2000년도 대한전기학회 하계학술대회 논문집 2000.7.17-20

원격제어를 위한 임베디드 네트워크 시스템

이상협* 허원석* 권판조* 이호근** 이석규** 이달해*
*영남대학교 전기공학과 **해동정보통신

Embedded Network System for Teleoperation

S. H. Lee*, W. S. Heo*, P. C. Kwan*, H. G. Lee**, S. G. Lee**, D. H. Lee**
*Dept. of Electrical Engineering, Yeungnam Univ. **Headong Infomation & Communication Co., ltd

Abstract - 인터넷을 이용한 원격 제어에 있어서는 통신 프로토콜은 TCP/IP를 기반으로 이루어지고 있는데 대개의 경우는 소켓을 이용한 PC기반에서 해석되어 지고 처리되어 진다. 본 논문에서는 PC기반의 원격제어 시스템을 임베디드화함으로써 시스템의 안정성과 속도면에서 우수한 시스템을 구현하려고 한다. 이를 위하여 임베디드 시스템은 HD860-R3보드를 이용하였고 실시간 운영체제로 리눅스를 사용하였다. 실질적인 제어물의 제어를 위해서 제어기를 설계 HD860-R3보드와 시리얼을 통하여 통신하였다. 나아가 이 시스템을 PC기반의 시스템에서 탈피한 시스템을 홈 오토메이션의 제어 시스템에 적용시켜 보려 한다.

1. 서 론

원격 제어에 있어서 인터넷은 안정성과 편리성 그리고 누구나 접속이 가능한 용이성을 동시에 제공하므로 인터넷과 원격 제어와의 접목이 계속 시도되고 있다.[1][2][3] 또, 네트워크라는 인프라가 구축되어진 곳이라면 네트워크에 접속할 수 있는 모든 기기들간에 상호 작용을 통해서 다양한 서비스를 우리에게 제공해 주고 있다.[4] 그러나 이러한 네트워크는 데스크탑 컴퓨터의 네트워크 기능을 이용하여 이루어지고 있어 다양한 요구의 네트워크 기능을 수행하기가 쉽지 않다. 이에 네트워크 기능의 임베디드 시스템에 대한 연구가 활발히 이루어지고 있는데, 이러한 임베디드 시스템은 지능형 기기들의 원격관리가 가능해지고 부가 서비스를 지속적으로 추가할 수 있고, 임베디드 시스템에서 채택하고 있는 MPC(Main Processor Unit)의 성능향상으로 인해 임베디드 기기들의 컴퓨팅 능력이 커짐으로 인해 중요성이 더욱 증대되고 있다. 본 논문은 네트워크 임베디드 시스템인 HD860-R3를 이용하여 PC환경에서 이루어지던 원격 제어시스템을 임베디드 시스템하에서 구현하고 그 실효성을 검증해 보았다.

2. 본 론

2.1 실시간 리눅스

RT Linux는 리눅스에 Hard Real Time 기능을 부여하기 위하여 시작된 프로젝트이다. 리눅스 아래에 RT Linux 커널을 올리고 기존 리눅스 커널을 RT Linux 커널에서 돌아가는 하나의 프로세스로 만들어서 Real Time task와 기존의 리눅스 커널이 공존하는 형태를 만들었다. RT Linux 커널은 real time 프로세스를 생성하고 이들을 스케쥴링하며, 인터럽트가 발생하였을 때 이를 처리하며 리눅스와의 통신을 담당하는 역할을 한다. 기존 리눅스 커널은 그대로 일반 리눅스 프로세스를 관리하고, 자신의 인터럽트를 처리한다. RT Linux의 구현 특징을 살펴보면 다음과 같다.

▸ Linux OS Kernel Task : RT Linux는 기존 리눅스 커널을 RT Linux 상에서 돌아가는 하나의 task로 생각한다. 여기에는 가장 낮은 priority가 부여되므로, 실행할 수 있는 RT task가 하나도 없을 때 실행된다.

▸ Scheduling : RT Linux는 현재 두가지 scheduling 방법을 제공한다. 하나는 priority-based preemptive scheduler로서 모든 task에 priority를 부여하고, 실행할 수 있는 task 중에서 가장 priority가 높은 task를 수행하며, 이보다 더 높은 priority를 갖는 task가 등장하면 이를 곧바로 수행하는 것이다. 다른 스케쥴러는 earliest deadline first (EDF) 알고리즘을 사용하며, priority가 아니라 deadline을 기준으로 scheduling을 한다.

▸ Kernel Preemption : 기존 리눅스 커널은 선점되지 않는다는 가정 하에 만들어져 있다. 하지만 Real Time에서는 커널 작업 중이라도 이를 중단할 수 있어야 하는데, 현실적으로 커널을 재진입 가능하게(reentrant) 수정하는 것은 커널을 다시 설계해야 하는 문제가 있다. RT Linux는 이 문제를 RT task와 RT 인터럽트가 기존 리눅스 커널에서 제공하는 시스템 콜을 사용하지 못하게 함으로써 해결한다. 즉 RT 인터럽트는 기존 리눅스 커널과 무관하기 때문에 기존 리눅스 커널을 선점하더라도 문제가 되지 않는다. 하지만 RT Linux 커널 그 자체는 선점할 수 없도록 설계되어 있다. 이 커널은 아주 작고 빠르게 설계되어서 큰 시간 지연을 발생하진 않는다.

▸ Interrupt Blocking : RT Linux에서 인터럽트의 제어권은 RT Linux 커널에 있다. 기존 리눅스 커널은 인터럽트에 의해 중단되고 싶지 않은 경우, 하드웨어 인터럽트를 막을 수 있다. 이렇게 인터럽트를 막아버리면(cli) 그 사이 발생한 인터럽트는 인터럽트가 허용될 때까지(sti) 기다리고 있어야 하고(pending), 이는 인터럽트 반응 속도를 느리게 하는 요소가 된다. 이에 RT Linux는 기존 리눅스 커널이 인터럽트를 막는 것을 소프트웨어적으로 에뮬레이션을 하고, 실제로 인터럽트를 막을 수 없도록 한다. 리눅스 커널이 인터럽트를 막으면 별도의 플래그에 이를 표시해 두었다가, 실제로 인터럽트가 발생하면 이 플래그를 참조하여 인터럽트 가능 상태이면 인터럽트를 전달하고, 인터럽트 금지 상태이면 인터럽트가 발생했다는 표시를 하고, 인터럽트를 허용할 때까지 전달하지 않는 것이다. 이렇게 하여 기존 리눅스 커널의 입장에서는 자신이 인터럽트되지 않는 효과를 주고, 실제로 real time 인터럽트를 바로 처리할 수 있도록 한다.

▸ Interrupt 처리 : RT Linux에서 인터럽트는 기존 리눅스 커널의 관리하에 있는 인터럽트와, RT Linux 커널의 관리하에 있는 RT 인터럽트 두가지로 나뉜다. RT Linux는 RT 인터럽트를 위하여 request_RTirq(), free_RTirq() 함수를 제공한다. 인터럽트가 발생했

을 때 real time 인터럽트 핸들러가 등록되어 있다면 이를 불러주고, real time 인터럽트 핸들러가 없거나, 인터럽트를 기존 리눅스 커널과 공유하는 경우 이를 리눅스 커널에 전달한다.

- Resource : RT 커널은 리눅스에 메모리나 세마포어같은 자원을 요청하지 않는다. RT task는 동적으로 메모리를 요청할 수 없고, 리눅스 커널과 자료구조를 공유하면서 동기화를 해야하는 경우도 없다.

- Real Time FIFO : RT Linux를 이용하여 실제로 프로그램을 짤 때 프로그램을 RT task와 일반 리눅스 프로세스로 나누어서 하도록 한다. RT task에서는 할 수 있는 일이 극히 제한되기 때문에, 실제 RT 기능이 필요한 부분은 RT task에서 구현하고, 그렇지 않은 부분은 기존의 리눅스 프로세스로 만드는 것이다. 이를 위해 RT Linux는 이 두 프로세스가 서로 데이터를 교환할 수 있는 기능을 제공하는데, 이것이 Real Time FIFO이다. 이는 커널 주소공간에 공유 메모리를 이용하여 구현되며, 이에 대한 접근은 세마포어같은 것을 통하지 않고 atomic하게 이루어진다. RT task가 여기에 접근할 때 blocking을 하지 않고, 이 영역의 크기가 고정되어 있기 때문에 버퍼가 넘치거나(overflow) 부족하는(underflow) 현상이 발생할 수 있다.

2.2 시스템의 구성

그림 1은 전체 시스템의 구성도를 보여 주고 있다. 이더넷에 연결되어 있는 비디오 서버는 사용자에게 원격지의 상황을 모니터해준다. 그리고, 임베디드 시스템은 사용자의 명령을 받아서 제어기에 넘겨줌으로써 팬과 램프의 제어가 가능해 진다.

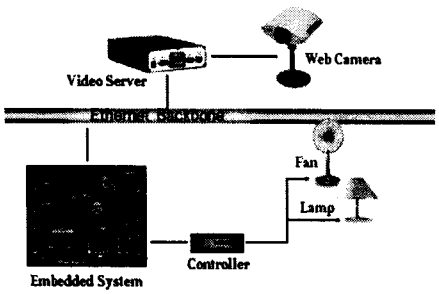

그림 1. 시스템 구성도

2.2.1 HD860-R3

HD860-R3는 다음과 같은 특징을 가지고 있다.

- MPC860CPU
- 2Mbytes 16-bit Flash Memory
- 16Mbytes 32-bit SDRAM
- 8Kbytes Serial EEPROM
- 10Base-T Ethernet Port
- RS-232 Port
- BDM Port
- 3.3V Operation
- H/W Reset Button
- Power, Run, Ethernet Status Monitering LEDs
- 5MHx X-TAL Operation

그림 2는 HD860-R3의 구조를 보여주고 있다.

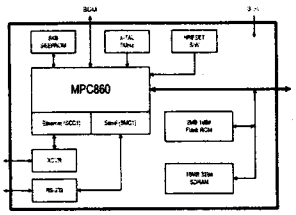

그림 2. HD860-R3의 구조

2.2.2 실시간 운영체제

실시간 운영 체제로는 RT-Linux가 사용되었다. 아직 완전한 실시간 OS로서의 기능을 하려면 많은 개선책이 필요하지만, TCP/IP와 웹 서버의 지원 그리고 안정성이 뛰어난 관계로 사용하게 되었다. RT리눅스는 MPC860 프로세서에 포팅되어 있다. 실제의 물리적 계층을 고려해 보면 아래 그림과 같이 RTOS층과 TCP/IP계층이 리눅스가 담당하고 여기에 JVM을 올려 Java 프로그램의 지원이 가능하게 된다. 그림 3은 HD860-R3보드에 RT-Linux를 탑재하기 위한 BDM프로그램을 보여준다.

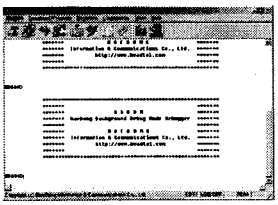

그림 3. BDM 프로그램

2.2.3 제어기

제어기는 실제로 피제어물을 제어하기 위한 시스템이다. 실제에 있어서는 팬과 전등을 직접 제어하게 된다. HD860-R3와는 시리얼을 통한 연결이 이루어진다. 마이크로 프로세서와 주변 메모리는 피제어물을 위한 버스 시스템을 통하여 연결하며 그 유연성과 확장성이 용이하도록 구성되었다.

2.2.4 애플릿 프로그램

애플릿은 자바의 특징인 플랫폼 독립성을 유지하며 안정성있게 동작할 수 있다. 자바 바이트 코드는 임베디드 시스템에 보관되고 임베디드 시스템과 호스트는 네트웍, 특히 인트넷으로 연결되어 진다. 임베디드 시스템에는 작은 웹서버가 내장되어 있어야 한다. 호스트 컴퓨터가 웹브라우저를 이용해 임베디드 시스템에 접속하면 임베디드 시스템 내의 웹서버는 자바 바이트 코드 즉, 자바 애플릿을 호스트 컴퓨터에 전송한다. 여기서 사용되는 애플릿은 임베디드 시스템을 제어하기 위한 것이다. 이때, 애플릿은 임베디드 시스템에서 수행하는 것이 아니라 단지 저장만 하고 있을 뿐이다.

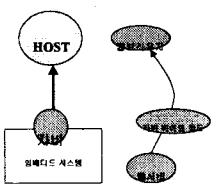

그림 4. 애플릿 구동 원리

그림 5는 HD860-R3의 RT-Linux에 아파치 서버를 탐재시켜서 외부에서 웹으로 접속한 그림이다. 웹을 통하여 사용자는 제어물을 제어하게 된다.

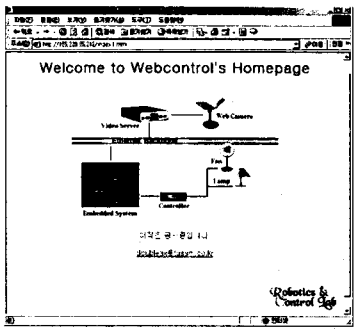

그림 5. 웹으로 HD860-R3보드에 접속한 모습

2.2.5 소켓 프로그램

본 시스템의 인터넷 통신 부분은 클라이언트/서버 모델의 소켓 통신을 사용하여 구축하였다. 이 프로그램은 실제 제어명령을 클라이언트에서 받아 제어기에 전달하는 역할을 한다. 이러한 중의 데이터의 손실과 오류를 방지하기 위하여 프로토콜이 필요하게 되고 프로토콜은 데이터와 에러검출부로 이루어진다.

3.실 험

본 논문은 해동 정보 통신의 HD860-R3보드를 이용하여 시스템을 구성하였다. HD860보드는 모토로라의 MPC860을 주 프로세서로하여 RT-Linux가 탐재된 시스템이다. 그림 6는 HD860-R3보드에 랜과 시리얼과 BDM을 연결한 그림이다. 이 시스템은 IP 어드레스를 가져 랜을 통해 외부에서 접속이 가능하며 시리얼을 통하여 제어기와 연결된다.

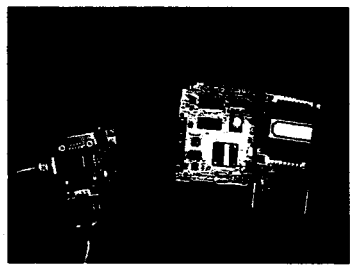

그림 6. HD860-R3보드

4.결 론

본 논문에서는 그 동안 PC기반에서 이루어져오던 원격제어 시스템을 임베디드 시스템으로 구현하였다. 이를 구현하기 위해서 RT-Linux를 사용하였고, Linux 기반 위에서 사용자와의 통신을 위한 소켓 프로그램, 제어기와의 통신을 위한 시리얼 프로그램을 구현하였다. 또, 인터넷을 통한 제어를 위해서 자바 기술을 사용하였다. 이러한 시스템은 네트워크가 필요한 모든 기기 특히 가정용 기기에 연결되어 웹을 통한 제어가 가능하도록 할 것이다.

[참 고 문 헌]

[1] K.Goldberg, M.Mascha, S.Genter, C.Sutter, N.Rothenberg, J.Wiegley, "Desktop teleoperation via the World Wide Web", Proc.IEEE Int, Conf.Robotics and Automation, Nagoya, JAPAN, May, 1995.

[2] 이명진, 문재철, 강순주, "인터넷상에서 WWW을 이용한 무선 비행체 원격 제어", 제어·자동화·시스템공학회 합동학술 발표회 논문집, pp191-195, 1998.

[3] Eric Paulos, John Canny, "Delivering Real Reality to the World Wide Web via Telerobotics, Proc.IEEE Int, Conf .Robotics and Automation,Minneapolis, Minnesota, April, 1996

[4] Sun Microsystems, Java Remote Method Invocation Specification bata draft, Dec. 1996

[5] Java WhitePaper - Moving from the Console to the Web in Real Time, March, 1997.

[6] Engineering Web Technologies for Embedded application, IEEE Internet Computing, May June, 1998.

[7] Elliotte Rusty Harold, Java Networking Programming, O'reilly.

[8] Rob Gordon, Essential JNI java native interface, Prentice Hall PTR

[9] http://java.sun.com

[10] http://www.aglance.com