실시간 리눅스 기반 개방형 로봇 제어기 설계

이금수, 문승빈
세종대학교 컴퓨터공학과
e-mail: sbmoon@sejong.ac.kr

Design for Open Architecture Robot Controller Based on Real Time Linux

Keumsu Lee and Seungbin Moon
Dept. of Computer Engineering, Sejong University

요 약

본 논문에서는 PC 바탕에 리눅스 기반의 개방형 로봇 제어기 설계를 제안 하고자 한다. 본 논문에서 제안한 로봇제어기는 하드웨어적으로는 PC, Motion Board, Motor Driver등을 필요로 한다. 그리고 소프트웨어적으로는 Motion Board 장치에 대한 Device Driver, Path Planning, 로봇언어, GUI 등을 필요로 한다. 소프트웨어적인 요소들은 모두 리눅스 상에서 구현하게되며, 최종적으로는 RTOS를 적용하게 된다. 로봇 제어기는 시스템의 수행결과가 기능적으로 정확해야 할 뿐만 아니라, 결과가 도출되는 시간 역시 주어진 제약 조건을 만족시켜야 하는 시스템이기 때문이다.

1. 서론

현재 전 세계적으로 많은 기업과 연구 단체들이 개방형 제어기를 구현하기 위해 많은 노력을 기울이고 있다. 유럽에서는 독일의 Stuttgart대학 연구소가 중심이 되어 유럽의 8개 기업과 3개의 연구소가 참여하여 FA용 제어장치의 개방화에 관한 OSACA 프로젝트를 수행하고 있다.[1] OSACA는 제어기능에 대한 객체화, 객체간의 인터페이스 표준안 제시 등 가장 활발한 활동을 하고 있다. 그러나 사용하기에 너무 어렵고, 객체간의 인터페이스가 지나치게 계층화되어 있어서 효율성이 떨어져 실시간 성능에 대한 보장이 힘들다는 단점이 존재한다. 일본을 중심으로 하는 OSEC[2]는 여러 개의 제어기 제조업체들간의 제어언어의 호환을 위한 언어인 FADL를 개발하였으며 중점적으로는 사용자의 편이성을 위해서 CAD/CAM과의 연동을 위한 연구를 수행하였다. 미국에서는 자동차 업체를 중심으로 한 단체인 OMAC[3]에서 모듈구조의 개방형 제어기 개발에 관한 연구를 하고있으며, 제어기에 필요한 여러 가지 API를 개발하였다.

관련 업체로서는 Cimetrix와 Kuka등이 있는데, 특히 Cimetrix사의 경우는 IEEE/NEMI의 표준안을 적용하여 로봇 제어기뿐만 아니라 다른 제어기로도 활용할 수 있는 개방형 제어기를 상용화하였다.[4]

국내에서도 DSP를 기반으로 한 PLC 제어기를 구현하고자 하는 연구가 시도되고 있다.[5] 이처럼 개방형 제어기 개발을 위한 연구는 세계 여러 곳에서 많이 이루어지고 있지만, '개방성' 이라는 내용에 대해 각기 추구하는 연구의 방향은 조금씩 다른 면을 가지고 있다. 본 논문에서는 PC에 실시간 리눅스를 기반으로 하는 개방형 로봇 제어기를 제안하고, 개방형 로봇언어와 GUI를 Lex & Yacc와 QT-Graphic Library를 사용하여 각각 구현한다.

2. 개방형 제어 시스템

개방형 제어 시스템을 분류하기 위해서는 시스템

에 대한 구조적인 분석을 할 필요가 있다. 일반적으로 시스템의 구조는 응용 프로그램, 시스템 소프트웨어, 하드웨어의 3계층으로 나뉘어 질 수 있다.[4] 응용 프로그램은 사용자가 원하는 최상위 기능을 구현하기 위한 부분이며 제어 시스템에서 제어 알고리즘, 사용자 인터페이스 등에 해당하는 부분이다. 시스템 소프트웨어 계층은 응용 프로그램 계층의 하부 계층으로 응용 프로그램을 관리하기 위한 기본적인 통신 프로그램이나 운영체제 등의 소프트웨어를 말한다. 하드웨어는 실제로 시스템을 구동하기 위한 프로세서나 메모리 등의 물리적인 장치를 칭하게 된다. 그림 1에서 살펴보면 응용 프로그램과 시스템 소프트웨어 사이에 이들의 인터페이스를 담당하는 API 가 필요하고 시스템 소프트웨어와 하드웨어와의 인터페이스를 담당하는 디바이스 드라이버가 필요하다. 보통 개방형 제어기 연구는 하드웨어와 시스템 소프트웨어 계층까지는 사실상의 표준으로 인정되는 PCI나 VMEbus등을 사용하고 주로 상위의 응용 프로그램과 API까지의 개방화에 대한 연구를 지칭하게 된다.

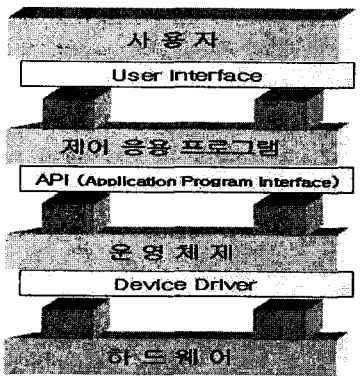

그림1 로봇 시스템의 구조도

3. 로봇제어기의 하드웨어 구조

본 논문에 사용된 로봇제어기의 하드웨어 구조는 크게 4부분으로 나누어 볼 수 있다. 상위제어기 역할을 담당하는 PC가 있고, 그리고 PC의 연산결과를 이용해 실질적으로 하드웨어를 컨트롤하는 Motion Board, 그리고 Motion Board의 값을 모터에 전달하기 위해 필요한 모터드라이버 그리고 마지막으로 모터드라이버의 지령에 따라 실질적인 동작을 하는 로봇으로 구성되어 있다.[6] 본 논문에서는 로봇제어기의 하드웨어 구성요소 중 모션보드와 로봇의 전체적인 구조에 대해서만 살펴보겠다.

3.1 모션보드

최근의 산업용 FA제어기는 종전의 전용기 형태에서 범용제어기 형태로 바뀌어 가고 있다. 이 같은 추세에 맞춰 Motion 제어기 또한 PC-Based Open Architecture 구조로 가는 추세이다. 그림2는 아경산업의 AMC-PC41A-PCI 보드의 전체적인 모습을 보여준다. AMC-PC41A-PCI 보드는 PC Based Slot형태의 Open Architecture 구조로 설계되어, FA-Computer 및 일반 PC에도 장착하여 Motor의 종류에 관계없이 Motion 제어가 가능하도록 설계되어 있으며 특징으로는 Half Size로 설계되었으며 TMS320C31 DSP를 Processor로 사용하여 고정도의 Motion 제어가 가능하며, Host(Computer)와 AMC 보드간에 원활한 통신을 위해 DPRAM (Dual Port RAM)을 이용함으로서 고속통신이 가능하게 설계되어 있다.

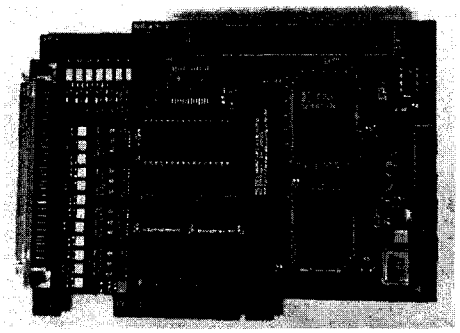

그림2 모션보드 (AMC-PC41A-PC)

3.2 전체 시스템구성

그림3은 로봇 제어기의 전체 시스템 구성도를 나타낸다. 상위제어기로 사용한 Industrial PC는 Half Size PC로서 어드벤텍의 PCI-6771 모델을 사용하였다. PCI-6771은 펜티엄3 프로세서를 사용하고 있으며 최대속도 850MHz까지 지원한다. PCI-6771은 PCI 인터페이스 방식을 사용하고, 2개의 RS-232 시리얼포트를 가지고 있으며, 1개의 병렬포트와 USB포트를 가지고 있고, VGA컨트롤러와 네트워킹 컨트롤러를 내장하고 있다. 로봇의 동작방식을 살펴보면 PC안에 내장된 모션보드에서 모션에 관련된 지령을 생성하고 이 지령을 Servo Driver를 통해 Motor로 전달하는 구조를 가진다. 그림3에서 보는 것처럼 PC와 로봇이 통신하기 위해서는

PCI 타입의 모션보드가 필요하며, 그리고 모션보드를 동작시키기 위해서는 모션보드에 대한 디바이스 드라이버가 필요하다. 개방형 제어기에서는 스트림 기반의 데이터를 통신하므로 Character Device Driver[7]를 사용하였다.

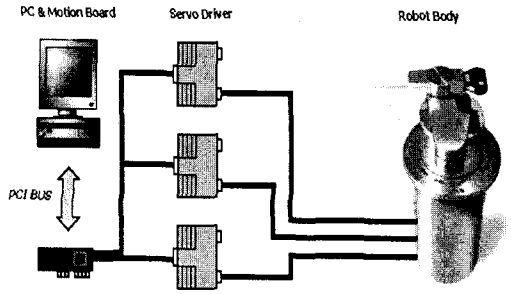

그림3 로봇 제어기의 시스템 구성도

4. 로봇언어

로봇 언어는 사용자와 로봇간에 사용되는 대화 수단으로 교시된 위치 또는 변수 조작을 통해 로봇을 동작시키는 언어를 말한다. 로봇 언어의 사용은 시스템의 구조적인 변경 없이 단지 사용자가 로봇 언어를 이용하여 작성된 프로그램을 실행 시켜 다양한 작업을 구현할 수 있게 한다. 로봇 언어가 필요한 것은 로봇이 단순한 작업을 벗어나 보다 복잡하고 다양한 작업을 구현하기 위해서 필요하다. 개방형 로봇 언어가 필요한 것은 많은 로봇 언어들은 이식성을 고려하지 않게 만들어지기 때문에 만약 다른 로봇 언어를 작성한다면 새로운 작업을 해야 한다.[6] 이에 개방형 로봇 언어는 이식성을 고려한 다양한 로봇 언어를 구사하여 이식성을 높이기 위해 제작되었다. 로봇 언어 Interpreter는 Lex & Yacc로 Compile 되었다.[8]

5.1 어휘 분석기 생성 툴 lex

Lex는 어휘 분석기(lexical analyzer, lexer)를 만드는 도구이다. 어휘 분석기는 주어진 입력 스트림을 의미 단위인 token으로 분리하는 작업을 수행한다. Lex자체로는 실행 화일을 생성하지 않고 주어진 lex specification(lex가 주어진 입력에 대하여 match하는 pattern과 action의 집합)을 수행하는 함수 yylex()포함하는 file(lex.yy.c)로 변환하는 작업을 수행한다. 그러므로 작성한 어휘 분석기를 수행하고자 한다면 main()함수 내에서 yylex()를 호출하면 된다.

5.2 구문 분석기 생성 툴 Yacc

Yacc는 컴파일러 생성을 위한 프로그램을 compiler generator 또는 compiler compiler라고 부르는데, Yacc가 BNF와 같은 형식의 rules의 항목들로부터 parser를 만들어내는 프로그램이다. (Yacc입력은 BNF를 간단하게 만든 버전이다.) yacc은 lex에 의해 구해진 token간의 relationship을 구성하는 구문 분석기를 생성하는 툴로서 lex의 상위 단계에 위치하며 lex가 생성한 어휘 분석기 yylex()를 내부적으로 호출한다. 즉 yacc specification 파일 내에서 user routine내에서 yacc가 생성하는 구문 분석기 루틴인 yyparse()를 호출하면 yyparse()는 내부적으로 yylex()를 호출하여 필요한 token을 요구하고 리턴되는 token들을 받아가면서 주어진 문법에 부합하는지를 검사한다. 문법이란 token간의 relationship에 대한 정의이며, 주어진 문법 규칙에 입력 토큰들이 일치하는 지를 확인하는 과정을 파싱(parsing)이라고 한다. 로봇언어의 전체적인 구성은 다음과 같다.

-Typed Data
  · INT, REAL, LOC
-Sequent Control
  · *IF, ELSIF, ENDIF*
  · FOR, ENDFOR
  · WHILE, ENDWHILE
  · GOTO
  · DELAY
  · FUNC, ENDFUNC
- Motion
  · MOVE, SPEED
- Location
  · HERE, LCOPY, LSAVE, LSET
- I/O
  · IFSIG, SIGOUT, SIGIN, WAIT

5. GUI

X 윈도우 상에서 동작하는 애플리케이션을 만드는 방법은 크게 보아서 GNOME 데스크탑에서 사용하는 GTK+(gimp Tool Kit)와 KDE 데스크탑에서 사용하는 Qt로 나눌 수 있다. 가장 큰 차이는 Qt는 클래스 화 된 라이브러리이고 GTK+는 C중심의 개발 환경을 제공한다는 점이다.[9] QT는 완벽한 객체지향 구조로, 모든 위젯과 대화상자들이 C++ 객체

이며 상속을 이용하여 새로운 위젯의 생성이 쉽고 자연스럽다. 또한, 시그널/슬롯 메커니즘은 진정한 컴포넌트 프로그래밍을 가능하게 해주는데, 이런 재사용 가능한 컴포넌트들은 서로에 관한 정보교환 없이 완전하게 같이 동작될 수 있다.[9]

본 논문에서 구현한 개방형 로봇 시스템 GUI는 표준 윈도우 시스템인 X 윈도우 환경에서 개발되었으며, QT-Graphic Library를 사용하였다. GUI환경은 기존의 텍스트환경이나 단일 기능의 GUI환경에서 탈피하여 다기능의 통합 그래픽 사용자 환경을 제공 함으로써, 사용자는 더욱 편리해진 환경에서 로봇을 제어할 수 있으므로 사전의 많은 교육 시간이나 노력을 줄일 수 있으며, 로봇 사용자에게 직관적이고 편리한 환경을 제공하며 그와 동시에 사용자가 행한 동작에 대해 확신을 가질 수 있도록 작성했다. 전체적인 GUI구성을 그림4 에 나타내었다.

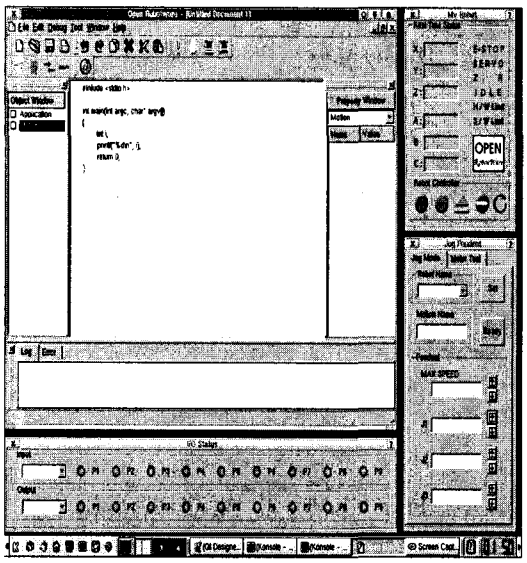

그림 4 QT로 작성한 GUI

6. 결론 및 향후 연구과제

개방형 제어기를 구현하기 위해 여러 기업과 연구 단체들이 많은 노력을 기울이고 있으며 개방형 제어기에 관한 표준을 만들어 가며 활발히 연구하고 있다. 본 논문에서 작성한 GUI는 유연성과 확장성을 고려하여 디자인 하였고 그리고 로봇언어 또한 시스템의 구조적인 변경 없이 단지 사용자가 로봇언어를 이용하여 작성된 프로그램을 실행 시켜 다양한 작업을 구현할 수 있게 하였다.

향후 과제는 본 논문에서 제시한 개방형 제어기를 기반으로 개방형 제어기를 연구하는 여러 단체에서 제시하는 표준에 근거하여 이식성, 확장성, 재사용성 등을 고려한 개방형 제어기를 개발하여 다양한 형태의 로봇에 적용시킬 수 있게 하는 것이 최종적인 목표가 될 것이다.

참고문헌

[1] G. Haidegger, J. Nacsa, "Shop-floor communication with OSACA-compliant controllers", IEEE Int Workshop on Factory Communication Systems, pp.355-362, 1997.

[2] C. Sawada, O. Akira, "Open controller architecture OSEC-II: architecture overview and prototype systems", 6th Int Conf on Emerging Technologies and Factory Automation, pp.543-550, 1997.

[3] S. Kolla, J. Michaloski, W. Rippey, "Evaluation of component-based reconfigurable machine controllers", Proceedings of the 5th Biannual World Automation Congress, pp.625-630, 2002.

[4] Software Solutions for Machine Control and Equipment Connectivity http://www.cimetrix.com.

[5] 김형석, 장래혁, 권욱현 "고속 프로그램형 논리 제어기 구현을 위한 래더 다이어그램 해석 방법", Journal of Control Automation and Systems Eng, vol. 5, no. 1, pp. 33-38. Jan 1999.

[6] 신주호, 문승빈 "리얼타임 리눅스 기반 개방형 로봇 제어기", 정보처리학회 2002년 추계학술발표 논문집, pp.505 - 508, 2002.

[7] Alessandro Rubini, Jonathan Cobert, *Linux Device Drivers, 2nd ed., O'reilly,* 2001

[8] John R. Levine, Tony Mason, Doug Brown, *lex & yacc, O'reilly,* 1995

[9] Matthias Kalle Dalheimer, *Programming with Qt, O'reilly,* 2002