13정도이므로 미리 정한 15가 적당하다. 나머지 메일박스는 우선 순위가 낮은 태스크에 의해 서비스를 받으므로 큐의 길이가 상당히 짧다. 따라서 이것들의 크기는 시스템의 여유가 없을 때 작게 잡아도 무리가 없다.

6. 결론

현재까지 상용의 실시간 운영체제 하의 로봇 제어용 소프트웨어 구조에 대한 연구가 미흡하다. 제안된 로봇 제어기 소프트웨어의 구성 방법을 통하여 실시간 동작이 요구되는 로봇 제어 소프트웨어에 실시간 운영체제의 멀티태스킹 기능과 태스크간의 통신 기능이 성공적으로 적용할 수 있는 규격을 제시하여 누구나 쉽게 응용할 수 있다. 또한 운용되는 환경이 PC 기반의 도스용 실시간 운영체제이므로 저렴한 비용을 투자하고도 효과적인 로봇 제어를 위한 다양한 태스크들을 충분히 제시간에 스케쥴링이 가능하므로 비용 대 성능 면에서 상당한 장점이 있다. 또한 성능분석을 통하여 시스템의 성능을 미리 예측해 볼 수도 있다.

참고 문헌

[1] G. C. Buttazzo, M. D. Natale, "HARTIK: A Hard Real-Time Kernel for Programming Robot Tasks with Explicit Time Constraints and Guaranteed Execution", *IEEE International Conference on Robotics and Automation*, pp. 404-409, 1993.

[2] W. Garbe, *Visual SimNet 1.34 Manual*, 1995.

[3] P. A. Laplante, *"Real-Time Systems Design and Analysis: An Engineer's Handbook"*, IEEE Computer Society Press, 1992.

[4] *RTKernel 4.5 User's Manual*, On-Time Informatik GMBH.

[5] D. Schmitz, R. Hoffman, P. K. Khosla, T. Kanade, "CHIMERA: A Real-time Programming Environment For Manipulator Control", *IEEE International Conference on Robotics and Automation*, pp. 846-852, 1989.

[6] D. Simon, P. Freedman, E. Castillo, "Analyzing the Temporal Behavior of Real-time Closed-loop Robotic Tasks", *IEEE International Conference on Robotics and Automation*, pp. 841-847, 1994.

[7] J. A. Stankovic, Krithi Ramamritham, *"Advances in Real-Time Systems"*, IEEE Computer Society Press, 1993.

[8] D. B. Stewart, P. K. Khosla, "Rapid Development of Robotic Applications using Component-Based Real-Time Software", *IEEE International Conference on Intelligent Robots and Systems*, pp. 465-470, 1995.

[9] 이동익, "페트리 넷 이론의 기초", *정보처리 제 2권 제 2호*, pp. 76-84, 1995.