공학석사 학위논문

5자유도 로봇 팔의 모션 제어기 설계와 적용

Motion Controller Design of the Robot Arm with 5 DOF and Applications

지도교수 최 형 식

2005년 2월

한국해양대학교 대학원

기 계 공 학 과

이 창 만

本 論文을 金庾信의 工學碩士 學位論文으로 認准함.

委員員 工學博士 鄭 在 鉉 (印)

委 長 工學博士 柳 三 相 (印)

委 員 工學博士 崔 炯 植 (印)

2005 年 2 月

韓國海洋大學校 大學院

機 械 工 學 科

李 昌 晩

목 차

Motion Controller Design of the Robot Arm with 5 D.O.F. and Applications

Lee, Chang Man

Department of Mechanical Engineering Graduate School, Korea Maritime University

Abstract

In this paper, we have made a motion controller to control robot arm with 5 DOF. In the case of Biped Robot, power and controller have to be installed in robot body. Until now, commercial controller is not appropriate for robot because of size and install problem. Therefore, we have designed motion controller and have made a communication protocol. So, we have established a system that is to control robot at local area. In the result, we have simplified the robot structure and have accomplished a independent system.

기 호 설 명

A_1	각 관절의 동차 변환 행렬
θ_i	각 관절의 각도
d_i	각 관절의 오프셋
a_i	각 관절의 링크 길이
α_i	각 관절의 비틀림
$R_{z,\theta}$	z 축을 기준으로 θ 만큼 회전한 행렬
$T_{z,d}$	z 축을 따라 d 만큼 회전한 행렬
$T_{x,a}$	x 축을 따라 a 만큼 이동한 행렬
$R_{x,\alpha}$	x 축을 기준으로 α 만큼 회전한 행렬
T_0^4	기구의 동차 변환 행렬

그 림 목 차

표 목 차

제 1 장 서 론

1.1 연구배경

대부분의 이동 로봇들은 독립주행을 위해서 임베디드 시스템을 구축하고 있다. 그렇지 못한 로봇들은 이동 거리 및 동작에 제한을 받게 된다. 로봇이 독립적으로 동작하기 위해서 가장 중요한 것은 전원, 통신 및 구동기이다. 이동로봇의 경우에는 특정용도에 맞게 제작되므로 일반적으로 쓰이는 것들은 적용하기 어렵다. 크기, 소비전력, 호환성등이 문제가 된다. 본 논문에서 실험 중인 이족 보행 로봇의 경우 독립 보행을 구현하기 위해서는 PC기반이 아닌 완전히 독립된 임베디드 시스템이어야 한다. 일본 아시모[16]의 경우 혼다 자사에서 개발한 구동기, 제어기와 실시간 OS를 탑재하고 있다. 한국과학기술원의 KHR[8]의 경우 자체 개발한 모션 제어기를 사용한다.

지금까지 많은 상용 모션 제어기들이 개발 되었다. 또한 제어기의 설계 방법에 대한 연구도 많이 이루어 지고 있다.[9]~[15] 하지만 대개의 경우 산업용 로봇에 맞추어 설계 되어졌고, PC를 기반으로 동작한다. 그리고 서보 드라이버 역시 BLDC, 스텝핑 모터용으로 나오고 있고, 이족 보행 로봇의 구동기로 쓰고 있는 DC서보 모터용은 찾기 어렵다. 소용량의 모터일 경우에는 몇몇 IC 타입의 제품이 나오고 있으나, 용량이 커질 경우 대개 자체 제작하여 사용하여야 한다. 이족 보행 로봇의 특수성을 고려한다면 전용 모션 제어기와 서보 드라이버, 통신 프로토콜의 설계가 필요하다.

1.2 연구 목적 및 내용

본 논문에서는 그림1.1의 이족 보행 로봇의 팔을 제어하기 위한 모션 제어기를 개발하고 적용하였다. 로봇의 키는 173cm, 무게는 92kg이다. 로봇의 자율보행을 위해서 제어기는 독립적인 시스템을 갖추어야 하고, 그 시스템의 부피가 작아야 한다. 본 연구에서는 제어기가 2축을 동시에 제어하고, 통신을 통해 명령을 받도록 설계하여 RC서보모터와 같이 구동기를 모듈화 하고자 하였다. 또한 PC에서 모션 제너레이션 프로그램으로 로봇의 현재 상태를 모니터하고, 실시간으로 명령을 내릴 수 있도록 하였다.

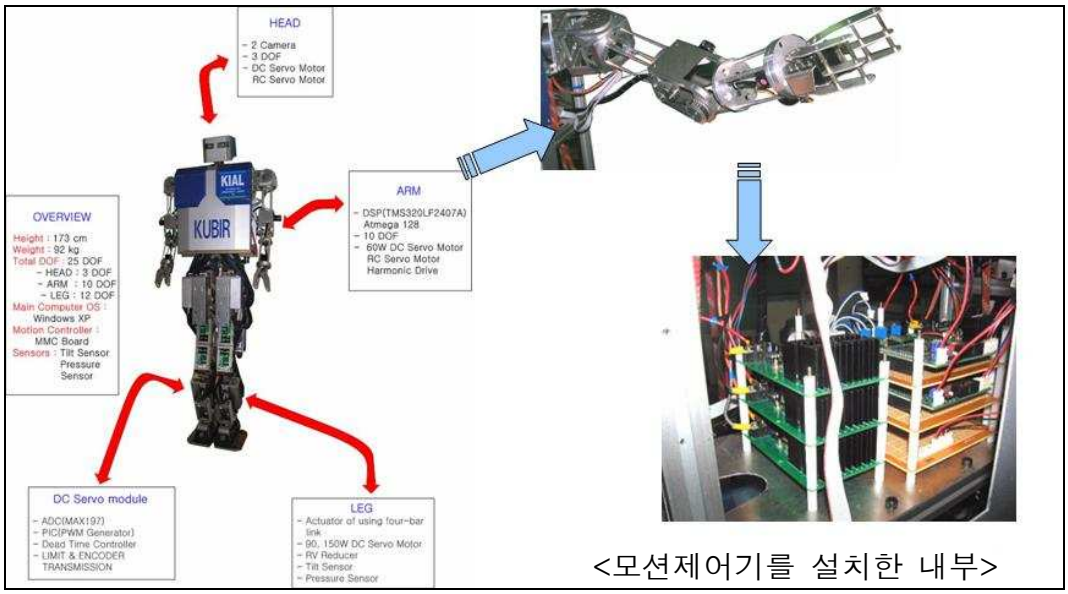

그림 1.1 이족 보행 로봇 KUBIR

본 논문의 2장에서는 본 논문에서 제어하고자 하는 5자유도 로봇 팔에 대한 순기구학, 역기구학 해석에 대해 기술하였고, 3장에서는 모션제어기의 기본 개념, 설계 방법과 모션 제네레이션 프로그램에 대해 기술하였다. 4장에서는 설계한 모션 제어기로 이용하여 로봇 팔에 적용하여 실험한 결과를 나타냈다. 5장에서는 결론 및 향후과제에 대해 논하였다.

제 2 장 로봇 팔의 기구학적 해석

로봇 팔의 말단 좌표에 대한 각 관절의 각을 알기 위하여 기구학적 해석을 행하였다. 먼저 순기구학을 해석하여 로봇의 원점으로부터 끝점에 대한 동차 변환행렬을 구한 후 역기구학 해석을 통해 각 관절의 각을 계산하였다.

2.1 순기구학 해석

그림2.1은 실제 적용 대상인 5자유도를 가진 로봇 팔의 사진이다. 몸체로부터 0, 1, 2축은 DC서보모터이고, 3,4축은 RC서보모터이다.

그림 2.1 5자유도 로봇의 사진

로봇 팔의 순기구학 해석을 하기 위해서 D-H 규약에 따라 그림 2.2와 같이 좌표계를 설정하였다.

좌표계 설정시 Denavit-Hartenberg 규약은 다음과 같다.

(D-H1) x_1축은 z_0축과 수직이다.

(D-H2) x_1축은 z_0축과 만난다.

위의 조건을 만족하기 위해서 3축의 원점을 2축으로 옮겨 좌표계를 설정하였다.

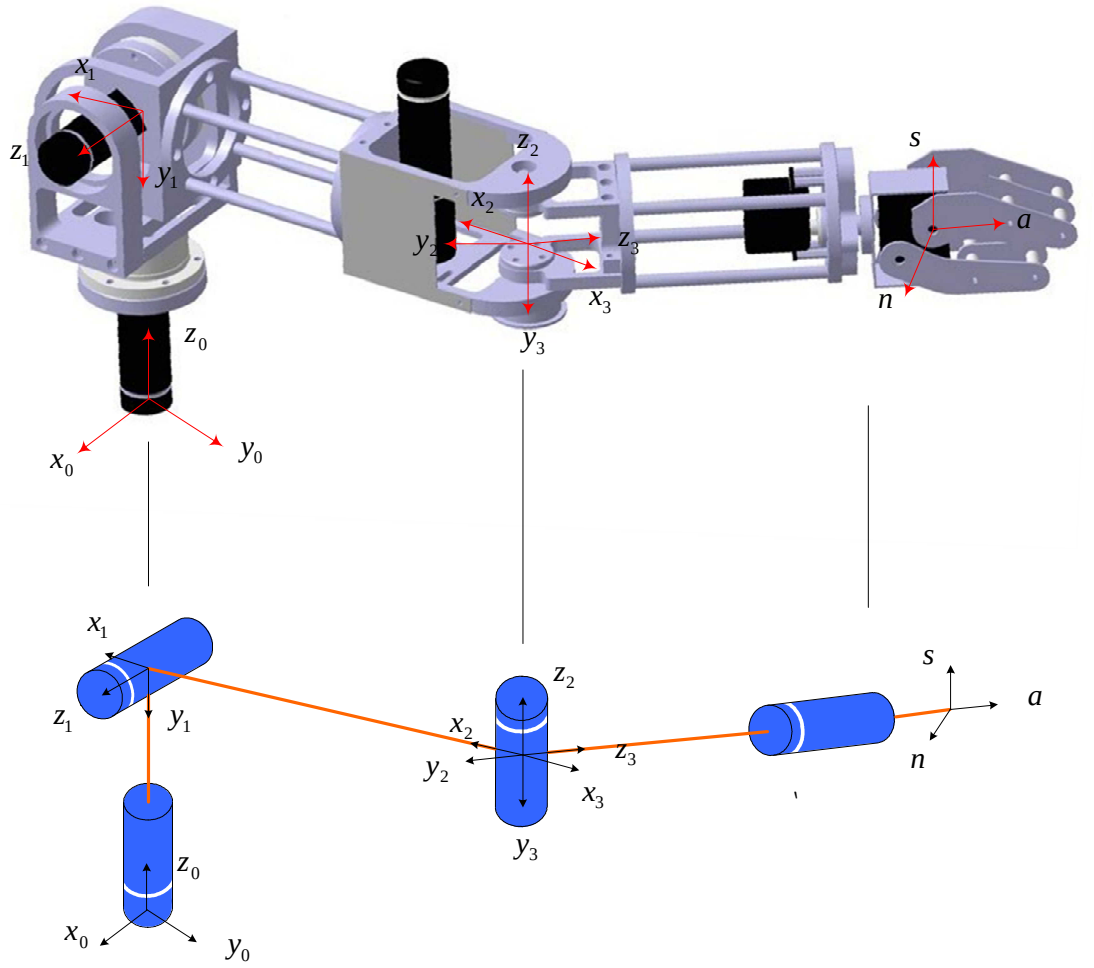

그림 2.2 Denavit-Hartenberg 표시법에 따른 좌표계

설정한 좌표계에 따른 링크 i와 관절 i의 4개의 파라미터를 구하여 표2.1에 나타내었다.

표 2.1 로봇 팔의 파라미터

	θ_i	d_i	a_i	α_i
1	θ_1^*	d_1	0	−90°
2	θ_2^*	0	a_2	90°
3	θ_3^*	0	0	−90°
4	θ_4^*	d_4	0	0

d_1= 86mm, a_2= 275mm, d_4= (192 + 126) mm

각 링크 i와 관절 i의 동차변환 행렬 A_i를 구하면, 다음과 같다.

$$A_1 = R_{z,\theta} \cdot T_{z,d} \cdot T_{x,a} \cdot R_{x,\alpha} = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} c_2 & 0 & s_2 & a_2c_2 \\ s_2 & 0 & -c_2 & a_2s_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} c_3 & 0 & -s_3 & 0 \\ s_3 & 0 & c_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

로봇 팔의 순기구학을 계산하면 다음과 같다.

$T_0^4 = A_1A_2A_3A_4 =$

$$\begin{bmatrix} (c_1c_2c_3 - s_1s_3)c_4 - c_1s_2s_4 & -(c_1c_2c_3 - s_1s_3)s_4 - c_1s_2c_4 & -c_1c_2s_3 - s_1c_3 & (-c_1c_2s_3 - s_1c_3)d_4 + a_2c_1c_2 \\ (s_1c_2c_3 - c_1s_3)c_4 - s_1s_2s_4 & -(s_1c_2c_3 + c_1s_3)s_4 - s_1s_2c_4 & -s_1c_2s_3 + c_1c_3 & (-s_1c_2s_3 + c_1c_3)d_4 + a_2s_1c_2 \\ -s_2c_3c_4 - c_2s_4 & s_2c_3s_4 - c_2c_4 - c_2c_4 & s_2s_3 & s_2s_3d_4 - a_2s_2 + d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2 역기구학 해석

역기구학 문제를 풀 때 짧은 시간에 방정식을 풀기 위해 반복적 계산을 하는 수치해보다는 해석적 해를 구하는 것이 바람직하다. 대개 역위치 기구학과 역방향 기구학으로 나누어 풀면 간단해진다. 먼저 로봇 손목 중심의 위치를 찾고, 다음 손목의 방향을 찾는 방법이다.

2.2.1 역위치 기구학 문제

역기구학적 문제를 다루는 일반적인 방법은 거의 없다. 구조가 단순한 경우에는 기하학적인 방법으로 푸는 것이 간단하다. 아래에 로봇 팔을 x_0y_0 평면에 투영한 그림이 있다.

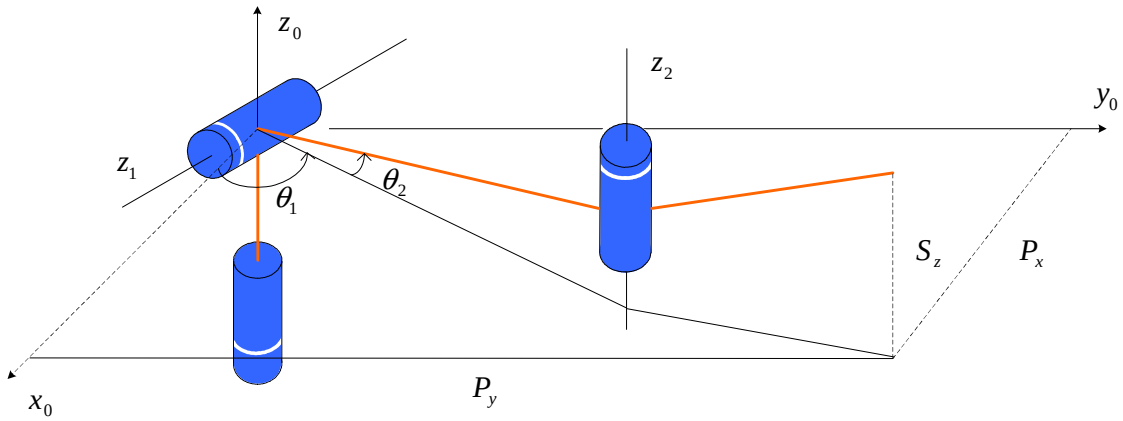

그림 2.3 x_0y_0 평면에 투영

그림 2.3에서 $\theta_2 = A\tan(S_z, \sqrt{P_x^2 + P_y^2}) = A\tan(P_z - d_1, \sqrt{P_x^2 + P_y^2})$ 이다.

θ_1, θ_3 을 구하기 위해서 투영된 그림2.3 를 세워서 보면 그림2.4와 같다.

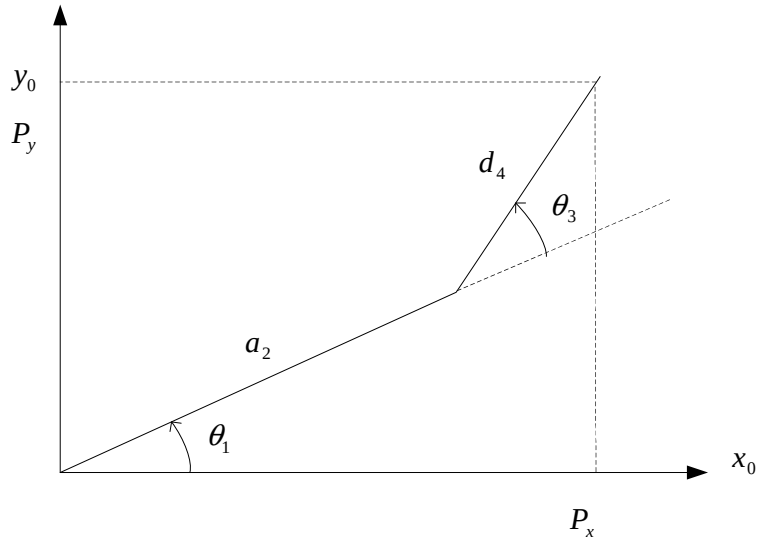

그림 2.4 링크 a_2, d_4으로 형성된 평면에 투영

코사인 법칙을 이용하면

$\cos\theta_3 = \dfrac{P_x^2 + P_y^2 - a_2^2 - d_4^2}{2a_2 d_4} := \mathrm{D}$ 이고, 따라서 $\theta_3 = A\tan(D, \pm\sqrt{1-D^2})$ 이다.

$\theta_1 = A\tan(P_x, P_y) - A\tan(a_2 + d_4\cos\theta_3, d_4\sin\theta_3)$ 이다. θ_3에 대한 해는 상향 팔꿈치,하향팔꿈치 2가지이다. 사람의 팔 구조와 맞추기 위해 하향 팔꿈치일 경우만 해로 정한다.

2.2.2 역방향 기구학 문제

로봇 팔의 행렬 R_0^3은

$R_0^3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & c_1s_2 & -c_1c_2s_3 - s_1c_3 \\ s_1c_2c_3 + c_1s_3 & -s_1s_2 & -s_1c_2s_3 + c_1c_3 \\ -s_2c_3 & -c_2 & s_2s_3 \end{bmatrix}$ 이고, $R_3^4 = \begin{bmatrix} c_4 & -s_4 & 0 \\ s_4 & c_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 이다.

마지막 θ_4에 대해서 풀면, $R_3^4 = (R_0^3)^T R$ 이다. 여기서

$$(R_0^3)^T = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & s_1c_2c_3 + c_1s_3 & -s_2c_3 \\ c_1s_2 & -s_1s_2 & -c_2 \\ -c_1c_2s_3 - s_1c & -s_1c_2s_3 + c_1c_3 & s_2s_3 \end{bmatrix}$$ 이고,

$$R = \begin{bmatrix} (c_1c_2c_3 - s_1s_3)c_4 - c_1s_2s_4 & -(c_1c_2c_3 - s_1s_3)s_4 - c_1s_2c_4 & -c_1c_2s_3 - s_1c_3 \\ (s_1c_2c_3 - c_1s_3)c_4 - s_1s_2s_4 & -(s_1c_2c_3 + c_1s_3)s_4 - s_1s_2c_4 & -s_1c_2s_3 + c_1c_3 \\ -s_2c_3c_4 - c_2s_4 & s_2c_3s_4 - c_2c_4 - c_2c_4 & s_2s_3 \end{bmatrix}$$ 이다.

위의 항등식으로 θ_4 를 구하면 수치적으로 수많은 값을 가진다. 따라서, 로봇 팔의 손모양에 따라 수정되어야 할 변수이다.

제 3 장 모션 제어기의 설계

3.1 모션 제어기의 구성

3.1.1 하드웨어 구성

이족 로봇의 팔을 제어하기 위해 설계한 모션제어기의 전체 시스템 구성도는 그림 3.1과 같다. 호스트 PC에서 로봇 팔의 이동 경로를 생성하고, RS232 통신을 통하여 하위 제어기에 명령을 내린다. 호스트 PC와 모션 제어기 사이에는 멀티 통신을 위한 중계기가 있다. 모션제어기는 호스트 PC로부터 받은 명령에 따라 증폭부로 PWM과 Direction 신호를 보낸다.

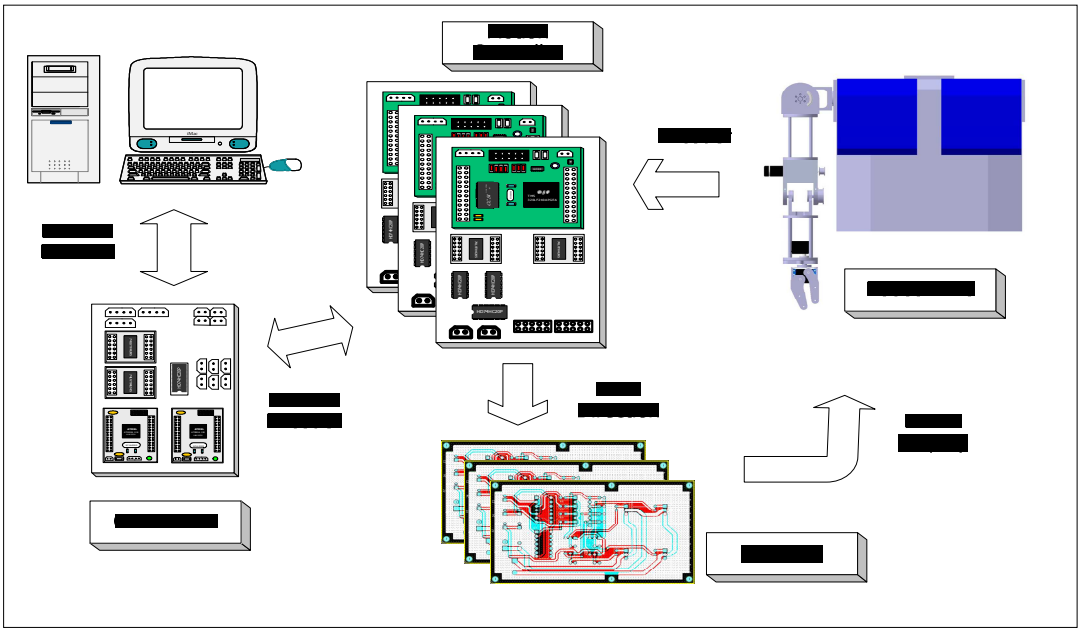

그림 3.1 모션 제어기의 구성

다음에 그림 3.2와 그림 3.3은 설계한 모션 제어기를 로봇에 장착한 모습이다.

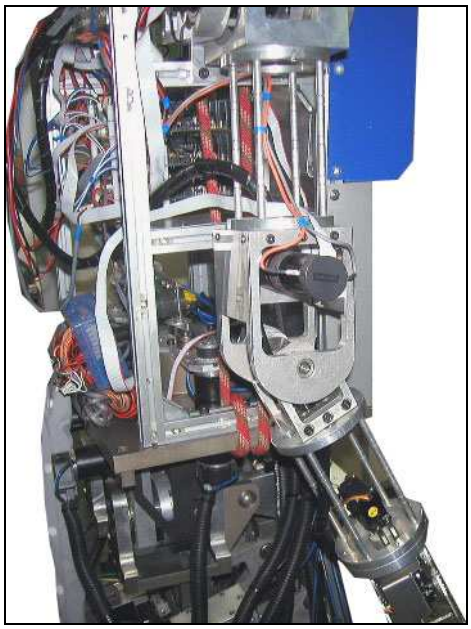

그림 3.2 로봇의 좌측 사진

그림 3.3 제어기가 장착된 로봇의 내부사진

(1) Host PC

로봇 팔의 상태를 모니터하고, 로봇 팔의 경로를 생성하여 하위 제어기로 명령을 보낸다. Host PC의 사양 표3.1과 같다.

표 3.1 Host PC의 사양

품 명	사 양
OS	Windows XP
CPU	AMD 1.8 GHz
RAM	512MB
Software	MFC

(2) 멀티 통신 중계기

모션 제어기는 보드 한장으로 2축을 제어하도록 설계하였다. 따라서 호스트 PC와 1대 다수의 통신을 구현하여야 한다. 멀티통신 중계기는 호스트 PC로부터 받은 명령을 각각의 모션 제어기로 보내고, 각 모션 제어기로부터 받은

엔코더값을 모아서 Host PC로 전송한다.중계기의 구성은 그림 3.4와 같다.

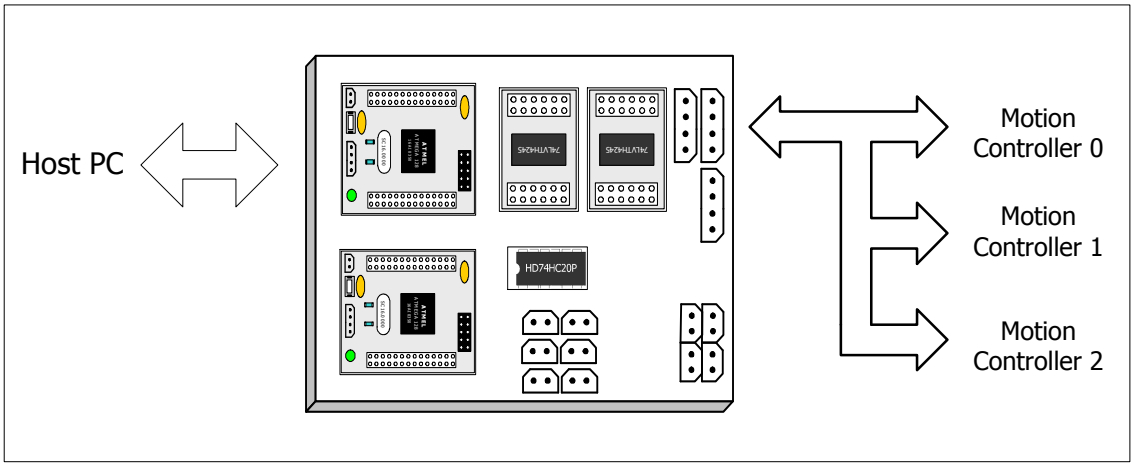

그림 3.4 멀티 통신 중계기

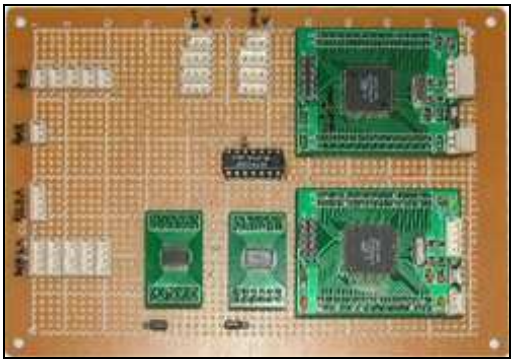

그림 3.5 멀티 통신 중계기 사진

멀티 통신 중계기의 제원은 표3.2와 같다.

표 3.2 멀티 통신 중계기 제원

부 품	부 품 명	기 능
MCU	ATmega128 - 20MHz	내장 UART0, UART1
Level Shifter	74LVHT4245	3.3V <-> 5V
OS	UCOS-II	Real-Time OS

(3) 2축 모션 제어기

모션 제어기는 TMS320LF2407A에 내장된 엔코더 카운터2개를 이용하여 보드 한장이 2축의 모터를 제어할 수 있도록 설계하였다.

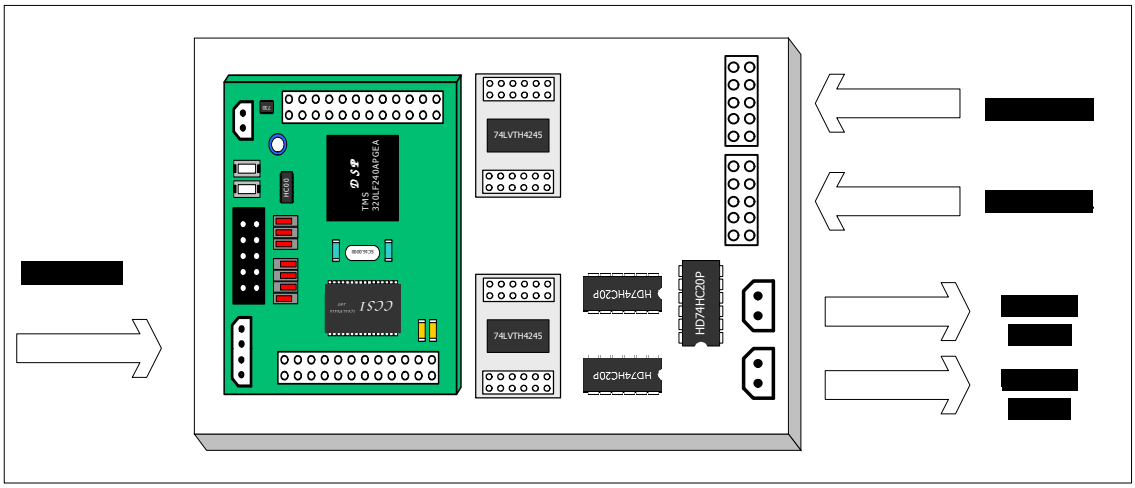

그림 3.6 모션 제어기 구성

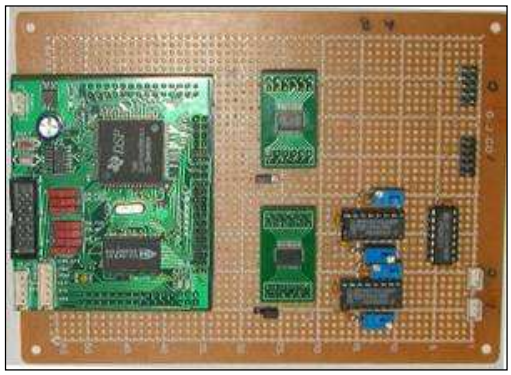

그림 3.7 모션 제어기 사진

모션 제어기의 제원은 아래와 같다.

표 3.3 모션제어기의 제원

부 품	부 품 명	기 능
MCU	TMS320LF2407A	PWM, EncoderCounter, UART, Timer, GPIO
Level Shifter	74LVHT4245	3.3V <-> 5V
DeadTime Generator	CD4538	Dual Precision Monostable

(4) 모터 드라이버 (증폭부)

모터 드라이버는 모션 제어기에서 PWM과 Direction 신호를 입력 받아 PWM 듀티비만큼 모터에 전압을 공급한다. 모터 드라이버의 실제 사진은 그림3.8과 같다.

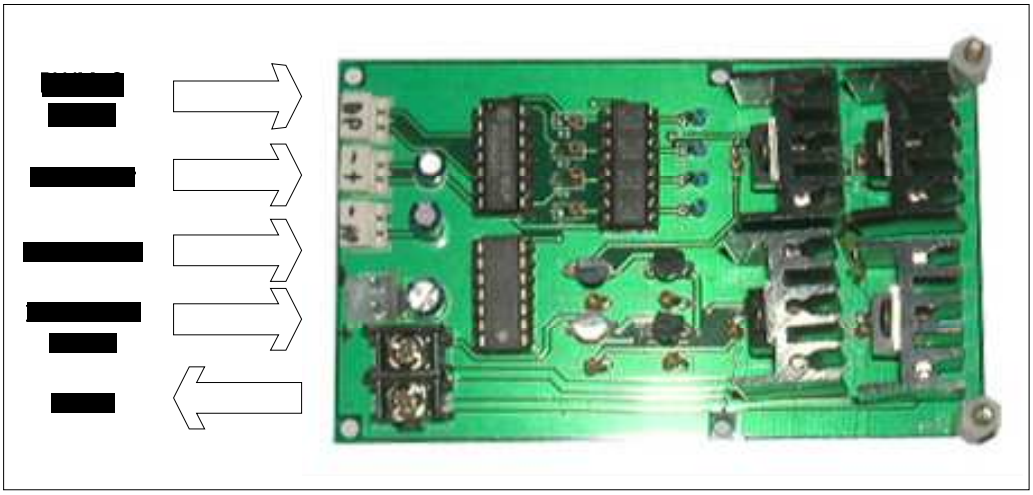

그림 3.8 모터 드라이버의 실제 사진

그림3.9는 모터 드라이버의 회로도이다.

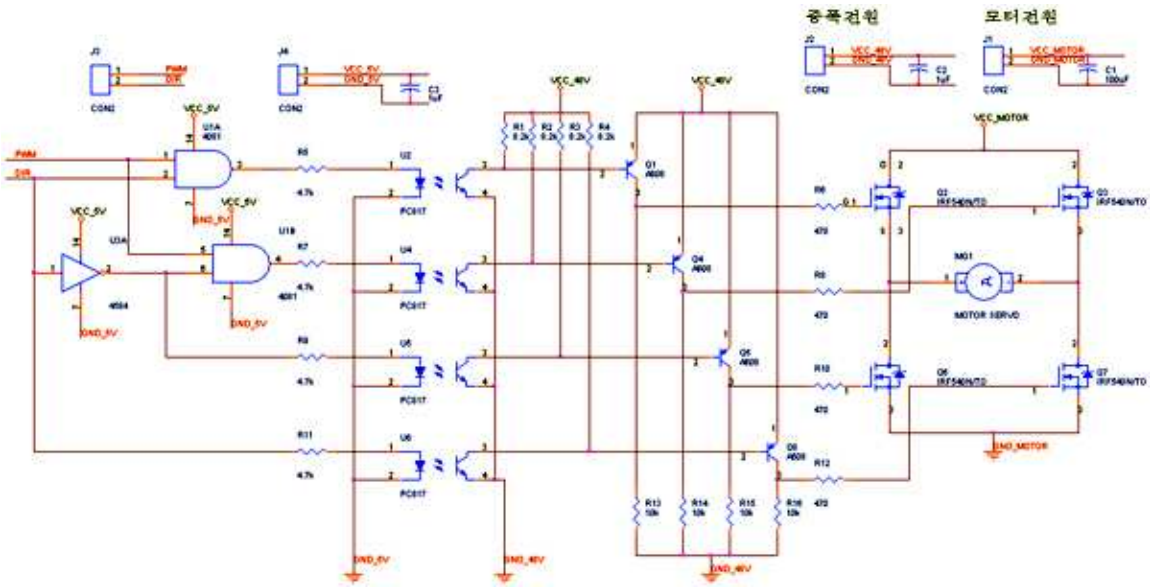

그림 3.9 모터 드라이버 회로도

모터 드라이버 회로도에서 효율을 높이기 위해서는 FET의 Drain 보다 Gate 에 전압이 1.5배 이상 높아야 한다. Gate 에 공급되는 전압원으로 그림3.10과

같은 전원부를 제작하였다.

그림 3.10 전원 증폭기사진

(5) 로봇 팔

로봇의 팔은 6개의 DC서보모터와 6개의 RC서보모터가 부착되어 있다. RRR 형태의 다관절 로봇과 구조가 같다. 그림3.11은 CATIA로 모델링한 그림이다.

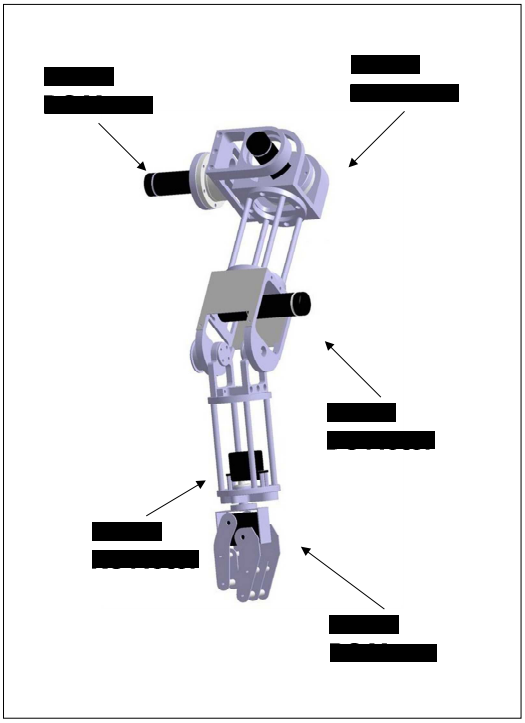

그림 3.11 로봇 팔의 구조

로봇 팔의 구동기 제원은 표3.4와 같다.

표 3.4 로봇 팔 구동기의 제원

축	모터	감속비
0	DC 서보 60W, 12V	하모닉(1:100)
1	DC 서보 60W, 12V	하모닉(1:100)
2	DC 서보 60W, 12V	기어헤드(1:51) 풀리(1:2)
3	RC 서보 모터	X
4	RC 서보 모터	X

3.2 모션 제어기의 설계

3.2.1. Tracking Control의 개념

Motion Control 에 관한 문제는 크게 두가지로 나뉜다. 하나는 제어의 목적점이 고정되어 있는 Regulating Problem이며, 둘은 목적점이 시간에 따라 변화하여 궤적을 구성하고, 이를 추종하는 것이 목적인 Tracking Problem 이다. DC Servo Motor의 위치, 속도제어를 위해서는 적합한 프로파일을 설계하는 것이 가장 큰 문제이다.

(1) 구동 프로파일을 생성

서보 테이블을 생성하기 위해서는 우선적으로 제어의 목표값으로 이용되는 레퍼런스 궤적이 생성되어야 한다. 이는 매 인터럽트마다 목표치로 참조되는 점들의 집합으로 구동프로파일이라고도 불린다. 고성능의 서보 시스템으로 갈수록 구동 프로파일의 중요도는 높아진다.

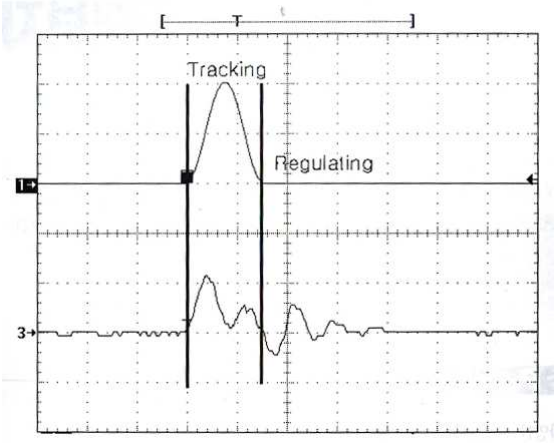

그림 3.12 구동 속도 프로파일과 에러 파형

그림3.12에 표시된 바와 같이 구동 프로파일 중 발생하는 에러를 보통 Tracking Error (or Dynamic Error)라 하고, 구동 후 안정화되기까지 몇 ms 동안 발생하는 error를 Settling Error (or Regulating Error)라 한다. 그림 3.13은 가장 많이 쓰이는 위치 및 속도 프로파일이다.

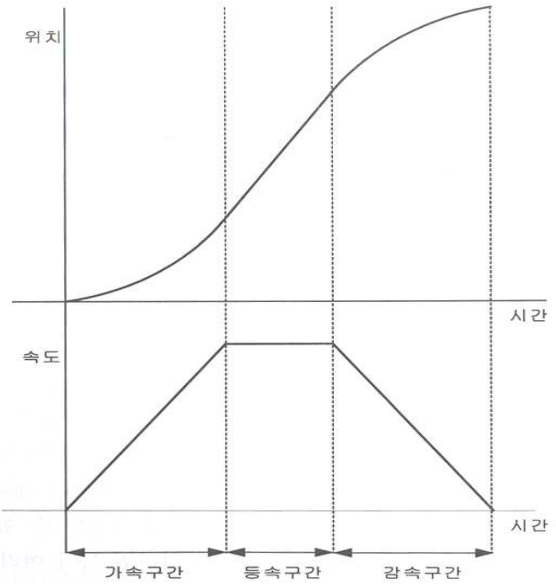

그림 3.13 사다리꼴 형상 속도 프로파일

구동 프로파일은 보통 위치 프로파일이 기준이 되며, 그 프로파일의 도함수인 속도 프로파일, 가속도 프로파일, 저크프로파일등이 한 Set을 이룬다. 사다리꼴 형상인 프로파일은 계산이 간단하므로 저가형 프로세서로도 구현이 가능하다. 하지만 가감속 구간의 시작점과 끝점에서 커다란 전류저크가 발생하므로, 구동부의 기구부에 충격이 가해지며, 이로 인한 진동의 유발과 순간적으로 큰 다이나믹 에러가 발생할 수 있다. 따라서 PTP 구동이 아닌 CTC 구동에서는 이러한 불연속적인 저크의 생성을 막기 위하여, 계산량이 많더라도 보다 부드러운 프로파일을 개발하여 사용한다. 흔히'S'형태의 매끄러운 위치 프로파일들은 서로 비슷하게 보이지만, 그 도함수를 취하면 상당히 다른 속도 및 가속도 프로파일을 가진다.

인터럽트 주파수가 빠른 경우(4kHz 정도 이상), 구동 프로파일 테이블(Motion Table)은 주로 위치에 대한 테이블로 저장되며, 보통 중력가속도를 기준으로 3G이상의 고속 서보 시스템에서는 2048개 이상의 테이블 분해능을 사용한다. 속도나 가속도 프로파일은 위치테이블을 그때그때 수치 미분하여 사용하지만, 수치 미분의 한계로 가속도 테이블을 하나 더 사용하는 경우도 있다.

(2) 구동 시간의 결정

구동 프로파일을 설계한후 가장 먼저 결정되어야 하는 것은 구동시간이다. 얼마나 빠른 시간내에 정해진 위치로 이동 시킬것인가를 결정해야한다.

① 최대 허용 속도 - 주로 엔코더의 최대 출력 주파수와 엔코더 카운터의 최대입력 주파수에 의한여 결정한다.
② 최대 허용 가속도 - 모터가 발휘할 수 있는 최대 힘과 부하의 크기에 의해서 결정된다. 즉 부하, 모터 상수, 증폭부의 최대 전류에 의하여 결정된다.
③ 최대 허용 저크 - 시스템의 제어 주파수, 모터 증폭부의 전류제어 밴드폭, 모터의 전기적 시상수, 기계적인 강성등 전체 시스템의 성능에 따라 결정되어진다. 초고속의 시스템일수록 이 최대 허용 저크를 얼마나 높일 수 있느냐에 따라 그 시스템의 운동시간에 대한 성능이 좌우된다.
④ 최대 허용 다이나믹 에러 - 다이나믹 에러는 서보 운동시 발생하는 시간에 따른 오차를 말한다. 최대로 허용되는 다이나믹 에러는 전체 시스템의 허용

오차를 결정짓는다.

3.2.2 속도 프로파일의 설계

(1) 사다리꼴 모양의 속도 프로파일 설계

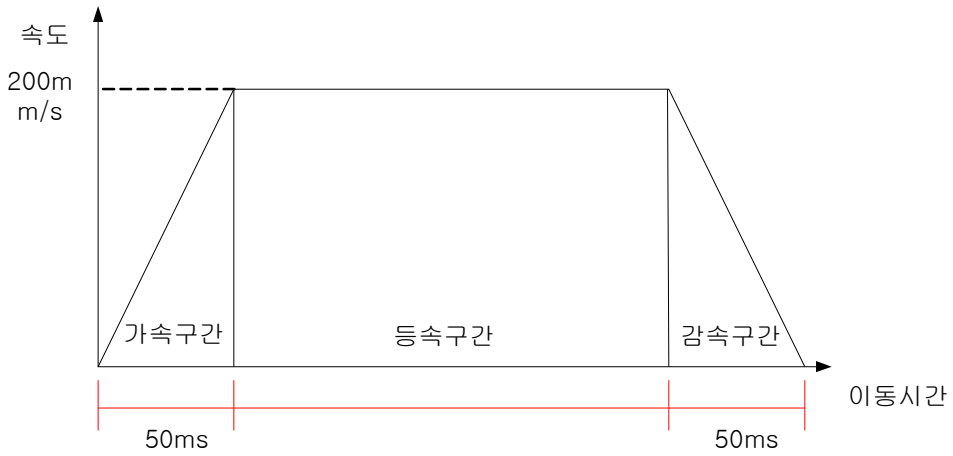

그림 3.14 사다리꼴 모양의 속도 프로파일

① 조건

1. 이동거리 = 1000mm
2. 최대속도 = 200mm/s
3. 가속시간 = 50ms
4. 감속시간 = 50ms
5. 인터럽트주기 = 2ms
6. 1회전당 이동거리 = 10mm
7. 1회전당 엔코더펄스수 = 2000 pulse

② 계산해야할 파라미터

1. 총이동거리 = 가속시 이동거리 + 등속시 이동거리 + 감속시 이동거리
2. 가속시 이동거리 = 200 * 50E-3 / 2 = 5mm
3. 감속시 이동거리 = 200 * 50E-3 / 2 = 5mm
4. 등속시 이동거리 = 총이동거리 - 가감속시이동거리

= 1000 - 10 = 990mm

6. 등속 시간 = 990mm / 200 = 4.95s
5. 이동시 총소요시간 = 가감속시간 + 등속시간
= 0.050 + 0.050 + 4.95 = 5.05초
7. 가속시 인터럽트 발생 수 = 50ms / 2ms = 25번
8. 등속시 인터럽트 발생 수 = 4950ms / 2ms = 2475번
9. 감속시 인터럽트 발생 수 = 50ms / 2ms = 25번
10. 1mm 이동시 엔코더 펄스수 = 200pulse
11. 1pulse 당 이동거리 = 1/200mm
12. 가감속 기울기 = 200 / 50 = 4 mm/s^2
13. 위치프로파일 생성

- 가속구간의 위치
- 등속구간의 위치
- 감속구간의 위치

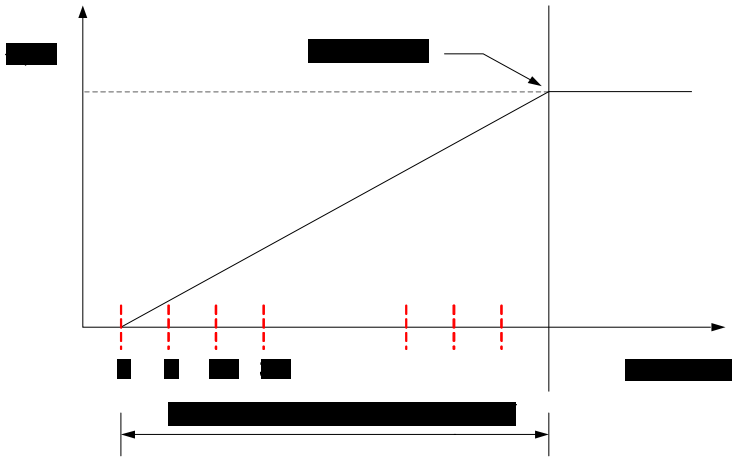

그림 3.15 가속구간의 속도 증가곡선

위치테이블은 설계한 속도 테이블을 매 인터럽트마다 적분하여 생성한다. 가속구간과 감속구간은 빠른 처리를 위해 미리 계산한 후 테이블을 만들어둔다.

(2) 'S' 자형의 프로파일 설계

위치프로파일(512개)을 생성하고, 속도는 인터럽트서비스 루틴 안에서 수치미분하여 사용한다.

$y = a + bt + ct^2 + dt^3$ - 위치 함수

$y' = b + 2ct + 3dt^2$ - 속도 함수

$y'' = 2c + 6dt$ - 가속도 함수

$y(0) = 0, y(1) = 1$, $y'(0) = 0, y'(1) = 0$ - 초기,최종 경계치

$y = 3t^2 - 2t^3$ - 최종 위치 함수

'1' 이라는 시간에 '1' 이라는 거리를 가는, 거리에 대하여 표준화된 프로파일을 구하는 것이 실제 프로세서에 프로그램해 넣기가 편리하다.

Matlab을 이용하여 trajectory를 생성한다.

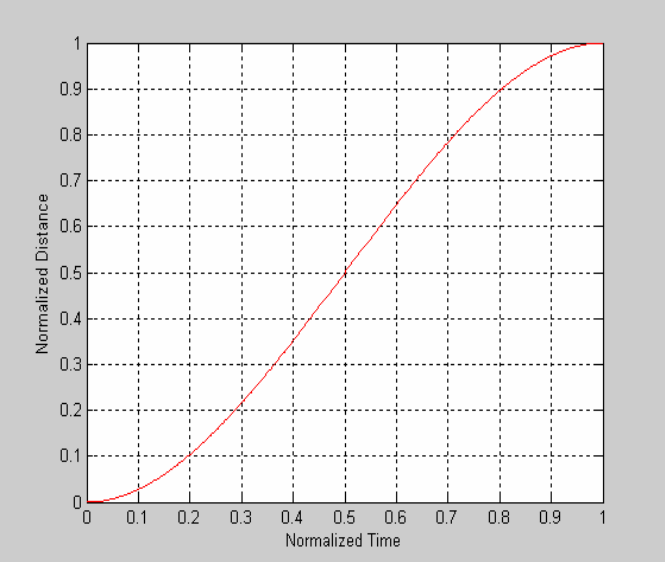

그림 3.16 위치 테이블

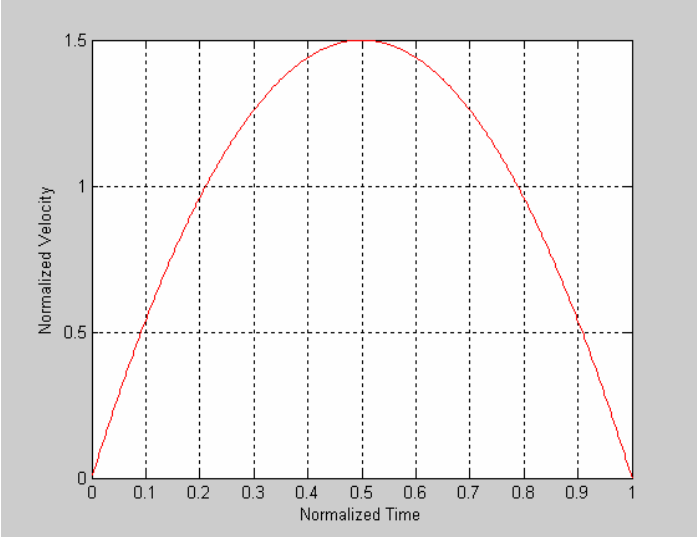

그림 3.17 속도 테이블

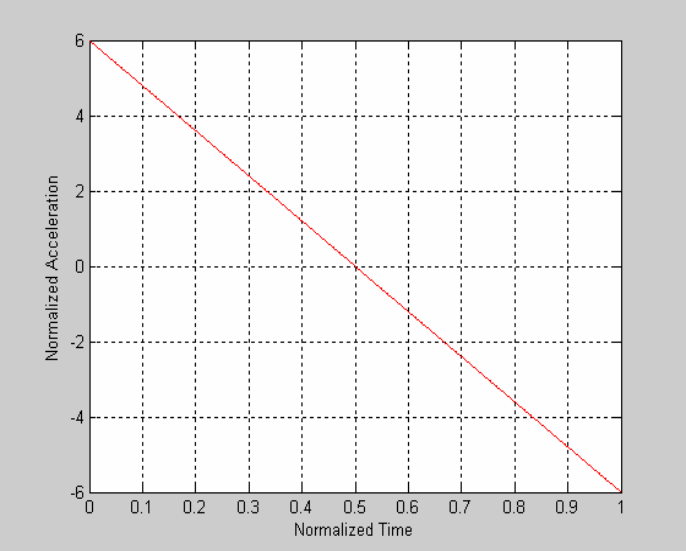

그림 3.18 가속도 테이블

3.2.3 모터 제어 알고리즘

(1) 제어기 구조의 설계

PID 제어기법은 단순하고, 높은 효율성 때문에 실제 가장 많이 쓰이는 제어 기법이다. 인터럽트 루틴안에서 수행되는 기본 동작은 그림3.19와 같다.

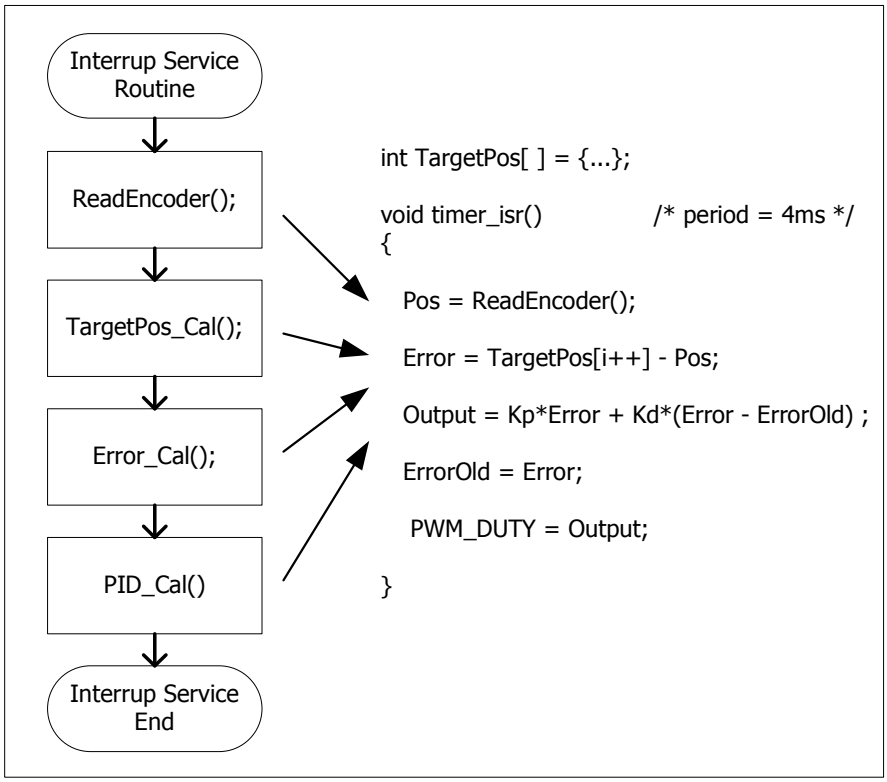

그림 3.19 인터럽트 루틴의 순서도

그림3.20은 일반적인 라플라스 함수를 이용한 PID 제어기의 블록선도이다.

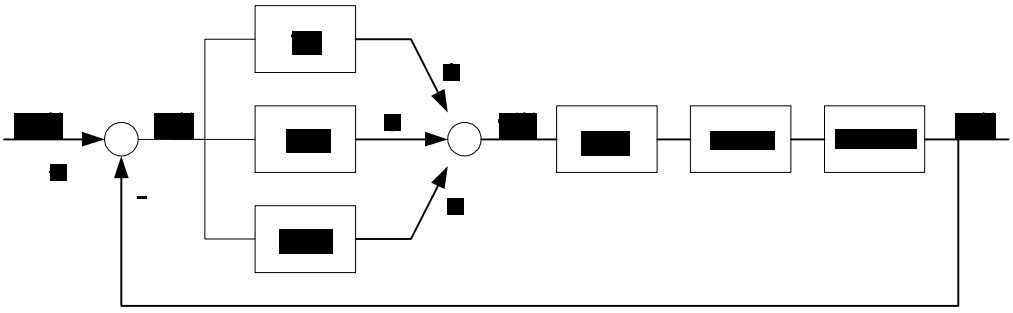

그림 3.20 일반적인 PID 제어기의 블록 다이어그램

그림3.20을 C 코드로 옮기면 다음과 같다.

```
PositionError = TargetPosition - CurrentPosition;
VelocityError = TargetVelocity - CurrnetVelocity;
ErrorSum += Ki * PositionError;
DAOut = Kp*PositionError + Kd*VelocityError + ErrorSum;
```

위의 기본적인 형태의 PID 제어기를 좀 더 개선하면 그림3.21과 같은 형태가 된다.

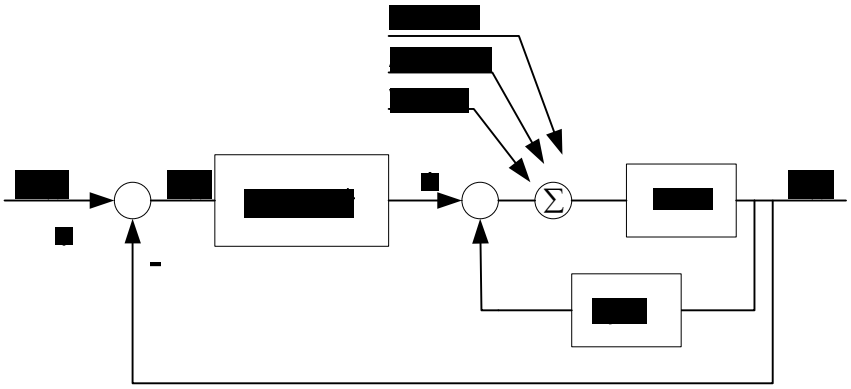

그림3.21 FeedForward 텀을 가지는 수정된 PID 제어기의 블록 다이어그램

위의 그림을 C 코드로 옮기면,

```
PositionError = TargetPosition - CurrentPosition;
VelocityError -= CurrentVelocity;
ErrorSum = LCoef*ErrorSum + Ki*PositionError;
if(ErrorSum > ER_Limit_Pos) ErrorSum = ER_Limint_Pos;
else if(ErrorSum<ER_Limit_Neg) ErrorSum = ER_Limit_Neg;
DAOut = Kp*PositionError + Kd*VelocityError+ErrorSum;
DAOut = VelocityFF + AccFF + JerkFF;
```

위와 같이 코딩하면 FeedForward 텀과 Feedback 제어기가 분리된 구조로 되어 실험이나 분석할 때 유리해진다.

3.3 uC/OS-II를 이용한 통신프로토콜 설계

PC와 제어기의 1:N의 멀티 통신을 구현하기 위해 uC/OS-II를 사용하였다. Windows XP는 내부 프로세스들의 동작 때문에 매 시간마다 정확한 동작을 예상하기 어렵다. 그래서 PC는 모니터링의 역할만을 수행하고 제어기들 사이의 통신은 중계기를 통하도록 하였다.

3.3.1 실시간 OS란?

Real Time OS는 태스크들의 동작하는 시간을 정확히 예상할 수 있는 OS이다. 일반적인 OS는 한 태스크의 동작시간이 정확하지 않아도 된다. 사용자가 조금 더 기다리면 된다. 하지만 항공기, 미사일등에서 조금의 시간 지연은 곧 사람의 생명과 직결된다. 커널은 크게 선점형 커널과 비선점형 커널로 나눌 수 있다.

(1) 선점형 커널

- 어떤 한 Task가 수행되고 있을 때, kernel이 그 Task의 수행을 강제로 중지시키고 다른 Task(높은 priority를 갖는)를 수행시킬 수 있음.

(2) 비선점형 커널

- 어떤 한 Task가 수행되고 있을 때, kernel이 그 Task의 수행을 강제로 중지시키고 다른 Task를 수행시킬 수 있는 능력이 없음.

uC/OS-II 는 선점형 커널이다.

3.3.2 uC/OS-II 기본 설명

uC/OS-II 는 소스가 공개되어 있고, 개발자의 자세한 설명이 첨부되어 있어 교육용, 개발용으로 많이 쓰이고 있다. 다음은 uC/OS-II 를 사용하기 위한 초기화 설정 방법이다.

< Initialzation >

초기화는 OSInit() 함수를 호출함으로써 이루어진다 . 단 , UCOS.H 파일에 다음과 같은 것들이 설정되어 있어야 한다 .

OS_IDEL_TASK_STK_SIZE : idel 태스크의 크기를 byte 로 나타낸 것이다 .

OS_MAX_TASKS : uC/OS 가 제어할 수 있는 태스크의 최대 수
OS_MAX_EVENTS : 응용 프로그램이 생성할 최대 ECB 수를 나타낸다 .
OS_MAX_QS : 응용 프로그램이 생성할 메시지 큐의 최대 수를 나타낸다 .
OSInit() 호출 후 OSStart() 함수를 호출하게 되면 멀티태스킹이 시작된다 .

이것이 호출되기 전에 하나 이상의 태스크가 생성되어 있어야 한다 . 이 함수는 highest priority 를 결정하고 OSTCBHighRdy 를 설정한다 . 멀티태스킹이 시작되었다는 것은 OSRunning 을 TRUE 로 설정함으로써 나타낸다 . 가장 높은 우선 순위를 갖는 태스크를 로드하고 실행하기 위해서는 OSStartHighRdy() 함수를 호출하게 되는데 이 함수는 어셈블리로 짜여진 것이다 .

Configuration

uC/OS 의 설정은 다음과 같은 단계를 거친다 .

1) UCOS.H 내의 #define 값을 고친다 .
   OS_IDLE_TASK_STACK_SIZE
   OS_MAX_TASKS
   OS_MAX_EVENTS
   OS_MAX_QUEUES
   uCOS
2) OSTimeTik()을 호출하는 ISR 을 작성한다 .
3) context switch 코드인 OS_TASK_SW()에 대하여 인터럽트 벡터를 할당하고 설정한다 .
4) tick ISR 에 대한 인터럽트 벡터를 할당하고 설정한다 .
5) OSStart()를 호출하기 전에 main()에서 OSInit()을 호출한다 .
6) semaphore, mailbox, queue 를 생성한다 .
7) 태스크 스택 영역을 할당한다 .
8) 태스크 중 하나를 생성한다 .
9) 멀티태스킹을 행할 준비가 되면 OSStart()를 호출한다 .

다음은 uC/OS-II의 세부 내용이다.

그림 3.22 uC/OS-II의 기능

아래의 그림은 uC/OS-II의 태스크 상태도를 나타내고있다.

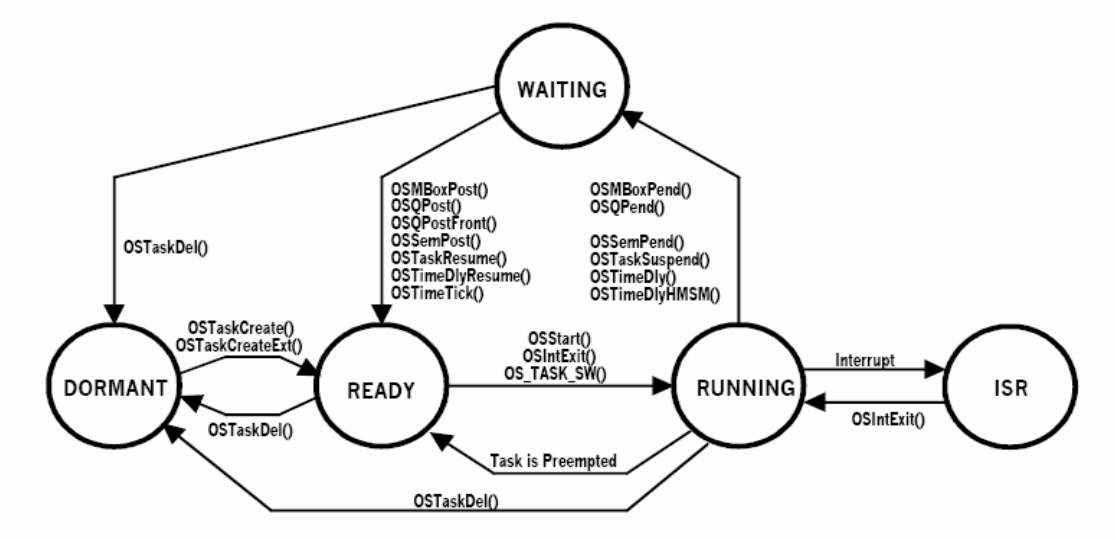

그림 3.23 uC/OS-II 의 태스크 상태도

3.3.2 모션 제어기를 위한 멀티 통신 프로토콜 설계

1:N의 멀티 통신을 구현하기 위하여 uC/OS-II를 포팅한 ATmega128의 펌웨어 소스의 구성은 다음과 같다.

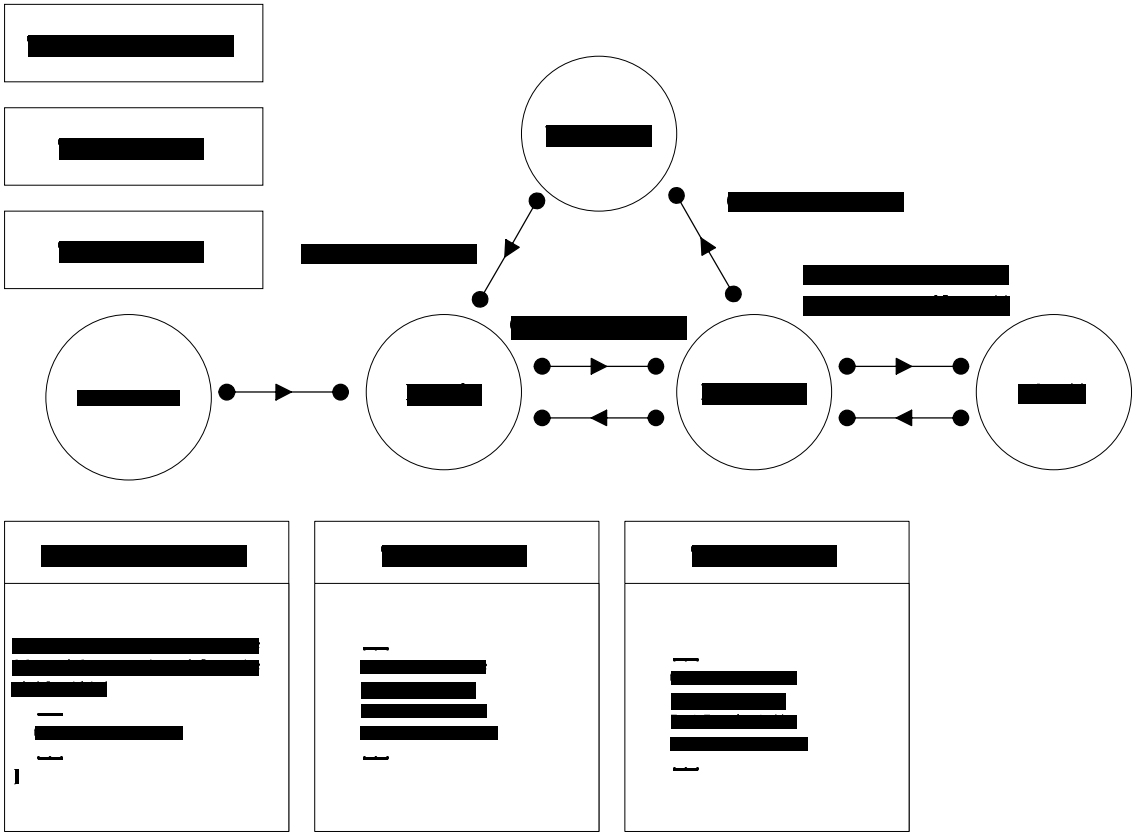

그림 3.24 멀티 통신 중계기의 태스크 구성 및 상태도

프로그램은 백그라운드에서 동작하는 통계태스크와 아이들 태스크를 제외하고 3개의 태스크로 구성된다. TaskAppStart 태스크는 Task1App와 Task2App를 생성하고, 10ms 마다 각 모터 축의 엔코더 값을 읽어온다. Task1App은 PC와 통신을 하고, Task2App는 모션 제어기들과 통신을 담당한다. RxIntHandler는 UATR0, UART1로부터 데이터를 전송받으면 각 태스크로 메시지박스를 이용하여 데이터를 수신하였음을 알리다. 각 태스크들은

메시지박스로부터 시그널을 받을 때 까지 대기 상태로 있다.

3.4 모션 제너레이션 프로그램의 설계

모션 제너레이션 프로그램에서 로봇의 모델링은 OpenGL, 직렬통신은 공개 라이브러리 인 glcomport14d.lib와 glcomport14.h를 이용하였다. 전체 화면 구성은 아래와 같다. 모델링한 로봇의 치수는 실제 로봇의 실치수를 일정하게 축소하였다.

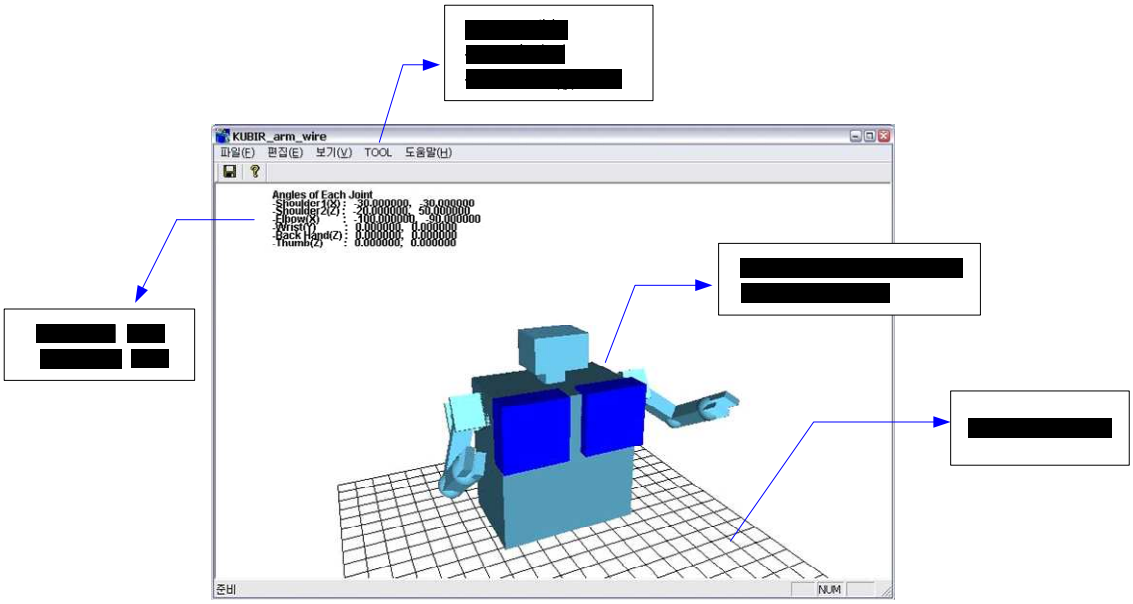

그림 3.25 모션 제너레이터 프로그램의 화면 구성

역학적인 시뮬레이션은 없고, 모니터링 역할을 한다. 명령입력 대화상자에서 각 관절의 값을 입력하고, 데이터를 전송하면 모션제어기에서 그 명령을 받아 로봇의 팔을 제어하게 된다. 100ms 마다 각 관절의 엔코더 값을 입력받아 모델링한 로봇의 팔을 실제 위치와 매칭한다.
모션 제너레이션 프로그램의 개발환경은 Microsoft Visual Studio 6.0 & MFC 사용하였고, 아래 그림은 클래스의 구조이다.

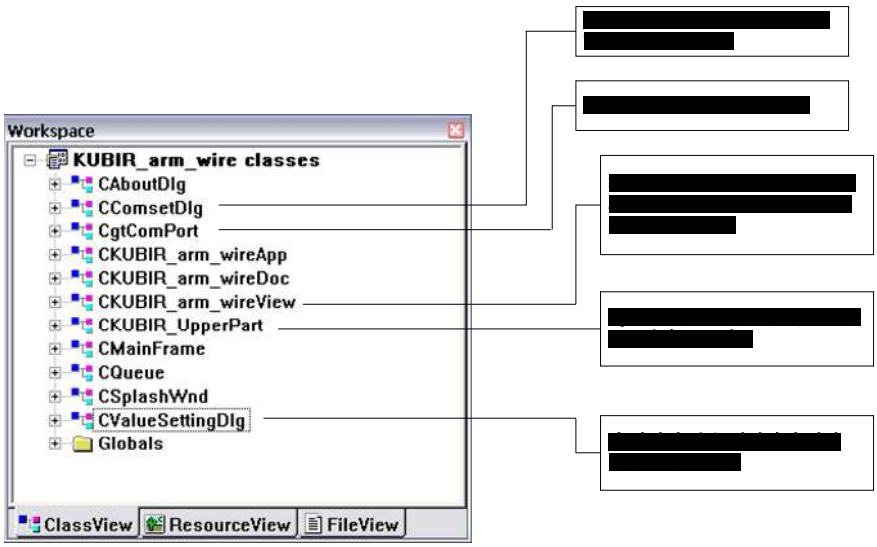

그림 3.26 모션 제너레이터 프로그램의 클래스 구조

시리얼 통신 설정하는 다이얼로그 창이다. 시리얼 포트와 통신속도를 설정한다.

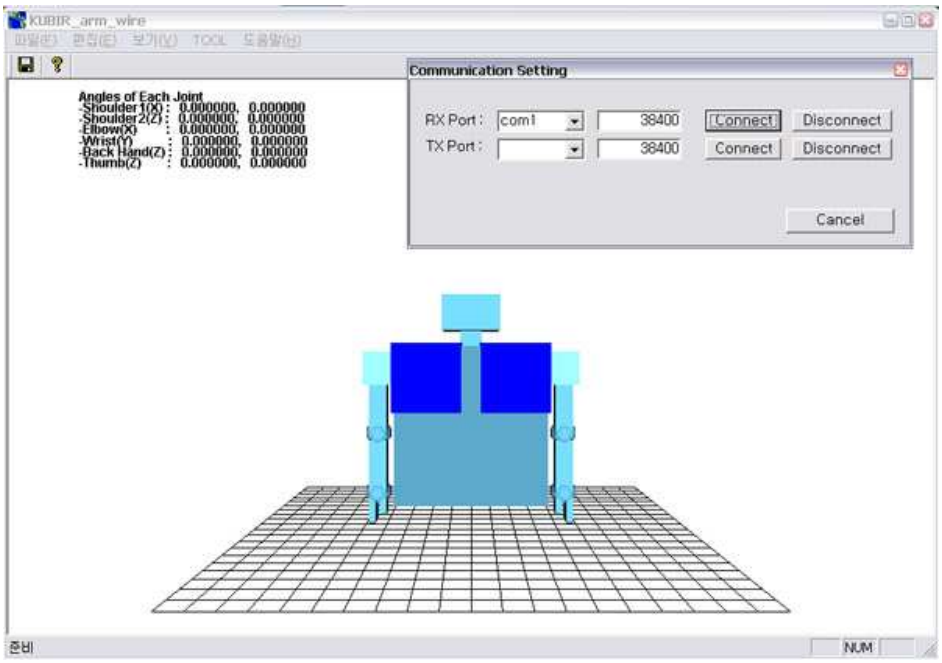

그림 3.27 모션 제너레이터 프로그램의 통신설정 대화상자

각 관절의 위치 명령을 입력하는 다이얼로그 창이다.

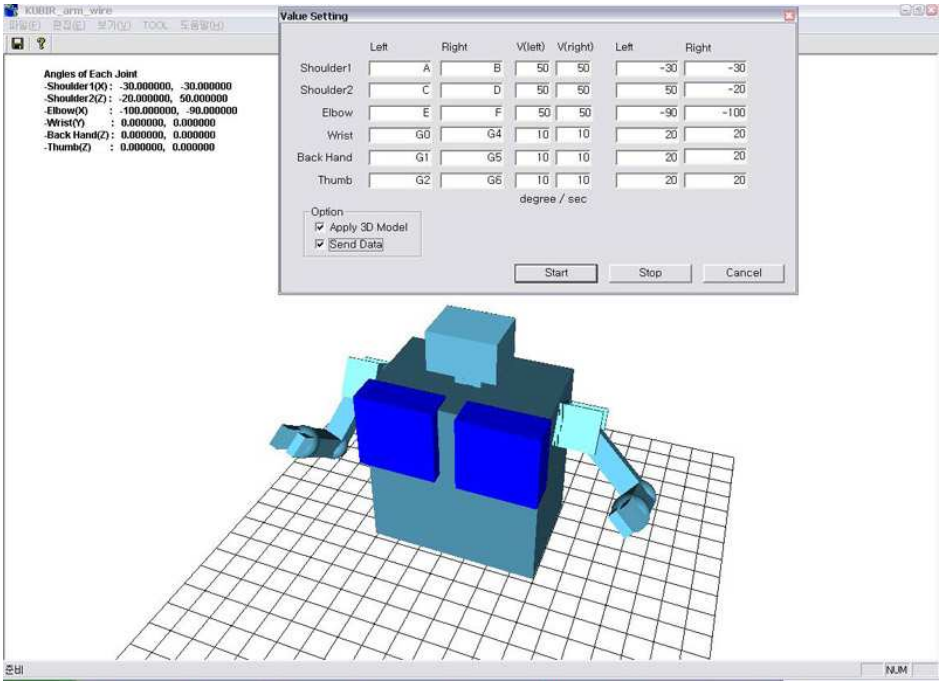

그림 3.28 모션 제너레이터 프로그램의 명령입력 대화상자

각 관절의 초기 속도, 위치값과 목표 속도, 위치값을 입력하고 데이터전송 체크박스를 클릭하고, Start 버튼을 누르면 명령을 전송한다. Apply 3D Model 체크박스를 클릭하면, 입력한 값에 따라 모델링한 로봇의 팔이 동작한다.

제 4 장 실험 및 고찰

실험은 기본적인 위치제어와 3장에서 설계한 사다리꼴 모양의 속도 프로파일과 3차식을 이용한 속도 프로파일을 사용하여 실험하였다. 실험결과 감속기를 부착하고 있어 비교적 모터의 진동은 없었으나 최종위치에서 에러는 커졌다. 반대로 감속기가 없는 모터의 경우는 초기 출발시에 조금의 진동은 생기나 최종위치에서 에러는 줄었다. 감속기가 부착되어 기동토크가 커졌음을 알 수 있다. 짧은 구간을 이동할 때 500Hz 보다는 1kHz 로 제어주기가 짧아 졌을 때, 보다 안정한 구동을 하였다. 위치 그래프는 모든 실험에서 비슷한 모양을 가지나 속도 그래프는 구동 프로파일에 따라 많이 달라짐을 볼 수 있다. 구동 프로파일을 사용한 실험에서 나타난 속도 그래프를 보면 속도의 진동이 많이 생김을 볼 수 있다. 제어의 어려움을 알 수 있는 부분이었다. 많은 개선이 필요하다.

4.1 각 축의 모터 튜닝

각 축의 모터 튜닝을 하기 위해 여러 조건에서 실험을 행하였다.

(1) 제어주기 = 500Hz, 기어비 = 1:51, Kp = 1, Kd = 6, 목표위치 = 3000 펄스

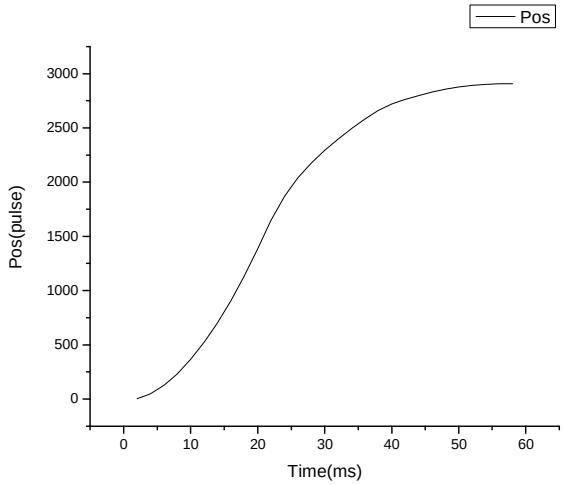

그림 4.1 위치 그래프

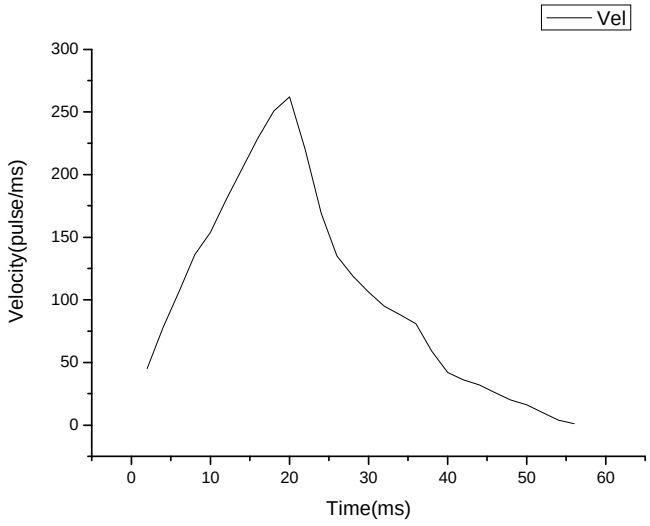

그림 4.2 속도 그래프

(2) 제어주기 = 500Hz, 기어비 = 1:51, Kp = 1, Kd = 6, 목표위치 = 5000 펄스

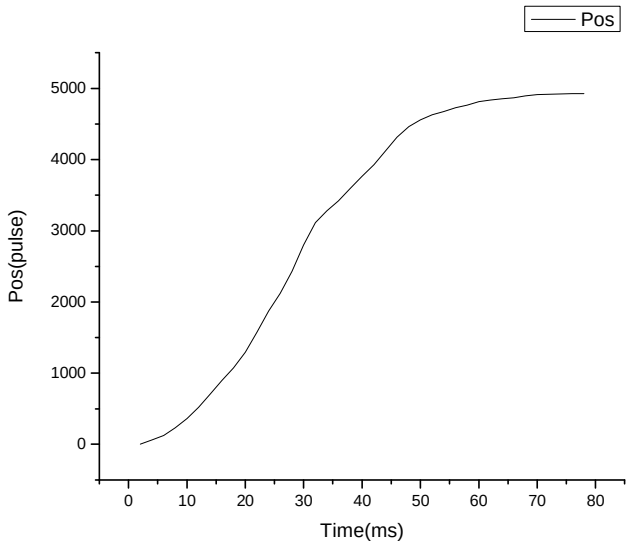

그림 4.3 위치 그래프

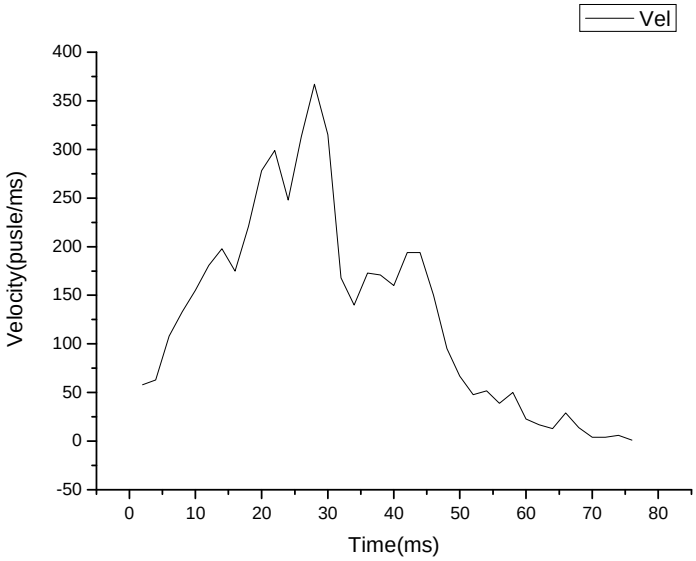

그림 4.4 속도 그래프

(3) 제어주기 = 1kHz, 기어비 = 1:51, Kp = 1, Kd = 6, 목표위치 = 1000 펄스

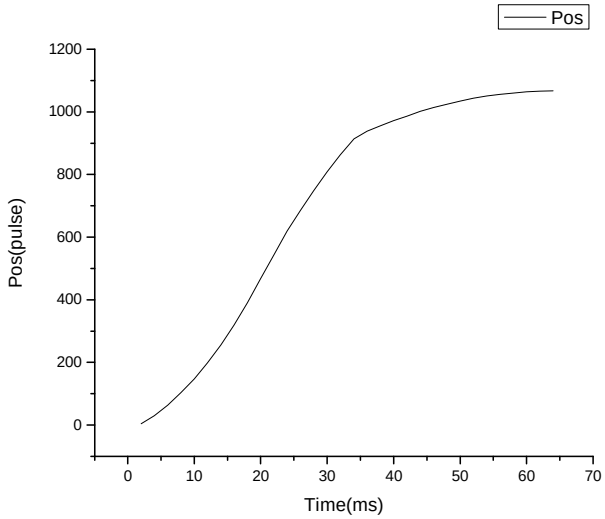

그림 4.5 위치 그래프

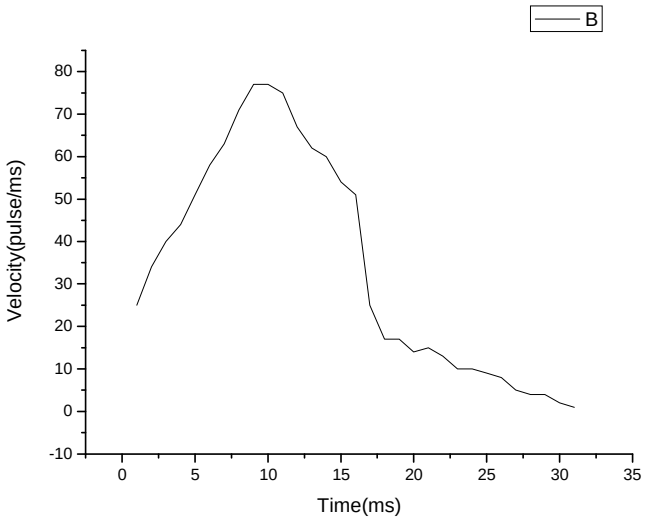

그림 4.6 속도 그래프

(4) 제어주기 = 1kHz, 기어비 = 1:51, Kp = 1, Kd = 6, 목표위치 = 10000 펄스

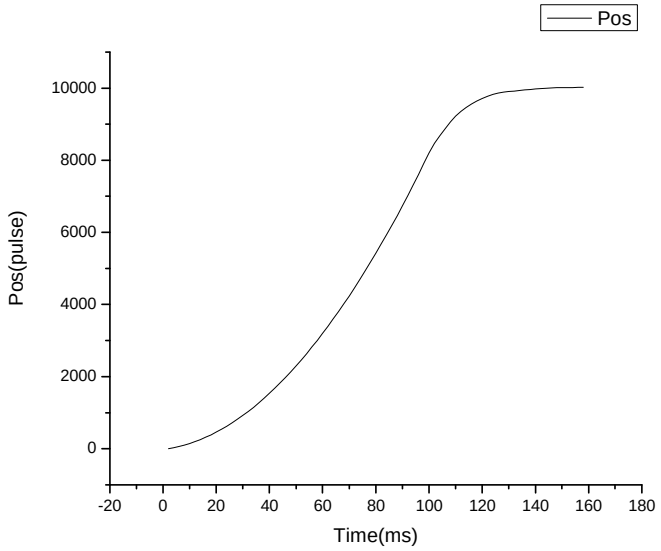

그림 4.7 위치 그래프

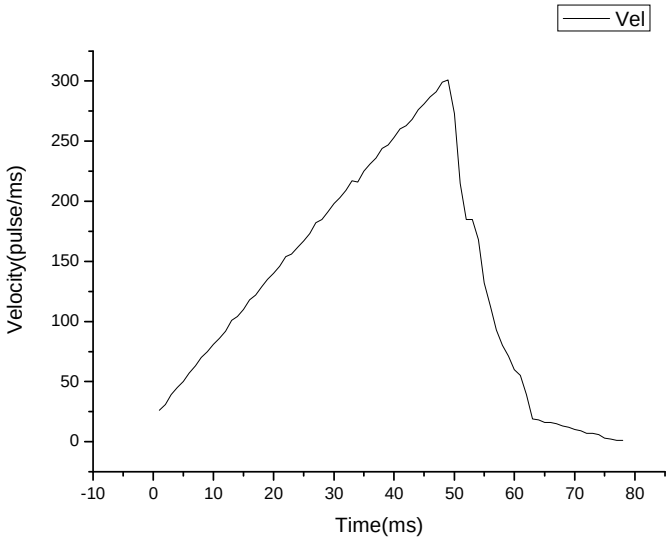

그림 4.8 속도 그래프

4.2 트랙킹 제어에 의한 모터 튜닝

다음은 사다리꼴 모양의 속도 프로파일에 따른 모터의 위치, 속도 그래프이다.

(1) 제어주기 = 1kHz, 기어비 = 1:51, Kp = 1, Kd = 6,

목표위치 = 45000 펄스 평균속도 = 45펄스/ms

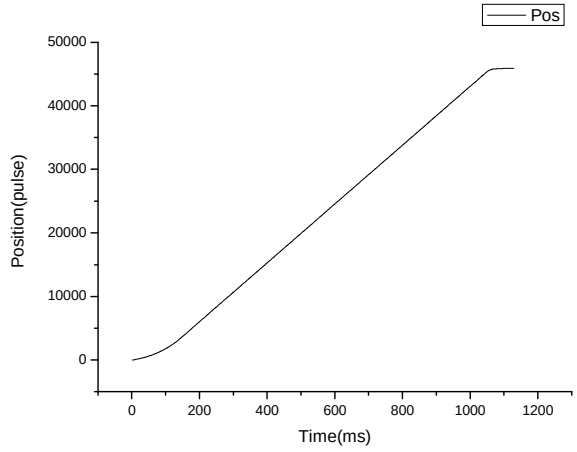

그림 4.9 위치 그래프

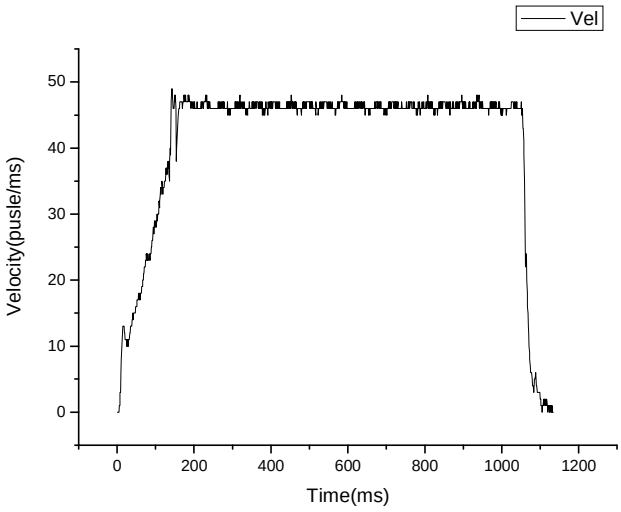

그림 4.10 속도 그래프

(2) 제어주기 = 1kHz, 기어비 = 1:51, Kp = 1, Kd = 6,
목표위치 = 45000 펄스, 평균속도 = 90펄스/ms

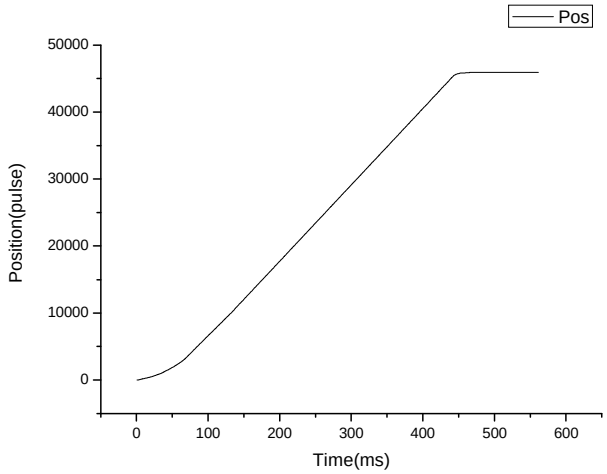

그림 4.11 위치 그래프

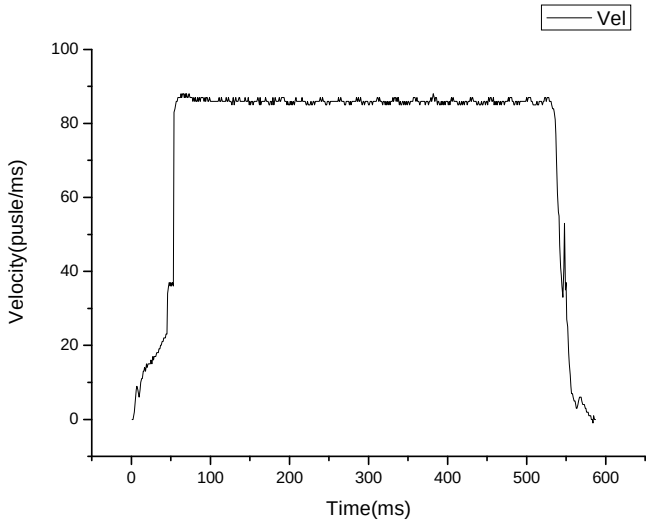

그림 4.12 속도 그래프

다음은 3차식의 속도 프로파일에 따른 모터의 위치, 속도 그래프이다.

(1) 제어주기 = 1kHz, 기어비 = 1:51, Kp = 1, Kd = 6,
목표위치 = 10000 펄스

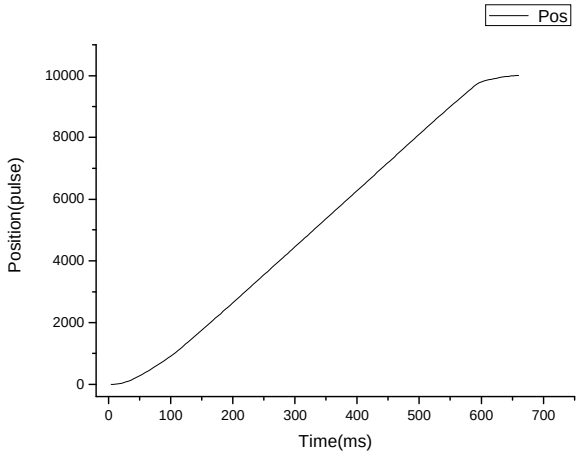

그림 4.13 위치 그래프

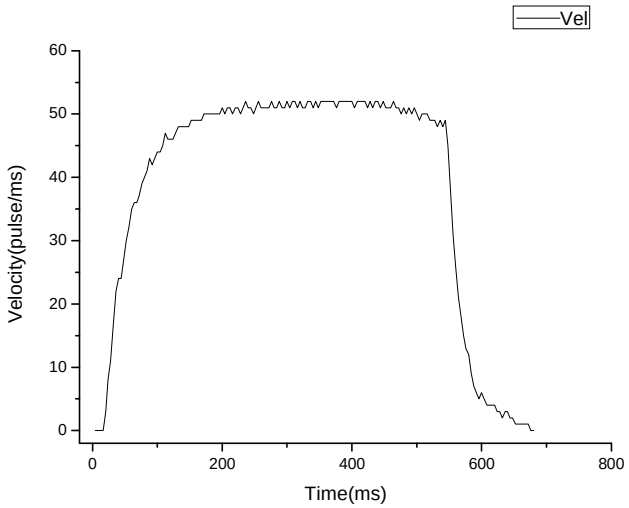

그림 4.14 속도 그래프

그림 4.15과 그림 4.16은 다축을 동시에 제어할 때 로봇 팔의 움직임을 찍은 사진이다.

#동작1

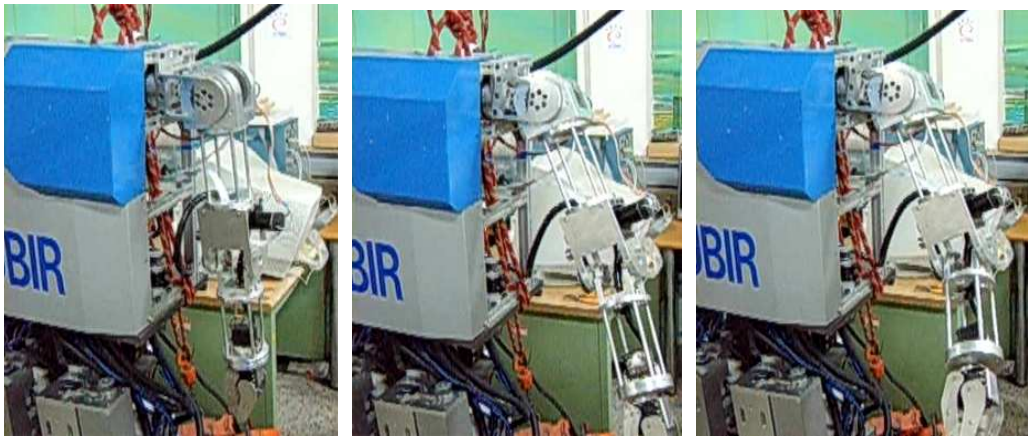

그림 4.15 로봇팔의 동작 사진

#동작2

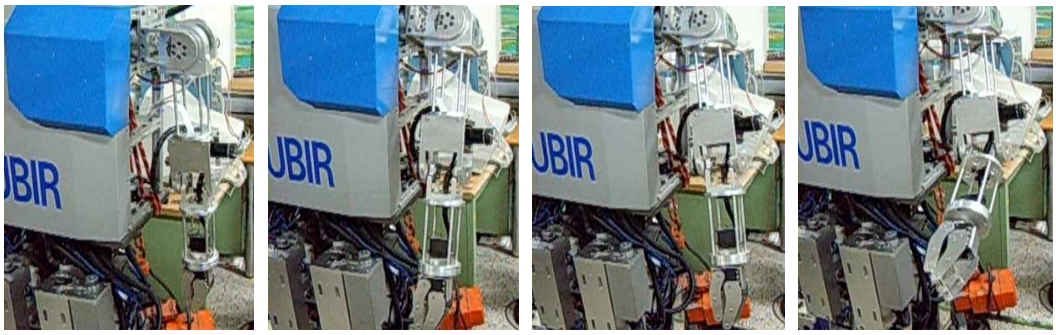

그림 4.16 로봇 팔의 동작 사진

5. 결 론

본 논문에서는 5자유도 로봇 팔의 움직임을 제어하기 위한 소형화된 모듈 형태의 모션 제어기를 개발하였다. 기존의 PC 기반의 제어기에서 벗어나 독립된 형태를 가짐으로써 부피가 작고, 추가 설치 및 고장 진단이 간편한 모션 제어기를 개발하였고, 이를 이족 보행 로봇에 장착된 5자유도 로봇 팔에 적용하였다. 모션 제너레이션 프로그램을 이용하여 보다 쉽게 로봇 팔의 움직임을 만들 수 있고, 현재 로봇 팔의 움직임을 모니터링 할 수 있다. 모션 제어기 1개는 2축을 동시 제어할 수 있고, 2축이상을 동시에 제어하기 위해서 1:N 통신 중계기를 설계하였다.

이를 이용하여 이족 보행 로봇의 경량화와 제어의 용이함을 얻을 수 있었다. 하지만 성능면에서는 아직 개선의 여지가 있다. 모션 제어기의 성능 및 사용상의 유연성, 다축 동시제어시 동기의 확실성등 검증되어야 할 부분도 남아있다.

현재는 공간 상의 한점에 대한 위치점을 teaching에 의해서 찾아가지만 향후 해석한 팔의 역기구학을 적용하고, 각 링크 상호간의 미치는 힘도 고려한 제어 방법을 찾아야 하겠다.

참고문헌

[1] DSP로 리니어 모터 제어하기, 김정한, 동일출판사

[2] TMS320LF2407A User Guide, Texas Instrument. Inc.

[3] TMS320C24계열을 이용한 DSP 하드웨어 설계, SyncWorks

[4] ATmega128 User Manual , Atmel. Inc.

[5] Robot Dynamics Control, Spong Vidyasagar

[6] Embedded Systems Building Block, Jean J. Labrosse

[7] MicroC/OS-II Real Time Kernel, Jean J. Labrosse

[8] Jung-Hoon Kim, Seo-Wook Park, Ill-Woo Park, and Jun-Ho Oh, "Development of a Humanoid Biped Walking Robot Platform KHR-1 - Initial Design and Its Performance Evaluation," in Proceedings of 3rd IARP International Workshop on Humanoid and Human Friendly Robotics, pp. 14-21, Tsukuba, Japan, Dec. 11-12, 2002

[9] 김일환, "브러쉬리스모터의디지털제어기",서울대학교 학위논문, 1988

[10] 최병준,"다구찌방법을 이용한 한 축 서보모터의 제어이득", 조정 서울대학교 학위논문, 1999

[11] 정치연, "실시간 운영체제를 이용한 로봇 제어기 소프트웨어 디자인 및 구현", 서울대학교 학위논문, 1995

[12] 임석구, "직류전동기 속도제어를 위한 PID 자기동조제어기의 Single-chip microcomputer에 의한 구현", 서울대학교 학위논문, 1987

[13] 박상규, "VR형 DD모터의 고성능 디지탈 제어기 설계 및 구현",서울대학교 학위논문,1991

[14] 서용호, 휴먼로봇 프로토타입의 설계 및 구현, 한국과학기술원 학위논문, 2001

[15] 김형일,마이크로 프로세서를 이용한 PWM 직류 서보 모터 속도 제어 장치에 관한 연구, 한국과학기술원, 1982

[16] http://asimo.honda.com/

[17] http://www.sony.net/SonyInfo/QRIO

감사의 글

처음 아무것도 모른체 실험실에 들어와서 무언가를 배워보겠다는 열정만 가지고 덤벼들었던게 엊그제 같다. 지금와서 뒤돌아 생각해보면 그땐 정말 아무것도 몰랐던거 같다. 대학원에 와서 많은 것을 보고 배웠다. 무언가에 열중해 보기는 운동이후로 처음이었다. 삶이 보람차다고 느꼈다. 시간은 왜 이렇게 빨리 가는 것일까? 난 벌써 많은 시간들을 써버렸다. 왜 좀더 일찍 알지 못했을까? 지금 알고 있는 것을 그때 알았더라면..하는 생각을 많이 하게 된다. 지금이라도 알게 되었음을 감사히 여기고 내가 원하는 일에 전력투구할 것을 다짐한다. 지금 내가 있기까지 수많은 실수를 해왔다. 앞으로도 더 많은 실수를 하겠지만 나의 앞날을 생각하면 푸른 빛이 나는거 같다. 이렇게 지난 시간들이 값지게 느껴지는 것은 주위에 많은 친구, 선후배들이 있었기 때문이다. 지도 교수님이신 최형식 교수님과 실험실 식구들이 무엇보다 고맙다. 다른 실험실보다 많은 식구들이 있어 즐거운 대학원 생활이 되었다. 그리고 나의 사랑스런 동문 사람들도.. 나의 가족.. 모두 너무나 소중한 사람들이다. 조금있으면 이제 사회에 첫발을 딪게 된다. 친구들보단 조금 늦게 출발하지만 가슴엔 더 큰 열정과 자신을 가지고 나간다.

추운 겨울 자식 뒷바라지 하시느라 밖에서 고생하시는 아버지, 어머니께 진심으로 감사드린다. 누나, 자형도..

2005.01.06

이 창 만