차 례

로봇 액추에이터 제어 소프트웨어 개발 학습모듈의 개요

학습모듈의 목표

주어진 목표성능과 신뢰성을 만족하도록 로봇의 액추에이터를 제어하는 소프트웨어를 개발하는 능력을 갖출 수 있다.

선수학습

제어 공학, 로봇 공학, 전기전자기초, C 언어, C++ 언어

학습모듈의 내용 체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 로봇 액추에이터 제어 소프트웨어 사양 분석 및 구조 설계하기	1-1. 로봇 액추에이터 구조 설계	1903080302_14v1.1	로봇 액추에이터 사양 분석하기
		1903080302_14v1.2	로봇 액추에이터 제어 소프트웨어 구조 설계하기
2. 로봇 액추에이터 제어기 펌웨어 설계하기	2-1. 로봇 액추에이터 제어기 펌웨어 설계	1903080302_14v1.3	로봇 액추에이터 제어 펌웨어 설계하기
3. 로봇 액추에이터 제어기 게인 설정하기	3-1. 로봇 액추에이터 제어기 게인 설정	1903080302_14v1.4	로봇 액추에이터 제어기 게인 설정하기
4. 로봇 액추에이터 제어기 제어 성능 시험하기	4-1. 로봇 액추에이터 제어기 제어 성능 시험	1903080302_14v1.5	로봇 액추에이터 제어기 제어 성능 시험하기

핵심 용어

통합 개발 환경, 마이크로프로세서, 인터페이스, 프로토콜, 펌웨어 규격서, 라이브러리 규격서, 레지스터, 순서도, 주변기기

학습 1	**로봇 액추에이터 제어 소프트웨어 사양 분석 및 구조 설계하기**
학습 2	로봇 액추에이터 제어기 펌웨어 설계하기
학습 3	로봇 액추에이터 제어기 게인 설정하기
학습 4	로봇 액추에이터 제어기 제어 성능 시험하기

1-1. 로봇 액추에이터 구조 설계

학습 목표

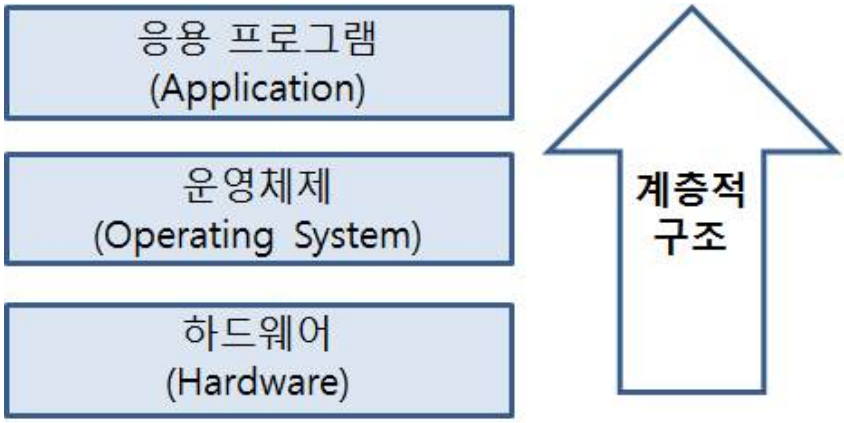
- 로봇 액추에이터에 의해 구동되는 기구의 제원을 분석할 수 있다.
- 로봇 액추에이터를 구동하기 위한 전기회로 제원을 분석할 수 있다.
- 기구와 전기회로의 제원을 분석한 결과로부터 구동 프로그램 작성에 필요한 사항들을 정리할 수 있다.
- 로봇 액추에이터의 동역학적 구동 프로파일을 도출할 수 있다.
- 로봇 액추에이터 제어구조 설계를 위한 요구 사항과 계획을 작성할 수 있다.
- 로봇 제어기 인터페이스와 로봇동작 제어함수를 포함하는 로봇제어 소프트웨어 구조를 파악할 수 있다.
- 로봇 액추에이터 제어 시뮬레이션을 이용하여, 제어로직이 적절하게 구현되었는지 평가할 수 있다.
- 로봇 제어 요구 사항을 만족하는지 여부를 분석할 수 있다.

필요 지식 /

[1] 제어 소프트웨어

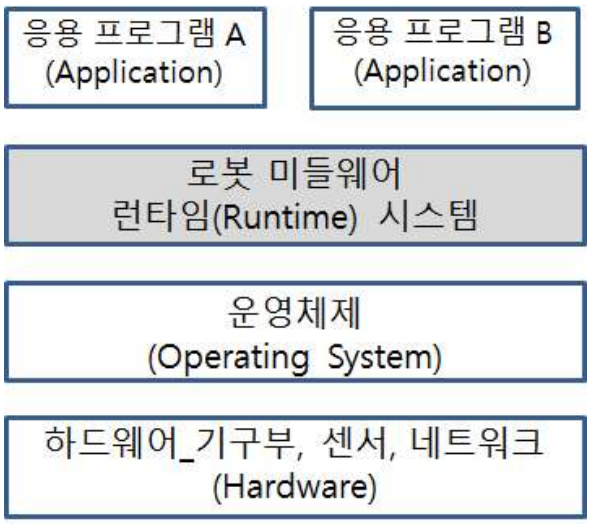
1. 액추에이터 소프트웨어의 구조

일반적으로 소프트웨어는 하드웨어와 함께 컴퓨터 시스템의 주요 구성 요소이다. 하드웨어와 소프트웨어가 함께 구성되어야 컴퓨터 시스템이 존재한다. 일반적으로 컴퓨터 시스템은 계층적(layer)으로 표현할 수 있는데 소프트웨어는 하드웨어의 상부에서 운영되는 것으로 표현된다.



[그림 1-1] 소프트웨어와 하드웨어 구조

하드웨어가 연산장치, 기억장치 그리고 입출력장치로 구분하듯이 소프트웨어는 다시 운영체제와 어플리케이션으로 구분할 수 있다. 어플리케이션은 사용자 응용프로그램으로 워드프로세서 및 게임 등을 예로 생각할 수 있다. 운영 체제는 하드웨어의 관리와 어플리케이션이 실행될 수 있는 환경을 지원한다. 운영 체제와 응용프로그램은 실제 프로그래머가 작성한 코드와 데이터의 집합 프로그램이다. 이들 프로그램은 저수준의 언어 또는 고수준의 언어로 작성되며, 운영 체제는 해당 기술에 특화된 업체 또는 개발자에 의하여 개발되지만 응용프로그램은 운영 체제의 환경에서 응용분야에 적합한 응용프로그램에 의하여 개발된다.



[그림 1-2] 로봇용 소프트웨어의 구조

로봇 시스템에서 컴퓨터는 여러 가지로 생각할 수 있는데 액추에이터와 센서 등을 구성하는 개별 부품 또는 모듈의 MCU와 이들 모듈들을 통합적으로 제어하는 로봇 제어기 또는 로봇 제어용 컴퓨터의 MPU를 생각할 수 있다. 일반적으로 컴퓨터 시스템과 동일하게 로봇

제어용 컴퓨터 또한 운영 체제와 응용프로그램으로 생각할 수 있는데 통산 운영 체제와 응용프로그램 사이에 로봇을 위한 미들웨어 또는 런타임 시스템을 가지는 경우가 많다.

로봇의 운영 체제는 실시간 운영 체제(real-time OS)를 주로 사용하는데 실시간 운영 체제가 아닌 경우 로봇 미들웨어에서 실시간 처리를 지원하는 경우도 있다. 로봇 미들웨어는 운영 체제의 환경에서 로봇을 위해 실행되는 관리 프로그램과 응용프로그램의 작성을 위한 API 등으로 이루어진다.

런타임 시스템은 미들웨어와는 다른 개념으로, 특정언어로 작성된 응용프로그램이 다른 프로그램에 의하여 실행되고 제어되는 개념인데 일반 PC 또는 스마트폰의 자바(JAVA), 가상머신(virtual machine)과 비슷한 개념이고 PC의 윈도우즈 운영 체제 공급업체의 닷넷(.Net)과 비슷한 개념이다.

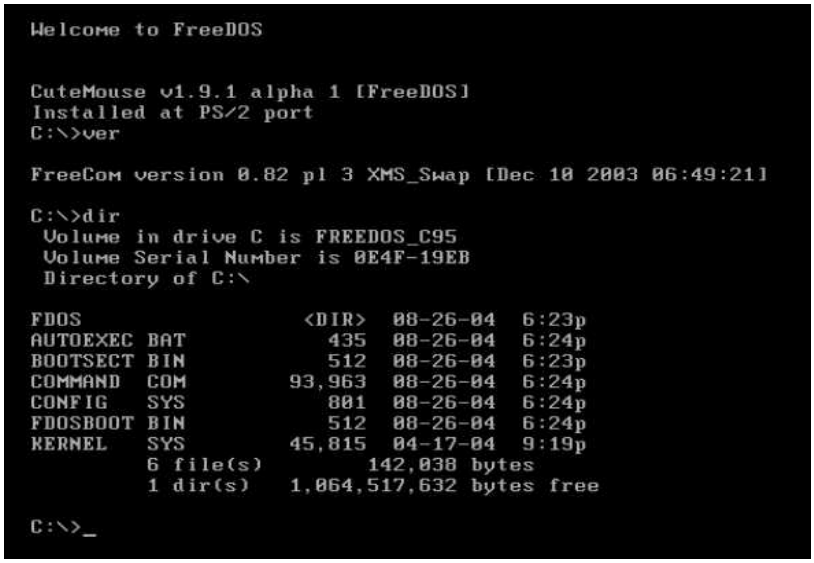
2. 액추에이터 운영 체제(operating system)

운영 체제는 하드웨어의 관리와 어플리케이션의 관리, 그리고 어플리케이션이 하드웨어를 활용할 수 있도록 인터페이스를 제공하는 일련의 소프트웨어의 집합이다. 여기서 관리(management)의 의미는 하드웨어의 접근의 권한과 제한된 하드웨어를 어플리케이션들이 어떻게 공유하는지에 대한 방법들을 의미한다. 로봇의 운영 체제는 PC 또는 워크스테이션에서 사용하는 일반적인 운영 체제를 사용하거나 실시간 처리 능력을 향상하기 위하여 실시간 운영 체제를 사용하기도 한다.

로봇 제어기를 포함하여 일반적인 컴퓨터의 운영 체제가 다수의 프로그램 또는 작업을 동시에 실행시킬 수 있게 하는데 이렇게 다수의 응용프로그램이 실행될 수 있는 운영 체제를 멀티태스킹(multi-tasking) 또는 멀티프로세서(multi-process)를 지원하는 운영 체제라고 한다. 멀티태스킹 운영 체제에서는 실제 실행되는 작업을 부분적으로 시분할(time-sharing)하여 CPU가 실행하게 된다. 다음은 일반적으로 많이 사용되는 운영 체제이며 로봇제어기에도 사용되는 시스템이다.

(1) DOS(disk operating system)

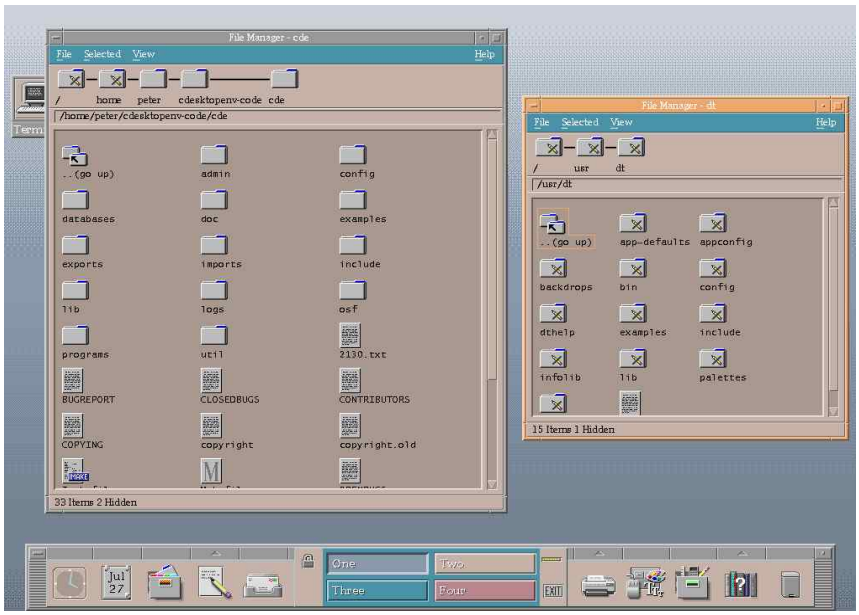
DOS는 윈도우즈가 보편화 되기 전에 PC에서 사용되던 운영 체제이다. 디스크에 저장되었다가 PC가 시작될 때 실행되는 운영 체제라는 의미이다. DOS는 마이크로소프트社의 MS-DOS가 대표적인데 IBM社의 요청으로 IBM社가 개발한 PC를 위하여 개발되었다. DOS는 한 명의 사용자와 한 번에 하나의 프로그램만을 실행 할 수 있는 싱글유저/싱글태스킹 기능을 기본으로 하지만 구조가 간단하여 실행 속도가 빠르기 때문에 윈도우즈가 도입된 후에도 제어 분야에서 사용이 지속되어 왔다.



출처: Wikipedia(2016). https://en.wikipedia.org//에서 2016. 09. 30. 검색.
[그림 1-3] DOS 실행 환경

(2) 유닉스(UNIX)

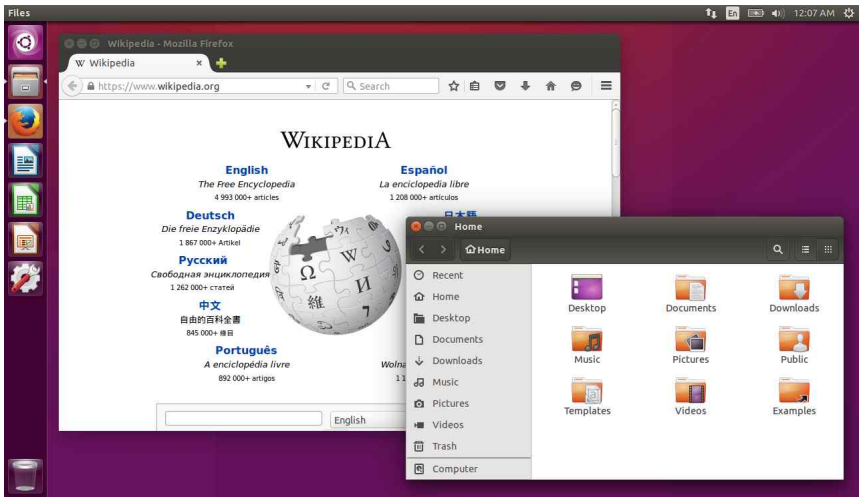
유닉스 운영 체제는 멀티유저/멀티태스킹을 지원하는 운영 체제이며 1969년 미국 전화국인 AT&T에서 개발된 오래된 운영 체제이다. 유닉스 운영제제는 사람들에게 보다 이해가 쉬운 C라는 고수준 언어로 만들어 졌는데 1990년 초까지 PC는 DOS가 기본 운영체제이고 보다 고성능의 워크스테이션 및 대형 컴퓨터들은 UNIX를 기본으로 사용하였다.



출처: Wikipedia(2016). https://en.wikipedia.org//에서 2016. 09. 30. 검색.
[그림 1-4] 유닉스의 표준 데스크탑 화면

(3) 리눅스(LINUX)

리눅스는 유닉스과 비슷한 운영 체제이다. 리누스 토발즈(LINUS TORVALS)라는 학생이 PC에서 실행되는 운영제제를 개발하여 소스 코드까지 공개하면서 널리 알려지게 되었다. 현재는 유닉스를 대체하기도 하며 특수 목적의 제어 및 임베디드 분야에서도 적용된다. 스마트폰의 안드로이드 운영 체제는 리눅스에 기반하고 있다.



출처: Wikipedia(2016). https://en.wikipedia.org//에서 2016. 09. 30. 검색.
[그림 1-5] 리눅스의 데스크탑 화면(UBUNTU)

(4) 윈도우즈(Windows)

마이크로소프트社에서 DOS를 대체하기 위하여 개발된 운영 체제이다. 마이크로소프트社 는1980년대에 DOS에서 실행되는 윈도우즈 운영 체제를 만들었으나 널리 사용되지는 못하였고 1900년대에 Windows95 개발함으로부터 널리 적용되기 시작하였다. 현재 Windows7에 이어 Windows8, Windows8.1,를 출시하였고 최근에는 Windows10을 보급하고 있다.

출처: Wikipedia(2016). https://en.wikipedia.org//에서 2016. 09. 30. 검색.
[그림 1-6] 윈도우의 데스크탑 환경(Windows10)

(5) OS X

OS X는 애플社의 PC인 MAC 컴퓨터를 위한 운영 체제이다. 실제 PC분야에서 애플社는 마이크로소프트社 보다 먼저 윈도우 기술을 적용하였고 OS X 이전의 MAC OS에서부터 사용되었다. MAC OS는 실제 powerPC라는 MPU에서 동작하였으며 현재 인텔(INTEL)社의 64-bit MPU에서 동작한다.

출처: Wikipedia(2016). https://en.wikipedia.org//에서 2016. 09. 30. 검색.

[그림 1-7] OS X의 데스크 탑 환경(OS X El capitan)

실시간 운영 체제는 일반 운영 체제에 비하여 시분할의 방법 및 정밀도가 우수하며 정밀 제어를 위해 폭넓게 사용되는데, 프로세서(프로그램이 실행된 형태) 또는 프로그램 내에서의 작업을 나눈 태스트(task 또는 thread)가 다수 존재하는 경우 이들의 우선순위와 실행되는 빈도를 정밀하게 조정할 수 있다. 예를 들어 로봇의 경우 센서를 처리하는 태스크는 10ms마다 실행하게 하고 모터를 제어하는 태스크는 5ms마다 실행되게 하는 것이다.

실시간 운영 체제는 로봇을 비롯하여 산업 제어 시스템, 통신 장비 및 국방 장비 등 광범위한 분야에 적용되고 있고 다양한 제품 및 방식으로 존재하는데, 크게 일반 운영 체제에 실시간 기능이 덧붙여지는 형태와 전용의 실시간 운영 체제로 개발된 것으로 나눌 수 있다.

(가) 일반 운영 체제 기반 실시간 운영 체제

운도우즈 또는 리눅스와 같은 범용 운영 체제에서 운영 체제 일부를 바꾸거나 미들웨어 형식으로 실시간 기능을 추가한 시스템이다. 윈도우즈에서는 RTX라는 시스템과 리눅스에서는 RTAI라는 추가적 시스템이 많이 사용되고 있다.

(나) 전용 실시간 운영 체제

처음부터 실시간 기능을 중점으로 만들어진 운영 체제이며 수천 가지의 제품이 존재한다. 일반적으로 유명한 것은 WxWorks와 pSos이며 eCos와 FreeRTOS는 무료로 제공되고 있고 소형의 시스템에 많이 응용되고 있다.

2 용어 정리

1. 로봇 운영 체제

운영 체제는 하드웨어의 관리와 어플리케이션의 관리, 그리고 어플리케이션이 하드웨어를 활용할 수 있도록 인터페이스를 제공하는 일련의 소프트웨어의 집합이다.

2. DOF(degree of freedom)

역학계에서 질점의 위치를 정하는 좌표로서 임의의 독립한 변화를 줄 수 있는 것의 수, 공간을 자유로이 운동할 수 있는 n개의 질점계의 자유도는 3n이며, 구속 조건이 m개 있으면 독립된 변수의 수, 즉 자유도는 3n−m이 된다. 자유로운 강체의 자유도는 6이지만, 직선 모양의 분자를 강체로 하여 다룰 때는 결합 방향을 축으로 하는 회전의 자유도는 없는 것으로 쳐서 자유도를 5로 한다.

3. 매니퓰레이터

회전 운동이나 직선 운동을 하는 관절들이 연속적으로 연결된 링크들로 구성된 것이 매니퓰레이터이다. 매니퓰레이터는 직접 운동을 하는 관절과, 이 관절들 사이에 연결된 링크로서 이루어져 있다. 또 관절은 일반적으로 회전 운동을 하는 회전 관절과 직선 운동을 하는 직동 관절(미끄럼 관절)로 나눌 수 있으며, 이러한 관절의 수는 매니퓰레이터의 자유도를 결정한다. (※자유도= 관절의 수)

4. 액추에이터(actuator)

사람의 팔을 움직이는 힘의 근원은 근육인데 반하여, 로봇을 포함한 기계 장치의 구동은 전기 모터에 의한 회전력, 공기압 실린더, 유압 실린더에 의한 직동력이 대부분이다. 로봇이라는 기계를 움직이는 근원은 이와 같은 회전 또는 직동의 동력원을 액추에이터(actuator)라 한다.

5. 토크 (torque)

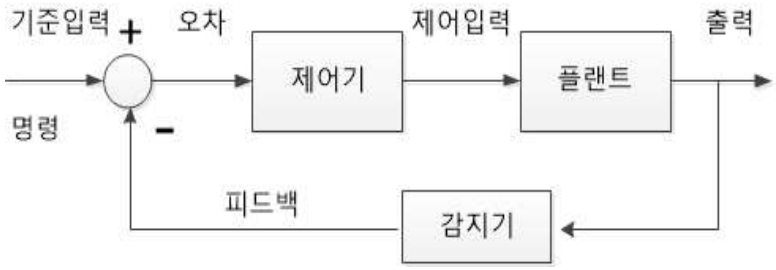
로터의 회전축 중심으로부터 1cm 또는 1m 떨어진 부하(물체)를 단위 시(1초)에 1cm 또는 1m를 움직일 수 있는 힘(단위: gfcm 또는 Kgfcm 등)

6. ZMP(zero moment point)

운동의 제어 시스템에서 동역학적인 운동의 균형을 위해 분석되는 것의 일종이다. 운동의 균형을 잡기 위해서는 흐트러진 운동을 초기 프로그램의 특성을 지닌 운동으로 복원하기 위한 알고리듬이나 기술 등이 필요하다. 그 복원 기술로 사용되는 방법 중 하나로서 ZMP 궤적이 이용된다. 동역학적인 움직임의 불균형은 프로그램으로 허용되는 영역 바깥에 놓인 ZMP 궤적 선분들로 파악될 수 있다.

3 액추에이터 제원 분석

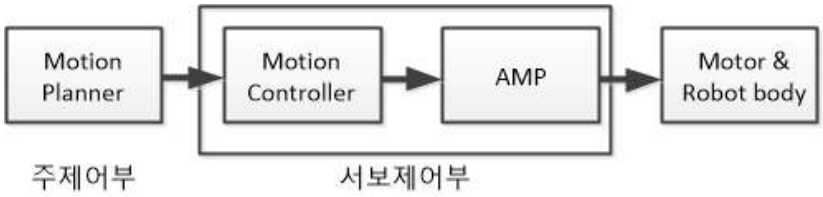
1. 제어란 어떤 시스템의 상태나 출력이 원하는 특성을 따라가도록 입력 신호를 적절히 조절하는 방법을 말하며, 이 동작을 수행하는 장치를 제어기라고 부른다.



[그림 1-8] 제어 시스템의 구성

2. 제어기를 해석적으로 구하려면 플랜트의 수학적 모델을 사용하는데, 제어기 설계 및 구현이 가능하도록 하기 위해서 될 수 있는 한 간단하면서도 실제 시스템의 특성을 잘 나타내는 모델을 기본 모델(nominal model)로 쓴다. 따라서 실제 시스템과 기본 모델 사이에는 오차가 존재하는데, 이 모델링 오차는 기본 모델을 간단하게 정하였기 때문이거나 모델의 매개변수(parameter)가 변하기 때문에 생긴다고 볼 수 있다. 또 <그림1.6>에서 볼 수 있듯이, 실제 플랜트에는 불확실한 성분으로서 모델링 오차 외에 외부에서 들어오는 외란이 있는 경우도 있으며, 출력 신호를 측정하는 경우에 측정 잡음이 발생하기도 한다.

3. 제어기를 써서 이루고자 하는 대상 시스템의 특성을 제어 목표라고 한다. 제어기를 설계할 때에는 불확실한 성분들이 존재하는 실제 환경을 고려하여 안정성 유지, 명령 추종, 외란 제거, 잡음 축소 따위를 제어 목표로 잡는다. 이 제어 목표 가운데 안정성이 이루어지면 많은 경우에 성능 목표도 어느 정도 달성되기 때문에, 안정성을 가장 중요한 목표로 하여 제어기를 설계한다. 따라서 제어기를 설계한다는 것은 안정성을 이루면서 나머지 제어 목표들을 이루는 제어 입력 신호를 만들어내는 장치를 구성하는 것을 뜻한다. 그런데 이 제어 목표들은 상충 관계에 있기 때문에 제어기를 설계할 때에는 이 목표들 사이에 절충을 이룰 수 있도록 제어기의 구조와 설계 변수들을 적절히 조정해야 한다.



[그림 1-9] 로봇 운동 제어부의 구성

4 로봇 동역학

로봇을 설계하거나 이미 설계된 로봇의 움직임을 조사하려면 실제 로봇을 대상으로 다양한 변수값들을 직접 측정하여야 한다. 하지만 로봇의 크기나 가격 등의 문제로 실제 로봇에의 적용이 어려울 경우가 있다. 이러한 경우에 실제 로봇이 움직이는 행동을 조사하려면 로봇을 모델링하면 된다. 먼저 기구학을 통하여 로봇 조인트의 움직임을 직교좌표 공간에서 나타낼 수 있다. 하지만 이러한 로봇의 움직임은 실제 로봇의 움직임을 고려한 것이 아니라 기구학의 수식적인 관계를 나타낸 것이다. 실제 로봇의 움직임을 나타내기 위해서는 각 조인트에 적용되는 힘이나 토크값을 구해야 한다. 각 조인트의 토크값은 다양한 힘으로 구성되어 있는데 이 값을 구하면 토크값에 의한 로봇의 움직임을 동적으로 나타낼 수 있게 된다.

동역학이 주어지면 입력으로 힘이나 토크 값이 주어지고 로봇의 동역학 식으로부터 로봇의 상태변수를 구할 수 있는데, 이 상태변수 값으로부터 기구학을 통하여 로봇의 움직임을 가상적으로 나타내는 각과 위치를 계산 할 수 있다. 역으로 입력으로 각 조인트의 각의 값이 주어지면 토크값을 구하게 되는데 이를 '역동역학'이라 하며, 이는 실제로 로봇 모델을 사용하여 로봇을 제어하는데 유용하다.

따라서 로봇의 동역학을 구하는 목적은 첫째로, 새로운 로봇을 설계할 경우에 그 로봇의 움직임을 미리 알아보기 위함이다. 둘째로, 로봇이 없을 경우에 가상적으로 움직임을 연구하기 위함이다. 셋째로, 새로운 로봇 제어 이론을 실제 로봇에 적용하기에 앞서 가상적으로 로봇이 움직이는 것을 시뮬레이션하기 위함이다. 넷째로, 가상현실과 같은 컴퓨터 그래픽에서 로봇의 현실감이 요구될 경우에 동역학을 사용한 움직임을 구성하기 위함이다. 이처럼 가상적으로 로봇의 움직임을 미리 점검하므로 실제 응용에서 접하게 되는 문제점들을 보완할 수 있다. 하지만 아무리 로봇의 동역학식을 정확하게 구한다 할지라도 실제 로봇과는 일치하지 않으므로 시뮬레이션의 한계가 있기 마련이다. 또 로봇의 모델을 근거로 제어할 때에 실제 로봇과 모델과의 불일치에서 생기는 동역학의 오차의 보정은 많이 연구되고 있는 분야 중의 하나이다.

로봇의 동역학을 구한다 함은 일반화된 좌표에서의 벡터 q와 그에 상관되는 일반 힘 τ와의 관계식을 구하는 것을 말한다. 로봇의 동역학 식을 유도하는 데는 두 가지 방법이 있다. 하나는 newton-euler 방식이고 다른 하나는 euler-lagrangian 방식이다. 각각의 특성은 다음과 같다.

1. newton-euler 방식

이 방식은 동적 시스템을 뉴턴의 두 번째 법칙을 직접적으로 적용하여 각 링크의 좌표에서 링크의 힘과 모멘트로 나타낸 것이다. 따라서 이 방식은 힘의 균형을 기반으로 한 접근 방식으로 동역학 방정식의 유도에 효율적이다. 따라서 시뮬레이션과 컴퓨터 계산에 있어서 효율적이다.

2. euler-lagrangian formulation

시스템의 동적 특성을 일과 에너지의 개념으로 나타낸 것으로 뉴튼 오일러방식보다 간단하다. 에너지를 기반으로 한 동역학적 모델 접근 방법으로 비교적 간단한 운동에서 robot의 운동에 작용하는 여러 가지 변수에 의한 효과를 이해하는 데 유용하다. 일반좌표에 근거한다.

로봇의 동역학 식을 구성하는 성분으로는 관성, 원심력, 코리올리스 힘, 중력 등과 같이 모델이 가능한 성분이 있는 반면에 마찰력, 백래쉬 등과 같은 모델이 어려운 성분들도 있다.

5 newton-euler 방식과 lagrangian euler 방식

1. lagrangian euler 방식

라그랑지안 방적식은 운동에너지와 위치에너지의 차이를 나타낸다. euler-lagrangian 방정식은 시스템의 일과 에너지에 근거해서 나타낸 방정식으로 라그랑지안 방정식 L로부터 구한다. 일반적인 식은 아래와 같은 미분 방정식의 형태로 표현된다.

$$\frac{d}{dt}[\frac{\partial L}{\partial \dot{q}_i}] - \frac{\partial L}{\partial q_i} = \tau_i \qquad \langle 1\text{-}1 \rangle$$

여기서 각 항목들은 다음과 같이 정의된다.

L : lagrangian 함수 = 운동에너지 K - 포텐셜에너지 P

K : 전체 운동에너지

P : 전체 포텐셜에너지

q_i : 조인트 i의 각

$\dot{q}_i$: q_i의 일차 미분

τ_i : 링크 i를 움직이기 위해 조인트 i에 적용되는 힘(토크)

따라서 동적방적식을 구하기 위해 먼저 각 조인트의 운동에너지와 위치에너지를 구한다. 라그랑지안 함수 L을 구하고 미분하여 $\frac{\partial L}{\partial q_i}$와 $\frac{\partial L}{\partial \dot{q}_i}$를 구한 뒤에 $\frac{\partial L}{\partial \dot{q}_i}$를 한번 더 시간으로 미분한 값을 구한 뒤 정리한다.

(1) 운동에너지

무게가 m인 물체가 선형속도 v로 움직일 때 물체의 운동에너지는 다음과 같다.

$$K_l = \frac{1}{2}mv^2 \qquad \langle 1\text{-}2 \rangle$$

마찬가지로 회전하는 물체의 운동에너지는 다음과 같이 표현된다.

$$K_r = \frac{1}{2}Iw^2 \qquad \langle 1\text{-}3 \rangle$$

여기서 I는 관성모멘트로 다음과 같이 표현된다.

$$I = \int_{vol} \rho(r) r^2 dr \qquad \langle 1\text{-}4 \rangle$$

$\rho(r)$는 반지름이 r인 부피의 mass distribution을 나타낸다.
간단하게 만약, m이 점질량(point mass)이면 관성모멘트는 질량에 거리의 제곱을 곱한 다음과 같이 된다.

$$I = mr^2 \qquad \langle 1\text{-}5 \rangle$$

그렇게 되면 회전하는 점질량 물체의 운동에너지는 다음과 같다.

$$K_r = \frac{1}{2}mr^2\dot{\theta}^2 \qquad \langle 1\text{-}6 \rangle$$

만약 선속도 성분이 벡터 라면

$$v = \begin{bmatrix} x_i^2 & y_i^2 & z_i^2 \end{bmatrix}^T \qquad \langle 1\text{-}7 \rangle$$

운동에너지는 다음과 같고

$$K = \frac{1}{2}mvv^T \qquad \langle 1\text{-}8 \rangle$$

회전하는 물체의 운동에너지는 다음과 같다.

$$K = \frac{1}{2} w^T I w \qquad \langle 1\text{-}9 \rangle$$

예를 들어 반지름이 r인 바퀴의 각속도가 w라 하자.
바퀴의 운동에너지는 다음과 같다.

$$K = \frac{1}{2} I w^2 = \frac{1}{2} I \dot{\theta}^2 \qquad \langle 1\text{-}10 \rangle$$

토크를 구하기 위해 $\dot{\theta}$로 미분하면

$$\tau = \frac{d}{dt}[\frac{\partial L}{\partial \dot{q}_i}] = \dot{I}\dot{\theta} \qquad \langle 1\text{-}11 \rangle$$

만약 $I = mr^2$이라 하면 바퀴에 걸리는 토크는 다음과 같다

$$\tau = mr^2 \ddot{\theta} \qquad \langle 1\text{-}12 \rangle$$

(2) 위치에너지

표면으로부터 거리가 h만큼 떨어진 곳에 위치한 질량이 m인 물체의 위치에너지는 다음과 같이 표현된다.

$$P = mgh \qquad \langle 1\text{-}13 \rangle$$

여기서 g는 가속도로 $g = -9.8 m/\sec^2$ 이다.
따라서 lagrangian 함수 L은 다음과 같다.

$$L = K - P = \frac{1}{2} mr^2 \dot{\theta}^2 - mgh \qquad \langle 1\text{-}14 \rangle$$

위 식으로부터 동역학 식을 구할 수 있다.

2. newton euler 방식

newton euler 방식은 lagrangian euler 방식에 비해 계산량이 적어 실시간 제어가 가능하다.

m_i : 링크 i의 전체 질량

$\overline{r_i}$: 원점에서부터 링크 i의 무게 중심점까지의 거리

$\overline{s_i}$: i 좌표의 원점에서부터 링크 i의 무게 중심점까지의 거리

p_{i-1}^{i} : $i-1$좌표에 대한 i좌표의 원점까지의 거리

$\overline{v_i}$: 링크 i의 무게 중심점에서의 선속도

F_i : 링크 i의 무게 중심점에 적용되는 전체 외부 힘

N_i : 링크 i의 무게 중심점에 적용되는 전체 외부 모멘트

I_i : 원점 좌표에 관한 링크 i의 무게 중심에서의 관성

f_i : 링크 $i-1$이 링크 i에 적용하는 힘

n_i : 링크 $i-1$이 링크 i에 적용하는 모멘트

(1) 순방향 계산

(가) 속도 계산

$$v_i = w_i \times p_{i-1}^{i} + v_{i-1} \qquad \text{회전 조인트}$$

$$v_i = z_{i-1}\dot{q}_i + w_i \times p_{i-1}^{i} + v_{i-1} \qquad \text{선운동 조인트}$$

$$w_i = w_{i-1} + z_{i-1}\dot{q}_i \qquad \text{회전 조인트}$$

$$w_i = w_{i-1} \qquad \text{선운동 조인트}$$

(나) 가속도 계산

$$\dot{w}_i = \dot{w}_{i-1} + z_{i-1}\ddot{q} + w_{i-1} \times (z_{i-1}\dot{q}_i)$$

$$\dot{w}_i = \dot{w}_{i-1} + z_{i-1}\ddot{q} + w_{i-1} \times (z_{i-1}\dot{q}_i)$$

$$\dot{w}_i = \dot{w}_{i-1}$$

$$\dot{v}_i = \dot{w}_i \times p_{i-1}^{i} + w_i \times (w_i \times p_{i-1}^{i}) + \dot{v}_{i-1}$$

$$\dot{v}_i = z_{i-1}\ddot{q}_i + \dot{w}_i \times p_{i-1}^{i} + 2w_i \times (z_{i-1}\dot{q}_i) + w_i \times (w_i \times p_{i-1}^{i}) + \dot{v}_{i-1}$$

$$\overline{a_i} = \dot{w}_i \times \overline{s_i} + w_i \times (w_i \times \overline{s_i}) + \dot{v}_i$$

(2) 역방향 계산

$$F_i = m_i \overline{a_i}$$

$$N_i = I_i\dot{w}_i + w_i \times (I_i w_i)$$

$$f_i = F_i + f_{i+1}$$

$$n_i = n_{i+1} + p_{i-1}^i \times f_{i+1} + (p_{i-1}^i + \overline{s_i}) \times F_i + N_i$$

$$\tau_i = n_i^T z_{i-1} + b_i\dot{q}_i$$

$$\tau_i = f_i^T z_{i-1} + b_i\dot{q}_i$$

6 개발 환경 구성

엑추에이터 구동을 위한 임베디드 개발 환경은 호스트라고 불리는 개발용 PC를 준비한 후 RoboEX-Brain과 같은 타깃을 호스트와 케이블로 연결하고 그에 따른 드라이버 설치와 각종 툴 체인을 설치하는 작업으로 구성된다.

1. 호스트와 타깃 연결 이해

PC 기반의 호스트에서 소프트웨어를 개발하면 이를 RoboEX-Brain에 설치하고 디버깅하기 위한 물리적인 통신 연결 방법이 필요하다. 이러한 연결 방법은 RoboEX-Brain의 유지관리 등에서도 사용된다.

(1) 시스템 이미지 설치

호스트에서 개발한 부트로더나 커널과 같은 시스템이다. 소프트웨어를 타깃에 설치하기 위한 기본 연결 방법으로 시리얼이나 이더넷, JTAG 등이 사용되며 최근에는 USB가 많이 사용된다. 시리얼은 가장 고전적인 방법으로 타깃과 호스트간에 RS232C 시리얼 케이블을 연결해 사용하며 속도가 느려 현재는 거의 사용하지 않는다. 이더넷은 10/100M Base-T 크로스 케이블을 타깃과 호스트 간에 연결하는 방법으로 고속의 이더넷 통신을 사용하므로 최근까지 가장 널리 사용해 온 방법이다.

JTAG는 고가의 전용 에뮬레이터를 필요로 하며 프로세서에 내장된 JTAG 셀이라고 불리는 하드웨어와 직접 통신하므로 시리얼이나 이더넷처럼 타깃에 별도의 소프트웨어가 미리 설치되어 있을 필요가 없다. 호스트는 USB와 같은 전용 케이블로 에뮬레이터와 연결하고 에뮬레이터는 JTAG 케이블로 타깃과 연결한 후 호스트에 전용 소프트웨어를 설치해 사용한다. 초기 인텔은 PXA 프로세서를 보급하면서 고가의 JTAG 에뮬레이터 없이 호스트의 프린터 포트와 연결하는 JTAG 케이블 회로도 및 jflash_mm이라는 별도의 소프트웨어를 무료로 제공했으나 현재는 사용되지 않는다.

(2) 모니터링 터미널

키보드나 모니터와 같은 기본 입출력 장치가 없거나 아직 해당 하드웨어 사용을 위한 소프트웨어 작업이 완료되지 않은 상태에서 호스트의 키보드와 모니터를 타깃의 기본 입출력 장치로 사용하기 위한 연결 방법으로, 고속 통신이 필요하지 않으므로 가장 보

편적인 RS232C 시리얼 케이블을 사용한다.

호스트에는 하이퍼터미널(윈도우XP)이나 PuTTY(윈도우/리눅스), 미니컴(리눅스)과 같은 터미널 프로그램을 실행한 후 타깃을 부팅시키면 호스트의 키보드와 모니터를 타깃의 기본 입출력장치로 사용하게 된다. 호스트가 타깃의 기본 입출력장치 역할을 수행할 때 이를 터미널이라고 부른다.

최근에는 시리얼 포트를 사용하지 않는 PC나 노트북이 증가해 시리얼 신호를 USB로 변환하는 칩을 내장하거나 칩이 내장된 전용 케이블(USB-to-serial 컨버터)을 사용하는 사례가 늘고 있다. 타깃에 이러한 칩이 내장된 경우에는 타깃과 호스트 간에는 USB 케이블을 연결하며 호스트는 가상 시리얼 포트를 사용하게 된다.

출처: OO업체(http://www.amazon.com). 2016. 08. 21 스크린샷.

[그림 1-10] USB-to-Serial 컨버터 케이블

(3) 타깃 관리

타깃의 파일 시스템에 접근하거나 쉘을 획득해 명령을 수행하기 위한 연결 방법으로 시리얼이나 이더넷, USB 등이 사용된다. 최근에는 유선 기반의 이더넷 연결이 Wi-Fi 기반의 무선 네트워크로 바뀌고 있으며 Wi-Fi 네트워크 환경을 갖추지 않는 타깃을 위해 고속의 USB 연결도 많이 사용하는 추세다. 타깃에는 서버 역할을 수행하는 전용 소프트웨어가 설치되며, 호스트 역시 해당 서버와 연결되는 클라이언트를 설치해 사용한다. 단, 이러한 소프트웨어들은 표준화된 것을 사용하므로 대부분 무료로 사용할 수 있다.

```
U-Boot 2012.07 (Feb 14 2014 - 15:03:12) for ROBOEXBRAIN

CPU: Exynos5410 Rev2.3 [Samsung SOC on SMP Platform Base on ARM CortexA15]
APLL = 900MHz, KPLL = 600MHz
MPLL = 532MHz, BPLL = 800MHz
DRAM:  2 GiB
WARNING: Caches not enabled

TrustZone Enabled BSP
BL1 version:
PMIC VER : 0, CHIP REV : 6
VDD MIF : 1.00000V
VDD ARM : 1.00000V
VDD INT : 1.00000V
VDD G3D : 1.00000V
VDD KFC : 1.00000V

Checking Boot Mode ... EMMC4.41
MMC:   S5P_MSHC0: 0, S5P_MSHC2: 1
MMC Device 0: 14.7 GiB
MMC Device 1: [ERROR] response error : 00000006 cmd 8
[ERROR] response error : 00000006 cmd 55
[ERROR] response error : 00000006 cmd 2
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   No ethernet found.
Hit any key to stop autoboot:  2
```

출처: 이상문(2014). 『옴니휠 모듈로 배우는 RoboEX 시리즈 아두이노 펌웨어 프로그램 입문』. 한백전자. p. 16
[그림 1-11] 리눅스에서 미니컴을 이용한 타깃 연결

2. 툴 체인 이해

RoboEX-Brain에서 동작하는 임베디드 소프트웨어를 개발하려면 최소한 소스 코드를 빌드할 수 있는 컴파일러와 링커 및 각종 유틸리티로 구성된 툴 체인이 필요하다. 툴 체인은 개발 도구들의 집합으로 여기에 포함된 개발 도구들은 유기적으로 사용된다.

리눅스 커널을 비롯해 리눅스 기반의 공개용 플랫폼들은 RoboEX-Brain 소프트웨어를 개발하는데 필요한 호스트 툴도 소스 코드로 함께 제공되므로 툴 체인은 호스트와 RoboEX-Brain용 모두 필요하다.

(1) 호스트 툴 체인

리눅스 기반의 타깃용 소프트웨어를 개발할 때는 리눅스용 호스트를 권장하므로 오픈 플랫폼인 GCC툴 체인이 가장 널리 쓰인다. C/C++ 컴파일러를 비롯해 링커, 어셈블러, 라이브러리 등으로 구성되며 개발자용 리눅스 호스트에는 기본적으로 설치되어 있다.

주의할 점은 안드로이드 플랫폼처럼 배포 소스를 가져다가 작업할 때는 툴 체인의 버전이 배포한 개발자가 사용한 버전과 일치해야 할 경우가 있다는 점이다. 간혹 상위 버전이나 너무 하위 버전을 사용하며 빌드 할 때 오류가 발생하기도 한다.

(2) 크로스 툴 체인

타깃용 소스 코드를 빌드 할 때도 툴 체인이 필요하다. 호스트 툴 체인과 다른 점은 빌드 한 결과를 호스트에서 실행하는 것이 아니라 타깃에서 실행한다는 점인데 이를 크로스 툴 체인이라 한다. 임베디드 리눅스 기반의 타깃용 소프트웨어 역시 GCC 체인을 주로 사용하며, 호스트 운영 체제 종류와 타깃의 프로세서 아키텍처에 따라 크로스 툴 체인 종류가 달라지는데, 리눅스 호스트에서 ARM 아키텍처를 사용하는 타깃용 소스를 빌드 할 때는 리눅스 버전의 ARM용 GCC 크로스 툴 체인을 사용한다.

윈도우 호스트를 사용한다면 Cywin 기반의 ARM용 GCC 크로트 툴 체인을 사용하면 된다. 크로스 툴 체인의 경우 리눅스용은 바이너리 상태로 배포되지만 Cywin용은 권장 사항이 아니므로 경우에 따라서는 GCC 툴 체인 소스 코드를 다운로드해 직접 생성해야 하는 경우도 있다.

수행 내용 / 로봇 액추에이터 제어 소프트웨어 구조 설계하기

재료 · 자료

- 전자 부품 카탈로그
- 도면 용지
- 해당 장비 매뉴얼
- 설계 사양서
- KS 및 ISO 규정집
- 기구 도면

기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 기계 설계(CAD) 프로그램
- 프로그램 성능 평가용 에뮬레이션 소프트웨어
- 로봇 액추에이터 제어 성능시험 소프트웨어

안전 · 유의 사항

- layout은 개발 제품에 필요한 전자 부품 및 기계 요소 부품을 이해하여 작성한다.
- 동작 선도를 통한 운동 성능을 구현하도록 개발기획보고서의 내용을 숙지한다.
- 각종 부품 배치 및 인터페이스 관련 지식을 갖춘다.

수행 순서

1 AndroX Studio를 설치한다.

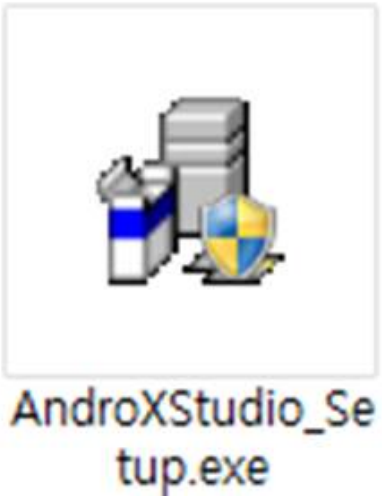
AndroX Studio는 리눅스 커널 기반의 스마트 디바이스 통합개발 환경으로 호스트 및 ARM 크로스 툴 체인을 포함해 플랫폼 개발에 필요한 다양한 툴들을 통합해 제공한다. 무엇보다 한 번의 설치로 윈도우 호스트 상에서 리눅스 커널 및 디바이스 드라이버의 개발이 가능하고, 타깃 관리에 필요한 다양한 툴을 제공해 개발 환경 구축에 필요한 시간과 예외 상황을 줄일 수 있다.

호스트 운영 체제는 윈도우7 64bit를 권장하나 윈도우XP 및 윈도우8도 실행에 문제가 없다. C드라이버에 10GB(gigabyte) 정도의 여유 공간을 필요로 하며 RAM은 4GB 이상, CPU는 i5 정도면 사용할 수 있다.

1. 안드로엑스 스튜디오 설치

AndroX Studio는 C 드라이버에만 설치되므로 전용 하드웨어나 소프트웨어로 C 드라이버가 보호 되는 컴퓨터는 보호를 해제한 후 설치한다.

(1) 제공되는 AndroX Studio DVD의 루트 경로에서 AndroXStudio_Setup.exe를 실행한다.

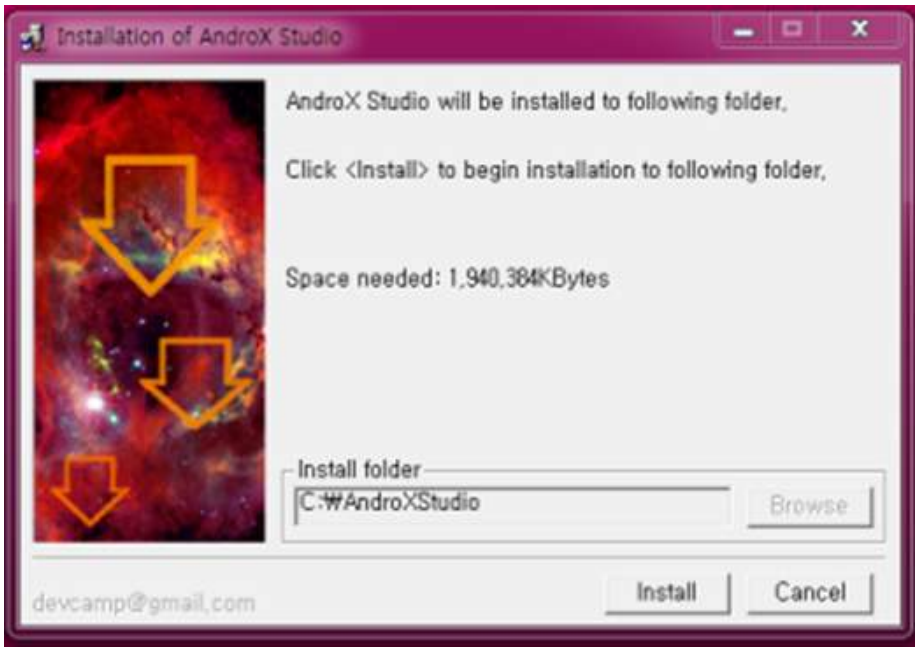


[그림 1-12] AndroX Studio 실행화일

(2) 설치 화면이 나타나면 Next 버튼을 누른다.

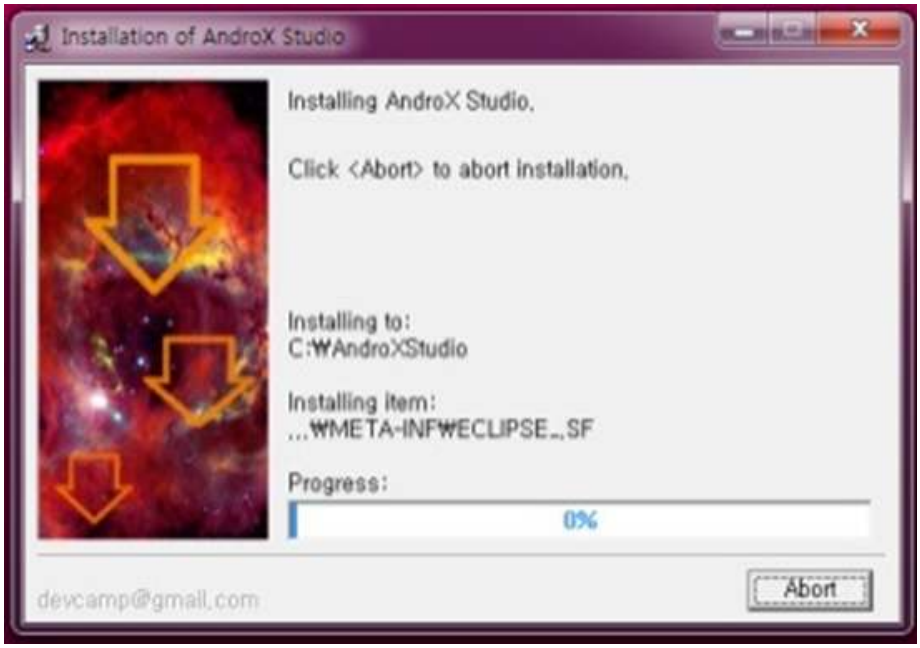
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p19.
[그림 1-13] AndroX Studio 설치 화면

(3) 설치 경로는 "C:\AndroXStudio"로 고정되어 있어 경로 변경은 불가능하다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p19.
[그림 1-14] AndroX Studio 경로 설정

(4) 호스트 환경에 따라 수분 정도 시간이 소요된다.

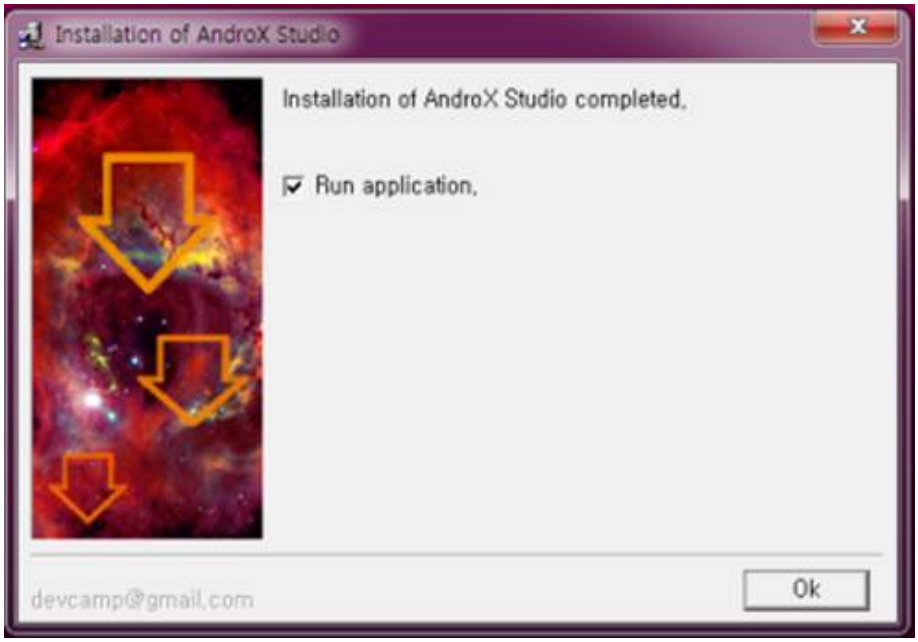


출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p19.
[그림 1-15] AndroX Studio 설치진행 환경

(5) 레지스트리에 값을 추가하는 창이 실행되면 예를 선택한다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p19.
[그림 1-16] AndroX Studio 레지스트리 편집 추가 설정

(6) 설치 완료



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.19
[그림 1-17] AndroX Studio 실행 완료

2. AndroX Studio 설치 결과

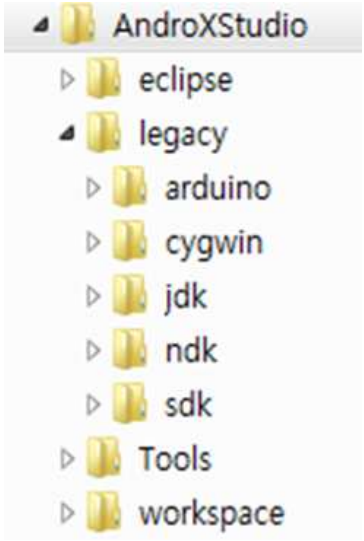
AndroX Studio를 설치한 후 탐색기로 C드라이버를 확인해 보면 루트 경로에 AndroXStudio 폴더가 위치한다. C:\AndroXStudio\AndroXStudio.exe가 메인 실행 파일인 런처 프로그램이며 그 밖에 하위 폴더의 기능은 다음과 같다.

(1) eclipse : 임베디드 리눅스 커널 및 디바이스 드라이버와 아두이노 펌웨어, 안드로이드 네이티브 라이브러리 등을 개발하는데 사용하는 이클립스 플랫폼이 위치한다.

(2) legacy : 임베디드 리눅스 커널과 ARM 크로스 컴파일러가 포함된 Cygwin을 비롯해 자바 개발환경인 JDK와 아두이노 플랫폼 및 안드로이드 NDK와 SDK가 위치한다.

(3) tools : 타깃 관리용 툴들이 위치한다.

(4) workspace : 이클립스에서 생성하는 프로젝트들이 위치하게 된다. workspace 내부에는 이클립스 내부에서 사용하는 툴 및 환경 설정과 관련된 파일들이 위치하므로 함부로 삭제하면 안 된다.



[그림 1-18] AndroX Studio 폴더 구조

AndroX Studio의 실행은 윈도우 시작 메뉴에서 AndroX Studio를 실행하면 된다. AndroX Studio를 실행하면 화면 상단 중앙에 런처가 실행되며, 사용자는 이 런처를 통해 작업에 필요한 프로그램을 실행하게 된다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.20.
[그림 1-19] AndroX Studio 런처

2 케이블을 연결하고, 드라이버를 설치한다.

이제 실습 진행에 필요한 호스트와 RoboEX-Brain 간의 케이블 연결 방법과 디바이스 드라이버 설치 방법에 대해 알아보도록 한다. RoboEX-Brain은 시스템 이미지 설치 및 모니터링과 관리를 위해 2개의 USB 케이블을 연결해 사용한다. 1개는 시스템 이미지 설치 및 관리를 위한 것이고, 나머지 하나는 모니터링 터미널을 위한 것이다.

1. 전원 공급과 RoboEX-Brain 부팅

RoboEX-Brain의 DC 잭에 DC 어댑터를 연결하면 전원이 공급되면서 RoboEX-Brain이 부팅된다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.21.
[그림 1-20] 전원 공급과 부팅

2. RoboEX-Brain 관리를 위한 케이블 연결과 adb 드라이버 설치

안드로이드 adb 서비스는 호스트와 타깃 간 USB 또는 TCP/IP 연결을 통해 파일 설치, 원격 디렉터리 탐색, 디버깅, 원격 쉘과 같은 타깃 관리에 필요한 많은 기능들을 제공한다. 호스트에서 개발한 프로그램 역시 adb 서비스를 통해 RoboEX-Brain에 설치한다.

(1) RoboEX-Brain adb 서비스를 위한 드라이버 설치와 연결 확인

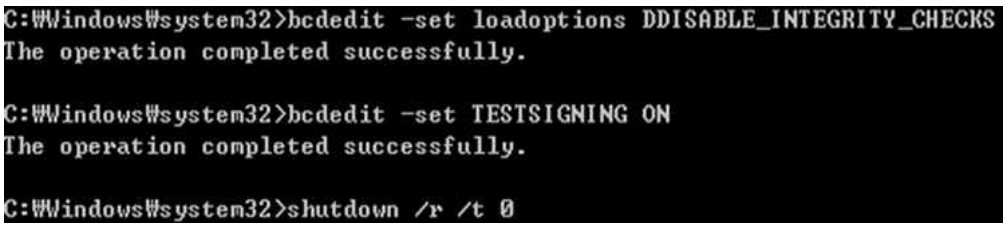
adb 서비스를 위한 케이블 연결 및 드라이버 설치 절차는 다음과 같다.

(가) RoboEX-Brain이 부팅된 상태에서 USB 3.0 OTG 포트에 마이크로USB 타입 커넥터를, 호스트에 USB A 타입 커넥터를 연결한다. 마이크로USB 타입 커넥터는 USB 3.0 커넥터의 왼쪽 포트에 연결한다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』
한백전자. p.22
[그림 1-21] 마이크로USB 타입 커넥터의 호스트USB A타입과 연결

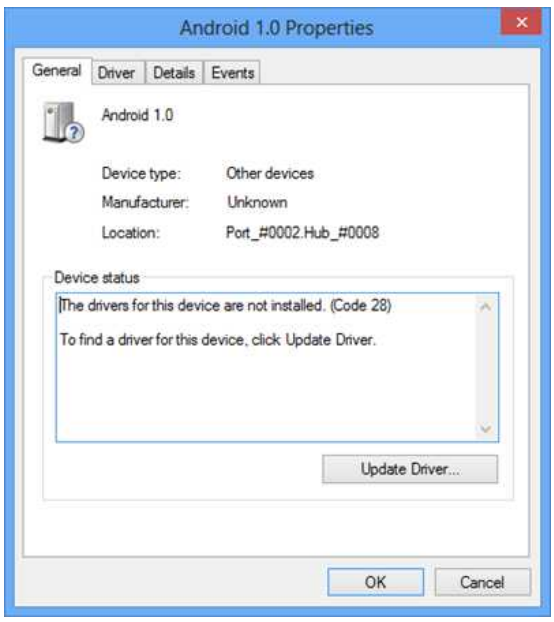
(나) 드라이버 설치 전이라면 윈도우 XP와 윈도우 7은 새 장치 인식과 함께 드라이버 설치를 요구 받게 되는데 수동 설치를 위해 진행을 취소한다. 윈도우 8은 별도 창이 표시되지 않는다.

1) 윈도우에서 adb 드라이버를 설치하려면 반드시 “관리자 명령 프롬프트“에서 다음 명령을 수행한 후 진행해야 한다.
2) 윈도우 키를 눌러 윈도우 런처로 이동한다.
3) 키보드로 “cmd“를 입력한 후 Shift+Ctrl+Enter 키를 눌러 관리자 명령 프롬프트를 실행한다.
4) bcdedit -set loadoptions DDISABLE_INTEGRITY_CHECKS 명령을 실행한다.
5) bcdedit -set TESTSIGNING ON 명령을 실행한다.
6) shutdown /r /t 0 명령을 실행하면 윈도우 8이 리부팅 된다.



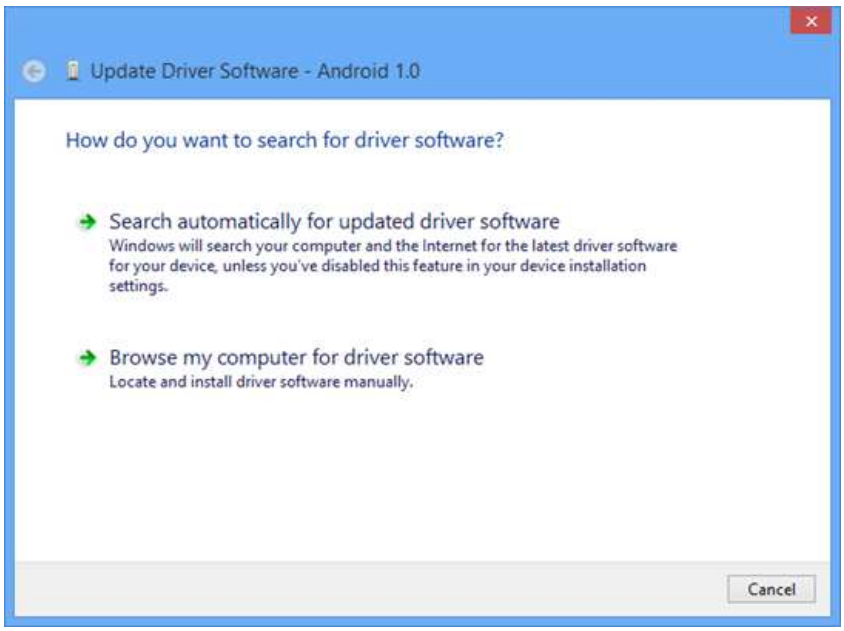
[그림 1-22] 윈도우 cmd에서 관리자 명령 프롬프트 실행 결과

(다) RoboEX-Brain용 adb 드라이버를 수동으로 설치하기 위해 장치 관리자를 실행한 후 [기타 장치 > Android 1.0] 항목을 마우스로 더블클릭해 속성 창을 실행한다 (표시 이름은 다를 수 있다).



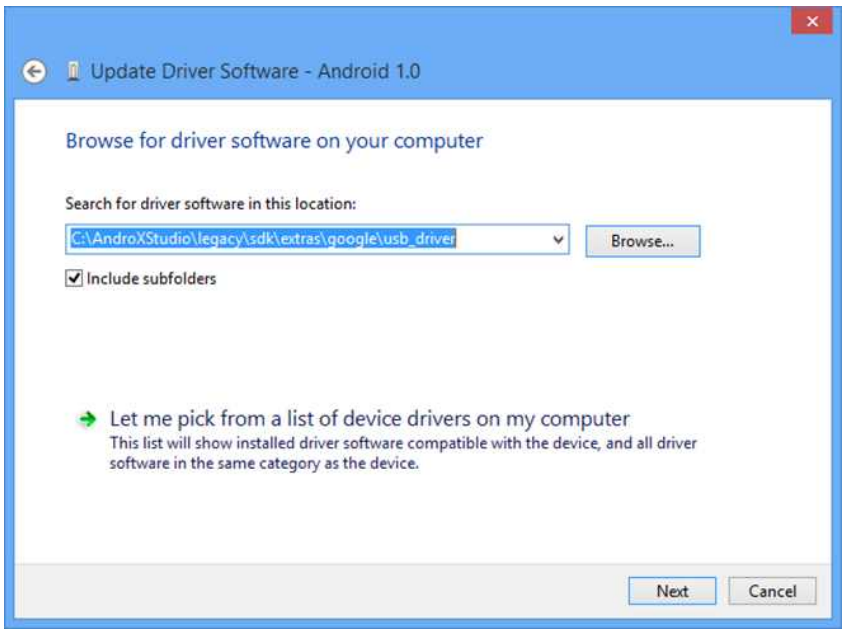
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.23
[그림 1-23] RoboEX-Brain의 속성 대화창

(라) 속성 창이 실행되면 드라이버 업데이트 버튼을 선택한 후 컴퓨터에서 드라이버 소프트웨어 찾아보기를 선택한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.23
[그림 1-24] 드라이브 소프트웨어 찾아보기 대화창

(마) 드라이버 설치 경로로 C:\AndroXStudio\legacy\sdk\extras\google\usb_driver를 지정한 후 다음을 선택한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.24
[그림 1-25] 드라이브 설치 경로 대화창

(바) 보안 경고 창이 실행되면 "이 드라이버 소프트웨어를 설치합니다."를 선택해 드라이버 설치를 계속한다. 드라이버 설치가 완료되면 윈도우 장치 관리자를 통해 드라이버 설치 결과를 확인한다.

[그림 1-26] Android Device 설치 확인

(사) 안드로엑스 스튜디오의 실행에서 Command를 실행한다. 명령 창이 표시되면 adb devices 명령을 실행한다. "0123456789ABCDEF device" 메시지가 표시되면 정상적으로 연결된 것이다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.24
[그림 1-27] 드라이버 정상 철치에 따른 Command 실행 결과

(아) 메시지가 정상적으로 표시되지 않을 때는 다음과 같이 조치해본다.

- USB 케이블을 분리한 후 다시 연결해본다.
- 장치 관리를 통해 드라이버가 정상적으로 인식되는지 확인해본다.

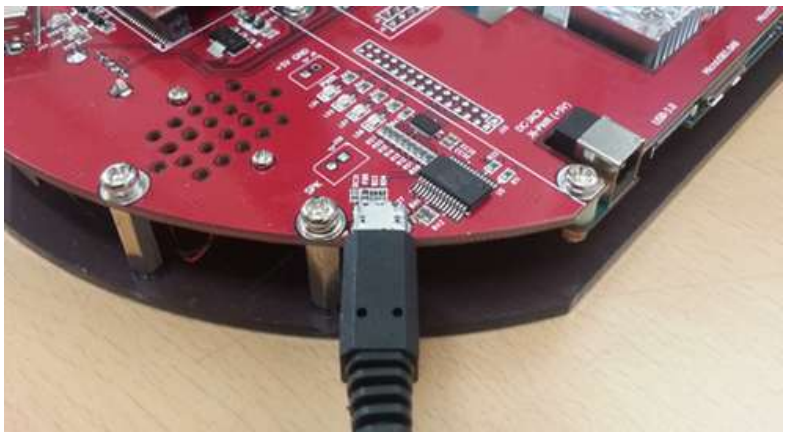
3. 모니터링 터미널을 위한 케이블 연결과 가상 시리얼 드라이버 설치

임베디드 모듈에는 호스트의 키보드와 모니터를 RoboEX-Brain의 기본 입출력 장치로 사용하는 시리얼 연결을 USB로 변환하는 칩을 내장하고 있으므로 모니터링 터미널을 위한 통신 연결 방법은 USB 케이블을 사용한다.

(1) RoboEX-Brain 터미널 연결을 위한 가상 시리얼 드라이버 설치와 연결 확인

터미널 연결을 위한 케이블 연결과 드라이버 설치는 다음과 같다.

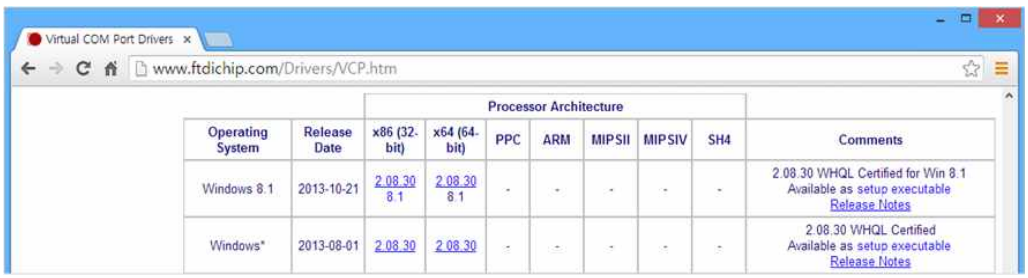
(가) RoboEX-Brain의 Console 포트에 새로운 마이크로USB 타입 커넥터를 연결하고 호스트에 USB A 타입 커넥터를 연결한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.25
[그림 1-28] 콘솔 포트에 마이크로USB 타입 커넥터 연결

(나) 드라이버 설치 전이라면 윈도우 XP와 윈도우 7은 새 장치 인식과 함께 드라이버 설치를 요구 받게 되는데 수동 설치를 위해 진행을 취소한다. 윈도우 8은 별도 창이 표시되지 않는다.

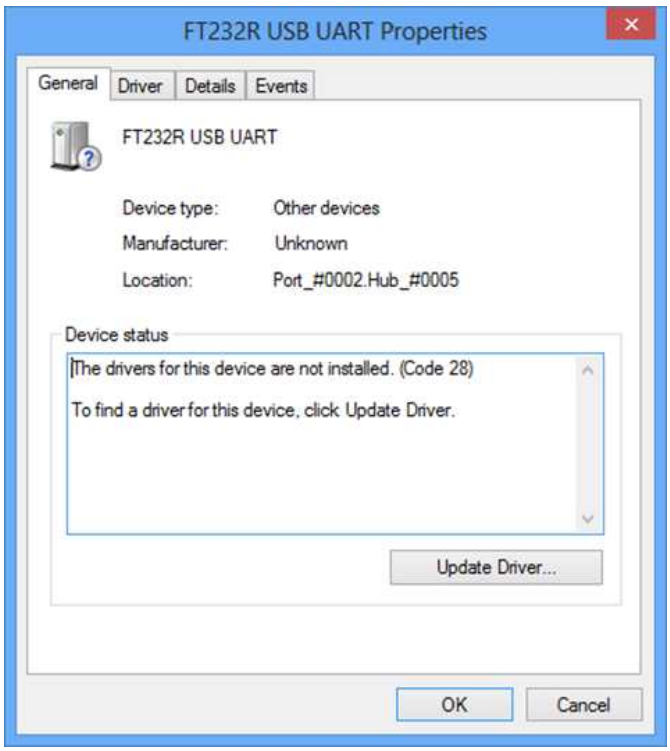
(다) http://www.ftdichip.com/Drivers/VCP.htm 사이트에서 최신 배포 파일을 다운로드해 적당한 위치에 압축을 해제한다. 64bit 운영 체제는 x64(64-bit) 드라이버를 다운로드 한다.



Operating System	Release Date	Processor Architecture: x86 (32-bit)	x64 (64-bit)	PPC	ARM	MIPSII	MIPSIV	SH4	Comments
Windows 8.1	2013-10-21	2.08.30 8.1	2.08.30 8.1	-	-	-	-	-	2.08.30 WHQL Certified for Win 8.1 Available as setup executable Release Notes
Windows*	2013-08-01	2.08.30	2.08.30	-	-	-	-	-	2.08.30 WHQL Certified Available as setup executable Release Notes

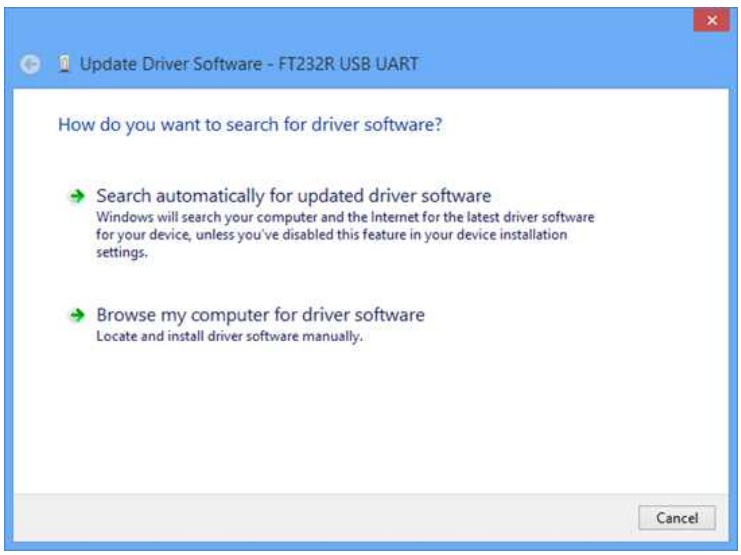
출처: FTDIchip(2016). virtual COM Port Driver. http://www.ftdichip.com//에서 2016. 9. 30. 검색.
[그림 1-29] ftdichip 사이트의 드라이브 배포 그림 제목

(라) RoboEX-Brain용 가상 시리얼 드라이버를 수동으로 설치하기 위해 장치 관리자를 실행한 후 [기타 장치 > FT232R USB UART 항목을 마우스로 더블클릭해 속성 창을 실행한다.



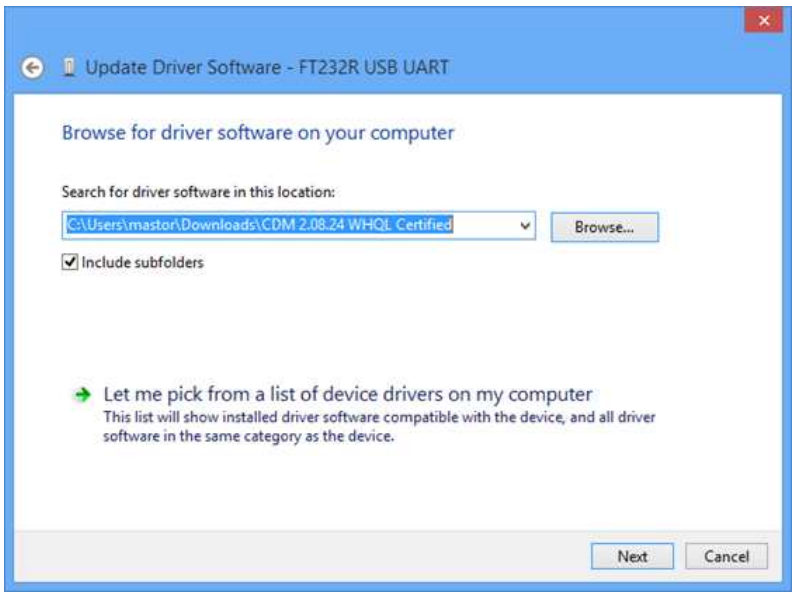
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.26
[그림 1-30] FT232R USB UART 항목 속성 대화창

(마) 속성 창이 실행되면 드라이버 업데이트 버튼을 선택한 후 컴퓨터에서 드라이버 소프트웨어 찾아보기를 선택한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.26
[그림 1-31] 속성창의 드라이버 업데이트 대화창

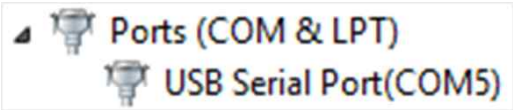
(바) 드라이버 설치 경로로 http://www.ftdichip.com 사이트에서 다운로드해 압축을 해제한 경로를 지정한 후 다음을 선택한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.27
[그림 1-32] 드라이버 다운로드 경로 설정 대화창

(사) 윈도우 XP와 윈도우 7은 FT232R USB UART 드라이버 설치가 완료되면 USB Serial Port 드라이버 설치가 자동으로 시작된다. 윈도우 8은 수동으로 설치해야 한다. 앞과 동일한 방법으로 설치를 진행한다.

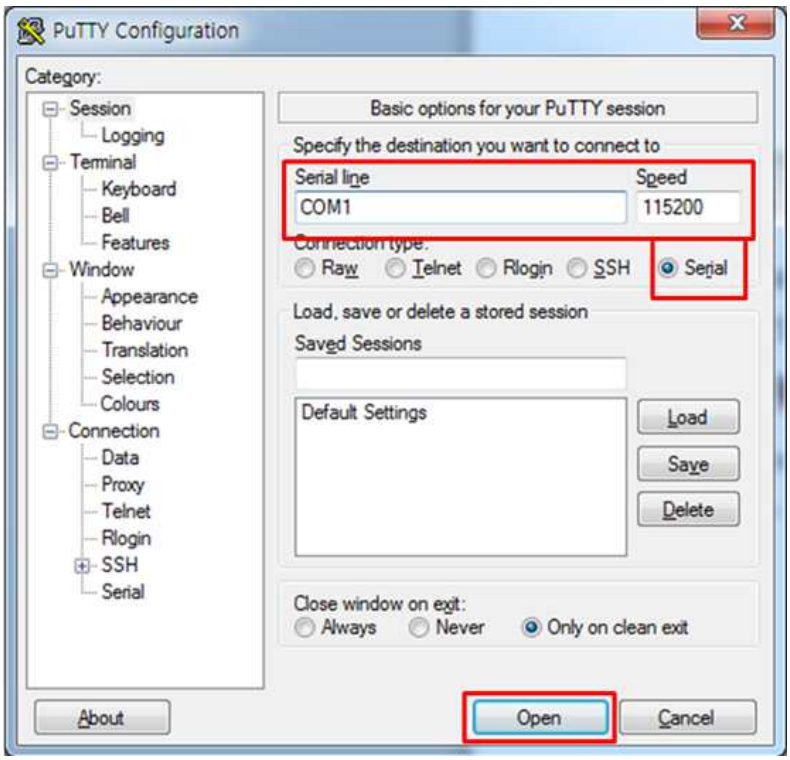
(아) 윈도우 장치 관리자를 통해 드라이버 설치 결과를 확인한다. 할당되는 포트 번호는 사용자마다 다를 수 있다.



[그림 1-33] 윈도우 장치 관리자를 통한 드라이버 설치 확인

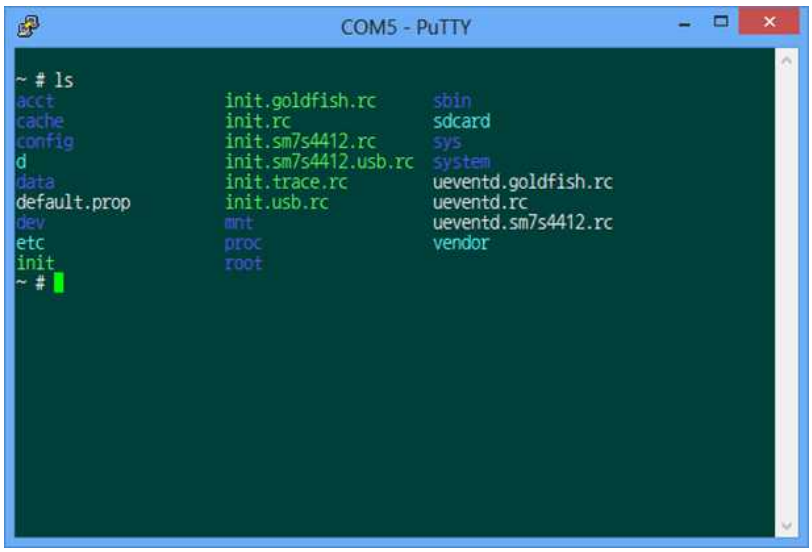
(자) 터미널 프로그램을 실행하기 위해 윈도우 시작 메뉴에서 프로그램 > AndroX Studio > putty를 실행한다.

(차) Connection type을 Serial로 변경해준다. 그 다음 Serial line은 장치 관리자를 통해 COM Port를 확인하여 적어주고 Speed는 115200이다. 완료되면 Open을 클릭한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.28
[그림 1-34] 드라이버 통신 속도 설정

(카) 창이 실행되면 Enter키를 누른다. RoboEX-Brain과 정상적으로 연결되면 RoboEX-Brain의 내부 정보 및 디버깅 메시지 표시를 비롯해 리눅스 명령을 입력할 수 있는 상태가 된다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.28
[그림 1-35] RoboEX-Brain의 내부 정보 및 디버깅 메시지 표시

4. 시스템 이미지 설치를 위한 케이블 연결과 fastboot 드라이버 설치

안드로이드의 fastboot 서비스를 이용하면 부트로더나 커널과 같은 시스템 이미지를 쉽게 RoboEX-Brain에 설치할 수 있다. fastboot 서비스를 이용하기 위해서는 먼저 RoboEX-Brain을 부트로더로 부팅한 후 fastboot 모드로 전환해야 한다. 이후 호스트와 RoboEX-Brain 간에 USB 케이블을 연결하면 윈도우 운영 체제는 fastboot 장치를 인식한 후 디바이스 드라이버를 로드하고 호스트용 fastboot 프로그램을 이용해 시스템 이미지를 RoboEX-Brain으로 전송할 수 있게 된다.

(1) RoboEX-Brain fastboot 서비스를 위한 드라이버 설치와 연결 확인

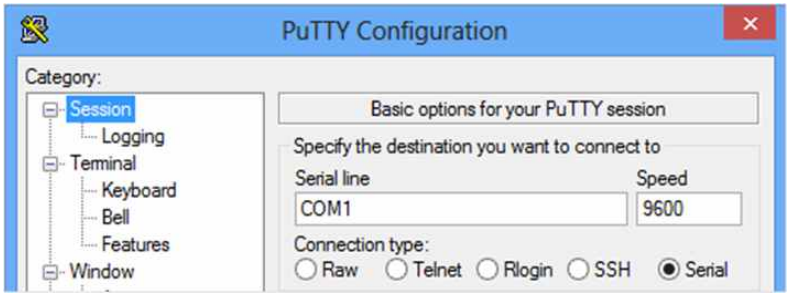
fastboot 서비스를 사용하기 위한 케이블 연결과 fastboot 장치 드라이버 설치 과정은 다음과 같다.

(가) 부트로더의 fastboot 모드로 진입하기 위해 RoboEX-Brain의 전원이 꺼진 상태에서 진행한다.

(나) RoboEX-Brain의 USB 3.0 OTG 포트에 adb 서비스를 위한 케이블 연결과 동일한 방법으로 마이크로USB- USB 타입 A 케이블을 연결한다. 그 다음 Console 포트에 새로운 마이크로USB 타입 커넥터를 연결하고 호스트에 USB A 타입 커넥터를 연결한다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.29
[그림 1-36] 마이크로USB- USB 타입 A 케이블 연결과 호스트의 USB A 타입 커넥터 연결

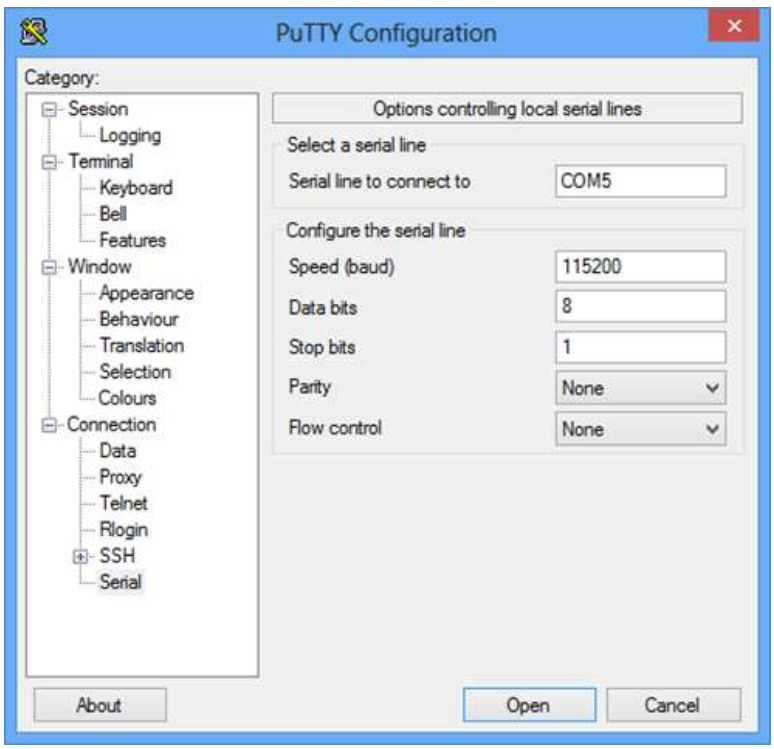
(다) 터미널 프로그램을 실행하기 위해 윈도우 시작 메뉴에서 프로그램 > AndroX Studio > putty를 실행한다. PuTTY Configuration 창이 실행되면 [Session] 항목에서 Serial를 체크한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.29
[그림 1-37] Configuration의 Serial 통신속도 설정

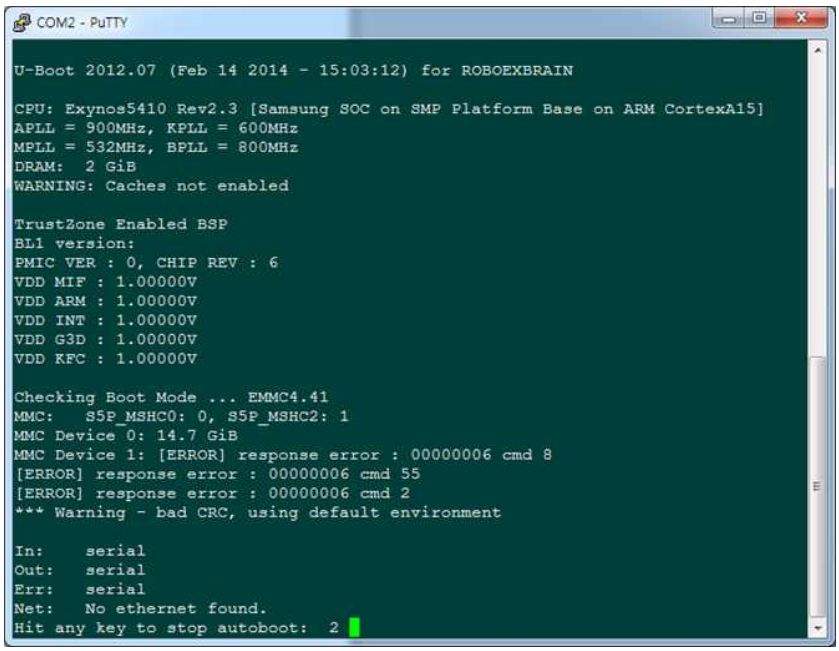
(라) [Serial] 카테고리를 선택한 후 [Serial line to connect to] 할당된 가상 시리얼 장치 이름을 입력하고 [Speed]는 115200을 입력하고, [Flow control]은 None을 선택한 후 Open 버튼을 누른다.

(다) (라)의 진행은 RoboEX-Brain 터미널 연결을 위한 가상 시리얼 드라이버 설치와 연결 확인과 동일하게 진행하면 된다. Putty가 연결된 화면은 닫지 않고 유지한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.30
[그림 1-38] Connection의 serial의 환경 설정

(마) RoboEX-Brain에 전원을 인가한다. 전원을 인가하게 되면 Putty창에 다음과 같은 메시지가 나타난다.



[그림 1-39] 전원 인가에 따른 Putty창의 명령 상황

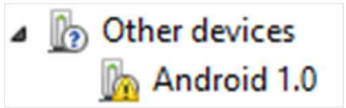
(바) 위의 화면이 출력되고 3초의 카운트가 진행된다. 이때 키보드의 버튼을 눌러서 부팅을 중지한다. fastboot 명령을 입력해준다.

```
Hit any key to stop autoboot:  0
ROBOEXBRAIN # fastboot
there are pending interrupts 0x00000001
[Partition table on MoviNAND]
ptn 0 name='fwbl1' start=0x2 len=N/A (use hard-coded info. (cmd: movi))
ptn 1 name='bl2' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 2 name='bootloader' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 3 name='tzsw' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 4 name='kernel' start=N/A len=N/A (use hard-coded info. (cmd: movi))
ptn 5 name='ramdisk' start=N/A len=0x0(~16777216KB) (use hard-coded info. (cmd
: movi))
ptn 6 name='system' start=0x2 len=0x0(~1073741824KB)
ptn 7 name='userdata' start=0x2 len=0x0(~-2147483648KB)
ptn 8 name='cache' start=0x2 len=0x0(~268435456KB)
ptn 9 name='fat' start=0x2 len=0x0(~-750780416KB)
USB cable Connected![0x4]
```

[그림 1-40] fastboot 명령 리스트

(사) 드라이버 설치 전이라면 윈도우 XP와 윈도우 7은 새 장치 인식과 함께 드라이버 설치를 요구 받게 되는데 수동 설치를 위해 진행을 취소한다. 윈도우 8은 별도 창이 표시되지 않는다.

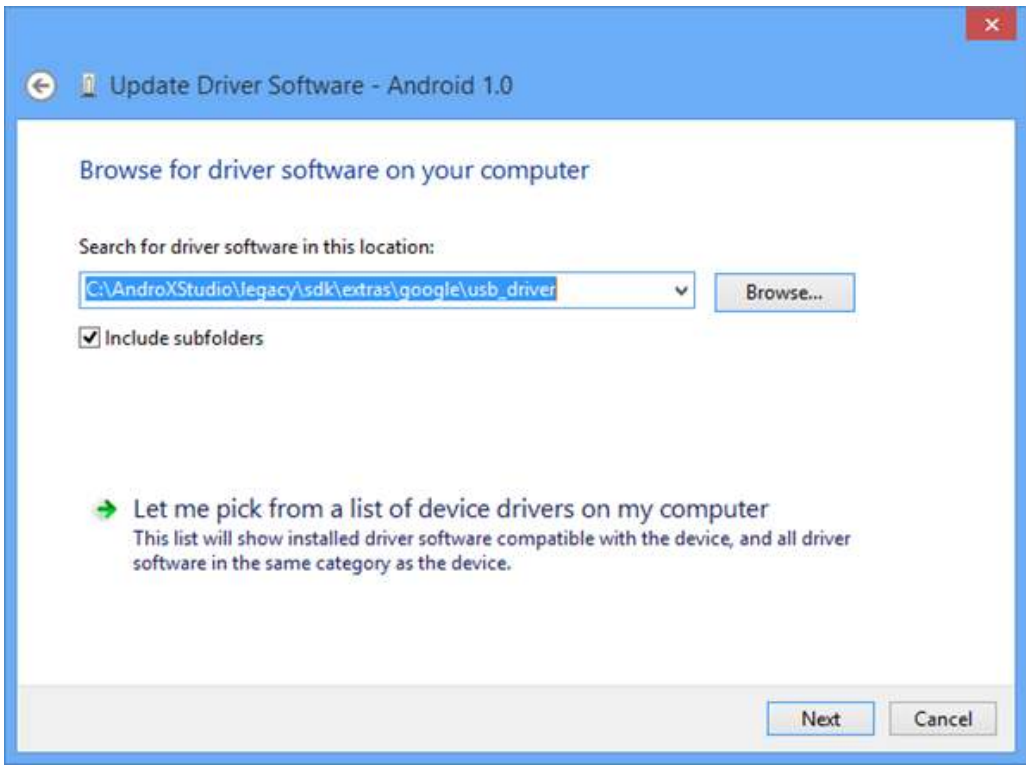
(아) RoboEX-Brain은 adb와 fastboot 서비스에 같은 드라이버를 사용하므로 fastboot 드라이버 설치는 adb 드라이버와 동일한 방법으로 한 번 더 진행하면 된다. 수동으로 설치하기 위해 장치 관리자를 실행한 후 [기타 장치 > Android 1.0] 항목을 마우스로 더블클릭해 속성 창을 실행한다.



[그림 1-41] Android 1.0항목 속성 실행

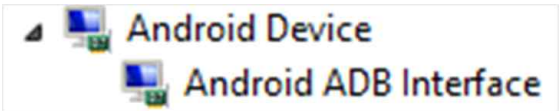
(자) 속성 창이 실행되면 드라이버 업데이트 버튼을 선택한 후 컴퓨터에서 드라이버 소프트웨어 찾아보기를 선택한다.

(차) 드라이버 설치 경로로 C:\AndroXStudio\legacy\sdk\extras\google\usb_driver를 지정한 후 다음을 선택한다.



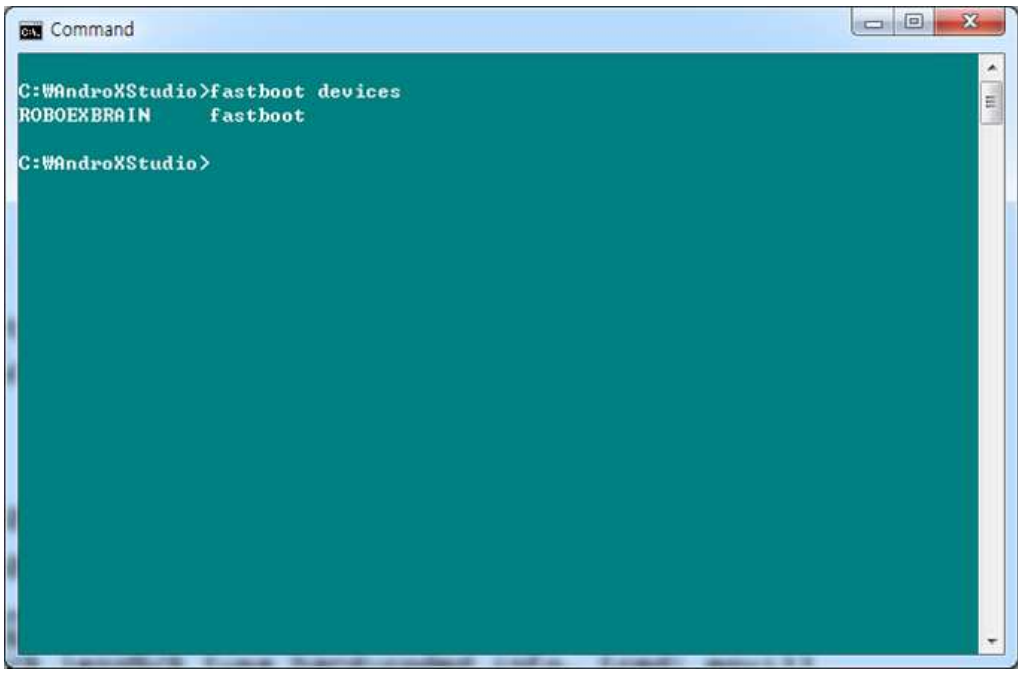
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.32
[그림 1-42] Addroid 1.0 드라이버 설치 경로

(카) 보안 경고 창이 실행되면 "이 드라이버 소프트웨어를 설치합니다." 를 선택해 드라이버 설치를 계속한다. 드라이버 설치가 완료하면 윈도우 장치 관리자를 통해 드라이버 설치 결과를 확인한다. RoboEX-Brain의 fastboot은 adb 드라이버를 사용하므로 장치 관리자에도 Android ADB Interface로 표시된다.



[그림 1-43] Addroid 1.0 드라이버 실치완료 확인

(타) 안드로엑스 스튜디오의 런처에서 Command를 실행한다. 명령창이 표시되면 fastboot devices 명령을 실행한다. “ROBOEXBRAIN fastboot“ 메시지가 표시되면 정상적으로 연결된 것이다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용프로그램 입문』 한백전자. p.33
[그림 1-44] 안드로엑스 스튜디오 실행에 따른 Command 창

(파) 메시지가 정상적으로 표시되지 않을 때는 다음과 같이 조치해본다.

1) USB 케이블을 분리한 후 다시 연결해본다.
2) 장치 관리를 통해 드라이버가 정상적으로 인식되는지 확인해본다.
3) fastboot 드라이버 설치가 완료되면 fastboot 서비스를 이용해 RoboEX-Brain을 복원하기 전까지는 사용할 필요가 없으므로 RoboEX-Brain을 재부팅한다. RoboEX-Brain의 재 부팅은 전원 스위치를 off 시켰다가 on 시킨다.

학습 1	로봇 액추에이터 제어 소프트웨어 사양 분석 및 구조 설계하기
학습 2	**로봇 액추에이터 제어기 펌웨어 설계하기**
학습 3	로봇 액추에이터 제어기 게인 설정하기
학습 4	로봇 액추에이터 제어기 제어 성능 시험하기

2-1. 로봇 액추에이터 제어기 펌웨어 설계

학습 목표

- 로봇 액추에이터 펌웨어 설계를 위한 요구 사항을 분석할 수 있다
- 로봇 액추에이터 제어기 펌웨어 코딩을 위한 펌웨어 구조를 파악할 수 있다.
- 로봇 액추에이터 제어기 펌웨어를 위한 함수들을 코딩할 수 있다.
- 로봇 액추에이터 제어 펌웨어 에뮬레이션 소프트웨어를 이용하여, 펌웨어 함수들이 적절하게 구현되었는지 평가할 수 있다.
- 로봇 제어 요구 사항을 만족하는지 펌웨어 성능을 분석할 수 있다.

필요 지식 /

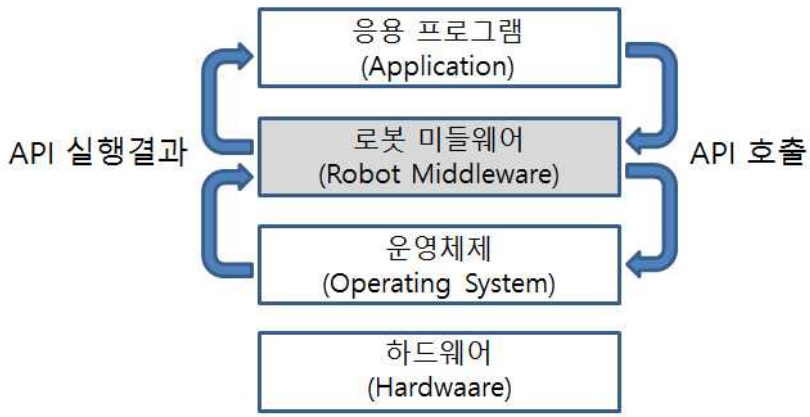
① 로봇 응용프로그램(robot application program)

응용프로그램은 운영 체제의 도움으로 실행되고 관리될 수 있는 프로그램이다. 응용프로그램은 하드디스크 등의 외부 기억장치에 저장되어 있다가 메모리로 이동하여 실행되면 프로세서라고 불린다. 응용프로그램은 운영 체제를 구성하는 한 프로그램인 로더(loader)라는 프로그램 또는 기능에 의해 주기억장치로 이동되며 이후 MPU에 의해 실행되고 운영체제에 의해서 관리된다.

<표 2-1> 저수준 언어와 고수준 언어의 비교

표-제목	저수준 언어	고수준 언어
언어	어셈블리어	C, C++, Pascal, Java
표현 형식	기계어의 1:1 매핑	인간의 자연에 가까운 형태
기계어 번역 방법	어셈블리	컴파일
장점	불필요한 연산이 없는 최적화에 유리함	저수준 언어대비 사용이 용이
단점	하드웨어 이해가 필요하고 프로그램 작성이 보다 어려움	컴파일에서 불필요한 코드가 추가 될 수 있음

일반적으로 응용프로그램은 상위 또는 하위 수준의 언어로 작성될 수 있는데 현재에는 프로그래밍의 생산성을 위해 상위 언어를 사용하는 실정이다.
로봇에서의 응용프로그램 또한 일반 컴퓨터와 같이 미리 구성된 로봇의 기구적 하드웨어와 제어를 위한 컴퓨터 하드웨어를 바탕으로 운영 체제의 API(application programming interface)를 기반으로 작성되고 운영 체제의 관리 하에서 실행된다. 복잡한 로봇의 경우 앞서 언급한 로봇 미들웨어가 통상 사용되는데 이때 응용프로그램을 로봇 미들웨어의 API를 사용한다.

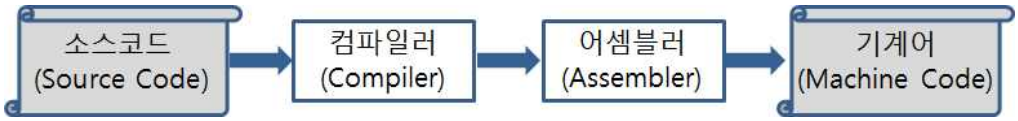


[그림 2-1] 로봇 응용프로그램의 동작 흐름

로봇 응용프로그램은 일반 컴퓨터에서와 마찬가지로 CPU가 실행할 수 있는 머신코드의 집합인데, 개념적으로 로봇의 특정 동작 또는 처리를 위해 미들에어에서 제공하는 API를 호출하면, 운영 체제의 API들이 호출되어 이의 결과를 받는 구조로 되어 있고, 일반 컴퓨터의 프로그래밍은 경우와 마찬가지로 아래의 컴파일 과정을 통하여 머신 코드를 생성할 수 있다.

1. 오브젝트 코드(Object Code)

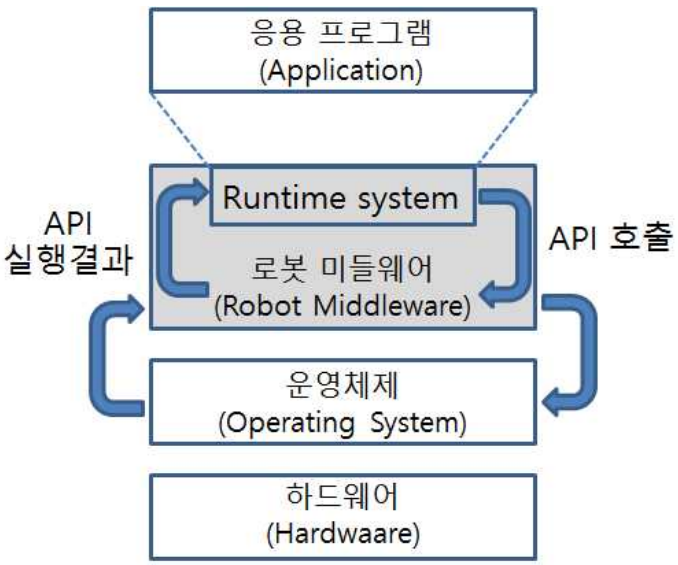
컴파일러와 어셈블러를 거쳐 생성된 머신코드의 집합을 오브젝트 코드(object code)라고하며 이들 코드가 저장된 파일을 오브젝트 파일이라고 한다. 오브젝트 파일은 링커라는 프로그램에 의해서 실제 바이너리 형태의 실행파일로 만들어지게 된다. 이와 같이 응용프로그램이 CPU의 명령어로 구성된 바이너리 형태인 경우 CPU와 운영 체제 그리고 미들웨어가 달라지면 응용프로그램이 제대로 실행되지 않는다. 이와 같은 구조의 응용프로그램은 하위 계층 구조에 의존적인데 응용프로그램이 실행되는 하위 계층의 변화가 있는 경우 응용프로그램은 소스 프로그램을 다시 컴파일 해야 하므로 소스 코드 레벨에서의 호환성을 지원한다고 생각할 수 있다.



[그림 2-2] 고수준 언어의 기계어 번역 과정

2. 런타임 시스템(runtime system)

하위 계층의 변화가 있더라도 응용프로그램의 재컴파일 없이 수행될 수 있는 환경이 있는데 이를 런타임 시스템(runtime system) 또는 가상머신(virtual machine)이라고 한다. PC 및 스마트 기기에서도 런타임 시스템이 폭넓게 사용되고 있는데 자바(JAVA)와 닷넷(.Net)이 대표적이며 인터넷 웹 환경에서 실행되는 프로그램들이 런타임 시스템만 올바르게 설치되어 있으면 운영 체제와 하드웨어의 종류에 관계없이 실행되게 하여 준다.



[그림 2-3] 런타임 시스템에서 응용프로그램의 실행

런타임 시스템에서 실행되는 응용프로그램은 CPU에서 실행되는 머신모드가 아닌 가상머신에서 실행되는 코드이다. 통상 CPU에 의존적인 코드를 바이너리라고 부르기 때문에 가상머신에서 실행되는 코드들은 바이트 코드라고 부른다.

런타임 시스템은 실제 로봇 시스템에서 폭넓게 적용 및 사용되고 있다. 그리고 하위 계층의 종류에 관계없이 응용프로그램의 동작 호환성을 보장하지만 런타임 시스템 또한 소프트웨어이므로 런타임 시스템 자체는 운영 체제 및 미들웨어 그리고 하드웨어에 맞게 제작되어야 한다.

3. 펌웨어(firmware)

(1) 개념

펌웨어는 소프트웨어의 한 종류이지만 특정기기를 위한 기능만을 위한 프로그램이라고 할 수 있다.

로봇의 응용프로그램은 로봇의 동작 및 작업을 정의하는 것으로, 로봇의 응용에 따라 사용자에 의하여 작성된 프로그램인데 로봇 미들웨어의 API를 이용하거나 런타임 시스템에서 지원하는 언어로 작성된다.

앞서 설명한 소프트웨어의 구조와 다르게 간단한 로봇 또는 기기는 응용프로그램의 교체 없이 항상 같은 프로그램으로 동작하는데 이때는 운영 체제를 사용하지 않는 경우가 많다. 이와 같은 소프트웨어는 별도로 펌웨어로 구분될 수 있는데 흔히 운영 체제와 어플리케이션의 구분 없이 간단하게 동작하는 형태를 의미하며 보다 간단한 구조의 MCU 등에서의 프로그램은 펌웨어라고 할 수 있다.

(2) 실행 구조

펌웨어는 항상 같은 프로그램만 실행되는 것이므로 복잡한 운영 체제가 필요가 없거나 아예 운영 체제를 사용하지 않는 경우도 많다. 펌웨어는 비교적 같은 일을 하는 기기에서 동작하며 보통 MCU에서 동작하고 Flash나 ROM에 저장되어 있다가 바로 실행되는 실행 구조를 가진다.

PC에서는 다양한 사용자에게 다양한 응용프로그램이 바뀌어가며 실행되거나 동시에 실행되지만, TV의 리모컨의 경우 항상 리모컨의 기능만 담당하는 프로그램이 실행된다. 다시 말해 하나의 하드웨어에서 하나의 응용프로그램만이 실행된다는 것이다. 이와 같이 응용프로그램의 변동 없이 계속 같은 프로그램이 실행되는 경우 해당 프로그램을 펌웨어라고 한다.

[그림 2-4] TV의 리모컨의 기능구현 펌웨어

2 임베디드 시스템 개요

임베디드 시스템이란 정해진 특정 기능을 수행하기 위해 하드웨어와 소프트웨어가 내장된 전자 제어 시스템을 말한다. 즉, 단순 회로만으로 구성된 장치가 아닌 마이크로프로세서가 내장되어 있고, 이러한 마이크로프로세서를 운용하여 원하는 작업을 수행 및 관리하는 프로그램이 포함된 시스템을 의미한다.

임베디드 시스템은 자동차, 우주, 항공, 군사, 의료장비와 공정 제어 등 산업용으로 시작하여, 이제는 가전제품, 스마트폰 등으로 영역을 확대하여 우리 생활에 밀접하게 관련되어 있다. 특히, 스마트폰이나 인터넷 접속이 가능한 스마트 TV, 휴대가 간편한 Tablet PC, 원격 모니터링 기능을 갖는 보일러 등이 이미 시장의 주류가 되어 있다. 스마트폰이라는 작은 크기의 종합적인 무선 통신 전화기와 같은 새로운 형태의 제품이 출현하여 불과 2-3년 만에 휴대전화의 새로운 패러다임으로 발전하고 있다. 이미 마이크로프로세서 및 소프트웨어가 탑재되지 않는 가전제품이 거의 없다는 점과 소프트웨어와 하드웨어 양쪽 측면에서 점점 많은 고기능을 요구하는 점을 감안하면 임베디드 시스템의 성장속도는 더 급격히 증가될 것으로 기대되고 있다.

3 임베디드 리눅스

1. 운영 체제는 컴퓨터 시스템 계층 구조에서 하드웨어와 가장 가까운 쪽을 차지하는 소프트웨어로서 시스템의 여러 자원(CPU, 메모리, 파일 시스템, 각종 입출력 장치, 네트워크)을 관리하는 역할을 한다. 운영 체제의 도움으로 응용프로그램들은 임베디드 시스템 내의 자원을 공유하여 사용할 수 있으며 자신에게 할당된 자원을 보호 받게 된다.
2. 현재의 운영 체제는 용도에 따라 크게 두 가지 분류로 나누어 볼 수 있는데, 하나는 리눅스나 마이크로소프트 윈도우와 같이 서버나 개인용 컴퓨터에서 실행되는 응용프로그

램들을 지원하는 것이고, 다른 하나는 실시간 운영 체제(RTOS)와 같이 우주, 항공 군사, 의료 장비 등과 같은 임베디드 시스템에서 엄격한 시간 제약성을 만족시켜야 하는 환경에 사용되기 위한 목적으로 만들어진 것이다.

3. 현재 가장 많이 사용되는 실시간 운영 체제로는 VxWorks를 비롯하여, pSOS, QNX, VRTX, nucleus, LynxOS, ChorusOS, QnX Neutrino, RTEMS, OS9, RealTime Linux 등이 있으며, 최근 스마트폰과 태블릿 열풍으로 인기 있는 운영 체제로는 APPLE의 iOS, GOOGLE의 android를 비롯하여 삼성의 Tizen 같은 것들이 있다.

4. 과거에는 많은 임베디드 시스템들이 운영 체제 없이 개발되었지만, 최근에는 대부분의 임베디드 시스템들이 어떤 형태든 운영 체제를 가지고 개발되고 있는 추세이다. 임베디드 시스템에서 운영 체제를 점점 더 많이 도입하는 중요한 이유는 좀 더 좋은 성능을 가진 복잡한 하드웨어가 사용되고 예전과는 비교할 수 없는 많은 소프트웨어 기능이 요구되기 때문이다.

4 시스템 소프트웨어

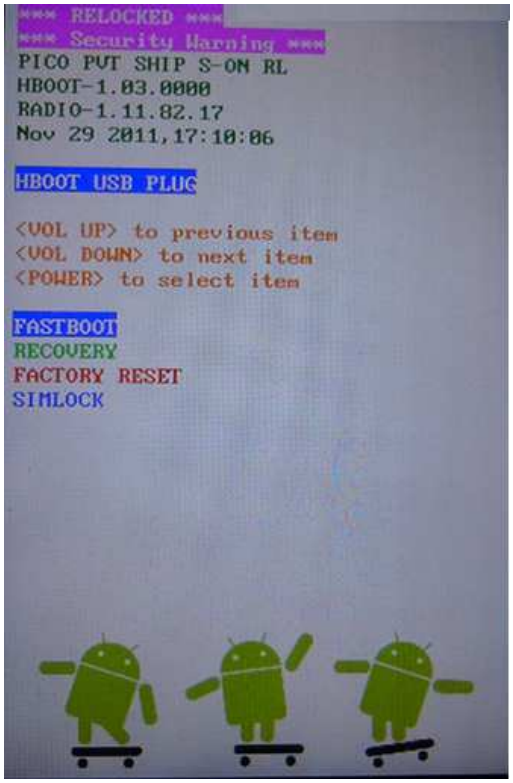
1. 부트로더

펌웨어에서는 프로그램이 바뀌지 않더라도 앞서 설명한 로더를 가질 수 있는데 이때는 기기의 기능 보완을 위한 프로그램의 업그레이드 용도나 프로그램이 실행을 시작할 주기억장치의 위치를 알려주는 등의 제한적 기능만이 사용된다. 이런 제한적 기능의 로더를 부트로더(boot loader)라고 한다.

임베디드 시스템에도 펌웨어를 가지고 있는데 PC가 처음 시작할 때 BIOS라는 프로그램이 항상 실행되며 이는 ROM에 저장되어 있다가 바로 실행된다. BIOS는 운영 체제가 실행되기 전에 최소한의 하드웨어 설정을 담당한다. 즉, PC도 펌웨어가 동작하고 이후 운영 체제가 실행된다.

부트로더는 타깃 장비를 초기화하고 임베디드 운영 체제로 부팅 할 수 있게 처음에 동작하는 프로그램이다. 또 임베디드 장비에 처음 전원이 인가되었을 때 동작하는 작은 프로그램이기도 하다. 부트로더는 운영 체제가 동작할 수 있도록 하드웨어를 초기화하고, 운영 체제 커널을 메모리로 적재시키는 역할을 수행한다.

오래된 시스템들에서 사용되었던 프로세서는 부팅 시작 시에 특정 번지에서 실행 코드를 읽어서 수행하도록 구성되어 있었다. 그래서 보통 시작 번지에는 명령어가 있는 실제 주소로 jump하는 코드가 들어 있었다. 그러나 최신 프로세서들은 프로세서 안에 적은 용량의 ROM과 RAM을 포함하고 있어서 초기 동작하는 프로그램을 프로세서 안에 포함하고 있다.



[그림 2-5] 임베디드 시스템의 부트로더 화면(안드로이드 폰)

2. 리눅스 커널

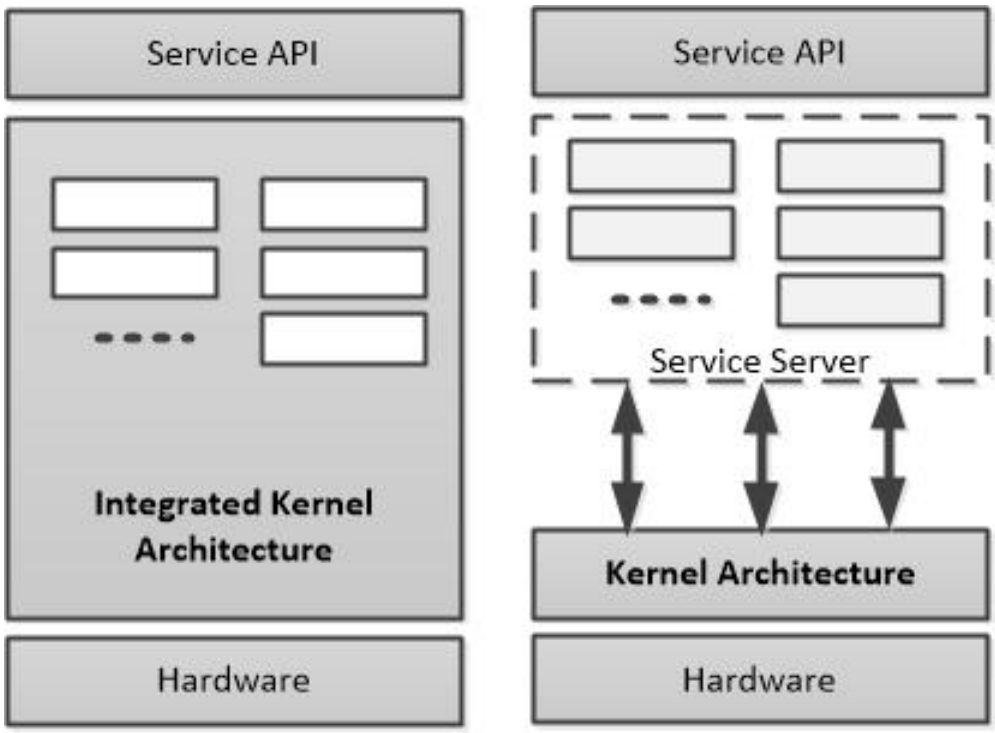
(1) 커널의 개요

리눅스 커널은 리눅스 운영 체제를 이루는 가장 핵심적인 소프트웨어이다. 리눅스 커널이 곧 리눅스 운영 체제라고도 할 수 있다. 하지만 커널이 리눅스 전체라고는 볼 수 없다. 커널은 "kernel" 이란 단어 뜻에서도 유추할 수 있듯이 운영 체제에서 가장 중요한 부분을 담당하고 있기 때문에 커널=리눅스란 표현을 많이 사용되는 것이다.

지금부터 커널에서 수행하는 역할에 관하여 살펴보겠다.

커널은 시스템의 구동에 필요한 환경 설정과 수행되는 프로그램들을 스케줄링 하는 소프트웨어이다. 커널은 크게 마이크로 커널(micro kernel)과 모놀리틱 커널(monolithic)로 나눌 수 있으며, 마이크로 커널은 커널이 가져야 하는 핵심적인 기능만을 구현한 최소 커널로써 나머지는 서비스 프로세스로 이루어진다. 모놀리틱 커널이란 커널 내부에 시스템 운영에 필요한 많은 서비스 루틴들을 포함한 구조를 가지고 있다. 이러한 모놀리틱 커널은 구현이 간단하며 시스템 자원을 보다 효율적으로 관리할 수 있는 장점을 가지고 있지만 다양한 환경의 시스템에 포팅하기 어렵고, 커널의 크기가 상대적으로 커진다는 단점을 가진다. 하지만 마이크로 커널은 핵심 기능과 작은 서버 모듈로 나누어 설계되고, 최소 기능만을 커널 내부에 포함하므로 기능 확장 및 다른 시스템에 재사용이 쉬워진다. 반면, 서버에 메시지 전달 방식의 접근은 태스크 스위칭에 많은

오버헤드를 초래하며 자원의 효율적 이용도 낮아진다. Solaris, AIX, HP-UX, Linux등이 모놀리틱 커널을 사용하며 Mach OS, sunsoft, Digital UNIX등은 마이크로 커널을 사용하고 있다.



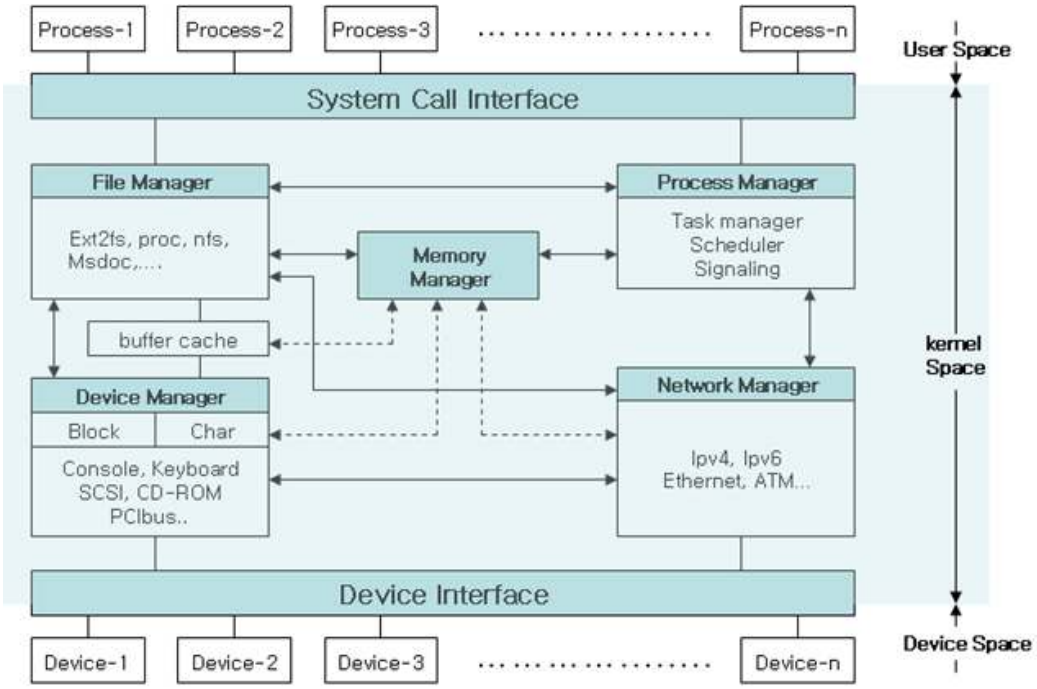
[그림 2-6] 커널 종류

(2) 리눅스 커널의 기능

다음 그림과 같이 커널은 프로세스 관리, 메모리 관리, 파일 시스템 관리, 디바이스 관리, 네트워크 관리의 5개의 기능을 블록으로 구분할 수 있다.

(가) 프로세스 관리

리눅스 운영 체제에서는 시스템이 동작 이후에 최소한 하나 이상의 프로세스가 동작한다. 이 프로세스는 다른 말로 태스크라고도 하며, 주어진 일을 수행하는 기본 단위다. 커널은 프로세스 스케줄러를 이용하여 여러 프로세스가 동작할 수 있도록 각 프로세스를 생성하고 제거하며, 외부 환경과 프로세스를 연결하고 관리한다.



출처: MCLAB(2016). mclab wikipedia. http://mclab.hufs.ac.kr/에서 2016. 9. 30. 검색.

[그림 2-7] 리눅스 커널 구조

(나) 메모리 관리

시스템에서의 메모리는 프로세서와 마찬가지로 가장 핵심적이고, 중요하게 관리해야 하는 지원이다. 메모리를 관리하는 정책은 시스템 성능을 결정하는 중요한 요소로, 각각의 프로세스가 독립적인 공간에서 수행할 수 있도록 가상 주소 공간을 제공한다. 이 가상 메모리 관리를 바탕으로 보조 기억 장치와 연동하여 물리적인 한계를 극복할 수 있는 기능을 제공한다.

(다) 파일 시스템 관리

리눅스 커널은 유닉스 시스템에서 사용하는 파일 시스템을 근간으로 설계되었다. 그래서 리눅스 커널에서 동작하는 응용프로그램은 시스템에 동작하는 모든 자원을 파일처럼 다룰 수 있도록 통일된 인터페이스를 제공한다. 또 리눅스 커널은 가상 파일 시스템(VFS)를 이용하여 현존하는 대부분의 파일 시스템 형식을 지원한다.

(라) 디바이스 제어

운영 체제에서 동작하는 응용프로그램의 결과는 하드웨어와 반드시 연결되어야 사용자가 인지할 수 있다. 메모리와 프로세서를 제외한다면 그 외의 동작은 반드시 하드웨어적인 처리를 수반한다. 하드웨어에 관련된 처리는 디바이스 드라이버에서 담당하며, 커널이 반드시 구현해야 하는 것 중 하나다. 리눅스 커널은 파일 시스템의 구조에 디바이스 드라이버를 연동하여 구현하며, 표준화된 형식으로 하드디스크로부터 키보드, 이더넷과 같은 모든 주변 장치를 관리한다.

(마) 네트워크 관리

현대의 시스템은 반드시 네트워크 처리를 수반한다. 현존하는 운영 체제 중에는 네트워크 처리를 수행하지 않는 운영 체제도 있겠지만, 리눅스 커널은 네트워크를 필요로 하는 시스템에서 주로 개발되었기 때문에 가장 우수한 네트워크 관리 시스템을 갖추고 있다. 리눅스 커널은 네트워크 스택을 이용하여 응용프로그램과 네트워크 디바이스 드라이버를 연결하며, 매우 효율적인 네트워크 처리를 구현하고 있다. 또 리눅스 커널에서 동작하는 네트워크 시스템은 암호화와 보안 특성이 연계된 매우 견고한 시스템을 구성한다.

5 안드로이드(android)

안드로이드는 OS, middleware, key application을 포함하는 모바일기기의 S/W 집합체이다. 그리고 안드로이드 SDK는 안드로이드 플랫폼 상에서 자바 프로그래밍 언어를 이용하여 개발을 시작할 수 있도록 툴과 API를 제공하고 있다.

1. Android 특징

(1) application framework: 컴포넌트의 재사용 및 대체를 가능하게 함.

(2) dalvik virtual machine: 모바일 디바이스를 위해 최적화 됨.

(3) integrated browser: open source webkit 엔진 기반

(4) optimized graphics: 구글이 만든 2D 그래픽 라이브러리에 의해 강화 됨. open GL ES 1.0/2.0스펙을 기반으로 하는 3D 그래픽(하드웨어 가속은 선택사항)

(5) SQLite: 정형화된 데이터 저장 공간을 위한 것.

(6) 미디어 지원: 일반적인 오디오, 비디오, 그리고 정지 이미지 포맷들을 지원(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

(7) GSM 지원(하드웨어 의존적)

(8) Bluetooth, EDGE, 3G, WiFi 지원(하드웨어 의존적)

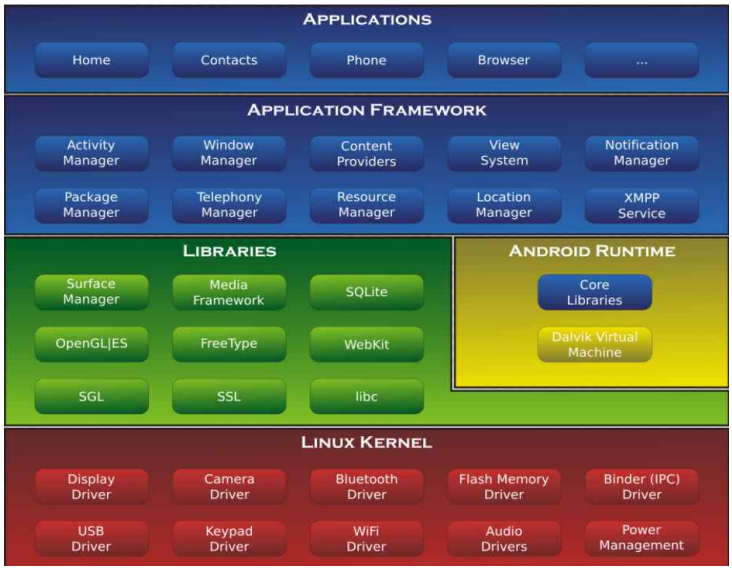
(9) 카메라, GPS, 나침반과 가속도계 지원(하드웨어 의존적)

(10) 풍부한 개발환경: 디바이스 에뮬레이터, 디버깅 도구, 메모리 및 성능 프로파일링, 그리고 eclipse IDE를 위한 플러그인을 포함함.

2. android 구조

(1) applications

안드로이드는 이메일 클라이언트, SMS 프로그램, 달력, 지도, 브라우저, 전화번호부, 그리고 다룬 것들을 포함하는 핵심 응용프로그램을 탑재할 것이다. 모든 프로그램은 자바 프로그래밍 언어로 작성된다.



출처: Wikipedia(2016). https://en.wikipedia.org/에서 2016. 9. 30. 검색.
[그림 2-8] 안드로이드 구조

(2) application framework

개발자는 핵심 응용프로그램에서 사용된 것과 같은 프레임워크 API에 모두 접근 가능하다. 응용프로그램 아키텍쳐는 컴포넌트 재사용을 손쉽게 할 수 있도록 디자인 되었으며, 어떤 응용프로그램의 기능으로 제작하는 데 사용된다(단, 프레임워크의 보안 제약을 따라야 한다). 이 같은 메카니즘은 사용자에 의한 컴포넌트의 대체를 가능하게 한다. 아래에 나열된 모든 어플리케이션은 하나의 서비스와 시스템의 집합체이다.

(가) 풍부하고 확장성 있는 뷰 집합 : 리스트, grid, 텍스트 박스, 버튼, 심지어는 임베디드 가능한 웹 브라우저를 포함하는 어플리케이션을 제작하는 데 사용될 수 있음

(나) 컨텐트 제공자 : 어플리케이션이 다른 어플리케이션(전화번호부 같은)의 데이타에 접근하는 것이나 자신의 데이타를 공유시키는 것을 가능하게 함

(다) 리소스 관리자 : 문자열, 그래픽, 레이아웃 파일과 같은 코드화되지 않는 자원에 대한 접근을 제공함

(라) 알림 관리자 : 모든 어플리케이션이 상태바에 어플리케이션이 만든 알림메시지를 표시하는 것을 가능케 함.

(마) 액티비티 관리자 : 어플리케이션의 생명 주기를 관리하며, 일반적인 네비게이션 히스토리를 제공함

(3) libraries

안드로이드는 안드로이드 시스템에서 다양하게 사용되는 C/C++ 라이브러리들을 포함하며, 안드로이드 application framework를 통해 개발자들은 이런 사항을 알 수 있다.

(가) System C library : 임베디드 리눅스 기반의 디바이스를 위해서 튜닝된 표준 C 시스템 라이브러리의 BSD 상속 구현체(libc)

(나) Media Libraries : PacketVideo의 OpenCORE기반이며, 인기 있는 오디오 및 비디오 포맷, MPEG4, H.264, MP3, AAC, AMR, JPG, PNG를 포함하는 정적 이미지 파일의 재생 및 녹화를 지원함.

(다) Surface Manager : 디스플레이 서브 시스템 및 다수의 응용프로그램의 2D, 3D 그래픽 레이어

(라) LibWebCore : 안드로이드 브라우저와 임베딩 가능한 웹 뷰와 같은 최신 웹 브라우저 엔진

(마) SGL : 2D 그래픽 지원

(바) 3D libraries : OpenGL ES 1.0/2.0 API들을 기반으로 하며, 하드웨어 3D 가속 또는 최적화된 3D S/W 래스터라이저를 사용함.

(사) FreeType : 비트맵 또는 벡터 폰트 렌더링

(아) SQLite : 모든 응용프로그램에서 사용 가능한, 강력하며 경량화 된 관계형 데이터베이스 엔진

(4) android runtime

안드로이드는 Java 프로그래밍 언어의 핵심 라이브러리에서 사용 가능한 대부분의 기능들을 포함하는 핵심 라이브러리들을 포함한다.

모든 안드로이드 응용프로그램은 Dalvik 가상 머신내의 자신의 인스턴스를 가지고, 자신의 프로세스 내에서 동작한다. Dalvik은 기기가 다수의 버추얼 머신에서 효율적으로 실행될 수 있도록 제작되었으며, 최소의 메모리 영역에 최적화된 Dalvik Executable(.dex) 포맷 파일을 실행시킨다. 버추얼 머신은 레지스터 기반이며, 자바 컴파일러로 컴파일된 클래스들을 "dk"툴을 이용하여 .dex 포맷으로 변경한 클래스들을 실행한다.

Dalvik 버추얼 머신은 스레딩과 저수준 메모리 관리와 같은 리눅스 커널 기능을 사용한다.

(5) linux kernel

안드로이드는 보안, 메모리 관리, 프로세스 관리, 네트워크 스택, 드라이버 모델과 같은 리눅스 버전 2.6의 핵심 시스템 서비스를 이용하며, 커널은 하드웨어와 소프트웨어 간 추상 계층으로 동작한다.

수행 내용 / 로봇 액추에이터 제어기 펌웨어 설계하기

재료 · 자료

- 액추에이터 데이터 시트
- 액추에이터 구동 회로도
- 기구 동작 제원서
- 로봇에 대한 사용자 요구 사양서

기기(장비 · 공구)

- 컴퓨터, 프린터
- 프로그램 개발용 소프트웨어
- 문서 작성용 소프트웨어
- 프로그램 성능 평가용 에뮬레이션 소프트웨어
- 로봇 액추에이터 제어 성능 시험 소프트웨어

안전 · 유의 사항

- 액추에이터는 대동력 기기로서 높은 전압과 큰 전류를 소비를 하므로 취급 시 특히 전기 안전에 유의해야 한다.
- 액추에이터의 정 · 역회전 제어와 같이 상반된 동작의 경우에는 오조작이나 제어기 고장에 대비해 전기적 인터록은 물론 상황에 따라 기계적 인터록도 고려해야 한다.
- 액추에이터의 선정 계산 방법은 해당 제품의 카탈로그 및 규격화된 계산 공식을 따른다.
- 액추에이터의 설치 방법은 해당 제품의 사용 설명서와 제조사에서 규정한 내용을 포함하며, 회사 내 해당 작업의 규정에 따른다.

수행 순서

1 UART 통신 프로그램을 작성한다.

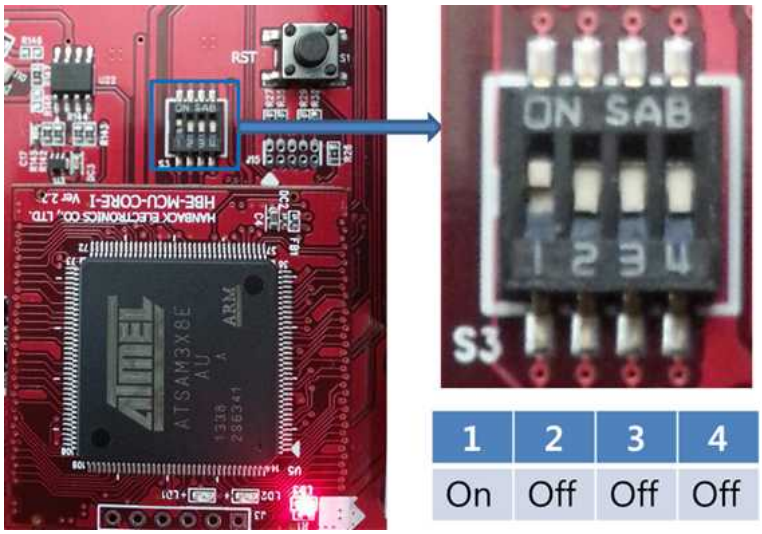
액추에이터를 제어하기 위해서는 embedded 모듈과 MCU-CORE가 uart을 하며 미리 정해진 패킷으로 통신을 해야 한다. 그래서 이번 장에서는 uart통신의 기본 개념 및 간단한 통신 프로그램을 작성해 보도록 한다.

1. 프로그램 구성

프로그램 작성에 앞서 장비 설정 및 프로젝트 생성을 먼저 진행한다.

(1) 장비 설정

embedded모듈에서 MCU-CORE로 UART를 통해 데이터를 보내면 그 데이터 값을 그대로 다시 embedded 모듈로 전송해 주도록 되어 있다. 먼저 RoboEX-Brain에 좌측에 MCU-CORE 위에 DIP스위치 중에 1번 스위치를 on 시킨다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.96.
[그림 2-9] uart test를 위한 DIP SW 변경

DIP스위치 변경 후 스위치 우측에 있는 RST버튼(reset)을 누르면 TextLCD의 문자가 변경되고 설정이 완료된다.

출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.96.
[그림 2-10] uart test시 TextLCD 화면

(2) 프로젝트 생성

uart 통신을 위한 프로그램 작성을 위해 다음과 같이 프로젝트를 생성한다.

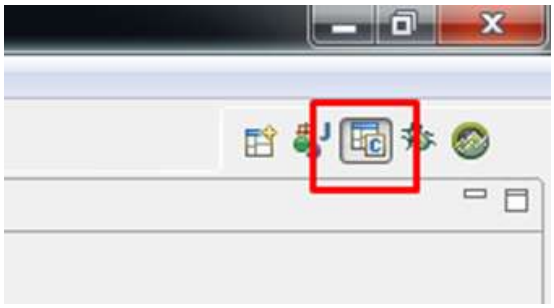
<표 2-2> 프로젝트 생성

구 분	내 용
Project name	roboex_brain_uart
Toolchains	Other Toolchain
External Tools	C Executable
File Name	roboex_brain_uart.c

프로젝트 생성 과정은 다음과 같다. 우선 AndroX Studio를 실행하고, 아래와 같이 버튼을 클릭하여 "IDE" (이클립스)를 실행한다.

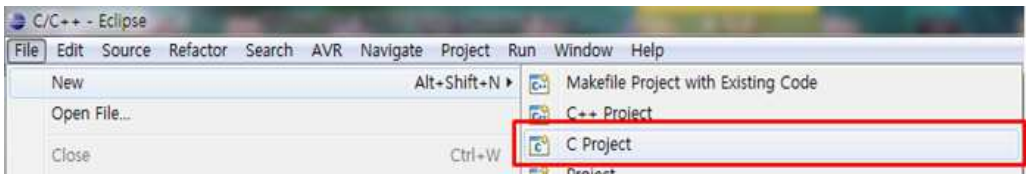
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.97.
[그림 2-11] IDE 실행

우측 상단의 아이콘 중 "C/C++ Perspective" 를 선택한다.



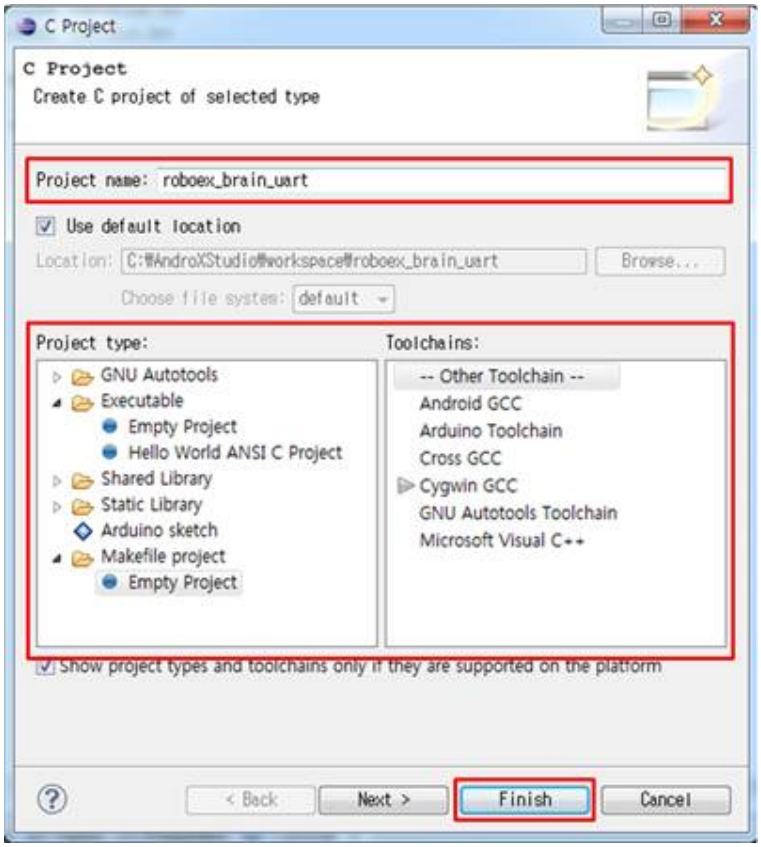
[그림 2-12] C/C++ Perspective

"File > New > C Project" 를 선택한다.



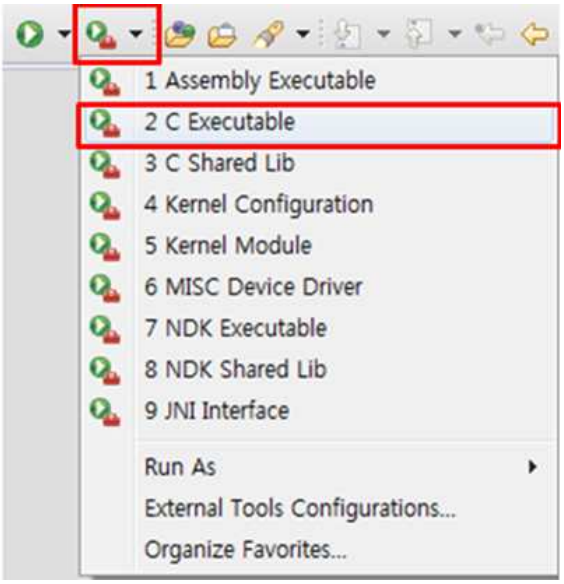
[그림 2-13] C Project 생성 (1)

"Makefile > Empty Project > Other Toolchain" 을 선택한다. 프로젝트 이름은 [그림 2-14]와 같이 "roboex_uart" 를 입력한다. 마지막으로 "finish" 버튼을 클릭한다.



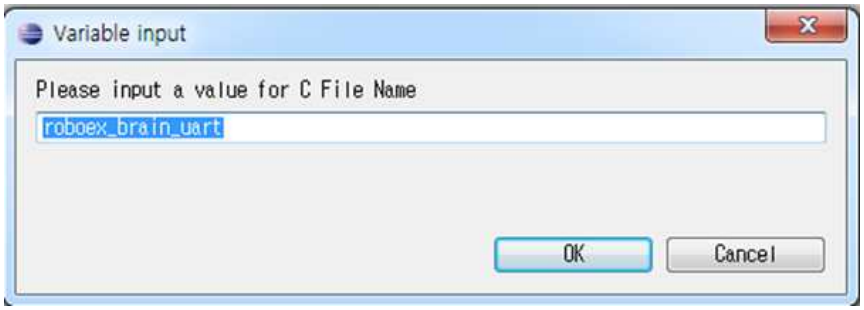
[그림 2-14] C Project 생성 (2)

"External Tools > C Executable" 를 선택한다.



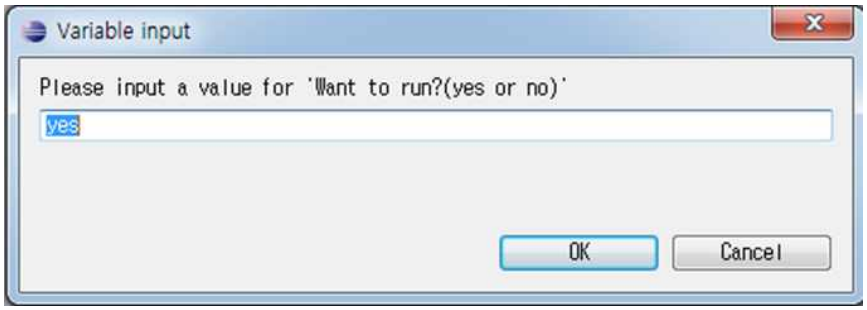
[그림 2-15] 코드 생성

프로젝트 이름과 동일하게 프로그램 파일 이름을 입력하고 "OK" 버튼을 누른다.



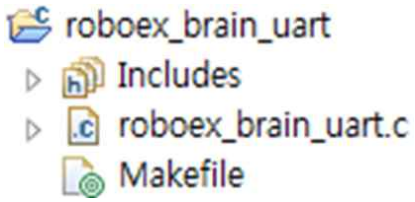
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.98.
[그림 2-16] 프로그램 파일 이름 입력

"OK" 버튼을 누른다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.99.
[그림 2-17] 자동 실행 확인

아래 그림과 같이 프로젝트 내에 2개의 파일이 생성된 것을 확인할 수 있다.



[그림 2-18] 생성된 파일 리스트

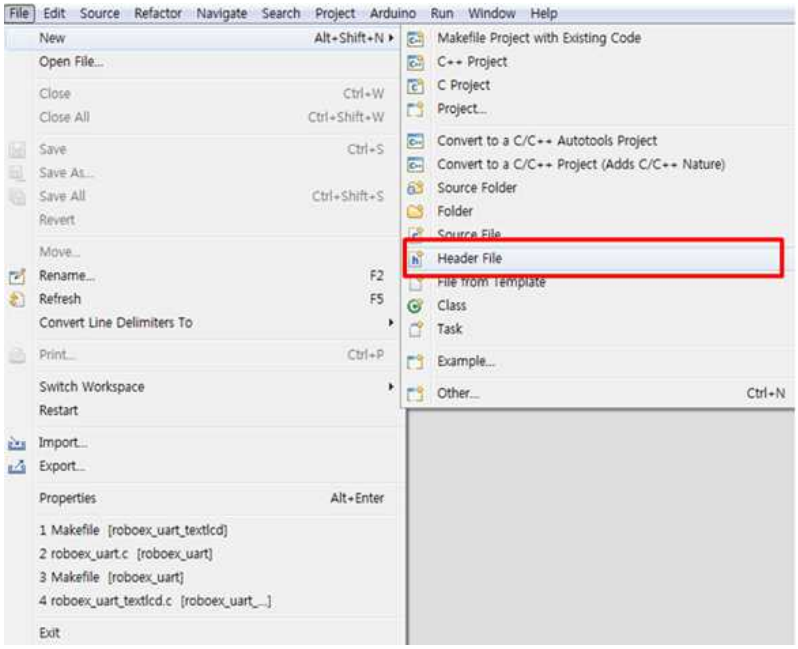
(3) 파일 추가

프로젝트 내에 다음 파일을 추가한다.

<표 2-3> 파일 추가

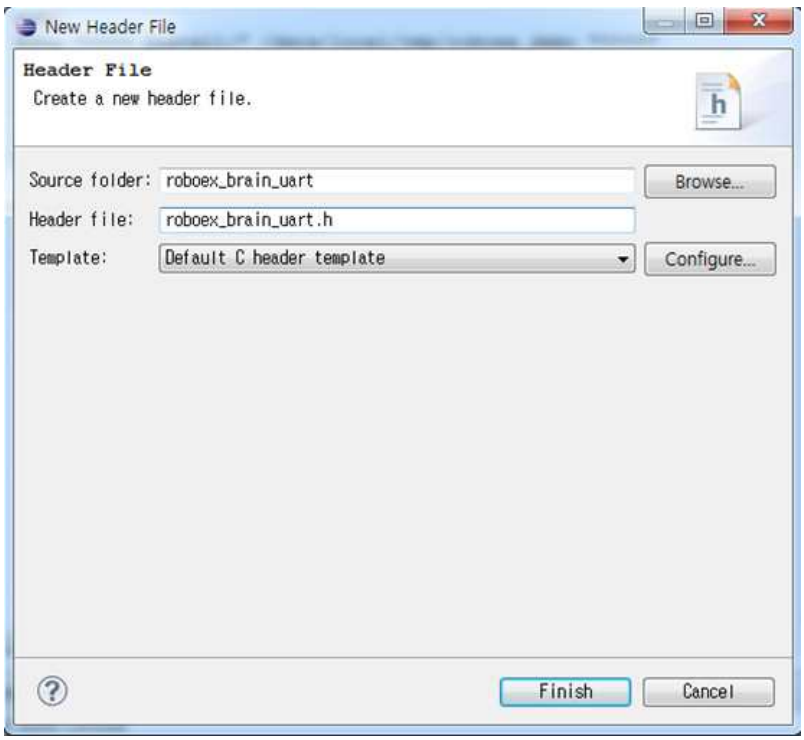
프로젝트 명	파일명
roboex_brain_uart	roboex_brain_uart.h

파일을 추가하기 위해 "File > New > Header File" 을 선택한다.



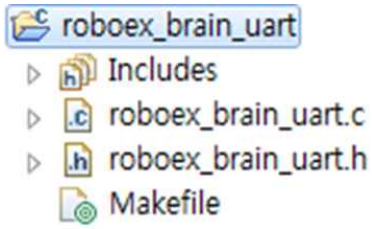
출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.99.
[그림 2-19] 헤더 파일 추가(1)

[그림 2-20]과 같이 Header file 이름을 "roboex_uart.h"로 입력하고, "Finish" 버튼을 클릭한다.



출처: 이상문(2014). 『브레인 모듈로 배우는 RoboEX 시리즈 응용 프로그램 입문』 한백전자. p.100.
[그림 2-20] 헤드 파일 추가(2)

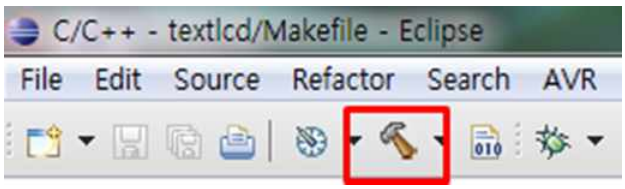
project explorer를 보면 다음과 같이 파일이 추가된 것을 확인할 수 있다.



[그림 2-21] 파일 추가 결과

(4) 프로젝트 실행

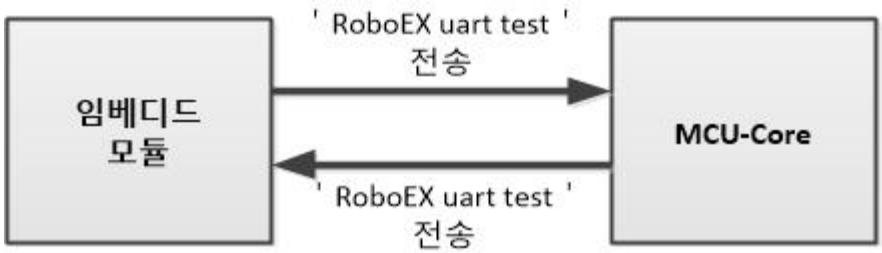
프로젝트를 생성하고 앞 절을 참조하여 소스 코드를 작성한다. 모든 과정을 완료했다면 프로그램을 컴파일을 하고 타깃 장비 내에 등록하도록 한다. 컴파일 및 등록 과정에 필요한 모든 코드는 프로젝트 내 Makefile에 저장되어 있다. 아래의 버튼을 눌러 프로젝트를 실행하면 Makefile에 저장된 코드가 실행된다.



[그림 2-22] 프로젝트 실행

2. 소스 코드 및 구현 설명

MCU-CORE는 Embedded 모듈로 받은 데이터를 그대로 보내준다.



[그림 2-23] 데이터 전송

(1) roboex_brain_uart.c

프로그램을 실행하면 동작하는 파일이다.

```
001: #include <unistd.h>
002: #include <stdlib.h>
003: #include <stdio.h>
004: #include "roboex_uart.h"
005:
006:
007: int main(int argc, char * argv[]) {
008:
```

```
009:   unsigned char   send_buf[20]="RoboEX uart test";
010:   unsigned char read_buf[20];
011:   int i,data;
012:
013:   init_uart();
014:
015:   write( uart, send_buf, 20   );
016:   sleep(1);
017:
018:   data = read( uart,   read_buf, 20 );
019:   if ( data > 0 )
020:   {
021:     printf("return data   = %s\n",read_buf);
022:   }else{
023:     printf("READ   ERROR!!!\n");
024:   }
025:
026:   close_uart();
027:   exit(0);
028: }
```

(2) roboex_brain_uart.h

Uart관련 함수를 정의한 파일이다.

```
001: #ifndef ROBOEX_BRAIN_UART_H_
002: #define ROBOEX_BRAIN_UART_H_
003:
004: #include <termios.h>
005: #include <fcntl.h>
006:
007: int uart;
008: struct termios oldtio,newtio;
009:
010: void init_uart(void)
011: {
012:   uart =   open("/dev/ttySAC0",O_RDWR | O_NOCTTY);
013:   if(uart < 0)
014:   {
015:       perror("/dev/ttySAC0");
016:     exit(1);
017:   }
018:     tcgetattr(uart,&oldtio);
019:     bzero(&newtio,sizeof(newtio));
020:   newtio.c_cflag = B115200 | CS8 | CLOCAL |   CREAD;
021:   newtio.c_iflag = IGNPAR |   ICRNL;
022:   newtio.c_oflag = 0;
023:   newtio.c_lflag = ~(ICANON |   ECHO | ECHOE | ISIG);
024:
025:   tcflush(uart,TCIFLUSH);
026:     tcsetattr(uart,TCSANOW,&newtio);
027: }
028:
029: void close_uart(void)
030: {
031:     tcsetattr(uart,TCSANOW,&oldtio);
032:   close(uart);
033: }
034:
035: #endif /* ROBOEX_BRAIN_UART_H_ */
```

2 servo motor 제어를 수행한다.

이번 장에서는 앞에서 배운 uart통신을 응용하여 servo motor를 제어 해본다. HBE-RoboEX-Brain에는 2개의 Servo 모터가 있다. 하단에 아래쪽에 연결된 모터는 좌우 방향 전환을 담당하고 위쪽에 연결된 모터는 위 애래 방향의 전환을 담당한다. 먼저 servo motor 구조에 대해 알아보고 servo motor를 동작시키기 위한 uart 패킷을 학습하고 프로그램을 작성해 실행 시켜보도록 한다.

1. 동작 원리

servo 모터의 제어 요청 패킷의 구조는 다음과 같다.

<표 2-4> servo모터 제어 요청 패킷 구조 (brain모듈 → MCU모듈)

1	2	3	4	5	6	7	8	9
Start	0x00	ID1	ID2	DATA1	DATA2	DATA3	DATA4	CSC
0x76	0x00	0x20	Servo ID	Angle_ H	Angle_ L	Speed _H	Speed _L	CSC

모터 제어 요청 패킷의 시작은 0x76이며 ID1의 값은 0x20이다.

<표 2-5> servo ID

	Servo ID	Angle	Speed
하단 모터	0	60 ~ 240	0 ~ 1023
상단 모터	1	80 ~ 220	0 ~ 1023

servo ID는 상단과 하단의 2개의 모터를 구별하기 위한 ID값이고 angle은 모터의 위치를 나타내며 speed는 모터의 위치가 변경될 때 동작 속도를 나타낸다.

제어 패킷에서 angle과 speed의 값이 H와 L로 나눠지는데 데이터 값을 1바이트로는 0부터 255까지 밖에 표현 할 수가 없어서 2바이트를 이용하여 데이터 값을 표현한다. 예를 들어 1023의 경우 16진수로 변환하면 0x03FF가 되고 이 값을 표현하면 다음과 같다.

<표 2-6> 1023 데이터 값의 표현

스피드	비 트							
Speed_H	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	1	1
Speed_L	7	6	5	4	3	2	1	0
	1	1	1	1	1	1	1	1

servo 모터제어 응답 패킷은 앞에서 학습한 LED와 같이 ID1에 1이 더해져서 응답 패킷을 확인한다.

2. 프로그램 구성

장비 설정 및 프로젝트 생성 및 파일 추가는 5-2-2의 내용을 참고한다.

(1) 프로젝트 생성

servo 모터 제어를 위한 프로그램 작성을 위해 다음과 같이 프로젝트를 생성한다.

<표 2-7> 프로젝트 생성

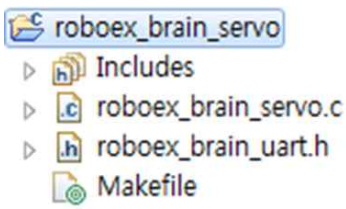
구 분	내 용
project name	roboex_brain_servo
toolchains	other toolchain
external tools	C executable
file name	roboex_brain_servo.c

(2) 파일 추가

<표 2-8> 파일 추가

프로젝트명	파일명
roboex_brain_servo	roboex_brain_uart.h

프로젝트가 생성이 완료 되면 다음과 같은 구성이 된다.



[그림 2-24] 프로젝트 트리

(3) 프로젝트 실행

프로젝트를 생성하고 [그림 2-13]의 내용을 참조하여 소스 코드를 작성하여 실행한다. 실행 과정은 1 UART 통신을 참조한다.

3. 소스 코드

(1) roboex_brain_servo.c

프로그램을 실행시키면 동작하는 파일이다.

```
001: #include   <stdio.h>
002: #include   <stdlib.h>
003: #include   "roboex_brain_uart.h"
004:
005: unsigned char   servo_left[9] = {0x76, 0x0, 0x20, 0x0, 0x0, 0xB0, 0x0, 0x30,
                                    0x00};
006: unsigned char    servo_right[9] = {0x76, 0x0, 0x20, 0x0, 0x0, 0x7c, 0x0,
                                     0x30, 0xCC};
007:
008: int main(void)
009: {
010:   init_uart();
011:
012:   write( uart, servo_left, 9 );
013:   sleep(3);
014:   write( uart, servo_right, 9 );
015:   sleep(1);
016:
017:   close_uart();
018:   return 0;
019: }
```

(2) 실행 결과

프로그램을 실행시키면 servo 모터가 좌우로 이동하는 것을 확인한다.

시작은 0x76이고 ID1은 0x20이고 DATA부분의 속도는 48이고 좌우 각도는 176도와 124도이다.

학습 2 교수 · 학습 방법

교수 방법

- 로봇 액추에이터 제어기 펌웨어 코딩을 위한 펌웨어 구조를 설명한다.
- 로봇 액추에이터 제어기 펌웨어를 위한 함수 코딩 방법을 설명한다.
- 제어 펌웨어 기능에 대하여 작성한 후 모듈별로 사용된 제어기기에 대하여 설명한다.
- 제어 펌웨어 모듈과 제어 시스템과의 연관성에 대하여 비교 설명한다.
- 제어 프로그램의 특성에 적합하면서, 실제 작동과 유사한 환경의 시뮬레이션을 해볼 수 있도록 지도한다.
- 실제 시스템과 I/O 파라미터 사이의 연관성을 파악하고, 파라미터 결정 방법을 설명한다.
- 제어 프로그램과 제어 인터페이스 매뉴얼에 의해 제어 대상물과 소프트웨어 컴퓨터와의 인터페이스 방법을 설명한다.
- 제어 사양서를 바탕으로 하여 제어 구현 방법(PC, PLC, 펌웨어, 릴레이 등)을 설명한다.

학습 방법

- 로봇 액추에이터 펌웨어 설계를 위한 요구 사항을 숙지한다.
- 로봇 액추에이터 제어기 펌웨어 코딩을 위한 펌웨어 구조를 이해한다.
- 로봇 액추에이터 제어기 펌웨어를 위한 함수 코딩 방법을 이해한다.
- 로봇 액추에이터 제어 펌웨어 에뮬레이션 소프트웨어를 이용하여, 펌웨어 함수들이 적절하게 구현하는 방법을 숙지하다.
- 제어 사양서 및 플로 차트를 참고하여, 동작을 구현할 수 있는 모듈 구성을 이해한다.
- 제어 대상물의 특성에 적합하고, 제어 사양서의 동작조건에 충족하는 프로그램 작성 방법을 습득한다.
- 제어 시뮬레이션 소프트웨어 관련 지식에 있는지 파악한 후 부족한 부분은 사용 매뉴얼을 참조하여 보완한다.

학습 2 평 가

평가 준거

- 평가자는 학습자가 학습 목표를 성공적으로 달성하였는지를 평가해야 한다.
- 평가자는 다음 사항을 평가해야 한다.

학습 내용	평가 항목	성취수준		
		상	중	하
로봇 액추에이터 제어기 펌웨어 설계	- 로봇 액추에이터 펌웨어 설계를 위한 요구 사항을 분석할 수 있다.			
	- 로봇 액추에이터 제어기 펌웨어 코딩을 위한 펌웨어 구조를 파악할 수 있다.			
	- 로봇 액추에이터 제어기 펌웨어를 위한 함수들을 코딩할 수 있다.			
	- 로봇 액추에이터 제어 펌웨어 에뮬레이션 소프트웨어를 이용하여, 펌웨어 함수들이 적절하게 구현되었는지 평가할 수 있다.			
	- 로봇 제어 요구 사항을 만족하는지 펌웨어 성능을 분석할 수 있다.			

평가 방법

- 문제 해결 시나리오

학습 내용	평가 항목	성취수준		
		상	중	하
로봇 액추에이터 제어기 펌웨어 설계	- 로봇 액추에이터 제어기 펌웨어 코딩을 위한, 펌웨어 구조 대한 이해			
	- 로봇 액추에이터 제어기 펌웨어를 위한, 함수들의 코딩에 대한 이해			
	- 펌웨어 함수들의 적절한 구현 능력 숙지			
	- 로봇 액추에이터 작업을 위한, 펌웨어 성능 분석에 대한 이해			

• 평가자 질문

학습 내용	평가 항목	성취수준		
		상	중	하
로봇 액추에이터 제어기 펌웨어 설계	- 로봇 액추에이터 제어기 펌웨어 코딩을 위한, 펌웨어 구조에 대한 지식			
	- 로봇 액추에이터 제어기 펌웨어를 위한, 함수들의 코딩에 대한 지식			
	- 로봇 액추에이터 제어 펌웨어 에뮬레이션 소프트웨어를 이용한, 펌웨어 함수 구현 방법에 대한 지식			
	- 로봇 액추에이터 작업을 위한, 펌웨어 성능 분석에 대한 지식			

• 구두 발표

학습 내용	평가 항목	성취수준		
		상	중	하
로봇 액추에이터 제어기 펌웨어 설계	- 로봇 액추에이터 제어기 펌웨어 코딩을 위한, 펌웨어 구조에 대한 이해 정도			
	- 로봇 액추에이터 제어기 펌웨어를 위한, 함수들의 코딩에 대한 이해 정도			
	- 로봇 액추에이터 제어 펌웨어 에뮬레이션 소프트웨어를 이용한, 펌웨어 함수 구현 방법에 대한 이해 정도			
	- 로봇 액추에이터 작업을 위한, 펌웨어 성능 분석에 대한 이해 정도			

피 드 백

1. 문제 해결 시나리오
   - 피교육자들이 작성한 펌웨어에 대한 장단점을 평가하여 피드백한다.
   - 평가된 펌웨어에 대한 장단점을 서로 비교하고 평가한 결과를 공유한다.

2. 평가자 질문
   - 제어 펌웨어에 관한 질문 목록을 만들어 학습 시간에 피평가자에게 질문을 한 후 답변에 대하여 조언을 한다.
   - 답변을 잘 하지 못한 학생은 제어 펌웨어에 관한 보고서 제출 기회를 부여한다.

3. 구두 발표
   - 구두 발표한 내용을 평가하고 그 결과를 확인할 수 있도록 한다.

학습 1	로봇 액추에이터 제어 소프트웨어 사양분석 및 구조 설계하기
학습 2	로봇 액추에이터 제어기 펌웨어 설계하기
학습 3	**로봇 액추에이터 제어기 게인 설정하기**
학습 4	로봇 액추에이터 제어기 제어 성능 시험하기

3-1. 로봇 액추에이터 제어기 게인 설정

학습 목표

- 로봇 액추에이터 제어게인 튜닝을 위한 요구 사항과 계획을 작성할 수 있다.
- 로봇 액추에이터 제어기 게인설정을 위한 제어 구조를 파악할 수 있다.
- 로봇 액추에이터 제어 성능평가 시뮬레이션 소프트웨어를 이용하여, 제어 게인이 적절하게 설정되었는지 평가할 수 있다.
- 로봇 액추에이터 제어기 게인 설정을 통해 로봇 제어 요구 사항을 만족하는지 제어 성능을 분석할 수 있다.

필요 지식 /

1 게인 튜닝

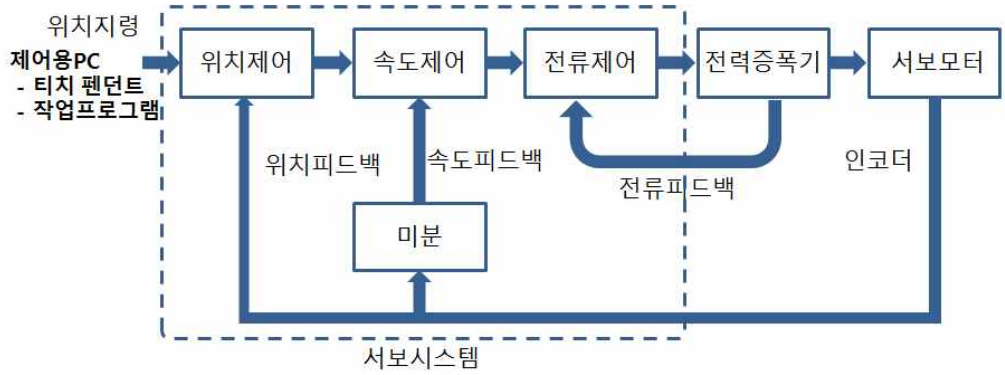
로봇과 같은 액추에이터 제어의 목적은 액추에이터를 정확하고 빠르게 위치, 속도, 힘을 제어하는 것이다. 그리하여 제어계는 주 CPU를 내장한 주 제어계(main controller)와 서보 제어계(servo driver)로 크게 나누어진다.

1. 주 제어계(main controller)

주 제어계는 교시 조작반(teach pendant)의 키(key) 조작을 통해 로봇 위치 지령을 생성하고, 교시된 프로그램을 통해서 로봇 위치 지령을 반복 수행토록 한다.

2. 서보 제어계(servo driver)

서보 제어계는 지령치에 따른 로봇이 동작하도록 추종 제어를 수행한다. 그리하여 제어계 설계 과제를 로봇의 고속, 고정도 위치 도달을 위한 요구 사항에 대해 모터의 과열과 발진이 없고 팔(arm)에 진동이 발생하지 않도록 제어기와 보상기를 설계하는 것이다. 여기서는 실제로 수직 관절형 로봇에 사용된 디지털 서보 제어 시스템을 설명하도록 한다. 제어계는 [그림 3-1]에서 보는 바와 같이 주 CPU로부터 발생된 위치 지령을 추종하는 시스템으로서 각 역할은 다음과 같다.



[그림 3-1] 로봇 제어계 및 서보 시스템 구성도

(1) 위치 제어 루프

최종 제어 목적인 위치를 제어하며 응답 주파수는 로봇 기구부를 포함하여 5-0Hz 범위가 된다.

(2) 속도 제어 루프

위치 루프를 안정화하는 내부 루프로서 모터 속도를 제어하며 응답 주파수는 100-200Hz 범위가 된다.

(3) 전류 제어 루프

위치, 속도 루프를 안정화하는 내부 루프로서 모터 전류를 제어하며 응답 주파수는 2-3Hz 범위가 된다.

3. 서보 게인 튜닝(servo gain tuning)

로봇 제어 서보 시스템에서 보는 바와 같이 전류 루프(loop)가 가장 안쪽에 위치하여 가장 먼저 모터 전류를 제어하여 원하는 전류를 출력한다. 그리고 모터의 엔코더(encoder)로부터의 신호를 미분하면 각속도가 산출되므로 이를 통해 속도 제어를 하고, 마지막으로 엔코딩 신호를 통해 원하는 위치로 제어를 수행한다. 여기서는 각 제어 루프에서의 서보 게인 튜닝(servo gain tuning)에 대해 설명하고자 하며 튜닝 순서는 다음과 같다.

1 단계: 모터 전류 루프 게인 튜닝(주파수 응답을 측정한다)

2 단계: 모터 속도 루프 게인 튜닝(주파수 응답을 측정한다)

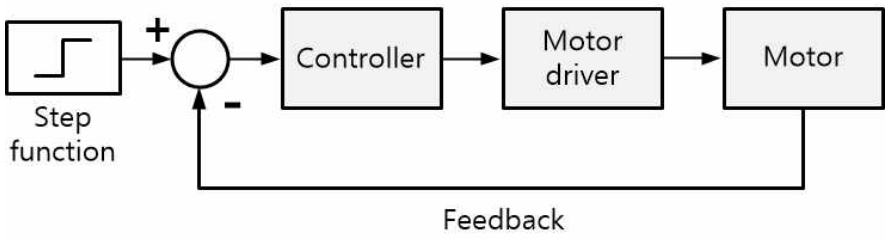
3 단계: 모터 속도-토오크 특성 변수 튜닝(모터의 T-N 특성을 측정한다)

4 단계: 로봇 속도 루프 게인 튜닝(주파수 응답을 측정한다)

5 단계: 로봇 위치 루프 게인 튜닝(로봇 끝단의 동적 응답을 측정한다)

2 제어기 개요

본 절에서는 단입출력 선형시불변 시스템에 대한 산업 현장에서 가장 많이 쓰이고 있는 PID제어기 설계법을 다룬다. 이 제어기들은 제어 이론에서 다룬 앞섬·뒤짐 보상기와 함께 주로 근궤적법이나 주파수응답법 따위로 설계되며 제어 공학의 초기에서부터 쓰여 지고 있기 때문에, 다양한 설계 기법을 사용하는 현대 제어기에 대비하여 고전적 제어기라고 부른다.



[그림 3-2] step 입력에 따른 제어기 구성도

1. 기본 용어

(1) PID 제어기

구조가 간단하고 제어 성능이 우수하고 제어 이득 조정이 비교적 쉽기 때문에 산업 현장에서 약 80% 이상을 차지할 정도로 많이 사용되고 있다. PID 제어는 비례 제어, 적분 제어, 미분 제어를 단독으로 쓰거나 혹은 두 가지 이상을 결합한 형태로 사용한다.

(2) 비례(P) 제어

PID 제어기에서 반드시 사용하는 가장 기본적인 제어이며 구현하기가 쉽다. 그러나 이 제어만으로는 적분기가 플랜트에 없을 경우에 정상상태오차가 발생할 수 있다.

(3) 적분(I) 제어

정상상태오차를 없애는데 사용된다. 그러나 적분 이득을 잘못 조정하면 시스템이 불안해지고 반응이 느려진다.

(4) 미분(D) 제어

미분 제어를 잘 활용하면 안정성에 기여하고, 예측 기능이 있어 응답 속도를 빠르게 한다. 단점으로는 시스템에 잡음 성분이 있을 때 미분값이 커지게 되어 제어 입력에 나쁜 영향을 미친다는 점이 있다.

(5) 동조(tuning)

PID 제어기의 계수를 자동적으로 조정하는 것을 자동 동조(auto-tuning)라고 하며, 산업계에서 많이 필요로 하고 있다. 대표적인 방법으로는 지글러-니콜스(ziegler-nichols) 동조법, 계전기(relay) 동조법 따위가 있으며, 이 방법들은 제어 대상 시스템의 모델을 사용하지 않고 간단한 동조 과정을 거쳐 계수를 결정한다.

(6) 계수 설정

PID 제어기 계수는 제어 대상 시스템의 모델이 주어질 경우에 주파수 영역 설계법, 근궤적법, 과도응답법 등을 사용하여 반복 과정을 통해 설계할 수 있다.

(7) 누적 방지

PID 제어기에 구동기를 연결하여 사용할 때 구동기에 포화 특성이 있으면 적분 누적(integrator windup) 현상이 생겨 불안정하게 되는데, 이를 막기 위하여 누적 방지(antiwindup) 기법을 사용한다.

(8) 제어기 형태

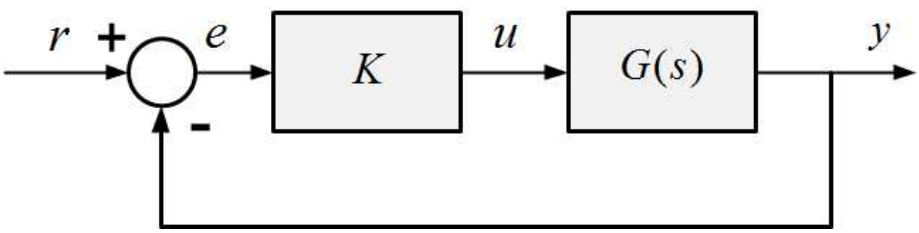
PID 제어기의 형태는 병렬형이 기본형이지만 실제로 구현할 때에는 필요에 따라 직렬형으로 하거나 미분기 앞단에 필터를 부착하는 등 여러 가지로 변형되어 쓰이고 있다.

2. 제어기 응용

PID 제어기는 제어 성능이 우수하고 또한 제어 이득의 조정이 비교적 쉽기 때문에 산업 현장에 많이 쓰이고 있으나 적용 대상이 단입출력 시스템에 한정되는 제약성이 있다. 따라서 입력과 출력이 각각 2개 이상씩인 다변수 시스템에 그대로 적용할 수는 없으며, 만일 이 경우에도 PID 제어기를 쓰고자 한다면 입력과 출력을 일대일로 대응시키는 분해 과정을 거쳐 여러 개의 단입출력 모델을 구하고 각각에 대해 이 제어기를 적용해야 한다. 그러나 이 모델 분해 과정은 대부분의 경우에 매우 어렵기 때문에 이 경우에는 다변수 시스템에 직접 적용할 수 있는 현대 제어 기법을 사용한다.

3 비례(P) 제어기

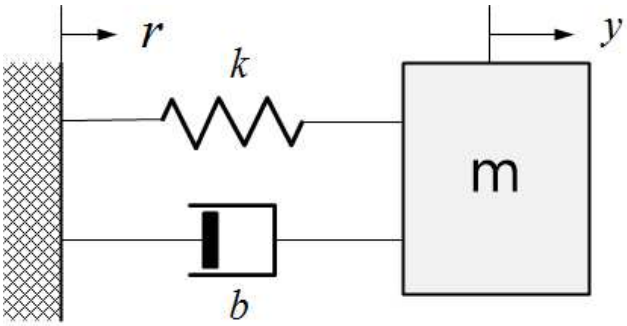
비례 제어란 기준신호와 되먹임신호 사이의 차인 오차신호에 적당한 비례상수 이득을 곱해서 제어신호를 만들어내는 제어 기법을 말하며, 오차신호에 비례하는(proportional) 제어신호를 만든다는 뜻에서 이 기법에 의한 제어기를 비례 제어기(proportional controller), 또는 영문약자를 써서 P 제어기라고 부른다. [그림 3-3]의 토막선도는 플랜트에 비례 제어기를 연결해서 구성한 되먹임 제어 시스템의 일반적인 형태를 보여주고 있다. 이 그림에서 K는 비례 제어기의 이득이다.



[그림 3-3] 비례 제어기에 의한 되먹임 시스템

위의 토막선도에서 볼 수 있는 바와 같이 비례 제어기는 구성이 간단하여 구현하기가 쉽

다. 그러나 이득 K의 조정만으로는 시스템의 성능을 여러 가지 면에서 함께 개선시키기는 어렵다. [그림 3-4]와 같은 스프링-댐퍼 시스템의 예를 들어서 이 점에 대해 살펴보기로 한다. 이 시스템에서의 제어 문제는 스프링과 댐퍼가 달려있는 질량의 변위 y를 기준 입력 r 을 사용하여 조절하는 것이다.



[그림 3-4] 스프링-댐퍼 시스템

이 시스템의 개로 전달함수는 다음과 같이 주어진다.

$$G(s) = \frac{Y(s)}{R(s)} = \frac{k}{ms^2+bs+k} \qquad \langle 3\text{-}1 \rangle$$

여기서 b는 댐퍼의 점성마찰계수, k는 스프링의 탄성계수이다. 이 시스템에 이득이 $K>0$ 인 비례 제어기를 달아서 [그림 3-3]과 같은 폐로 시스템을 구성할 때, 폐로 특성방정식은 $1+KG(s)=0$ 으로부터 다음과 같이 구할 수 있다.

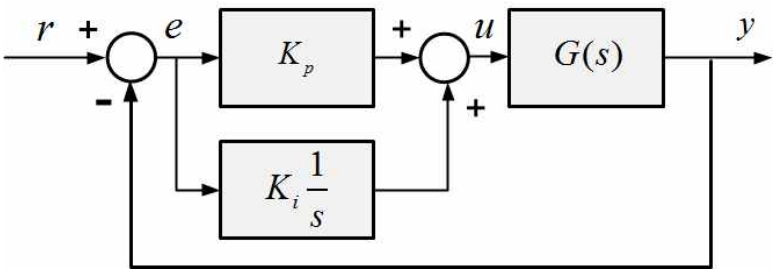
$$ms^2+bs+k+Kk = 0 \qquad \langle 3\text{-}2 \rangle$$

이 특성방정식에서 알 수 있는 것처럼 비례 제어기의 이득은 시스템의 탄성계수를 k에서 $(1+K)k$로 증가시키는 역할을 한다. 따라서 비례 제어기에 의한 되먹임은 마치 탄성계수가 Kk인 스프링을 시스템에 병렬로 더 연결한 것과 같은 효과를 내게 된다. 이 사실은 위의 폐로 전달함수에서$K=0$인 경우에 특성방정식이 개로 시스템의 특성방정식과 같아지는 것을 생각해보면 쉽게 알 수 있다.

4 비례적분(PI) 제어기

비례적분 제어란 오차 신호를 적분하여 제어 신호를 만들어내는 적분 제어를 앞 절에서 살펴본 비례 제어에 병렬로 연결하여 사용하는 제어 기법을 가리킨다. 비례 제어 부분과 더불어 오차 신호를 적분(integral)하여 제어 신호를 만드는 적분 제어를 함께 쓴다는 뜻에서 비례적분 제어기(proportional-integral controller), 또는 영문약자를 써서 PI 제어기라고 부른다. [그림 3-5]의 토막선도는 플랜트에 비례적분 제어기를 연결해서 구성한 되먹임 제어 시스템을 보여주고 있다. 오차 신호와 제어 신호 사이의 전달함수로 표시되는 PI제어기의 전달함수는 다음과 같은 꼴로 나타난다.

$$C(s) = K_p + \frac{K_i}{s} \qquad \langle 3\text{-}3 \rangle$$



[그림 3-5] PI 제어기에 의한 되먹임 시스템

여기서 K_p는 비례계수, K_i는 적분계수라고 부른다. 이 제어기의 제어 신호를 시간 영역에서 나타내면 다음과 같다.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau \qquad \langle 3\text{-}4 \rangle$$

[그림 3-5]에서 제어 대상 플랜트의 전달함수가 다음과 같은 2차 시스템으로 주어진다고 가정하자:

$$G(s) = \frac{\omega_n^2}{s(s + 2\zeta\omega_n)} \qquad \langle 3\text{-}5 \rangle$$

여기서 ζ와 ω_n는 상수이다. 이와 같이 2차 시스템을 제어 대상으로 하여 PI 제어기로 구성된 되먹임 제어 시스템의 동작을 살펴보기로 한다. 이 경우에 식 <3-3>의 PI 제어기를 결합한 전체 시스템의 개로 전달함수는 다음과 같이 구해진다.

$$L(s) = G(s)C(s) = \frac{\omega_n^2 (K_p s + K_i)}{s^2 (s + 2\zeta\omega_n)} \quad \langle 3\text{-}6 \rangle$$

이 개로 전달함수에서 볼 수 있듯이, PI 제어기는 $s = -K_i/K_p$에 있는 영점과 원점 $s = 0$에 있는 극점을 개로 전달함수에 첨가하는 역할을 한다. P I제어에 적분 제어가 들어감으로써 한 가지 분명한 효과는 시스템의 형(type)이 1차 증가시키므로 해서 시스템의 정상상태오차가 한 차수만큼 개선된다. 즉 어떤 입력에 대한 정상상태오차가 0이 아닌 상수라면 적분 제어는 그 오차를 0으로 만든다. 물론 이처럼 정상상태오차를 감소시키는 것은 되먹임 시스템이 안정하게 설계된 경우이다. PI 제어기의 적분 제어 동작에 의해 1형 시스템이 2형 시스템으로 바뀜으로써 계단입력과 경사입력에 대한 정상상태오차는 항상 0이 된다. 이 정상상태 동작은 적분 제어에 의해 결정되는 것이며, 여기에 비례계수 K_p는 영향을 미치지 않으므로 이 계수를 조정하여 다른 특성을 개선하는데 활용할 수 있다. 이제 남은 문제는 만족할만한 과도응답을 얻을 수 있도록 K_p, K_i의 적절한 조합을 선택하는 일이다. 이 문제를 셈툴을 써서 다루어보자.

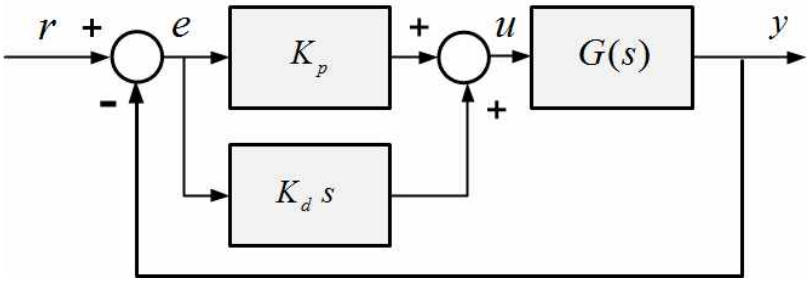
5 비례미분(PD) 제어기

비례미분 제어란 오차신호를 미분하여 제어신호를 만들어내는 미분제어를 비례제어에 병렬로 연결하여 사용하는 제어 기법이다. 비례제어 부분과 더불어 미분제어를 함께 쓴다는 뜻에서 비례미분 제어기(proportional-differentiation controller), 또는 영문약자를 써서 PD 제어기라고 부른다. [그림 3-6]의 토막선도는 플랜트에 비례미분 제어기를 연결한 되먹임 제어시스템을 보여주고 있다. 오차신호와 제어신호 사이의 전달함수로 표시되는 PD제어기의 전달함수는 다음과 같은 꼴로 나타난다.

$$C(s) = K_p + K_d s \quad \langle 3\text{-}7 \rangle$$

여기서 K_p는 비례계수, K_d는 미분계수이다. 이 경우에 제어신호는 시간영역에서 다음과 같이 나타낼 수 있다.

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t) \quad \langle 3\text{-}8 \rangle$$



[그림 3-6] PD 제어기를 포함하는 되먹임 제어시스템

[그림 3-6]에서 제어 대상 플랜트의 전달함수가 다음과 같은 2차 시스템으로 주어진다고 가정한다.

$$G(s) = \frac{\omega_n^2}{s(s+2\zeta\omega_n)} \qquad \langle 3\text{-}9 \rangle$$

위의 2차 시스템은 앞 절에서 PI 제어기의 특성을 살펴볼 때 썼던 것과 똑같은 시스템이다. 여기서도 같은 시스템을 제어 대상으로 하여 비례미분(PD) 제어기로 구성된 되먹임 제어시스템의 동작을 살펴보기로 한다. 이 경우에 식 〈3-7〉)의 PD 제어기를 결합한 전체 시스템의 개로 전달함수는 다음과 같이 구해진다.

$$L(s) = G(s)C(s) = \frac{Y(s)}{E(s)} = \frac{\omega_n^2(K_p + K_d s)}{s(s+2\zeta\omega_n)} \qquad \langle 3\text{-}10 \rangle$$

이 식에서 알 수 있듯이, PD 제어기는 개로 전달함수에 $s = -K_p/K_d$인 영점을 첨가하는 역할을 한다. 미분제어는 오차신호의 미분값에 비례하는 제어신호를 되먹임 시켜 오차신호의 변화를 억제하는 역할을 하기 때문에 감쇠비를 증가시키고 초과를 억제하는 데에 효과적이다. 이러한 미분제어의 효과를 고려하여 PD 제어기를 적절히 설계하면 시스템의 과도응답특성을 개선시킬 수 있다. 그러나 PD 제어기를 사용하는 경우에는 시스템 형식이 증가하지 않기 때문에 정상상태응답특성은 개선되지 않으므로 주의해야 한다. 그러면 예제8.1에서 다루었던 것과 같은 항공기 자세제어에 PD 제어기를 적용하는 문제를 풀어본다.

6 비례적분미분(PID) 제어기

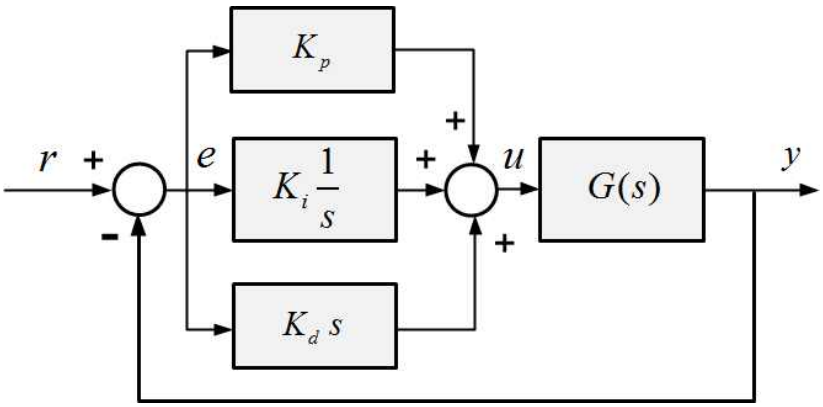
앞 절에서 살펴본 바와 같이, PD 제어기는 시스템의 감쇠비를 증가시키지만 정상상태응답을 개선하는 데에는 효과가 없으며, PI 제어기는 감쇠비도 증가시키고 동시에 정상상태오

차도 개선시키지만 상승시간이 느려지는 등 과도응답에는 불리하다는 것을 알았다. 따라서, 정상상태응답과 과도상태응답을 모두 개선하려면 PI와 PD 제어기의 장점들을 조합하는 방법을 자연스럽게 생각할 수 있는데, 이러한 목적으로 제안된 제어기가 바로 PID 제어기이다. PID 제어기는 비례(P), 적분(I), 미분(D) 제어의 세 부분을 병렬로 조합하여 구성하는 제어기로서 [그림 3-7]의 토막선도는 플랜트에 PID제어기를 연결한 되먹임 제어시스템을 보여주고 있다. PID 제어기의 전달함수는 다음과 같다.

$$C(s) = \frac{U(s)}{E(s)} = K_p + K_d s + \frac{K_i}{s} \qquad \langle 3\text{-}11 \rangle$$

여기서K_p, K_d, K_i는 비례계수, 미분계수, 적분계수이다. 제어신호는 시간영역에서 다음과 같이 나타낼 수 있다.

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t) + K_i \int_0^t e(\tau) d\tau \qquad \langle 3\text{-}12 \rangle$$



[그림 3-7] PID 제어기에 의한 되먹임 제어 시스템

이 제어기를 설계하는 방법에는 여러 가지가 있을 수 있는데, 앞에서 익혀온 PI 제어기와 PD 제어기 설계법을 활용할 수 있는 방법으로서 다음과 같은 분해식 접근 방법이 있다.

1. PID 제어기 전달함수는 다음과 같이 분해하여 쓸 수 있다.

$$C(s) = K_p + K_d s + \frac{K_i}{s} = (1 + K_{d1} s)\left(K_{p2} + \frac{K_{i2}}{s}\right) \qquad \langle 3\text{-}13 \rangle$$

이 식은 PID 제어기가 PI 부분과 PD 부분의 직렬 접속으로 이루어질 수 있음을 보여주고 있다. PID 제어기에서는 3개의 계수가 필요하므로 식 <3-13>의 둘째 등식에서 PD 부분의 상수항을 1로 놓은 것이다. 이 등식은 항등식이므로 양변의 계수를 비교하여 대응시키면 다음과 같은 관계를 얻는다.

$$\begin{aligned} K_p &= K_{p2} + K_{d1}K_{i2} \\ K_d &= K_{d1}K_{p2} \\ K_i &= K_{i2} \end{aligned} \qquad \text{<3-14>}$$

2. 먼저 K_{i2}와 K_{p2}로 이루어지는 PI 부분을 설계한다. 적분계수에 의해 시스템의 형식이 한 차수 커져서 정상상태오차가 개선되므로 시스템의 상승시간에 대한 요구를 만족하게끔 K_{i2}와 K_{p2} 값을 택한다. 이 단계에서는 최대초과가 크더라도 다음 단계에서 설계할 미분제어에 의해 조절할 수 있으므로 상관하지 않는다.
3. 이제 최대초과를 감소시키기 위해 D 부분을 이용한다. 단계 2)에서 설계한 PI 부분을 포함하는 대상 공정에 대해 설계 목표를 만족시킬 수 있는 감쇠비 요구 조건에 맞게끔 K_{d1}값을 택한다.
4. 단계 2와 단계 3에 의해 K_{i2}, K_{p2}, K_{d1} 값이 정해지면 식 <3-14>의 관계를 써서 K_p, K_d, K_i를 구한다.

   다른 방법으로서, PD 부분을 먼저 설계하고 다음에 PI 부분을 설계하는 방식도 있다. 이 경우에는 초과나 상승시간 등의 설계 목표에 맞는 K_{d1}값을 먼저 적절히 선택한다. 그리고 PD 제어 단독으로는 필요한 상대 안정성에 맞지 않게 될 가능성이 있으므로, 이 문제는 추가되는 PI 부분에 의해 필요한 성능 목표를 만족하도록 조정함으로써 PID 제어기를 설계한다.

수행 내용 / 로봇 액추에이터 제어기 게인 설정하기

재료 · 자료

- 액추에이터 데이터 시트
- 액추에이터 구동 회로도
- 기구 동작 제원서
- 로봇에 대한 사용자 요구 사양서
- 로봇 소프트웨어 요구 사항 명세서
- 로봇 기능 명세서

기기(장비 · 공구)

- 컴퓨터, 프린터
- 빔 프로젝트, 모니터
- 프로그램 개발용 소프트웨어
- 문서 작성용 소프트웨어
- 프로그램 성능 평가용 에뮬레이션 소프트웨어
- 로봇 액추에이터 제어 성능 시험 소프트웨어

안전 · 유의 사항

- 액추에이터는 대동력 기기로서 높은 전압과 큰 전류를 소비를 하므로 취급 시 특히 전기 안전에 유의해야 한다.
- 액추에이터의 정 · 역회전 제어와 같이 상반된 동작의 경우에는 오조작이나 제어기 고장에 대비해 전기적 인터록은 물론 상황에 따라 기계적 인터록도 고려해야 한다.
- 액추에이터의 선정 계산 방법은 해당 제품의 카탈로그 및 규격화된 계산 공식을 따른다.
- 액추에이터의 설치 방법은 해당 제품의 사용 설명서와 제조사에서 규정한 내용을 포함하며, 회사 내 해당 작업의 규정에 따른다.

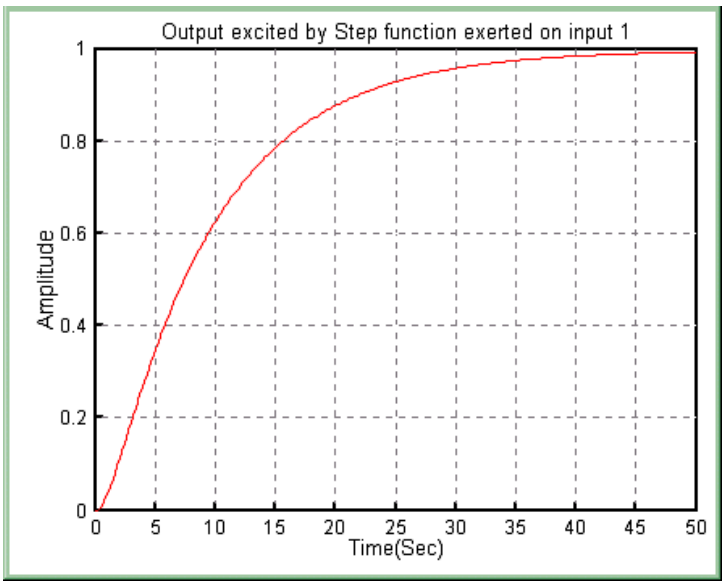
수행 순서

1 비례(P) 제어기를 구현한다.

시스템 계수들이 $m=1$, $b=1$, $k=0.1$ 인 경우에 다음과 같은 명령을 실행시켜서 이 시스템의 개로 계단응답을 구해본다.

계단응답의 결과가 [그림 3-8]과 나오는지를 확인한다.

```
num=0.1;
den=[1 1 0.1];
step(num,den);
```



[그림 3-8] 스프링-댐퍼 시스템의 개로 계단응답

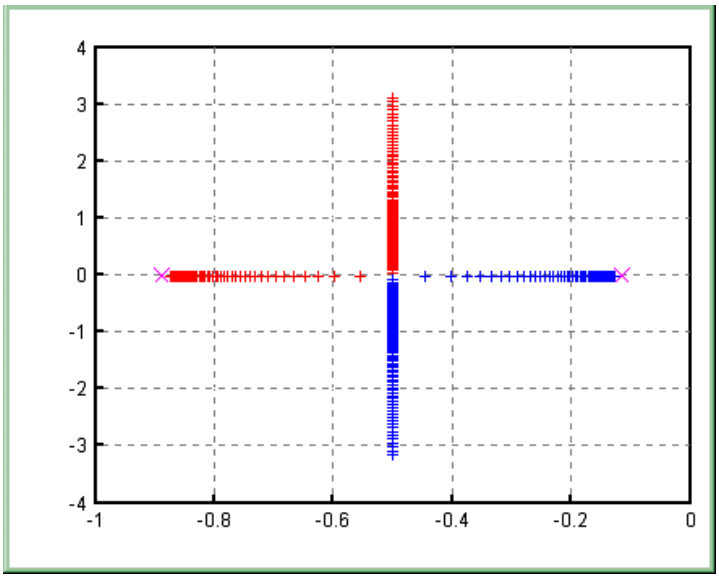
[그림 3-8]의 응답을 보면 개로 시스템에서 정상상태오차는 0이지만 상승시간이 20초 정도로서 반응이 상당히 느린 특성을 파악한다.

이제 비례 제어기를 써서 되먹임 시스템을 구성할 경우에 제어 이득 K를 조정해서 이 시스템의 성능을 얼마나 개선시킬 수 있는지 알아보기로 한다.

이것을 알아보기 위해서 제6장에서 다룬 근궤적법을 사용한다. 이 경우에 근궤적을 그리기 위한 묶음 파일은 다음과 같이 실행한다.

```
num = 0.1;
den = [1 1 0.1];
rlocus(num,den);
```

위의 프로그램을 실행하여 [그림 3-9]와 같은 근궤적을 그려본다.

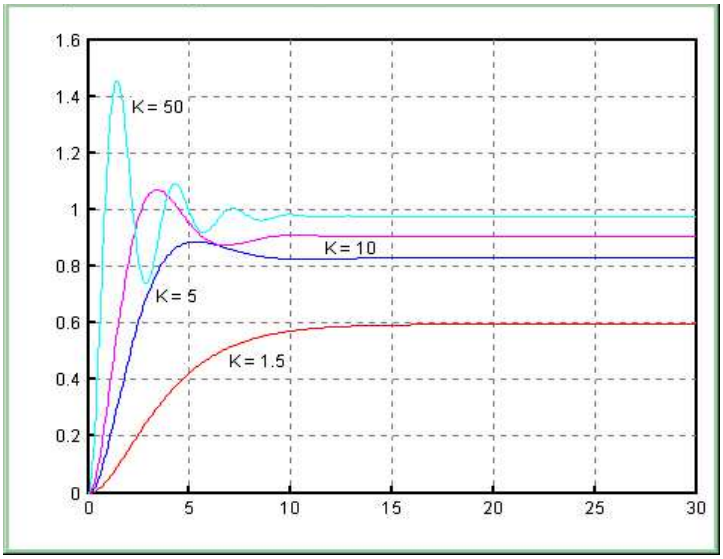


[그림 3-9] P 제어기 되먹임 시스템의 근궤적

이 근궤적에서 근이 실수축을 벗어나는 경계점은 $s=-0.5$ 이고, 이 점에 해당하는 이득 K 의 값을 다음과 같이 구해본다.

```
CEMTool>>rlocval(num,den,-0.5)
          1.5000
```

따라서 $0<K\leq 1.5$ 일 때에는 폐로 극점이 모두 실수축 위에 있으며 K의 값이 커짐에 따라 1개의 근은 원점에서 멀어지고, 다른 1개의 근은 원점 쪽으로 다가온다. 일반적으로 근이 원점에서 멀어지면 시정수가 작아져서 빠른 응답을 나타내게 되는데, 이 시스템의 경우에는 이득이 $K=1.5$ 일 때 근이 실수축 위에 있으면서 원점으로부터 가장 멀리 있게 된다. 이때의 계단응답을 구해보고 응답특성을 [그림 3-10]과 같이 나오는지를 확인한다.
이 응답을 보면, 상승시간이 8초 정도로서 개로 시스템응답에 비해 상당히 빨라졌으나, 정상상태오차가 40%로 크게 나타나고 있다. 이 정상상태오차를 줄이려면 비례 이득을 더 증가시켜야 하는데 이렇게 할 경우에는 폐로 근들이 실수축을 벗어나면서 근의 허수부분이 커지기 때문에 정상상태오차와 상승시간은 줄어들지만, 반면에 초과가 커지는 현상을 관찰하도록 한다.
이러한 현상에 대한 결과를 [그림 3-10]의 계단응답을 통해 확인하도록 한다.



[그림 3-10] P 제어기의 이득변화에 따른 계단응답 특성

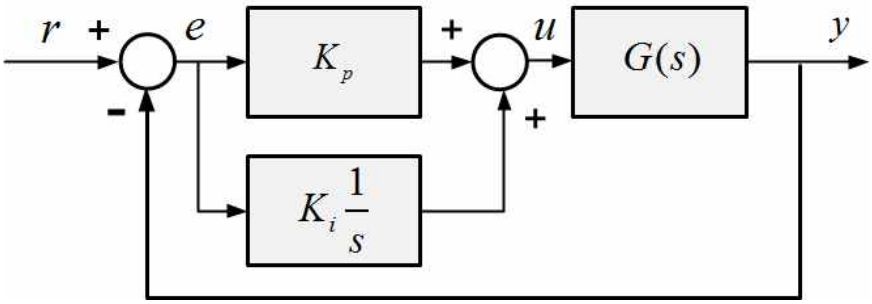
일반적으로 2차 이상의 고차시스템에서는 이득을 크게 하면 정상상태오차를 줄일 수 있지만 지나치게 이득을 높이면 시스템이 불안정해질 수도 있다. 위의 예에서 살펴본 바와 같이, 비례 제어기를 쓰면 시스템의 성능 가운데 한 가지를 개선할 수는 있으나 두 가지 이상을 함께 개선할 수는 없다. 따라서 비례 제어기는 아주 단순한 시스템의 경우를 제외하고는 단독으로 쓰이는 경우가 거의 없으며 다음에 다룰 적분제어나 미분제어와 함께 쓰이게 됨을 알 수 있다.

2 비례적분(PI) 제어기를 구현한다.

항공기에서 승강타와 상승각(pitch angle) 사이의 전달함수가 다음과 같다.

$$G(s) = \frac{4500K}{s(s+361.2)}$$

여기서 K는 공정 내부에서 정해지는 이득으로, 여기서는 단위경사 입력에 대한 정상상태 오차를 0.000443으로 하기 위하여 $K=181.17$로 택한다. 이 항공기의 자세제어 시스템을 [그림 3-11]과 같은 PI 제어기로 설계한다.



[그림 3-11] PI 제어기에 의한 되먹임 시스템

위의 항공기의 자세제어를 위하여 식 <3-15>의 PI 제어기를 연결한 전체 시스템의 개로 전달함수가 다음과 같이 됨을 확인한다.

$$L(s) = G(s)C(s) = \frac{815265\,K_p\,(s + K_i/K_p)}{s^2\,(s + 361.2)} \qquad \langle 3\text{-}15 \rangle$$

이 시스템의 폐로 특성방정식은 $1 + G(s)C(s) = 0$로부터 다음과 같이 된다.

$$s^3 + 361.2\,s^2 + 815265\,K_p\,s + 815265\,K_i \;=\; 03\text{-}16 \qquad \langle 3\text{-}16 \rangle$$

루쓰-허위츠의 안정성 판별법을 이용하여 이 식에 적용해 보면 조건 $0 < K_i < 361.2\,K_p$이 만족될 때에 이 시스템이 안정하게 됨을 확인한다.
이 조건은 $s = -K_i/K_p$에 있는 $L(s)$의 영점이 s평면 좌반면에서 왼쪽으로 너무 멀리 있으면 이 시스템이 불안정하게 됨을 뜻한다. 따라서 실제로 PI 제어기를 설계할 때에는 $s = -K_i/K_p$에 있는 영점이 $L(s)$의 주극점으로부터 될 수 있는 한 멀리 떨어져 있되 원점에 상대적으로 가깝게 있도록 선택한다. 이 예제의 경우에는 $L(s)$의 주극점이 $s = -361.2$에 있으므로 K_i와 K_p의 값은 다음 조건을 만족하도록 선택하도록 한다.

$$\frac{K_i}{K_p} \ll 361.23\text{-}17 \qquad \langle 3\text{-}17 \rangle$$

이 조건을 만족하는 한 예로서 $K_i/K_p = 10$ 으로 할 때 식 <3-16>의 근궤적을 구하는 셈툴 파일은 다음과 같이 입력하여 실행한다.

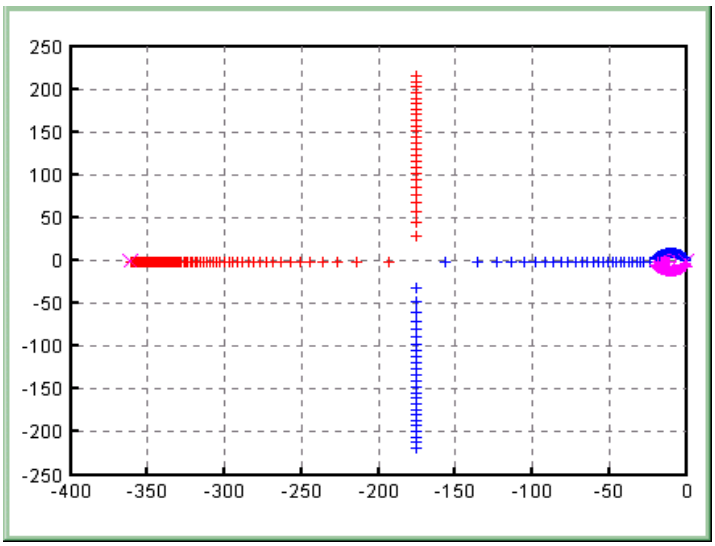
```
num=815265*[1  10];
den=[1  361.2  0  0];
k=logspace(-4,-1,300);
rlocus(num,den,k);
```

입력 코드를 실행한 결과는 [그림 3-12]와 같이 됨을 확인한다.

식 <3-15>에서 분자의 K_i/K_p항은 식 <3-17>의 조건을 만족시키면 감쇠비 $\zeta = 0.707$ 을 실현시키는 점 근처에서 s의 크기에 비교하여 아주 작으므로 무시할 수 있으며 다음과 같이 근사화할 수 있다.

$$L(s) \approx \frac{815265\,K_p}{s\,(s+361.2)} \qquad \text{<3-18>}$$

식 <3-18>과 같이 근사화된 2차 시스템의 감쇠비를 $\zeta = 0.707$ 로 잡을 때 $K_p = 0.08$ 이 된다. 이 값은 K_i/K_p가 식 <3-17>을 만족한다면 PI 제어기를 갖는 3차 시스템에 대해서도 역시 변함이 없음을 확인할 수 있다.



[그림 3-12] $K_i/K_p = 10$ 이고 K_p가 변할 때 식(8.4)의 근궤적

그러므로, $K_p = 0.08$ 및 $K_i = 0.8$ 로 잡은 [그림 3-10]의 근궤적은 두 복소근의 감쇠비가 약 0.707 임을 보여준다. 사실 $K_p = 0.08$ 이고 K_i의 값을 식 <3-17>이 만족되도록 취하는 한, 복소근의 상대적인 감쇠비는 0.707 에 매우 접근하는 것을 관측한다.

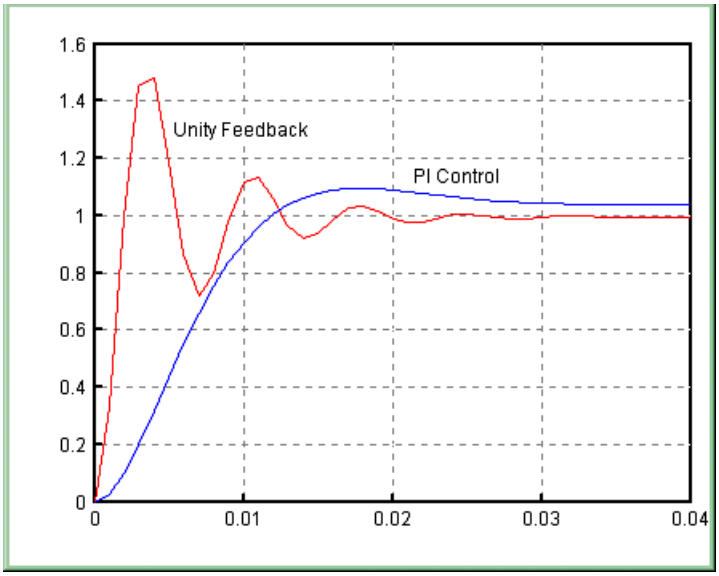
예를 들어 $K_i/K_p=5$ 이면 3개의 특성방정식 근은 $s=-5.145,\ -178.03\pm j178.03$ 에 있고 두 복소근의 상대적인 감쇠비는 0.707 이다. 폐로 전달함수의 실수극점이 이들 각 경우에 이동된다 하더라도 실수극점은 $s=-K_i/K_p$의 영점에 가까워지기 때문에 이 실수극점에 의해 생기는 과도현상은 무시할 수 있다.
예를 들면 $K_i=0.4$ 이고 $K_p=0.08$ 일 때 폐로 전달함수는 다음과 같다.

$$T(s)=\frac{L(s)}{1+L(s)}=\frac{65221.2(s+5)}{(s+5.145)(s+178.03+j178.03)(s+178.03-j178.03)}$$

여기서 $s=-5.145$ 의 극점은 $s=-5$ 의 영점에 매우 접근되어 있으므로 이 극점에 의한 과도응답은 무시되고 이 시스템의 동특성은 근본적으로 나머지 두 복소근에 의해서 지배된다.
설계된 PI 제어기의 성능을 확인하기 위해 단위되먹임만 쓰는 경우와 $K_p=0.08$, $K_i=0.8$ 인 PI 제어기를 써서 보상된 시스템의 단위계단응답을 구하여 함께 그리는 셈툴 파일을 작성하면 다음과 같이 입력한다.

```
num=815265;
den=[1 361.2 815265];
t=0:0.04:0.001;
y=step(num,den,t);
kp=0.08;
ki=0.8;
num=815265*[kp ki];
den=[1 361.2 815265*kp 815265*ki];
y1=step(num,den,t);
plot(t,[y y1]);
```

이 파일을 실행하여 [그림 3-13]과 같은 결과를 관찰한다.
이 응답을 보면 단위되먹임만 사용하는 경우에는 초과가 50% 정도로 크게 나타나지만, PI 제어기를 쓰면 적절히 감쇠가 걸리면서 초과가 10% 정도로 크게 줄어듦을 알 수 있다. 그러나 PI 제어기의 적분 제어항은 계단응답에서 상승시간과 정착시간을 길게 하기 때문에 응답 속도가 늦어지는 것을 관찰하도록 한다.

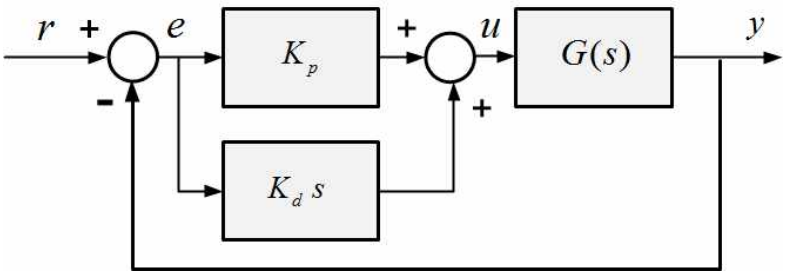


[그림 3-13] PI제어를 포함하는 시스템의 단위계단응답

3 비례미분(PD) 제어기를 구현한다.

항공기 승강타와 상승각 사이의 전달함수가 다음과 같이 주어질 때 항공기의 자세제어 시스템을 [그림 3-14]와 같은 PD 제어기로 설계해보도록 한다.

$$G(s) = \frac{4500K}{s(s+361.2)}$$



[그림 3-14] PD 제어기를 포함하는 되먹임 제어 시스템

여기서 K는 예제8.1에서와 같이 단위경사입력에 대한 정상상태오차를 0.000443으로 하기 위하여 $K = 181.17$ 로 택한다.

게인 설정 : 주어진 K 값에 대하여 이 시스템의 감쇠비는 $\zeta = 0.2$ 이고 최대초과는 52.7%로서 과도응답특성이 만족스럽지 못하다(이러한 특성은 셈툴에서 단위경사응답을 구해보면 쉽게 확인할 수 있다).

PD제어기를 이 시스템의 앞먹임 경로에 삽입하여 정상상태오차를 0.000443으로 유지하면서 이 시스템의 감쇠비와 최대초과가 개선되도록 설계해보자. [그림 3-14]의 PD 제어기가

포함된 개로 전달함수는 다음과 같이 구해보도록 한다.

$$L(s)=\frac{Y(s)}{E(s)}=\frac{815265(K_p+K_d s)}{s\,(s+361.2)} \qquad \langle 3\text{-}19\rangle$$

따라서 이 시스템의 폐로 전달함수는 다음과 같이 되고,

$$T(s)=\frac{Y(s)}{R(s)}=\frac{815265\,(K_p+K_d s\,)}{s^2+(361.2+815265\,K_p\,)\,s+815265\,K_p} \qquad \langle 3\text{-}20\rangle$$

경사오차상수는 다음과 같이 구해진다.

$$K_v=\lim_{s\to\infty} sL(s)=\frac{815265\,K_p}{361.2}=2257.1\,K_p \qquad \langle 3\text{-}21\rangle$$

따라서 단위경사입력에 대한 정상상태오차는 $E_s=1/K_v=0.000443/K_p$이 되며, 이 오차를 0.00043으로 하기 위해서는 비례상수는 $K_p=1$로 하면 된다.

식 <3-20>의 폐로 전달함수에서 볼 수 있듯이, $s=-K_p/K_d$인 영점이 첨가되고 감쇠를 나타내는 분모 1차항의 계수가 원래의 값인 361.2로부터 $361.2+815265\,K_d$로 증가되는 PD 제어기의 효과가 나타난다. 이 시스템의 폐로 특성방정식은 다음과 같으므로,

$$s^2+(361.2+815265\,K_d\,)s+815265\,K_p\;=\;0 \qquad \langle 3\text{-}22\rangle$$

$K_p=1$ 일 때에 감쇠비는 다음과 같다.

$$\zeta=\frac{361.2+815265\,K_d}{2\sqrt{815265}}\;=0.2+451.46\,K_d$$

이 식은 PD 제어기에 의해 감쇠비가 원래값 0.2로부터 $451.46\,K_d$만큼 증가하여 미분제어계수 K_d가 감쇠비에 미치는 결정적인 효과를 보여준다. 또 K_d가 증가하면 폐로 극점 중 하나가 $s=-K_p/K_d$에 있는 영점에 아주 가깝게 접근해서 서로 상쇄되기 때문에 초과가 증가하지 않는 것을 관찰할 수 있다.

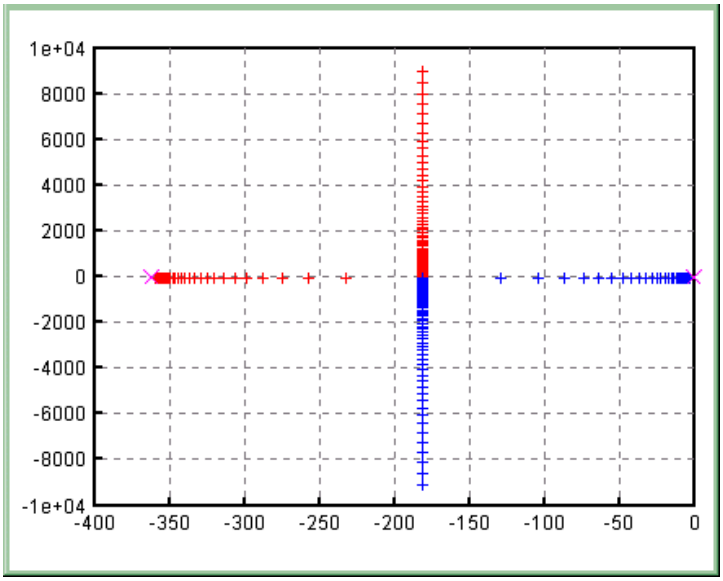
K_p와 K_d의 효과를 알아보기 위해서 식 <3-22>의 특성방정식에 근궤적법을 적용해본다.

먼저 K_d를 0으로 놓으면 식 <3-22>는 다음과 같이 된다.

$$s^2 + 361.2\,s + 815,265\,K_p = 0 \qquad \text{<3-23>}$$

K_p가 변할 때 이 식의 근궤적은 [그림 3-15]와 같고, 이를 위한 셈툴 파일은 다음과 같이 입력하여 출력을 관찰한다.

```
num=[815265];
den=[1 361.2 0];
rlocus(num,den);
```



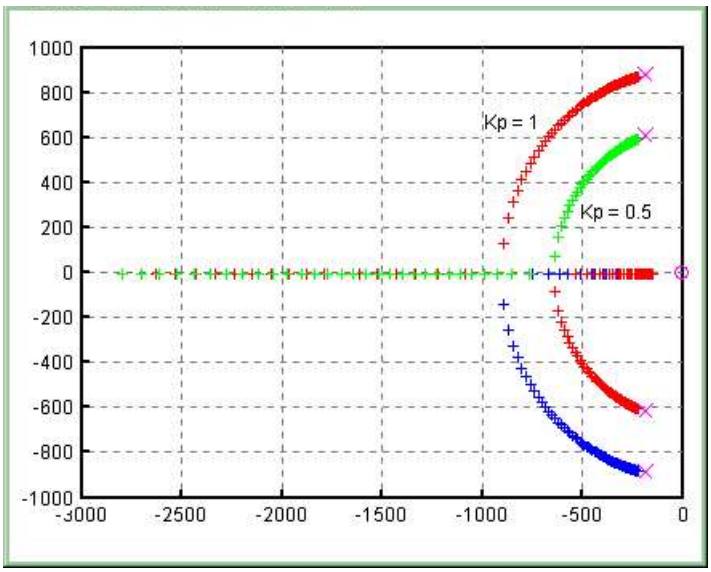
[그림 3-15] 식<3-23>의 근궤적

$K_d \neq 0$ 일 때 근궤적을 조사하려면 식 <3-22>의 특성방정식을 다음과 같이 변형시킨다.

$$1 + K_d G_{eq}(s) = 1 + K_d \frac{815265\,s}{s^2 + 361.2\,s + 815265\,K_p} = 0$$

여기서 $G_{eq}(s)$는 식 <3-22>를 근궤적법의 표준형으로 표시할 때 대응되는 전달함수이다. K_p가 일정하고 K_d가 변할 때 식 <3-22>의 근궤적은 $G_{eq}(s)$의 극영점 형태에 근거하여 구성되는데, $K_p = 1$ 과 $K_p = 0.5$ 인 경우에 근궤적을 구하는 셈툴 파일은 다음과 같이 입력하고 출력을 관찰한다.

```
num=[815265 0];
den=[1 361.2 815265];
k=logspace(-4,-2.5,100);
rlocus(num,den,k);
holdon
den=[1 361.2 815265*0.5];
rlocus(num,den,k);
```



[그림 3-16] K_p가 일정하고 K_d가 변할 때 식 <3-22>의 근궤적

이 파일을 실행시킨 결과는 [그림 3-16]과 동일한지를 관찰한다.

이 그림을 보면, $K_p=1$ 이고 $K_d=0$ 일 때 특성방정식의 근은 $-180.6\pm j884.67$ 이며 감쇠비는 $180.6/\sqrt{180.6^2+884.67^2}=0.2$ 임을 알 수 있다.

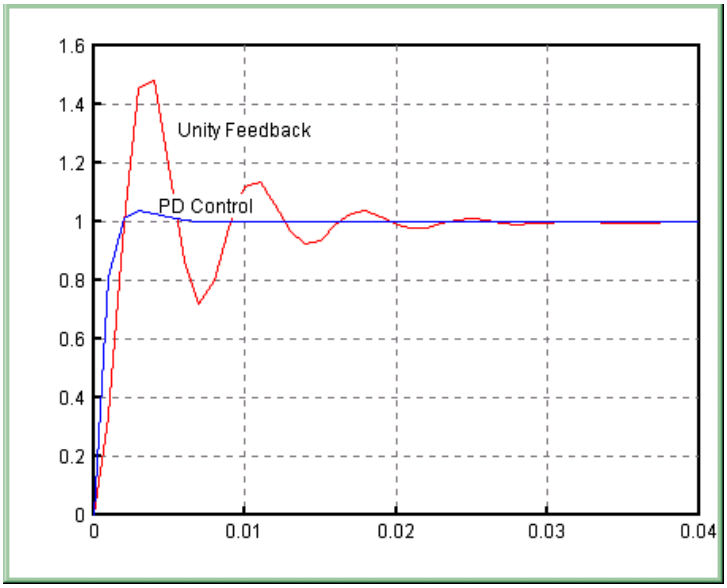
K_d값이 증가할 때 2개의 특성방정식 근은 원호를 따라 실수축을 향하여 이동한다. $K_d=0.001772$ 일 때 근들은 실수 이중근으로서 $s=-902.92$ 에 있으며, 따라서 임계감쇠를 나타낸다. $K_d>0.001772$ 이면 두 근은 서로 다른 실수가 되고 시스템은 과감쇠 특성을 보인다.

이 근궤적은 K_d값이 증가함에 따라 PD 제어기의 효과에 의해 감쇠특성이 개선되는 것을 다시 보여준다.

설계된 PD 제어기의 성능을 확인하기 위해 PD 제어기가 없는 경우와 비교하여 폐로 시스템의 단위계단응답과 함께 나타내는 셈툴 파일을 작성하여 출력을 관찰한다.

```
num=815265;
den=[1 361.2 815265];
t=0:0.04:0.001;
y1=step(num,den,t);
kp=1;
kd=0.001772;
num=815265*[kd kp];
den=[1 361.2+815265*kd 815265*kp];
y2=step(num,den,t);
plot(t,[y1 y2]);
```

이 파일을 실행한 결과인 [그림 3-17]을 보면, PD 제어기가 없는 단위되먹임 시스템의 응답에서는 최대초과가 50% 정도로 크게 나타난다. 그러나 $K_p=1$, $K_d=0.001772$ 인 PD 제어기를 사용할 경우에는 최대초과가 4% 정도로 아주 작게 나타난다(이 경우에 K_d를 임계감쇠인 경우로 선정해도 초과가 나타나는데 이것은 폐로 전달함수의 $s=-K_p/K_d$에 있는 영점 때문이다). 또 PD 제어기의 미분보상효과에 의해 상승시간이 더 짧아짐을 알 수 있다.



[그림 3-17] PD 제어기 자세제어 시스템의 단위계단응답

PD 제어기의 계수 K_p와 K_d의 효과를 분석하는 다른 해석적인 방법은 K_p와 K_d 계수평면에서 성능 특성을 계산하는 것이다. 예제 시스템 경우에 식 <3-22>의 특성방정식으로부터 감쇠비 ζ는 다음과 같이 비례계수 K_p와 미분계수 K_d의 함수로 나타낼 수 있다.

$$\zeta = \frac{361.2 + 815265 K_d}{2\sqrt{815265 K_p}} = \frac{0.2 + 451.46 K_d}{\sqrt{K_p}} \quad \langle 3\text{-}24 \rangle$$

식 <3-22>)에 루쓰-허위츠 안정성 판별법을 적용하여 폐로 시스템이 안정하기 위한 조건을 구하면, $K_p > 0$ 및 $K_d > -0.000443$ 이 된다. 이 안정성조건 아래에서 일정한 ζ에 대하여 식 <3-24>로부터 K_p와 K_d 사이의 관계를 구하면 다음과 같은 2차 함수 관계가 성립함을 관찰 한다.

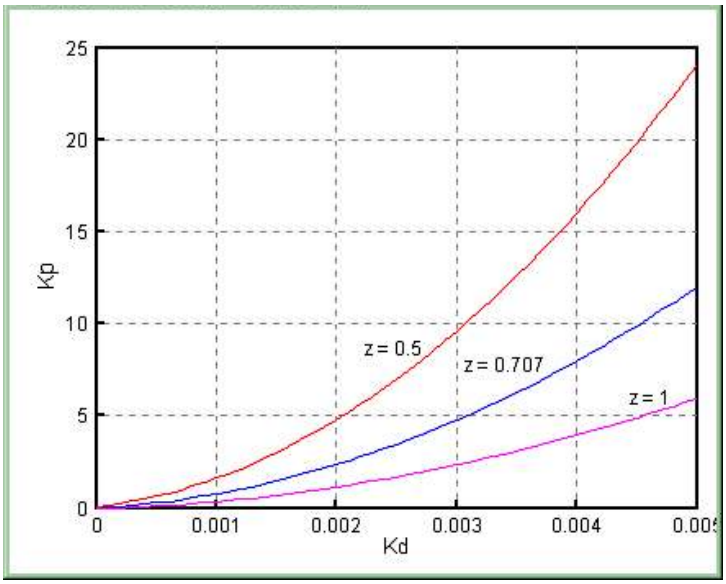
$$K_p = \left(\frac{0.2 + 451.46 K_d}{\zeta} \right)^2$$

[그림 3-18]은 $K_p - K_d$ 평면에서 $\zeta = 0.5, 0.707, 1.0$ 인 세 가지 경우에 대한 일정한 ζ 궤적을 보여주고 있다. 이 그림은 K_p와 K_d의 값이 시스템의 각종 성능지표에 어떻게 영향을 미치는지를 분명히 나타내주고 있다.

경사오차상수 K_v는 식 <3-21>에서와 같이 비례계수 K_p에 정비례하는 함수이고 미분계수 K_d에는 무관하므로, [그림 3-18]의 $K_p - K_d$ 계수평면에서는 수평선으로 표시할 수 있다.

```
kd=[0:0.005:0.0001]';
[n1,n2]=size(kd);
//초기화
kp1=kd;
kp2=kd;
kp3=kd;
zeta=0.5;
for(i=1;i<=n1;i=i+1)
    kp1(i)=((0.2+451.46*kd(i))^2)/(zeta^2);
zeta=0.707;
for(i=1;i<=n1;i=i+1)
    kp2(i)=((0.2+451.46*kd(i))^2)/(zeta^2);
zeta=1.0;
for(i=1;i<=n1;i=i+1)
    kp3(i)=((0.2+451.46*kd(i))^2)/(zeta^2);
plot(kd,[kp1 kp2 kp3]');
xtitle("Kd");
ytitle("Kp");
```

예를 들어 K_v가 2257.1 로 맞춰진다면 이 조건은 $K_p = 1$의 수평선에 대응하는데, 이 그림으로부터 K_d가 증가함에 따라 감쇠도 단조롭게 증가됨을 보여준다. 또 이 그림은 일정한 K_v와 일정한 ζ의 궤적이 만나는 점의 좌표에 의해 원하는 K_v와 ζ에 필요한 K_d, K_p의 값을 위치를 관찰 할 수 있다.



[그림 3-18] PD 제어기를 포함하는 자세제어 시스템의 K_d-K_p 계수평면

4 PID 제어기를 구현한다.

앞 학습에서 다룬 직류서보모터를 제어대상으로 하여 위에서 요약한 설계절차에 따라 PID 제어기를 설계하고 이 설계된 제어기가 PD나 PI 제어기에 비해 어떤 장점이 있는가를 살펴보기로 한다.

제어 대상인 직류서보모터의 전달함수는 식 <3-25>로 주어지고 이 모델에서 전기·기계 상수들이 다음과 같은 조건을 만족한다고 가정할 때, 식 <3-26>과 같이 전달함수를 구해본다.

$$G(s) = \frac{\Theta(s)}{E(s)} = \frac{K_t K_a}{s(L_a s + R_a)(Js + B) + K_t K_b s} \qquad \langle 3\text{-}25 \rangle$$

$K_t = 6 \times 10^{-5}$[Nm/A], $K_b = 5 \times 10^{-2}$[Vsec/rad]

$R_a = 0.2\,[\Omega]$, $L_a \approx 1.33 \times 10^{-2}$[H], $K_a = 5$

$J = 4.5 \times 10^{-6}$[kgm2], $B \approx 0$ [Nm/rad/sec]

이 경우에 직류서보모터의 전달함수는 다음과 같다.

$$G(s)=\frac{K}{s\,(1+0.1s\,)(1+0.2s\,)} \qquad \langle 3\text{-}26\rangle$$

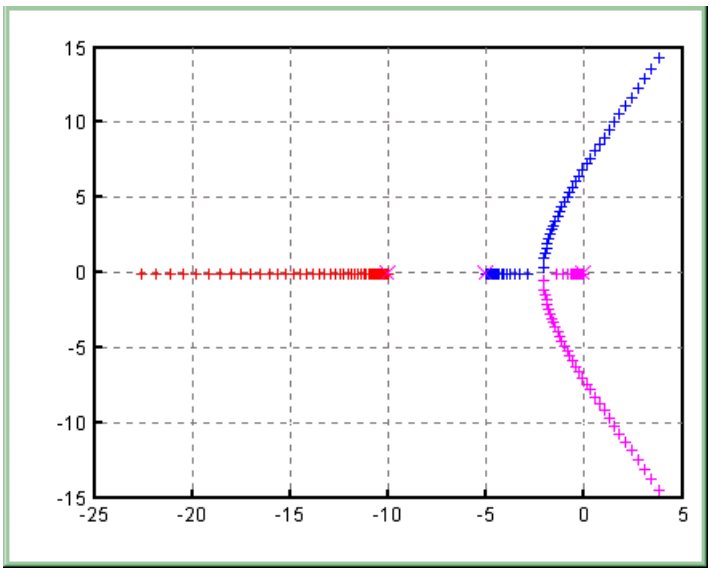
여기서 서보모터 이득상수 $K=100$ 이다. 이 시스템에 대해 단위되먹임을 갖는 폐로 시스템을 구성한다면 개로 전달함수는 $L(s)=G(s)$이므로 경사오차상수는

$$K_v=\lim_{s\to\infty} sL(s)=100$$

이 된다.

[그림 3-19]는 K가 변할 때 시스템의 근궤적을 셈툴 파일과 함께 보인 것이다. $K=100$ 일 때 특성방정식의 2개의 근은 $3.8\pm j14.4$ 로서 s평면 우반면에 있다. 따라서 적당한 제어기를 추가하지 않고 단일 되먹임만으로 구성된 폐로 시스템은 불안정한 상태임을 관찰할 수 있다.

```
num=[1];
den=[0.02 0.3 1 0];
k=logspace(-4,2,100);
r=rlocus(num,den,k);
```



[그림 3-19] 식 <3-26>으로 표시된 시스템의 근궤적

이 시스템에 대해 PD, PI, PID 제어기를 각각 적용하여 되먹임시스템의 성능 평가 과정은 학습 4에서 구체적으로 다루기로 한다.

학습 1	로봇 액추에이터 제어 소프트웨어 사양분석 및 구조 설계하기
학습 2	로봇 액추에이터 제어기 펌웨어 설계하기
학습 3	로봇 액추에이터 제어기 게인 설정하기

학습 4 로봇 액추에이터 제어기 제어 성능 시험하기

4-1. 로봇 액추에이터 제어기 제어 성능 시험

학습 목표

- 로봇 제어성능 시험을 위한 요구사항과 계획을 작성할 수 있다.
- 로봇 제어성능 시험을 위한 제어 구조를 파악할 수 있다.
- 로봇 액추에이터 제어 성능시험 소프트웨어를 이용하여, 로봇 액추에이터 제어 프로그램이 적절하게 개발되었는지 평가할 수 있다.
- 로봇 액추에이터 제어 프로그램이 로봇 제어 요구사항을 만족하는지 제어 성능을 분석할 수 있다.

필요 지식 /

1 제어 성능 시험

본 학습에서는 로봇 제어에 따른 성능을 PID 제어기를 이용한 제어 성능을 평가하는 방법과 제품으로서 로봇의 안전성 및 성능 등이 규정된 요구 사항에 대하여 적합한지 여부를 평가하고 인증하는 국내외 주요 인증제도에 대해서 설명한다.

1. 로봇의 성능

로봇의 성능은 크게 안전성과 기능성으로 나눌 수 있다. 이번에는 로봇의 기능성을 평가하는 방법에 대해 설명한다. 로봇의 다양한 기능성을 객관적으로 평가하기 위한 관련 주체들의 노력은 평가 방법의 표준화로 나타난다. 본 학습에서는 매니퓰레이터형 로봇의 PID 제어기의 주요 성능을 평가하기 위한 방법을 제시한다.

2. 성능 인증

인증이란 제품 서비스 등과 같은 평가 대상이 정해진 표준이나 기술 규정 등에 적합하다는 평가를 받음으로써 그 사용 및 출하가 가능하다는 것을 입증하는 행위를 말하는데, 각 국가에서 인증결과의 국제적 통용을 보장하기 위하여 인증기준을 국제표준에 부합하는 일

과 함께 체계적인 인증 시스템을 도입하여 상호인정 활동을 추진하고 있다. 자유무역의 확대와 더불어, 국제적으로 공히 인정되는 요구 사항의 충족을 통해 시장에서 신뢰를 제공하고 산업계의 경쟁력을 제고한다는 취지 아래 인증제도의 적용 영역이 점차 확대되고 있다.

2 PID 제어기 제어 성능

이 절에서는 안정성 확보와 함께 상승시간과 정착시간을 줄이기 위해 PID 제어기를 식 <4-1>의 직류서보모터에 적용하기로 한다.

$$G(s) = \frac{K}{s\,(1+0.1s\,)(1+0.2s\,)} \qquad \langle 4\text{-}1 \rangle$$

식 <4-2>(8.15)로 표현되는 PID제어기는 설계변수로서 비례계수 K_p, 적분계수 K_i, 미분계수 K_d 등 세 개를 갖고 있기 때문에 설계변수가 2개뿐인 PD나 PI 제어기와는 달리 3차 시스템의 제어 문제에서도 좋은 성능을 보일 것으로 예상할 수 있다.

$$C(s\,) \;=\; \frac{U(s)}{E(s)} \;=\; K_p + K_d s \,+ \frac{K_i}{s} \qquad \langle 4\text{-}2 \rangle$$

PID 제어기를 사용하는 경우에 개로 시스템의 전달함수는 다음 식으로 된다.

$$L(s) \;=\; G(s)C(s) \;=\; \frac{5000\,(1+K_{d1}\,s\,)(K_{p2}\,s+K_{i2}\,)}{s^2(s+5)(s+10)} \qquad \langle 4\text{-}3 \rangle$$

먼저 $K_{d1}=0$ 로 놓고 제어기의 PI 부분을 설계하자. $K_{i2}/K_{p2}=0.1$ 로 잡으면 식 <4-3>의 개로 전달함수는 다음과 같이 바뀐다.

$$L(s) \;=\; \frac{5000\,K_{p2}(s+0.1)}{s^2(s+5)(s+10)}$$

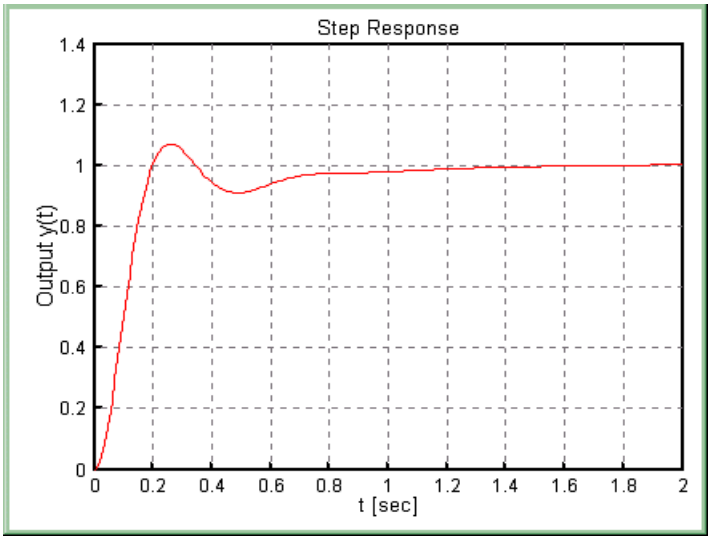
$s=-0.1$ 에 있는 $G(s)$의 영점은 원점에 아주 가깝기 때문에 원점에 있는 이중극점 중 하나를 상쇄시키는 효과를 내므로 $K_{p2}=1$ 일 때 이 시스템이 보상되지 않은 시스템의 과도응답과 아주 비슷한 과도응답을 갖도록 실현하기로 한다.

PI 제어기만 이용할 때 상대적인 감쇠비가 0.707이상이 되도록 하려면 앞 절에서 PI 제어기를 설계할 때 선정하였듯이 $K_{p2} > 0.0163$ 으로 해야 한다. 좀 더 빠른 상승시간을 실현

시키기 위해서 $K_{p2}=0.07$ 로 택하기로 하면 $K_{i2}=0.007$ 이 된다. 이렇게 PI 부분의 설계한 다음에는, 최대초과를 줄이고 상승시간과 정착시간을 빠르게 유지하기 위해 K_{d1} 의 값을 조정한다. 여기서 $K_{d1}=0.5$ 로 선정하면 이들 계수에 식 <3-14>를 적용하여 다음과 같이 PID 제어기 계수를 구할 수 있다.

$$K_p = 0.0735,\ K_i = 0.007,\ K_d = 0.035$$

이 경우에 대응되는 폐로 시스템의 계단응답을 구해보면 [그림 4-1]과 같다. 이 응답의 초과는 약 7%, 상승시간은 $t_r \approx 0.17$[sec], 정착시간은 $t_s \approx 0.90$[sec]로서 매우 만족스러운 특성을 보이고 있다.

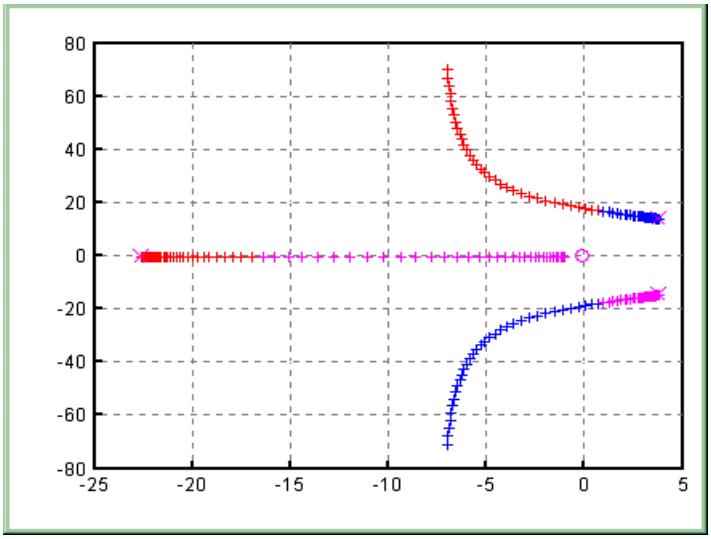


[그림 4-1] PID 제어기에 의한 되먹임 시스템의 계단응답

PID 제어기의 다른 설계 방법으로서 임의로 $K_{d1}=0.5$ 로 놓고 PID 제어기의 PD 부분을 먼저 결정할 수도 있다. [그림 4-2]의 PD 제어기 근궤적에서 $K_{d1}=0.5$ 일 경우에 이 제어기 단독으로는 제어 목표가 달성되지 않음을 알 수 있다.

$$s^3 + 15s^2 + (50 + 5000\,K_d)s + 5000 \;=\; 0$$

```
num=[5000 0];
den=[1 15 50 5000];
k=logspace(-4,0,100);
rlocus(num,den,k);
```



[그림 4-2] PD 제어기 근궤적

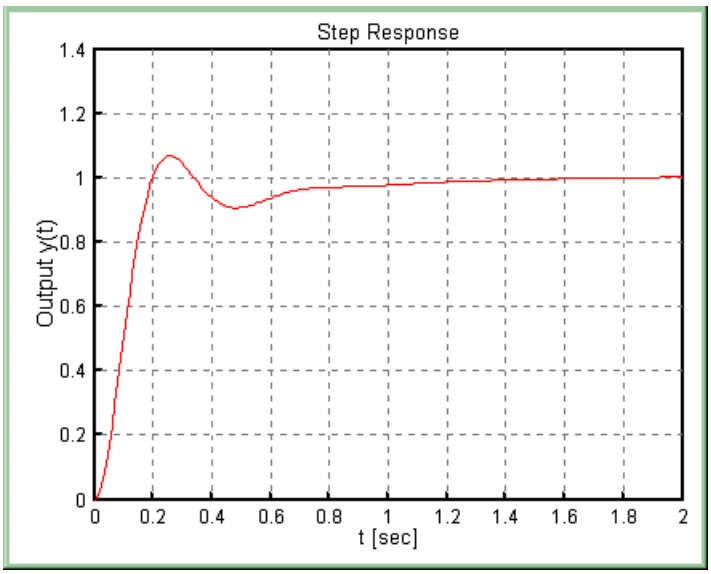
그 다음에 PD 제어기와 대상 시스템을 포함하는 보상 시스템에 대해 PI 제어기를 적용함으로써 PID 제어기를 설계할 수 있다. K_{p2}의 값은 PD 제어기를 포함하는 보상 시스템에 대해 아래식을 적용시켜 구한다.

$$K_p = \frac{\text{원하는 감쇠비에 대한 } K\text{의 값}}{\text{정상상태에 대한 } K\text{의 값}}$$

PD 제어기에 의한 보상 시스템은 $s=-2$에 영점을 가지므로 제어 목표인 감쇠비 $\zeta=0.707$가 달성되지는 않는다. 실제로 이 방식으로 제어기를 설계하면 $K_{p2}=0.707$ 이고 $K_{i2}=0.007$ 이며 PI 부분을 포함할 때 특성방정식의 주요근은 $s=-6.59\pm j12.55$로서 이에 대응하는 감쇠비는 $\zeta=0.46$이다. 이렇게 설계하였을 때 식 <3-14>로부터 계산한 PID 제어기 계수는 다음과 같다.

$$K_p=0.07105,\ K_i=0.007,\ K_d=0.03535$$

이 계수에 대응되는 폐로 시스템의 계단응답을 구해보면 [그림 4-3]과 같다. 이 응답의 초과는 약 7.2%, 상승시간은 $t_r \approx 0.17$[sec], 정착시간은 $t_s \approx 0.98$[sec]로서 [그림 4-1]의 결과와 같이 이 경우에도 매우 만족스러운 특성을 보이고 있다.



[그림 4-3] PID 제어기에 의한 되먹임 시스템의 계단응답

본 학습에서는 PID 제어기의 효과를 알아보기 위해 3차 시스템에 대한 제어기 성능 평가를 보기로 하여 PD, PI, PID 제어기를 설계하고 각각의 성능을 비교해 보았다. 그 결과 PD와 PI 제어기는 3차 시스템에 대한 제어에서 안정성은 이룰 수 있으나 과도응답이나 정상상태응답의 성능 목표를 함께 달성하기가 어렵다는 것을 알았다.

그러나 PID 제어기는 계수를 적절히 조절하면 3차 시스템에 대해서도 안정성과 성능 목표를 함께 만족시킬 수 있음을 확인하였다.

일반적으로 PID 제어기는 전달함수의 분자다항식이 2차로서 2개의 영점을 갖고 있는데, 이 제어기의 영점을 이용하여 제어 대상 시스템에 있는 바람직하지 않은 극점의 위치를 적절히 바꿈으로써 과도응답특성을 개선시킨다. 또 제어기에 적분기를 포함하고 있어 원점에 극점을 갖고 있기 때문에, 이 제어기의 극점이 폐로 시스템에 첨가됨으로써 시스템의 꼴을 증가시켜서 정상상태특성을 개선시키는 효과를 얻을 수 있다. PID 제어기는 이러한 특성에 의해 2차 이하의 주극점을 갖는 대상 시스템의 제어에 매우 효과적임을 알 수 있다.

이제 수행 내용에서 PID 제어기의 계수에 따른 시스템 제어 성능을 평가하는 방법을 학습하도록 한다.

3 지글러-니콜스 PID 계수조정법

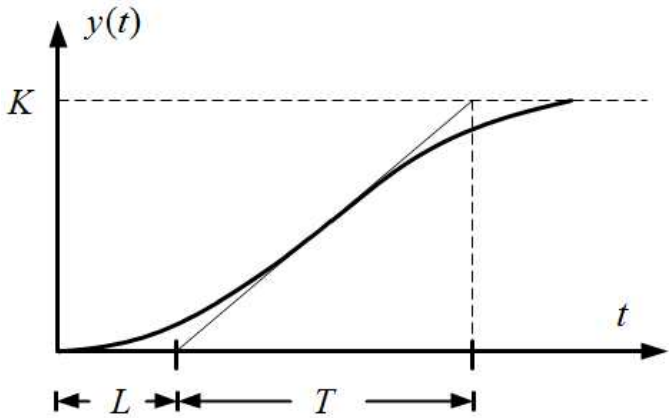
1942년에 지글러(ZIEGLER)와 니콜스(NICHOLS)는 제어 대상 플랜트가 나타내는 과도응답의 형태로부터 PID 제어기의 계수들을 정하는 방법을 제안하였다. 지글러-니콜스 계수조정법이라 불리는 이 방법의 장점은 실제의 제어 대상 시스템에서 간단한 몇 가지의 사전 실험을 하고, 이 실험결과로부터 PID 계수를 간단한 공식에 의해 결정할 수 있다는 것이다. 이 방법에서 사용하는 PID 제어기의 전달함수는 다음과 같다.

$$C(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \qquad \langle 4\text{-}4 \rangle$$

여기서 K_p는 비례계수, T_i는 적분시간, T_d는 미분시간이다. 이 계수조정법에는 두 가지가 있는데, 두 가지의 방법에서 구해지는 계수들을 쓰면 시스템의 계단응답에서 최대초과가 약 25% 정도로 나타난다. 이 계수조정법들은 모두 많은 경험과 실험에 의해 얻어진 방법들로서 이 방법으로 PID 제어기를 설계할 경우 대체로 무난한 성능을 보이기는 하지만 최적의 성능을 보장하는 방법은 아니기 때문에 설계 후에는 반드시 성능 검증을 해야 하며 필요에 따라 정밀한 계수조정작업을 추가로 수행해야 한다.

1. 지글러-니콜스 조정법(1)

첫째 방법은 계단응답곡선을 이용하는 것으로서, 먼저 주어진 플랜트에 단위계단 입력을 넣은 후에 그 출력응답을 구하고, 이 단위계단응답 곡선으로부터 어떤 특성계수를 구하여 이 계수로부터 PID 계수를 선정한다. 이 방법은 대상 플랜트에 적분기가 포함되어 있지 않고 주극점이 복소근이 아닌 안정한 시스템에만 적용할 수 있다. 이 조건이 만족되는 시스템에서의 개로 계단응답 곡선의 모양은 일반적으로 [그림 4-4]와 같은 S자 형태가 된다. 만일 계단응답이 S자형이 아닌 경우에는 이 방법을 사용할 수 없다. S자 모양의 응답곡선의 특성은 [그림 4-4]에서 볼 수 있듯이, 지연시간(delay time) L, 시정수 T, 직류이득 K의 세 가지 계수로써 나타낼 수 있다. 여기서 지연시간과 시정수는 S자 모양의 응답곡선의 변곡점에서 접선을 그은 다음, 그 접선이 시간축과 만나는 점과 직선 $y(t) = K$와 만나는 점으로부터 각각 구할 수 있다. 이 과정이 [그림 4-4]에 나타나 있다.



[그림 4-4] S자 모양의 응답 곡선

이와 같이 구한 특성계수 K, L, T를 써서 플랜트의 전달함수를 다음과 같이 시간지연을 갖는 1차 시스템으로 근사화할 수 있다.

$$\frac{Y(s)}{U(s)} = K\frac{e^{-Ls}}{Ts+1} \qquad \langle 4\text{-}5 \rangle$$

이 모델은 공조기 시스템과 같은 꼴로서, 반응이 느린 열 시스템들이 이 모델로 표시된다. 지글러-니콜스 조정법의 첫째 방법은 이 근사모델 식 <4-5>에 근거하여 PID 제어기의 계수를 선정하는 것으로서, 이 방법에서 제시하는 PID 계수 K_p, T_i, T_d의 값은 [표 4-1]과 같다. 이 표에 따라 계수를 정하면 PID 제어기의 전달함수는 다음과 같이 된다.

$$\begin{aligned} C(s) &= K_p\left(1+\frac{1}{T_i s}+T_d s\right) \qquad \langle 4\text{-}6 \rangle \\ &= \frac{1.2T}{KL}\left(1+\frac{1}{2Ls}+0.5Ls\right) \\ &= \frac{0.6\,T}{K}\frac{(s+1/L)^2}{s} \end{aligned}$$

이 경우 PID 제어기는 원점에 1개의 극점과 $s=-1/L$에 이중영점을 갖는다. [표 4-1]에 따라 계수를 정한 PID 제어기를 적용하였을 때, 만일 최대초과가 크게 나오는 경우에는 L 값을 더 크게 하여 PID 계수를 조정하면 초과를 적절히 줄일 수 있다. 그 까닭은 L이 커질수록 제어기의 이중영점이 허수축에 가까워지면서 폐로 극점을 허수축에 가까워지게 만들기 때문이다. 그러나 대신에 상승시간은 느려지게 된다.

<표 4-1> 글러-니콜스 계수조정법(첫째 방법)

제어기 종류	K_p	T_i	T_d
P	$\frac{T}{KL}$	∞	0
PI	$\frac{0.9\,T}{KL}$	$\frac{L}{0.3}$	0
PID	$\frac{1.2\,T}{KL}$	$2L$	$0.5L$

이 방법은 시스템에 계단입력을 넣고 출력 곡선을 얻은 다음 이 곡선의 특성계수로부터 PID 계수를 구할 수 있는 아주 간단한 조정법이다. 그러나 이 방법은 1차 주극점으로 근사화 할 수 있는 안정한 플랜트에만 적용할 수 있기 때문에 적용범위가 매우 제한되는 약점이 있다.

2. 지글러-니콜스 조정법(2)

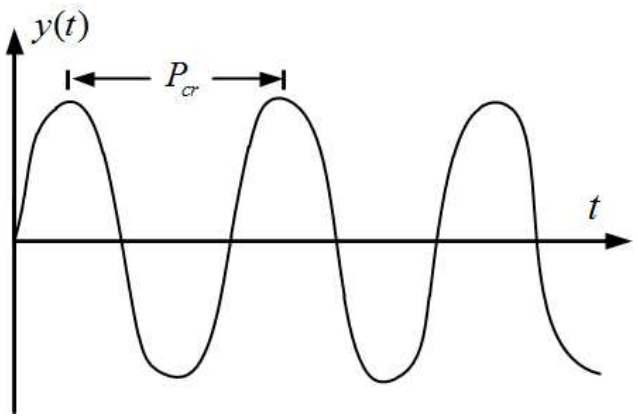
지글러-니콜스 조정법의 둘째 방법은 대상 시스템이 원점에 극점을 갖거나 불안정한 경우에도 적용할 수 있는 방법이다. 이 방법에서는 우선 식 <4-4>)의 PID 제어기 계수 가운데

$T_i = \infty$, $T_d = 0$ 으로 놓고 주어진 플랜트에 비례 제어기만을 적용하여 비례계수 K_p값을 0부터 증가시키면서 출력에 진동이 나타나는 임계이득(critical gain) K_{cr}에까지 이르게 한다. 이 임계이득 K_{cr}은 출력에 [그림 4-5]와 같은 지속진동(sustained oscillation)이 나타날 때의 비례계수 값으로 정의된다. 만약 K_p의 값을 증가시켜도 이러한 진동을 보이지 않는 경우에는 이 방법을 사용할 수 없다. 이때에 기준입력 r은 상수값을 갖도록 하는데 보통 $r(t) = 0$으로 한다.

임계이득 K_{cr}가 구해지면 [그림 4-5]에서 보는 바와 같이, 이에 대응하는 지속진동의 임계주기 P_{cr}을 구할 수 있다. 지글러-니콜스 조정법의 둘째 방법에서는 이 값들을 이용하여 [표 4-2]와 같이 PID 제어기의 계수값을 선정한다. 이 표와 같이 계수를 정하면 제어기의 전달함수는 다음과 같이 된다.

$$\begin{aligned} C(s) &= K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \\ &= 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s\right) \\ &= 0.075K_{cr}P_{cr}\frac{(s + 4/P_{cr})^2}{s} \end{aligned} \qquad \langle 4\text{-}7 \rangle$$

따라서 PID 제어기는 원점에 1개의 극점과 $s = -4/P_{cr}$에 2개의 영점을 갖는다. [표 4-2]에 따라 계수를 정한 PID제 어기를 적용하였을 때, 만일 초과가 크게 나오는 경우에는 임계주기 P_{cr}값을 더 크게 하여 PID 계수를 조정하면, 폐로 극점이 허수축에 가까워지면서 상승시간을 느리게 하는 대신에 초과를 적절히 줄일 수 있다.



[그림 4-5] 지속진동을 나타내는 응답 곡선

<표 4-2> 글러-니콜스 계수조정법(둘째 방법)

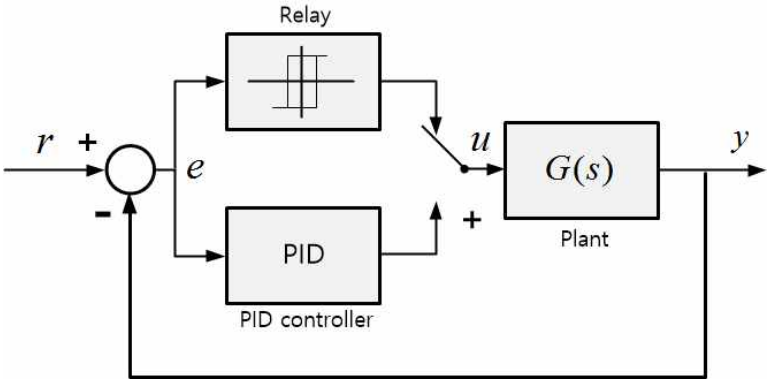
제어기 종류	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

④ 계전기 자동 동조법(relay autotuning)

지글러-니콜스 계수조정법은 많은 경험에 의해 얻어진 방법이기는 하지만, 주로 상승시간과 최대초과를 고려한 조정법으로서 적용대상이 상당히 한정되는 제약성을 지니고 있다. 예를 들면 안정한 복소극점을 주극점으로 갖지만 비례 제어만으로는 임계진동을 하지 않는 시스템의 경우에는 이 방법을 적용할 수 없다. 이러한 한계를 극복하기 위한 방법의 하나로서 제시된 것이 계전기 자동 동조법이다.

이 방법은 1988년에 ÅSTROM과 HAGGLUND가 제시하였으며, 계전기에 의해 출력을 강제로 진동시키면서 출력의 진폭과 주기를 이용하여 PID 계수를 조정하는 방법으로서, 동조과정이 간단하여 동조방식의 구현이 쉽다. 이 방법은 상대안정성의 척도인 위상여유를 고려하여 PID 제어기의 계수를 조정하기 때문에 시스템의 상대안정성을 필요한 만큼 확보하고 견실성을 향상시킬 수 있다. 그리고 대상 시스템이 안정한 경우에는 모두 적용되고 원점에 극점이 있는 경우에도 적용할 수 있어서 앞 절에서 다룬 지글러-니콜스 계수조정법의 한계를 보완할 수 있다. 그러나 이 방법은 극점이 우반면에 있는 불안정한 시스템에는 적용할 수 없다.

이 동조법에서는 [그림 4-6]과 같이 제어기와 병렬로 계전기를 폐로에 추가하여 계전기 되먹임(relay feedback)에 의해 출력을 강제로 진동시켜서 출력이 발진할 때의 한계이득과 한계주기를 결정한 다음, 이 값들로부터 PID 계수를 선정한다. 즉, PID 제어기를 동조하기 위해 먼저 계전기를 동작시키면서 오차 신호가 양이면 계전기 출력을 $+$로, 음이면 $-$로 제어하는 방식으로 출력을 발진시킨다. 이때 기준입력은 $r(t)=0$으로 하거나 어떤 상수값을 갖도록 한다. 그리고 이 진동출력의 계수들로부터 PID 계수가 결정되기 전까지는 PID 제어기를 폐로에 연결하지 않는다. 즉, 동조방식에서는 플랜트가 계전기 되먹임에 먼저 연결이 되고, 동조방식이 끝난 후의 제어방식에서는 PID 제어기가 플랜트에 연결된다.



[그림 4-6] 계전기 자동동조기(relay autotuner)

대부분의 플랜트에 대해서 계전기 되먹임은 임계진동에 가까운 주기를 갖는 진동을 나타나게 한다. [그림 4-6]의 시스템에서 이러한 진동이 나타날 때 계전기의 진폭을 A_r, 출력 진폭을 A_o라 하면, 한계주기 T_u는 바로 진동출력의 주기와 같고, 한계이득 K_u는 다음과 같은 관계식으로부터 구한다.

$$K_u = \frac{4A_r}{\pi A_o} \qquad \langle 4\text{-}8 \rangle$$

계전기 동조법에서는 이렇게 해서 얻어진 한계이득 K_u와 한계주기 T_u를 써서 다음의 간단한 설계공식에 의해 PID 제어기 계수를 선정한다.

$$\begin{aligned} K_p &= K_u \cos\phi_m \\ T_i &= \frac{T_u}{4\pi}\left(\tan\phi_m + \sqrt{1+\tan^2\phi_m}\right) \\ T_d &= \frac{T_i}{4} \end{aligned} \qquad \langle 4\text{-}9 \rangle$$

여기서 ϕ_m은 설계자가 미리 정해주는 위상여유로서 보통 $\pi/6 \le \phi_m \le \pi/3$ [rad] 범위의 값을 사용한다. 이렇게 설계한 PID 제어기를 적용하였을 때에 만일 최대초과가 크게 나오는 경우에는, 위상여유 ϕ_m을 증가시켜 PID 계수를 조정하면 초과를 상당히 줄일 수 있다. 그러나 ϕ_m이 커질수록 응답속도가 느려지므로 적절히 절충시켜야 한다.

안정하거나 극점이 원점에 있는 시스템에서는 계전기 되먹임에 의해 임계진동이 나타나기 때문에 이 설계기법을 적용할 수 있다. 특히, 상대적으로 작은 시간지연을 가지고 있는 공정에 대해서 잘 동작한다. 이 동조법을 실제의 시스템에 적용하려면 [그림 4-6]과 같은 계전기 되먹임 시스템을 구성해야 하지만, 플랜트 모델을 아는 경우에는 간단한 모의실험을 통해 계전기 동조과정을 구현할 수 있다. 다음의 수행 내용을 통하여 계전기 되먹임으로 동조한 다음에 PID 제어기를 통하여 제어하는 과정을 살펴보기로 한다.

5 극배치를 이용한 방법

어떤 시스템의 극점의 위치는 그 시스템의 특성과 직결되는 밀접한 관련을 갖고 있다. 극배치법(pole placement)이란 극점의 위치와 시스템 성능과의 관계를 고려하여 폐로 함수의 극점들의 위치를 적절히 지정하고 이 위치에 폐로 극점이 놓이도록 제어기를 설계함으로써 원하는 성능 목표를 이루는 제어기 설계 방식을 말한다. 만약 시스템이 낮은 차수의 전달함수의 형태로 표현되면, 완전한 극배치 설계 방식이 적용될 수 있다.

다음과 같은 모델로 표현되는 2차 플랜트를 생각해보자.

$$G(s) = \frac{K}{(1+s\,T_1)(1+s\,T_2)}$$

이 모델은 세 개의 계수를 가지고 있다. PID 제어기를 이용하면, PID 제어기 역시 계수가 3개이므로, 폐로 시스템의 3개의 극점을 임의의 자리로 위치시킬 수 있다. PID 제어기의 전달함수 식 <4-4>(8.24)는 다음과 같이 표현할 수 있다.

$$C(s) = \frac{K_p\,(1 + s\,T_i + s^2\,T_i\,T_d)}{s\,T_i}$$

따라서 PID 제어기가 삽입된 폐로 시스템의 특성방정식은 다음과 같이 나타난다.

$$s^3 + s^2\left(\frac{1}{T_1} + \frac{1}{T_2} + \frac{K_p K T_d}{T_1\,T_2}\right) + s\left(\frac{1}{T_1\,T_2} + \frac{K_p K}{T_1\,T_2}\right) + \frac{K_p K}{T_i\,T_1\,T_2} \;=\; 0$$

여기서 원하는 위치에 극점을 갖는 폐로 특성방정식이 다음과 같이 3차의 시스템으로 표현 된다면,

$$(s+\alpha\omega)(s^2 + 2\zeta\omega s + \omega^2) = 0 \qquad \text{<4-10>}$$

위의 두 특성방정식의 계수를 비교하여, PID 계수값을 다음과 같이 정할 수 있다.

$$K_p = \frac{T_1 T_2 \omega^2 (1+2\zeta\alpha) - 1}{K} \qquad \langle 4\text{-}11 \rangle$$

$$T_i = \frac{T_1 T_2 \omega^2 (1+2\zeta\alpha) - 1}{T_1 T_2 \alpha \omega^3}$$

$$T_d = \frac{T_1 T_2 \omega (\alpha+2\zeta) - T_1 - T_2}{T_1 T_2 \omega^2 (1+2\zeta\alpha) - 1}$$

위에서 설명한 극배치를 이용한 PID 제어기 설계 과정을 수행 과정에서 구체적으로 다루어 보기로 한다.

수행 내용 / 제어기 제어 성능 시험하기

재료 · 자료

- 액추에이터 데이터 시트
- 액추에이터 구동 회로도
- 기구 동작 제원서
- 로봇에 대한 사용자 요구 사양서

기기(장비 · 공구)

- 컴퓨터, 프린터, 인터넷
- 프로그램 개발용 소프트웨어
- 문서 작성용 소프트웨어
- 프로그램 성능 평가용 에뮬레이션 소프트웨어
- 로봇 액추에이터 제어 성능 시험 소프트웨어

안전 · 유의 사항

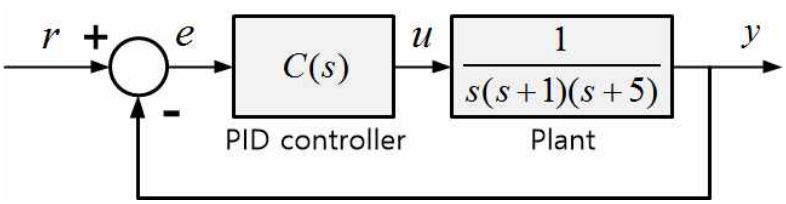
- 시스템 제어 성능 시험 방법 및 분석 결과를 숙지해야 한다.
- 성능 평가 에뮬레이션 소프트웨어 작성 내용을 사전에 숙지해야 한다.

• 시스템 성능 분석서, 시스템 분석 보고서의 내용에 대해 사전에 숙지해야 한다.

수행 순서

① 지글러-니콜스 PID 계수조정법을 구현한다.

[그림 4-7]과 같이 PID 제어기가 사용되는 제어 시스템에서 제어기의 계수를 지글러-니콜스 조정법으로 정해보도록 한다.



[그림 4-7] PID 제어 시스템

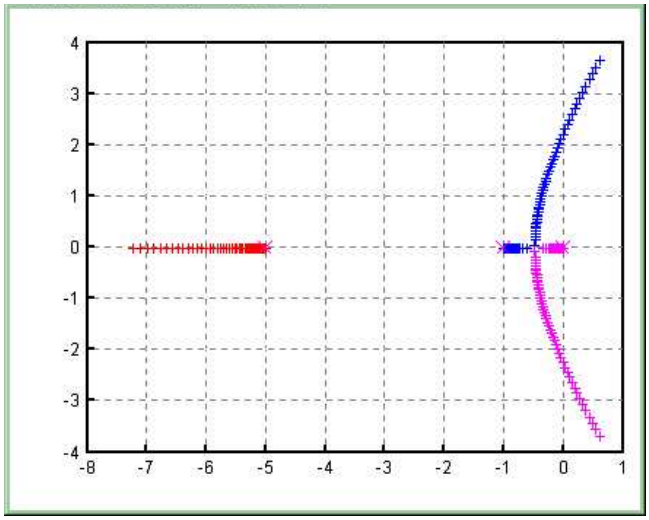
1. 이 문제에서 다루는 제어 대상 플랜트는 적분기를 포함하고 있어서 지글러-니콜스의 첫째 방법을 적용할 수 없으므로 둘째 방법을 사용하기로 한다. 먼저 $T_i = \infty$, $T_d = 0$ 로 놓으면 다음의 폐로 전달함수를 구해본다.

$$\frac{C(s)}{R(s)} = \frac{K_p}{s(s+1)(s+5)+K_p} \qquad \langle 4\text{-}12 \rangle$$

2. 이 시스템이 지속진동을 갖는 계단응답을 나타내기 시작하는 비례계수 K_p의 임계값을 먼저 구해야 한다. 이를 위해서는 [그림 4-7]의 플랜트를 나타내는 개로 전달함수에 대하여 근궤적을 그린 후, 이 궤적이 허수축과 만날 때의 K_p의 값을 구해본다.

3. 이 값이 곧 위에서 설명한 임계이득 K_{cr}이다. 이 과정을 셈툴 명령창에서 다음과 같이 실행한다.

```
CEMTool>>num = 1;
CEMTool>>den = conv([1 0],conv([1 1],[1 5]));
CEMTool>>k = logspace(-2,2,100);
CEMTool>>rlocus(num,den,k);
```



[그림 4-8] K_{cr} 을 구하기 위한 근궤적

4. 위의 결과로 [그림 4-8]의 근궤적이 만들어 본다.

5. 이 근궤적에서 허수축을 지나는 점의 좌표를 구해야 하는데, [그림 4-9]와 같이 추적기능을 이용하면 그 점이 $s = 0 \pm j2.292$ 라는 것을 알 수 있다. 근궤적으로부터 원하는 좌표를 구한 후 그 지점에 해당하는 이득 K_p의 값을 구하기 위해 다음과 같이 셈툴로 입력하여 실행한다.

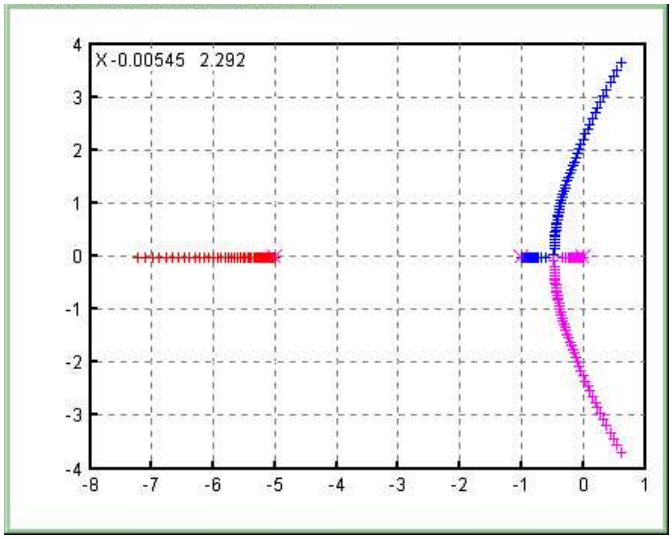
```
CEMTool>>rlocval(num,den,0+2.292*i)
            30.0870
```

6. 이 결과로부터 $K_{cr} = 30.0870$ 임을 확인한다.

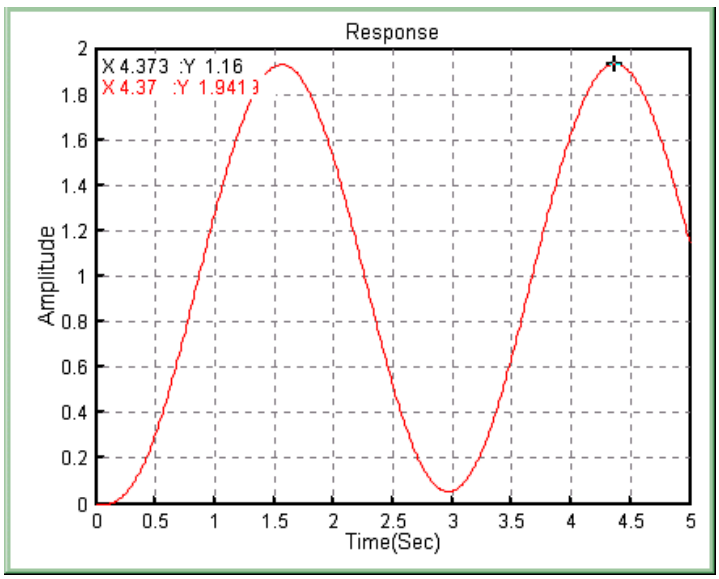
7. 이제 식 <4-12>의 K_p의 값을 $K_p = 30.0870$ 로 대입한 다음에 다음과 같이 계단응답을 구하고 실행결과를 [그림 4-10]과 됨을 확인한다.

```
CEMTool>>num = 30.0870;
CEMTool>>den = [1 6 5 30.0870];
CEMTool>>t = 0:5:0.01;
CEMTool>>step(num,den,t);
```

8. 이 그림에서 추적기능을 이용하여 지속진동이 한 마루에서 이웃하는 마루에 이르는 시간을 구하면 각각 1.57[sec]과 4.37[sec]임을 알 수 있다. 따라서 임계주기는 $P_{cr} = 4.37 - 1.57 = 2.8$ [sec]이 된다.



[그림 4-9] K_{cr} 을 구하기 위한 근궤적(마우스 좌표표시)



[그림 4-10] 지속진동입력에 의한 곡선

9. 위와 같은 과정을 거쳐 $K_{cr} = 30.0870$, $P_{cr} = 2.8$ 의 결과를 얻을 수 있으며, 이를 <표 4-2>에 대입하면 PID 제어기의 계수들은 각각 다음과 같이 구해진다.

$$K_p = 18.0522, \quad T_i = 1.4, \quad T_d = 0.35$$

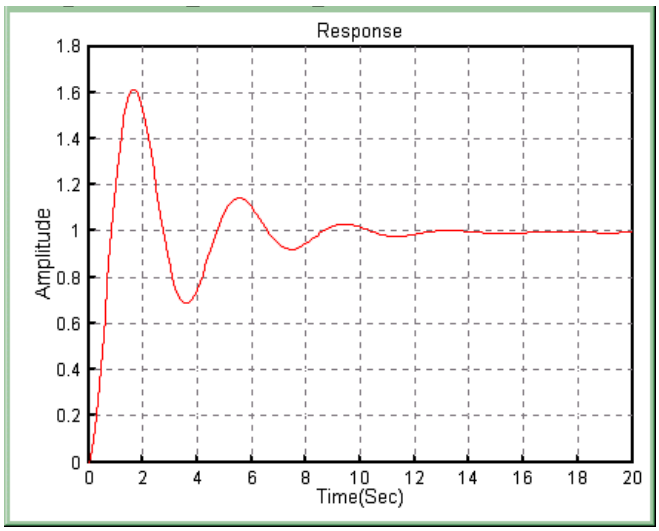
10. 이것을 식 <4-7>에 대입하면 PID 제어기의 전달함수는 다음과 같이 된다.

$$C(s) = \frac{6.3183\,s^2 + 18.0522\,s + 12.8944}{s}$$

11. 이제 이렇게 설계된 제어 시스템의 계단응답을 살펴보기 위해 다음과 같은 셈툴 명령을 수행하고 그 결과가 [그림 4-11]과 같음을 확인한다.

```
CEMTool>>num = [6.3183 18.0522 12.8944];
CEMTool>>den =[1 6 5 0 0];
CEMTool>>[n,d] = cloop(num,den);
CEMTool>>t = 0:30:0.1;
CEMTool>>step(n,d,t);
```

12. 이 그림을 보면 시스템은 단위계단 입력에 대하여 약 62%의 초과를 보이는 것을 알 수 있다. 이를 개선시키기 위해서는 위에서 얻은 PID 계수 값들을 좀 더 조정해야 한다(한 방법으로 임계주기 P_{cr}값을 증가시켜 $P_{cr} = 5$ 로 하여 설계하면 이때의 초과는 약 25%로서 상당히 줄어든다).



[그림 4-11] 결과 시스템의 단위계단응답 곡선

이 절에서 다룬 지글러-니콜스 PID 계수조정법의 둘째 방법은 불안정한 시스템에도 적용할 수 있어서 단위계단응답을 이용한 첫째 방법보다 적용 범위가 넓다. 그러나 이 방법으로 계수조정을 하려면 비례계수를 증가시키면서 임계진동이 나타날 때까지 출력을 계속 관찰하는 과정이 필요하기 때문에 계수조정과정이 첫째 방법보다는 더 복잡한 것을 알 수 있다.

2 계전기 자동 동조법(relay autotuning)을 구현한다.

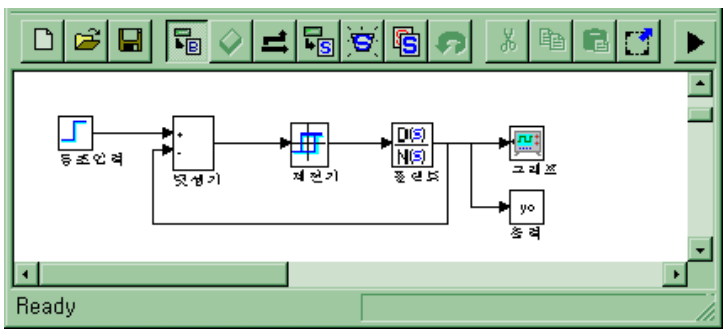
다음의 1차 시스템 $G(s)$에 대한 PID 제어기를 계전기 동조법으로 설계하라. 단, 제어목표는 상승시간 $t_r \leq 0.1$[sec], 정착시간 $t_s \leq 0.6$[sec], 정상상태오차 $E_s = 0$ 이다.

$$G(s) = \frac{1}{s+1}$$

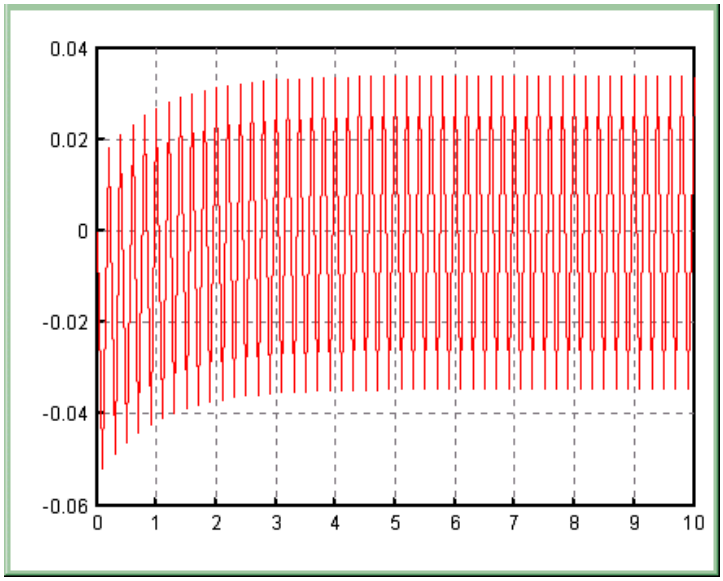
1. [그림 4-12]의 심툴 파일은 위의 1차 시스템 $G(s)$에 계전기를 연결하여 되먹임 시스템을 구성한 것이다. 이 시스템의 출력은 그림표로도 나타나고, 출력 데이터 y_o로 저장되기도 하는데, y_o는 필요한 경우에 셈툴 창에서 사용할 수 있다.

2. [그림 4-12]의 심툴 창에서 플랜트 토막을 두 번 누르기 한 다음에, 나타나는 계수창 안의 Numerator 항목에 1을 입력하고, Denominator 항목에 [1, 1]을 입력한다. 계전기 토막에서는 Input_for_On과 Input_for_Off를 모두 0, Output_when_On을 1, Output_when_Off를 -1로 한다. 그리고 Simulation 항목의 Parameter에서 Start Time을 0, Stop Time을 10, Step Size를 0.1로 설정한 후, Start 메뉴(▶)를 누르면 심툴 파일이 실행되어 개로 응답 데이터를 얻을 수 있다.

3. 이와 같은 절차를 거쳐 [그림 4-12]의 심툴 파일을 실행시키면 [그림 4-13]과 같은 강제진동 출력을 얻을 수 있다. 이 그림창에서 선택 사항의 추적 기능을 이용하여 이 응답의 진폭과 주기를 구하면 진동진폭은 $A_o = 0.033$, 진동주기는0.5로 구해진다. 따라서 한계주기는 $T_u = 0.5$ 가 되고, 식(8.29)로부터 한계이득은 $K_u = 37.44$ 가 된다.



[그림 4-12] 계전기 되먹임 시스템

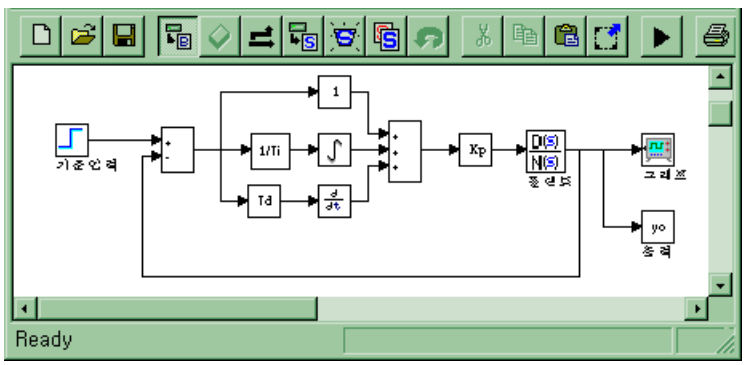


[그림 4-13] 계전기 되먹임에 의한 한계진동

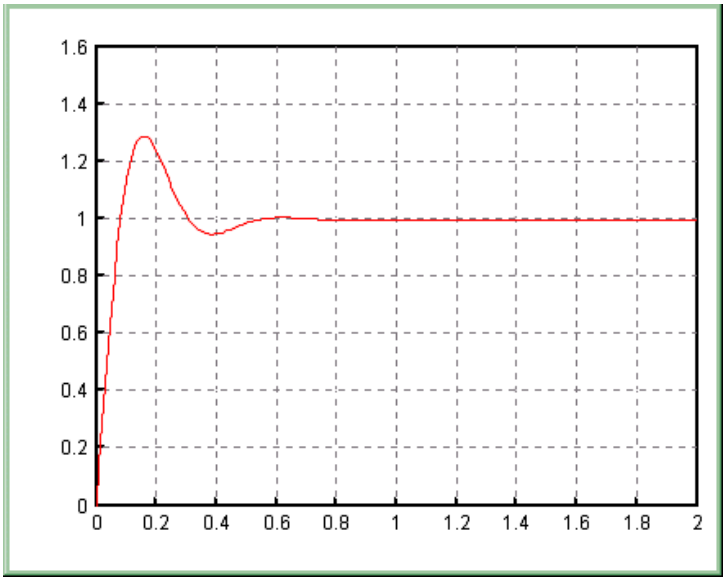
4. 이 결과를 식 <4-9>에 적용하면 PID 제어기 관련 계수를 구할 수 있다. 식 <4-9>로부터 PID 제어기 계수를 얻으려면 먼저 위상여유 ϕ_m을 지정하여 주어야 한다. 여기서는 위상여유를 $\phi_m = \pi/3$ [rad]로 지정하였다. 그러면 PID 계수는 다음과 같이 구해진다.

$$K_p = 18.725, \quad T_i = 0.0594, \quad T_d = 0.0148$$

5. 이렇게 설계된 PID 제어기를 연결한 폐로 시스템을 심툴 파일로 나타내면 [그림 4-14]와 같이 구성할 수 있다. 셈툴 창에서 위와 같은 PID 계수를 입력한 다음에 이 심툴 파일을 실행시키면 단위계단응답을 얻을 수 있는데, 실행 결과는 [그림 4-15]와 같이 출력됨을 관찰한다.



[그림 4-14] PID제어 시스템



[그림 4-15] 계전기를 이용한 PID 동조 제어기의 계단응답

6. 위에서 살펴본 계전기 자동 동조법은 안정성과 견실성을 고려한 설계기법으로서 지글러-니콜스 계수동조법에 비해 적용범위가 넓은 방법이다. 그리고 제어 장치를 컴퓨터로 구성할 경우에 계전기를 실제로 사용하지 않고서도 계전기 되먹임을 무른모로써 처리할 수 있기 때문에 동조 과정을 자동으로 수행하는 자동 동조기 형태로 구현하기가 쉽다. 그러나 이 방법은 대상 시스템이 불안정한 경우에는 적용할 수 없으며, 안정한 경우라 하더라도 동작특성 가운데 시간지연이 클 경우에는 이득여유가 나빠질 가능성이 있으므로 이 점에 유의해야 한다.

3 극배치를 이용한 방법을 구현한다.

대상 시스템의 전달함수가 다음과 같이 주어졌을 때 극배치법을 이용하여 PID 제어기의 계수를 조정하여 성능을 관찰하도록 한다.

$$G(s) = \frac{1}{(1+5s)^2} = \frac{0.04}{s^2+0.4s+0.04} \qquad \langle 4\text{-}13 \rangle$$

1. 이 시스템의 개로 계단응답은 다음과 같은 간단한 명령을 실행하여 결과를 [그림 4-16]과 같이 출력됨을 관찰한다.

```
CEMTool>>step(0.04,[1 0.4 0.04]);
```

2. 실행 결과를 보면 개로 시스템의 정상상태오차는 없으나, 상승시간이 17[sec], 정착시간이 30[sec] 정도로 응답 속도가 상당히 느린 편이다. 이 시스템에 PID 제어기를 적용하여 원하는 성능목표로서 상승시간 2[sec] 이내, 정착시간 10[sec] 이내, 초과 10% 이내로 할 때, 이러한 성능특성을 갖는 극점에 대응하는 폐로 특성방정식 <4-10>)을 모의실험을 통해 구해보면 다음과 같다.

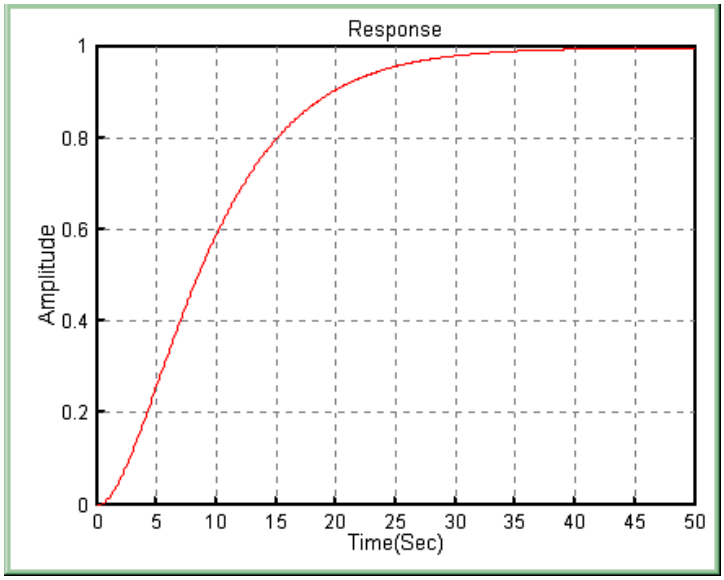
$$(s+2)(s^2+4s+1)=0 \qquad \text{<4-14>}$$

3. 여기서 식 <4-11>을 이용하면, PID 제어기 계수를 찾아낼 수 있다. 식 <4-13>과 <4-14>의 계수들로부터 K, T_1, T_2, ω, α, ζ를 찾아내고, 이 값들과 식 <4-11>로부터 PID 제어기 계수를 구해내는 셈툴 파일을 구현하면 다음과 같다.
이 명령을 수행하면 그 결과로서 Kp = 224, Ti = 4.48, Td = 0.625를 각각 확인하다.

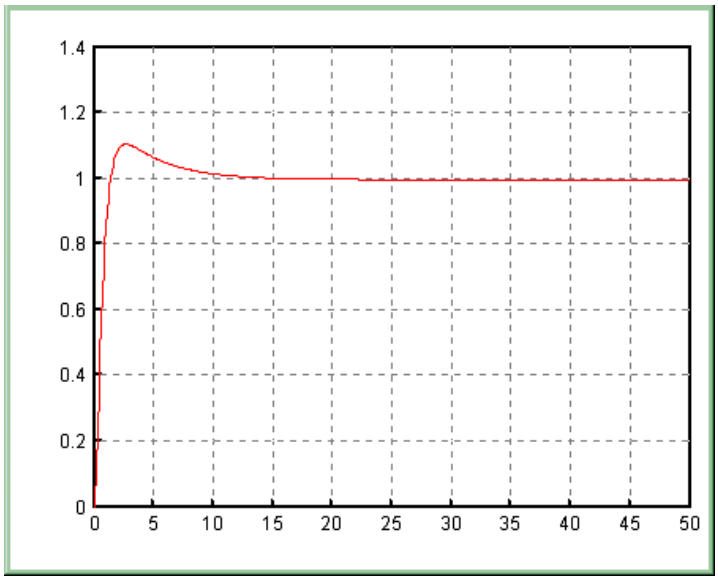
```
T1=5;T2=5;K=1;
w=1;alpha=2;zeta=2;
Kpnum=T1*T2*(w^2)*(1+2*zeta*alpha)-1;
Kp=Kpnum/K
Ti=Kpnum/(T1*T2*alpha*(w^3))
Td=(T1*T2*w*(alpha+2*zeta)-T1-T2)/Kpnum
```

4. 이 계수를 갖는 PID 제어기를 식 <4-13>의 플랜트에 연결하여 폐루프 시스템을 구성하였을 때 단위계단응답을 구해보자. 이 폐로 응답을 구하는 모의실험을 수행하는 심툴 파일은 [그림 4-14]와 같다.

5. 이 심툴 파일을 실행하기 전에 플랜트의 토막에서 Numerator 항에 0.04를, Denominator 항에는 [1, 0.4, 0.04]를 입력하며, 윈도우의 Simulation 항목에서 Parameter를 누르기 하여 나타나는 계수창의 Start Time 항에 0, Stop Time 항에 50, Step Size 항에 0.1을 입력한다. PID 제어기 계수인 Kp, Ti, Td는 위의 셈툴파일을 실행하여 설정되었으므로 Start를 누르기 하면 폐로 응답 데이터와 그림표를 얻을 수 있다. 실행결과는 [그림 4-17]과 같다. 이 그림을 보면 PID 제어기에 의해 상승시간과 정착시간이 아주 빨라졌고 초과가 10% 정도로서 성능목표가 만족되고 있음을 알 수 있다.



[그림 4-16] 개로 시스템 단위계단응답



[그림 4-17] PID 제어기를 사용한 폐로 계단응답 곡선