리얼타임 리눅스기반 개방형제어기 개발

Development of Real time Linux-based Open Architecture Controller

조 영 준*, 강 희 석, 김 영 진, 강 성 복, 서 연 곤
(Young June Cho, Heui Seok Kang, Young Jin Kim, Sung bok Kang, and Yeon Gon Seo)

Abstract : In the manufacturing industries, agility and flexibility are the key issues to meet the various market demands. From this point of view, open architecture control has many advantages. This paper suggests a real-time Linux-based open architecture controller. Linux-based and windows-based controllers are compared in several aspects, and Linux is shown to be advantageous over the windows. With the user friendly GUI, the suggested controller is applied to the X-Y stage which is made of two linear motors. Path following and repeatability performances are successfully observed, which shows the validity of the suggested controller.

Keywords : real time linux, windows platform, module, open architecture controller.

I. 서론

산업화 및 자동화가 가속되어감에 따라 우수한 제어성능을 가진 전용 제어기들이 개발되었다. 그러나 이들 제어기들은 특정한 용도와 목적을 수행하기 위한 전용제어기로서의 성향을 지니고 있으며, 일부 제어기 개발업체의 독점적인 기술로서, 고객의 다양한 요구에 신속하게 대응하기가 곤란하였다. 또한, 다양한 종류의 시스템 환경에 대한 적응성 및 호환성, 프로그램의 모듈화와 교체가능성의 관점에서 볼 때 기 구성된 제어기들을 통합하거나 수정할 경우에 추가적인 비용과 노력이 필요하였다. 개발환경에 있어서도 윈도우 OS를 기반으로 하고 있어, 다양한 기능을 구현하기 위하여 추가한 소스코드들이 실 시간성이 중요시되는 시스템에서는 오히려 기능저하로 작용하였다.

따라서, 본 논문에서는 윈도우 OS기반이 아닌 리얼타임 리눅스 기반의 개방형제어기를 개발하여, 윈도우에 대한 리눅스 운영체제의 장단점을 비교하였다. 또한, 개방형 제어기를 위해 필요한 모듈화와 실 시간성에 대하여 비교, 리눅스 사용의 타당성을 확인하였다. 그리고, 개발한 개방형 제어시스템(X-Y Stage-Linear Motor 2축)에 적용하여 정밀 위치제어 및 위치제어 반복성 실험을 통하여 그 효율성을 확인하였다.

II. 본 론

1. Real Time Linux vs. Windows 개발환경 비교

1.1 개방형 제어기를 위한 안정성

윈도우 시스템에 대한 대안으로 차세대 운영체제의 기수로 리눅스가 거론되고 있는 이유로는 Windows OS의 불필요한 기능 추가로 인한 Code Bloat현상이다. 즉 불필요한 소스코드 증가(Windows 2000은 약 35 to 40 million line)로, 실시간성이 중요시되는 개방형제어기를 개발 시 동작시간의 지연으로 작용하였다. 이에 반해 리얼타임 리눅스는 커널 컴파일을 통하여 사용자의 시스템에 맞도록 리눅스를 재구성할 수 있어, 플로피 디스크와 같은 공간에도 리눅스를 포팅할 수 있다. 이로 인하여 임베디드 시스템에서 리눅스가 많이 이용되고 있다.

1.2 개발환경의 유연성 비교

개방형 제어기가 대두된 이유로 고객 요구의 다양성과 Life cycle 단축에 따른, 유연하고 신속한 제품개발의 필요성을 들 수 있다. 그 동안 윈도우 플랫폼을 이용한 이유로는 사용자 친화적인 인터페이스와 다양한 서비스 제공, 교육비용의 저렴성 등이었다. 그러나, 리눅스도 지속적인 개발로 이제는 이러한 장점들을 누릴 수 있게 되었다. 그림 1은 Windows 개발환경인 Visual Studio의 screen shot이고, 그림 2는 리눅스의 개발환경인 K-develop의 screen shot이다. 지원하는 언어는 두 개의 개발환경 모두 C와 C++이다. 리눅스의 개발환경이 윈도우의 개발환경과 거의 차이가 없음을 알 수 있다. 리눅스는 Windows에는 거의 없는 터미널을 이용한 원격 디버깅 환경을 구축할 수 있어, Cross-Development 환경을 구축할 수 있다.

그리고, 리눅스의 소스는 공개되어 있어, 개발 대상이나 적용 환경에 따라 소스 수정이 가능하여, 좀더 유연하고 신속하게 적응할 수 있다.

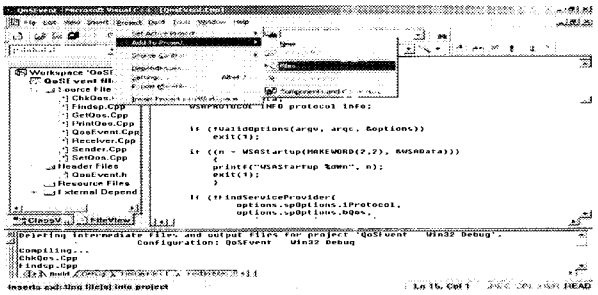

그림 1. Windows의 개발환경 Visual Studio.
Fig. 1. Visual Studio screenshot.

* 책임저자(Corresponding Author)
논문접수 : 2002. 9. 28., 채택확정 : 2003. 7. 3.
조영준, 강희석, 김영진, 강성복 : 한국생산기술연구원
(choyj@kitech.re.kr/hskang@kitech.re.kr/yjk574@kitech.re.kr/sbkang@kitech.re.kr)
서연곤 : 광운대학교제어계측공학(hamsum@hanmail.net)

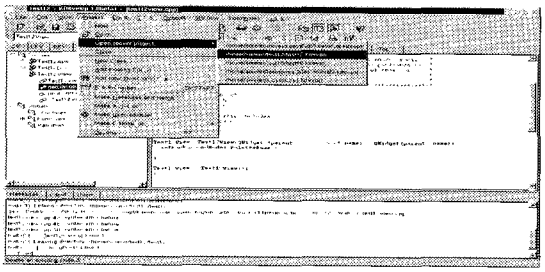

그림 2. Linux 개발환경 - Kdevelop.
Fig. 2. Kdevelop screenshot.

표 1. 프로그램 모듈화 및 교체가능성.
Table 1. Program Module and Pluggable.

OS	Code	비 고

Windows	HINSTANCE hInst;
	bool (*pfn)(double, double);
	hInst = LoadLibrary("MotionRun.dll");
	if(hInst == NULL){
	AfxMessageBox("DLL을로드할 수없습");
	return;
	}
	pfn = GetProcAddress(hInst, "CxMoveRobot");
	if(pfn == NULL){
	AfxMessageBox("함수를찾을수 없습");
	return;
	}
	pfn(x좌표, y좌표);
	FreeLibrary(hInst);
	}
		DLL (Dynamic Link Library)

Linux	void* handle;
	bool (*pfn)(double, double);
	char *error;
	handle = dlopen("/lib/MotionRun.so", RTLD_LAZY);
	if(!handle){
	fputs(dllerror(), stderr);
	exit(1);
	}
	pfn = dlsym(handle,"CxMoveRobot");
	if((error = dlerror()) != NULL){
	fputs(error, stderr);
	exit(1);
	}
	pfn(x좌표, y좌표);
	dlclose(handle);
		Dynamic Library이용 //// ELF (Excutable Library Formats)

1.3 개방형 제어기를 위한 리눅스 컴포넌트

1) 프로그램 모듈화 와 교체가능성(Pluggable)

Windows Platform상에서 프로그램을 모듈화로 구성 시 DLL를 이용하여 구성한다. 즉, 각각의 기능을 모듈화 하여 필요시마다 메모리에 로드하여 쓸 수 있고, 또한 실행 시에도 다른 모듈로 교체할 수 있다. 표1 에서는 윈도우즈와 리눅스에서의 동적 라이브러리의 실제 프로그램 소스를 보여주고 있다. 두 플랫폼에서 모듈화를 위한 동적 라이브러리 구성이 같음을 알 수 있다. 서로 다른 점이 있다면 리눅스에서는 커널 자체도 모듈화 되어 있어서 실행 중에 필요한 커널 모듈을 커널 모드로 로드시킬 수 있다. 그러나, 윈도우즈에서는 한 번의 rebooting이 필요하다. 따라서, 개방형 제어기에서 필요한 하드웨어를 실행 중에 메모리에 올리는 것은 윈도우즈에서는 불가능함을 알 수 있다.

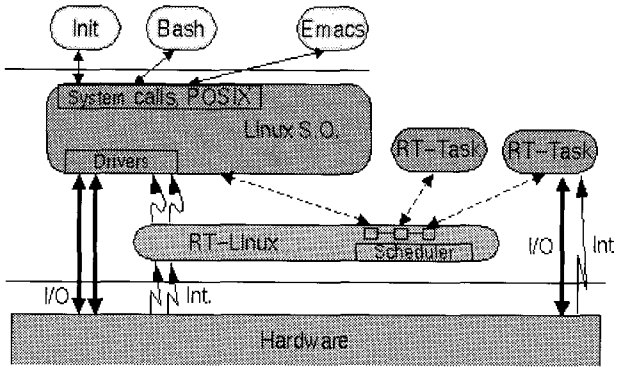

그림 3. RTLinux(hard real time) 구성도.
Fig. 3. RTLinux Architecture.

2) Real Time Linux

개방형 제어기에서 꼭 필요한 기능 중의 하나가 실시간 기능이다. 윈도우즈와 리눅스, 두 운영체제 모두 엄격한 실시간 수행은 불가능하다. 따라서, 각 운영체제는 기본 운영체제 커널 외에 실시간 커널을 따로 제시하여 필요한 곳에 쓰도록 하고 있다. 윈도우에는 대표적으로 RTX등이 쓰인다. 리눅스에서도 실 시간성을 위하여 여러 가지 프로젝트가 수행중이며, 그 중 대표적인 것이 Real Time Linux로서 그림3과 같이 기본 리눅스 커널과 Hardware 사이에 RT-Linux 커널을 추가하여 좀더 Real Time에 접근하는 방법으로 실 시간성을 구현하고 있다. 본 논문에서도 이처럼 RT-Linux를 이용함으로서 Hard Real Time에 좀더 접근할 수 있었으며, 개방형 제어기의 응답시간을 줄일 수 있었다. RT-Linux의 시스템 구성도 는 그림 3과 같다.

2. Software 구성 및 개발

최근 자동화 시스템들을 자유롭게 구성해서 사용하려는 요구의 증가로 미국의 NIST에서 1999년에 EMC (Enhanced Machine Controller) S/W과제가 수행되었다. EMC는 OMAC(Open Modular Architecture Controller)의 한 부분으로 수행되었기 때문에 개방형 제어기의 개념을 정확히 구현하였다. 또한 EMC는 Linux, Windows NT, SUN Solaris와 같이 다양한 OS platform에 적용되었다. 그리고, 소스코드를 공개하여서, 수정하거나 바꾸어서 다양한 개방형 제어기에 적용할 수 있게 하였다. 그림 4는 미국의 NIST(National Institute of Standards and Technology)에서 개발한 EMC S/W의 구조이며, 구성은 Motion Controller, I/O Controller, Task Executor, GUI의 4가지 요소로 구성되어있다.

2.1 Motion Controller

모션제어기 구조를 그림 5에 나타내었다. Task Executer에서 제공된 명령을 수행하기 위해 경로를 생성하고, 축별로 할당된 움직임을 계산하여 각축을 제어한다. 또한 내부적인 모션 제어 루틴이 최고 500 μsec를 주기로 동작한다. 이 수치는 기존의 상용제어기가 아닌 개방형 제어기를 구현하려 할 때 Real-Time Linux를 운영체제로 사용하므로 가능할 수 있다.

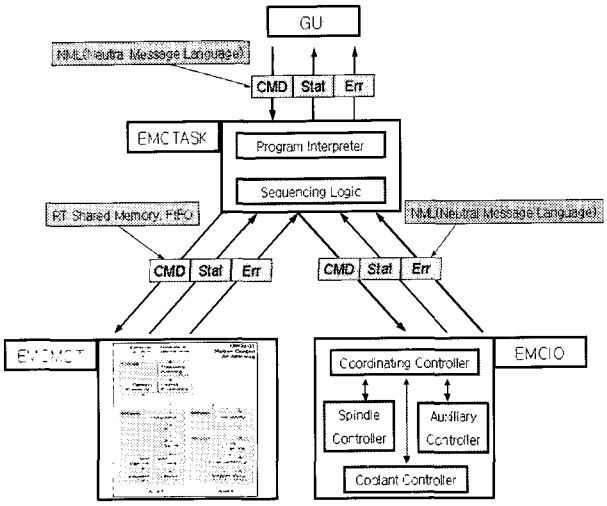

그림 4. EMC SoftWare 구성.
Fig. 4. Config. EMC SoftWare.

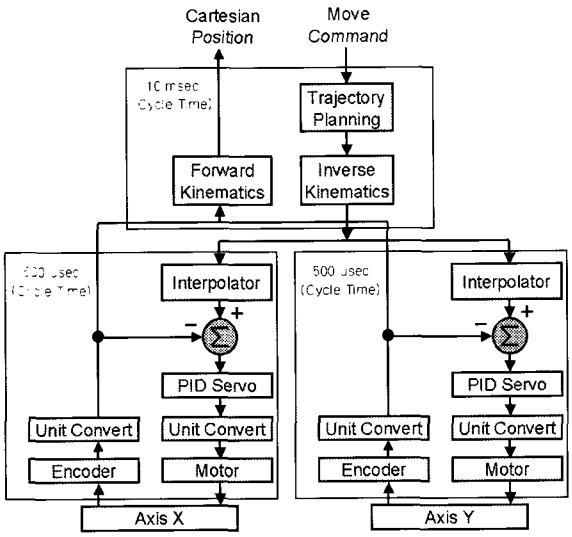

그림 5. 모션제어기 구조.
Fig. 5. Motion Control Architecture.

2.2 I/O Controller

I/O Controller는 병렬포트를 통해 각종 장치들을 제어한다. 그림 5에서 보듯이 Spindle, Coolant(Mist, Flood)그리고, E/STOP 및 Spindle Brake등을 ON/OFF 하거나 부가 장치들을 제어한다.

2.3 Task Executor

Task Executor는 프로그램을 해석하고 순서대로 수행하도록 Motion Controller와 I/O Controller에게 명령을 전송하고 각 축별 모션제어 및 입출력장치들의 작동상태를 전송 받아 GUI에서 표시할 수 있도록 메시지를 전달하는 역할을 한다.

2.4 G.U.I.(Graphic User Interface)

사용자에게서 받은 명령을 Task Executor에 전달하거나 Task Executor 상태를 사용자에게 보여주는 Inter- face 부분이다. Text 모드에서 직접 Command 입력에 의해서 구동될 수도 있고, Linux의 X-Window에서 구동될 수도 있다. 그림 6은 X-Window 기반 하에서 개발한 GUI를 나타내고 있다.

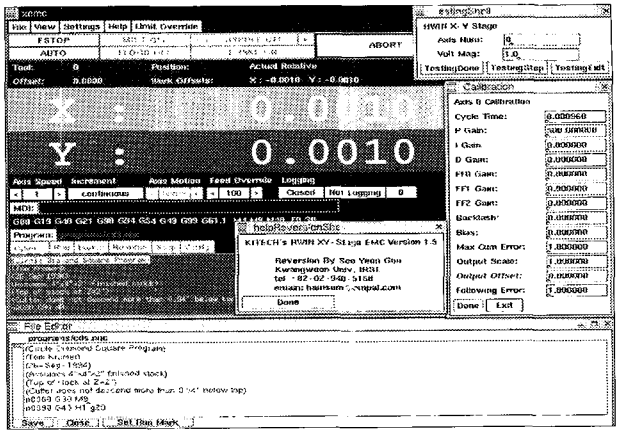

그림 6. X-Window 기반 GUI.
Fig. 6. GUI on X-Window based.

GUI의 기능은 Manual로 G-Code를 입력하여 각축 및 I/O Controller에 연결된 장치들을 직접 제어할 수 있는 MDI(Manual Data Input) Mode와 작성된 프로그램을 자동으로 실행시킬 수 있는 Auto Mode가 있으며, 리니어 모터로부터 피드백 되어진 각 축의 위치정보를 ㎛단위로 표시할 수 있는 기능이 있다.

제어기의 변수(P.I.D. F/F, F-Error)를 Motor Driver와 EMC S/W에서 각각 설정해주어야 하는데 이를 간편하게 설정할 수 있는 Calibration Mode 기능을 추가하였다. 또한, G-Code 프로그램에 의하지 아니하고 Velocity Command로 Motor Driver를 통하여 부하를 직접 제어할 수 있는 Testing shell 기능도 추가되어 사용자가 좀더 용이하게 접근하도록 구성하였다.

3. Hardware 구성

전체 시스템의 블록선도를 그림 7에 나타내었고, 그림 8과 같이 개방형 제어 시스템을 구성하여 실험하였다. PC에는 Interface장치가 없어 Servo To Go사의 Interface Card (STG I/F V2 8 Axis)를 PC 내부에 ISA Slot를 통하여 내장시켜 각 축의 위치제어를 위한 Command를 생성 시켰으며, Motor Driver로부터 전송되는 위치정보 Data를 받아 EMC Task에 전달할 수 있도록 Interface 역할을 하였다. D/A Converter의 출력전압은 그 가변범위가 ±10V로 이를 Motor 구동용 Driver의 속도 모드의 제어 Command로 공급하였으며, 극성에 따라 정, 역 방향으로 방향을 전환시켰다. 본 시스템에서는 2축 리니어 모터를 사용하였으며, 각 단계별 작동상태의 보정을 Step by step Mode를 이용 순차적으로 제어상태를 확인, 보정하는 방법을 사용하였다. Motor구동을 위한 제어기로는 HIWIN(LMDS3 X 2 EA)Driver를 사용하였으며, 부하는 Linear Motor (X축:2상x2, Y축:3상x1)를 사용하였다. 그리고 STG I/F Card가 Velocity Mode만을 지원하므로 Pulse Mode로는 제어할 수 없어, Velocity 출력을 Pulse로 변환시킬 수 있는 Velocity to Pulse Converter를 설계하여 Pulse Mode에서도 제어가 가능하게 하였다. 그리고 위치 제어성 실험의 Data 획득을 위하여 Pen-Plotter 형태의 지그를 제작하여 실험결과를 출력하였다. 또한, 특성검사기를 제작하여 I/O 장치들의 작동 상태를 시각적으로 확인하였다.

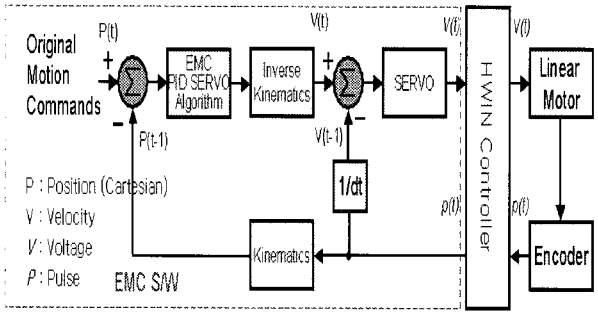

그림 7. 시스템 블록선도.

Fig. 7. Block Diagram for system.

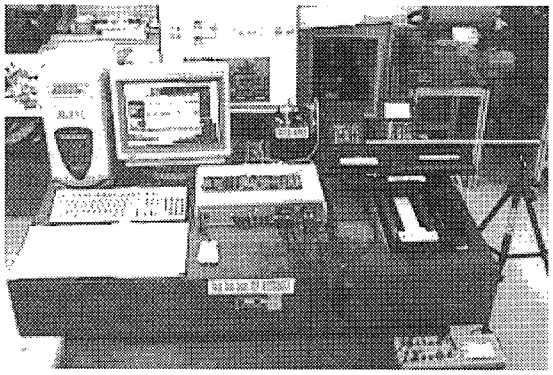

그림 8. 개방형 제어시스템 구성.

Fig. 8. Config. of open architecture controller.

III. 실험 및 고찰

1. 실험 목적

리얼타임 리눅스기반 개방형제어시스템을 구축하여, 전용제어기와 부하를, 본 시스템에 범용제어기와 부하로 접목시켜 제어를 실현하여, 전용제어시스템에서 제시하는 제어특성을 본 개방형제어시스템에서도 만족하는가를 확인하기 위하여 본 실험을 실시하였다. 즉, 제어특성에 영향을 주지 않고 동등한 제어가 가능한지 여부를 평가하기 위하여 실험을 실시하였다.

2. 실험 항목

2.1 위치제어 특성실험

전용 제어시스템의 개방화에 따른 오차의 증가 및 누적 등의 영향을 평가하기 위하여 G-Code Program을 작성, 제어목표 좌표에 대한 추종 결과치를 출력하는 방법으로 그림 11과 같이 Square, Diamond, Cross Line과, Circle, Arc, Spline 형태로 위치제어특성을 확인하였다. 위치좌표 기본단위는 ㎛로서 전용제어기에서 제시하는 공차는(±2um)이다. 실험결과 전용제어기 공차 범위 내에서 임의좌표 지점으로 위치제어가 가능하였다. 제어정밀도는 전용부하 및 구동용 Driver의 정밀도에 의존되었고, 개방화함에 따른 제어정밀도의 증가 및 변화는 없었다.

2.2 위치제어 반복성(Repeatability)실험

전용 제어시스템을 개방형 제어 시스템으로 변경하였을 경우의 위치제어 반복에 따른 영향을 평가하기 위하여 본 실험을 표 2와 같이 실시하였다. 실험결과 위치제어의 정확도가 Encoder등 H/W적인 영향과, S/W적으로 Error를 판별하기 위한 공차의 설정조건에 의존되었고, 개방형제어기로의 변경에 따른 공차의 증가 및 Error는 발생하지 않았다.

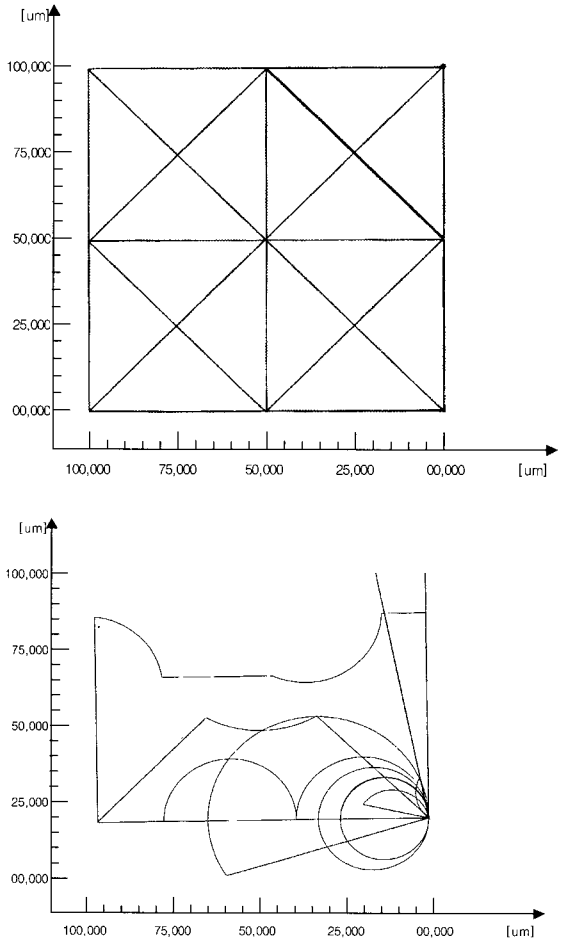

그림 9. 위치제어 특성실험 결과.

Fig. 9. Result of position control.

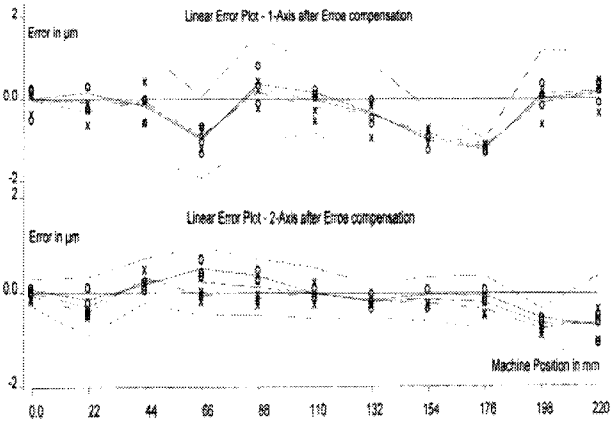

그림 10. 반복성시험 결과(전용제어기).

Fig. 10. Test report of repeatability.

그림 10는 전용제어기(HIWIN LMD-S3)측에서 제시하는 Laser로 측정한 반복성 시험결과를 나타내었으며, 그래프 상측에 X축, 하측에 Y축을 나타내었다. 그리고, 표 2는 본 시스템에서의 위치제어 반복성 실험결과 및 위치제어 오차를 나타내었으며, 그림 11은 이를 도시화하여 나타내었다.

표 2. 위치제어 반복성 실험결과 및 오차.
Table 2. Test Result & Error.

구분	시험 방법	제어결과(㎛)평균		오차(㎛)평균	
		X Axis	Y Axis	X	Y
조건 1	원점->100,000.0 이동	99,998.3	100,001.6	1.7	+1.6
	100,000.0->120,000.0	119,998.1	119,999.0	-2.0	1.0
	120,000.0->100,000.0	99,998.1	100,002.0	1.9	+2.0
조건 2	원점->100,000.0 이동	99,998.6	100,001.2	-1.4	+1.2
	100,000.0->180,000.0	179,998.3	180,001.4	-1.7	+1.4
	180,000.0->100,000.0	99,998.1	99,997.9	1.9	2.1

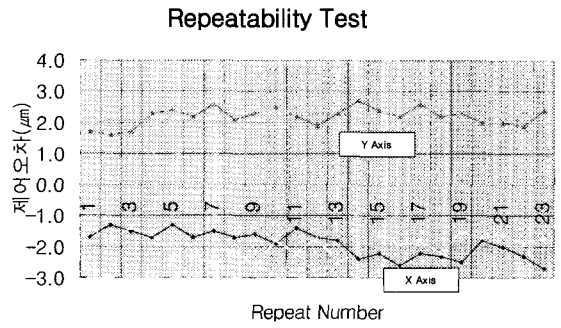

그림 11. 실험결과 및 오차.
Fig. 11. Test result & Error.

3. 실험 고찰

RT-Linux 기반 하 에서 개방형제어기를 구축, 위치제어특성 및 반복성 실험을 실시하였다. 전용제어기 및 부하를 개방형 제어 시스템에 적용할 경우, 제어특성의 변화 및 영향을 평가하였는데, 위치제어 특성 및 제어 정밀도에서 전용부하 측에서 제시하는 사양과 동등한 제어특성을 유지할 수 있었다.

IV.결 론

고객의 다양한 요구에 따른 개방형 제어기의 필요성과 그를 위한 리눅스의 이용에 대하여 비교하여 보았다. 개방형 제어기 개발 시 리얼타임리눅스의 장점은 소스가 공개되어서, 제어기의 특성과 환경에 따라 유연성 있게 적용할 수 있다는 점이 윈도우에 비하여 가장큰 장점이고, 모듈화와 실 시간성을 제공함으로써 제어효율성이 있음을 확인하였다. 이러한 리눅스 시스템의 장점과 특징을 이용, 리얼타임리눅스 기반에서 개방형제어기를 구축, 전용 제어기 및 부하를 개방화하여 범용 제어기로서 제어를 실현하였다. 제어 정밀도 및 위치제어반복성에 있어서 전용 제어기에서 요구하는 정밀도를 유지하면서, 저 비용으로 신속하게 개방형제어시스템을 구축하여 효율적으로 운용할 수 있었다. 따라서 리얼타임리눅스 기반 개방형제어기는 윈도우기반의 전용제어기를 대체할 수 있는 효과적인 제어시스템이라고 사료된다.

참고문헌

[1] R. Stones, Neil Matthew, "*Beginning linux programming*", Wrox Press, 1999.
[2] M. Beck, H. Bohme, M Dziadzka, U Kunitz, R Magnus, D. Verworner, "*Linux kernel internals 2nd*", Addison-Wesley Press, 1998.
[3] A. Rubini , "*Linux device drivers*", O'Reilly, 1998.
[4] Silberschatz, Galvin, "Operating system concepts 5th", Wiley & Sons Ltd, 1998.
[5] M. Barabanov, Victor Yodaiken, "Real-time Linux", 1996.
[6] J. Liberty & D. B. Horvath, C++ for Linux, 2000.
[7] K. Wall, M. Watson and M. Whitis, Linux programming, 2000.
[8] 김선영, 김기영, Redhat Linux 6, (주)교학사, 1999.
[9] 이만용, 이상만, 알기쉬운 알짜 레드햇 5, 정보문화사, 1998.
[10] http://www.rtlinux.org
[11] http://rtlinux.postech.ac.kr
[12] http://kldp.org
[13] http://kesl.org
[14] http://www.linuxcnc.org

조 영 준

1955년 10월 22일생, 1979년 한양대학교 기계공학과 졸업, 1981년 한국과학기술원 석사(기계공학) 졸업, 1986년 한국과학기술원 박사(기계공학) 졸업, 1986년~ 1989년 대우조선공(주), 1989년~ 현재 한국생산기술연구원 근무 중, 관심분야 자동제어, 광부품 응용개발.

강 희 석

1963년 6월 11일생, 1986년 서울대학교 기계설계학과 졸업, 1988년 동 대학원 유압제어 석사 졸업, 1996년 The University of Texas at Austin 박사(제어응용) 졸업, 1997년~현재 한국생산기술연구원 근무 중, 관심분야 자동제어 광응용분야, 비전 시스템 응용개발.

김 영 진

1964년 7월 26일생, 2002년 단국대학교 전자컴퓨터공학과 석사 졸업, 2002년~현재 동 대학원 박사과정 재학 중, 1994년~2001년 대한칼소닉(주) 기술연구소 근무, 2001년~현재 한국생산기술연구원 위촉연구원 근무 중, 관심분야 리니어 서보 모터 제어, 잉크젯분사 응용개발.

서 연 곤

1977년 11월 23일생, 2000년 서경대학교 컴퓨터공학과 졸업, 2003년 광운대학교 대학원 제어계측공학과 석사 졸업, 2003년~현재 미래산전(주) 근무 중, 관심분야 로봇제어, 리눅스 응용개발.

강 성 복

1973년 9월 30일생, 2000년 단국대학교 전자컴퓨터공학과 졸업, 2002년 동 대학원 제어계측공학과 석사 졸업, 2002년~현재 한국생산기술연구원 위촉연구원 근무 중, 관심분야 자동제어, 광통신.