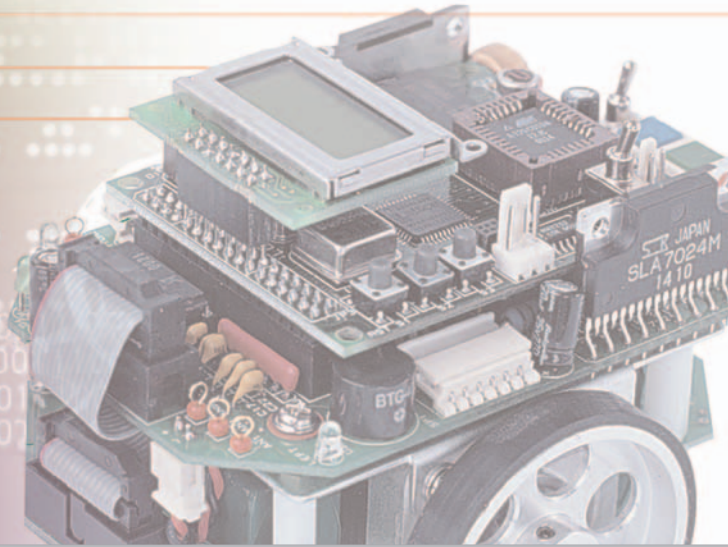
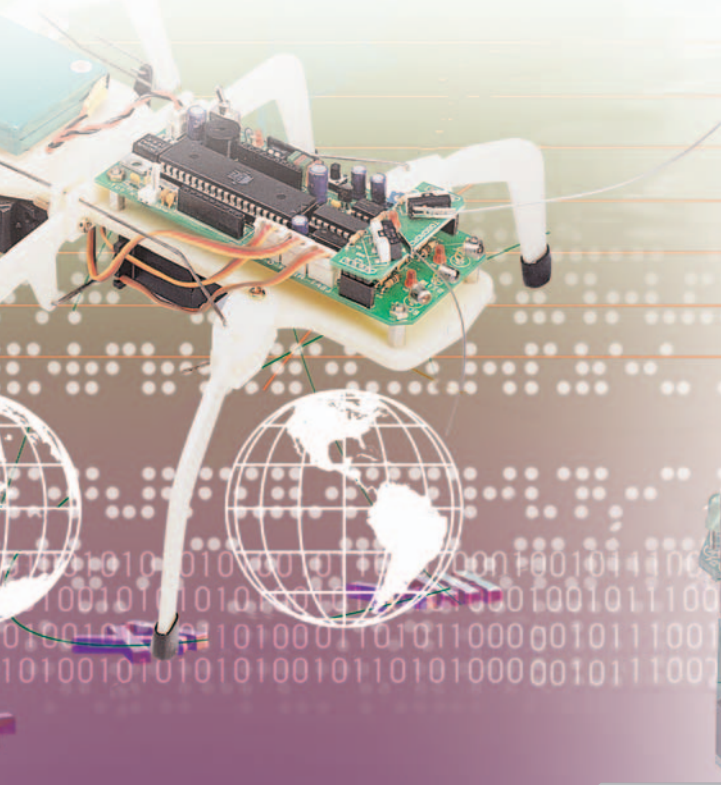


# 로봇 제어 시스템





# 로봇 제어 시스템

## 머리말

**이 교재는** 로봇 및 전자·기계, 자동화 관련 학과에서 로봇의 기본적인 제어를 학습할 수 있도록 엮은 교과서이다.

인간의 새로운 기술에 대한 끊임없는 도전은 삶의 질을 향상시키고 발전시켜 왔다. 로봇 기술은 초기 자동화 생산라인을 중심으로 산업 현장에서부터 적용되기 시작하여 현재에는 단순 공정에서부터 초정밀 반도체 공정까지 없어서는 안 될 중요한 위치를 차지하게 되었다. 로봇 시장이 해마다 고속성장을 계속하고 있고, 우리나라에서도 미래 국가 성장 동력산업의 하나로 선정하여 많은 역량을 쏟고 있다. 따라서 자동화 및 생산 현장에서 로봇을 조작, 운용, 관리할 수 있는 로봇 기초 인력 양성의 필요에 의해 본 교재를 개발하게 되었다.

이 교재는 기본적으로 다음과 같은 특징에 초점을 두고 집필되었다.

- 로봇 제어를 위한 흥미있고 참신한 프로그램
- 로봇의 A to Z를 체험할 수 있는 프로그램
- 예제 및 체험 중심의 프로그램
- 탐구 문제 해결 중심의 프로그램
- 학습자의 창조적 사고 활동을 장려하는 프로그램

이 교재를 통하여 로봇 학습 활동을 하는 학습자가 각 주제별 실습을 통하여 능동적이고 창의적인 학습을 한다면 기대하는 학습 목표를 충분히 달성할 것이다.

교재의 전체 구성은

첫째 마당. 로봇이란 무엇인가?

둘째 마당. 로봇은 무엇으로 구성되는가?

셋째 마당. 로봇을 만들어 보자.

넷째 마당. C언어로 로봇 움직이기

다섯째 마당. 로봇 길 찾기로 구성되어 있다.

로봇 제어의 광범위한 분야를 한 권의 책으로 엮기에는 한계가 있을 수밖에 없다. 그러나 이 교재를 통해서 꾸준한 학습과 연구를 계속 한다면 로봇 제어의 개념에 한걸음 가까이 다가갈 수 있다고 본다.









## I 로봇이란 무엇인가?

1. 로봇의 개념 • 10
2. 로봇의 역사 • 13
3. 로봇의 종류 • 17
4. 로봇 산업의 동향 • 24

## II 로봇은 무엇으로 구성되는가?

1. 로봇의 구성과 원리 • 32
2. 기어의 종류와 쓰임새 • 36
3. 기계 부품의 종류와 쓰임새 • 40
4. 제작 도구의 종류와 쓰임새 • 46

## III 로봇을 만들어보자

1. 89T51로 구동하는 라인트레이서 • 60
2. 라인트레이서 조립하기 • 68

# CONTENTS

## ○ IV C언어로 로봇 움직이기

1. 로봇 두뇌 기초 • 94
2. 로봇에 명령어 심어주기 • 110
3. C언어 기초 • 130
4. C언어로 로봇 움직이기 • 192

## ○ V 로봇! 길을 찾다

1. 모터를 구동해서 로봇 움직이기 • 256
2. LED를 사용해서 적외선 센서 값 확인 • 263
3. 갈림길을 감지하고 길 선택하기 • 266
4. 로봇의 미로 찾기 • 282





ROBOT CONTROL SYSTEM

THE NUMBERS

Energy Prices	0.5%
Food	0.2%
Car Prices	0.9%
Commodities	0.8%

# I **로봇**이란 무엇인가?

1. 로봇의 개념
2. 로봇의 역사
3. 로봇의 종류
4. 로봇 산업의 동향

**사람**과 비슷한 구조를 가지면서 스스로 동작하는 인간 형태의 기계를 로봇이라 한다. 이 단원에서는 로봇의 개념과 역사, 분류 기준에 따른 로봇의 종류, 로봇 산업의 동향에 대해 학습하기로 한다.



# 01. 로봇의 개념

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇의 유래를 알고, 개념에 대해 설명할 수 있다.
- 로봇의 3원칙에 대해 설명할 수 있다.
- 로봇 속의 과학 기술에 대해 설명할 수 있다.

### 1. 로봇의 개요

로봇(Robot)이란 말은 1921년 체코슬로바키아의 극작가 카렐 차פק(Karel Capek)이 발표한 희곡 <로섬의 만능 로봇(R.U.R : Rossum's Universal Robots)>에서 강제 노역을 의미하는 ‘로보타(robota)’에서 처음 유래되었다. 그러나 로봇의 개념은 로봇이라는 말이 탄생하기 이전부터 ‘자동 인형(automata)’, ‘살아 움직이는 인형(animated do it)’ 등의 말로 이미 널리 알려져 있었다.

웹스터(Webster) 사전에서 로봇은 “인간의 일을 대신하는 자동장치 또는 인간 형태의 기계”로 정의되어 있다. 하지만 로봇에 대한 현재의 경향은 “사람과 유사한 구조를 가지고 사람의 명령에 따라 스스로 동작하는 자동화된 기계”라는 일반적인 의미로 사용되고 있다.

산업 혁명을 통해 분업 사회가 되면서 로봇으로 하여금 사람이 하기 힘든 일이나, 위험한 일 등을 대신 처리하도록 함으로써 사람에게 편한 세상을 만들어 주기 위해 이용하게 되었다. 용접 로봇 등 산업 현장에서 주로 이용되



#### 로봇(Robot)

사람과 유사한 구조를 가지고 사람의 명령에 따라 스스로 동작하는 자동화된 기계로서 인조 인간과 같은 의미로 불린다.



#### 카렐 차פק [1890.1.9~1938.12.25]

체코슬로바키아의 극작가, 소설가. 20세기 세계 문학가 중에서 가장 빛나는 대표적 작가 중의 한 사람이다. 대표작으로 사회적 SF의 선구적 작품인 <절대자 제조공장>, <도롱뇽 전쟁>이 있다.



던 로봇은 이제 산업용 로봇, 청소 로봇, 안내 로봇 등 다양한 형태로 사람들의 실생활에 점점 가깝게 다가오고 있으며, 공상 과학 영화에서나 볼 수 있었던 로봇이 현실화 되어가고 있다.

현재 자동차, 컴퓨터, 텔레비전 등이 사람들의 일상에서 빼놓을 수 없는 것들이 되었듯이 로봇도 가까운 시간 안에 1가정 1로봇의 시대가 올 것으로 예상된다.

## 2. 로봇의 3원칙

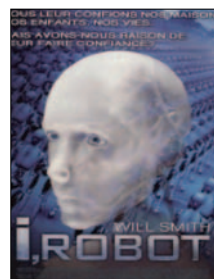
러시아 출생의 미국인 작가인 아이작 아시모프(Isaac Asimov)가 쓴 공상 과학 소설인 <나의 로봇(I, Robot)>에서 인조 인간인 로봇과 인간과의 관계에 있어 로봇이 반드시 지켜야 할 3대 원칙을 제시하였다.

- 제 1원칙 : 로봇은 인간에게 해를 끼쳐서는 안되며, 위협에 처해 있는 인간을 방관해서도 안된다.
- 제 2원칙 : 제 1원칙에 어긋나지 않는 경우 로봇은 인간의 명령에 반드시 복종해야만 한다.
- 제 3원칙 : 제 1원칙, 제 2원칙에 어긋나지 않는 경우 로봇은 자기 자신을 보호해야만 한다.

로봇의 3원칙은 오늘날 개발되어 나오고 있는 모든 로봇에 적용되고 있으며, 미래의 모든 로봇에도 적용되어야 할 것이다.

## 3. 로봇 속의 과학 기술

로봇은 다양한 과학 기술들이 접목되어 탄생한 복합적인 결과물이다. 따라서 로봇에는 우주 항공 분야와 마찬가지로 물리학, 열역학, 동력학, 전자, 제어, 전기, 기계공학, 전산학 등 과학 기술이 총망라되어 있다.



### [ROBOT] 공상과학소설

#### 아이작 아시모프(1920.12~1992.4.6)

미국의 SF 작가, 생화학자, 과학해설자. 전공은 생화학이었으나 천문학, 물리학, 화학, 생물학 등 광범위한 과학 일반에 대하여 뛰어난 해설자로서 유명하다. 또 수많은 SF 작품을 발표하였으며, 대표작으로는 <파운데이션(Foundation)>과 <강철도시(The Caves of Steel)> 등이 있다.



## 로봇 속의 과학 기술

### 물리학/열역학/동역학/화학

로봇의 센서, 전원,  
구동 방식 등을 구현하기 위한  
이론적 배경을 제공

### 전자/전기/제어공학

로봇을 제어하기 위한 전자 회로,  
전기 이론, 제어 이론 등을 통한  
하드웨어 구성 방식을 제공



### 전산학/소프트웨어

로봇의 하드웨어를 제어하고,  
로봇의 구동 알고리즘을  
최적화하여 소프트웨어로  
구현하는 방법을 제공

### 기계공학/산업디자인

로봇을 구동하기 위한  
기계적 구현 방법과 외형의  
표현 방법을 제공

## 02. 로봇의 역사

### ROBOT CONTROL SYSTEM

#### 학습목표



- 로봇의 발전 과정에 대해 설명할 수 있다.
- 로봇의 진화에 대해 설명할 수 있다.
- 로봇의 활용 분야에 대해 설명할 수 있다.

#### 1. 로봇의 태동

로봇은 기원전 3세기 그리스 신화 <아르고 탐색선>에 나오는 청동거인 '탈로스'로부터 그 유래를 찾아 볼 수 있다. 중세에는 시계 등에 부착된 자동인형 등으로 이어지다 20세기 들어 '유니메이트(Unimates)' 라는 산업용 로봇이 만들어지면서 로봇의 시대가 개막되게 되었다.

1999년 일본 소니사에서 출시한 애완견 로봇인 '아이보(AIBO)' 를 시작으로 서비스 로봇이 급속도로 발전하여 현재에는 인간과 거의 유사한 움직임을 가진 2족 직립보행 로봇이 개발되고 있다.

최근에는 국내에서도 청소 로봇을 비롯하여 엔터테인먼트 로봇, 2족 보행 로봇 등을 개발하여 선보이고 있다. 2003년에는 KAIST 의 오준호 박사가 한국형 휴머노이드 로봇인 'KRH-2' 를 만들었으며, 2004년에는 '휴보(Hubo)' 를 완성시켰다. 공상과학 만화영화에서나 보았던 '로봇태권 V' 와 같은 로봇이 현실화되어 가고 있는 것이다.



#### 아이보(AIBO)

1999년 일본 소니사에서 발표한 세계 최초의 본격적인 감성 지능형 완구 로봇 애완견이다. 인공 지능을 뜻하는 'AI' 와 로봇의 'BO' 를 뽑아 만든 합성어이다. 아이보는 애완동물을 대체하는 개념으로 시작되었으며, 현재는 인간과 친구가 될 수 있는 하나의 새로운 개체로서 인정 받고 있다. 6가지 감성(기쁨, 슬픔, 성냄, 놀람, 공포, 혐오)과 4가지 본능(성애욕, 탐색욕, 운동욕, 충전욕)이 구현되어 있으며, 외부의 자극과 자신의 행동으로 인하여 감성과 본능 수치가 항상 변화한다.

[표 1-1] 로봇의 발전 과정

1921년	로봇이라는 용어가 처음 등장
1942년	로봇의 행동 원칙을 규정한 로봇의 3원칙 발표
1956년	조지 데볼과 조셉 앵겔버거가 최초의 로봇 회사인 유니메이션(Unimation) 설립
1962년	제너럴 모터스(GM)사 뉴저지 공장 생산라인에 최초의 산업용 로봇 <유니메이트(Unimate)>배치
1969년	스탠퍼드 대학 빅터 샤인먼이 최초의 로봇팔인 스탠퍼드 암(Stanford Arm) 제작
1970년	스탠퍼드 연구소가 최초의 인공지능 이동 로봇 세이키 제작
1973년	신시내티 마이크로론이 마이크로컴퓨터에 의해 작동하는 최초의 산업용 로봇인 T3 출시 이탈리아에서 제1회 로봇 회의 개최
1976년	무인 우주 탐사선인 바이킹 1, 2호 탐사 작업에 로봇팔 사용
1994년	카네기멜론 대학 로봇 연구소가 제작한 단테 2 탐사 로봇이 알래스카 스퍼산에서 화산재 등 탐사
1997년	나사 화성 탐사선 패스파인더 화성에 착륙 - 탐사 로봇인 <소저너 로버>가 화성 탐사에 나서 화성 및 다른 행성들의 사진을 지구로 전송
1997년	혼다사에서 인간처럼 걷고 계단을 오르는 휴머노이드 로봇 <P3> 공개 한국과학기술연구원에서 한국 최초 보행 로봇 <센토> 제작
1999년	소니사에서 애완견 로봇 <아이보(AIBO)> 출시
2000년	혼다사에서 차세대 휴머노이드 로봇 <아시모> 개발
2001년	소니사에서 애완견 로봇 아이보 2세대 출시 한국과학기술원에서 감정표현이 가능한 로봇 아미 개발
2002년	카네기 멜론이 제작한 사회성 로봇 <그레이스(Grace)> 등장
2004년	한국과학기술원에서 휴머노이드 로봇 <휴보(HUBO)> 개발



## 2. 로봇의 진화

미국의 로봇 공학자 한스 모라벡(Hans Moravec)은 로봇을 이루고 있는 마이크로프로세서의 처리 능력과 메모리 용량, 소프트웨어 기술 등에 따라 향후 10년을 주기로 하여 세대가 바뀔 정도로 지능이 향상될 것으로 전망하고 있다.



[표 1-2] 세대별 로봇의 진화

세 대	년 도	처리 속도	지능 수준	특 징
제1세대	2000년대	3000 MIPS	도마뱀	혼자서 길을 찾을 수 있고, 청소 및 배달은 물론 공장에서 더욱 발전된 작업을 수행할 수 있다.
제2세대	2010년대	10만 MIPS	생쥐	학습을 통해 일을 배울 수 있고, 스스로 수행 능력을 개선할 수 있다.
제3세대	2020년대	300만 MIPS	원숭이	각 사물의 용도에 대해 이해하게 되고, 생명체를 구분할 수 있다. 새로운 작업에 대해 모의 실험을 통해 준비를 하게 되고, 주위 사람들의 기분을 파악할 수 있다.
제4세대	2030년대	1억 MIPS	인간	인간처럼 말을 할 수 있고 이해할 수 있게 되며 독창적 사고가 가능하다. 자신의 행위 결과를 예측할 수도 있다. 인간과 같은 수준 혹은 그 이상의 추론 능력을 지닌다.
제5세대	2040년대 이후	100억 MIPS	인간 초월	로봇은 소프트웨어로 만든 인류의 정신적 유산, 이를테면 지식·문화·가치관을 모두 물려받게 된다. 이때의 로봇을 지혜를 가진 로봇이라 하여 '로보사피언스(Robosapiens)'라 한다.



### 3. 로봇의 활용

1960년대 초부터 산업 현장에서 사용되던 산업용 로봇은 위험한 환경에서 인간 대신 작업을 수행하는 용도로 활용되었다. 전통적으로 산업용 로봇이 가장 많이 활용되고 있는 분야는 자동차 생산 공정과 전자제품 조립 공정이다. 특히 100kg에 육박하는 공구로 작업하는 자동차 공장의 용접 공정은 로봇이 가장 먼저 활용되었으며, 지금도 가장 많이 활용하고 있는 분야이다. 전자제품 공장에서는 작은 부품을 정해진 위치에 조립하는 정밀 작업을 오차없이 사람보다 훨씬 빠른 속도로 수행한다.

사람의 지시에 따라 단지 반복 작업을 수행하던 초기의 산업용 로봇은 인공 지능을 포함한 소프트웨어와 센서 기술의 발전과 더불어 점차 지능화되고 있는 추세이다.



#### MIPS

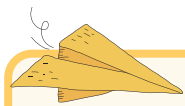
Million Instructions Per Second의 약어로 초당 100만 개의 명령어를 실행하는 능력을 의미한다.

이러한 흐름에도 불구하고 산업용 로봇의 시장을 확대하는 데에는 한계가 있었다.

1990년대에 들어서면서 산업용 로봇 외에 비제조업용 로봇, 즉 서비스 로봇에 대한 연구가 활발히 진행되었다. 초기의 서비스 로봇은 서비스업과 관련된 산업용으로 활용되었으나 개인이나 가정을 위한 여러 분야들로 그 활용이 확대되고 있다.

[표 1-3] 로봇 활용 분야

제조업 분야		용접 시스템, 도장 시스템, 연마 및 연삭 시스템, 화물 입·출입 시스템, 조립 시스템
비제조업 분야		농업 분야(6족 임업용 로봇, 자동 모내기 로봇) 축산 분야(우유 짜는 로봇)
서비스 분야	생활 분야	가정 자동화 분야(경비 로봇, 청소 로봇, 커뮤니케이션 로봇) 엔터테인먼트 분야(팻 로봇, 경기용 로봇, 탑승 로봇) 다목적용(연구용 로봇, 교육용 로봇) 업소용(안내 로봇, 서빙 로봇)
	의료 및 복지 분야	의료 분야(내시경 수술 지원, 원격 수술) 복지 분야(운송 로봇, 심리 치료 로봇, 식사 도우미 로봇, 간호 지원 로봇, 로봇 휠체어)
	공공 분야	재해 구조 로봇, 군사용 로봇, 해양 로봇, 원자력 로봇, 우주 로봇, 건설 로봇, 항공 로봇



### 쉬어가기

#### 로봇의 지능은 어느 정도일까?

로봇의 지능은 얼마나 많은 정보를 수록하고 이를 빠르게 행하는가에 달려 있다. 이러한 것은 로봇의 머리에 있는 컴퓨터가 수행하기 때문이다. 컴퓨터의 성능이 좋으면 당연히 지능이 높은 것으로 취급된다. 그러나 컴퓨터의 능력을 지능으로만 판단할 수 있을까? 특히 사람에게 적용된 지능으로 표시한다는 것은 무리이다. 사람의 지능이란 한계가 있지만 컴퓨터는 계속 그 성능이 발전해 가기 때문이다.

## 03. 로봇의 종류

### ROBOT CONTROL SYSTEM

#### 학습목표



- 산업용 로봇에 대해 알고, 종류별 특징에 대해 설명할 수 있다.
- 서비스 로봇에 대해 알고, 종류별 특징에 대해 설명할 수 있다.
- 로봇 산업의 발전 전망에 대해 설명할 수 있다.

국제로봇연맹(IFR)에 의한 로봇의 일반적인 분류는 산업용 로봇과 비산업용 로봇으로 구분된다. 산업용 로봇은 말 그대로 산업 현장에서 적용되어 제품(Products) 생산, 관리 등의 업무를 수행하는 로봇이며, 비산업용 로봇은 산업용 로봇을 제외한 나머지 로봇을 통칭한다.

산업용 로봇은 다시 자동차, 전기, 전자 분야의 공장 자동화에 활용되는 제조업 부문과 농업, 임업 등 1차 산업과 건설업 분야의 현장 자동화에 활용되는 비제조업 부문으로 구분된다.

서비스 로봇은 제조 작업을 제외한 인간과 장비에 유용한 서비스를 제공하기 위해 자동으로 작동하는 로봇을 말한다. 따라서 산업용 로봇이 비제조업 분야에 사용된다면 이 역시 서비스 로봇의 범주에 포함된다. 서비스 로봇은 비제조업 부문에 이용되는 서비스 부문에서 시작되어 가사, 간병, 안내, 도우미 등 우리 생활을 보다 편리하게 하는 영역으로 발전하고 있다. 이렇듯 공공의 영역에서 활용되던 서비스 로봇은 이제 개인의 삶의 질을 향상시키기 위한 개인용 로봇 중심으로 발전하고 있다.



#### 산업용 로봇(industrial robot)

로봇의 한 종류로서 인간을 대신하여 작업 현장에서 노동을 행하는 기계

[표 1-4] 용도에 따른 로봇 분류

구분	용도	대표적인 작업	
산업용 로봇	제조업	용접, 조립, 반송, 도장	
	비제조업	농업, 임업, 건설업	
비산업용 로봇	서비스 로봇	가사용	청소, 심부름, 경비, 조리, 제조
		생활 지원용	간병, 재활
	여가 지원용	음악 연습, 게임, 스포츠	
	공공 복지용	경비, 안내, 의료, 도우미	

## 1. 산업용 로봇

### (1) 산업용 로봇이란?

산업용 로봇은 인간을 대신하여 작업 현장에서 노동을 행하는 기계로, 주로 사람의 손과 팔의 동작을 하는 매니퓰레이터(Manipulator) 형태이다.

산업용 로봇은 1960년대 초 제너럴 모터스(GM) 사 뉴저지 공장 생산라인에 최초의 산업용 로봇 유니메이트(Unimate)가 설치되면서 시작되었다. 로봇은 소음, 온도, 위험한 환경에서 인간 작업자에게 악영향을 주는 피로감, 불편함, 지루함 등에 민감하지 않을 뿐만 아니라 더럽고 반복적인 작업이 필요한 경우에도 작업을 잘 수행할 수 있다는 장점이 있어 전자 산업, 자동차 산업 등에 주로 사용되고 있으나 앞으로는 인간의 지적 노동까지 담당하게 될 것으로 기대하고 있다.

한국에서는 1980년대 초 자동차 산업의 급속한 발전과 함께 산업용 로봇이 보급되기 시작하여, 현재는 자동차 산업뿐만 아니라 전기·전자 산업의 대량 생산 설비 중 용접용, 조립용, 도장용 등에 주로 사용되고 있다.

### (2) 산업용 로봇의 분류

산업용 로봇은 분류 기준에 따라 일반적 분류, 제어 방법에 의한 분류, 동작특성에 따른 분류 등으로 나눌 수 있다.





조립용 로봇



도색용 로봇



용접용 로봇

[그림 1-1] 산업용 로봇 활용의 실제

### 1) 일반적 분류

로봇이 보유하고 있는 기능에 따라 분류한 것이다.

#### ① 조종 로봇(Operating Robot)

로봇이 해야 할 작업의 일부나 전체를 인간이 직접 조작함으로써 작업을 할 수 있는 로봇이다.

#### ② 시퀀스 로봇(Sequence Control Robot)

순서, 조건, 위치 등과 같은 미리 설정된 정보에 따라 동작의 각 단계를 차례대로 진행해 가는 로봇이다.

#### ③ 플레이백 로봇(Playback Robot)

인간이 로봇을 작동시킴으로써, 순서, 조건, 위치 및 그 밖의 정보를 가르쳐 주면 그 정보에 따라 작업을 할 수 있는 로봇이다.

#### ④ 수치 제어 로봇(Numerically Controlled Robot)

로봇을 작동시키지 않고 순서, 조건, 위치 및 그 밖의 정보를 수치, 언어 등으로 가르쳐 주면 그 정보에 따라 작업할 수 있는 로봇이다.

#### ⑤ 지능 로봇(Intelligent Robot)

인식 능력, 학습 능력, 추상적 사고 능력, 환경 적응 능력 등을 인공적으로 실현한 인공 지능에 의해 행동을 스스로 결정할 수 있는 기능을 가진 로봇이다.



#### 서보제어

출력(위치, 회전수, 각도 등)을 되먹임시켜 원하는 동작을 하게 하는 장치

## 2) 제어 방식에 의한 분류

산업용 로봇을 제어하는 방식에 따라 분류한 것이다.

### ① 서보 제어 로봇(Servo Controlled Robot)

정확한 위치 또는 방향을 제어하기 위하여 서보 모터와 같은 서보 제어 장치를 사용하여 로봇의 동작을 제어하는 로봇이다. 설치하는 비용이 높고 유지 보수에 높은 기술이 필요하나 유연한 프로그램 제어를 통해 복잡한 제조 작업을 수행하는 데 적합한 로봇이다.

### ② 비서보 제어 로봇(Non-servo Controlled Robot)

로봇의 동작을 공압 실린더와 같은 비서보 장치를 사용하여 수행하는 로봇이다. 속도를 제어할 수 없고 낮은 정확도를 나타내지만, 설치하는 데 드는 비용이 저렴하고 조작이 편리하여 비교적 간단한 작업을 수행하는 데 적합한 로봇이다.

### ③ 경로 결정 제어 로봇(Continuous Path Controlled Robot)

로봇이 정해진 위치로부터 다음 위치로 이동할 때 로봇의 이동 경로가 경로 보간에 의하여 생성되고, 이를 추종함으로써 로봇의 이동 경로가 제어되는 로봇이다.

### ④ 위치 결정 제어 로봇(Point-to-point Controlled Robot)

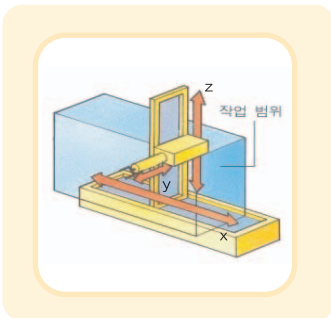
로봇이 정해진 위치로부터 다음 위치로 이동할 때 로봇의 이동 경로와 무관하게 지정된 작업 위치를 찾아가도록 제어되는 로봇이다.

## 3) 동작 특성에 따른 분류

산업용 로봇을 구조적 동작 특성에 따라 분류한 것으로, 대부분 산업용 로봇이 아래와 같은 5개의 디자인 형상들을 가지고 있다.

### ① 직교 좌표 로봇

직교 좌표 로봇(Cartesian Coordinate Robot)은 직선 운동을 하는 X축, Y축, Z축으로 구성된 3개의 슬라이드를 사용하나 경우에 따라서는 1개 또는 2개의 축만을 사용하기도 한다. 직교 좌표 로봇은 3개의 슬라이드를 서로 움직임으로써 발생하는 직교형의 작업 영역 내에서 동작할 수 있다.





## ② 원통 좌표 로봇

원통 좌표 로봇(Cylindrical Coordinate Robot)은 직선 운동을 하는 Y축, Z축으로 구성된 두 개의 슬라이드와 X축인 하나의 회전축으로 구성된 로봇이다. 로봇은 회전축을 회전함으로써 원통형의 작업 영역 내에서 동작할 수 있다.

## ③ 구 좌표 로봇

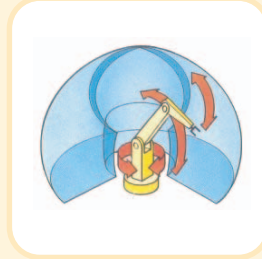
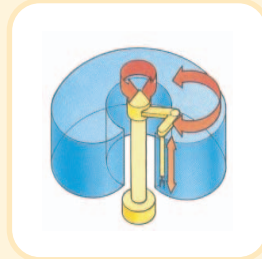
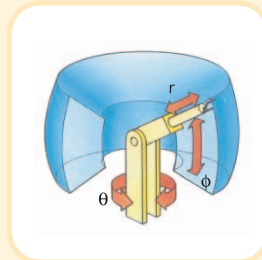
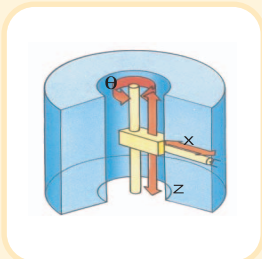
구 좌표 로봇(Spherical Coordinate Robot)은 직선 운동을 하는 X축 슬라이드와 Y축, Z축으로 구성된 두 개의 회전축으로 구성된 로봇이다. 구 좌표 로봇은 작업 영역이 넓고 경사진 위치에서 작업을 수행할 수 있어 용접 작업이나 도장 작업에 적합하다.

## ④ 수평 다관절 로봇

스카라 로봇(Selective Compliance Assembly Robot Arm, SCARA)으로 불리는 수평 다관절 로봇(Horizontal articulated Robot)은 어깨와 팔꿈치 관절들이 수평축 대신 수직축에 대하여 회전하는 관절 로봇의 특별한 형태의 로봇으로 비교적 큰 원통형의 작업 영역을 가진다.

## ⑤ 수직 다관절 로봇

수직 다관절 로봇(Vertical articulated Robot)은 관절의 구동이 3개의 회전축에 의해 움직이는 구조로 된 로봇이다. 수직 다관절 로봇은 인간의 팔의 형상과 유사한 형태로 되어 있어 불규칙한 작업 영역을 가진다.



## 2. 지능형 서비스 로봇

### (1) 서비스 로봇이란?

국제로봇연맹(International Federation of Robotics, IFR)은 서비스 로봇을 “제조 작업을 제외하고 인간과 장비에 유용한 서비스를 제공하기 위해 반자동 또는 완전 자동으로 작동하는 로봇”이라고 정의하고 있다. 산업용 로봇과는 달리 서비스 로봇은 사람과 동일한 공간에서 함께 생활하면서 사람이 필요로 하는 서비스를 제공하는 인간 공존형 로봇이다.

서비스 로봇은 사람의 지시에 따라 피동적으로 반복 작업을 수행하던 전통적인 서비스 로봇에서 외부 환경을 인식하고 스스로 상황을 판단하여 자율적으로 동작하는 지능형 서비스 로봇으로 급격히 변화하고 있다.

### 1) 지능형 서비스 로봇의 분류

지능형 서비스 로봇은 크게 개인 서비스 로봇과 전문 서비스 로봇으로 분류된다.

#### ① 개인 서비스 로봇

개인 서비스 로봇은 인간 생활 범주에서 제반 서비스를 제공하는 인간 공존형 대인지원 서비스 로봇으로 가사용 로봇, 생활 지원용 로봇, 여가 지원용 로봇, 교육용 로봇 등이 있다.

가사용 로봇	→	세탁 로봇, 청소 로봇, 잔디 깎기 로봇 등과 같이 가사 노동을 대신하여 주는 로봇
생활 지원용 로봇	→	재활 로봇, 이동 로봇 등과 같이 고령자나 환자, 장애인들의 생활을 도와주는 로봇
여가 지원용 로봇	→	애완용 로봇, 경기용 로봇, 오락 로봇 등과 같이 인간의 여가 생활 지원과 감정적 동반자의 역할을 수행하는 로봇
교육용 로봇	→	가정교사 로봇과 같이 학교나 가정에서 교육이나 훈련을 위한 보조 도구로 활용되는 로봇



동화구현로봇 D2E-EMR(한국)



휠체어로봇 Wheelesley(미국)



강아지로봇 i-ROBO(미국)

[그림 1-2] 개인 서비스 로봇 활용의 실제



## ② 전문 서비스 로봇

전문 서비스 로봇은 불특정 다수를 위한 서비스를 제공하고 전문화된 작업을 수행하는 로봇으로 군사용 로봇, 의료복지용 로봇, 극한 작업용 로봇, 빌딩서비스용 로봇 등이 있다.

군사용 로봇	→	정찰 로봇, 경계 로봇, 위험물 탐지 및 제거 로봇 등과 같이 군사적 목적으로 사용되는 로봇
의료복지용 로봇	→	수술 로봇, 자동 검사 로봇, 내시경 로봇, 간호 보조 로봇 등과 같이 질병을 진단하고 수술을 직접 하거나 보조하는 로봇
극한 작업용 로봇	→	인명 구조 로봇, 탐사 로봇 등과 같이 사람이 직접 하기 힘든 여건에서 작업을 수행할 수 있는 로봇
빌딩 서비스용 로봇	→	안내 로봇, 보안/경비 로봇 등과 같이 하나의 건물을 통합하고 관리하여 주는 서비스를 제공하는 로봇



안내 로봇 IGURO(한국)



간호 로봇 Pearl(미국)



화성 탐사 로봇 스피릿(미국)



정찰 로봇 ARV(미국)



경비 로봇 D1(일본)



휴머노이드 로봇 MAHRU & AHRA

[그림 1-3] 전문 서비스 로봇 활용의 실제

# 04. 로봇 산업의 동향

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇 산업의 동향에 대해 설명할 수 있다.
- 미래 로봇 산업의 전망에 대해 설명할 수 있다.
- 지능형 로봇과 관련된 기술에 대해 설명할 수 있다.



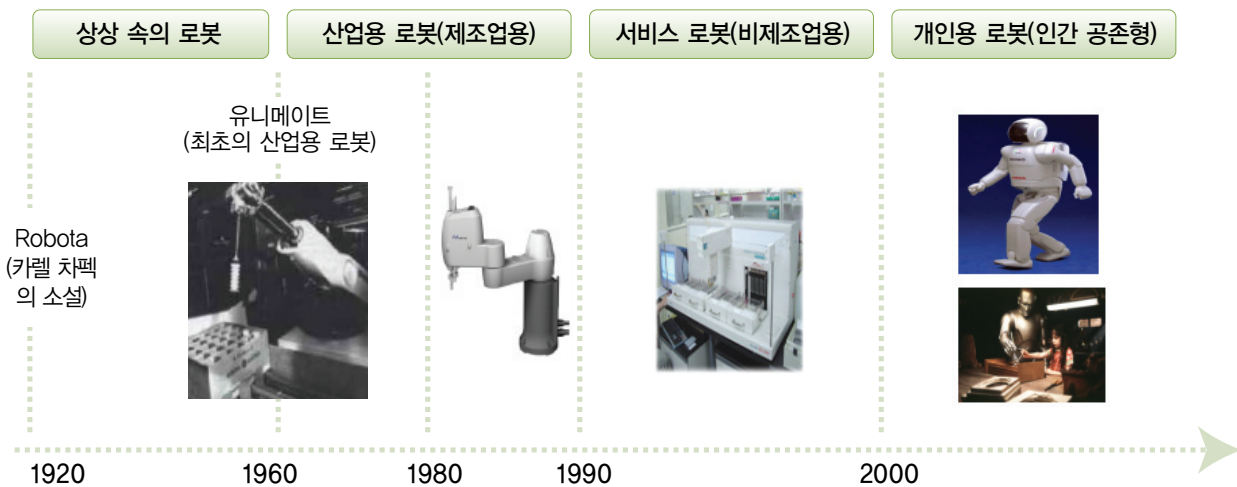
### 1. 로봇 산업의 동향

1960년대 초 최초의 로봇 ‘유니메이트(Unimate)’가 개발되면서 로봇 산업이 시작되었다. 그러나 로봇이 본격적으로 연구되기 시작한 것은 이로부터 20여 년이 지난 1980년대에 일본이 대량 생산을 위한 다수의 산업용 로봇을 만들어 내면서부터이다. 산업용 로봇 시대는 로봇으로 하여금 인간을 대신해 단순하고 반복적인 작업을 쉬지 않고 계속적으로 수행하게 함으로써 제품의 생산 단가를 낮추고 균일한 제품의 생산이 가능하게 하였다.

그러나 1986년을 기점으로 각국에서 로봇 시장이 침체되기 시작해 많은 로봇 업체들이 도산했다. 로봇 시장은 1988년에 되살아나 1990년에는 세계의 로봇 판매가 총 8만 1000대에 이르렀다.

1990년대 후반부터 산업용 로봇 시장이 점차 포화 상태에 접어들면서 인간과 같이 지능적으로 판단하고 행동하는 인간 공존형 지능형 서비스 로봇 시장이 빠르게 형성되었다. 1999년 일본의 소니사에서 개발한 장난감 강아지 로봇인 ‘아이보’를 시작으로 개인용 로봇 시대가 열리게 되었다.

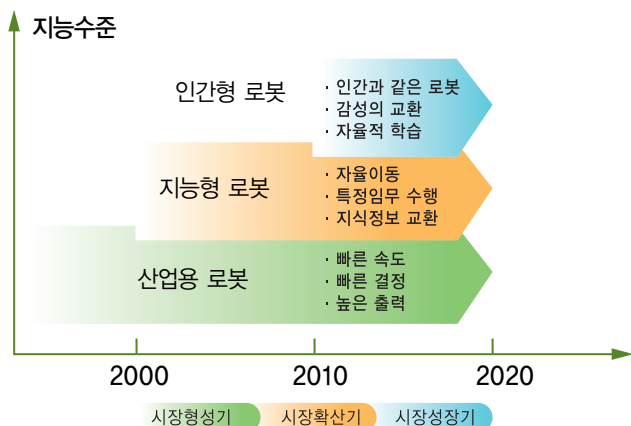




[그림 1-4] 로봇 산업의 변화

## 2. 로봇 산업의 전망

초기 로봇 시장의 성장은 제조 부문이 주도해 왔다. 하지만 많은 전문가들은 2000년대 로봇 시장의 중심은 비제조업 부문, 특히 서비스 부문이 될 것으로 보고 있다. 2000년대는 지능형 로봇의 시장 형성기로 개인용 로봇이 보급되기 시작하였고, 애완용 로봇과 청소 로봇을 중심으로 200억 불대의 시장



[그림 1-5] 로봇 산업의 전망





**생체 공학 기술(Bio Technology)**  
 생물체의 유용한 특성을 이용해서 여러 가지 공업적 공정, 공업적 규모로 이루어지는 생화학적 공정을 의미하며, 생명공학 또는 BT라 한다.

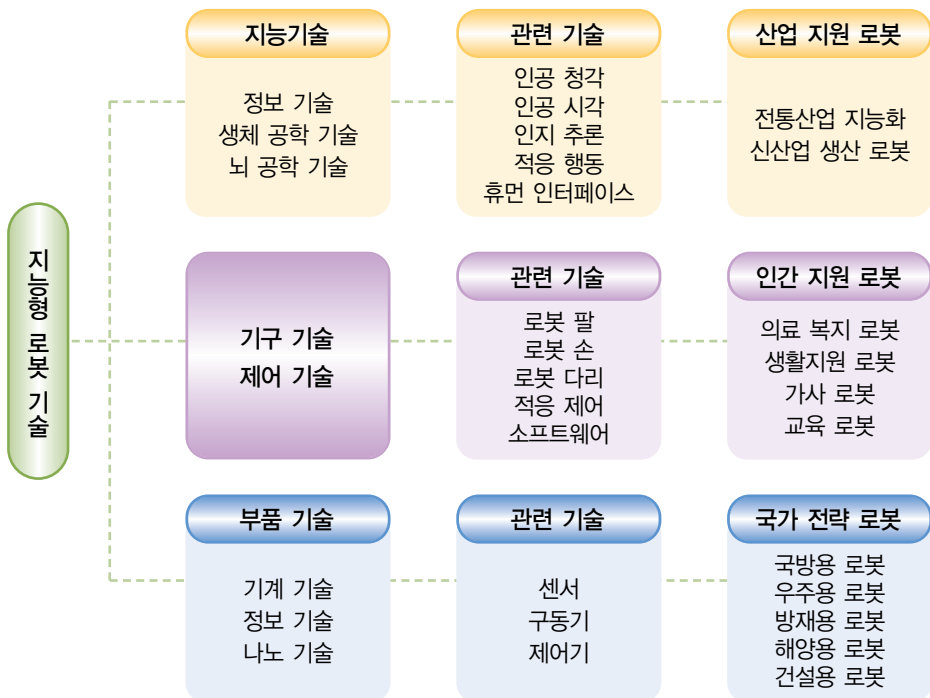


**나노기술(Nano Technology)**  
 10억 분의 1 수준의 정밀도를 요구하는 극미세 가공 과학기술이다. 1 나노미터(nm)는 10억 분의 1[m]로서 사람 머리카락 굵기의 10만 분의 1로서 대략 원자 3~4개의 크기에 해당한다.

이 형성되었다. 2010년에는 시장의 확산기로 서비스 로봇과 의료 복지용 로봇이 확산되어 1,500억 불대의 로봇이 형성될 것으로 보이며, 시장 성장기인 2020년경에는 극한 지역에서 작업하는 필드 로봇과 서비스 로봇의 양산으로 5,000억 불대에 이르는 1가구 1로봇의 시대가 다가올 것으로 전망된다.

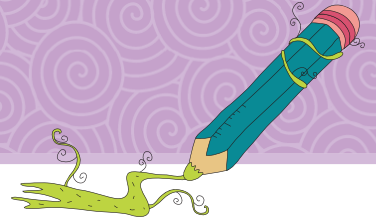
### 3. 지능형 로봇 기술

지능형 로봇 기술의 흐름은 현재의 반도체 기술, 전자 기술 및 정보 통신 기술의 성과를 계승하고 있다. 또한 인공 지능 기술, 뇌 공학 기술, 초소형 기계 및 전자 기술, 생체 공학 기술(Bio Technology), 나노 기술(Nano Technology) 등의 신기술을 접목하여 21세기 인류의 삶을 급진적으로 변화시키기 위한 높은 생산력 수준을 대표하는 도구로 발전하고 있다.



[그림 1-6] 지능형 로봇 기술 계통도

# 단원 학습 정리



01. 로봇이란 일반적으로 “인간의 일을 대신하는 자동장치 또는 인간 형태의 기계”라 할 수 있다.
02. 로봇에는 물리학, 열역학, 동력학, 전자, 제어, 전기, 기계공학, 전산학 등 과학 기술 등이 총망라되어 있다.
03. 로봇은 일반적으로 산업용 로봇과 비산업용 로봇으로 분류되며, 산업용 로봇은 다시 제조업 부문과 비제조업 부문으로 구분된다.
04. 산업용 로봇은 인간을 대신하여 작업 현장에서 노동을 행하는 기계로, 주로 사람의 손과 팔의 동작을 하는 매니퓰레이터(Manipulator) 형태이다.
05. 산업용 로봇은 일반적으로 조종 로봇, 시퀀스 로봇, 플레이백 로봇, 수치 제어 로봇, 지능 로봇으로 분류할 수 있다.
06. 산업용 로봇은 제어 방식에 따라 서보 제어 로봇, 비서보 제어 로봇, 경로 결정 제어 로봇, 위치 결정 제어 로봇으로 분류할 수 있다.
07. 산업용 로봇은 동작 특성에 따라 직교 좌표 로봇, 원통 좌표 로봇, 구 좌표 로봇, 수평 다관절(스카라) 로봇, 수직 다관절 로봇으로 분류할 수 있다.
08. 서비스 로봇은 일반적으로 “제조 작업을 제외하고 인간과 장비에 유용한 서비스를 제공하기 위해 반자동 또는 완전 자동으로 작동하는 로봇”이라 할 수 있다.
09. 지능형 서비스 로봇은 크게 개인 서비스 로봇과 전문 서비스 로봇으로 분류된다.
10. 개인 서비스 로봇은 인간 생활 범주에서 제반 서비스를 제공하는 인간 공존형 대인지원 서비스 로봇으로 가사용 로봇, 생활 지원용 로봇, 여가 지원용 로봇, 교육용 로봇 등이 있다.
11. 전문 서비스 로봇은 불특정 다수를 위한 서비스를 제공하고 전문화된 작업을 수행하는 로봇으로 군사용 로봇, 의료복지용 로봇, 극한 작업용 로봇, 빌딩 서비스용 로봇 등이 있다.
12. 로봇 산업은 산업용 로봇, 지능형 로봇, 인간형 로봇 순으로 성장하고 있다.

# 단원 종합 문제

01

로봇이 우리의 일상생활에서 활용되는 사례 3가지를 열거하시오.

02

로봇과 인간과의 관계에 있어 로봇이 반드시 지켜야 할 원칙에 대한 설명으로 옳지 않은 것은?

- ① 로봇은 인간에게 해를 끼쳐서는 안 된다.
- ② 위험에 처해있는 인간을 방관해서도 안 된다.
- ③ 로봇은 일반적으로 자기 자신을 보호해야 한다.
- ④ 로봇은 어떠한 경우에도 인간의 명령에 반드시 복종해야 한다.

03

일상생활에서 발견한 응용 로봇에서 로봇의 3원칙은 어떻게 적용되고 있는지 설명해 보자.

04

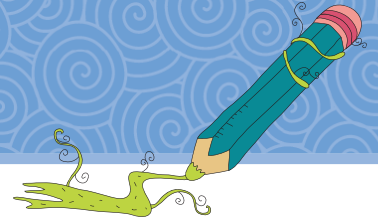
최초의 산업용 로봇인 것은?

- ① 센토                      ② 아이보                      ③ 유니메이트                      ④ 스탠퍼드암

05

다음 중 산업용 로봇의 활용 분야인 것은?

- ① 반송                      ② 청소                      ③ 간병                      ④ 안내



06

다음 중 로봇의 분류 기준이 아닌 것은?

- ① 제어 방식      ② 보유 기능      ③ 사용 용도      ④ 동작 특성

07

일반적으로 스카라 로봇이라 불리는 것은?

- ① 직교 좌표 로봇      ② 원통 좌표 로봇  
③ 수직 다관절 로봇      ④ 수평 다관절 로봇

08

다음 중 개인 서비스 로봇인 것은?

- ① 재활 로봇      ② 수술 로봇      ③ 탐사 로봇      ④ 안내 로봇

09

전문 서비스 로봇으로 볼 수 없는 것은?

- ① 군사용 로봇      ② 교육용 로봇  
③ 의료복지용 로봇      ④ 빌딩 서비스용 로봇

10

일상생활을 보다 편리하게 하기 위해 로봇이 필요한 이유를 3가지만 열거하여 보자.

The background is a blue-tinted collage. On the right, a globe is partially visible. In the center, a complex wireframe sphere is overlaid. On the left, a financial data table is visible with various percentages and category names.

# ROBOT CONTROL SYSTEM

## THE NUMBERS

Energy Prices	0.5%
Food	0.2%
Car Prices	0.9%
Commodities	2.4%
	0.8%

Energy  
Food  
Car Prices  
Commodities



# II 로봇은 무엇으로 구성되는가?

1. 로봇의 구성과 원리
2. 기어의 종류와 쓰임새
3. 기계 부품의 종류와 쓰임새
4. 제작 도구의 종류와 쓰임새

**로봇**의 움직임은 바퀴를 활용하거나 기어와 지레를 이용한 관절 운동을 통해 이루어진다. 기어와 지레를 활용한 관절 운동은 일반적으로 다리 모양을 만들어 구현한다. 또한, 납땀 등의 작업을 통해 로봇 제어 회로를 만들고 측정 도구를 사용하여 제어 회로의 동작을 확인한다. 이 단원에서는 로봇의 구성 원리, 기계 부품과 도구의 종류 및 그 쓰임새에 대해 학습하기로 한다.

# 01. 로봇의 구성과 원리

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇의 구성 장치에 대해 설명할 수 있다.
- 지레의 역할과 동작 원리에 대해 설명할 수 있다.

### 1. 로봇의 구성

로봇 전문가들은 로봇이 2030년 즈음에는 사람과 거의 유사한 지능을 가지게 되며, 로봇의 형태도 사람과 거의 유사한 형태와 구조를 가지게 될 것으로 전망하고 있다.

이미 ‘아시모(Asimo)’와 같은 휴머노이드 로봇은 직립 보행뿐만 아니라 컵을 잡고 악수를 하는 등 실제로 인간과 유사한 행동을 구현하고 있으며, 더 자연스러운 움직임을 위해 많은 연구가 진행되고 있다.

로봇은 크게 주 제어 장치, 센서 장치, 구동 장치, 전원 장치로 구성된다.

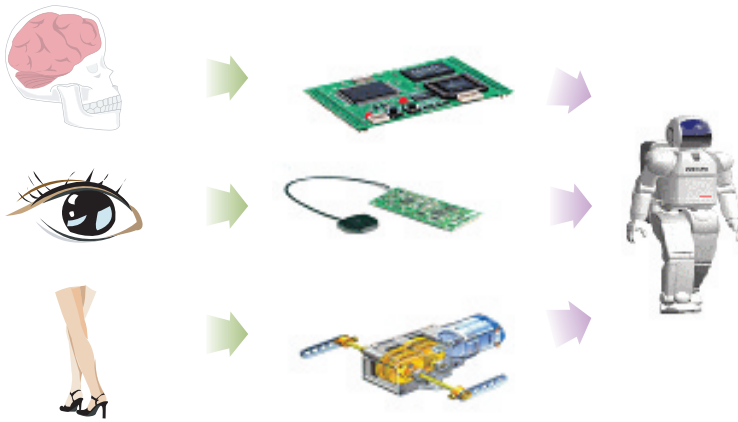
- 주 제어 장치 : 사람의 뇌에 해당하는 장치로, 각종 주변 환경에 대한 판단과 행동에 대한 명령을 내리는 역할을 담당하는 장치이다.
- 센서 장치 : 사람의 감각 기관과 같은 역할을 담당하는 장치로, 주변 환경에 대한 정보를 인식하는 센서로 이루어져 있는 환경 인식 장치이다.
- 구동 장치 : 사람의 팔과 다리와 같은 역할을 담당하는 장치로, 로봇이





실질적으로 일을 수행하는 장치이다.

- 전원 장치 : 로봇이 활동할 수 있는 에너지를 공급하는 역할을 담당하는 장치이다.



[그림 2-1] 사람과 비교한 로봇의 구성



### 아시모(Asimo)

일본 Honda 사에서 제작한 휴머노이드 로봇.

1,2미터 크기로 두 다리를 가졌고 중량이 52kg에 이르며 흰색과 회색 두 종류로 제작된 아시모는 시속 6[km]의 속도로 달릴 수 있으며, 계단 오르내리기, 곡선 주행, 음성 인식, 동작 인식, 감정 표현 등의 기능이 있다.



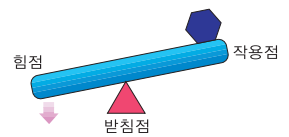
## 2. 로봇의 동작 원리

로봇은 물체를 들어 올리거나 이동할 때 지레의 원리를 사용한다.

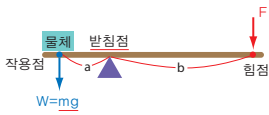
### (1) 지레

무거운 돌덩어리를 옮길 때 어떤 점에 막대를 받쳐서 힘을 가하면 쉽게 옮겨진다. 이처럼 지레는 받침점과 막대를 이용하여 작은 힘으로 물체를 쉽게 움직이게 하는 도구로서 BC220년경 아르키메데스(Archimedes)가 발견했다. 지레는 막대를 받치는 받침점, 힘을 가하는 힘점, 물체에 힘이 작용하는 작용점으로 구성된다.

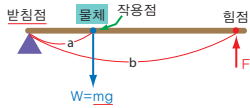
지레는 받침점, 힘점, 작용점의 위치에 따라 1종 지레, 2종 지레, 3종 지레로 분류된다.



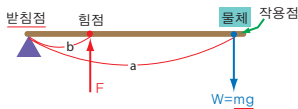
[그림 2-2] 지레의 원리



[그림 2-3] 1종 지레의 원리



[그림 2-4] 2종 지레의 원리



[그림 2-5] 3종 지레의 원리

### ① 1종 지레

[그림 2-3]과 같이 받침점이 작용점과 힘점 사이에 있으며 시소, 가위 등이 이에 속한다.

### ② 2종 지레

[그림 2-4]와 같이 작용점이 받침점과 힘점 사이에 있으며 병따개, 작두 등이 이에 속한다.

### ③ 3종 지레

[그림 2-5]와 같이 힘점이 받침점과 작용점 사이에 있으며 핀셋, 낚시대, 족집게 등이 이에 속한다.

지레는 받침점과 힘점 사이의 거리, 받침점과 작용점과의 거리 등으로 힘이 가해지는 크기와 방향이 결정된다. 로봇 팔을 이용해 물건을 들려고 할 때, 이와 같은 원리를 적용해서 설계해야 한다.

## (2) 도르래

도르래는 바퀴의 일종으로 벨트나 끈을 바퀴의 홈에 끼워 무거운 짐을 아래에서 위로 끌어올릴 때 사용되는 도구이다. 힘의 방향을 바꿔주거나 힘의 크기를 변화시키는 작용을 한다.

도르래는 1796년 정약용이 수원 화성 성곽을 쌓으며 사용된 거중기를 비롯하여 국기 게양대, 타워 크레인 등에서 사용된다.

도르래는 고정 도르래, 움직 도르래, 복합 도르래로 분류된다.

### ① 고정 도르래

대표적인 것은 우물의 두레박과 엘리베이터이다. 힘의 방향만 바꿀 수 있고, 힘의 크기는 변화가 없다.

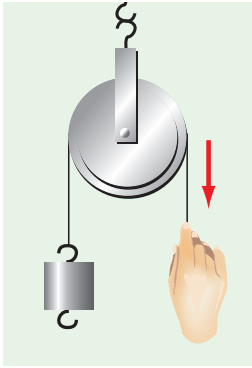
### ② 움직 도르래

힘의 방향은 변화가 없고 물체와 도르래의 무게를 합한 것의 반만큼의 힘으로도 들어 올릴 수 있다.

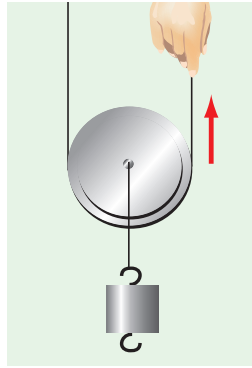


### ③ 복합 도르래

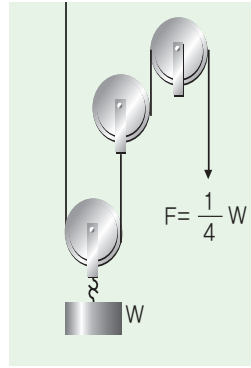
고정 도르래는 힘의 크기를 변화시키지 못하고 움직 도르래는 힘의 방향을 변화시키지 못하는 단점을 보완한 도르래, 고정 도르래와 움직 도르래를 함께 사용한다.



[그림 2-7] 고정 도르래



[그림 2-8] 움직 도르래

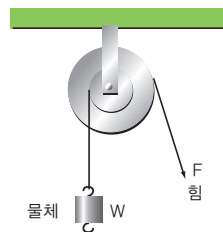


[그림 2-9] 복합 도르래

### (3) 축바퀴

축바퀴는 작은 바퀴와 큰 바퀴를 하나의 축에 고정시켜서 함께 회전하도록 만든 것이다. 고정된 축이 지레의 받침점과 같이 작용을 하고 바퀴는 지레의 힘점과 작용점에 해당한다.

축바퀴를 사용하면 물체를 무게보다 작은 힘으로 들어 올릴 수 있다. 축바퀴가 사용된 예로는 자동차 핸들, 드라이버, 출입문 손잡이, 전화 다이얼 등이 있다.



[그림 2-10] 축바퀴

## 02. 기어의 종류와 쓰임새

### ROBOT CONTROL SYSTEM

#### 학습목표



- 기어의 종류별 특징과 쓰임새에 대해 설명할 수 있다.
- 감속 기어와 가속 기어의 원리에 대해 설명할 수 있다.
- 모터의 회전 수와 토크와의 관계에 대해 설명할 수 있다.



#### 기어(gear)

2개 또는 그 이상의 축 사이에 회전이나 동력을 전달하는 장치

기어는 톱니가 달린 바퀴의 일종으로 2개 이상의 축 사이에 힘을 전달하거나 힘의 방향, 속도를 변경할 수 있다. 기어는 톱니의 물림에 따라 평형 기어, 교차 기어, 엇갈림 기어로 나뉘는데, 이것은 다시 톱니의 형태에 따라 스퍼 기어, 베벨 기어, 랙 기어, 워엄 기어, 피니언 기어 등이 있다.



### 1. 기어의 종류

#### ① 스퍼 기어

스퍼 기어는 가장 흔히 사용되는 기어로 평 기어라 한다.

톱니가 맞물려 두 축이 역방향으로 회전하며 시계, 내연 기관 등에 주로 사용한다.

#### ② 헬리컬 기어

헬리컬 기어는 톱니가 축과 경사지며, 맞물림이 원활하다는 특징이 있다.

헬리컬 기어는 스퍼 기어보다 쉽게 회전하고, 소음이 적다는 장점으로 인



해 공작 기계, 크랭크 축 기어 등에 사용한다.

헬리컬 기어는 축 방향으로 힘이 작용하므로 힘을 받아 주는 스러스트 베어링(thrust bearing)이 필요하다.

### ③ 레크 기어

평면상에 톱니가 있으며, 회전 운동을 직선 왕복 운동으로 바꾸거나 직선 왕복 운동을 회전 운동으로 바꾸는 기능을 수행하는 기어이다. 레크 기어는 사진기의 삼각대, 자동차 조향 장치 등에 사용한다.

### ④ 베벨 기어

원뿔의 표면에 톱니가 있으며, 회전의 방향을 직각 방향으로 바꾼다. 자동차 구동 장치, 핸드 드릴 등에 사용한다.

### ⑤ 월 기어

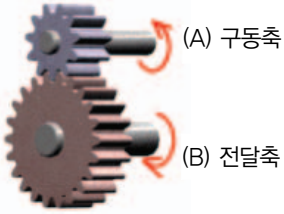
나사 모양의 기어로서 회전 속도를 크게 낮추는 특징이 있어 감속 장치에 사용한다.



[그림 2-11] 기어의 종류

## 2. 감속기어와 가속기어

기어는 힘을 전달하고 회전 방향과 회전 속도 등을 변경하는 경우에 사용된다. 2개 이상의 기어를 맞물리면 각 기어의 톱니 수에 따라 축의 회전수가 결정된다. 축의 회전수를 줄이는 기어를 감속 기어라 하고, 회전수를 늘리는 기어를 가속 기어라 한다.



[그림 2-12] 감속 기어

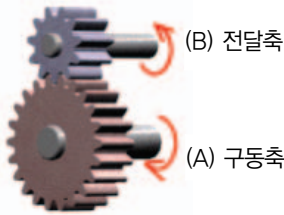
### (1) 감속 기어

[그림 2-12]와 같이 모터가 있는 구동축에 물려있는 기어(A)의 톱니 수가 바퀴가 있는 전달축에 물려 있는 기어(B)의 톱니 수보다 작으면 감속 기어로서 회전 수가 줄어든다. 감속비는 구동축에 물려 있는 A기어의 톱니 수와 전달축에 물려 있는 B기어의 톱니 수의 비로 나타내며, [그림 2-12]에서는 1:2이다. A기어가 2회전할 때 B기어는 1회전하여 모터의 회전 수의 1/2로 감소되어 바퀴가 회전한다.



#### 예제

구동 축의 톱니의 수가 4개이고, 전달 축의 톱니의 수가 16개인 경우, 감속비는 얼마인가?



[그림 2-13] 가속 기어

### (2) 가속 기어

[그림 2-13]과 같이 모터가 있는 구동축에 물려있는 기어(A)의 톱니 수가 바퀴가 있는 전달축에 물려 있는 기어(B)의 톱니 수보다 많으면 가속 기어로서 전달축의 회전 수가 증가한다. 가속비는 A 기어의 톱니 수와 B 기어의 톱니 수의 비이며, [그림 2-13]에서 가속비는 2:1이다. A기어가 1회전할 때, B기어는 2회전하여 모터의 속도의 2배로 증가되어 바퀴가 회전한다.



#### 예제

구동 축의 톱니의 수가 16개이고, 전달 축의 톱니의 수가 4개인 경우, 가속비는 얼마인가?

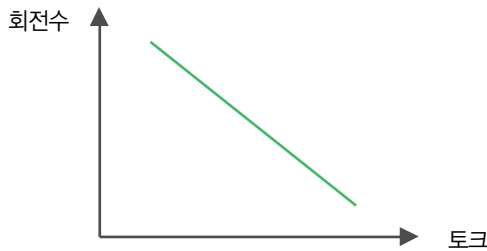






### (3) 회전 수와 토크의 관계

로봇의 구동부는 대부분 모터로 구성된다. 바퀴를 굴려 이동하는 로봇은 이동시켜야 하는 무게에 따라 모터의 토크가 커져야 이동을 원활하게 할 수 있다. 물체를 들어 올릴 때도 모터의 토크가 작으면 물체를 들어 올리질 못 할 수 있다. 이렇게 로봇의 구동부 역할을 하는 모터의 토크를 고려하여 설계를 하는데, 모터의 회전수와 토크는 서로 상충되는 관계를 갖는다.



[그림 2-14] 모터의 회전 수와 토크의 관계

[그림 2-14]와 같이 모터의 회전 수가 많으면 토크가 약하고 모터의 회전 수가 적으면 경우 토크가 크다. 그러므로 로봇의 구동 목표에 따라 회전 수가 많은 모터를 사용하거나, 토크가 센 모터를 사용해야 한다.

모터만으로 로봇의 무게로 인해 움직이지 않으면 감속 기어를 통해 모터의 토크를 증가시켜 사용한다.

서보 모터는 감속비가 높아 속도는 느리지만, 토크가 커 무거운 물체를 들어 올릴 수 있어서 로봇 팔, 이족 로봇 등에 사용한다.

DC 모터는 감속비는 낮아서 속도가 빠르고, 토크가 적어 바퀴를 이용해 이동하는 로봇에 많이 사용한다.



#### 토크(Torque)

회전력을 말하는 것으로 모터가 회전하는 힘을 의미한다.

# 03. 기계 부품의 종류와 쓰임새

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇을 구성하는 기계 부품들의 기능과 역할에 대해 설명할 수 있다.
- 로봇을 제작하는 도구들의 기능과 사용할 때 주의할 점을 이해한다.

로봇에는 다양한 기계 부품들을 사용한다. 흔히 볼 수 있는 볼트, 너트에 서부터 운동 방향을 바꾸거나 힘의 전달 방법 등에 따라 캐터필러, 스프로킷, 스프링 등이 있다.

### 1. 결합용 기계 부품

프레임과 프레임간의 결합을 위해 사용되는 부품인데, 나사, 볼트, 너트, 와셔 등이 있다.

#### ① 나사

직각삼각형 종이를 원통에 감았을 때 생기는 나선선을 따라 홈을 파 만들어진 것을 말하며, 프레임과 프레임을 고정할 때 사용한다. 나선선의 모양에 따라 삼각나사, 사각나사, 톱니나사, 사다리꼴나사, 둥근나사로 구분된다.



[그림 2-15]  
결합용 부품의 형태

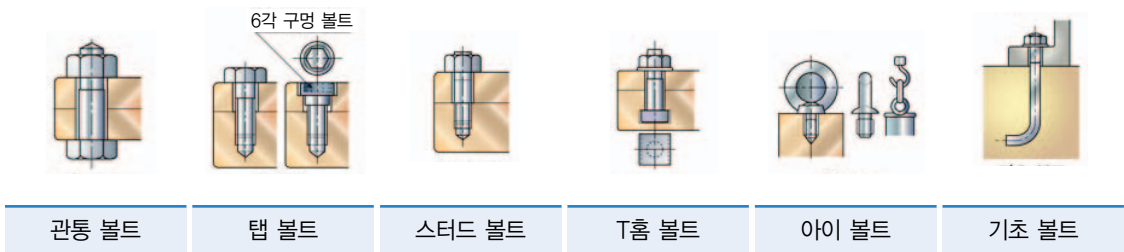


[표 2-1] 나사의 종류

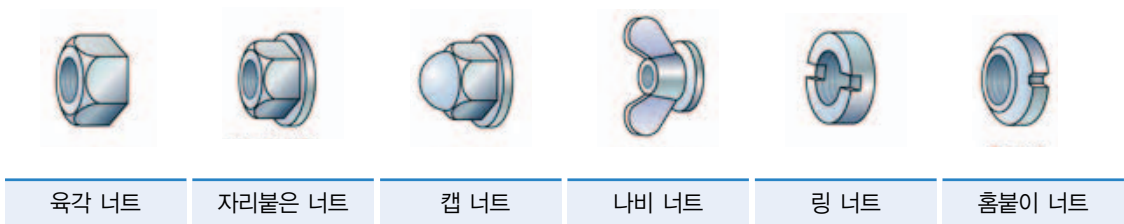
나사의 종류	용도	사용 예
삼각나사	두개의 물체를 결합할 때 사용	볼트, 너트
사각나사	힘을 전달할 때 사용	바이스
톱니나사	한 방향으로 큰 힘을 전달할 때 사용	프레스, 잭
사다리꼴나사	운동을 전달할 때 사용	공작기계
둥근나사	먼지가 들어가기 쉬운 곳에 사용	백열전구

### ② 볼트와 너트

일반적인 나사를 말하기도 하며, 강철 머리 모양이 달린 수나사를 볼트라고 하고, 암나사를 너트라고 한다.



[그림 2-16] 볼트의 종류



[그림 2-17] 너트의 종류

### ③ 와셔

볼트와 너트를 사용할 때 결합된 물체 표면에 덧대어 두 물체가 움직여도 볼트와 너트가 잘 풀리지 않게 하거나, 물체 표면이 손상되지 않도록 사용하는 부품이다.



차축



전동축

[그림 2-18] 차축과 전동축



마찰차



기어

[그림 2-19] 마찰차와 기어

## 2. 축용 기계 부품

회전 운동을 하는 물체의 중심에 끼워진 막대를 축이라 한다. 축으로는 풀리, 스프라킷, 기어, 캠 등을 사용한다. 축에 작용하는 힘에 따라 차축과 전동축으로 구분하고, 축의 모양에 따라 직선 축과 크랭크축으로 분류한다.

### ① 차축

축은 고정되고 바퀴만 회전한다. 주로 힘 작용만을 받는 회전축이나 정지축이 있고, 자동차의 앞바퀴에 사용한다.

### ② 전동축

축과 회전체가 함께 고정되어 축의 동력을 바퀴에 전달할 수 있는 축을 말하며, 자동차의 뒷바퀴에 사용한다.

### ③ 크랭크축

직선 왕복 운동을 회전 운동으로 바꿔주는 축이다.

### ④ 베어링

회전하는 축의 마찰을 최소화하며 축을 받쳐주는 부품이다.

## 3. 전동용 기계 부품

기어와 같이 힘의 전달, 속도 조절, 힘의 방향 전환 등을 위한 기계부품으로 기어, 마찰차, 캠, 벨트, 체인, 링크가 있다.

### ① 마찰차

회전하는 두 개의 바퀴를 맞붙여 두 바퀴의 마찰력으로 회전력을 전달한다. 미끄러짐 때문에 정확한 회전 운동에는 적합하지 않으나 접촉을 유지한 상태에서 회전력을 전달하는 변속기 등에 이용한다.

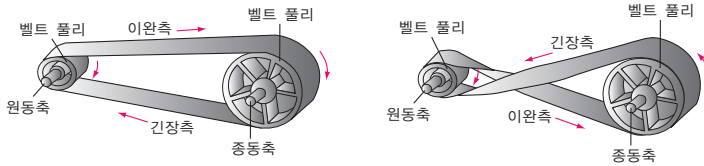


## 2 기어

톱니와 톱니의 접촉을 통해 회전력을 전달한다. 회전 방향의 전환, 회전 속도의 조절을 하며, 회전력을 정확하게 전달한다. 두 축 간의 거리가 짧은 전동 장치에 사용한다.

## 3 벨트

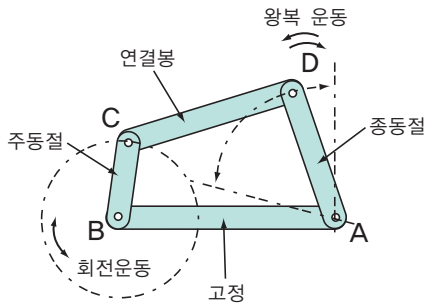
두 개의 축에 연결된 폴리에 고무로 된 띠를 둘러 회전력을 전달한다. 마찰처럼 미끄러짐 때문에 정밀한 회전력의 전달할 수 없다. 두 축 사이의 거리가 먼 경우에 적합하다. 두 개의 축을 벨트로 연결하는 방식은 평형걸기, 엇갈림걸기로 분류된다. 벨트 안쪽에 일정한 간격으로 홈을 파고, 기어와 유사하게 폴리에 홈을 파서 일정한 간격으로 움직이게 한 것을 타이밍 벨트라 한다. 타이밍 벨트는 미끄럼 없이 회전이 원활하다.



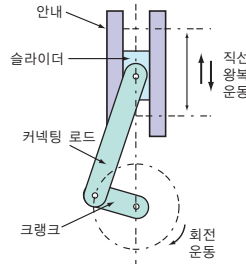
[그림 2-20] 평행 벨트와 엇갈림 벨트

## 4 링크

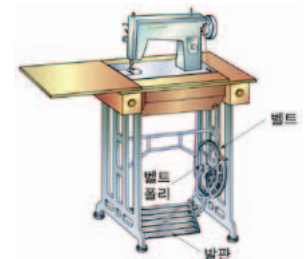
모터의 축에 크랭크를 연결하고, 크랭크에 다시 긴 막대를 연결하여 회전 운동을 직선 운동과 같이 운동 방향을 전환하는 기구이다. 3절 링크, 4절 링크, 5절 링크가 있으며, 크레인이 물체를 들어 올릴 때, 선풍기의 몸체를 좌우로 움직이게 할 때 등에 사용한다.



[그림 2-21] 링크 기구의 원리



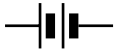
왕복 슬라이드 크랭크 링크 기구



재봉틀의 크랭크 링크 기구

[그림 2-22] 링크 기구의 사용





D.C 전원



A.C 전원

[그림 2-23] 전원의 기호



[그림 2-24] 1차 전지

## 4. 전지의 종류

MP3 플레이어, 디지털 카메라, 휴대 전화기 등 휴대용 전자 기기가 발달하여 전지의 활용이 점차 증가하고 있다. 전지에 대한 기술이 발전되어 장기간 사용할 수 있는 고용량의 전지가 개발되고 있다.

전지는 들어있는 화학 물질의 화학 작용으로 전기를 발생시키는데 일반적으로 1차 전지와 2차 전지로 구분된다.

### (1) 1차 전지

1차 전지는 화학 물질의 화학 변화가 끝나면 수명을 다해 사용할 수 없다. 1차 전지에는 망간 건전지, 알칼리 건전지 등이 있다.

#### ① 망간 건전지

일반적으로 사용하는 건전지로서 시계나 장난감 등에 많이 사용한다. 사용하다가 사용을 잠시 중단하면 전압이 회복되는 특징이 있어 사용 방법에 따라 수명을 길게 사용할 수 있다.

#### ② 알칼리 건전지

망간 건전지보다 이산화망간과 아연을 많이 넣어 수명이 길고 성능이 좋다. 연속적으로 큰 전류가 필요한 카세트나 카메라에 적합하다. 최근에는 망간 건전지보다 알칼리 건전지를 많이 사용한다.

#### ③ 기타 건전지

수은 전지, 리튬 전지 등이 있다.

### (2) 2차 전지

2차 전지는 전기 에너지를 공급하면 화학 물질이 재생되어 다시 사용할 수 있게 되는 전지이다. 2차 전지에는 니켈-카드뮴 전지, 니켈-수소 전지, 리튬 이온 2차 전지, 납축 전지 등이 있다.



### ① 니켈-카드뮴 전지

가장 널리 사용하는 충전식 전지로서 건전지 소모가 많은 로봇이나 무선 전화기, 면도기 등에 많이 사용한다. 그러나 메모리 효과가 있어 완전히 방전하지 않고 충전하면 용량이 줄어드는 단점이 있다.

### ② 니켈-수소 전지

니켈-수소 전지는 같은 중량으로 Ni-Cd 전지에 비해 2배의 전기 용량을 가지고 있어 1회 충전으로 보다 장시간 사용할 수 있는 특징이 있다.

니켈-수소 전지는 기기의 소형화를 가능하게 하여 노트북, 휴대용 전화 등 비교적 고가의 휴대용 기기에 사용한다.

### ③ 리튬 이온 2차 전지

리튬 이온 2차 전지는 작고 가벼운 고출력 전지로 작은 기기인 휴대 전화나 캠코더 등에 사용한다.

### ④ 납축 전지

축전지 중에서 가장 오래된 전지로서 저렴한 비용으로 제조가 가능하며 기술 완성도가 비교적 높다. 작은 전기 제품부터 지게차의 동력용, UPS용, 전기 자동차용, 태양 전지 발전용 등 다양한 부분에 적용되고 있다.

## (3) 전지의 사용

- ① 사용되는 목적에 따라 1차 전지나 2차 전지를 선정하여 사용한다. 전지의 소모가 많은 경우에는 2차 전지를 주로 사용한다.
- ② (+), (-)극성이 있다. 양극이 합선되면 과전류가 흘러서 파손되거나 열이 발생하여 폭발하거나 외부에 손상을 입힐 수 있다.
- ③ 2차 전지는 전용 충전 장치를 사용해야 오래 사용할 수 있다.



[그림 2-25] 2차 전지

# 04. 제작 도구의 종류와 쓰임새

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇 제작용 도구의 종류와 특징에 대해 설명할 수 있다.
- 로봇을 제작하는 도구와 측정 도구의 사용법을 이해한다.

로봇을 제작할 때 사용하는 도구에는 로봇의 전자 제어부를 제작할 때 사용하는 전자 도구, 몸체를 구성할 때 사용하는 기계 도구, 로봇의 동작 상태를 점검하기 위해 사용하는 측정 도구가 있다.

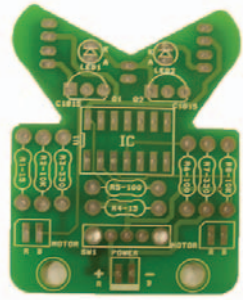
### 1. 전자 도구

로봇의 전자 제어부를 제작하는 도구에는 기판, 납땀 인두, 인두 받침대, 납 흡입기, 니퍼, 롱노우즈, 기판 받침대 등이 있다.

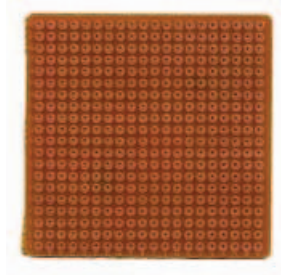
#### (1) 기판(PCB)

회로 기판에는 다양한 전자 회로를 제작할 수 있는 만능 기판과 특정한 회로가 기판에 이미 제작되어 있어서 부품만 꽂도록 만든 인쇄 회로 기판(PCB)이 있다.

인쇄 회로 기판의 녹색 코팅의 아래에 전기를 통하는 구리선이 있어서 부품들을 연결한다. 만능 기판은 구리 원판이 있는 곳에 납땀을 하여 부품을 고정하고, 부품과 부품 사이는 점퍼선을 이용하여 연결한다.



인쇄 회로 기판(PCB)



만능기판

[그림 2-26] 기판의 종류

## (2) 납땜 인두

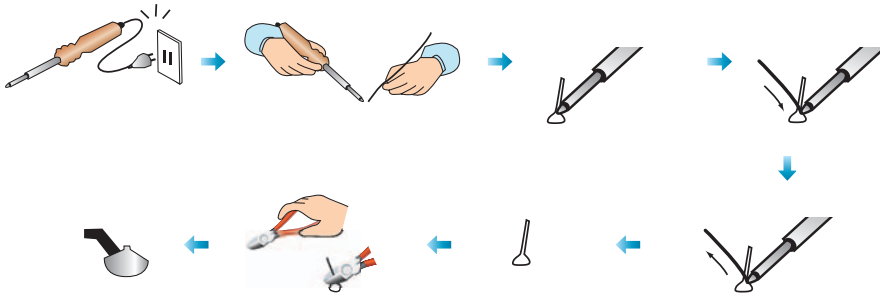
부품을 기판에 끼워서 고정하고 회로를 연결할 때 납땜 인두를 사용한다. 납이 녹는 약 300℃의 고온을 내는 도구이므로 사용할 때 각별히 주의해야 한다. 고정막대형 인두기, 세라믹 인두기, 초음파 인두기 등이 있다. 주로 저렴한 막대형 인두기나 세라믹 인두기를 많이 사용한다.



[그림 2-27] 납땜 인두의 종류

### 1) 납땜 순서

인두기를 오른손에 쥐고, 납은 왼손에 잡고 팔을 테이블에 받쳐 팔이 움직이지 않도록 하고, 침착하게 다음의 순서를 따라 납땜을 한다.

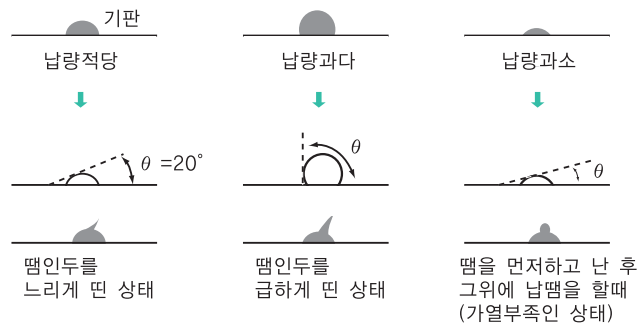


[그림 2-28] 납땜 순서

- ① 인두를 납땜할 위치의 동판에 대어 동판을 약간 가열한다.
- ② 인두를 계속 동판에 대고, 납을 인두 끝에 대서 녹인다.
- ③ 납이 어느 정도 녹으면, 납을 먼저 인두에서 떼낸다.
- ④ 납이 동판 전체에 퍼질 때까지 잠깐 인두를 동판에 대고 있어야 한다.
- ⑤ 인두를 동판에서 떼낸다.

## 2) 올바른 납땜의 상태

좋은 납땜을 하려면 납량과 납땜의 속도가 적절하게 이루어져야 한다.



[그림 2-29] 납땜의 상태



## (3) 인두 받침대

납땜 인두는 약 300℃ 이상의 고온에서 사용됨으로 종이, 나무 등을



태울 수 있고, 사람의 피부에 닿으면 화상을 입을 수 있다. 안전을 위하여 인두를 받쳐두는 받침대를 사용한다.

#### (4) 납 흡입기

부품이 기판에 잘못 부착되었거나, 납땜 상태가 좋지 않아서 납을 기판에서 제거할 때 사용한다.

먼저 흡입기의 뒷부분을 밀어 납을 빨아들일 수 있는 상태를 만든 다음에 인두를 제거할 납땜에 대서 녹인다. 납이 어느 정도 녹으면 흡입기의 납 흡입구를 대고 버튼을 눌러 납을 빨아들인다.



이 경우 인두나 흡입기를 먼저 제거하면 납이 충분히 제거되지 않는다.

#### (5) 니퍼

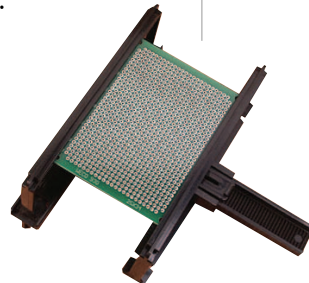
기판에서 납땜할 구리 동판을 벗어난 부품의 다리 부분을 제거하거나, 전선의 피복을 벗기거나, 전선을 자르는 작업에 사용한다. 니퍼는 앞부분이 날카로우므로 주의해야 한다.

#### (6) 롱로우즈

롱로우즈는 라디오 펜치라고도 한다. 부품의 다리를 구부리거나, 두꺼운 철사를 자를 때 사용한다.

#### (7) 기판 받침대

부품의 다리를 구부리기만 해서 고정한 상태로 뒷면으로 돌려 납땜을 한다. 이때 기판을 움직이지 않도록 고정하기 위해서 사용한다.





## 2. 기계 도구

[표 2-2] 기계 도구의 종류

도구 명칭	실물 사진	용 도
미니바이스		아크릴 등 가공할 재료를 고정시킬 때 사용하는 기계 도구이다.
전동드릴		핸드 타입으로 드릴 비트를 끼워 드릴로 사용하기도 하고, 드라이버 팁을 끼워 전동 드라이버로도 사용할 수 있다.
드라이버		(+), (-)모양인 끝 모양에 따라 십자, 일자 드라이버로 구분한다. 나사를 조이거나 풀 때, 나사 머리의 홈에 따라 구분하여 사용한다.
스패너		너트를 조이거나 풀 목적으로 사용한다.
글루건		실리콘과 같은 고체 접착제를 전기로 녹여 접착할 때 사용한다.
스트리퍼		전선의 피복을 벗길 때 사용한다. 전선의 굵기에 따라 사용할 수 있다.

이밖에도 용도에 따라 다양한 기계 도구가 있다. 기계 도구는 사용할 때 자신과 주변 사람에게 해를 미칠 수 있으므로 용도와 사용법 등을 바르게 익혀 사용해야 한다.

## 3. 측정 도구

로봇의 구조물을 제작하기 위하여 길이, 모터 속도 등을 측정하거나 전자



제어 회로의 동작을 점검할 때 사용하는 도구를 측정 도구라 한다. 측정 도구는 여러 가지가 있으나 멀티미터, 오실로스코프, 룸라이터, 타코미터, 버니어캘리퍼스 등이 많이 사용된다.

### (1) 멀티미터

멀티미터(Multimeter)는 전자 회로의 전압, 전류, 저항을 측정하는 도구로서 회로 시험기 또는 테스터(Tester)라고도 한다.

멀티미터는 아날로그 방식과 디지털 방식이 있으며, 아날로그 방식의 멀티미터가 디지털 멀티미터에 비해 측정의 오차가 크나 가격이 저렴하여 일반적으로 많이 사용한다.

아날로그 방식의 멀티미터의 사용 방법에 대해 알아보자.



[그림 2-30] 멀티미터의 형태

#### 1) 멀티미터 부위별 기능

- ① 0옴 조정 다이얼 : 저항을 측정할 때 두 개의 측정 봉을 맞붙여 0옴 기준을 정확하게 맞추려고 사용한다.

- ② TR 측정 소켓 : 트랜지스터의 극성을 확인할 때 사용한다.
- ③ 측정 대상/범위 선택 스위치 : 저항 측정, DC 전류, DC 전압, AC 전압 등 측정할 대상과 최대 측정 범위를 선택한다.
- ④ (+), (-)측정 봉 소켓 : (+)극성에 해당하는 적색 측정 봉을 (+)측정 봉 소켓에 삽입하고, (-)극성에 해당하는 검정색 측정 봉을 (-)측정 봉 소켓에 삽입하여 측정 대상에 측정 봉을 대고 측정한다.
- ⑤ 전류 측정 봉 소켓 : DC 10[A]까지의 전류를 측정하고자 할 때, (+)측정봉인 적색 측정 봉을 꽂고 DC 전류를 측정한다.
- ⑥ 측정값 표시 눈금 : 저항 측정, DC 전압, AC 전압, 다이오드 내압 측정 등에 따라 각각 표시되는 눈금이 다르다. 눈금의 색깔에 따라 구분한다.

## 2) 저항 측정

- ① 저항을 측정하려면 [그림 2-31]과 같이 선택 스위치를 저항 측정 범위에 위치시킨다.



[그림 2-31] 저항 측정 범위



[그림 2-32] 저항 측정 눈금



[그림 2-33] 멀티미터를 활용한 저항 측정

- ② 적색 측정봉과 검정색 측정봉을 각각 저항 다리에 대고 측정한다.
- ③ 저항을 측정하여 측정값을 읽을 때 [그림 2-32]와 같이 맨 위쪽에 있는 녹색으로 표시된 눈금을 읽는다. 최대 측정 범위에 따라 눈금을 통해 읽은 값에 측정 배율을 곱해 준다.
- ④ 예를 들면 선택 스위치를 'x1'에 두면 측정 범위가 0[Ω]부터 2[kΩ]까지 측정할 수 있다. 이 때 저항값은 지시한 눈금의 값이 된다.



- ⑤ 만약 선택 스위치를 'x1K'에 두었을 때는 측정 범위는 0[Ω ]부터 2[MΩ]까지 측정할 수 있다. 이 때 저항값은 지시한 눈금의 값에 1000을 곱한 값이 된다.
- ⑥ DC 전압과 AC 전압을 측정하고자 할 경우에도 선택 스위치를 조정하여 측정 대상과 범위를 선택한 다음 측정된 값을 눈금으로 읽는다.

### (2) 버니어 캘리퍼스

버니어 캘리퍼스(Vernier calipers)는 일반 자와 같이 길이를 측정하는 도구이며, 정밀한 측정이 가능하여 기계 도구에서 많이 사용된다. 사용 방법에 따라 깊이의 측정, 원의 지름, 내부 원의 지름 등을 측정할 수 있다.



### (3) 오실로스코프

오실로스코프(Oscilloscope)는 멀티미터와 유사한 기능을 수행하는 측정 도구로, 전자 회로가 구동될 때 실제 신호의 움직임을 확인하고자 할 때 사용한다.



아날로그 신호와 디지털 신호를 측정할 수 있고 빠른 속도로 움직이는 신호를 자세하게 볼 수 있다. 멀티미터는 신호의 크기 정도만 알 수 있으나, 오실로스코프는 신호의 크기, 폭, 실제 구동되는 과정 등을 볼 수 있어서 회로 점검 등에 많이 사용한다.



.....

.....

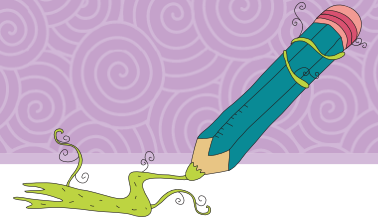
.....

.....

# 단원 학습 정리

01. 로봇은 주 제어 장치, 센서 장치, 구동 장치, 전원 장치로 구성된다.
02. 지레는 힘점, 받침점, 작용점이 있는 막대로서 힘의 크기를 바꿀 수 있고 힘점, 받침점, 작용점의 위치에 따라 1종, 2종, 3종으로 분류된다.
03. 도르래에는 고정 도르래와 움직 도르래, 복합 도르래가 있다.
04. 고정 도르래는 힘의 크기는 변화가 없고 힘의 방향만 바꿀 수 있는 도르래이다.
05. 움직 도르래는 힘의 방향은 변화가 없으나 힘의 크기를 반으로 줄일 수 있는 도르래이다.
06. 복합 도르래는 고정 도르래와 움직 도르래를 함께 사용하는 도르래이다.
07. 축바퀴는 큰 바퀴와 작은 바퀴를 하나의 축에 고정시킨 것으로 물체를 작은 힘으로 들어올리는 경우에 사용한다.
08. 기어는 두 축 사이에 힘을 전달하며 힘의 세기, 방향, 속도를 변경할 수 있다.
09. 기어에는 스퍼 기어, 헬리컬 기어, 레크 기어, 베벨 기어, 워 기어 등이 있다.





10. 모터의 회전 수가 크면 토크가 약하고, 모터의 회전 수가 작으면 토크가 크다.
11. 로봇을 제작할 때 사용하는 결합용 기계 부품에는 나사, 볼트, 너트, 와셔 등이 있다.
12. 축은 회전 운동을 하는 물체의 중심에 끼워진 막대를 의미하며, 축용 기계 부품에는 차축, 전동축, 크랭크축, 베어링 등이 있다.
13. 전동용 기계 부품은 힘의 전달, 속도 조절, 힘의 방향 전환 등과 같은 역할을 수행하며, 기어, 마찰차, 캠, 벨트, 체인, 링크 등이 있다.
14. 전지는 들어있는 화학 물질의 화학 작용으로 전기를 발생시키는데 일반적으로 1차 전지와 2차 전지로 구분된다.
15. 로봇의 전자 제어부를 제작하는 전자 도구에는 기판, 납땀 인두, 인두 받침대, 납 흡입기, 니퍼, 롱노우즈, 기판 받침대 등이 있다.
16. 로봇의 몸체를 작성할 때 사용하는 기계 도구에는 미니바이스, 전동 드릴, 드라이버, 스패너, 글루건, 스트리퍼 등이 있다.
17. 로봇의 동작 상태를 점검하기 위해 사용하는 측정 도구에는 멀티미터, 오실로스코프, 룸라이터, 타코미터, 버니어캘리퍼스 등이 있다.



# 단원 종합 문제

01

로봇을 구성하는 장치로 볼 수 없는 것은?

- ① 센서 장치      ② 제동 장치      ③ 구동 장치      ④ 주 제어 장치

02

자동차에서 지레가 활용되는 예를 찾아 어떤 종류의 지레가 사용되는지 쓰시오.

	자동차 부품명	사용하는 지레 종류
(예 1)		
(예 2)		
(예 3)		

03

큰 바퀴의 지름이 15[cm]이고 작은 바퀴의 지름이 5[cm]인 축바퀴가 있다. 작은 바퀴에 작용하는 힘은 큰 바퀴에 작용하는 힘의 몇 배인가?

- ① 1/3배      ② 2배      ③ 3배      ④ 변화 없음

04

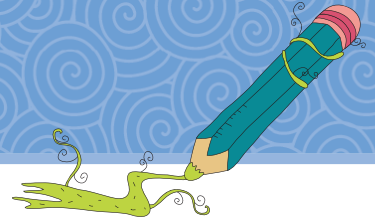
회전 속도를 낮추는 효과가 있어 감속 장치에 사용되는 기어는?

- ① 웜 기어      ② 레크 기어      ③ 베벨 기어      ④ 스퍼 기어

05

한 방향으로 큰 힘을 전달할 때 사용되는 나사는?

- ① 삼각나사      ② 사각나사      ③ 톱니나사      ④ 사다리꼴나사



06

직선 왕복 운동을 회전 운동으로 바꿔주는 기능을 수행하는 축은?

- ① 차축                      ② 베어링                      ③ 전동축                      ④ 크랭크축

07

회전력을 전달하기 위해 기어를 사용할 수 없어 벨트를 사용하였다. 이에 대한 원인으로 적합한 것은?

- ① 정확한 전달을 위해                      ② 가격이 저렴하여  
③ 거리가 너무 멀어서                      ④ 높은 안전성으로 인해

08

다음 중 2차 전지인 것은?

- ① 수은 전지                      ② 망간 건전지  
③ 알칼리 건전지                      ④ 니켈-카드뮴 전지

09

전선의 피복을 벗길 때 사용하는 기계 도구는?

- ① 스페너                      ② 글루건                      ③ 스트리퍼                      ④ 미니바이스

10

멀티미터를 사용하여 전압을 측정하는 절차를 열거해 보시오.



ROBOT CONTROL SYSTEM

THE NUMBERS

0.5%

0.2%

0.9%

2.4%

0.8%

Energy

Food

Car Price

Oil

Market

2004  
14.33  
14.33  
14.33  
14.33  
14.33

0.2%

0.9%



# III 로봇을 만들어 보자

1. 89T51로 구동하는 라인트레이서
2. 라인트레이서 조립하기

**실습**에 사용되는 마이크로로봇은 8비트 마이크로프로세서인 89T51을 응용한 라인트레이서이다. 이 장에서는 특별히 고안된 교육용 라인트레이서 로봇을 조립해 봄으로써 로봇 제어 회로의 구성과 설계에 관한 기본 개념과 원리에 대해 학습하기로 한다.

# 01. 89T51로 구동하는 라인트레이서

## ROBOT CONTROL SYSTEM

### 학습목표



- 89T51을 사용하는 라인트레이서의 특징을 말할 수 있다.
- 89T51을 사용하는 라인트레이서의 구성 회로를 열거할 수 있다.
- 89T51을 사용하는 라인트레이서의 구성 회로별 역할을 말할 수 있다.



#### 라인트레이서

센서를 이용하여 정해진 주행선을 따라 움직이는 자율 이동 로봇으로, 산업체에서 물건을 운반하는 무인 차(AGV) 등에 응용된다.



#### 적외선

전자기파 스펙트럼 중 가시광선의 적색광보다 길고 마이크로파보다 짧은 파장으로, 파장이  $0.75\mu\text{m}\sim 1\text{mm}$ 의 복사선을 의미한다.



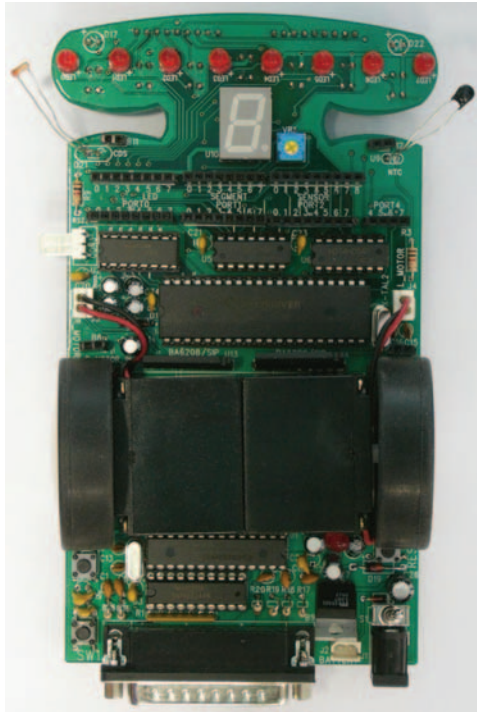
## 1. 실습용 마이크로로봇 개요

실습용 마이크로로봇은 GC89T51 마이크로컨트롤러를 이용하여 설계한 라인트레이서이다. 일반적인 라인트레이서 로봇의 경우 적외선 센서를 이용하여 선 따라가기 기능을 수행하나, 실습용 마이크로로봇은 기본 기능에 로봇 설계에 관한 학습에 활용하기 위한 기능을 추가한 교육용 라인트레이서이다.

89T51은 순수 국내 벤처기업에서 개발한 마이크로컨트롤러로서, 인텔에서 개발한 8051과 명령어가 모두 실행되기 때문에 기존의 8051 프로그램을 수정하지 않고 사용할 수 있을 뿐 아니라 속도를 3배 정도 향상시켰다.

또한, 89T51은 14[KByte]의 플래시 메모리를 내장하고 있으며, 프로그램의 동작 상태를 확인할 수 있는 기능을 포함하고 있다. 작성된 프로그램은 별도의 장치 없이도 쉽게 실행시킬 수 있고 프로그램에 오류가 있으면 동작 상태를 확인하는 기능을 이용하여 쉽게 수정할 수 있다.

1,792[Byte]의 데이터 메모리, A/D 변환 모듈, 폭이 조정되는 펄스 발생 모듈을 내장하고 있어 제어 시스템을 구성할 때 비용을 절감할 수 있는 장점



[그림 3-1] 89T51로 구동하는 라인트레이서

이 있다.

[그림 3-1]은 89T51 기반 라인트레이서의 실제 모습이며, 효율적 교육과 학습을 위해 CPU의 각 입출력 포트와 라인트레이서 주변 장치 간의 연결이 1열 헤더 커넥터를 통해 이루어지도록 설계하였다. 즉, 학습자가 라인트레이서의 학습 수행 상황에 따라 자신이 실습하고자 하는 주변 장치의 헤더 커넥터와 CPU 모듈의 입출력 포트의 헤더 커넥터를 손쉽게 도선으로 직접 연결할 수 있도록 하였다.

실습용 라인트레이서는 단순히 선을 추적하는 기능을 수행하는 것뿐만 아니라 마이크로컨트롤러 응용 기술에 대한 실습이 이루어지도록 7-세그먼트, 조도 센서, 온도 센서 등과 같은 기타 주변 장치가 장착되어 있다.

실습용 89T51 기반 라인트레이서의 특징은 다음과 같다.



**적외선 센서(infrared sensor)**

적외선을 이용해 온도, 압력, 방사선의 세기 등의 물리량 및 화학량을 감지하여 신호 처리가 가능한 전기량으로 변환하는 장치





중앙처리장치  
(CPU, Central Processing Unit)  
컴퓨터의 가장 중요한 부분으로 프  
로그램의 명령을 해독하여 그에 따  
라 실행하는 장치

### ○ 상황에 따른 즉각적인 회로 배선

CPU 모듈의 각 입출력 포트와 라인트레이서에 부착된 다양한 주변 장치의 입출력 포트를 1열 헤더 커넥터를 통해 연결할 수 있도록 구성하여 다양한 실습 상황에 따라 즉각적으로 배선할 수 있다.

### ○ A/D 변환 모듈을 이용하여 적외선 센서, 조도 센서, 온도 센서 활용

적외선 센서, 조도 센서, 온도 센서의 실습에서 A/D 변환 모듈을 활용함으로써 각각의 센서 동작 원리를 이해하고 센서와 A/D 변환 기술의 활용 능력을 습득한다.

### ○ 8개의 LED 및 1개의 7-세그먼트를 이용한 데이터 표시

8개의 LED 및 1개의 7-세그먼트를 이용하여 CPU와 각 주변장치의 상태 값을 2진 및 10진으로 표시하는 기술을 학습한다.

### ○ 2개의 DC 모터

2개의 DC 모터를 이용하여 DC 모터 구동 원리와 라인트레이서 로봇의 제어 원리를 학습한다.

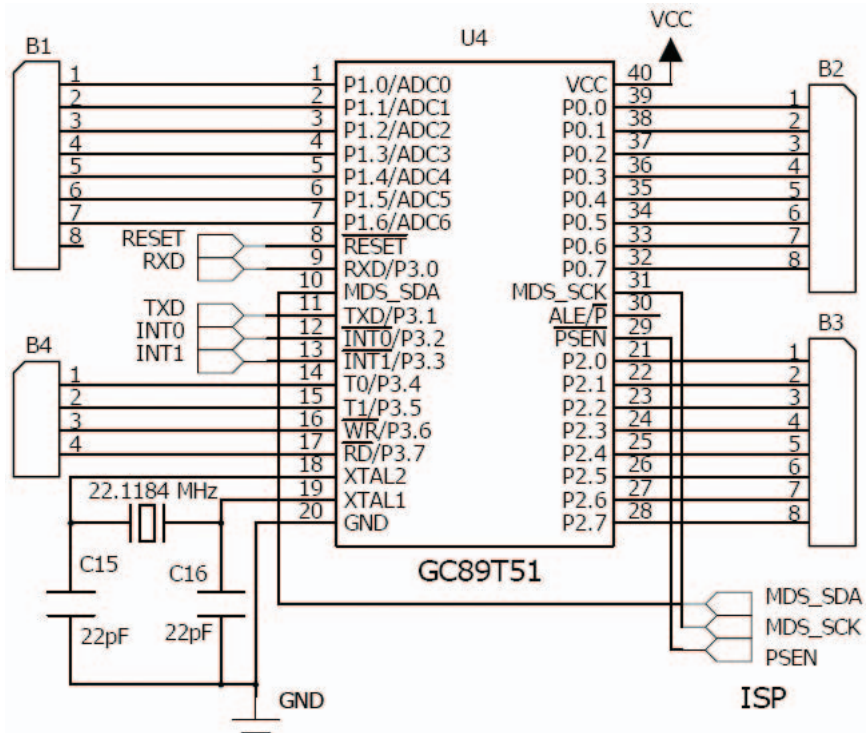
### ○ RS-232 및 ISP 통신

RS-232를 통하여 라인트레이서 로봇의 구동 상태를 확인할 수 있고, ISP 통신을 통하여 사용자 프로그램을 PC로부터 라인트레이서 로봇에 전송할 수 있다.

## 2. 실습용 마이크로로봇의 구성

### (1) CPU 모듈

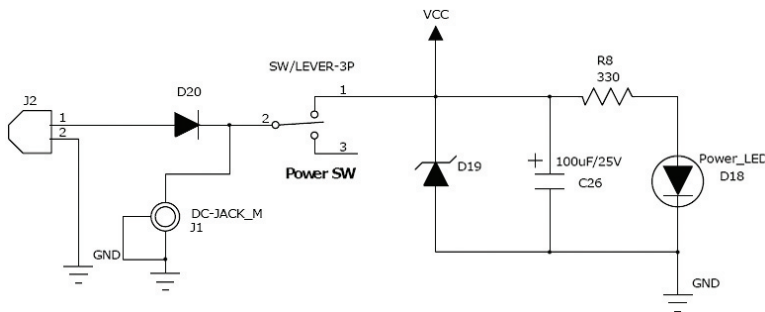
라인트레이서는 ISP 통신을 이용하여 사용자 프로그램을 전송 가능한 플래시 메모리를 내장하고 있는 89T51 마이크로프로세서를 기반으로 설계하였다. 또한, [그림 3-2]와 같이 89T51 마이크로프로세서의 각 입출력 포트에 헤더 커넥터가 연결되어 있기 때문에 실습 상황에 따라 배선을 구성할 수 있다.



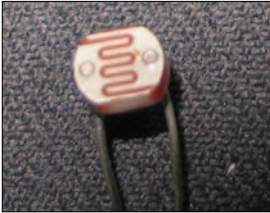
[그림 3-2] 89T51로 구동하는 제어회로

## (2) 전원

89T51 라인트레이서는 전원을 외부에서 공급해야 한다. 전원은 라인트레이서의 우측 하단에 있는 +5V 어댑터나 2핀 모렉스 커넥터를 통해 공급되며, 토글 스위치를 통해 전원을 제어한다.



[그림 3-3] 89T51 라인트레이서 전원 인터페이스



#### CDS

빛의 밝기에 따라 저항 값이 변화하는 반도체이다. 빛이 강할 때는 저항 값이 작아지고, 빛이 약할 때는 저항 값이 커져 암흑 상태에서는 거의 절연 상태에 가까운 값이 된다. Cds 셀은 카메라의 노출계, 가로등의 자동 점멸, 연기의 검지, 광전스 위치 등에 응용되고 있다.

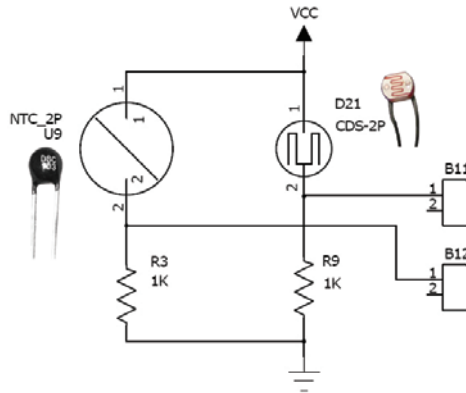


#### NTC

소형의 디스크 형태 써미스터는 에폭시 코팅된 제품으로 주로 온도 감지용과 온도 보상용으로 사용된다. 적정 사용 온도 범위는 자동차 부품과 일반 전자제품에 이상적으로 적용될 수 있는  $-20^{\circ}\text{C}\sim 130^{\circ}\text{C}$  이다. 용도는 온도 감지 및 컨트롤용, 자동 제어용, 전자 시스템, 전자 정보처리 시스템, 지연 회로 등에 사용된다.

### (3) A/D 변환 모듈과 조도 센서 및 온도 센서 인터페이스

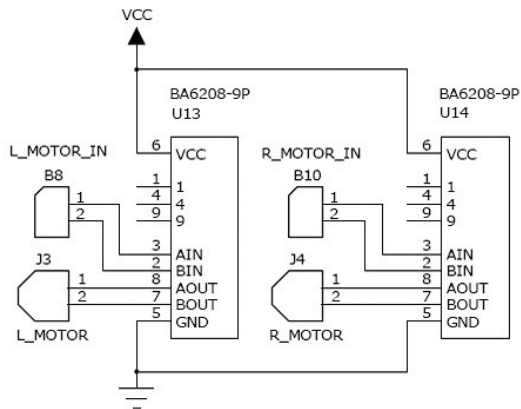
89T51에 내장된 A/D 변환 모듈은 10[Bit]의 분해 기능을 가지고 있다. 아날로그 신호는 CdS 조도 센서인 경우 B11 커넥터를, NTC 온도 센서인 경우 B12 커넥터를 통해 마이크로프로세서로 전달한다. 전달된 아날로그 신호는 내장된 A/D 변환 모듈에 의하여 디지털 신호로 변환된다.



[그림 3-4] 조도 센서, 온도 센서 인터페이스

### (4) RS-232 인터페이스

RS-232 인터페이스의 내부 회로 구성은 [그림 3-5]와 같다.

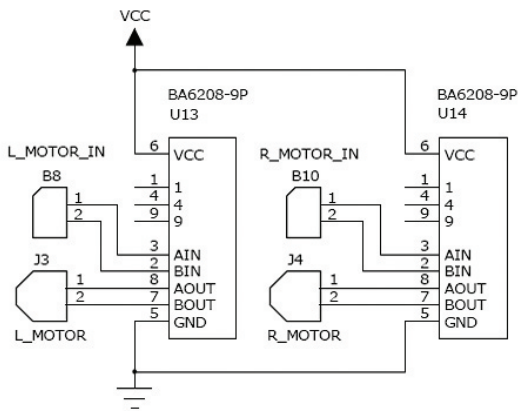


[그림 3-5] RS-232 인터페이스



### (5) 모터 드라이브 인터페이스

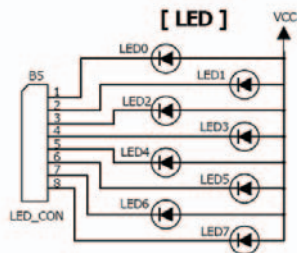
89T51 라인트레이서를 구동하기 위해서 소형 DC 모터 2개를 사용하며, 모터 드라이브는 BA6208을 사용한다. BA6208 모터 드라이브는 최대 500[mA]의 전류 구동 능력을 가지고 있으며, 사용하기 편리하여 5[V] 정도의 소형 모터를 구동하는 데 주로 사용한다.



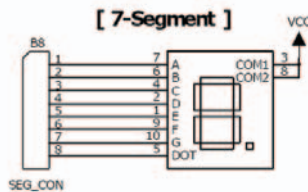
[그림 3-6] BA6208 모터 드라이브 인터페이스

### (6) LED 및 7-세그먼트 인터페이스

89T51 라인트레이서는 LED와 7-세그먼트를 사용하여 각종 주변 장치를 통하여 출력되는 데이터를 외부에서 확인한다. 8개의 LED는 각종 센서의 A/D 변환 결과 값을 2진으로 확인할 수 있으며, 1개의 7-세그먼트는 10진 의 형태로 센서의 A/D 변환 결과 값을 확인할 수 있다.



(a) LED 인터페이스

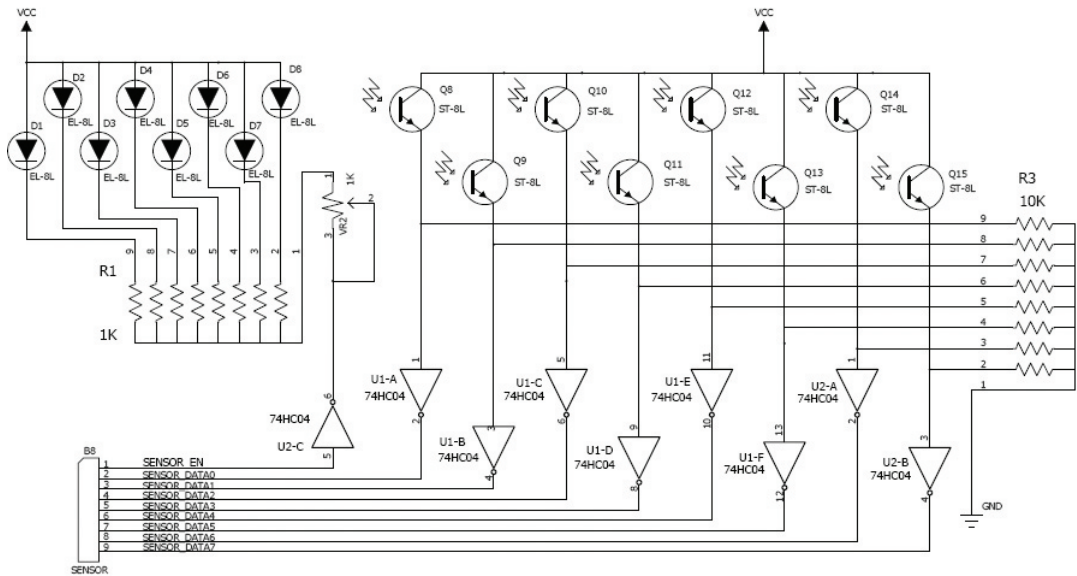


(b) 7-세그먼트 인터페이스

[그림 3-7] LED 및 7-세그먼트 인터페이스

### (7) 적외선 센서 인터페이스

적외선 센서는 바닥에 깔린 선(Line)을 감지하는 도구로 로봇의 눈과 같은 역할을 한다. 89T51에 사용되는 적외선 센서는 모두 8개로 기존의 라인트레이서가 5개의 적외선 센서만 사용하는 것에 비해 3개가 더 추가되어 더욱 정밀하게 선을 추적할 수 있다.

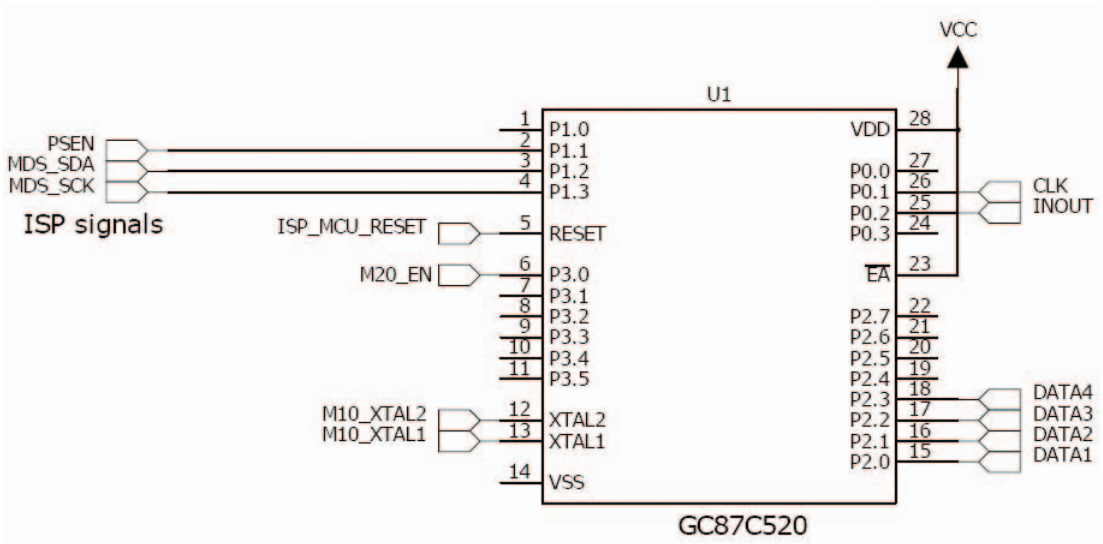


[그림 3-8] 적외선 센서 인터페이스

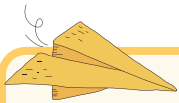
### (8) ISP 통신 인터페이스

89T51 마이크로프로세서는 내부에 플래시 메모리가 탑재되어 있어 사용자가 간단하게 구동 프로그램을 ISP 통신을 통해 라인트레이서로 전송할 수 있다. 이를 위해 89T51 마이크로프로세서는 10번 핀인 MDS\_SDA, 29번 핀인 PSEN, 31번 핀인 MDS\_SCK를 ISP 통신을 위한 인터페이스 핀으로 사용하도록 할당하였다.

또한 89T51 마이크로프로세서의 ISP 신호를 발생시키는 목적을 위하여 GC87C520 ISP 전용칩이 사용된다.



[그림 3-9] ISP 통신 인터페이스



### 쉬어가기

#### 국내 최초의 인조 인간 로봇이 발라드 가수로 데뷔.

한국생산기술연구원은 2006년 10월 18일 서울 삼성동 코엑스에 서 개막된 <2006 로보월드> 전시회에 국내 모 음반사와 작곡가가 새롭게 제작한 발라드곡을 부르는 로봇 <에버투(EveR-2)>를 공개했다. 여성의 모습을 한 에버투는 선 자세로 무릎을 구부리거나 팔을 흔드는 등 가벼운 울동과 함께 발라드곡에 맞춰 입모양을 모음과 자 음에 따라 자유자재로 바꾸면서 노래를 한다. 또한 <에버투>는 명랑 한 성격과 우울한 성격 등 대조적인 2가지 성격을 갖고 있으며, 각 성격에 따라 음성톤과 얼굴 표정이 변화한다. 이밖에 <에버투>는 네 트워크와 연결돼 다양한 인간의 문장을 이해하고, 사람과 대화하며 감정을 표현할 수 있다. <에버투>는 머리부터 발끝까지 전신을 실리콘 피부로 처리했다.





## 02. 라인트레이서 조립하기

### ROBOT CONTROL SYSTEM

#### 학습목표



- 각종 회로 소자들의 특성과 조립할 때 유의점에 대해 말할 수 있다.
- 인쇄된 기판(PCB)에 회로 소자를 조립하여 제어 보드를 만들 수 있다.
- 라인트레이서의 기계적인 부분을 조립할 수 있다.

#### 1. 라인트레이서 조립 순서

실습용 라인트레이서는 일반적으로 다음과 같은 15단계 순으로 조립한다.

- 1단계 : 89T51 라인트레이서의 제어보드 준비
- 2단계 : 라인트레이서 제작에 필요한 전기/전자 및 기타 부품의 준비
- 3단계 : 저항 조립
- 4단계 : 다이오드 조립
- 5단계 : 콘덴서 조립
- 6단계 : LED 조립
- 7단계 : 적외선 센서 조립
- 8단계 : 크리스털 조립
- 9단계 : 7-세그먼트 조립
- 10단계 : IC 소켓 조립
- 11단계 : 가변저항 조립
- 12단계 : 전원 단자, 스위치 및 커넥터 조립

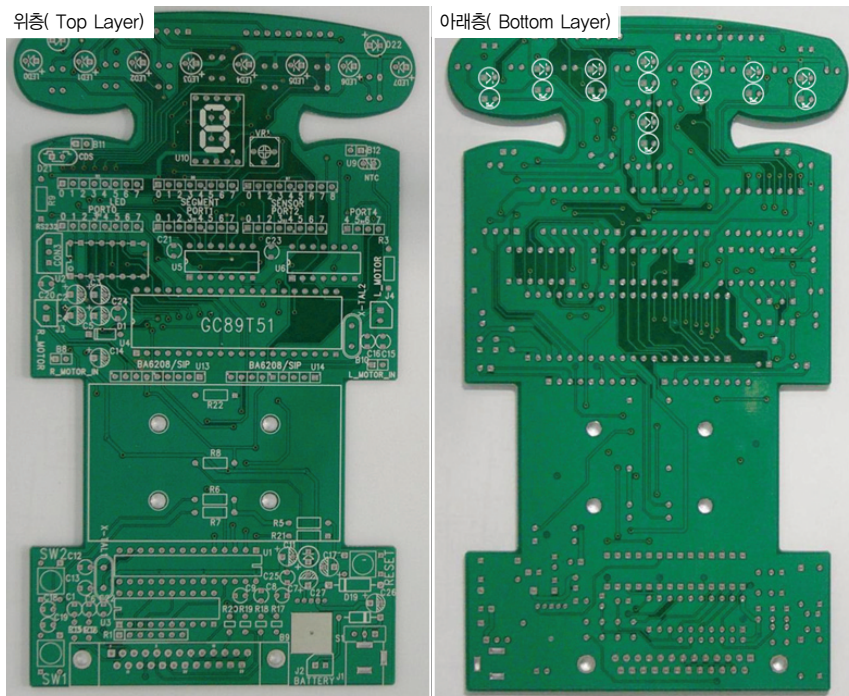


- 13단계 : CdS 조도 센서와 NTC 서미스터 온도 센서 조립
- 14단계 : 어레이 저항 조립
- 15단계 : IC 및 DC 모터 드라이브 조립

## 2. 라인트레이서 조립하기

### 【1단계】 89T51 라인트레이서의 제어 보드 준비

89T51로 구동하는 라인트레이서는 [그림 3-10]과 같이 하나의 기판에 양면(2 Layer)으로 설계되었기 때문에 기판의 윗면(Top Layer)과 아랫면(Bottom Layer)에 라인트레이서를 제어하기 위한 CPU와 적외선 센서를 포함한 주변 장치와 전원 회로가 모두 포함되어 있다.



[그림 3-10] 89T51 라인트레이서의 PCB

## 【2단계】 라인트레이서 제작에 필요한 전기/전자 및 기타 부품의 준비

실습용 라인트레이서를 제작하기 위해 [표 3-1]의 부품들을 준비 한다.

[표 3-1] 라인트레이서 제작용 부품 리스트

연번	부 품 명	규 격	부품번호	수 량
1	조도 센서	CdS 3 $\phi$	D21	1개
2	전해 콘덴서	100[uF]	C26	1개
3	전해 콘덴서	22[uF]	C2~5	4개
4	전해 콘덴서	47[uF]	C17, C27	2개
5	전해 콘덴서	1[uF]	C11, C14	2개
6	모노 콘덴서	0.1[uF]	C18~25,	8개
7	세라믹 콘덴서	22[pF]	C12, C13, C15, C16	4개
8	세라믹 콘덴서	470[pF]	C1, C6, C7, C8, C9	5개
8	다이오드	1N4148	D1, D20	2개
9	다이오드	1N4734	D19	1개
10	LED	적색 5 $\phi$	D17~18, D22, LED0~7	11개
11	적외선 발광 센서	3 $\phi$ (EL-8L)	D2~9	8개
12	적외선 수광 센서	3 $\phi$ (ST-8L)	Q8~15	8개
13	수컷 커넥터	D-SUB 25핀	B9	1개
14	DC-Jack	DC-Jack	J1	1개
15	Box 헤더 핀	4핀	B4	1개
16	Box 헤더 핀	2핀	B8, B10, B11, B12	4개
17	Box 헤더 핀	5핀	U10 (FND 소켓)	2개
18	Box 헤더 핀	8핀	B1~3, B5, B6	5개
19	Box 헤더 핀	9핀	B7	1개
20	모렉스 2핀 커넥터	수컷, 2핀 (5264-02)	J2~4	3개
21	저항	330[ $\Omega$ ]	R8	1개
22	저항	1[k $\Omega$ ]	R3, R9	2개
23	저항	10[k $\Omega$ ]	R6, R7, R21, R22	4개
24	저항	4.7[k $\Omega$ ]	R5	1개



연번	부 품 명	규 격	부품 번호	수 량
25	저항	100[Ω]	R15~20	6개
26	푸시 스위치	TACK S/W-1230A(5.0m/m)	RESET, SW1, SW2	3개
27	토글 스위치	AT1D-2M3	S1	1개
28	소켓	Round type 9핀	모터드라이버용	2개
29	소켓	Round type 3핀	Power s/w	1개
30	소켓	DIP 14핀		2개
31	소켓	DIP 28핀		1개
32	소켓	DIP 40핀		1개
33	소켓	DIP 16핀		1개
34	소켓	DIP 20핀		1개
35	IC	89T51	U4	1개
36	IC	MAX232	U2	1개
37	IC	74HC04	U5, U6	2개
38	IC	74LS244, 3-state buffer	U3	1개
39	IC	GC87C520	U1	1개
40	IC	BA6208, 모터 드라이버	U13, U14	2개
41	온도센서	NTC, 2핀, Straight	U9	1개
42	7-세그먼트	HSD-5501(H-)	U10	1개
43	레귤레이터	MIC29301-5	U7	1개
44	크리스탈	22.1184[Mhz]	X-tal1, X-tal2	2개
45	고무바퀴(휠 포함)	고무바퀴 大		2개
45	DC 모터 1조	DC 기어드 모터 1조		1조
47	PCB	라인트레이서 PCB		1개
48	나사	3Φ [mm]		4개
49	가변 저항	1[kΩ] 가변 저항	VR2	1개
50	어레이 저항	9핀 10[kΩ] 어레이 저항(A103)	R4	1개
51	어레이 저항	7핀 10[kΩ] 어레이 저항	R1	1개
52	어레이 저항	9핀 1[kΩ] 어레이 저항(A102)	R2	1개
53	모렉스 3핀 커넥터	모렉스 3핀, Angle, 5046-03	CON3 RS232C	1개



[그림 3-11] 색띠 저항의 의미

### 【3단계】 저항 조립

#### 1) 저항의 개요

저항은 전기의 흐름을 방해하는 성질을 가진 수동 소자이다. 저항은 외부에 4개의 색 띠들을 표시하여 저항 값과 오차율을 나타낸다. 각각의 색은 [표 3-2]의 수치들을 의미한다.

[표 3-2] 저항의 색 띠들이 나타내는 수치

색	수치	승수	정밀도(오차율, %)	온도계수 $10^{-6}/^{\circ}\text{C}$
흑(검정)	0	$10^0$	-	$\pm 250$
갈(갈색)	1	$10^1$	$\pm 1$	$\pm 100$
적(적색)	2	$10^2$	$\pm 2$	$\pm 50$
등(주황)	3	$10^3$	$\pm 0.05$	$\pm 15$
황(노랑)	4	$10^4$	-	$\pm 25$
녹(녹색)	5	$10^5$	$\pm 0.5$	$\pm 20$
청(청색)	6	$10^6$	$\pm 0.25$	$\pm 10$
자(보라)	7	$10^7$	$\pm 0.1$	$\pm 5$
회(회색)	8	$10^8$	-	$\pm 1$
백(흰색)	9	$10^9$	-	-
금(금색)	-	$10^{-1}$	$\pm 5$	-
은(은색)	-	$10^{-2}$	$\pm 10$	-
무	-	-	$\pm 20$	-

[표 3-2]를 바탕으로 실제 저항 값을 읽는 방법은 다음과 같다.



#### 예 제

색띠 저항에서 첫 번째 색부터 네 번째 색이 각각 적색, 보라색, 주황색, 금 색인 경우, 저항값과 오차율을 구하시오.



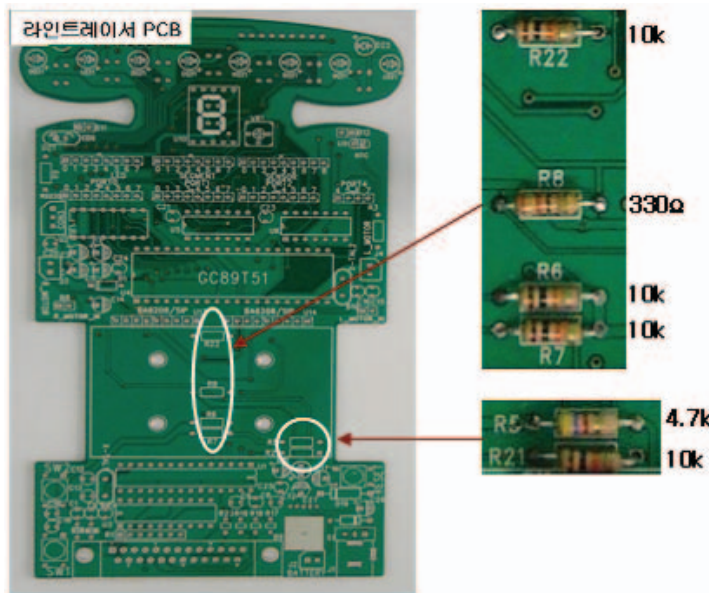
실습용 라인트레이서의 경우 모두 5가지 저항값을 사용하며, 사용되는 각각의 저항 값과 저항의 색 띠는 [표 3-3]과 같다.

[표 3-3] 89T51로 구동하는 라인트레이서에서 사용하는 저항

저항 값	저항 색 띠			
	첫 번째 색	두 번째 색	세 번째 색	네 번째 색
330[Ω]	주황색	주황색	갈색	관계없음
4.7[kΩ]	황색	자색	적색	관계없음
10[kΩ]	갈색	흑색	주황색	관계없음
1[kΩ]	갈색	흑색	적색	관계없음
100[Ω]	갈색	흑색	갈색	관계없음

## 2) 저항 조립하기

이제 [표 3-3]을 참고하여 저항값이 각각 330[Ω], 4.7[kΩ], 10[kΩ]을 찾아 라인트레이서 기판에 조립한다.



[그림 3-12] 저항 조립 위치

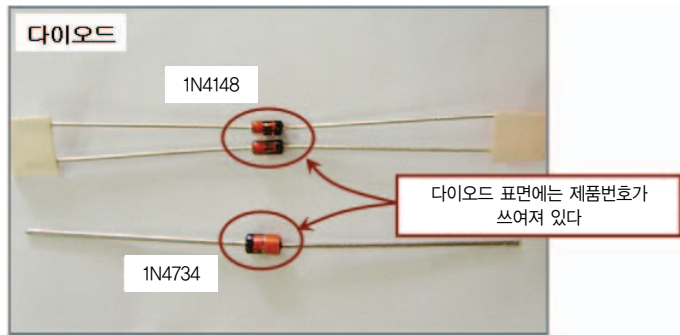


라인트레이서 PCB 뒷면의 각각의 저항이 위치해야 할 곳마다 저항 값이 기재되어 있다. 각각의 저항 값을 찾아 PCB에 기재된 위치에 조립하면 보다 쉽게 조립할 수 있다.

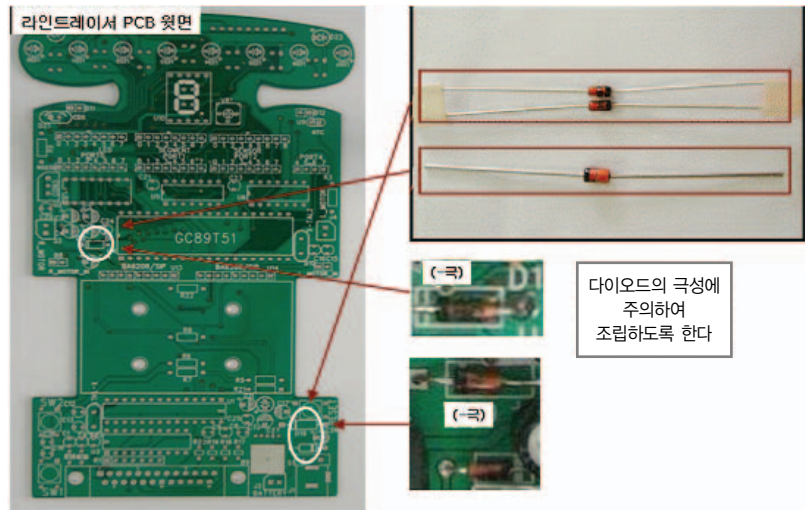


#### 【4단계】 다이오드 조립

다이오드는 다양한 형태가 있다. 실습용 라인트레이서에는 2가지 종류의 다이오드(1N4734, 1N4148)를 사용한다. 다이오드의 극성은 투명한 유리관에 검정선이 그어져 있는 부분이 캐소드(-) 극성을 나타낸다. 라인트레이서 기판의 다이오드 접합 부분에는 흰색 수직선이 캐소드(-) 극을 나타내므로 기판에 조립할 때에는 반드시 극성을 고려 하여야 한다.



[그림 3-13] 다이오드의 외형



[그림 3-14] 다이오드의 조립 방법



## 【5단계】 콘덴서 조립

### 1) 콘덴서의 개요

콘덴서는 전기 에너지를 저장하는 수동 소자이다. 만들어진 재료에 따라 전해 콘덴서, 필름 콘덴서, 세라믹 콘덴서, 탄탈 콘덴서, 모노 콘덴서 등으로 구분한다.

실습용 라인트레이서는 세라믹 콘덴서, 모노 콘덴서, 전해 콘덴서를 사용한다.

전해 콘덴서는 극성(+극 : 다리가 긴 쪽, -극 : 다리가 짧은 쪽)이 있으므로 반드시 극성을 고려하여 조립하여야 한다.

### 2) 콘덴서의 용량 판독

콘덴서는 종류별로 용량을 표기하는 방법에 있어 다소 차이가 있는데, 콘덴서 몸체에 인쇄된 숫자를 통해 확인할 수 있다.

#### ① 세라믹 콘덴서, 모노 콘덴서

- 용량표기 : (숫자1)(숫자2)(숫자3)(영문기호)
- 용량계산 : (숫자1)(숫자2) × 10<sup>(숫자3)</sup> [pF] (오차율)
- 단, 2자리 숫자로만 표기되어 있으면 단순히 [pF]을 기본 단위로 하여 용량을 읽는다.



#### 예제

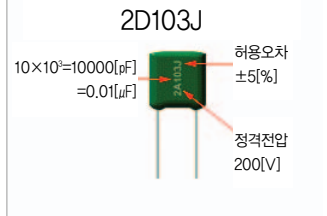
콘덴서 몸체에 '104J' 값이 표기되어 있는 경우, 용량과 허용 오차를 구하십시오.

#### ② 전해 콘덴서

전해 콘덴서는 콘덴서 용량이 몸체 표면에 직접 표시되어 있으므로 이 값을 읽으면 된다. 즉, 몸체 표면에 '100μF' 이 표시되어 있는 경우 콘덴서 용량은 100[μF]이 된다.

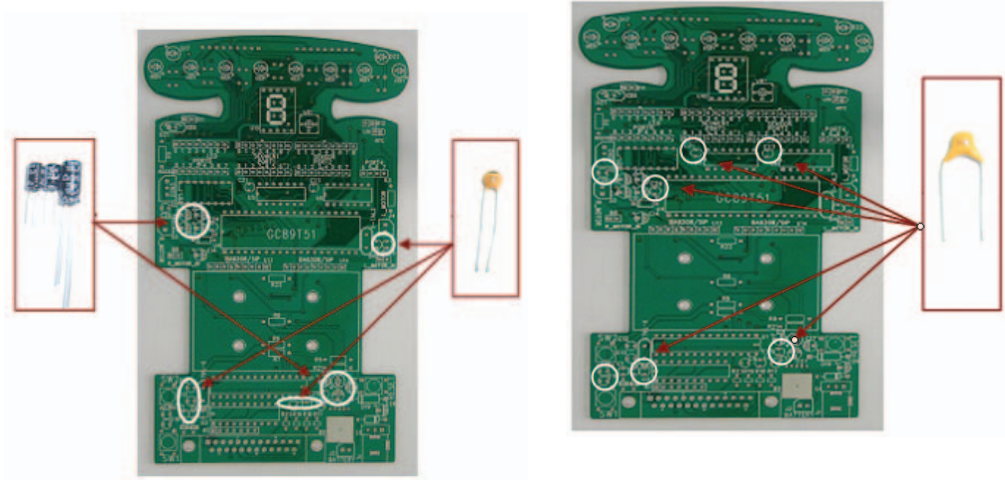


[그림 3-15] 89T51 라인트레이서에 사용되는 콘덴서



### 3) 콘덴서 조립하기

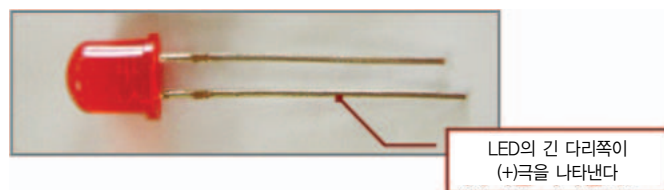
89T51 라인트레이서의 콘덴서 부분은 [그림 3-16]과 같은 위치에 조립한다.



[그림 3-16] 콘덴서 조립 위치

### 【6단계】 LED(발광 다이오드) 조립

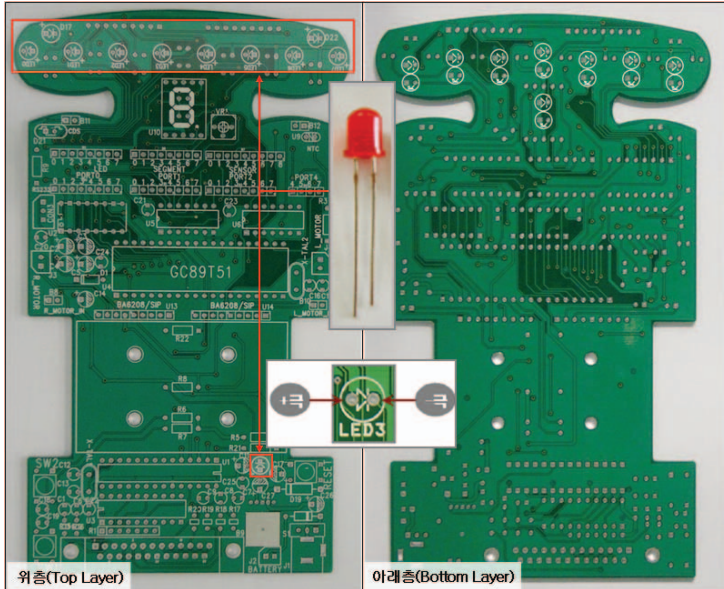
LED는 전류가 흐르면 빛을 방출하는 발광 다이오드를 말한다. 또한 LED는 [그림 3-17]과 같이 전기적 극성이 있으므로 애노드(+)극과 캐소드(-)극을 고려하여 조립하여야 한다.



[그림 3-17] LED(발광 다이오드)의 외형



89T51 라인트레이서는 기판의 윗면(Top Layer)과 아랫면(Bottom Layer) 모두에 LED가 부착되므로 이에 유의하여 [그림 3-18]과 같이 조립한다.

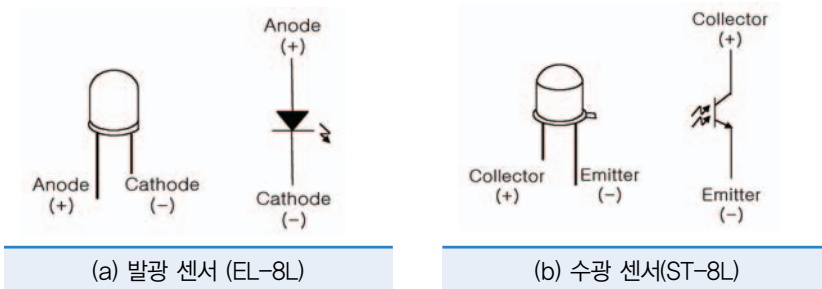


[그림 3-18] LED(발광 다이오드)의 조립 위치

### [7단계] 적외선 센서 조립

로봇에는 전방의 벽을 감지하거나 선을 감지하기 위해 센서를 부착한다.

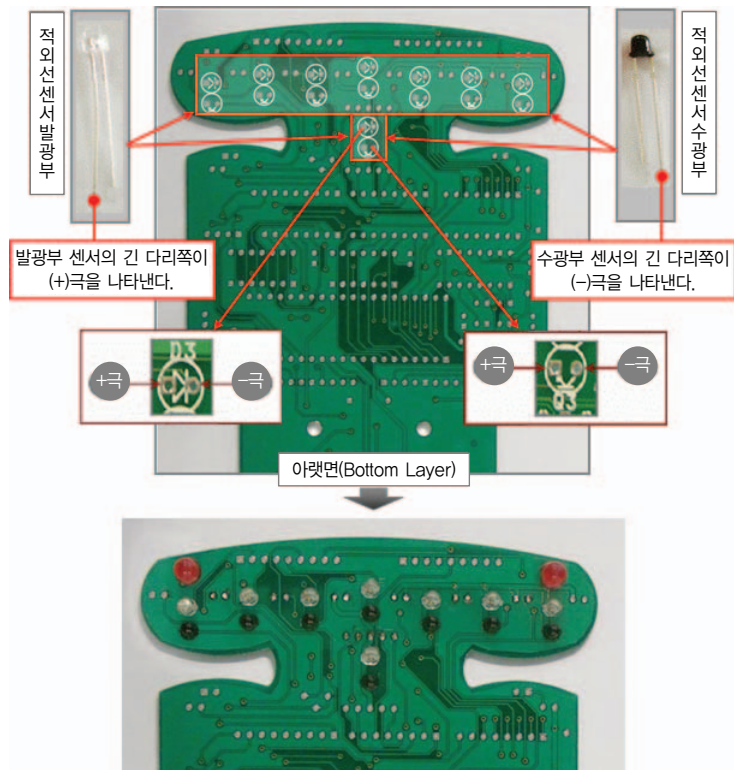
89T51 라인트레이서는 8개의 적외선 센서를 사용한다. 적외선 센서는 일정한 주파수의 빛을 발산하는 발광부와 발산된 빛을 받아들이는 수광부로 구성되며, 회로 상에서는 [그림 3-19]와 같은 기호로 표시한다.



[그림 3-19] 적외선 발광 센서와 수광 센서의 외형도와 회로 기호

적외선 센서는 전기적 극성(다리가 긴 쪽이 (+)극)을 띠고 있기 때문에 적외선 센서를 조립할 때 반드시 극성을 고려하여야 한다. 적외선 센서는 다리가 긴 쪽이 (+)극, 다리가 짧은 쪽이 (-)극이 된다.

이와 같은 적외선 센서의 특징을 고려하여 라인트레이서 기판의 아랫면 (Bottom Layer)에 적외선 센서를 조립한다.



[그림 3-20] 적외선 센서부 조립

### 【8단계】 크리스탈 조립

크리스탈은 특정 주파수의 펄스를 발생시키는 발진소자로서 인간의 심장 역할을 담당하는 중요한 부품이다.

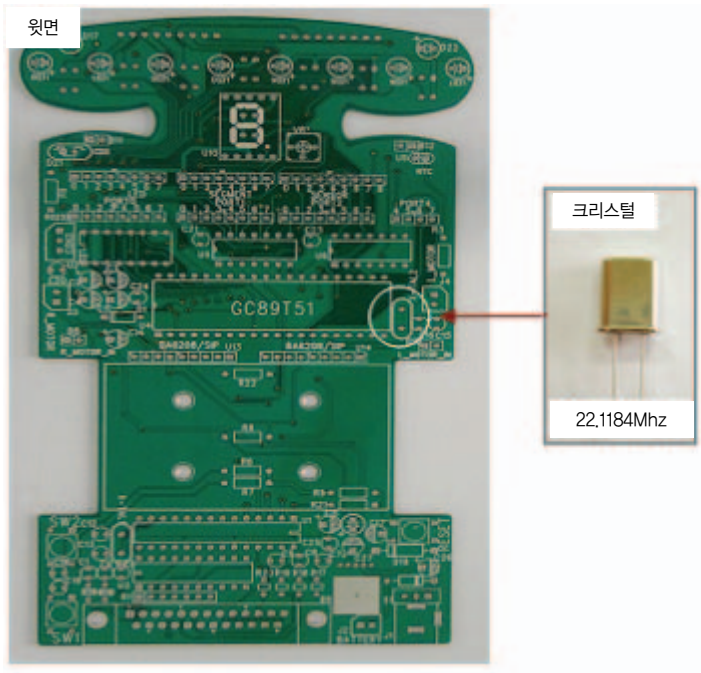
89T51 라인트레이서는 22.1184[MHz]의 주파수를 갖는 크리스탈을 사용하며, 극성이 없으므로 어느 방향으로 조립을 해도 무관하다. 그러나 라인트레이서의 모든 조립 과정이 끝나고 검사 과정에서 프로그램의 전송이 제대





로 이루어지지 않으면 크리스털의 조립 상태를 우선적으로 살펴야 한다. 이는 인간의 심장과도 같은 크리스털이 제대로 동작하지 못한다면 CPU가 동작하지 못하기 때문이다.

크리스털을 [그림 3-21]과 같이 89T51 라인트레이서에 조립한다.



[그림 3-21] 크리스털 조립 위치

### 【9단계】 7-세그먼트 조립

7-세그먼트(Segment)는 7개의 발광 다이오드를 이용해 0부터 9까지의 숫자와 A에서 F까지의 문자를 표현하는 소자이다. 하지만 실제의 7-세그먼트에는 소수점(DP : Decimal Point)을 표시하는 발광 다이오드가 하나 더 있기 때문에 총 8개의 발광 다이오드를 내장하고 있다.

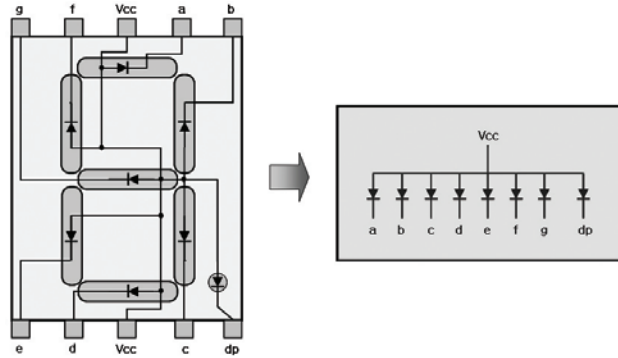
7-세그먼트에는 발광 다이오드의 접속 방식에 따라 애노드 타입(Anode Type)과 캐소드 타입(Cathode Type)이 있다.

- **캐소드 타입** : 발광 다이오드의 캐소드(Cathode) 측이 공통(Common)으로 접속되어 있다.



○ 애노드 타입 : 발광 다이오드의 애노드(Anode) 측이 공통(Common)으로 접속되어 있다.

89T51 라인트레이서에는 [그림 3-22]와 같은 구조를 갖는 애노드 타입 7-세그먼트를 사용한다.

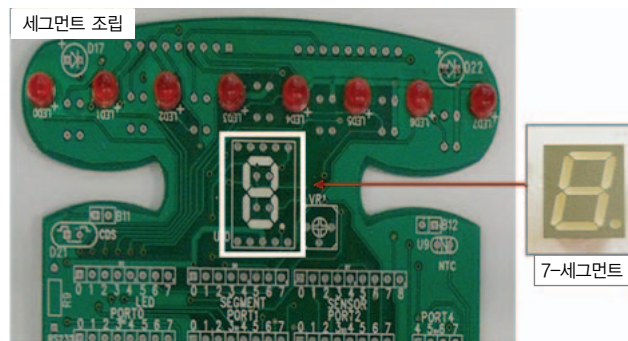


[그림 3-22] 애노드 형태의 7-세그먼트 내부 구조



[그림 3-23] 상용 제품의 7-세그먼트 형태

[그림 3-24]와 같이 7-세그먼트를 89T51 라인트레이서 기판에 조립한다.



[그림 3-24] 7-세그먼트 조립 위치



## 【10단계】 IC 소켓 조립

### 1) IC 소켓의 개요

89T51 라인트레이서는 각종 주변 장치와 라인트레이서의 구동을 제어하는 IC 칩을 손쉽게 장착하기 위해 IC 소켓을 사용한다. IC 소켓은 모양과 크기에 따라 그 종류가 다양하므로 각각의 IC 칩 크기에 맞는 소켓을 선택하여 사용하여야 한다.



[그림 3-25] 여러 가지 형태의 집적 회로(IC)의 형태

[그림 3-25]의 ①, ② 형태의 IC는 일반적으로 많이 사용되는 IC로, DIP 타입 IC라고 한다. 반면 ③, ④ 형태의 IC는 CPU와 같이 고성능에 주로 사용되는 IC로, PLCC 타입 또는 TQFP 타입 IC라 한다.

핀과 핀 사이의 간격을 나타내는 단위로는 밀[mil]을 사용하며, 1[mil]은 1000분의 1 인치(Inch)에 해당한다. 일반적인 IC의 핀과 핀 사이의 간격은 100[mil]이고, 우리가 흔히 전자 회로를 만들기 위해 사용하는 만능 기판의 홀과 홀의 간격도 100[mil]이다. 또한, IC의 각 핀에는 고유한 번호가 있어 각각의 핀을 구분한다.

### 2) IC 소켓 조립하기

89T51 라인트레이서는 5 종류의 IC 소켓을 사용한다.

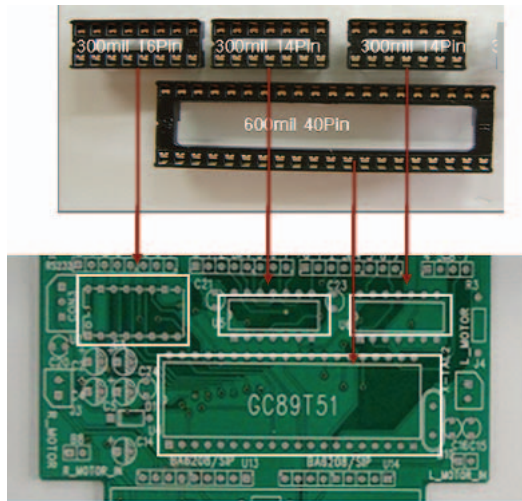
- 규격=(소켓 폭, 핀 간격, 핀 수)=(300mil, 100mil, 14) : 2개 사용
- 규격=(소켓 폭, 핀 간격, 핀 수)=(300mil, 100mil, 16) : 1개 사용
- 규격=(소켓 폭, 핀 간격, 핀 수)=(300mil, 100mil, 20) : 1개 사용
- 규격=(소켓 폭, 핀 간격, 핀 수)=(300mil, 100mil, 28) : 1개 사용
- 규격=(소켓 폭, 핀 간격, 핀 수)=(600mil, 100mil, 40) : 1개 사용

IC 소켓을 조립할 때 [그림 3-26]과 같이 IC 소켓의 위쪽에 위치한 홈과 89T51 라인트레이서 기판에 인쇄된 칩 모양의 그림에 위치한 홈이 서로 일치되도록 하여야 한다. 이렇게 하면 IC 칩을 IC 소켓에 끼울 때 IC 칩의 장착 방향을 손쉽게 알 수 있다.



[그림 3-26] IC 소켓에 IC 칩 장착 방법

이제 [그림 3-27]과 같이 89T51 라인트레이서 기판에 IC 소켓을 조립한다.



[그림 3-27] IC 소켓 조립 위치

### [11단계] 가변 저항 조립

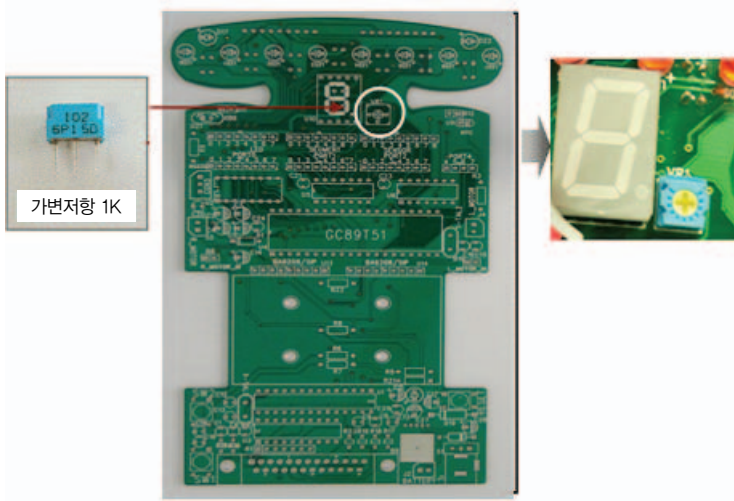
가변 저항은 일반적인 저항과 같은 역할을 수행하지만 저항의 값을 정해진 범위에서 조정할 수 있다는 차이점이 있다. 가변 저항은 위쪽의 (+) 모양의 홈을 십자 드라이버를 사용하여 좌우로 회전시키면 저항 값이 변화하게 된다. 가변 저항이 가질 수 있는 최대값은 가변 저항 측면에 인쇄된 숫자를 통해 알 수 있다.



## 예제

가변 저항 표면에 기록된 숫자가 '102' 인 경우 가변 저항이 가질 수 있는 최대값을 계산하여 보자.

[그림 3-28]과 같이 89T51 라인트레이서에 가변 저항을 조립한다.



[그림 3-28] 가변 저항 조립 위치

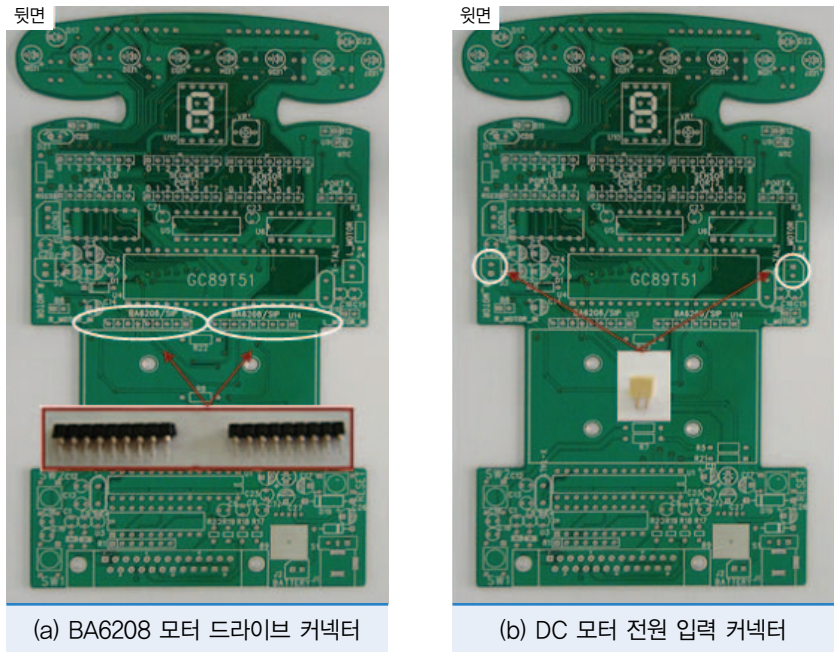
### 【12단계】 전원 단자, 스위치 및 기타 커넥터 조립

89T51 라인트레이서는 2핀 모렉스 커넥터와 DC-JACK을 사용하여 외부로부터 전원을 공급받고, 다시 1열 커넥터를 통해 주변 장치로 배분한다.

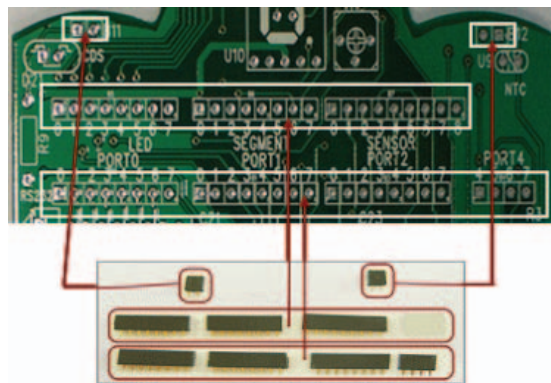
89T51 라인트레이서가 PC와 RS-232 통신을 하기 위해 3핀 모렉스 커넥터를 사용하며, 실습 상황에 따라 라인트레이서의 CPU와 각종 주변 장치(DC 모터, 온도 센서, 조도 센서, LED, 7-세그먼트, 89T51) 간의 연결을 쉽게 구성할 수 있도록 CPU의 입출력 포트와 각 주변 장치의 입출력 포트에 1열 헤더 커넥터를 사용한다.

또한 푸시 버튼 스위치는 전원을 ON/OFF하고 CPU를 리셋하는 기능을 수행하기 위해 사용한다.

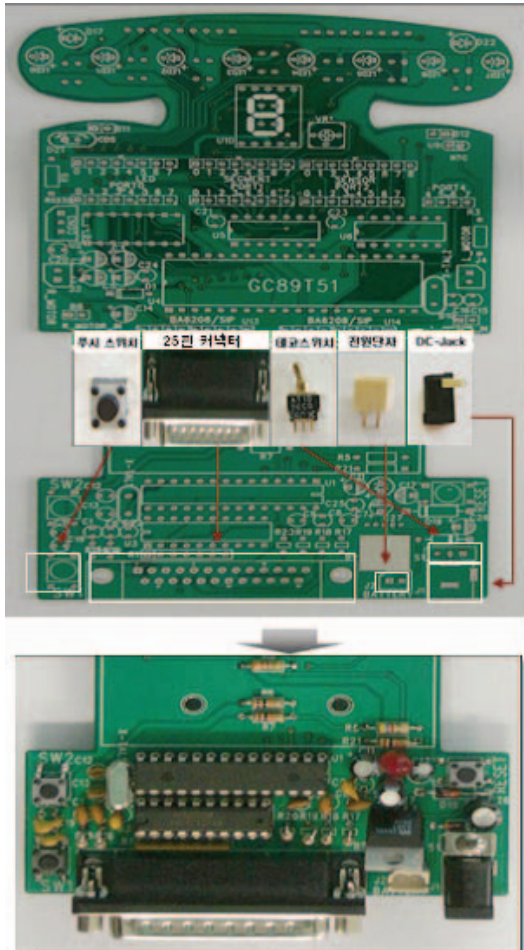
이제 [그림 3-29]에서부터 [그림3-31]의 순서대로 라인트레이서의 커넥터 및 스위치를 조립한다.



[그림 3-29] DC 모터 드라이브 소켓 및 모터 전원 입력 커넥터



[그림 3-30] 89T51의 포트 및 주변 장치의 커넥터 연결



[그림 3-31] 기타 스위치 및 전원 단자 조립

## 【13단계】 CdS 조도 센서와 NTC 서미스터 온도 센서 조립

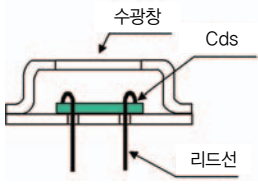
### 1) 조도 센서와 온도 센서의 특징

#### ① 조도 센서

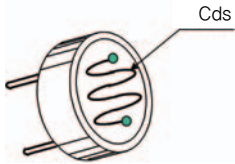
조도 센서는 N형 반도체에 속하는 소자로서 조도(빛의 닿는 세기)에 따라 전기 저항이 달라지는 일종의 가변 저항이다. 이러한 조도 센서는 조도에 따라 저항 차이를 발생시키고자 저항체 소재로 CdS(황화카드뮴) 혹은 CdSe(셀렌화카드뮴)를 사용하거나 이 둘을 적당한 비율로 혼합하여 사용한다.

89T51 라인트레이서에는 CdS(황화카드뮴)를 사용한 조도 센서가 부착되





(a) CdS 광도전 센서의 구조도



(b) CdS 광도전 센서의 외형도

[그림 3-32]  
조도 센서(CdS) 구성도

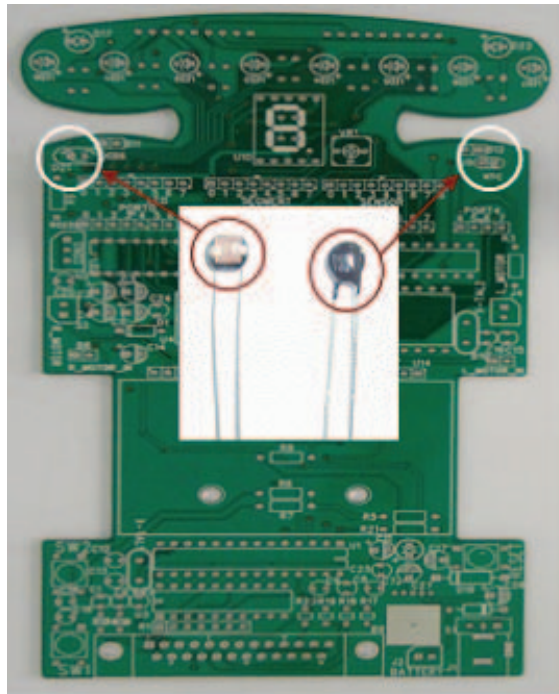
어 있다. [그림 3-32]는 이러한 CdS 조도 센서의 구조와 외형도이다. CdS 조도 센서는 [그림 3-32]와 같이 CdS(황화카드뮴)가 밀폐 용기에 담겨져 있으며, 수광창은 투명한 플라스틱으로 되어 있고, 외부로는 전기적 극성이 없는 2개의 리드선이 나와 있다.

## ② 온도 센서

온도 센서는 온도 감지기라는 뜻으로, 온도 변화에 따라 소자의 전기 저항이 변화하는 성질을 가진 소자이다. 온도 센서의 온도 측정 범위는 대략  $-50[^\circ\text{C}] \sim 350[^\circ\text{C}]$  정도이며, 외형은 용도에 따라 다양한 형태를 가진다.

89T51 라인트레이서에서는 직경 1.5[mm] 정도의 작은 원통에 반도체를 붙이고 극성이 없는 두 개의 리드 선을 가진 비드형이 사용된다. 또한, 비드형 온도 센서는 상온에서 1[KΩ]의 저항 값을 갖는 것을 이용한다.

[그림3-23]과 같이 온도 센서와 조도 센서를 조립한다.



[그림 3-33] 조도 센서와 온도 센서 조립 위치



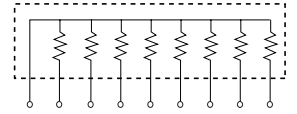
### 【14단계】 어레이 저항 조립

어레이 저항은 일반 저항과 같은 역할을 수행하나, 여러 개의 저항을 연결한 것이다. 어레이 저항의 내부 구조는 [그림 3-34]와 같다.

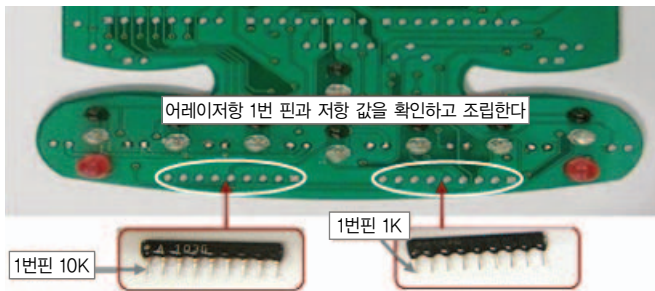
89T51 라인트레이서에 사용되는 어레이 저항은 모두 [그림 3-34]와 같은 구조를 갖으며, 필요에 따라 연결된 저항의 갯수를 맞추어 사용한다.

[그림 3-35]는 어레이 저항을 라인트레이서에 조립하는 과정이다. 이때 어레이 저항의 저항 값과 어레이 저항의 핀 방향을 주의해야 한다. 어레이 저항의 저항 값 계산 방법은 가변 저항과 동일하다. 어레이 저항의 1번 핀은 표면에 점(Dot)이 인쇄되어 있다.

어레이 저항의 특징을 바탕으로 [그림 3-35]와 같이 어레이 저항을 조립한다.

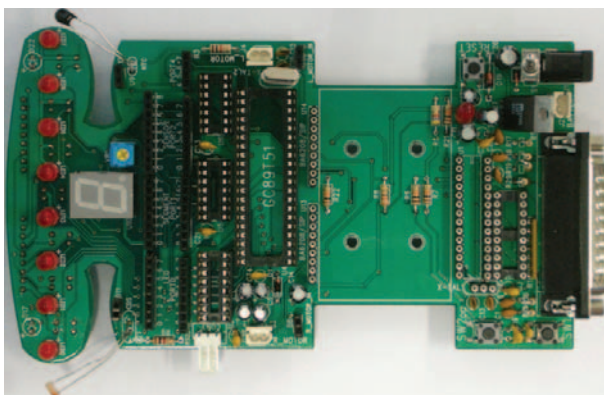


[그림 3-34]  
어레이 저항의 내부 구조



[그림 3-35] 어레이 저항 조립

1단계에서부터 14단계까지의 모든 조립 과정을 성공적으로 수행하였다면 [그림 3-36]과 같이 라인트레이서의 제어보드로서 기본적인 모습을 갖추게 된다.



[그림 3-36] 라인트레이서 제어보드 완성 형태

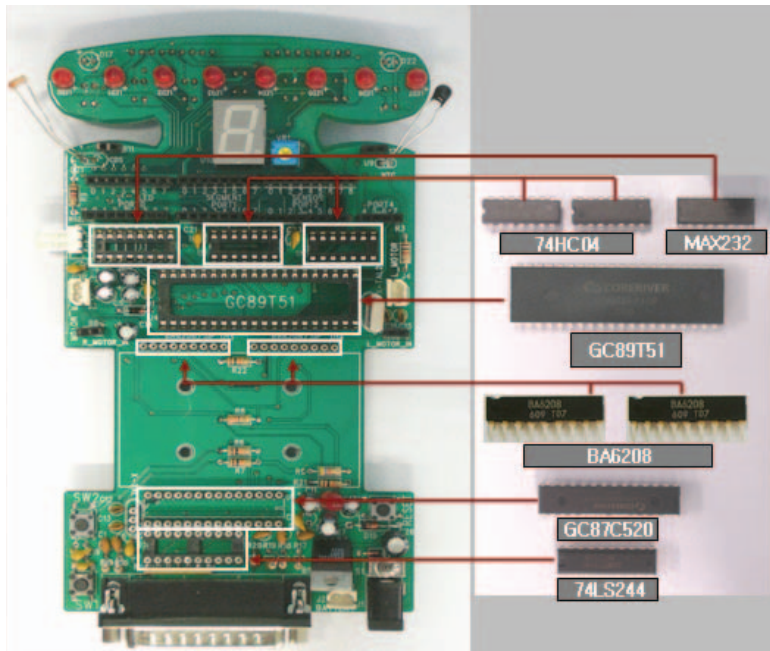
### 【15단계】 IC 및 DC 모터 드라이브 조립

15단계에서는 89T51 라인트레이서 제어보드에 위치한 각각의 IC 소켓에 해당 IC 칩을 삽입하고 DC 모터를 조립한다.

89T51 라인트레이서에서 사용하는 IC 부품은 89T51(CPU), MAX232(RS-232 통신 인터페이스 칩), 74HC04(인버터), 74LS244(버퍼), 87C520(ISP 신호 발생 칩) 등 5가지이다. IC 부품은 각각의 IC 칩을 단순히 IC 소켓의 홈과 IC 칩 윗면의 홈이 서로 일치하도록 삽입하면 된다.

또한, DC 모터를 구동하기 위한 드라이버로 BA6208을 사용하는데, 이는 BA6208 전면의 홈이 소켓의 1번 핀 방향으로 향하게 삽입한다.

IC 및 DC 모터 드라이브를 [그림 3-37]과 같이 조립한다.



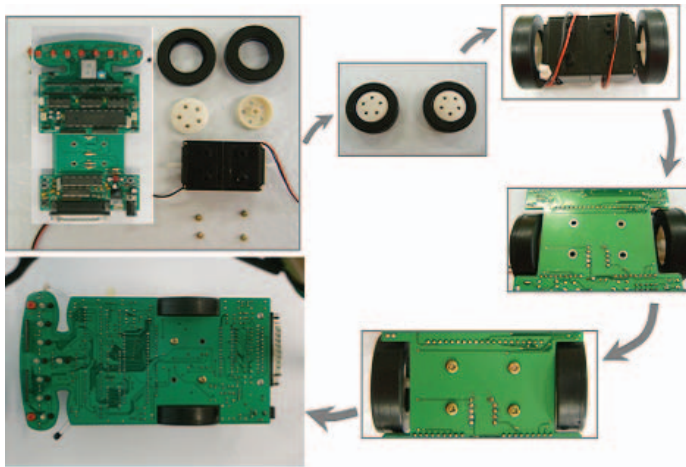
[그림 3-37] IC 칩 및 DC 모터 드라이브 조립

### 【16단계】 DC 모터 조립

89T51 라인트레이서는 바퀴 구동을 위해 5[V] 전압에서 동작하는 2개의 소형 DC 모터를 사용한다. 라인트레이서의 바퀴를 조립하려면 DC 모터 1조와 바퀴 1조, 나사 30크기 4개가 필요하다.

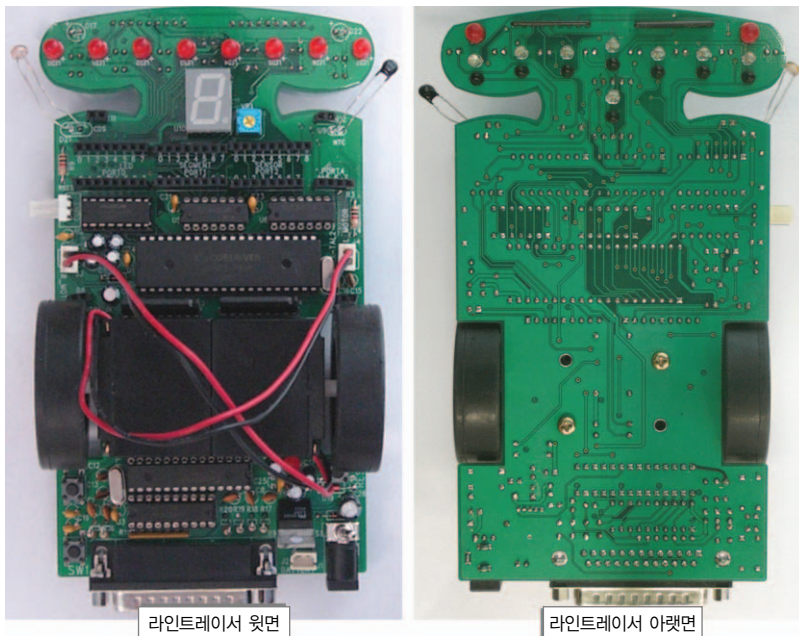


[그림 3-38]과 같이 DC 모터를 조립한다.



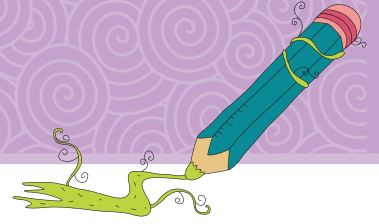
[그림 3-38] 라인트레이서 DC 모터 조립 과정

지금까지 89T51 라인트레이서의 모든 조립 작업을 수행하였다. [그림 3-39]는 조립이 완료된 89T51 라인트레이서의 모습이다.



[그림 3-39] 89T51 라인트레이서의 조립 완성 모습

# 단원 학습 정리

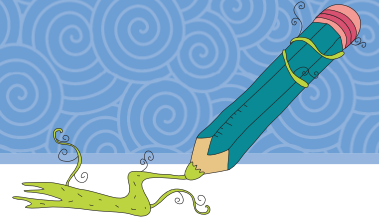


01. 89T51 라인트레이서는 CPU 입출력 포트와 다양한 주변 회로의 입출력 포트에 헤더 커넥터가 연결되어 있어 실습 상황에 따라 사용자가 쉽게 점퍼선으로 배선할 수 있다.
02. 조도 센서와 온도 센서에서 나오는 아날로그 신호는 89T51에 내장된 A/D 변환 모듈로 디지털 신호로 변환한다.
03. ISP 통신으로 사용자 프로그램을 PC로부터 라인트레이서에 전송한다.
04. 저항값은 색 띠로 표시하고 콘덴서 용량은 종류에 따라 고유한 방식으로 기록한다.
05. 극성을 가지는 회로 소자는 반드시 극성을 고려해서 PCB에 조립하여야 한다.





# 단원 종합 문제



01

저항에서 색 띠의 순서가 빨강, 빨강, 노랑, 은색이었다면 저항 값과 오차율은 얼마인가?

- ①  $4.7K\Omega, \pm 5\%$     ②  $4.7K\Omega, \pm 10\%$     ③  $220K\Omega, \pm 10\%$   
④  $220K\Omega, \pm 5\%$     ⑤  $22K\Omega, \pm 10\%$

02

콘덴서에 223이라고 써 있다면 용량 값은 얼마인가?

- ① 223 pF    ② 223 uF    ③ 0.022 uF    ④ 0.022 pF    ⑤ 22000 uF

03

마이크로로봇에서 크리스털의 역할은 무엇인가?

- ① 특정 주파수의 펄스를 발생시킨다.  
② 인간의 심장이 혈액을 순환시키는 것처럼 전기를 주기적으로 순환시킨다.  
③ 전기 에너지를 일시적으로 저장한다.  
④ 빛을 사방으로 퍼뜨려 로봇을 장식한다.

04

발광 다이오드가 장착된 3가지 전자기기를 찾고 발광 다이오드의 역할에 대해 이야기해 보자.

05

온도 센서가 사용되는 전자기기에는 어떤 것들이 있는가?

06

조도 센서가 사용되는 전자기기에는 어떤 것들이 있는가?

07

우리가 사용하는 일상 가전제품에서 마이크로프로세서가 장착된 것은 어떤 것들이 있는가?

08

적외선 센서를 사용하는 제품들에는 어떤 것들이 있는가?





# IV C 언어로 로봇 움직이기

1. 로봇 두뇌 기초
2. 로봇에 명령어 심어주기
3. C 언어 기초
4. C 언어로 로봇 움직이기

**마이크로프로세서**는 각 센서로부터 입력되는 정보를 분석하고 명령을 내리는 로봇의 두뇌와 같은 역할을 한다. 그리고 로봇을 움직이려면 프로그래밍 언어인 C 언어를 사용하여 프로그램을 작성한다. 이 단원에서는 마이크로프로세서의 개념 및 구조와 프로그래밍 언어인 C 언어에 대해 알아보고, 이를 바탕으로 로봇을 움직이는 방법에 대해 학습하기로 한다.

# 01. 로봇 두뇌 기초

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇의 두뇌가 되는 마이크로프로세서의 구조와 동작을 이해한다.
- 로봇의 제어 보드에 사용되는 메모리의 종류와 쓰임새를 파악한다.
- 외부와 데이터를 주고받는 입출력 장치를 이해한다.
- 메모리를 사용하기 위한 주소 지정 방법을 설명할 수 있다.

마이크로프로세서는 로봇의 머리에 해당한다. 로봇을 구동하려면 먼저 마이크로프로세서를 이해하고 사용할 줄 알아야 한다. 이 절에서는 마이크로프로세서의 개념과 구조 및 동작에 대해 알아보자.



### 1. 마이크로프로세서의 개요

마이크로프로세서는 중앙처리장치(CPU)를 단일 반도체 집적회로(IC)로 제작한 것이다. 따라서 마이크로프로세서는 컴퓨터 CPU처럼 정보를 처리하고 외부에 연결된 장치들을 제어한다. 마이크로프로세서에 기억장치, 입출력 장치를 연결하여 일반 개인용 컴퓨터를 구성하기도 한다. 반도체의 집적기술이 더욱 발전되면서 기억장치, 입출력장치들까지 하나의 반도체 집적회로로 만들게 되었다.

이런 복합 기능을 갖는 반도체 집적회로를 원칩 마이컴이라고 부르기도 한다. 하지만, 일반적으로는 CPU 기능이 포함된 단일 반도체 집적회로를





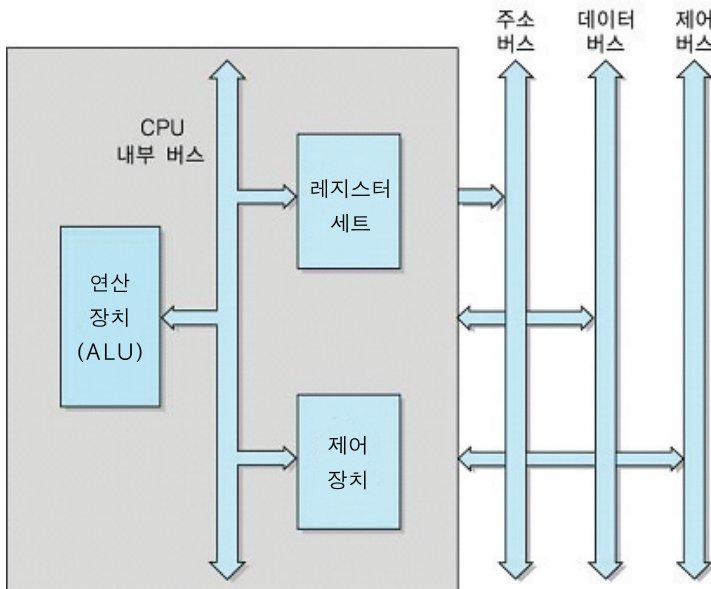
통칭해서 마이크로프로세서라고 부른다.

마이크로프로세서는 TV, 세탁기, 휴대 전화기, 복사기 등 거의 모든 전자기에 사용하며, 로봇에서는 사람의 두뇌 역할을 수행한다. 제조 회사마다 자신이 제작한 마이크로프로세서 내부의 메모리 용량이나 입·출력 포트의 갯수 등에 따라 구분하여 고유한 명칭을 붙인다. 그래서 명칭으로 제작사와 기능을 대략 파악할 수 있다.

컴퓨터를 구성하는 3요소에는 중앙처리장치(CPU), 기억 장치(Memory), 입출력 장치가 있다. 이 중에서 중앙처리장치는 비교와 판단 등을 처리하는 연산 장치, 명령어를 해석하고 실행하는 제어 장치, 연산 결과를 임시로 저장하는 기억 장소로서 어큐뮬레이터를 포함한 레지스터들로 구성된다.



- 인텔사는 8086, 80196, 8051, 80486, 8254 등과 같이 제조하고 있는 마이크로프로세서 및 주변 장치의 이름을 '8'로 시작하도록 하였으나, 최근에는 펜티엄Ⅲ, 펜티엄Ⅳ 등과 같은 이름도 사용하고 있다.
- 교재에서 사용하는 GC89T51은 국내 마이크로프로세서 전문회사 코아리버(Coreriver)에서 개발되었고 인텔 8052(8051에서 주변 회로가 보강됨) 호환 마이크로프로세서이다.
- 텍사스 인스트루먼트(TI)사의 경우 이 회사에서 제조되는 마이크로 프로 세 서 는 DSP(Digital Signal Processor)라고 불리지만, TMS320Cxx와 같이 번호를 붙이고 있다.



[그림 4-1] 중앙처리장치의 구성



마이크로프로세서는 내부에 입출력 장치를 비롯하여 타이머, 통신장치, 데이터 메모리, 프로그램 메모리, 레지스터, A/D 변환기, 아날로그 비교기 등 다양한 기능들을 선택적으로 포함하는 모델들이 있다. CPU 코어의 기능이 같으면서 메모리 용량이나 내장된 기능이 다른 마이크로프로세서들을 계열(Family)이라고 한다.

버스(Bus)는 마이크로프로세서에서 데이터를 주고 받는 통로이다. 한 번에 주고받는 데이터의 크기가 버스의 용량이다. 마이크로프로세서의 성능 차이는 버스의 용량에 따라 결정된다. 버스의 용량은 8비트, 16비트, 32비트, 64비트로 구분되며, 클수록 처리 속도나 성능이 높다.

학습용 로봇에는 대개 8비트 마이크로프로세서를 사용한다. 고속 처리와 다양한 기능을 갖는 PDA, 컴퓨터 등은 32비트나 64비트 마이크로프로세서를 사용한다.

내부 메모리의 용량과 내장된 기능을 조사해서 로봇의 기능에 맞는 마이크로프로세서를 사용하면 된다. 설계하려는 로봇을 쉽게 구현할 수 있는 기능을 포함하는지, 구입하기는 쉬운지, 가격은 저렴한지를 판단하여 선정하여야 한다.



## 2. 메모리의 종류와 쓰임

기억 장치인 메모리는 데이터를 저장한다. 한 번에 일정한 용량 단위로 데이터를 읽고 쓰기때문에 데이터의 용량 단위로 위치를 지정하는 주소(Address)가 필요하다.

[표 4-1] 메모리 저장 내용

어드레스	D7	D6	D5	D4	D3	D2	D1	D0	데이터 값
000 (0번지)	0	1	1	1	0	0	1	1	0x73
001 (1번지)	1	1	0	1	0	1	1	1	0xD7
010 (2번지)	0	1	1	1	1	0	1	1	0x7B
011 (3번지)	0	1	1	1	0	1	1	1	0x77
100 (4번지)	0	1	1	1	1	0	0	1	0x79
101 (5번지)	0	1	0	1	0	1	0	1	0x55
110 (6번지)	0	0	1	1	0	0	1	1	0x33
111 (7번지)	0	1	0	1	1	1	1	1	0x5F



저장 용량이 8바이트인 메모리의 데이터 저장 내용을 표로 보면, 표 4-1과 같이 메모리에는 주소가 번지로 지정되어 각 위치에 2진수로 데이터가 저장된다.



메모리에는 마이크로프로세서 내부에 있는 내부 메모리와 외부에 존재하는 외부 메모리가 있다. 내부 메모리는 용량이 적고 고속으로 읽고 쓸 수 있는 반면, 외부 메모리는 용량이 크지만 읽고 쓰는 속도는 내부 메모리보다 느리다.

내부 메모리는 RAM으로 구성되며 연산 결과를 임시로 저장하는 캐시 메모리와 프로그램 실행 위치 등을 저장하는 스택 메모리가 있다.

메모리는 저장하는 방식과 저장된 데이터를 보존하는 방법에 따라 RAM, ROM, FLASH ROM, EEPROM 등으로 구분된다. CPU 내부에 있는 어큐뮬레이터나 범용 레지스터와 같은 연산 처리용 메모리는 한정된 용량 안에서 사용해야 한다. 최근에는 프로그램을 내부에 저장하기 위하여 플래시 메모리를 내장하는 경우가 많다.

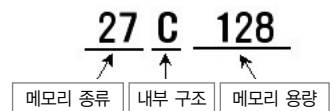
메모리 외부에 표기된 숫자를 통하여 메모리의 크기, 메모리의 종류 등을 확인할 수 있다.

### 1) 롬 (ROM : Read Only Memory)

롬은 마이크로프로세서가 데이터를 읽기만 하는 읽기 전용 메모리로, 로봇을 움직이는 제어 프로그램이 들어간다.

롬(ROM)에 로봇을 구동하는 프로그램을 넣으려면 롬 라이터라는 특수한 장비를 이용해야 하며, 전기가 끊겨도 데이터를 그대로 유지한다. 롬에 한번 기록된 데이터는 지워지지 않으며, 재활용을 위해 지우려면 롬 소거기라는 특수한 장비를 사용하여야 한다.

롬의 명칭은 27C64, 27C128과 같이 어떤 회사의 제품이든지 앞에 '27'이라는 숫자로 시작한다. 따라서 소자의 명칭이 '27'로 시작되면 롬(ROM)이라 것을 알 수 있다.中间的의 'C'는 롬이 CMOS 회로로 구성되었다는 의미이며, 마지막 숫자들은 메모리의 용량이다. 외형에 표기된 메모리 용량의 단위는 킬로비트이다.



[그림 4-2] ROM의 명칭





#### 메모리이야기

삼성전자가 세계 최초로 50나노미터 공정의 1기가 D램 반도체 개발에 성공.

나노미터는 10억분의 1[m]를 가리키며, 반도체에선 공정(工程)의 세밀도를 나타낸다. 수치가 낮을수록 반도체 집적도가 높아져 종전과 같은 면적이라면 보다 많은 반도체를 만들 수 있다.

삼성전자가 개발한 50나노미터 1기가비트(Gb) D램(PC에서 정보를 임시로 저장하는 반도체)은 양산 중인 80나노미터 공정에 비해 3세대나 앞선 기술로 같은 크기의 웨이퍼(반도체 재료)가 되는 원판에서 만들 수 있는 반도체가 2배 정도 많다.

삼성전자는 D램 사업을 시작한 지 23년째인 2006년 D램 매출이 반도체 단일 제품으로는 처음으로 100억 달러를 넘어섰다. 2006년 삼성전자의 세계 D램 시장 점유율이 32% 선에서 2007년에는 최대 40% 중반대까지 올라갈 수 있을 것"이라고 예상했다.

[표 4-2] ROM의 명칭과 용량

부품명	크기		부품종류
	비트	바이트	
27C32	32K	4K	ROM
27C64	64K	8K	
27C128	128K	16K	
27C256	256K	32K	
27C512	512K	64K	
27C010	1024K	128K	
27C020	2048K	256K	

#### ○ 롬(ROM)의 동작

ROM의 메모리 용량이 16바이트라면, 각 바이트마다 주소가 변지로 지정되어야 하므로 [표 4-3]과 같이 4개의 주소 비트(A0, A1, A2, A3)가 필요하다.

[표 4-3] 메모리의 주소 지정

A3	A2	A1	A0	지정된 위치
0	0	0	0	0 번지 지정
0	0	0	1	1 번지 지정
0	0	1	0	2 번지 지정
⋮				
1	1	1	0	14 번지 지정
1	1	1	1	15 번지 지정

## 2) 램 (RAM : Random Access Memory)

램(RAM)에는 데이터를 읽고 쓰기를 모두 할 수 있으나, 전원이 끊어지면 저장된 데이터는 모두 사라진다. 연산 과정의 데이터를 저장하는 역할을



하기 때문에 데이터 메모리라고 부르기도 하고, 전원을 차단하면 데이터가 사라져 버려서 휘발성 메모리라고도 부른다. 램(RAM)은 SRAM(Static RAM)과 DRAM(Dynamic RAM) 두 가지 종류가 있다.

[표 4-4] DRAM과 SRAM의 비교

종 류	DRAM	SRAM
가격	가격이 싸다	가격이 비싸다
집적도	집적도가 높다	집적도가 낮다
주용도	PC 혹은 영상 메모리	산업용 메모리

산업용이나 일반 자동화 장치 혹은 로봇 등에는 주로 SRAM을 사용하기 때문에, SRAM을 중심으로 학습하도록 한다.

램(RAM)의 명칭은 롬(ROM)에서와 마찬가지로 6264, 62128과 같이 어떤 회사의 제품이든지 '62' 라는 숫자로 시작한다. 따라서 소자의 명칭이 '62' 로 시작되면 램(RAM)이라 것을 알 수 있다. 마지막 숫자는 메모리의 용량을 나타내며, 단위로는 킬로비트(Kb)를 사용한다.

[표 4-5] RAM의 명칭과 용량

부품명	크기		부품 종류
	비트	바이트	
6264	64K	8K	RAM
62128	128K	16K	
62256	256K	32K	
621024	1024K	128K	

### ○ 램 (RAM)의 동작

램(RAM)은 롬(ROM)과 거의 비슷하게 동작한다. 메모리 용량 16 바이트인 램(RAM)은 롬(ROM)과 마찬가지로 [표 4-3]과 같이 주소 비트 4개(A0, A1, A2, A3)가 필요하다.

6 2 2 5 6



[그림 4-3] RAM의 명칭



#### 플래시 메모리

플래시 메모리 기술의 발전은 코드 저장용과 데이터 저장용의 두 가지 방향으로 이루어지고, 코드 저장용 플래시 메모리는 기존의 DRAM 및 SRAM 등과 같이 고속화 저전압화의 요구에 따라 발전해 왔다. 데이터를 읽거나 쓰는 속도에 있어서 현재 비동기 제품의 경우에는 100[ns]에서 50[ns]까지 발전하였다. 휴대폰 등의 이동 단말기는 저전력을 소모하는 제품을 필요로 하기 때문에 이후에 1.0[V] 이하의 제품 개발이 요구된다. NOR형 플래시 메모리는 미국 인텔, 일본 샤프, 미국 AMD와 일본 후지쯔의 제휴가 대표적이며 NAND형 플래시 메모리는 일본 도시바와 삼성 전자가 제휴해 기술 개발을 하고 있다

### 3) EEPROM (Electrical Eraser Programmable ROM)

EEPROM은 RAM과 ROM의 장점만을 취합하여 만든 메모리인데 읽고 쓰기가 가능하고, 전원이 끊어져도 저장된 데이터가 계속 남아 있다. 부품 명칭이 '28'로 시작되면 EEPROM인 것을 알 수 있다.

[표 4-6] EEPROM의 명칭과 용량

부품명	크기		부품 종류
	비트	바이트	
28C64	64K	8K	EEPROM
28C128	128K	16K	
28C256	256K	32K	
28C512	512K	64K	

EEPROM의 동작은 RAM과 같이 읽고 쓰기를 위한 제어 신호 입력 핀이 있으므로 RAM의 동작 설명을 참고한다.

### 4) 플래시 롬 (FLASH ROM)

EEPROM의 발달된 형태라고 할 수 있다. 용량이 커지고 전기적 충격에서도 데이터가 쉽게 지워지지 않도록 읽고 쓰는 방식에 약간의 차이가 있다.

최근 디지털 카메라, MP3 플레이어 등에 많이 사용한다. 마이크로프로세서가 대부분 프로그램 메모리를 외부에 장착하였으나, 최근에는 개발 과정을 편리하게 하고 전체 시스템의 크기를 줄이기 위하여 마이크로프로세서 내부에 프로그램 메모리로서 플래시 메모리를 내장하는 경우가 많다. 89T51이라는 마이크로프로세서는 14 킬로바이트의 플래시 메모리가 내장되어 있어서 외부에 프로그램 메모리를 별도로 장착할 필요가 없다. 플래시 롬을 구별하는 명칭으로는 '29'를 사용한다.



[표 4-7] Flash의 명칭과 용량

부품명	크기		부품 종류
	비트	바이트	
29C256	256K	32K	Flash ROM

### ○ 플래시 롬의 동작

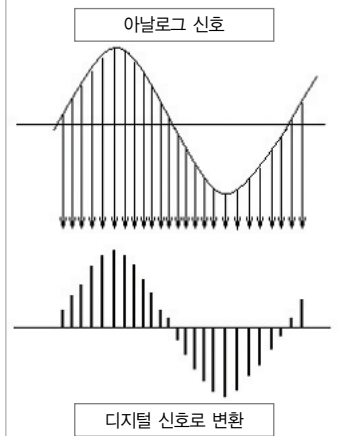
플래시 롬도 EEPROM과 마찬가지로 읽고 쓰기가 가능하고, 전원이 끊어져도 저장된 데이터가 보존된다. 데이터를 기록할 때 일정 블록 단위의 데이터 기록방식을 사용하고 있다. 자세한 동작 방법은 플래시 메모리의 제조사에서 제공하는 매뉴얼을 참고한다.

## 3. 입출력 장치

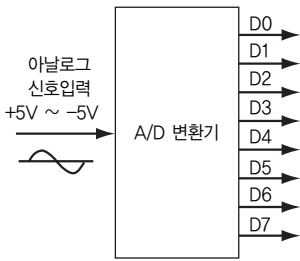
로봇이 인간과 유사하게 판단하고 행동하도록하기 위해 감각 장치인 센서를 사용한다. 센서는 외부의 물리적, 화학적 에너지를 전기 에너지로 바꿔준다. 센서에 의해 전기 에너지로 바뀐 정보는 값이 연속적으로 변화하는 아날로그 신호이기 때문에 '1'과 '0'만으로 구성되는 디지털 신호로 바뀌어야 한다. 또한 마이크로프로세서에서 아날로그 음성 신호를 전송하려면 디지털 신호로 되어 있는 음성 정보를 다시 아날로그 신호로 바꿔 전송해야 한다.

로봇에 입력되는 신호에는 연결/끊김과 같은 점점 신호가 입력되는 경우도 있어 '1'인지 '0'인지 바로 인식할 수 있는 경우도 있으며, 연결/끊김 신호를 외부로 내보내야 할 때도 있다.

아날로그 신호가 입력되는 경우는 A/D변환기를, 아날로그 신호를 출력하는 경우는 D/A변환기를 사용한다. 디지털 신호의 입·출력은 로봇의 회로에 외부의 잡음이 영향을 미치지 않도록 포토 커플러를 사용한다.



[그림 4-4] 아날로그 신호의 샘플링

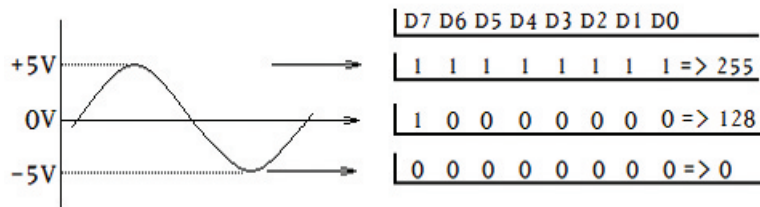


[그림 4-5] A/D 변환기

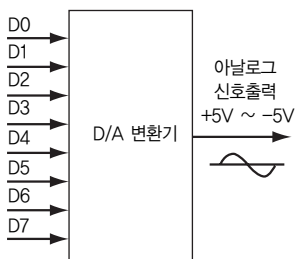
### 1) A/D 변환기

A/D 변환기는 아날로그 신호를 디지털 신호로 변환시키는 소자이다.

A/D 변환기는 아날로그 신호를 일정 시간 간격마다 신호값을 추출해서 디지털 값으로 변환한다. 일정 시간 마다 아날로그 신호값을 추출하는 과정을 '샘플링'이라 하고 샘플링된 값을 디지털 값으로 변화시키는 과정을 '양자화'라고 한다. 아날로그 신호값의 전체 범위를 얼마나 정밀하게 쪼개서 디지털 값으로 변환하느냐를 나타내는 것을 '분해능'이라 한다. 8비트 A/D 변환기라고 하는 것은 입력되는 신호의 범위를 최대 2를 8번 곱한 256개로 쪼개서 표현할 수 있다는 의미로 분해능이 8비트인 A/D 변환기를 말한다. 예를 들어 입력되는 아날로그 신호의 범위가 +5[V]~-5[V]인 경우에 8비트 A/D 변환기를 사용하면, +5[V]가 입력되면 디지털 값은 255가 되고, -5[V]가 입력되면 디지털 값은 0이 된다. 또한, 0[V]가 입력되면 8비트의 중간값 128 즉, 0x80이 디지털 값이 되고, 이것이 데이터 버스를 통하여 마이크로 프로세서에 입력된다.



[그림 4-6] A/D 변환기의 출력



[그림 4-7] D/A 변환기

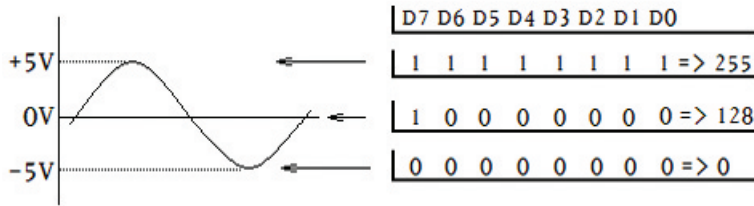
### 2) D/A 변환기

D/A 변환기는 A/D 변환기와 반대로서, 디지털 신호를 아날로그 신호 값으로 변환시키는 소자이다.

예를 들어, 8비트 분해능의 능력을 지니고 출력 범위가 +5[V]~-5[V]인 D/A 변환기를 사용할 때, 마이크로프로세서에서 0x80의 값을 출력하면,



D/A 변환기를 통해 출력되는 아날로그 신호는 0[V]가 출력된다.



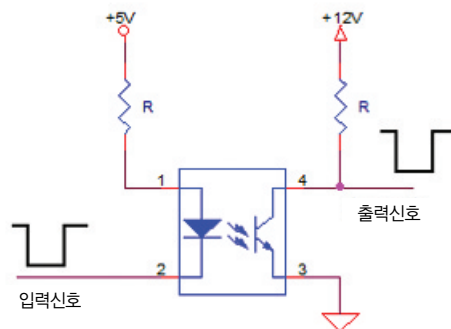
[그림 4-8] D/A 변환기의 출력

D/A 변환기를 통해 출력되는 아날로그 신호가 음성 신호이면 외부의 증폭기들을 통해 스피커로 출력된다.

### 3) 포토 커플러 (Photo Coupler)

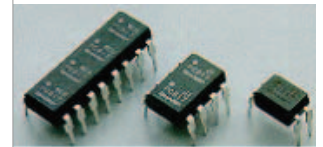
포토 커플러는 광센서의 일종으로 내부에 발광다이오드와 포토트랜지스터를 내장하고 있다. 외부의 다른 기기와 신호를 주고받을 때, 회로에 잡음이나 전기적 충격이 가해지는 것을 방지하거나, 전원 전압이 서로 다른 회로에서 신호를 주고 받을 때 사용한다.

마이크로프로세서에서 '1' 또는 '0'의 신호를 외부에 내보내거나, 외부에서 '1' 또는 '0'의 신호를 입력 받을 때 사용한다.



[그림 4-10] 포토 커플러 출력 회로 예

[그림 4-10]의 회로처럼 마이크로프로세서에서 5[V]의 신호가 출력되면 내부의 발광 다이오드가 빛을 내고 그 빛을 받아 포토 트랜지스터의 이미터

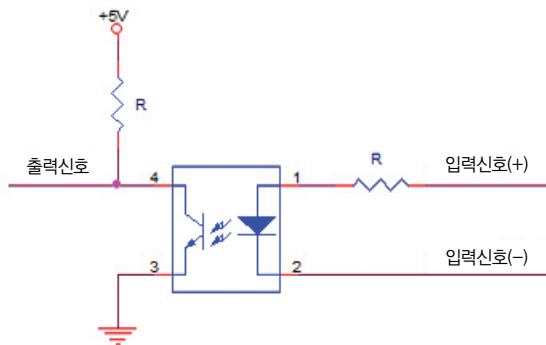


[그림 4-9] 포토 커플러 실제 모양과 기호



와 컬렉터 사이에 전기가 흘러서 컬렉터 단자에 연결된 외부 단자로 출력 신호가 전달된다.

이때, 전기 신호 레벨을 5[V]가 아닌 12[V] 또는 24[V]로 바꾸고자 할 때는 내부의 포토 트랜지스터 컬렉터 단자에 바꾸고자 하는 전압을 연결시켜 신호의 전압 레벨을 바꿀 수 있다. 이것을 DO(Digital Output)이라고 한다.



[그림 4-11] 포토 커플러 입력 회로 예

[그림 4-11] 회로는 외부로부터 신호를 입력받는 회로로, DI(Digital Input)이라고 한다. 입력되는 신호의 전압이 12[V]인지 24[V]인지에 따라서 포토 커플러 내부의 발광 다이오드가 빛을 내면서 손상되지 않도록 입력되는 단자의 저항값을 결정해야 한다. 또한, 갑작스러운 번개나 잡음과 같은 신호가 들어오는 것을 방지하기 위해 입력 단자에 바리스터와 같은 부품을 사용하기도 한다.

#### 4. 마이크로프로세서 주소 지정 회로

마이크로프로세서는 롬(ROM)이나 램(RAM)에 저장되어 있는 프로그램이나 데이터를 읽거나, 데이터를 저장하는 일을 수행한다. 데이터를 읽거나 저장하려면 위치를 지정해 주어야 하는데, 이것을 주소 지정이라 한다.

예를 들어 8비트 메모리의 경우 8비트 단위로 데이터를 저장하므로 메모리에는 내부적으로 8비트의 단위로 방이 만들어진다. 집마다 집을 가리키는 주



소가 있듯이 메모리의 각 방을 가리키는 주소가 존재하게 되며, 고유의 주소가 있어 주소 버스(A0~An)를 통해 2진수로 주소를 번지로 지정할 수 있다.

디코더나 PLD(Programmable Logic Device)를 사용하여 특정 주소를 가진 메모리 위치에 데이터를 읽거나 쓴다.

### 1) 주소 지정 회로

주소 지정 회로에는 74HC139 디코더 IC를 사용하거나 PLD라는 IC를 사용한다.

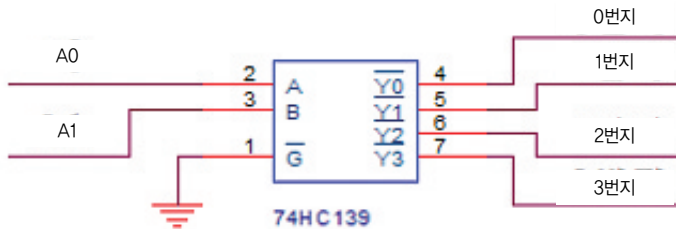


#### 예제 4-1

주소 비트가 A0, A1인 마이크로프로세서에서 4개의 주변 장치를 제어하기 위한 번지 지정 회로를 74HC139 디코더 IC를 이용하여 설계하자.

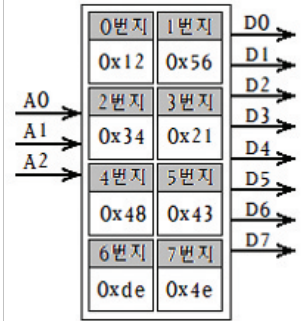
주소 1개마다 1개의 주변 장치를 지정할 수 있으므로, 이 마이크로프로세서는 총 4개의 주변 장치를 지정할 수 있다.

이 주소들을 구분하기 위해 74HC139를 이용하여 다음과 같은 번지 지정 회로를 설계할 수 있다.



[그림 4-13] 74HC139를 사용한 주소 지정

[그림 4-13]의 회로를 보면 입력되는 두개의 신호에 따라 출력  $\overline{Y0} \sim \overline{Y3}$ 의 출력 신호가 결정된다. 출력 신호 이름 위에 선이 그어진 것은 지정된 출력



[그림 4-12] 메모리의 주소

[표 4-8] 주변 장치의 주소 지정

번지	A1	A0
0번지	0	0
1번지	0	1
2번지	1	0
3번지	1	1

선에 '0'의 신호가 나타난다는 의미다. A1과 A0가 모두 '0'의 신호가 입력되면 출력은  $\overline{Y0}$ 단자에 '0'의 신호가 출력되고 나머지  $\overline{Y1} \sim \overline{Y3}$ 까지는 '1'의 신호가 출력된다.

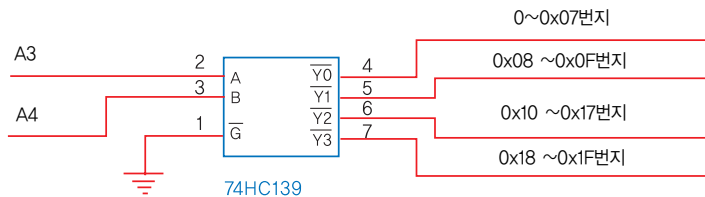


#### 예제 4-2

A0~A4의 주소 비트를 지닌 마이크로프로세서가 있다고 가정한다. 이 마이크로 프로세서를 이용하여 4개의 주변 장치를 제어하기 위한 주소 지정 회로를 74HC139를 이용하여 설계하자. 단, 주소 범위가 주변 장치보다 많으면 반드시 상위 주소부터 설계해야 한다.

**풀이** A0~A4까지 주소 비트 5개이므로  $2^5 = 32$ 개의 주변 장치의 주소를 지정할 수 있다. 사용할 주변 장치는 4개인 것을 생각하여 번지 지정 회로를 설계하여야 한다.

상위 주소부터 설계하기 위하여 주소 비트 A0, A1, A2, A3, A4 중에서 A4부터 이용하고 상위부터 A3, A2... A0 순서로 사용해야 한다.



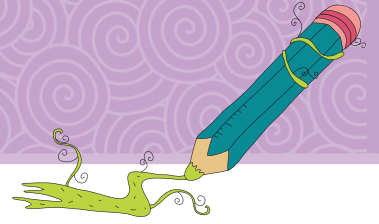
[그림 4-14] 상위 주소 비트를 이용한 주소 지정



[표 4-8] 상위 비트를 이용하여 주소 지정한 결과

주소	A4	A3	A2	A1	A0	해당 출력
0x00	0	0	0	0	0	출력 Y0에 '0'의 신호 출력
0x01	0	0	0	0	0	
		• • •				
0x07	0	0	1	1	1	출력 Y1에 '0'의 신호 출력
0x08	0	1	0	0	0	
0x09	0	1	0	0	1	
		• • •				
0x0F	0	1	1	1	1	출력 Y2에 '0'의 신호 출력
0x10	1	0	0	0	0	
0x11	1	0	0	0	1	
		• • •				
0x17	1	0	1	1	1	출력 Y3에 '0'의 신호 출력
0x18	1	1	0	0	0	
0x19	1	1	0	0	1	
		• • •				
0x1F	1	1	1	1	1	

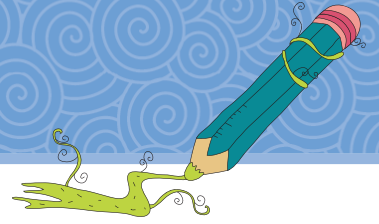
# 단원 학습 정리



01. 마이크로프로세서는 컴퓨터의 중앙처리장치(CPU)를 단일 집적회로로 제작한 것이다.
02. 컴퓨터는 중앙처리장치, 기억장치, 입출력장치로 구성된다.
03. 중앙처리장치는 비교와 판단 등을 처리하는 연산장치, 명령어를 해석하고 실행하는 제어장치, 연산 결과를 저장하는 레지스터들로 구성된다.
04. 기억장치인 메모리는 데이터를 저장한다. 데이터를 일정한 용량 단위로 읽고 쓰기 때문에 용량 단위별로 위치의 주소(Address)를 번지로 지정한다.
05. 메모리에는 프로그램을 저장하는 롬(ROM)과 작업 과정의 데이터를 저장하는 램(RAM)으로 구분된다.
06. 아날로그 신호를 컴퓨터가 처리하려면 A/D 변환기를 사용하고, 아날로그 신호를 출력하려면 D/A 변환기를 사용한다.



# 단원 정리 문제



01

중앙처리장치의 역할을 모두 고르시오.

- ① 비교와 판단      ② 작업과정 제어      ③ 전원 공급
- ④ 명령어 해석      ⑤ A/D 변환

02

기억 장치를 모두 고르시오.

- ① USB                  ② ALU                  ③ Flash                  ④ SRAM
- ⑤ EPROM              ⑥ DRAM              ⑦ MP3

03

다음 중에서 입출력 장치를 모두 고르시오.

- ① 포토 커플러      ② MP3 플레이어      ③ Flash
- ④ RS 232              ⑤ DVD 플레이어

04

1 킬로바이트의 메모리의 주소를 지정하려면 주소 비트가 몇 개 필요한가?

- ① 1000개              ② 10개                  ③ 100개
- ④ 20개                  ⑤ 알 수 없다

05

다음 중 성능이 가장 좋은 마이크로프로세서는?

- ① 8비트 마이크로프로세서                  ② 16비트 마이크로프로세서
- ③ 32비트 마이크로프로세서                  ④ 64비트 마이크로프로세서

06

0 ~ 5[V]의 아날로그 전압 값을 10비트 A/D 변환기를 사용하여 디지털 값으로 변환할때, 2.5[V]에 해당하는 디지털 값은?

- ① 0H                  ② 3FFH                  ③ FFH
- ④ 200H              ⑤ 2FFH

07

포토 커플러를 사용하여 신호를 입출력할 때 얻을 수 있는 장점 두 가지만 이야기해 보자.



# 02. 로봇에 명령어 심어 주기

## ROBOT CONTROL SYSTEM

### 학습목표



- 컴파일러의 역할을 설명할 수 있다.
- 원시 파일을 컴파일하여 실행 파일을 만들 수 있다.
- 실행 파일을 ISP 통신을 사용하여 라인트레이서에 전송할 수 있다.



### 컴파일러

컴파일러는 실행 시에 모든 문장을 먼저 구문적으로 하나씩 분해하고, (다른 문장을 참조하는 경우) 문장이 정확하게 참조될 수 있도록 여러 번의 연속적인 상태에서 결과코드를 만든다. 컴파일로 생긴 결과물은 목적 코드(object code) 또는 목적 모듈(object module)이라 부르는데 - 목적 코드는 프로세서가 한 번에 한 명령씩 처리하거나 또는 실행시킬 수 있는 기계코드(machine code)이다.

최근에 객체 지향 언어의 하나인 자바에서는 어떠한 컴퓨터 플랫폼에서도 실행될 수 있는 컴파일 결과물(이를 바이트코드라고 부른다)을 생산하는 체제를 도입했다. 이 바이트코드는 실행될 하드웨어의 종류에 관계없이 동일한 내용으로 생성되어 서버에 저장되는데, 필요할 때마다 네트워크를 통해 클라이언트로 다운로드된다.

## 1. 컴파일러의 개요

로봇을 구동시키려면 사람이 작성한 프로그램의 원시 파일을 기계어로 바꾸는 작업이 필요하다.

### 1) 컴파일(Compile)

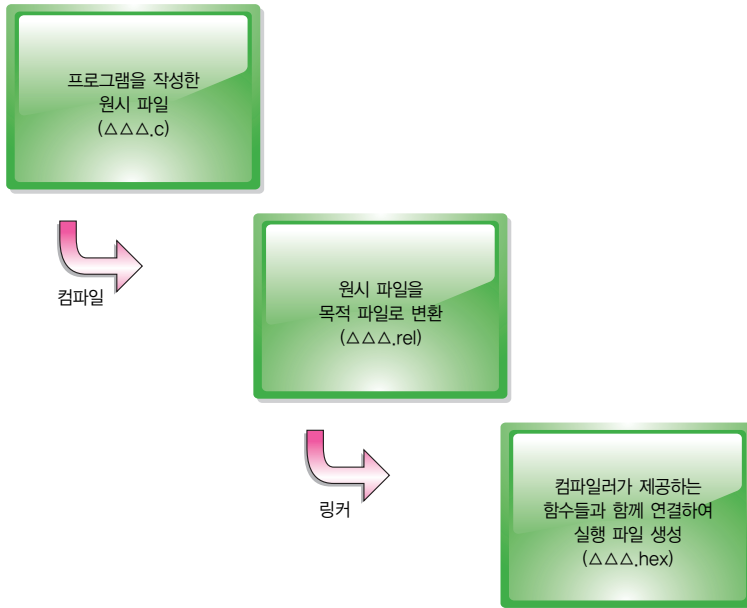
컴파일이란 사람이 작성한 원시 파일(△△△.c)을 컴퓨터가 인식할 수 있는 기계어로 번역하여 이진(0 또는 1)형태의 목적 파일을 만드는 작업이다.

### 2) 컴파일러(Compiler)

컴파일 작업을 수행하려면 언어 번역 프로그램이 사용되는데, 이를 컴파일러라 한다. 일반 컴퓨터는 컴파일러로서 주로 터보-C 등을 사용하고, 마이크로로봇을 위해서는 GENCC, IAR-C, Keil-C 등 전용 컴파일러를 사용한다.



아래 그림은 컴파일러가 원시 파일(△△△.c)을 기계어로 번역하여 실행 파일을 만드는 과정을 나타낸 것이다.



[그림 4.2-1] 원시 파일에서 실행 파일을 만드는 과정

- ① C-언어의 원시 파일은 '.C' 라는 확장자를 가져야 하며, 괄호 안의 확장자들은 각각의 과정을 통해 생성되는 파일의 확장자이다.
- ② 원시 파일의 이름이 'demo.c' 라면 컴파일러와 링커의 작업을 통해 생성되는 파일은 'demo.rel', 'demo.hex' 이다.
- ③ 목적 파일의 경우 컴파일러에 따라 확장자가 다를 수 있다.
- ④ 생성된 실행 파일을 ROM에 프로그램하거나, 프로그램 전송의 전용 포트를 통해 마이크로로봇에 전송하여 구동한다.

컴파일러를 통해 생성된 실행 파일을 코아리버사에서 제공하는 ISP 전송 유틸리티를 사용하여 로봇에 전송하여 구동한다.



### 컴파일러 종류

C 컴파일러에는 매우 다양한 종류가 있지만 대표적인 두 가지에는 MS-C(Microsoft C)와 Microsoft Visual C++가 있다.

MS-C는 소프트웨어 개발 회사나 전문 프로그래머가 제품의 개발을 위해 사용하는 '컴파일러의 원조'라고 할 수 있는 것으로 Windows 환경에서 많이 이용되는 컴파일러이다. 단순한 C 프로그래밍 이외에도 Windows 프로그래밍, 통신 프로그래밍 등 다양한 응용에 사용할 수 있다. GUI 환경의 통합 개발 환경을 제공하기 때문에 처음 사용하는 사용자에게 편리할 수 있다.

Windows에서 활용되는 주요 프로그램의 대부분은 Visual C++로 작성되어 있다. 과거의 프로그램은 프로그래머가 직접 소스 파일을 한 글자씩 모두 입력해서 만들었는데 비주얼 C++은 프로그래머가 최소한의 동작만 하고 나머지 소스코드 부분은 프로그램이 알아서 만들어 주는 방식을 취하고 있다. Visual C++은 프로그램의 편집과 실행을 동시에 할 수 있는 통합 환경을 가지고 있다.

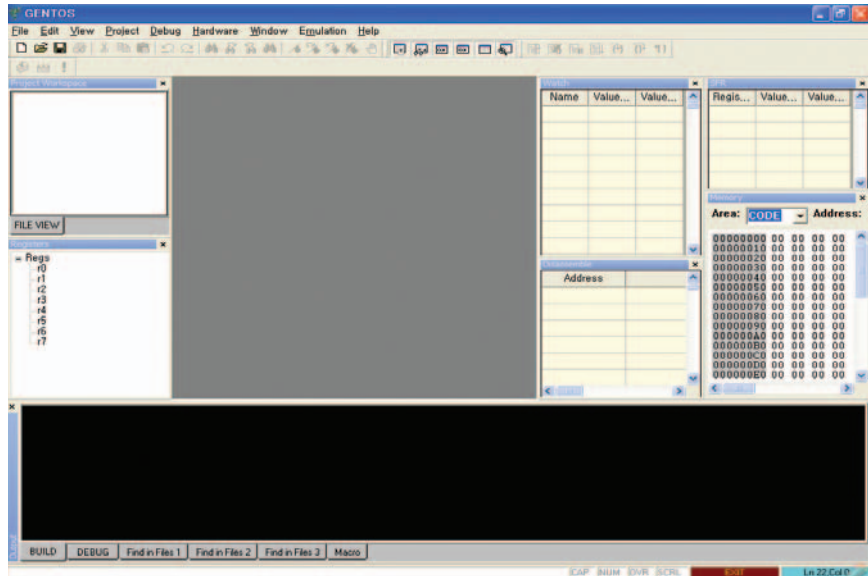
## 2. C 컴파일러 사용법(윈도우용 GENCC compiler)

89T51 마이크로프로세서의 프로그램을 작성하고 컴파일하는 도구로서 코어리버사의 GENCC 컴파일러에 대해 알아보자. GENCC는 마이크로로봇 제작에 사용되는 인텔 8051 호환의 MiDAS 계열의 마이크로프로세서에 사용할 수 있는데, 실행 파일의 크기와 속도에서 최적화된 결과를 생성할 뿐만 아니라 프로그램 메모리의 크기에 제약 없이 사용할 수 있는 장점이 있다. 이 장에서는 윈도 기반으로 구동되는 GENCC 컴파일러를 사용하는 방법을 설명하고, 간단한 데모 프로그램을 작성하고 컴파일을 수행하여 실행 파일을 생성하여 보자.

### [1단계] 프로그램의 시작

윈도상에 설치된 원시 파일 편집기와 GENCC 프로그램을 시작하기 위해 『시작 → 모든 프로그램 → Coreriver → Gentos』을 선택하면 [그림 4.2-2]과 같이 프로그램 초기화면이 나타난다.

### [2단계] 새 작업 공간 생성



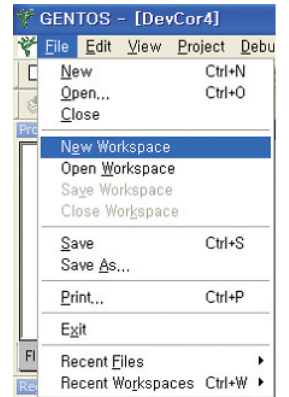
[그림 4.2-2] 프로그램 초기 화면



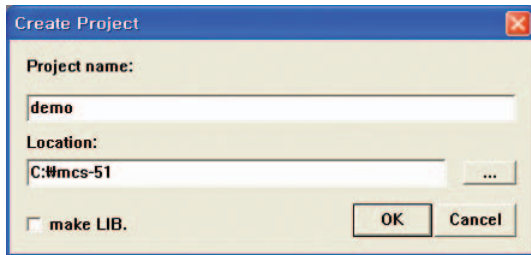
화면 상단의 주 메뉴에서 『File - New Workspace』를 선택하면 [그림 4.2-3]과 같이 새 작업 공간(작업 파일들이 저장되는 디렉토리) 생성창이 나타난다.

작업 공간의 위치(Location)를 지정하고(여기서는 'C:\mcs-51'로 지정한다.) [그림 4.2-4]와 같이 『Project name』 입력창에 프로젝트 이름(여기서는 'demo')을 입력한다.

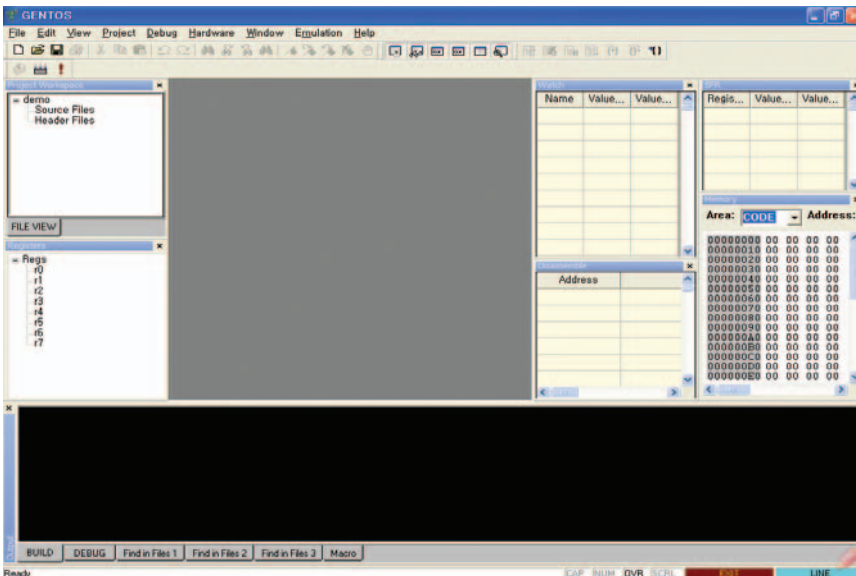
[그림 4.2-4]의 작업을 마치고 『OK』 버튼을 누르면 『Project Workspace』창에 [그림 4.2-5]와 같이 demo-Source Files 가지와 demo-Header Files 가지가 나타난다.



[그림 4.2-3] 새 작업 공간 생성창




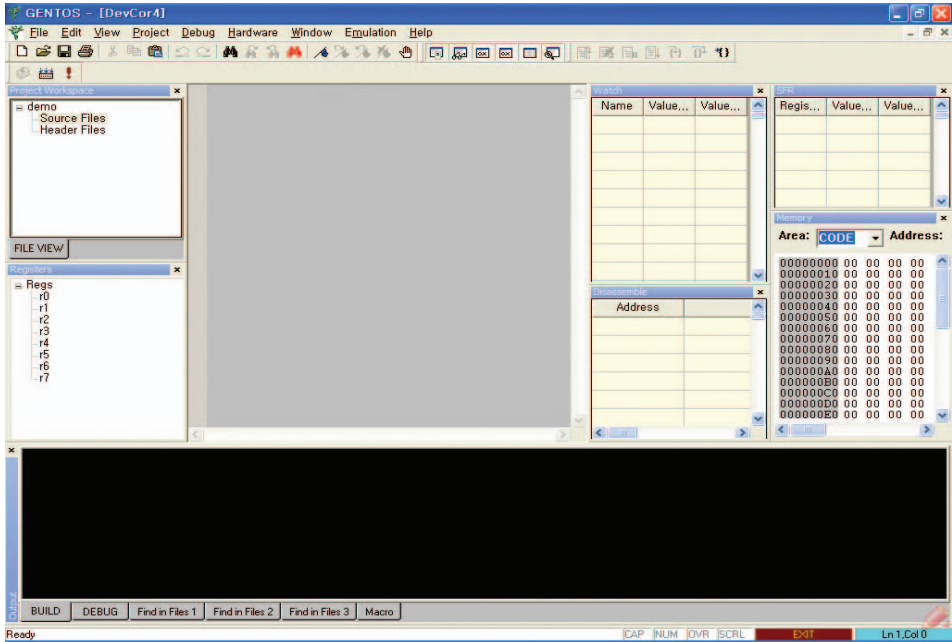
[그림 4.2-4] 새 작업 공간의 위치와 프로젝트 이름 저장



[그림 4.2-5] 『Project Workspace』창에 가지 생성

### [3단계] 새 소스 파일의 작성

주 메뉴의 『File - New』를 선택하거나 툴바의 (  ) 아이콘을 클릭하면 새 원시 파일을 편집할 수 있다.



[그림 4.2-6] 텍스트 편집 모드

[그림 4.2-6]처럼 『Project Workspace』 창 우측에 있는 편집창이 옅은 회색으로 변하며 편집 모드가 된다. [그림 4.2-6]의 편집창에 아래의 'demo.c'와 같은 원시 파일을 [그림 4.2-7]과 같이 작성한다.

#### • demo.c 원시 파일

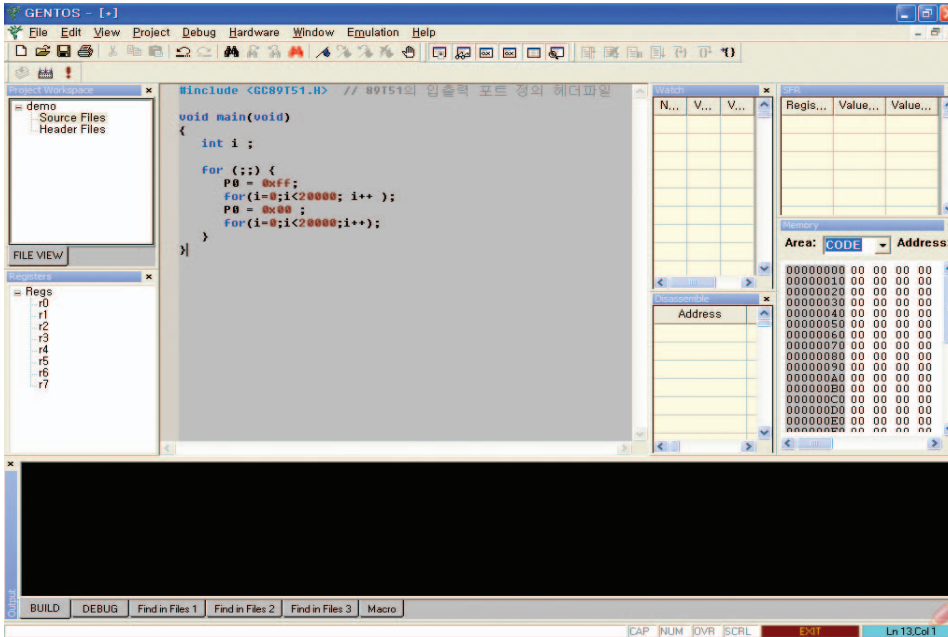
```
#include <GC89T51.H> // 89T51의 입출력 포트 정의 헤더 파일

void main(void)
{
    int i ;

    for (;;) {
        PO = ~0xff;
```

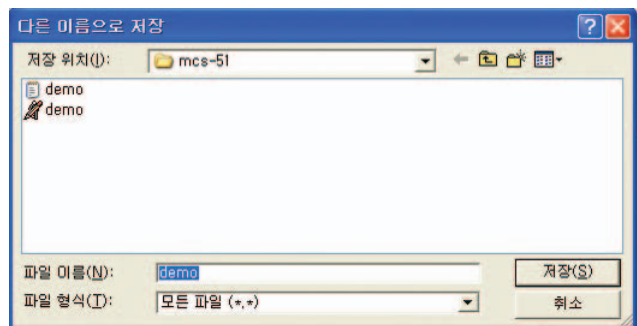


```
for(i=0;i<20000; i++ );  
P0 = ~0x00 ;  
for(i=0;i<20000;i++);  
}  
}
```



[그림 4.2-7] 편집 창에서 원시 파일 작성

[그림 4.2-7]과 같이 원시 파일 작성을 끝내면 주 메뉴의 『File-Save』를 선택하거나 혹은 툴 팔레트의 파일 저장 아이콘을 누르면 [그림 4.2-8]과 같이 파일 저장창이 나타난다. [그림 4.2-8]에서 저장 버튼을 눌러 작업 공간 디렉토리에 원시 파일을 'demo.c'의 이름으로 저장한다.

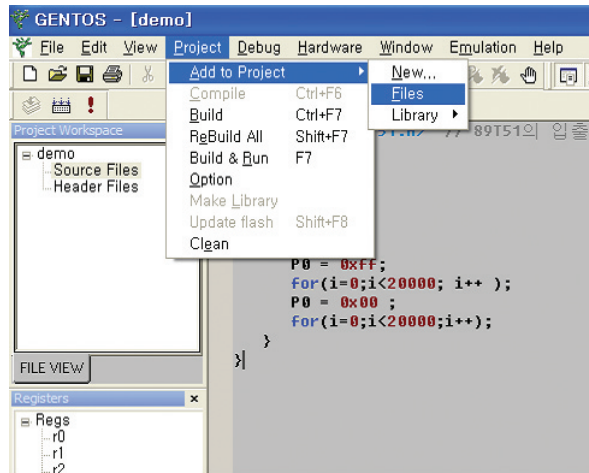


[그림 4.2-8] demo.c 원시 파일 저장



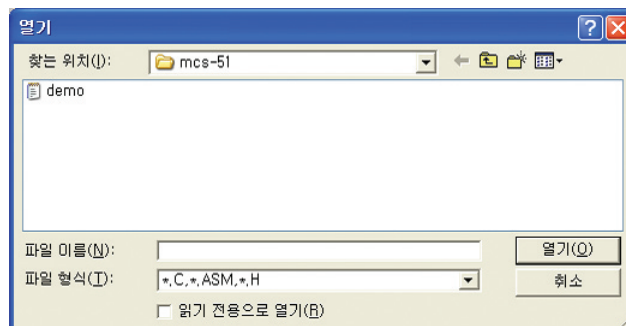
#### [4단계 ] 원시 파일을 프로젝트 파일 그룹에 추가하기

앞의 3단계에서 작성한 원시 파일 'demo.c'를 컴파일하려면 프로젝트 파일 그룹에 포함시켜야 한다. 원시 파일 'demo.c'를 프로젝트 파일 그룹에 포함시키기 위하여 [그림4.2-9]과 같이 주 메뉴에서 『Project - Add to Project - Files』을 선택한다.



[그림 4.2-9] 프로젝트 파일 그룹에 원시 파일을 추가하기 위한 메뉴 선택

[그림 4.2-9]를 수행하면 [그림 4.2-10]와 같이 파일 선택 창이 나타난다. 원시 파일 'demo.c'를 선택하고 나서, 하단의 『열기』 버튼을 누르면 『Project Workspace』 창에 원시 파일이 추가된다.



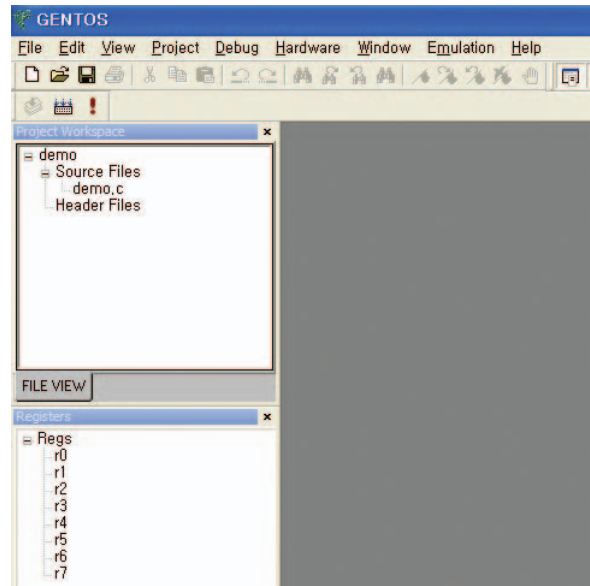
[그림 4.2-10] 원시 파일을 프로젝트 파일 그룹에 포함시킴

원시 파일 'demo.c'를 프로젝트 파일 그룹에 추가한 결과는 [그림 4.2-11]



과 같이 『Project Workspace』 창을 보면 알 수 있다.

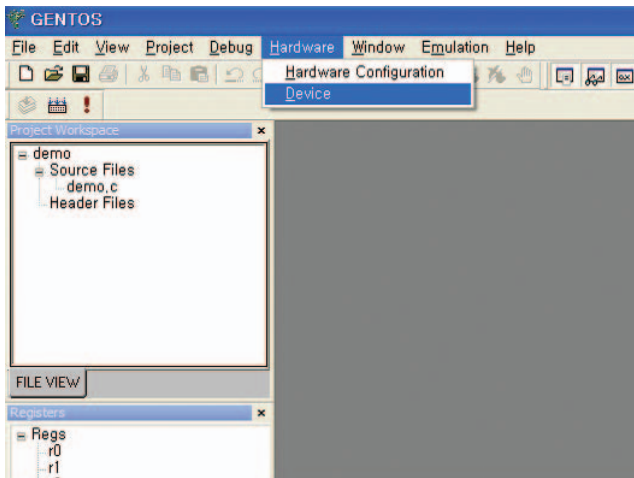
원시 파일 'demo.c'를 프로젝트 파일 그룹에 추가한 결과로 [그림 4.2-11]과 같이, 『Project Workspace』 창에 'demo - Source Files - demo.c'의 형식으로 가지가 구성된다. 만일 [그림 4.2-11]과 같이 되지 않았거나 원시 파일이 잘못된 경우에는 『Project Workspace』 창에서 잘못된 파일을 마우스 좌측 버튼을 이용하여 선택한 다음에 마우스의 우측 버튼을 클릭하여 나타나는 창에서 『Remove』 버튼을 클릭하여 제거하고 다시 위와 같은 작업을 수행하여야 한다.



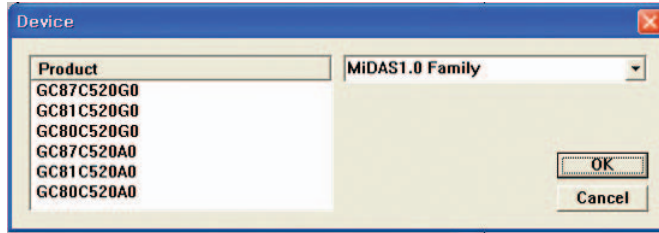
[그림 4.2-11] 원시 파일을 프로젝트 파일 그룹에 추가한 결과

#### [5단계] 컴파일을 위한 조건 설정하기

원시 파일을 컴파일하려면 먼저 컴파일과 관련된 설정을 해 주어야 한다. 이를 위해 [그림 4.2-12]과 같이 주 메뉴에서 『Hardware - Device』를 선택하면 [그림 4.2-13]의 설정 화면이 나타난다.

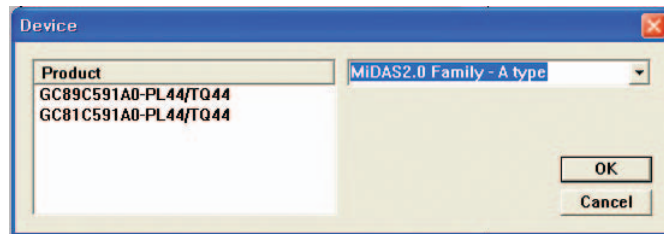


[그림 4.2-12] 주메뉴에서 『Hardware - Device』선택



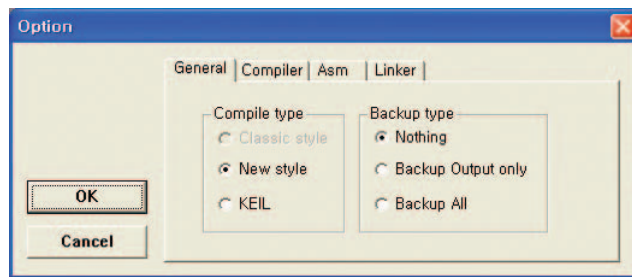
[그림 4.2-13] Device 설정 화면

[그림 4.2-14]처럼 'MiDAS2.0 Family - A type'를 선택하고 「OK」 버튼을 클릭한다.



[그림 4.2-14] MiDAS2.0 Family - A type 선택

그리고 나서 주 메뉴에서 『Project - Options』를 선택하면 [그림 4.2-15]의 조건 선택창이 나타난다.



[그림 4.2-15] 컴파일 type 옵션

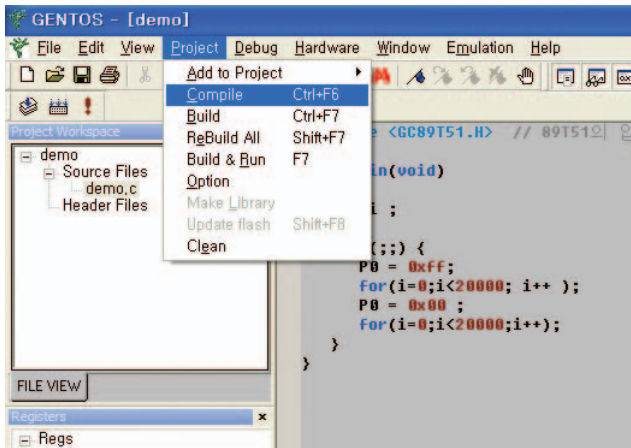
[그림 4.2-15]와 같이 'New style'과 'Nothing'을 선택한다.

#### [6단계] 원시 파일 'demo.c'을 컴파일하기

원시 파일 'demo.c'를 컴파일하기 위해 『Project Workspace』 창의 원시

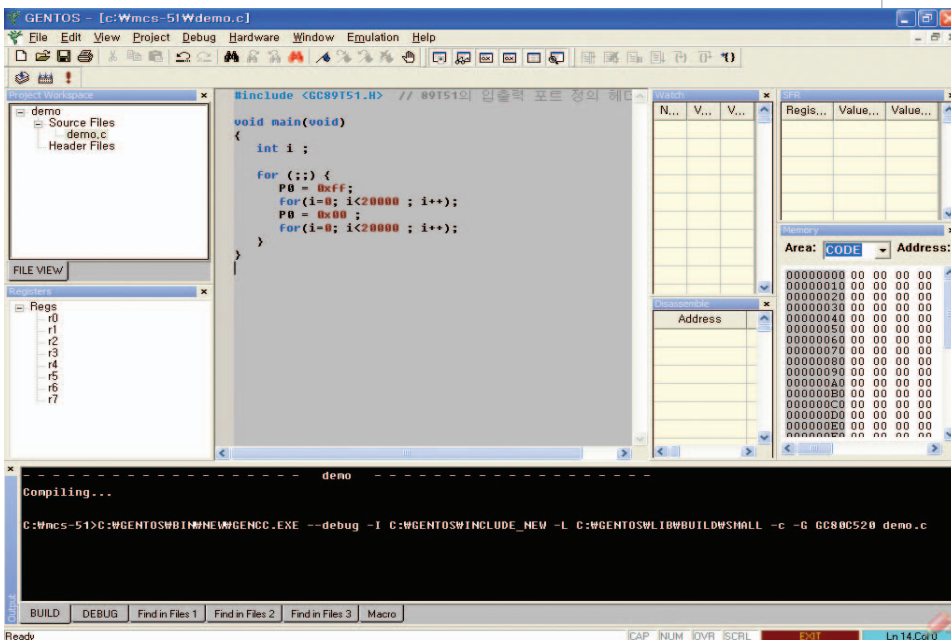


파일 가지를 두 번 눌러 원시 파일을 편집 창에 올리고 [그림 4.2-16]와 같이 주 메뉴에서 『Project - Compile』 메뉴를 실행한다.



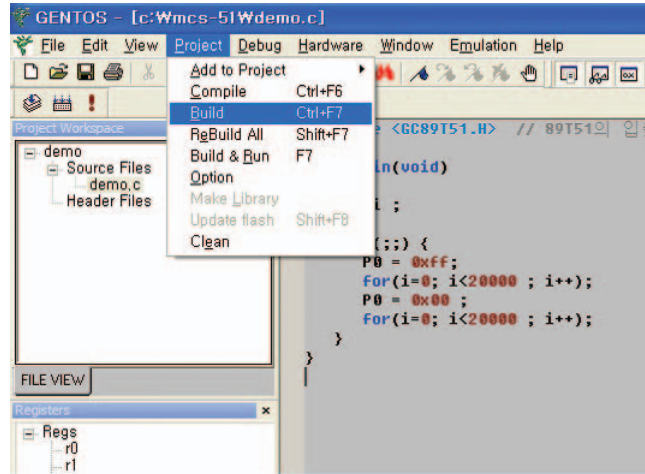
[그림 4.2-16] 'Compile' 메뉴를 실행 화면

실행한 결과는 [그림 4.2-17]과 같이 출력창을 통해 나타난다. 출력창의 결과에 오류가 있으면 오류 문장을 보고 오류를 수정한 다음 컴파일 과정을 다시 수행한다.

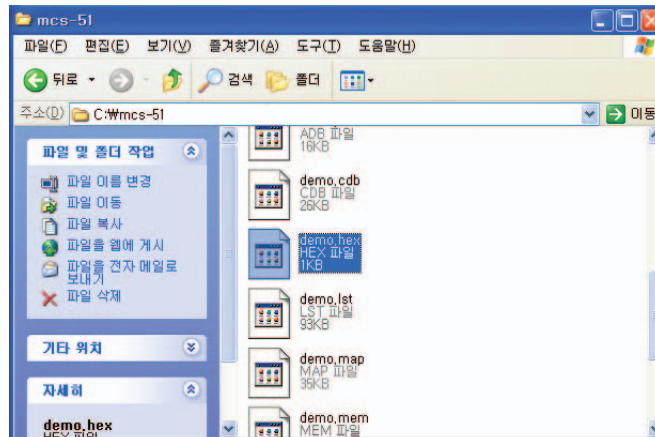


[그림 4.2-17] Compile 실행 후 출력 창

이제 [그림 4.2-18]과 같이 주 메뉴에서 『Project - Build』 메뉴를 실행한다. 만일 컴파일과 빌드(Build)가 모두 정상적으로 이루어졌다면, 작업 공간 디렉토리에 [그림 4.2-19]와 같이 hex 파일이 생성된다.



[그림 4.2-18] Build 실행 화면



[그림 4.2-19] demo.hex 파일 생성 화면

지금까지 GENCC 컴파일러를 이용하여 프로그램 원시 파일을 작성하고 컴파일하여 실행 파일을 생성하는 방법을 학습하였다. 다음 절에서는 코아리버서에서 개발한 MCU Programmer를 이용하여 생성한 demo.hex파일을 89T51 라인트레이서로 전송하고 라인트레이서를 구동시켜 보도록 하자.



### 3. 라인트레이서로 프로그램 전송 방법

#### 1) 89T51 라인트레이서로 프로그램 전송을 위한 사전 작업

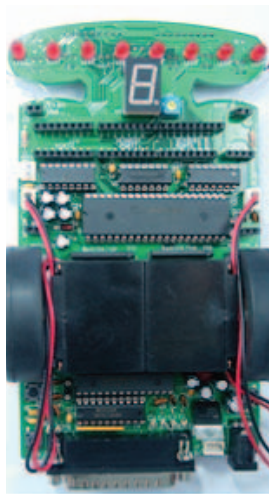
이제부터 4.2절에서 작성하고 컴파일한 데모 프로그램을 89T51 라인트레이서로 전송해서 라인트레이서의 구동을 확인한다. 전송을 위하여 먼저 수행할 사전 작업을 알아본다.

##### [1단계] 프로그램 전송을 위한 필요 물품 준비

[그림 4.2-20]에 제시된 프로그램 전송을 위한 필요 물품을 준비한다.



신호 점퍼선



GC89T51 라인트레이서



ISP 통신 케이블



5[V] 어댑터

[그림 4.2-20] 프로그램 전송을 위한 필요 물품

[그림 4.2-20]과 같이 프로그램을 전송하려면 89T51 기반의 라인트레이서, 라인트레이서 기판 위에 위치한 각각의 신호 커넥터를 연결하기 위한 점퍼선, 그리고 PC로부터 프로그램 전송을 받기 위한 ISP 통신 케이블, 라인트레이서에 전원을 공급하기 위한 5[V] 어댑터(혹은 1.5[V] 배터리 4개)가 필요하다.



## [2단계] GC89T51 라인트레이서 기판의 신호 연결

89T51 라인트레이서는 실험·실습 상황에 따라 쉽게 회로를 구성할 수 있도록 기판 상의 CPU와 주변 장치들의 입출력 포트와 제어 포트에 헤더 커넥터를 연결하였다. 앞절에서 생성한 'demo.hex' 파일을 라인트레이서에 서 실행시키려면 먼저 배선을 해야 한다.

### • 원시 파일 demo.c

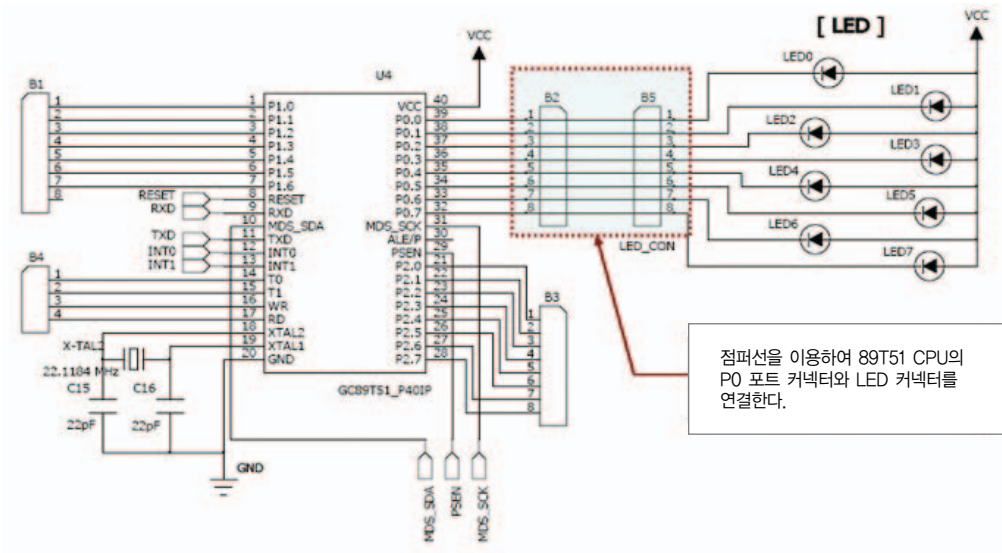
```
#include <GC89T51> // 89T51의 입출력 포트 정의 헤더 파일

void main(void)
{
    int i ;

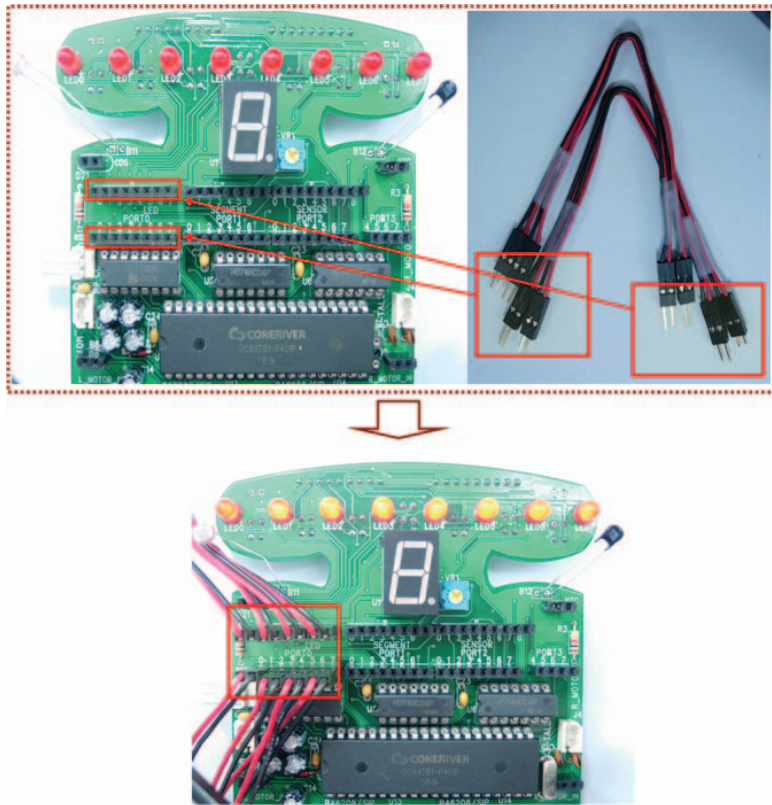
    for(;;)
    {
        P0 = ~0xff;    // 89T51의 P0을 통해 LED를 켜다.
        for(i=0;i<20000;i++ ); // 약 0.1초의 시간을 지연시킨다.

        P0 = ~0x00;    // 89T51의 P0을 통해 LED를 끈다.
        for(i=0;i<20000;i++); // 약 0.1초의 시간을 지연시킨다.
    }
}
```

위 원시 파일 'demo.c'는 설명에서도 알 수 있듯이 89T51의 P0 포트를 통해 약 0.1초 주기로 5[V]의 신호를 내어 보내는 역할을 수행한다. 그러나 89T51의 P0 포트의 출력 신호를 직접 눈으로 확인할 수 없어 [그림 4.2-21]과 [그림 4.2-22]처럼 LED와 연결시켜 P0 포트의 출력 신호를 확인한다.



[그림 4.2-21] 회로도로 본 P0 포트 커넥터와 LED 커넥터의 연결

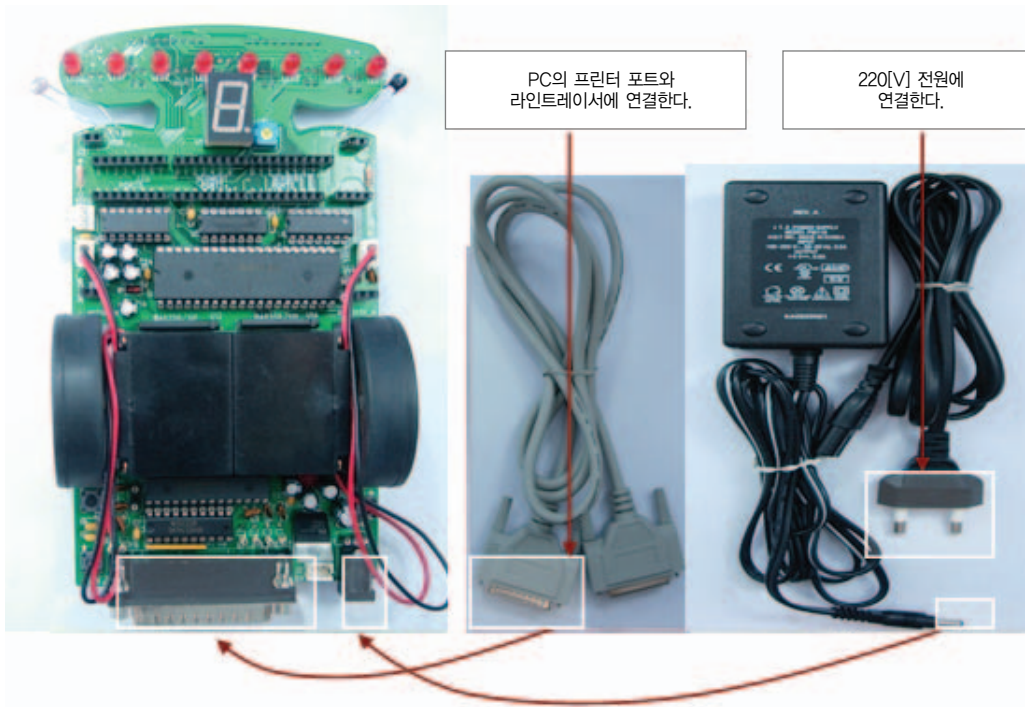


[그림 4.2-22] 실제 사진으로 본 P0포트 커넥터와 LED 커넥터의 연결

### [3단계] 89T51 라인트레이서 기판의 전원 및 ISP 통신 케이블 연결

89T51 라인트레이서는 5[V]의 전원을 사용한다. 여기서는 5[V] 어댑터를 사용하였으나 편의에 따라 1.5[V] 건전지 4개를 사용해도 무관하다. ISP 통신 케이블을 통하여 PC로부터 라인트레이서로 프로그램이 전송된다.

[그림 4.2-23]와 같이 라인트레이서의 전원과 ISP 통신 케이블을 연결한다.



[그림 4.2-23] 라인트레이서 전원 및 ISP 통신 케이블 연결

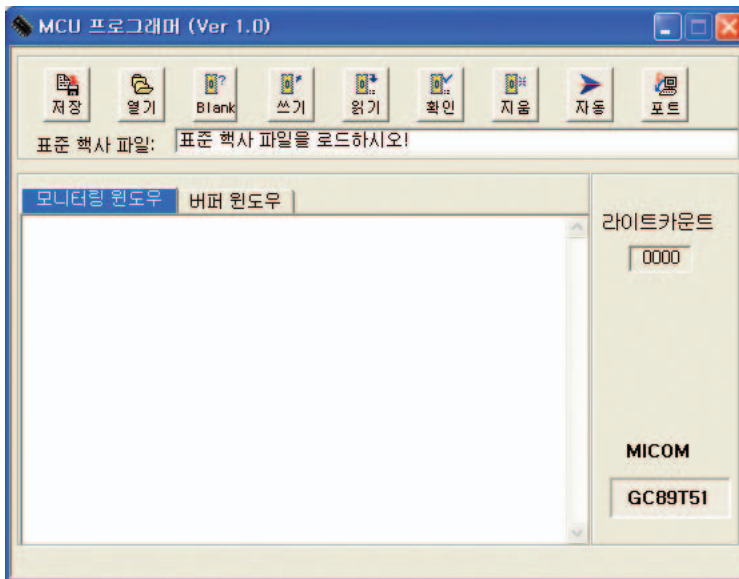
라인트레이서 실습을 위한 준비가 모두 완료되었다. 이제부터는 라인트레이서에 데모 프로그램(demo.hex)을 전송하고 실제로 로봇을 구동시켜 보자.

## 4. MCU 프로그래머 사용법

MCU 프로그래머(Programmer)를 실행하면 [그림 4.2-24]과 같은 초기







화면이 나타나고 우측 하단에 'MICOM'이라는 단어 아래 있는 CPU를 확인하는 창에 라인트레이서에서 사용된 CPU(GC89T51)가 검색된다. CPU를 정상적으로 검색하려면 라인트레이서에 전원을 연결하고, PC와 라인트레이서 간에 ISP 통신 케이블을 연결하여야 한다. 만약 CPU가 확인되지 않고 **확인 실패** 가 나타나면, 전원부와 ISP 통신 케이블 연결 상태를 먼저 확인한 다음에 **확인 실패** 아이콘을 눌러 CPU를 재검색한다. 그래도 라인트레이서의 CPU를 인식하지 못해서 **확인 실패** 가 계속 나타나면 라인트레이서 보드 및 ISP 케이블 불량을 수정하는 작업을 수행한다.



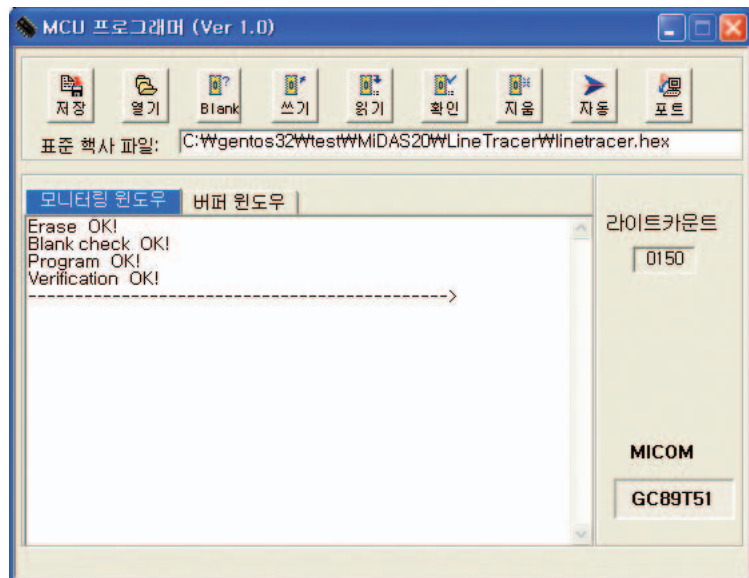
[그림 4.2-24] MCU Programmer 초기 화면

[그림 4.2-24]과 같이 MCU P프로그래머가 라인트레이서의 CPU를 정상적으로 인식하였다면, 다음의 아이콘을 이용하여 앞서 생성한 'demo.hex' 실행 파일을 라인트레이서로 전송한다.

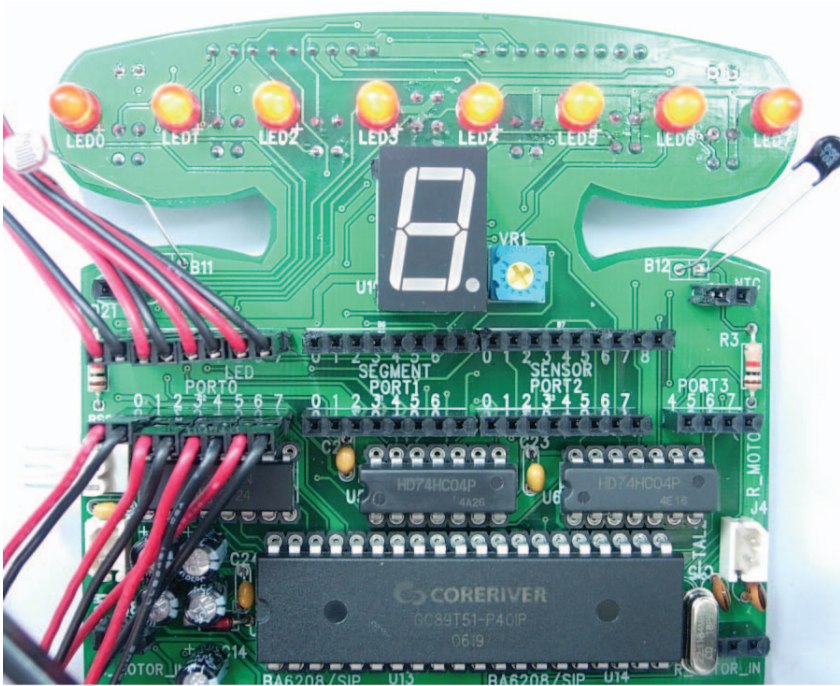
-  '열기' 아이콘을 눌러 전송할 실행 파일(예: demo.hex)을 읽어 들인다.

-  '쓰기' 아이콘을 누르면 89T51로 실행 파일을 전송하는데, 전송 중 데이터가 정확히 입력되지 않아서 비정상적인 동작이 일어날 수 있으므로 다음의 '자동' 아이콘을 이용하여 전송한다.
-  '자동' 아이콘을 누르면 89T51의 메모리를 지우는 것부터 데이터 무결성 검사까지 자동적으로 수행된다.
-  '지움' 아이콘을 누르면 89T51의 메모리에 있는 데이터 값이 모두 지워진다.

이와 같이 MCU 프로그래머의 아이콘들을 이용하여 'demo.hex' 파일을 전송하면 [그림 4.2-25]와 같이 각 항목에 대해 모두 'OK' 라는 확인 메시지가 텍스트 창에 나타나며, [그림 4.2-26]와 같이 라인트레이서에 탑재된 8개의 LED가 약 0.1초의 주기로 깜빡이기 시작한다.



[그림 4.2-25] demo.hex 파일 다운로드 완료 화면



[그림 4.2-26] demo.hex 파일 전송 결과

이와 같은 결과가 나타나지 않으면 지금까지의 과정을 다시 살펴보거나 프로그램의 원시 파일에 오류가 존재하는지 다시 점검한다.



.....

.....

.....

.....

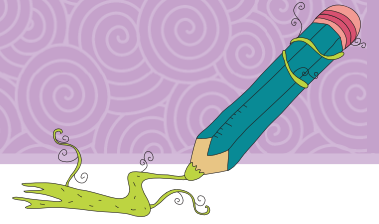
.....

.....

.....



# 단원 학습 정리



01.

컴파일러는 원시 파일을 기계어로 바꿔서 이진 형태의 실행 파일을 만드는 컴파일 과정을 실행한다. 마이크로로봇 구동을 위해서는 GENCC, IAR-C, Keil-C 등의 전용 컴파일러를 사용한다.

02.

GENCC는 마이크로 로봇제작에 사용되는 코아리버에서 개발한 인텔 8051 호환 마이크로프로세서 전용으로 사용하도록 개발되었다. 실행 파일의 크기와 속도에서 최적화된 결과를 생성할 뿐 아니라 프로그램 메모리의 크기에 제약 없이 사용할 수 있는 장점이 있다.

03.

라인트레이서로 프로그램 전송 방법

## 1) 프로그램 전송을 위한 필요 물품 준비

- 라인트레이서, 점퍼선, ISP 통신 케이블, 5[V] 어댑터

## 2) GC89T51 라인트레이서 기판의 신호 연결

- 라인트레이서 기판 상의 CPU와 주변장치들의 입출력 포트와 제어 포트의 헤더 커넥터를 연결

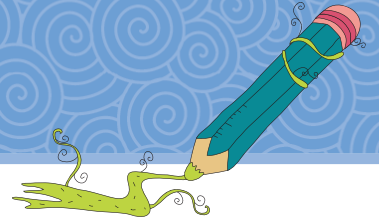
## 3) GC89T51 라인트레이서 기판의 전원 및 ISP 통신 케이블 연결

- PC의 프린터 포트와 라인트레이서 연결

〈MCU 프로그래머 사용법은 4장 참고〉



# 단원 정리 문제



01

다음 중 원시 파일의 확장자는?

- ① .c                      ② .obj                      ③ .exe
- ④ S.com                  ⑤ .h

02

다음 중에서 컴파일러의 역할은?

- ① 원시 파일을 작성하는 편집기의 역할을 한다.
- ② 원시 파일에서 오타를 찾아내는 역할을 한다.
- ③ 컴퓨터에서 프로그램을 실행시키는 역할을 한다.
- ④ 원시 파일을 실행 파일로 번역하는 역할을 한다.

03

다음 중 실행 파일의 확장자를 모두 찾아보시오.

- ① .exe                      ② .rel                      ③ .obj
- ④ .hex                      ⑤ .com

04

컴파일러에서 마이크로로봇을 프로그램 할 때 포함시키는 헤더 파일은?

05

ISP 통신 프로그램의 역할은?

06

원시 파일을 작성하고 컴파일하기 위해 작업 공간을 생성하는 과정을 기술해 보시오.

07

프로그램을 PC에서 마이크로로봇으로 전송하기 위하여 필요한 물품들을 열거해 보시오.

# 03. C 언어 기초

## 학습목표



- C 언어의 유래와 특징을 설명할 수 있다.
- 변수와 상수를 사용하여 C 언어 프로그램을 작성할 수 있다.
- C 언어 프로그램에서 필요한 연산자를 사용할 수 있다.
- 프로그램에서 어떤 부분을 반복하거나 상황에 따라 건너뛰게 할 수 있다.
- 자주 사용할 부분을 함수로 정의해서 사용할 수 있다.



### [ALGOL60]

ALGOL은 1965년 국제표준화기구 (ISO) 회의에서 채택된 프로그래밍 언어로서 algorithmic language의 약칭이다. 수치 계산 절차적인 알고리즘을 표준화된 산술 형식으로 기술하는 데 편리하고, 또한 이것을 컴퓨터에 정확하게 전달할 수 있는 프로그래밍 언어인 동시에 국제 대수언어에서부터 발전된 컴파일러 언어의 하나이다. ALGOL은 국제정보처리연합(IFIP)에서 1958년에 제정한 ALGOL58을 출발점으로 하여, 이를 수정하여 1960년에 발표한 ALGOL60부터 실제로 과학기술 계산용 언어로 이용하게 되었다.

지능형 로봇의 목적은 단순히 움직이는 것이 아니라 주변 상황을 파악하여 명령들을 수행하는 것이다. 이렇게 하려면 주변 환경에 대한 정보를 수집하고 분석하면서 명령을 실행하도록 프로그램을 작성해야 한다. 이런 목적으로 다양한 프로그래밍 언어들을 사용하지만, 원시 파일을 작성할 때 효율성이 높고 생성된 실행 파일의 속도가 빠른 **C 언어를 이용하여 프로그램을 작성하는 방법**을 학습한다.



## 1. C-언어는?

ALGOL60이라는 프로그래밍 언어를 시작으로 많은 시간과 노력을 들여 프로그램을 작성하기에 적절한 언어를 발전시켜 왔다. C 언어는 1972년 벨 연구소에서 데니스 리치(Dennis Ritchie)가 만들었다. 이후 C 언어가 여러 종류로 분화되자 ANSI-C(American Standard Institute)라는 C언어의 표준을 제정하여 권고함으로써 현재의 C 언어가 생겨났다. **프로그래머들이 C**



## 언어를 많이 사용하는 이유는 무엇일까?

### (1) C 언어의 특징

- ① 다양한 운영체제(MS-DOS, Windows, Unix 등)에서 프로그램을 개발할 수 있어서 호환성이 높다
- ② 다양한 데이터의 형태와 연산자를 제공하여 다양한 응용 프로그램을 구현할 수 있다.
- ③ 다른 언어(Cobol, Pascal 등)보다 원시 파일 작성 과정이 효율적이고 실행파일의 수행 속도가 비교적 빠르다.
- ④ 어셈블러와 같은 저급언어의 특징과 베이직, 코볼 등의 고급언어의 특징을 고루 갖춘 중성적 언어로서 시스템 프로그램과 일반 응용 프로그램 작업을 모두 수행할 수 있다.
- ⑤ 대형 프로그램을 작성할 때 프로그램 추적과 오류 수정이 편리해서 작업이 쉽다.

### (2) C 언어의 기본 형태

C 언어는 컴파일러를 사용하여 원시파일에서 목적 파일을 생성하고, 컴파일러가 제공하는 함수들이 있는 파일과 연결하여 최종 실행 파일을 생성하는 언어다.

프로그램이 실행될 대상이 마이크로 로봇이면 실행 파일의 확장자가 HEX인 헥사 파일이고, 일반 컴퓨터이면 확장자가 EXE인 실행 파일을 생성한다.

C 언어 프로그램의 기본 형태는 다음과 같다.



#### • 운영체제

컴퓨터를 작동시키고 운영을 도맡아 관리하여 사용자의 응용 프로그램이 효율적으로 실행될 수 있는 환경을 제공하는 기본 소프트웨어 또는 총괄 제어 프로그램. 보통 약어로 OS라고 한다.

#### • 유닉스 (UNIX)

다수 사용자를 위한 컴퓨터용 운영체제의 하나이다. 유닉스는 미국 벨연구소에서 프로그래밍 연구와 개발을 촉진시키는 환경조성을 목적으로 처음 선보였다. K. 톰프슨이 DEC사의 PDP-7 컴퓨터를 위해 처음에는 어셈블리어로 작성하였으나 1972년 D. 리치가 고급언어의 하나인 C 언어로 다시 작성했다.

#### 코볼 (Cobol)

컴퓨터 프로그래밍 언어의 하나로 일상생활에서 쓰이는 영어회화와 비슷한 구어체 문장 형태로 기술할 수 있도록 설계되어 입출력 데이터나 보고서 작성, 그밖의 각종 프로그램 작성이 쉬우며 다른 프로그래머가 보아도 쉽게 이해할 수 있다. 1968년 미국에서 사무처리 언어의 표준이 되었고, 1974년 미국 표준 코볼(ANSI COBOL)이 완성되었다.

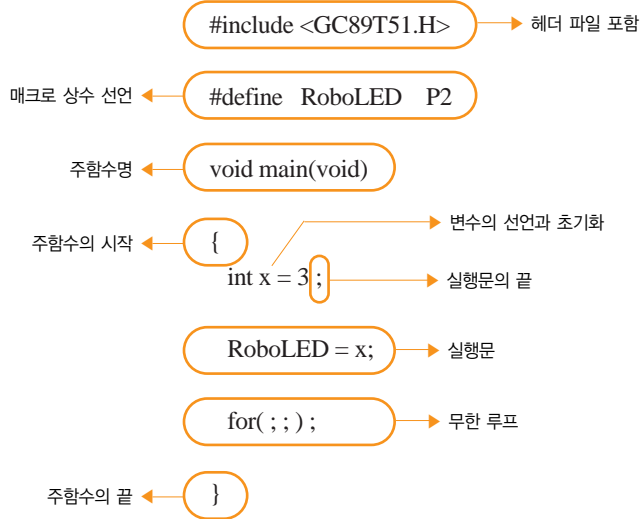
#### • 베이직(Basic)

1963년 무렵 미국 다트머스대학교의 J. K. 케메니와 T. E. 커츠가 개발한 대화형 고급 프로그래밍 언어이다. 비전문가들이 시분할(time-sharing) 처리 환경에서 간단히 사용할 수 있게 하는 것을 목표로 설계되었다. 언어가 간단하다는 점과 대화할 수 있는 환경을 제공하고 있다는 유용성 때문에 1975년 전후에 이용되기 시작한 마이크로 컴퓨터용 언어로 널리 보급되었다. 1984년에 국제 규격이 제정되었다.



### 여기서 잠깐

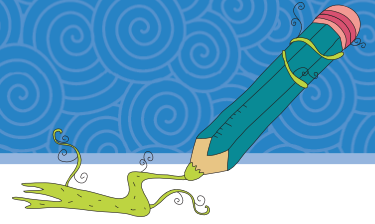
- ▶ 함수의 시작은 여는 중괄호({)로 함수의 끝은 닫는 중괄호(})를 사용한다.
- ▶ 실행문의 끝에는 세미콜론(; )을 찍는다.



[그림 4.3-1] 89T51 라인트레이서 전원 인터페이스

- 1 헤더 파일에는 프로그램을 쉽게 할 수 있도록 로봇을 움직이는 MPU (Microprocessor Unit)의 주요 기능을 미리 선언해 놓았다. 헤더 파일을 사용하려면 '#include'로 시작되는 문장을 사용해서 헤더 파일을 프로그램에 포함시킨다.
- 2 헤더 파일은 컴파일러 판매 회사 또는 마이크로프로세서 제조회사에서 제공하며, 마이크로프로세서와 맞는 헤더 파일을 사용하여야 한다.
- 3 로봇의 전원을 켜면, 주함수인 main() 함수로부터 프로그램을 시작한다. C 언어에는 main() 함수가 반드시 있어야 한다.
- 4 함수의 시작과 끝은 중괄호로 나타내고, 중괄호 안에 실행문들을 넣는다.
- 5 실행문의 끝에는 항상 세미콜론(; )을 찍어야 한다.

# 단원 정리 문제



01

C 언어의 특징이 아닌 것은?

- ① 호환성이 높다.
- ② 여러 가지의 응용프로그램을 만들 수 있다.
- ③ 오류 수정이 불편하다.
- ④ 중성적 언어이다.

02

C 언어의 주함수를 나타내는 함수는?

- ① main()
- ② sub()
- ③ #define
- ④ #include

03

C 언어 함수의 시작과 끝은 무엇으로 나타내는가?

시작 : (                    )  
끝 : (                    )

04

한 문장의 끝은 무엇으로 표현하는가?

- ① 세미콜론
- ② 콜론
- ③ 따옴표
- ④ 느낌표

05

#include <GC89T51.H> 라는 문장이 나타내는 의미를 쓰시오.





## 2. 변수와 상수

C 언어 프로그램에서는 변수와 상수를 사용한다. 프로그램을 작성하기 위해서는 변수와 상수의 개념과 활용 방법을 확실히 알아야 한다.

### (1) 변수의 이해

변수는 채팅방이나 온라인 게임에서 사용하는 방의 이름과 같다. 컴퓨터는 메모리에 데이터를 저장한다. 메모리는 데이터를 저장하는 여러 개의 방으로 분할되어 있다. 데이터를 꺼내거나 저장할 때 사용할 방을 구분하여 이름을 붙여놓은 것을 변수라고 한다. 한 개의 변수에는 하나의 데이터만 저장할 수 있다.

C 언어에서 변수를 사용할 때 꼭 지켜야 할 것이 있다.

○ 같은 이름으로 두 개 이상의 변수를 선언하여 사용할 수 없다.

‘x’ 라는 변수를 선언하고, 다시 다른 ‘x’ 라는 변수를 선언할 수 없다.

○ 대문자와 소문자를 서로 다르게 구분한다.

대문자 ‘X’ 라는 변수와 소문자 ‘x’ 라는 변수는 서로 다른 변수이다.

또한, ‘Rabo’, ‘rabo’, ‘RABO’ 라는 변수는 모두 다른 변수이다.

○ 함수 안에서 선언된 변수는 해당 함수에서만 사용할 수 있다.

함수 안에서 선언된 변수는 그 함수에서만 존재한다. 여러 함수가 있으면 각각의 함수에서 ‘x’ 라는 변수를 모두 사용할 수 있다.

○ C 언어에서 기본으로 사용하는 예약어는 변수 이름으로 사용할 수 없다.

C 언어에서 사용하는 많은 예약어(컴파일러에서 기본적으로 사용하고 있는 단어)들은 변수 이름으로 사용할 수 없다. 주함수를 나타내는 ‘main’ 이라는 예약어를 변수 이름으로 사용하면 컴파일 과정에서 오류가 발생한다.

### (2) 변수 선언하기

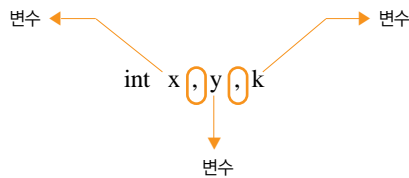
변수를 선언하는 것은 메모리의 일부분을 사용하겠다고 컴퓨터에게 알려



주고, 사용할 부분의 크기와 이름을 정하는 것이다.



메모리를 사용할 크기를 정하는 것이 “데이터 형”이다. 데이터 형에는 여러 가지가 있다.

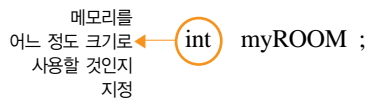


같은 데이터 형의 변수들을 여러 개 선언하려면 변수 이름들 사이에 쉼표 (,)를 넣어 구분한다. 변수에는 임의의 값이나 사칙연산을 한 결과가 저장될 수 있다.

변수에 값을 넣으려면 ‘=’ 기호를 이용한다. ‘=’ 기호는 수학에서는 “오른쪽의 값과 왼쪽의 값이 같다”라는 뜻이지만, C 언어에서는 “오른쪽의 값을 왼쪽의 변수에 넣는다.”라는 뜻이다.

### (3) 변수의 데이터 형

변수를 선언할 할 때는 데이터 형으로 변수의 크기도 함께 선언한다.



C 언어에서 데이터 형을 나타내는 예약어를 변수 이름 앞에 쓰면 변수가 사용할 메모리의 크기가 결정된다. C 언어에서 데이터 형을 나타내는 예약



#### signed/ unsigned

데이터를 표현하는 범위를 음수와 양수 모두를 포함할 것인지 양수만 표현할 것인지를 결정하는 예약어이다. signed는 음수와 양수 모두를 포함한다는 의미이며 생략될 수 있다. unsigned는 양수만 표현한다는 의미이며 생략할 수 없다. 데이터 형을 나타내는 예약어 앞에 붙이며, 아무것도 없으면 signed로 해석해서 음수, 양수 모두를 표현한다.

어로는 다음과 같은 것들이 있다.

char(문자), int(정수), long(긴 정수), float(실수), double(긴 실수), short(짧은 정수), signed(부호 있음), unsigned(부호 없음)

선언된 변수에 데이터 형의 범위를 넘는 값을 넣으면 프로그램 실행에 중대한 오류가 발생할 수 있다.

마이크로로봇의 프로그램을 작성할 때 주로 사용할 데이터 형만 다루겠다.

### 1) char형 데이터 형태 ( 문자형 )

char형 데이터는 1 바이트를 사용하며, 2를 8번 곱한 크기인 256개를 표현할 수 있다. 표현 범위는 음수와 양수를 모두 표현하면 -128부터 +127까지를 표현하며, 양수만 표현하면 0부터 255까지 표현한다. 보통 char형은 음수와 양수를 표현하며, char 앞에 unsigned를 붙이면 양수만 표현함을 의미한다.

[표 4.3-1] 문자형 데이터의 종류

데이터 형	크기	범위
char	1byte	-128 ~ +127
signed char		
unsigned char		0 ~ 255

char형 데이터 형은 문자형이라고도 부르며, 숫자에 대응하는 문자를 표현할 수 있다. 단, 음수 부분은 제외한다.

2진수만을 인식하는 컴퓨터는 사람의 문자를 인식하지 못함으로 문자를



2진수에 대응시켜 표현한다. 이를 위해 ASCII(아스키) 코드를 이용하여 2진수에 대응하는 문자를 표준으로 정해 사용하고 있다. 예를 들면, 문자 'a'는 숫자로 65로 대응한다. 특별한 의미가 있어서가 아니라, 컴퓨터와 사람이 알 수 있도록 정했을 뿐이다.

문자 자체를 나타내려면 그 문자를 홑 따옴표 (‘ ’)안에 넣어서 표현한다.

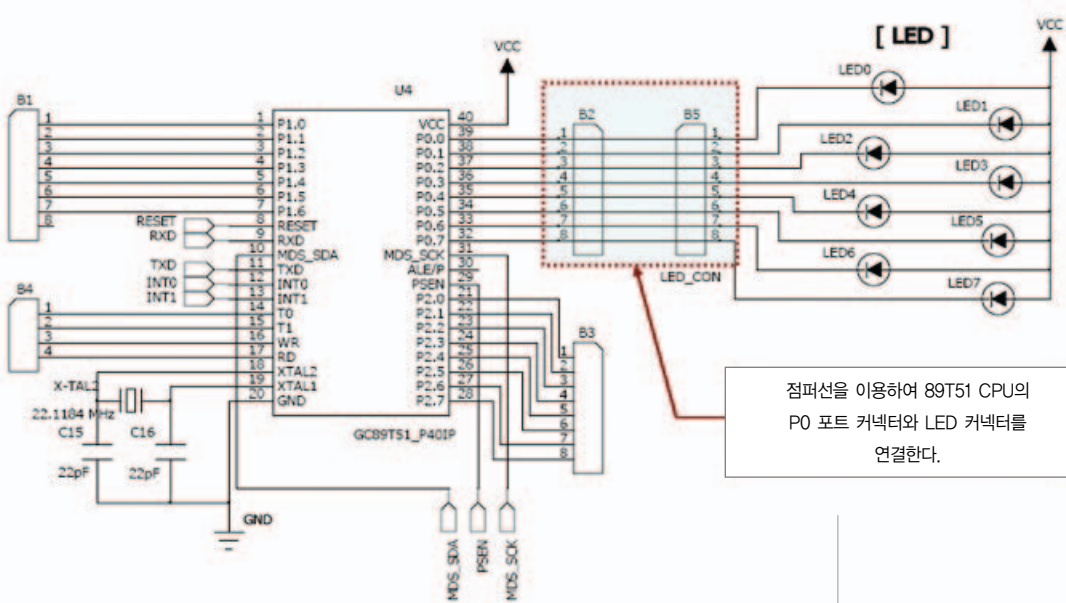
5는 숫자이기도 하고 문자이기도 하다. 그러나 컴퓨터는 숫자 5의 경우 그대로 5이지만 문자 '5'는 ASCII 코드에 대응하는 숫자로 인식한다.

앞으로 예제에서 제시될 프로그램의 동작은 [그림 4.3-2]와 같이 P0 포트에 LED를 연결하여 확인한다.



#### 아스키 코드 (ASCII code)

미국에서 제정된 데이터 처리 및 통신 시스템 상호간의 정보 교환용 7bit 표준 코드, ASCII라고도 하며 이는 American Standard Code for Information Interchange(미국 정보 교환용 표준 코드)의 약칭이다. 96개의 대문자, 소문자, 숫자, 특수 문자(연산자, 등호, 부등호, 괄호, 문장 부호 등과 같은 도형 문자)와, 32개의 제어 문자(서식 변경, 전송 데이터 개시·종료 등의 제어에 이용되는 특수 기능 문자)를 포함하는 128개 문자를 입력할 수 있다.



[그림 4.3-2] 변수 실습용 프로그램 동작 확인 회로



#### 예제 4-3

부호가 있는 문자형 변수에 100이라는 수를 대입하여 출력하면 어떻게 되는지 로봇에 연결된 LED로 확인해 보자.

```
#include<GC89T51.H> //헤더 파일을 포함시킨다.
```



- `#define RobotLED P0`  
LED가 연결된 P0 포트를 RobotLED라는 매크로 상수로 지정했다.
- `char ch1`  
ch1이라는 변수를 부호가 있는 문자형으로 선언했다.
- `ch1 = 100`  
ch1이라는 변수에 100이라는 값을 대입했다.
- `RobotLED = ~ch1`  
변수 ch1에 저장되어 있는 100의 값을 반전시켜 RobotLED로 출력한다. LED는 연결된 포트에 저전압 신호가 출력되어야 켜지므로, '~' 연산자로 반전시켜 출력한다.
- `for ( ;; )`  
무한 루프(무한 반복)를 뜻한다. 마이크로프로세서의 프로그램은 반드시 무한 반복되어야 한다. 무한 반복되지 않으면 어떻게 될까? 예를 들어 에어컨에 설치된 프로그램이 무한 반복되지 않고 한번 실행하고 끝난다면, 에어컨을 조작할 때마다 에어컨을 다시 켜야 하는 불편을 감수해야 한다. 따라서 마이크로프로세서의 프로그램은 반드시 무한 반복되어야 한다.

실행 결과를 보면 모두 세 개의 LED가 켜진다. 포트에 연결된 LED는 16진수로 표시 된다. 현재 켜진 LED를 16진수로 변환하여 읽으면 0x64이다. 이는 10진수 100을 16진수로 변환하면, 0x64인 것으로 확인할 수 있다.

```
//매크로 상수 RobotLED의 선언
#define RobotLED P0

void main(void){
    char ch1;           // 부호 있는 문자형 변수의 선언
    ch1 = 100;

    for ( ;; ) {       // 무한 루프
        RobotLED = ~ch1; //RobotLED에 ch1의 값을 출력한다.
    }
}
```

### 실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	○	○	●	●	○

10진수 100을 2진수로 변환하면 01100100<sup>(2)</sup>이며, 이를 반전시킨 값인 10011011<sup>(2)</sup>을 출력한다.

## 2) int형 데이터 (정수형)

정수형(int) 데이터는 크기가 2 바이트이므로 0부터 65535까지의 허용 범위를 갖는다. char형(문자형)보다 훨씬 큰 수를 저장할 수 있으므로 당연히 char형 데이터를 포함할 수 있다. 부호 있는 정수형은 int형이라고 하며, 부호 없는 정수형의 경우 unsigned int라고 한다.

[표 4.3-2] 정수형 데이터의 종류

데이터 형	크기	범위
signed short int	2byte	- 32768 ~ + 32767
signed int		
int		
unsigned short int		0 ~ 65535
unsigned int		



#### 예제 4-4

[예제4-3]의 프로그램에서 문자형 변수 대신 부호가 있는 정수형 변수를 사용할 경우 어떻게 되는지 로봇에 연결된 LED로 확인해 보자.

```
#include <GC89T51.H> //헤더 파일을 포함시킨다.

// 매크로 상수 RobotLED의 선언
#define RobotLED P0

void main(void){
    int i = 300; //i를 정수형 변수로 선언하고 초기값으로 300을 넣는다.

    RobotLED = ~i; // RobotLED에 i의 값을 출력한다.

    for (;); // 무한루프
}
```

#### 실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	●	○	●	○	○

[예제 4-4]처럼 정수형 변수 값을 출력 포트를 이용하여 출력할 경우 1바이트로 출력 되므로 자원의 낭비가 발생한다. 따라서 정수형 변수는 수식 계산을 위해서 사용하는 것이 바람직하다.

### 3) 그 외의 데이터 형태 (long, float, double 형)

데이터 형에는 문자형과 정수형 외에도 long(긴 정수형), float(실수형), double(긴 실수형)형의 데이터 형태가 있다.

[표 4.3-3] 그 외의 데이터 형 종류

데이터 형	크기	범위
long	4byte	$2^{-31} \sim (2^{31}-1)$
float	4byte	$3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
double	8byte	$1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$



#### • int i=300

i이라는 변수를 부호있는 정수형으로 선언하고, 초기값으로 300을 넣는다.

#### • RobotLED = ~i

RobotLED에 변수 i의 값을 반전시켜 출력한다.

실행 결과를 보면 모두 세 개의 LED가 켜진다. 켜진 LED를 16진수로 변환하여 읽어 보면 0x2C다. 하지만 10진수 300을 16진수로 변환하면 0x12C가 된다. 그럼 왜 LED로 표현된 결과는 다른 것일까? 포트에 연결된 LED가 8개인 것과 무슨 관계가 있을까? 여기서 사용하고 있는 마이크로프로세서는 출력 포트가 8비트로 구성되어 있다. 그러므로 16비트인 정수형 데이터를 출력하면 상위 8비트가 없어진다.



#### (4) 변수의 범위

변수는 사용하기 전에 그 변수가 사용되는 범위를 알아야 한다. 한정된 컴퓨터의 메모리를 효율적으로 사용하려면 변수가 사용되는 범위를 정확하게 알아야 한다.

변수에는 선언된 함수 안에서만 사용하는 지역 변수(Local Variable)와 모든 함수에 사용되는 전역 변수(Global Variable)로 분류된다.

예제를 통해 지역 변수와 전역 변수에 대해 알아보자.



##### 예제 4-5

main함수의 변수 ch와 sub함수의 변수 ch가 같은 변수인지 다른 변수인지를 확인해 보고, 지역 변수의 사용 범위에 대해 확인하여 보자.

```
#!/include<GC89T51.H>      //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED  P0

void delayTime(long d_t){
    long x;
    for(x=0;x<d_t;x++);
}

// 사용자 정의 함수
void sub(void){
    unsigned char ch;

    RobotLED = ~ch;
} // sub 함수를 종료하고 sub 함수를 부른 마지막 위치로 되돌아 간다.

void main(void){
    unsigned char ch = 240; // 부호없는 문자형 변수의 선언

    for(;;){
        RobotLED = ~ch;      // main 함수의 변수 ch 값을 출력
        delayTime(100000);  // 일정 시간 대기
        sub();              // sub() 함수를 실행하고 돌아온다.
        delayTime(100000);  // 일정 시간 대기
    } // for 문의 끝-조건식을 비교한다.
}
```



### 실행결과

⇒ RobotLED에 main 함수의 ch에 저장된 240(0×F0)을 출력한다.

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	○	●	●	●	●

⇒ RobotLED에 sub 함수의 ch에 임의로 저장된 값을 출력한다.

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	●	●	●	○	○

⇒ 위 결과를 반복하여 실행한다.



- `void delayTime(long d_t){...}`  
시간을 지연시키기 위해 사용자 정의 함수를 작성한다. 지연 시간을 지정할 수 있도록 시간 지연 함수를 호출할 때 전달되는 값을 저장할 변수를 long 형태로 선언하여 사용한다.

- `void sub(void){...}`  
사용자 정의함수 sub()를 작성하여 필요할 때 호출하여 실행한다.

- `unsigned char ch=240`  
ch라는 변수를 부호없는 문자형 변수로 선언하고, 초기값으로 240을 넣는다.

- `RobotLED = ~ch`  
로봇 LED에 변수 ch의 값을 반전시켜 출력한다.

- `sub()`

앞에서 작성한 사용자 정의 함수인 sub 함수를 호출하여 실행한다.

- `delayTime(100000)`

사용자 정의 함수인 시간 지연 함수를 호출하되, 지연될 시간을 지정하여 시간 지연 함수에 전달한다.

- main함수에서의 변수 ch값 240을 16진수로 보면 0xF0이다.

- sub함수에서의 변수 ch의 값은 메모리가 기존에 사용하던 임의의 값이 저장되어 있어 어떤 값이 출력될지 모른다.



### 예제 4-6

변수 ch1을 전역 변수로 선언하고 main 함수와 sub 함수에서 적용되는 결과를 통해 전역 변수의 적용 범위에 대해 확인하여 보자.

```
#include<GC89T51.H> //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED P0

int ch1,ch2; //전역 변수로 ch1을 정수형 변수로 초기화한다.

//시간 지연 함수
void delay(long d_t){
    long k;
    for(k=0;k<=d_t;k++);
}

// 사용자 정의 함수
void sub(void){
    RobotLED = ~ch1 ;    delay(100000);
    RobotLED = ~ch2 ;    delay(100000);
} // sub 함수를 종료하고 sub 함수를 부른 마지막 위치로 되돌아 간다.
```



- unsigned char ch1 :  
변수 ch1를 부호 없는 문자형 변수로 선언한다.
- main함수에서 출력포트에 240을 출력하면 16진수로 0xF0이므로 상위 4비트에 연결된 LED가 켜진다. 일정 시간이 지나서 15를 출력하면 0x0F가 되므로 하위 4비트에 연결된 LED만 켜진다.
- sub함수에서는 main함수에서와 같이 출력에 전역 변수 ch1의 값이 적용되는 범위를 알 수 있다.
- 실행 결과를 보면 8개의 LED중 상위 4비트의 LED와 하위 4비트의 LED가 번갈아가며 켜지고 꺼진다.

```

void main(void){

    ch1= 240;
    ch2 = 15;

    for( ; ;){
        RobotLED = ~ch1;      //main 함수의 변수 ch1값을 출력
        delay(100000);        //일정 시간 대기
        RobotLED = ~ch2;      //main 함수의 변수 ch1값을 출력
        delay(100000);        //일정 시간 대기
        sub();                //sub() 함수를 실행하고 돌아온다
    }                          //for 문의 끝-조건식을 비교한다.
}

```

### 실행결과

⇒ RobotLED에 240을 출력한 경우

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	○	●	●	●	●

⇒ RobotLED에 15(F)를 출력한 경우

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	●	●	●	○	○	○	○

⇒ 위 결과를 반복하여 실행한다.

위 실험을 통해 전역 변수의 값은 어느 함수에서나 변하지 않고 똑같이 적용됨을 알 수 있다.

### (5) 변수의 초기화

변수는 선언할 때 어떤 값을 가지고 있을까? 변수를 선언하는 것은 메모리의 어느 공간을 사용하겠다고 하는 것이지 메모리에 있는 내용을 바꾸겠다는 것이 아니다. 선언된 변수에 지정된 메모리에는 어떤 값이 있는지 알



수 없다. 그래서 변수를 선언하고 메모리에 남겨진 값을 0 또는 원하는 값을 넣어 초기화해야 한다.

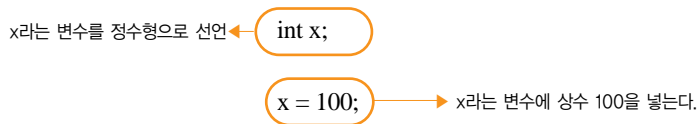
변수의 초기화 과정은 지역 변수와 전역 변수에 따라 다르게 적용된다.

- 지역 변수: 변수를 선언할 때 특정값을 넣어서 변수를 초기화한다.
- 전역 변수: 변수를 선언하면 자동으로 0으로 초기화된다.

지역 변수는 초기화하지 않으면 임의의 값이 저장되어 있다. 하지만, 지역 변수를 선언할 때 꼭 초기값을 지정해야 하는 것은 아니다. 정상적인 동작만 가능하다면 프로그램 실행 중에 초기화해도 된다. 하지만 누적 계산을 하는 프로그램에서는 꼭 초기화해야 한다.

## (6) 상수

상수는 변수와 마찬가지로 데이터를 넣는 메모리의 장소인데, 변수와는 달리 저장된 데이터 값을 바꿀 수 없다. C-언어에서 12라고 쓰면 10진수 12를 나타내는 문자 상수이다. 상수는 고정된 값이 할당되기 때문에 프로그램에서 값을 바꿀 수 없다. 12라는 문자 상수는 항상 정수 12의 값을 갖는다.



상수로 문자 상수를 사용할 수 있으나, 상수를 매크로 함수로 정의하여 기호 상수로 사용할 수 있다. 매크로는 #define 문으로 정의하며, 정의된 내용을 프로그램 중간에 변경할 수 없다.

예를 들어 12라는 문자 상수를 'Year' 라는 기호 상수로 매크로 정의하면, 프로그램을 종료할 때까지 'Year' 는 12라는 값을 갖는다.



LED를 동작시키기 위해 #define문을 사용하면 편리하게 프로그램을 작성할 수 있다.

다음은 센서 값에 따라 움직이는 방향을 결정할 때 매크로 상수를 #define문으로 선언하여 사용한 예이다.

```
예) #define LeftSensor 0x11
예) #define RightSensor 0x22
```

"만약 읽어온 값이 왼쪽 센서이면 왼쪽으로 모터를 돌려라."를 C-언어로 표현하면,

```
if( readSensor == LeftSensor )
    Motor = LeftTurn
```

왼쪽 센서의 값을 외우고 있지 않아도 이처럼 매크로 상수를 사용하면 간편하게 프로그래밍 할 수 있다.



• #define TEN 10

10의 값을 가진 문자 상수 10을 매크로 상수 TEN으로 사용하기 위해 매크로로 정의한다. 앞으로 TEN은 10의 값을 가진 매크로 상수이다.

• unsigned char sum=0, i

문자형 변수 sum과 i를 지역 변수로 선언하고 변수 sum은 0으로 초기화한다.

• RobotLED = 0xFF

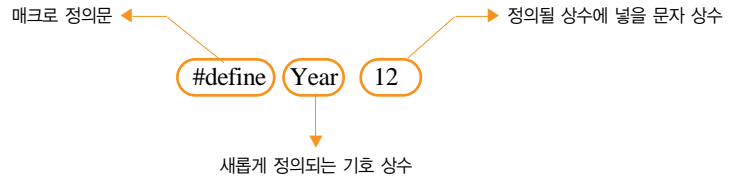
LED를 모두 끈다. 특별한 이유는 없다.

• for(i=1 ; i<=TEN ; i++)

sum = sum + i  
i를 1로 초기화하고 i가 매크로 상수 TEN이 가지고 있는 10보다 작거나 같으면 i를 1씩 증가시키면서 실행문을 반복하여 실행한다. 그 결과는 1부터 10까지 더한 수 55가 된다.

• RobotLED = ~sum

변수 sum에 저장되어 있는 결과값을 출력한다.



쉽게 설명하면 12라는 문자 상수는 일반적인 정수 12를 나타낸다. 1년은 12달로 되어 있다는 의미를 가진다면 #define 문에 의해 정의되는 Year는 12라는 수를 상징하는 기호 상수가 되며, Year에는 새로운 값을 넣을 수 없고 항상 12라는 값을 갖는다.



예제 4-7

1부터 10까지 더하는 과정의 값을 표시하는 프로그램에서 10을 TEN이라는 매크로 상수로 지정하여 프로그램을 작성하여 보자.

```
#include<GC89T51.H> // 헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED P0
#define TEN 10

void delay(double long d_t){
    double long k;
    for(k=0;k<=d_t;k++);
}

void main(void){
    unsigned char sum=0,i;

    RobotLED = 0xFF; // LED 초기화

    for(i=1;i<=TEN;i++)
    {
        sum = sum + i;
        delay(50000);
        RobotLED = ~sum;
    }
    for( ; );
}
```



### 실행결과

⇒ RobotLED에 1부터 10까지의 합인 sum을 출력한 경우

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	●	●	○	●	●	○	○

⇒ 1부터 10까지의 합은 55이고, 이것을 16진수로 나타내면 0x37이다.

[예제 4-6]에서 사용한 것처럼 매크로 상수는 프로그래밍의 과정을 편리하게 하려고 자주 사용하는 값을 상징적인 의미를 가진 단어로 표현하는 것이다. 만약 1부터 10까지의 합을 구하지 않고 1000까지의 합을 구하는 프로그램을 작성할 때는 매크로 상수로 선언되는 부분에 10대신 1000을 넣으면 된다.



.....

.....

.....

.....

.....

.....

.....

.....

.....

.....



# 단원 정리 문제

01

C 언어에서 변수를 선언할 때 지켜야 할 점이 아닌 것은?

- ① 대문자와 소문자를 서로 다르게 구분한다.
- ② 같은 이름으로 두 개 이상 선언할 수 없다.
- ③ main이라는 변수명을 사용할 수 있다.
- ④ 함수안에서 선언된 변수는 함수안에서만 사용된다.

02

다음 변수의 선언중 맞지 않은 것은?

- ① int x;
- ② int x, y;
- ③ int x; int y;
- ④ int x; y;

03

변수를 여러 개 선언할 때 무엇으로 변수를 구분할까?

- ① 세미콜론 (;)
- ② 콜론 (:)
- ③ 콤마(,)
- ④ 도트(.)

04

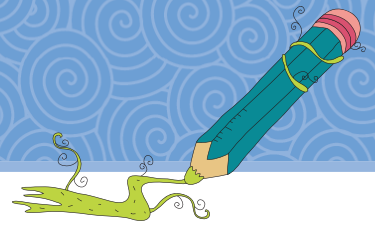
변수 x가 있을 때, 다음이 의미하는 것은?

```
X = 100 ;
```

05

데이터형 중에서 char형 데이터의 크기는?

- ① 1바이트
- ② 2바이트
- ③ 4바이트
- ④ 1비트



06

unsigned int 데이터형으로 표현되는 정수의 범위는?

07

전역 변수는 선언되면 자동으로 초기화가 된다. 어떤 값으로 초기화가 될까?

08

1부터 10까지를 순서대로 LED에 표시하시오.

09

두 개의 변수를 선언하여 각각 100과 200을 지연 시간을 두고 순서대로 LED에 표시하시오.

10

아래 그림처럼 LED가 점등되도록 프로그램 하시오.

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	●	●	○	○	●	○	●



### 3. 연산자(Operator)

연산수(Operand, 상수 또는 변수 또는 연산의 결과)로 주어지는 데이터 값들을 사용하여 사칙 연산, 대입, 비교, 논리 연산 등의 계산을 수행하는 기호들을 연산자라고 한다.

#### (1) 연산자의 종류

연산자는 수행하는 연산에 따라 산술 연산자, 대입 연산자, 증감 연산자, 비트 연산자, 관계 연산자, 논리 연산자로 분류된다.

[표 4.3-4] 연산자의 종류

연산자의 종류	
분 류	설 명
산술 연산자	수식과 수식의 사칙 연산을 수행하는 연산자 ( +, -, *, /, % )
대입 연산자	변수에 수를 대입하거나 연산의 결과를 대입하는 연산자 ( =, +=, -=, *=, /=, &= 등 )
증감 연산자	수의 값을 1씩 증가 또는 감소시키는 연산자 ( ++, -- )
비트 연산자	2진법에 의한 연산을 수행하는 연산자 ( <<, >>, &, ^,   )
관계 연산자	수와 변수의 관계를 비교하는 연산자 ( ==, <=, >=, !=, ! )
논리 연산자	AND, OR등의 논리 연산을 수행하는 연산자 ( &&,    )

연산자들을 이용하여 수식을 계산하거나 비교하는 작업을 반복하여 원하는 결과를 얻는다. 이 장에서는 마이크로로봇을 구동시키는 프로그램을 작성할 때 꼭 필요한 몇 가지 연산자를 예제로 살펴보자.

#### (2) 산술 연산자

변수의 값에 대한 사칙연산(덧셈, 뺄셈, 곱셈, 나눗셈, 나머지)을 수행하는 연산자이다.



[표 4.3-5] 산술 연산자의 종류

산술 연산자 (산술 기호)	설 명
덧셈 (+)	두 개의 변수가 가진 값에 대한 합을 구한다. $z = x + y$ ; ( 우측의 x와 y의 값을 더하여 z에 대입한다. )
뺄셈 (-)	두 개의 변수가 가진 값에 대한 차를 구한다. $z = x - y$ ; ( 우측의 x와 y의 값을 빼서 z에 대입한다. )
곱셈 (*)	두 개의 변수가 가진 값에 대한 곱을 구한다. $z = x * y$ ; ( 우측의 x와 y의 값을 곱하여 z에 대입한다. )
나눗셈 (/)	두 개의 변수를 나눈 값을 구한다. $z = x / y$ ; ( 우측의 x를 y로 나눈 결과를 z에 대입한다. )
나머지 (%)	두 개의 변수를 나눈 나머지의 값을 구한다. $z = x \% y$ ; ( 우측의 x를 y로 나눈 나머지를 z에 대입한다. )



#### 예제 4-8

x, y 두개의 변수에 대한 사칙 연산의 결과를 각각 출력하는 프로그램을 통해 산술연산자의 활용에 대해 알아보자.

```
#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void delay(long d_t){
    long k;
    for(k=0;k<=d_t;k++);
}

void main(void){
    unsigned char x,y,z;    //지역 변수로 sum을 선언하고 초기화함.

    x=10;    y=2;

    RobotLED = 0xFF;    delay(100000);
```



- `unsigned char x, y, z`  
변수 `x`, `y`, `z`를 부호 없는 문자형 변수로 선언한다.
- `delay(200000)`  
지연 시간을 처리하는 함수를 실행하여 일정 시간을 지연시킨다.
- `x = 10 ; y = 2`  
변수 `x`에는 10을 넣고, 변수 `y`에는 2를 넣는다.
- `z = x + y`  
변수 `x`와 `y`의 값을 더하여 그 결과를 변수 `z`에 넣는다.
- `RobotLED = ~z`  
산술 연산의 결과값을 반전시켜 변수 `z`의 값을 출력한다.

```

for( ; ){
    z= x + y;
    RobotLED = ~z;    delay(200000);
    z= x - y;
    RobotLED = ~z;    delay(200000);
    z= x * y;
    RobotLED = ~z;    delay(200000);
    z= x / y;
    RobotLED = ~z;    delay(200000);
    z= x % y;
    RobotLED = ~z;    delay(200000);
}
}

```

### 실행결과

⇒ 덧셈 연산자의 결과 ( $z = x + y$ 에서  $z$ 의 결과는 12)

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	●	○	○	○	○

⇒ 뺄셈 연산자의 결과 ( $z = x - y$ 에서  $z$ 의 결과는 8)

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	●	○	○	○	○

⇒ 곱셈 연산자의 결과 ( $z = x * y$ 에서  $z$ 의 결과는 20)

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	○	●	○	○	○

⇒ 나눗셈 연산자의 결과 ( $z = x / y$ 에서  $z$ 의 결과는 5)

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	○	●	○	○	○	○	○



⇒ 나머지 연산자의 결과 ( $z = x \% y$ 에서  $z$ 의 결과는 0)

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	○	○	○	○	○

각 연산의 결과값은 16진수로 표현되어 출력된다.

### (3) 대입 연산자

오른쪽의 연산수(상수 또는 변수에 저장된 값 또는 산술 연산의 결과)의 값을 왼쪽의 변수에 넣어준다. 예를 들어 'k = 12;' 라는 실행문은 오른쪽에 있는 12라는 값을 왼쪽에 있는 변수 k에 넣는것을 의미한다. 이후 변수 k는 12라는 값을 갖는다.

### (4) 증감 연산자

변수의 값을 1씩 증가시키거나 1씩 감소시키는 연산자이다.

#### ① 증가 연산자

'k++' 는 'k = k + 1' 와 같은 의미이다. k의 값을 1만큼만 증가시킬 때 사용한다.

#### ② 감소 연산자

'k--' 는 'k = k - 1' 와 같은 의미이다. k의 값을 1만큼만 감소시킬 때 사용한다.

1부터 10까지의 합을 구하는 예제를 보면, 1부터 10까지의 수를 1씩 증가시키면서 더했다. 이와같이 증감 연산자는 변수의 값을 1씩 증가 또는 감소시킬 때 사용한다. 이 연산자는 주로 제어문에서 사용한다.



## 예제 4-9

변수에 저장된 값을 1씩 증가시키다 다시 1씩 감소하도록 하는 프로그램을 통해 증가 연산자와 감소 연산자에 대해 알아보자.



• `unsigned char k = 0`

변수 `k`를 부호없는 문자형 변수로 선언하고, 초기값으로 '0'을 넣는다.

• `RobotLED = ~k`

변수 `k`에 있는 값을 반전하여 16진수로 `RobotLED`로 출력한다.

• `k++ ; k--`

변수 `k`의 값을 1 증가하거나, 1 감소한다.

```
#include<GC89T51.H>      //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void delay(long d_t){
    long k;
    for(k=0;k<=d_t;k++);
}

void main(void){
    unsigned char k=0; //지역 변수로 k를 선언하고 초기화함.

    for (; ) {
        RobotLED = ~k ;    //k가 0이므로 RobotLED에 0을 출력
        k++;              //k를 1증가하여 k는 1이 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 1이므로 RobotLED에 1을 출력
        k++;              //k를 1증가하여 k는 2가 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 2이므로 RobotLED에 2를 출력
        k++;              //k를 1증가하여 k는 3이 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 4이므로 RobotLED에 3을 출력
        k++;              //k를 1증가하여 k는 4가 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 4이므로 RobotLED에 4를 출력
        k++;              //k를 1증가하여 k는 5가 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 5이므로 RobotLED에 5를 출력
        k--;              //k를 1감소하여 k는 4가 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 4이므로 RobotLED에 4를 출력
        k--;              //k를 1감소하여 k는 3이 됩니다.
        delay(150000);
        RobotLED = ~k ;    //k가 3이므로 RobotLED에 3을 출력
        k--;              //k를 1감소하여 k는 2가 됩니다.
        delay(150000);
    }
}
```





```

RobotLED = ~k ; //k가 2이므로 RobotLED에 2를 출력
k--;           //k를 1감소하여 k는 1이 됩니다.
delay(150000);
RobotLED = ~k ; //k가 1이므로 RobotLED에 1을 출력
k--;           //k를 1감소하여 k는 0이 됩니다.
delay(150000);
LRobotLED = ~k ; //k가 0이므로 RobotLED에 0을 출력
delay(150000);
}
}

```

### 실행결과

⇒ k의 값을 출력하면 0~5까지 증가하였다가 다시 0으로 감소한다.

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	○	○	○	○	○
●	○	○	○	○	○	○	○
○	●	○	○	○	○	○	○
●	●	○	○	○	○	○	○
○	○	●	○	○	○	○	○
●	○	●	○	○	○	○	○
○	○	●	○	○	○	○	○
●	●	○	○	○	○	○	○
○	●	○	○	○	○	○	○
●	○	○	○	○	○	○	○
○	○	○	○	○	○	○	○

## (5) 비트 연산자

컴퓨터가 사용하는 정보의 기본 단위인 비트 단위로 연산하는 연산자이다.

### ① AND 연산자 ( & )

두 연산자 비트의 값이 모두 '1' 인 경우에만 결과가 '1' 이고, 하나라도 '0' 이면 결과가 '0' 이 되는 연산자이다.

[표 4.3-6] AND 연산 예

	a	b	a & b
c = a & b	0	0	0
	0	1	0
	1	0	0
	1	1	1

**ex** a = 4이고 b = 8일 때, AND연산에 의한 c의 결과를 알아보자.

a는 16진수로 표시하면 0x04이고, BCD값은 0000 0100이다.

b는 16진수로 표시하면 0x08이고, BCD값은 0000 1000이다.

a와 b를 AND 연산하면,

	0	0	0	0	0	1	0	0
&	0	0	0	0	1	0	0	0
	0	0	0	0	0	0	0	0

그 결과는 '0'이다.

AND 연산자를 좀 더 쉽게 설명하면 각각의 비트를 곱하기 한다고 생각하면 된다. 즉  $1 \times 1 = 1$ ,  $1 \times 0 = 0$ ,  $0 \times 1 = 0$ ,  $0 \times 0 = 0$ 이다.

예제를 통해 더 자세히 알아보자.



#### 예제 4-10

두 개의 변수에 저장된 값을 AND 연산자로 연산한 결과를 출력하여 AND연산자에 대해 알아보자.

```
#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void main(void){
    unsigned char x, y, z; //지역 변수로 x,y,z를 선언하고 초기화함.

    x=0x05;        y=0x0F;
```



```

for(;;){
    z=x & y;
    RobotLED = ~z ;
}
}

```

### 실행결과

⇒ z = x & y의 연산 결과를 출력하면

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	○	●	○	○	○	○	○

- unsigned char x, y, z ;  
변수 x, y, z를 부호 없는 문자형 변수로 선언한다.
- x = 0x05 ; y = 0x0F ;  
변수 x와 y에 각각 16진수의 값을 넣는다.
- z = x & y ;  
x와 y를 AND 연산하여 그 결과를 z에 넣는다.  
연산 결과를 자세하게 보면,  
x의값 0x05는 2진수로 0000 0101이다.  
y의값 0x0F는 2진수로 0000 1111이다.  
x와 y를 AND 연산하면,

	0	0	0	0	0	1	0	1
&	0	0	0	0	1	1	1	1
	0	0	0	0	0	1	0	1

위 결과를 보면, x의 값이 그대로 나온다.  
AND 연산자는 특정 비트를 마스크(MASK)시키고 싶을 때 사용한다. 이것을 이용하여 특정 비트가 1인지 0인지를 검사할 수 있으며, 아래와 같이 변수 x의 첫 번째 비트를 '0'으로 만들고 싶으면 변수 y의 첫 번째 비트만 '0'으로 하고 나머지 비트는 '1'로 하여 AND 연산을 하면, 변수 x의 첫 번째 비트가 '0'으로 바뀌어 결과를 얻게 된다.

	0	0	0	0	0	1	0	1
&	1	1	1	1	1	1	1	0
	0	0	0	0	0	1	0	0

## ② OR 연산자 ( | )

두 개의 연산수 비트의 값이 모두 '0' 인 경우에만 결과가 '0' 이고, 하나라도 '1' 이면 결과는 '1' 이 되는 연산자이다.

[표 4.3-7] OR 연산 예

	a	b	a   b
c = a   b	0	0	0
	0	1	1
	1	0	1
	1	1	1

OR 연산자는 각 비트를 더한다고 생각하면 쉽다.

즉, 1+0=1, 0+1=1, 0+0=0, 1+1=1 (1 더하기 1의 결과는 2진수에서의 더하기 이므로, '1' 이라고 생각하면 된다.)

**ex** a = 4이고 b = 8일 때, OR 연산에 의한 c의 결과를 알아보자.

a는 16진수로 표시하면 0x04이고, BCD값은 0000 0100이다.

b는 16진수로 표시하면 0x08이고, BCD값은 0000 1000이다.

a와 b를 OR 연산하면,

	0	0	0	0	0	1	0	0
	0	0	0	0	1	0	0	0
	0	0	0	0	1	1	0	0

그 결과는 16진수로는 0x0C이고 10진수로는 12가 된다.

AND 연산자는 특정 비트의 값이 1인지 0인지를 검사하거나, 특정 비트를 0으로 만들 때 사용하며, OR 연산자는 주로 특정 비트를 1로 만들 때 사용한다.

## ③ XOR 연산자 ( ^ )

이 연산자는 배타적 OR 연산자라고도 부르며, 각 연산수 비트의 값이 서로



로 다른 경우에는 결과가 1이고, 서로 같은 경우에는 그 결과가 0이다.

[표 4.3-8] XOR 연산 예

	a	b	a ^ b
c = a ^ b	0	0	0
	0	1	1
	1	0	1
	1	1	0

**ex** a = 0xA5이고 b = 0x55일 때, XOR연산에 의한 c의 결과를 알아보자.

a는 16진수로 표시하면 0xA5이고, BCD값은 1010 0101이다.

b는 16진수로 표시하면 0x55이고, BCD값은 0101 0101이다.

a와 b를 XOR 연산하면,

^	1	0	1	0	0	1	0	1
	0	1	0	1	0	1	0	1
	1	1	1	1	0	0	0	0

그 결과는 16진수로는 0xF0이다.



#### 예제 4-11

두 개의 변수에 저장된 값을 XOR 연산자로 연산한 결과를 출력하여 XOR 연산자에 대해 알아보자.

```
#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void main(void){
    unsigned char x, y, z; //지역 변수로 x,y,z를 선언하고 초기화함.

    x=0x05;  y=0xA0;
    z=x^y;
```

```

for(;;){
    RobotLED = ~z;
}
}

```

### 실행결과

⇒  $z = x \wedge y$ 의 연산 결과를 출력하면

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	○	●	○	○	●	○	●

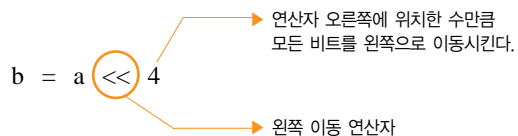
- unsigned char x, y, z ;  
변수 x, y, z를 부호없는 문자형 변수로 선언한다.
- x = 0x05 ; y = 0xA0 ;  
변수 x와 y에 각각 16진수의 값을 넣는다.
- z = x ^ y ;  
x와 y를 XOR 연산하여 결과를 z에 넣는다. 연산 결과를 보면, x의 값은 0x05인데 2진수로 0000 0101이다.  
y의 값은 0xA0인데 2진수로 1010 0000이다. x와 y를 XOR 연산하면,

	0	0	0	0	0	1	0	1
^	1	0	1	0	0	0	0	0
	1	0	1	0	0	1	0	1

그 결과는 0xA5이다.

### ④ 왼쪽 이동 연산자 ( << )

이 연산자는 연산자 오른쪽에 위치하는 수만큼 왼쪽으로 연산수 전체를 이동시키는 연산자이다.

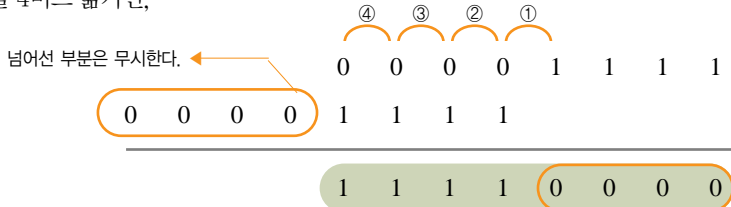




**ex** a가 0x0F일 때, 왼쪽으로 4비트 옮긴 결과를 알아보자.

a를 BCD로 표현하면, 0000 1111이다.

이것을 4비트 옮기면,



그 결과는 0xF0이다.

우측의 변수의 값을 왼쪽으로 x번 이동한 시키는 것은 우측에 있는 변수의 값에 2를 x번 곱하는 것이다.



#### 예제 4-12

변수에 저장된 값의 각 비트를 왼쪽으로 일정 비트를 옮기는 왼쪽 이동 연산자에 대해 알아보자.

```
#include <GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void main(void){
    unsigned char x,y; //지역 변수로 x,y를 선언하고 초기화함.

    x=0x01;
    y=x<<3;    // 좌측 이동 연산자 (좌측으로 3 비트)

    for(;;){
        RobotLED = ~y ;
    }
}
```



### 실행결과

⇒  $y = x \ll 3$  의 연산 결과를 출력하면

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	●	○	○	○	○

○ unsigned char x, y ;

변수 x, y를 부호 없는 문자형 변수로 선언한다.

○  $y = x \ll 3$  ;

x의 값을 왼쪽으로 3비트 옮겨 y에 넣는다.

x의 값 0x01은 BCD값으로 0000 0001이고, 이것을 왼쪽으로 3비트 옮겨 보면,

넘어진 부분은 무시한다. ←

0 0 0 0 0 0 0 1  
 0 0 0 0 0 0 0 1

0 0 0 0 1 0 0 0

없어진 부분은 '0'으로 채워진다. ←

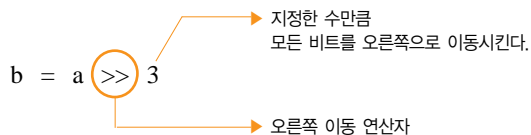
그 결과는 0x08이다.

옮겨진 비트 수만큼 2를 곱하면  $2 * 2 * 2 = 8$ 이 되며, 위 결과는 x의 값과 2를 옮겨진 비트 수만큼 곱한 것과 같다. y를 수식으로 연산하면,

$$y = x * 2^n \text{ (이때 } n \text{은 옮겨진 비트 수)}$$

### ⑤ 오른쪽 이동 연산자 ( >> )

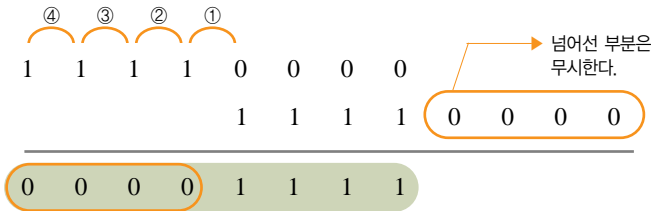
오른쪽 이동 연산자는 왼쪽 이동 연산자와는 반대로 연산자 오른쪽에 위치하는 수만큼 오른쪽으로 전체를 이동시키는 연산자이다.





**ex** a가 0xF0일 때, 오른쪽으로 4비트 옮긴 결과를 알아본다.

a를 BCD로 표현하면, 1111 0000이다. 이것을 4비트 옮기면,



그 결과는 0x0F이다.

### ⑥ 관계 연산자

두 개 연산 수의 크기에 대한 관계를 비교해서 참 또는 거짓을 판정하는 연산자이다. 이 연산자는 비교 제어문, 반복 제어문, 선택 제어문 등 프로그램의 흐름을 제어할 때 사용한다.

[표 4.3-9] 관계 연산자의 종류

연산자	의미	표현	설명
>	크다	$a > b$	a는 b보다 크다.
<	작다	$a < b$	a는 b보다 작다.
>=	크거나 같다	$a >= b$	a는 b보다 크거나 같다.
<=	작거나 같다	$a <= b$	a는 b보다 작거나 같다.
==	같다	$a == b$	a와 b는 같다.
!=	같지 않다	$a != b$	a와 b는 같지 않다.

관계 연산자를 예제를 통해 알아보자.



- unsigned char x, y ;  
변수 x, y를 부호 없는 문자형 변수로 선언한다.
- if(x > y) RobotLED = 0xFE ;  
else RobotLED = 0xFD ;  
만약 x가 y보다 크면 0xFE를 출력하고, 아니면 0xFD를 출력한다.  
다음 절의 제어문에서 배우게 될 if...else를 이용하여 x와 y의 값을 비교하여 그 결과에 따라 각각의 다른 결과를 출력한다.  
여기서는 변수 x의 값이 변수 y의 값보다 작으므로 0xFD의 값을 출력한다.



### 예제 4-13

두 개의 변수에 저장된 값을 비교 연산자로 비교하고, 결과가 참일 때와 거짓일 때 각각의 실행 결과가 달리 나타나는 것을 확인하자.

```
#include<GC89T51.H> //헤더 파일을 포함시킨다.
//매크로 상수 RobotLED의 선언
#define RobotLED P0

void main(void){
    unsigned char x,y; //지역 변수로 x,y를 선언하고 초기화함.

    x=0x08;    y=0x10;

    for(;;){
        if(x>y) RobotLED = 0xFE;
        else RobotLED = 0xFD;
    }
}
```

### 실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	●	○	○	○	○	○	○

### ⑦ 논리 연산자

논리 연산자는 참 또는 거짓의 논리 값을 갖는 두 개의 연산 수들을 이용하여 참이나 거짓을 판정하는 논리적인 연산을 수행한다.

보통 관계 연산자와 함께 프로그램의 진행 방향을 결정하거나 수식의 결과를 비교할 때 많이 사용한다.

#### ○ AND ( && )

두 개의 연산 수 중에서 모두 참인 경우에 참을 출력하고, 하나라도 거짓이면 거짓을 출력한다.

ex) 'x는 10보다 크고 100보다 작거나 같다'를 수식으로 표현하면 '10 < x <= 100'이다. 이것을 AND 논리 연산자로 표현하면, '( 10 < x ) && ( x <= 100 )'이다.



### ○ OR ( || )

두 개의 연산 수 중에서 하나라도 참인 경우에 참을 출력하고, 모두 거짓이면 거짓을 출력한다.

ex) 'x는 100보다 크거나 10보다 작다' 를 수식으로 표현하면, 'x < 10 , x > 100' 이다. 이것을 OR 논리 연산자로 표현하면, '( x < 10) || ( x > 100)' 이다.



### 예제 4-14

두 개의 비교식의 결과를 논리연산자로 논리를 판정하여 결과가 참일 때와 거짓일 때 각각 실행 결과가 다른 것을 확인하자.

```
#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void main(void){
    unsigned char x ; //지역 변수로 x를 선언하고 초기화함.

    x=10;

    for(;;){
        if(x>0 && x<=10)  RobotLED = 0xFE;
        else                RobotLED = 0xFD;
    }
}
```



- if( x > 0 && x <= 10 )  
 RobotLED = 0xFE ;  
 else  
 RobotLED = 0xFD ;  
 만약 x가 0보다 크고 10보다 작거나 같으면 0xFE를 출력하고, 그렇지 않으면 0xFD를 출력한다. x가 10이므로 조건을 만족하여 0xFE를 출력한다.

### 실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	○	○	○	○	○	○	○

### ⑧ 연산자의 우선 순위

컴퓨터에서는 여러 가지 일이 동시에 일어날 것을 대비하여 할 일의 순서를 정해 놓았다. 연산자들을 이용해서 수학 문제를 풀 때, 컴퓨터는 괄호와 더하기와 빼기 등의 연산들을 정해진 순서를 따라서 실행한다. 컴퓨터에서 정한 연산의 우선 순위는 다음의 표와 같다.

[표 4.3-10] 연산자 우선 순위

우선 순위	연산 종류	연산자
↑ 높다	괄호, 배열	{ }, [ ]
	단항 연산자	+, -, ++, --, 캐스트 연산자, 포인트 연산자( * ), 간접 연산자, sizeof
	곱셈, 나눗셈, 나머지	*, /, %
	덧셈, 뺄셈	+, -
	이동	<<, >>
	비교	<, >, <=, >=, ==, !=
	비트 논리곱	&
	비트 배타적 논리합	^
	비트 논리합	
	논리곱	&&
↓ 낮다	논리합	
	조건	?:
	대입	=, +=, -=, *=, /=, %=, &=, ^=,  =, <<=, >>=

수식에 여러 개의 연산자가 포함되어 있으면 우선 순위가 높은 연산부터 실행한다.



메모

.....

.....

.....

.....

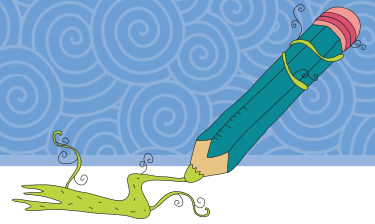
.....

.....

.....

.....

# 단원 정리 문제



01

다음 중 비트 연산자가 아닌 것은?

- ①  $\gg$                       ②  $\ll$                       ③  $\&\&$                       ④  $|$

02

다음 연산의 결과가 맞는 것은?

```
a = 10;
c = a << 1;
```

- ① 20                              ② 11  
③ 40                              ④ 21

03

서로 다른 변수의 값을 비교하는 연산자는?

- ① 산술 연산자              ② 대입 연산자              ③ 증감 연산자              ④ 비교 연산자

04

다음에서 나타내는 연산식과 관련이 없는 것은?

```
int i = 10;
i++;
```

- ①  $i = i + 1;$                       ②  $||$   
③ 증감 연산자                      ④  $i -= 1;$

05

다음 프로그램을 실행했을 때 변수 b에 저장되는 값은?

```
main()
{
    int a = 0x11, b;

    b = a | 0xE0;

}
```

- ① 0xE0  
② 0xF1;  
③ 0x00  
④ 0xFF

# 단원 정리 문제

06

다음 프로그램을 주어진 결과가 나오도록 완성하자.

```
main()
{
    int x = 0xAA, y;

    y = x ( ① ) 0x44 ;
    RobotLED = ~y;

    for(;;);
}
```

실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	●	●	●	○	●	●	○

07

다음 연산자 중에서 우선 순위가 가장 높은 것은?

- ① %                      ② \*                      ③ <<                      ④ &

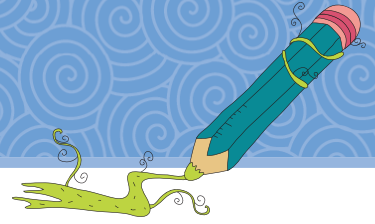
08

마이크로로봇의 LED를 0번부터 7번까지 순서대로 켜지도록하는 프로그램을 작성하자.

09

LED의 상위 4비트와 하위 4비트를 번갈아 켜지는 과정이 반복되는 프로그램을 작성하자.





10

8개의 LED 중에서 상위 4비트에서만 순서대로 켜지도록하는 프로그램을 작성하자.

LED7만 ON → LED6만 ON → LED5만 ON → LED4만 ON

11

문제 10번을 응용하여 8개의 LED중에서 하위 4비트에서만 순서대로 켜지도록하는 프로그램을 작성하자.

12

문제 10번과 11번을 응용하여 상위 4비트는 LED7부터 아래로 켜지게 하고, 하위 4비트는 LED0부터 켜지도록 하면서 계속 반복되는 프로그램을 작성하자.

LED7 ON → LED6 ON → LED5 ON → LED4 ON  
LED0 ON → LED1 ON → LED2 ON → LED3 ON

13

홀수 비트의 LED를 켜고 일정 시간이 지난 다음 짝수 비트의 LED가 켜지도록하는 프로그램을 작성하자.



## 4. 제어문

1부터 10까지 합을 구하는 프로그램을 작성할 때 제어문을 사용하지 않으면 더하기 실행문을 10번 써야 한다. 또는 특정 조건이 맞는 경우에만 실행시켜야 할 경우도 있다. 반복되는 과정을 간소하게, 특정 조건에 따라 동작하게 하는 것이 제어문이다.

### (1) 제어문의 종류

제어문은 조건에 따라 프로그램의 흐름을 제어하는 문장이며, 함수와 같이 중괄호 ({} )를 사용하여 제어문의 시작과 끝을 나타낸다.

제어문에는 조건을 비교하여 프로그램의 흐름을 제어하는 비교 제어문, 조건을 주어 작업을 반복시키는 반복 제어문, 조건에 맞는 경우에 해당 작업을 수행하는 선택 제어문으로 분류된다.

[표 4.3-11] 제어문의 종류

제어문의 형식	제어문의 종류
if... else	비교 제어문
for(초기값;조건식;증감식 )	반복 제어문
do ~ while(조건)	
while(조건)	
switch(조건)... case	선택 제어문

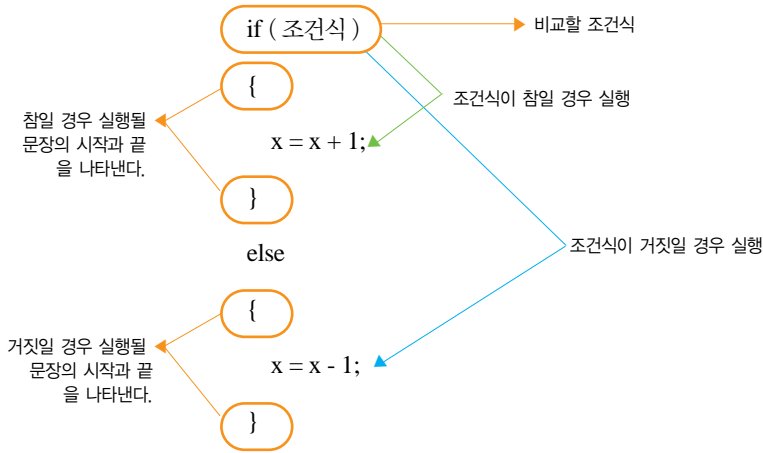
### (2) 비교 제어문

비교 제어문의 문법은 “만약 ...이면 ...이고, 아니면 ...이다.”이다. 예를 들면, “만약 목적지가 서울이면 오른쪽으로 가고, 아니면 왼쪽으로 가십시오.”와 같은 형태이다.

비교 제어문은 주어진 조건식이 참인 경우와 거짓인 경우 각각 다른 실행문을 수행한다. 비교 제어문을 if... else 제어문이라고도 한다.



## 1) 비교 제어문( if... else... )의 기본 구조



[그림 4.3-3] 비교 제어문의 기본 구조

앞의 그림처럼 괄호안의 조건식이 참이면 'x = x + 1;' 을 실행하고, 조건식이 거짓이면 'x = x - 1;' 을 실행한다.

비교 제어문은 조건식에서 대부분 비교 연산자를 사용한다.



### 예제 4-15

변수에 저장된 값을 비교하여 결과가 참일 때와 거짓일 때 비교제어문의 실행 결과가 다르게 나오는 것을 확인하자.

```

#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED      P0

void main(void){
    unsigned char x; //지역 변수로 x를 선언하고 초기화함.

    x=100;

```



- `if ( x > 100 ) { RobotLED = 0xFE ; }`  
`else { RobotLED = 0xFD ; }`  
 만약 x가 100보다 크면 0xFE를 출력하고, 그렇지 않으면 0xFD를 출력한다. 변수 x의 값이 100으로 조건을 만족하지 못하므로 0xFD를 출력한다.
- 위와 같이 실행문이 하나만 있는 경우에는 중괄호를 사용하지 않아도 된다.  
`if ( x > 100 ) RobotLED = 0xFE ;`  
`else RobotLED = 0xFD ;`

```
for(;;){
    if(x>100)    { RobotLED = 0xFE; }
    else        { RobotLED = 0xFD; }
}
}
```

### 실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	●	○	○	○	○	○	○

### (3) 다중, 이중 비교 제어문

비교하고 싶은 조건이 여러 가지일 때는 어떻게 할까? 또는 한번 비교하고 다시 비교 하려면 어떻게 할까? 다중 비교 제어문은 비교할 조건을 여러 개 두고 첫 번째 조건이 거짓이면 다른 조건을 비교하는 방식으로, 여러 개의 조건을 차례로 비교하여 결과에 따라 각각의 실행문을 실행한다.

```
○ if ( x > 300 )      RobotLED = 0x01;
  else if ( x >200 )  RobotLED = 0x02;
  else if ( x >100 )  RobotLED = 0x03;
  else                RobotLED = 0x04;
```

위 문장은 x의 값을 각각 다른 경우로 세 번 비교한다.

- ① 만약 x가 300보다 크면 0x01을 출력하고,
- ② 아니면 x가 200보다 크면 0x02를 출력하고,
- ③ 아니면 x가 100보다 크면 0x03을 출력하고,
- ④ 아무것도 해당이 되지 않으면 0x04를 출력하고 if문을 마친다.



### 예제 4-16

두개의 변수에 저장된 값을 비교 연산자로 비교하고, 참일 때와 거짓일 때 각각의 실행 결과가 다른 것을 확인하자.

```
#include<GC89T51.H> //헤더 파일을 포함시킨다.
```



```
//매크로 상수 RobotLED의 선언
#define RobotLED    P2

void main(void){
    unsigned char x; //지역 변수로 x를 선언하고 초기화함.

    x=120;

    for(;;){
        if(x>200)    RobotLED = 0xFE; //200보다 크면
        else if(x<100) RobotLED = 0xFD; //100보다 작으면
        else if(x==100) RobotLED = 0xFC; //100이면
        else        RobotLED = 0xFB; //아무 조건도 만족하지 않으면
    }
}
```

### 실행결과

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	○	○	○	○	○

이중 비교 제어문은 첫 번째 조건이 참일 때, 실행문을 실행하는 중에 다시 비교를 하는 방식이다.

```
if( x > 100 ){
    if( x > 200 ) RobotLED = 0x01;
    else        RobotLED = 0x02;
}
else {
    if( x < 50 )  RobotLED = 0x03;
    else        RobotLED = 0x04;
}
```

위 문장은 x의 값을 먼저 비교한 뒤, 실행문에서 다시 비교한다.

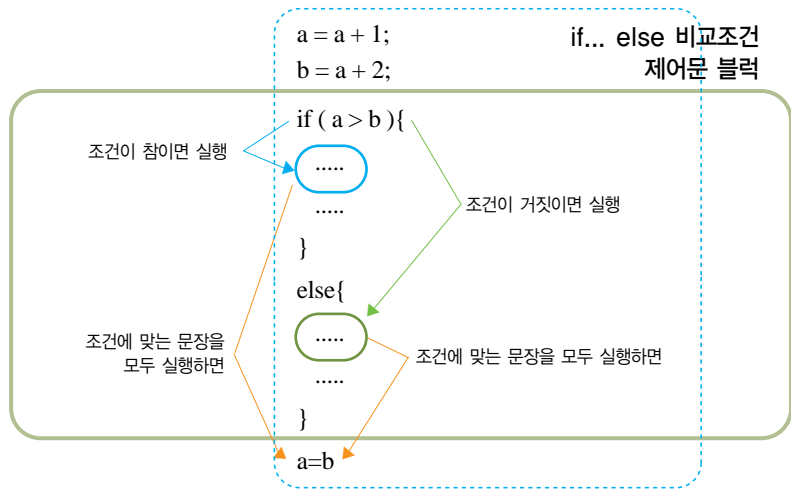
- ① 만약 x가 100보다 크면 ①'의 문장을 실행한다.
  - ①' 이때 만약 x가 200보다 크면 0x01을 출력한다.
  - 아니면 0x02를 출력한다.
- ② ①이 아니면 ②' 문장을 실행한다.
  - ②' 이때 만약 x가 50보다 작으면 0x03을 출력하고,
  - 아니면 0x04를 출력한다.



- 다중 if문을 이용하여 변수 x의 값에 대한 크기를 알아본 예제이다. x가 200보다 크면 0xFE를 출력하고, 아니면 다시 비교하여 100보다 작으면 0xFD를 출력하고, 그것도 아니면 다시 비교하여 100이면 0xFC를 출력하고, 그것도 아니면 0xFB를 출력한다.
- 결과는 x의 값이 120이므로 어느 조건에도 해당되지 않으므로 0xFB를 출력한다.

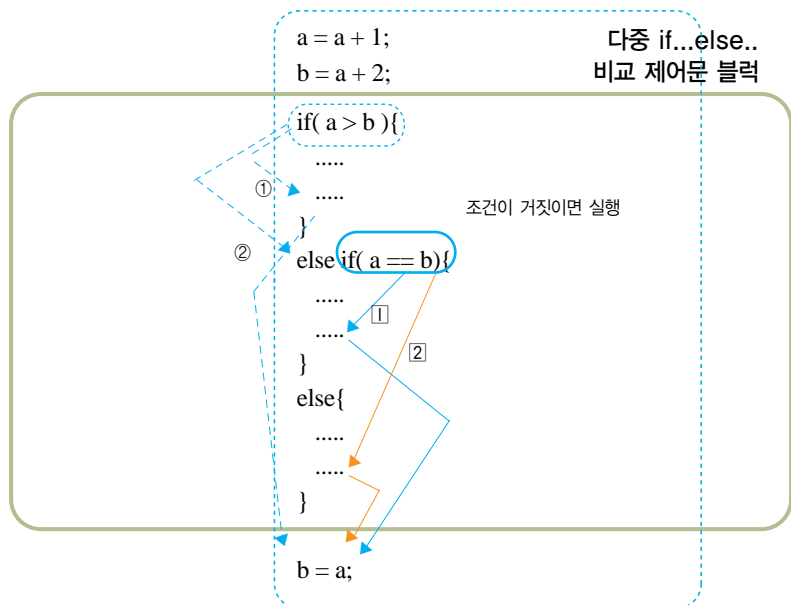
위의 예제와 같이 여러 개의 조건식이 있을 때 하나의 조건식이 참이 되어 해당 실행문을 수행하면 프로그램의 흐름은 어디로 이어질까? 조건 제어문에서 조건식에 해당하는 경우가 있을 경우에는 해당되는 중괄호 안의 내용을 실행한 다음 처음 비교한 조건식의 마지막을 지나 다음 문장을 실행한다. 이것을 그림으로 보면,

○ if... else 문장에서의 프로그램 흐름



[그림 4.3-4] if... else 문장에서의 프로그램 흐름

○ 다중 if... else 문장에서의 프로그램 흐름



[그림 4.3-5] 다중 if... else 문장에서의 프로그램 흐름



첫 번째 조건식이 참이면 ①의 파란 점선 방향으로 프로그램을 실행하고,  
거짓이면 ②의 파란 점선 방향으로 실행되어 두 번째  
조건식에서 조건을 비교하며,  
두 번째 조건식이 참이면 ③의 파란 실선 방향으로 프로그램을 실행하고,  
거짓이면 ④의 빨간 실선 방향으로 실행한다.

비교 제어문은 마이크로 로봇을 구동시키는 프로그램에 많이 사용한다.  
비교제어문을 사용하면 여러 개의 센서 중에서 각 센서마다 동작하는 방법  
을 다르게 하여 미로에서 길을 찾을 수 있다. 비교 제어문은 다음 절에서 배  
울 선택 제어문과 비슷한 기능을 수행한다.

#### (4) 반복 제어문

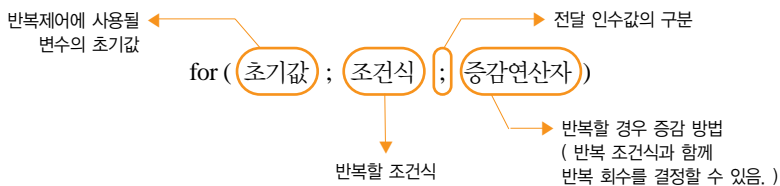
반복 제어문은 조건식이 참이면 거짓이 될 때까지 중괄호 안의 실행문을  
반복해서 실행한다. 만약 조건식이 영원히 거짓이 되지 않으면 프로그램은  
중괄호 안의 실행문을 무한 반복하여 수행해서 다른 실행문을 수행하지 않  
는데 이것을 무한 루프라고 한다.

마이크로프로세서를 사용하는 마이크로로봇의 프로그램은 전원 끄기 전  
까지 프로그램이 종료되지 않도록 반드시 무한 반복문을 사용하여야 한다.

반복 제어문에는 세가지 유형이 있다.

##### 1) for 반복 제어문

초기값으로 시작해서 조건식이 거짓이 될 때까지 증감 연산자에 의해 괄  
호( { } )안의 실행문을 반복하여 실행한다.



[그림 4.3-6] for 반복 제어문의 기본 형식

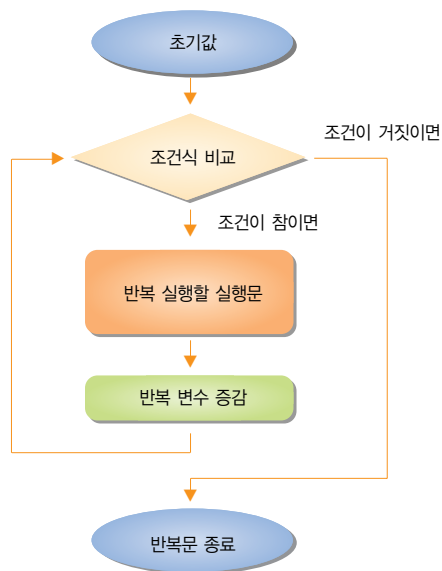


○ 초기값 : 어디서부터 반복제어에 사용할 변수의 초기값을 지정한다

○ 조건식 : 어디까지 반복할 조건을 비교하여 참이면 실행한다.

○ 증감연산자 : 어떻게 반복할 때 주어진 변수를 어떻게 증감할지 지정한다.

다시 설명하면 초기값으로 시작해서 증감 연산자로 주어진 변수를 증감하면서 조건식을 점검하면서 반복 작업을 수행한다. 조건 제어문과 같이 반복할 실행 문장들은 중괄호 ( )를 이용하여 블록으로 지정한다.



[그림 4.3-7] for 반복 제어문 순서도



#### 예제 4-17

for 반복 제어문을 이용하여 1부터 10까지의 수를 순서대로 LED에 출력하는 프로그램을 작성하여 보자.

```
#include<GC89T51.H> //헤더 파일을 포함시킨다.
```

```
//매크로 상수 RobotLED의 선언
```

```
#define RobotLED P0
```

```
void delay(long d_t){  
    long k;  
    for(k=0;k<=d_t;k++);  
}
```



```

}

void main(void){
unsigned char x;    //지역 변수로 x를 선언하고 초기화함.

for(;;){
for(x=1;x<=10; x++){
RobotLED = ~x;
delay(150000); //LED의 출력상태를 눈으로 확인하기 위해 delay시킴
}
}
}

```

### 실행결과

⇒ x = 1 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	○	○	○	○	○	○	○

⇒ x = 2 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	●	○	○	○	○	○	○

⇒ x = 3 일때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	●	○	○	○	○	○	○

⇒ x = 4 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	○	○	○	○	○

⇒ x = 5 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	●	○	○	○	○	○

⇒ x = 6 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	●	●	○	○	○	○	○

⇒ x = 7 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	●	●	○	○	○	○	○

⇒ x = 8 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	○	○	●	○	○	○	○

⇒ x = 9 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
●	○	○	●	○	○	○	○

⇒ x = 10 일 때 RobotLED 출력 상태

LED0	LED1	LED2	LED3	LED4	LED5	LED6	LED7
○	●	○	●	○	○	○	○



- `for ( x=1 ; x(<=10 ; x++) { ... }`  
변수 x를 1부터 1씩 증가시키면서 x가 10보다 작거나 같으면 블록의 실행문들을 반복 실행한다.
- `RobotLED = ~x`  
변수 x의 값을 반전하여 LED에 출력한다.

### 특징

① 초기값이나 증감 연산자를 여러 개 둘 수 있다.

○ `for ( i=0, j=0; i < 10; i++, j++)`



i와 j를 함께 초기화하고, i와 j를 증감 연산자를 이용하여 1씩 증가시키면서 블록의 실행문을 실행할 수 있다.

② 초기값이나 증감 연산자, 조건식은 생략할 수 있다.

○ for ( i=0 ; ; i++)

조건식을 생략하면 조건 검사에 결과가 항상 참이므로 무한 루프가 된다.

○ for ( ; i<10 ; i++)

변수 i의 초기값을 미리 정했거나 지정할 필요가 없으면 생략할 수 있다. 단, 조건식을 검사하여 참이면 반복문을 실행한다.

○ for ( i =0 ; i < 10 ; )

증감식을 생략하면 반복 실행 블록 안에서 증감시켜야 한다.

그렇지 않으면 조건식이 항상 참이므로 무한 루프가 된다.

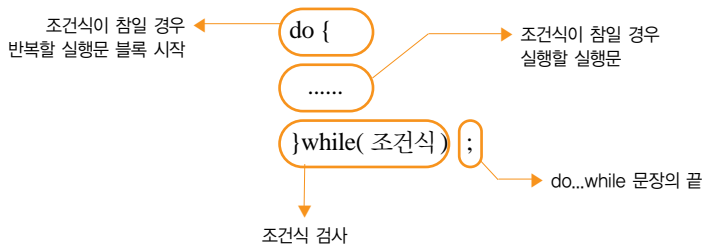
③ 초기값이나 증감 연산자, 조건식 중 하나를 생략하여 무한 루프를 만든다.

○ 초기값이나 증감 연산자, 조건식 중 하나를 생략하여 조건식이 항상 참이면, 중괄호의 블록을 무한 반복 수행한다.

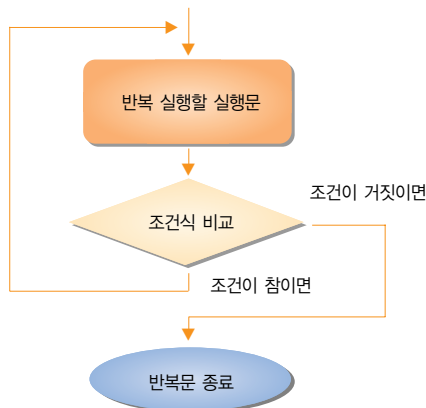
○ for ( ; ; )

초기값, 증감 연산자, 조건식 모두 생략하면 무한 루프가 된다.

## 2) do... while 반복제어문



[그림 4.3-8] do...while 반복 제어문의 기본 형식



[그림 4.3-9] do... while 반복 제어문 순서도



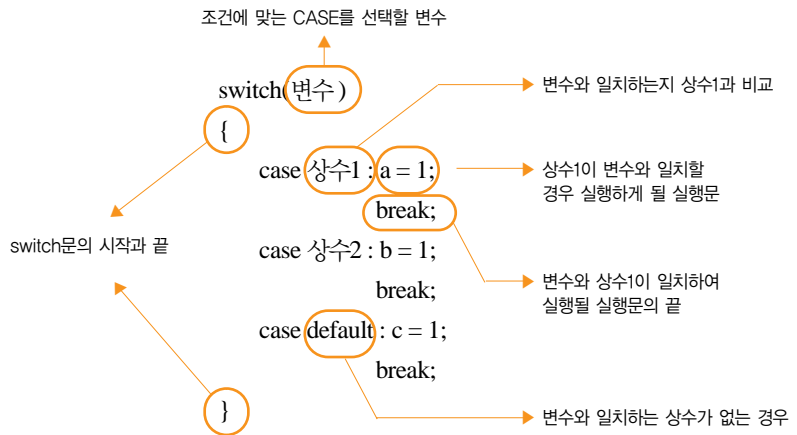
#### 무한 루프 만들기

- do... while 문에서 무한 루프는 for 반복 제어문과 같이 조건식이 항상 참이면 구현된다.
- ```
do {
    .....
} while( 1 )
```

do~while문은 먼저 실행을 하고 나서 조건식을 비교하므로 do {...} 의 실행문들을 꼭 한번은 수행한다.

### (5) 선택 제어문

선택 제어문은 비교 제어문 중에서 다중 비교 제어문과 비슷하게 작용한다. 다중 비교 제어문은 조건 변수가 여러 개의 경우로 주어진 상수 값과 비교해서 일치하는 경우의 해당 실행문을 실행하는데, 선택 제어문은 비교해서 수행할 대상을 선택하므로 더 효율적인 프로그램이 구현된다.



[그림 4.3-10] switch 선택 제어문의 기본 형식

- ① 변수 값과 상수 값과 일치하면 해당 실행문을 수행한다.
- ② 상수1, 상수2는 항상 상수 또는 매크로 상수 또는 문자 상수로 사용한다.
- ③ 상수1에서 수행할 실행문과 상수2에서 수행할 실행문을 구분을 위해 각각 case문의 끝에는 break; 문을 넣는다.
- ④ default는 변수의 값이 일치하는 상수가 없을 경우를 나타낸다. default문은 생략될 수 있다.
- ⑤ case문과 실행문의 구분은 콜론(:)을 사용한다.



### 예제 4-18

switch... case 선택 제어문을 이용하는 예제로  
if... else 조건 비교 제어문과의 차이를 알아보자.

```
#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void main(void){
    unsigned char x; //지역 변수로 x를 선언하고 초기화함.

    x=0x01;

    switch(x){
        case 0x01:  RobotLED = 0xEF; break;
        case 0x02:  RobotLED = 0xDF; break;
        case 0x04:  RobotLED = 0xBF; break;
        default:    RobotLED = 0x7F; break;
    }
    for(;;);
}
```

### 실행결과

⇒ 변수 x의 값이 0x01이므로 0xEF 출력하게 된다.

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ●    | ○    | ○    | ○    |

```
○ switch( x ) {
    case 0x01 : RobotLED = 0xEF    break ;
    ...
}
```

괄호 안의 변수 x와 값이 일치하는 case 문의 실행문들을 수행한다.  
이때, break 문을 만날 때 까지 모든 실행문을 수행한다.



#### 예제 4-19

switch... case 선택 제어문에서 선택된 case문에 break 실행문이 없을 때 break 실행문을 만날때 까지 수행되는 것을 예제로 확인하자.

```
#include<GC89T51.H>      //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

void main(void){
    unsigned char x=0x01; //지역 변수로 x를 선언하고 초기화함.

    switch(x){
        case 0x01:RobotLED = 0xEF;
        case 0x02:RobotLED = 0xDF; break;
        case 0x04:RobotLED = 0xBF; break;
        default: RobotLED = 0x7F; break;
    }
    for(;;);
}
```

#### 실행결과

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ●    | ○    | ○    |

```
○ switch( x ) {
    case 0x01 : RobotLED = 0xEF; break ;
    ...
}
```

괄호 안의 변수 x와 값이 일치하는 case 문의 실행문들을 수행한다. 위 예제에서는 case 0x01 : ... 문장을 수행하지만, break 문이 없기 때문에 아래 실행문까지 수행한 후 break 문을 만나기 때문에 결과는 0xDF가 된다.

[예제 4-19]의 switch... case 문장을 if...else문으로 바꾸면,

```
if( i = 0x01 )           RobotLED = 0xEF;
else if( i = 0x02 )     RobotLED = 0xDF;
else if( i = 0x04 )     RobotLED = 0xBF;
else                    RobotLED = 0x7F;
```





# 단원 정리 문제

01

제어문을 사용할 때 제어문의 시작과 끝은 무엇으로 나타내는가?

- ① ( 와 )      ② { 와 }      ③ [ 와 ]      ④ < 와 >

02

입력된 두 개의 수 중 어느 것이 더 큰지를 알려주는 프로그램을 작성할 때 사용하는 제어문은?

- ① 비교 제어문      ② 반복 제어문      ③ 선택 제어문      ④ 흐름 제어문

03

위 2번의 문제를 수행하는 프로그램을 작성하자.

04

for 반복 제어문에서 초기값, 조건식, 증감식을 구분하는 문자는?

- ① 세미콜론 (;)      ② 콜론 (: )      ③ 콤마 (,)      ④ 도트 (.)

05

선택 제어문에서 선택된 case 실행문들의 끝 문장은?

- ① continue      ② default      ③ break      ④ stop

06

1부터 10까지의 합을 구하는 프로그램을 for반복 제어문을 이용하여 작성하여 결과를 LED로 표시하여 보자.

07

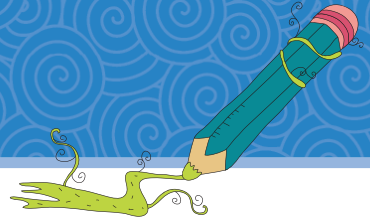
위 6번 문제를 do~while 반복 제어문을 이용하여 구하는 프로그램을 작성하여 보자.

08

모든 LED를 5회 켜고 끄는 프로그램을 for문장을 이용해 작성하여 보자.

09

16진수 0x30의 각 비트 중 '1'인 비트의 개수를 LED로 나타내어 보자.



10

다음 표와 같이 순서대로 LED를 점등하는 프로그램을 작성하여 보자. 단, for 반복문과 do~while 반복문을 최소한 1회이상은 사용해야 한다.

| 실행순서 | LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|------|
| 1    | ●    | ●    | ●    | ●    | ○    | ○    | ○    | ○    |
| 2    | ○    | ○    | ○    | ○    | ●    | ●    | ●    | ●    |
| 3    | ●    | ○    | ●    | ○    | ●    | ○    | ●    | ○    |
| 4    | ○    | ●    | ○    | ●    | ○    | ●    | ○    | ●    |
| 5    | ○    | ○    | ○    | ○    | ○    | ○    | ●    | ●    |
| 6    | ○    | ○    | ○    | ○    | ●    | ●    | ○    | ○    |
| 7    | ○    | ○    | ●    | ●    | ○    | ○    | ○    | ○    |
| 8    | ●    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |
| 9    | ○    | ○    | ●    | ●    | ○    | ○    | ○    | ○    |
| 10   | ○    | ○    | ○    | ○    | ●    | ●    | ○    | ○    |
| 11   | ○    | ○    | ○    | ○    | ○    | ○    | ●    | ●    |
| 12   | ●    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |
| 13   | ○    | ●    | ○    | ○    | ○    | ○    | ●    | ○    |
| 14   | ○    | ○    | ●    | ○    | ○    | ●    | ○    | ○    |
| 15   | ○    | ○    | ○    | ●    | ●    | ○    | ○    | ○    |
| 16   | ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |
| 17   | ○    | ○    | ○    | ●    | ●    | ○    | ○    | ○    |
| 18   | ○    | ○    | ●    | ●    | ●    | ●    | ○    | ○    |
| 19   | ○    | ●    | ●    | ●    | ●    | ●    | ●    | ○    |
| 20   | ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |
| 21   | ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |
| 22   | ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |
| 23   | ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

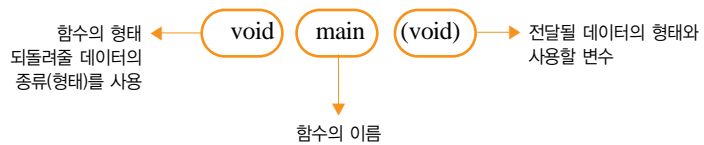
11

for문장을 이용하여 1부터 255까지 순서대로 LED에 표시하자.

## 5. 함수와 배열

함수는 간단히 "특정한 기능을 수행하는 실행문들의 집합체"라고 할 수 있다. 그래서 함수의 시작과 끝을 중괄호를 이용해서 표시한다. 또한, 함수의 형태를 선언해서 함수를 실행한 결과를 받을 수 있고, 특정 값을 함수에 전달할 수도 있다. 특정 기능을 함수로 묶어 사용하면 프로그램에 중복되는 기능들을 다시 작성하지 않고 함수를 호출하여 사용할 수 있다.

### (1) 함수의 기본 형태



[그림 4.3-11] 함수의 기본 형태

- ① **void** : 함수의 형태인데 함수의 실행 결과를 되돌려 주는 데이터의 형태이다. 결과를 되돌려 받지 않으려면 함수의 형태를 void로 지정한다.
- ② **main** : 함수의 이름을 나타낸다. 새로 함수를 작성하려면 함수의 이름을 정해야 한다. 이미 사용한 함수 이름은 다시 사용할 수 없다. main은 C 언어에서는 항상 있어야 하는 주함수이다. 프로그램의 실행은 주함수로부터 시작한다.
- ③ **(void)** : 함수에 전달할 데이터의 형태를 선언한다. 전달할 데이터가 없으면 아무것도 쓰지 않거나 void라고 쓰면 된다.

### (2) 함수의 호출

특정 기능을 수행할 함수를 선언하여 작성하면, 주함수 또는 다른 사용자 정의 함수에서 함수를 불러 사용할 수 있다. 이때 함수의 이름과 결과를 받을 변수의 데이터 형과 전달할 변수의 데이터 형이 작성한 함수의 것과 일치해야 한다.



## 예제 4-20

두개의 변수에 저장된 값을 더하는 연산을 수행하는 사용자 정의 함수를 작성하고 주함수에서 호출하여 더하기를 수행하자.

```
#include <GC89T51.H> //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED P0

unsigned char x, y; //전역 변수 x, y를 선언한다.

void sum(void){
    unsigned char z;

    z=x+y;
    RobotLED=~z;
}

void main(void){
    x=3; y=5;

    for(;;){
        sum();
    }
}
```

### 실행결과

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ○    | ○    | ○    | ○    |

### (3) 배열

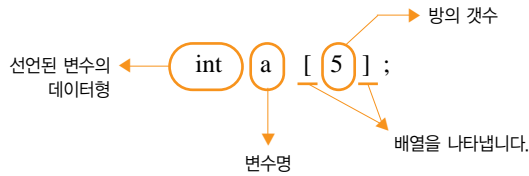
배열이란 같은 변수의 이름으로 여러 개의 방을 만들어 사용하는 것이다. 선언된 변수의 첫 번째 메모리 위치에서 선언된 방의 개수만큼 메모리를 연속으로 사용한다.

#### 1) 배열의 선언



- char x, y ;  
변수 x, y를 문자형 전역 변수로 선언한다.
- x = 3 ; y = 5 ;  
변수 x, y에 각각의 값을 대입한다.
- sum( ) ;  
사용자 정의 함수 sum함수를 호출한다. 실행 위치를 sum( ) 함수가 있는 곳으로 옮겨 간다. 현재 까지 실행된 메인 함수의 위치는 메모리에 저장된다.
- unsigned char z ;  
변수 z를 문자형 지역 변수로 선언한다.
- z = x + y ;  
변수 x와 y의 값을 더한 결과를 z에 넣는다.
- RobotLED = ~z ;  
변수 z의 값을 반전하여 LED에 출력한다.
- sum 함수의 실행문들을 모두 실행하고 나서, sum함수의 끝을 나타내는 종결호를 닫으면 sum 함수를 호출하였던 위치로 돌아간다.
- sum 함수를 모두 실행하고 돌아오면 다시 메인 함수의 실행문들을 실행하게 된다.

배열은 변수를 선언하는 것처럼 사용할 데이터의 형태와 변수의 이름을 선언한 다음, 변수의 이름 옆에 선언할 방의 개수를 선언한다.



[그림 4.3-12] 배열의 선언

배열의 선언에서 방의 개수는 '['와 ']'를 이용하여 나타낸다.

○ 배열의 메모리 주소

int a[5]; 라는 배열을 선언하였을 때 배열의 메모리 위치를 보면,

| 방 번호 | a[0] | a[1] | a[2] | a[3] | a[4] |
|------|------|------|------|------|------|
| 주소   | 0x00 | 0x02 | 0x04 | 0x06 | 0x08 |

※ 변수 a[0]의 주소는 임의로 정한 것이다.

위 표를 보면 메모리의 주소 값이 2씩 증가한다. 이것은 변수 a가 정수형으로 선언되었기 때문에 2바이트씩 차지하므로 메모리의 위치가 2바이트씩 증가한다.

또한 선언된 배열의 순서는 0부터 시작을 하며, 위에서처럼 배열의 선언을 5라고 하면 각각의 방을 부를 때는 배열 0부터 차례로 4까지 부르게 된다.

배열에 대한 여러 가지 설명 중에서 여기서는 정수형 배열을 이용하여 프로그램 작성하는 것을 설명한다.

## 2) 정수형 배열의 초기화

정수형 배열은 정수형으로 선언된 배열을 말한다. 각 방에 초기 데이터를 대입해서 배열을 초기화한다. 초기화에는 배열의 전체에 초기 데이터를 대입하는 것과 일부만 초기화하는 경우가 있다. 단, 전역 변수로 선언된 배열은 0으로 전체가 채워진다.



## 예제 4-21

a를 정수형 배열로 선언하고 초기화를 한 뒤, 값을 차례로 LED에 출력해서 배열의 선언과 초기화에 대해 알아보자.

```
#include<GC89T51.H>    //헤더 파일을 포함시킨다.

//매크로 상수 RobotLED의 선언
#define RobotLED    P0

    void delay(long d_t){
long k;
for(k=0;k<=d_t;k++);
}

void main(void){
int a[5] = {0x00,0x02,0x04,0x08,0x10}, x;

for(;;){
for(x=0; x<5; x++){
RobotLED = ~a[x];
delay(110000);
}
}
}
```

### 실행결과

⇒ a[0]의 값을 출력한 경우

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

⇒ a[1]의 값을 출력한 경우

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⇒ a[2]의 값을 출력한 경우

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ●    | ○    | ○    | ○    | ○    | ○    |



⇒ a[3]의 값을 출력한 경우

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ○    | ○    | ○    | ○    |

⇒ a[4]의 값을 출력한 경우

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ●    | ○    | ○    | ○    |

- `int a[5] = { 0x00, 0x02, 0x04, 0x08, 0x10 } ;`  
a를 정수형 배열로 선언하며, 배열의 개수는 5개이고 선언된 배열의 각 방에 값을 초기화하였다.
- `for( x = 0 ; x < 5 ; x++ ){ ... }`  
변수 x를 0으로 초기화하고 x의 값을 1씩 증가시키며 x의 값이 5보다 작은 동안 중괄호 안의 실행문들을 반복하여 실행한다.
- `RobotLED = ~a[ x ] ;`  
배열 a의 값을 x의 값에 따라 각 방의 값을 반전하여 출력한다.

지금까지 프로그램을 작성하기 위하여 C 언어의 기본적인 문법들 중에서 마이크로프로세서의 구동과 로봇 제작을 위해 필요한 내용만을 다루었다. 더 자세하게 배우려면 C 언어에 관한 전문도서를 참고하자.



.....

.....

.....

.....

.....

# 단원 학습 정리

01. C 언어는 컴파일러를 사용하여 원시 파일에서 목적 파일을 생성하고, 컴파일러가 제공하는 함수들이 있는 파일과 연결하여 최종 실행 파일을 생성하는 언어다.
02. C 언어에서 변수 선언은 데이터를 저장할 장소를 지정하여 이름을 짓는 것이다.
03. 데이터를 저장할 장소의 크기는 데이터 형으로 정한다.
04. 매크로 상수를 이용하면 필요한 수치를 외울 필요 없이 편리하게 프로그래밍할 수 있다.
05. 연산수(Operand, 상수 또는 변수 또는 연산의 결과)로 주어지는 데이터 값들을 사용하여 사칙 연산, 대입, 비교, 논리 연산 등의 계산을 수행하는 기호들을 연산자라고 한다.
06. 제어문은 조건에 따라 프로그램의 흐름을 제어하는 문장이다. 제어문에는 조건을 비교하여 프로그램의 흐름을 제어하는 비교 제어문, 조건을 주어 작업을 반복시키는 반복 제어문, 조건에 맞으면 해당 작업을 수행하는 선택 제어문으로 분류된다.
07. 함수는 “특정한 기능을 수행하는 실행문들의 집합체”이다. 특정 기능을 함수로 묶어 사용하면 프로그램에 중복되는 기능들을 함수를 호출하여 사용할 수 있다.

# 단원정리문제

01

함수의 형태를 나타낼 때 되돌려 줄 결과 값이 없을 때 사용하는 예약어는?

- ① int                      ② char                      ③ void                      ④ unsigned

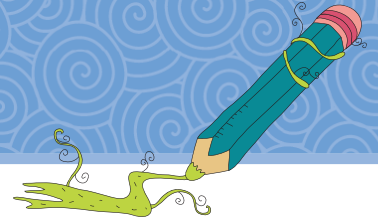
02

배열을 선언할 때 배열의 크기를 나타내는 문자는?

- ① { 와 }                      ② ( 와 )                      ③ [ 와 ]                      ④ “ 와 ”

03

1부터 10까지의 합을 구하면서 과정을 모두 LED에 표시하는 프로그램을 작성하여 보자. 이때 사용자 정의 함수를 사용하여 되돌려 받은 결과 값으로 출력하자.



04

두 개의 변수에 각각 값을 넣어 사용자 정의 함수에서 사칙 연산을 수행한 뒤, 결과 값을 되돌려 받아 1초에 한번 씩 덧셈을 수행한 결과부터 순차적으로 출력하는 프로그램을 작성하여 보자.

05

위의 4번 문제를 배열을 이용하여 사칙연산의 결과를 저장하고 저장된 값을 출력하는 프로그램을 작성하여 보자.

# 04. C 언어로 로봇 움직이기

## ROBOT CONTROL SYSTEM

### 학습목표



- 라인트레이서에서 상황에 따라 점퍼선을 연결하는 방법과 C 언어로 프로그램해서 로봇을 구동하는 것을 실습한다.



#### 인터럽트 (Interrupt)

프로그램 실행 중에 중앙제어장치가 강제적으로 제어를 특정 주소로 옮기는 것. 프로그램 실행 중에 인터럽트가 발생하면 그 프로그램의 실행을 중단하고 그 시점에서의 CPU 내의 중요 데이터를 주기억 장치로 되돌려 놓은 다음, 특정 주소로부터 시작되는 프로그램에 제어를 옮긴다. 인터럽트를 원인별로 분류하면 장치나 프로그램의 고장과 같이 비상시에 발생하는 것, 주기적인 시각(時刻)마다 발생하는 것, 입출력 장치의 완료 보고 등이 있다.

지금까지 학습한 C 언어를 기초로 프로그램 실습 예제를 통하여 로봇을 움직이기 위한 기본적인 포트제어와 A/D 변환, 인터럽트, 타이머/카운터, 모터제어 등을 학습한다. 이론과 실습을 병행하면서 지능형 로봇의 기초지식을 학습하게 된다.



### 1. LED 제어 실습

- 1 [그림 4.4-1], [그림 4.4-2],와 같이 라인트레이서의 P2포트에 LED를 연결하고 다음과 같이 약 0.5초 간격으로 LED0에서부터 LED7까지 켜지는 위치가 차례로 이동하고 나서, LED7이 켜지면 다시 역방향으로 LED7부터 LED0까지 차례로 이동하면서 켜지는, 패턴이 무한정 반복하도록 프로그램한다.



LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

⏴ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⋮

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

⏴ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |

⏴ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

⏴ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ●    | ○    | ○    |

⋮

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

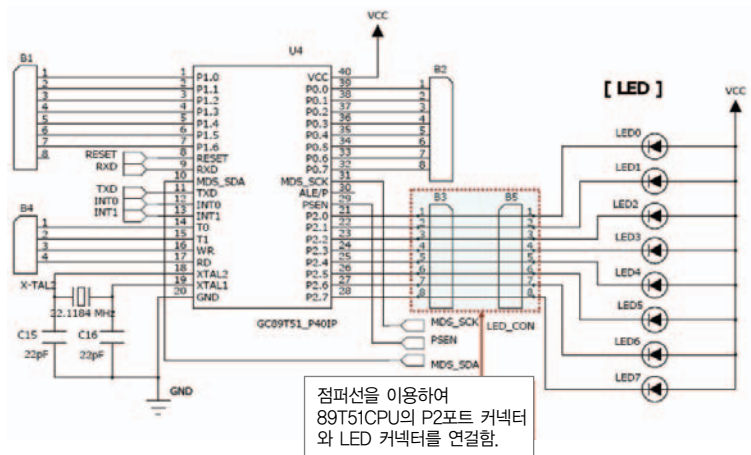
⏴ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

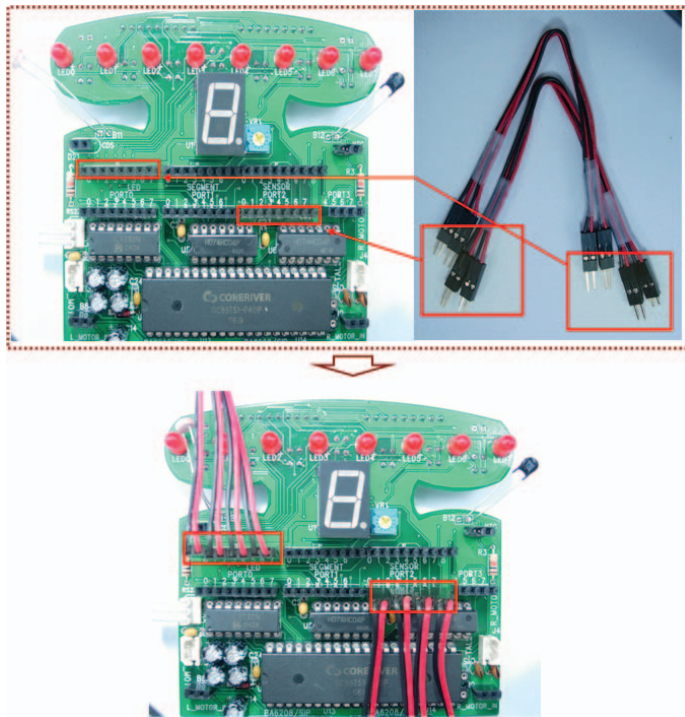
⏴ 다시 처음부터 시작한다.

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| LED 포트         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

● 라인트레이서 회로 결선 방법



[그림 4.4-1] P2 포트와 LED 연결 회로도



[그림 4.4-2] P2 포트와 LED 연결 사진





### 해답 소스

```
#include <GC89T51.H>          // 89T51의 입출력 포트 정의 헤더 파일

/* LED의 점등 패턴에 대한 데이터를 미리 배열로 선언하였다. */
char LED_data[9]= {0xff, 0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f};

void delay(int time)        //시간 지연을 위한 함수 정의
{
    unsigned int i=0,j=0;    //시간 지연을 위한 매개변수를 선언하고 초기화

    for(i=0;i<time;i++){
        for(j=0;j<=1500;j++);
    }
}

void main(void)
{
    char k=0;                //배열의 주소 변경을 위한 매개 변수를 선언하고 초기화
    for (;)                  //무한 루프
    {
        for (k=0;k<=8;k++) { // 배열 값을 0에서부터 8까지 LED에 출력
            P2=LED_data[k];
            delay(50);       //시간을 약 0.5초 지연
        }
        for(k=7;k>0;k--){    // 배열 값을 7에서부터 1까지 LED에 출력
            P2=LED_data[k];
            delay(50);       //시간을 약 0.5초 지연
        }
    }
}
```

- ② [그림 4.4-1], [그림 4.4-2]와 같이 P2 포트에 LED를 연결하고 다음과 같이 LED를 이용하여 0부터 255까지의 숫자를 2진수로 계수하는 프로그램 작성한다. 이때 최초의 LED의 상태는 모두 켜진 상태다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |

♡ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |

⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ●    | ●    | ●    | ●    | ●    | ●    |

⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ●    | ●    | ●    | ●    | ●    | ●    |

⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ○    | ●    | ●    | ●    | ●    | ●    |

⏏ (0.2초)



⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

⏏ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

⇒ 다시 처음부터 시작 한다.



| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| LED 포트         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

### 해답 소스

```
#include <GC89T51.H>           // 89T51의 입출력 포트 정의 헤더 파일

void delay(int time)          //시간 지연을 위한 함수 정의
{
  unsigned int i=0,j=0;       //시간 지연을 위한 매개 변수를 선언하고 초기화
  for(i=0;i<time;i++){
    for(j=0;j<=1500;j++){
    }
  }
}

void main(void)
{
  char LED_DATA=0;           //LED에 출력해서 매개 변수 선언하고 초기화
  for (;) {                  //무한 루프
    /* PORT2에 연결되어진 LED에 LED_DATA값 출력. */
    P2= LED_DATA;
    delay(20)                //시간을 0.2초 지연
    LED_DATA++;              //LED_DATA를 1 증가

    /* LED_DATA가 255보다 크면 LED_DATA를 0으로 초기화 */
    if(LED_DATA>255) LED_DATA=0;
  }
}
```

## 2. 스위치 제어 실습

- 1 [그림 4.4-3], [그림 4.4-4]와 같이 P2 포트에 LED를 연결하고 라인트레이서의 SW1 스위치를 이용하여 LED를 구동하는데, 다음과 같이 SW1 스위치를 누르면 전체 LED가 약 1초 간격으로 켜지고 꺼지기를 계속 반복하는 프로그램을 작성한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

SW1 스위치 누름



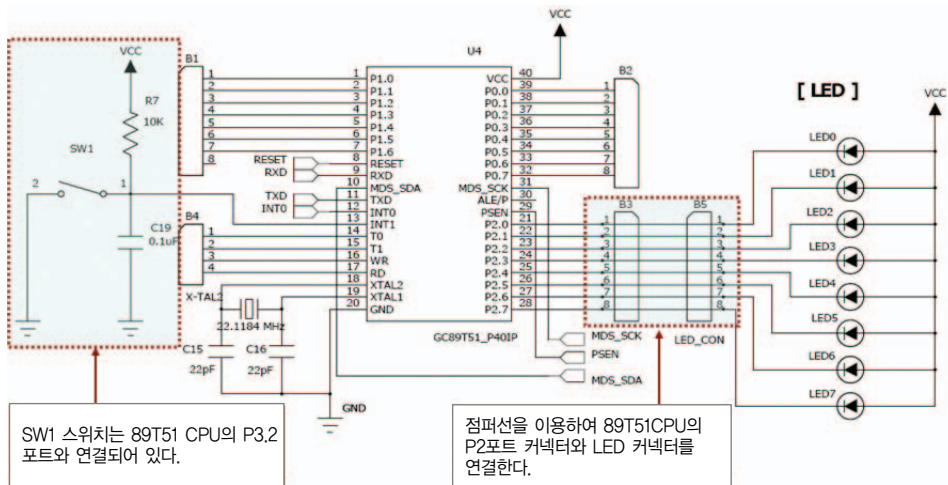
| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

♡ (1초)

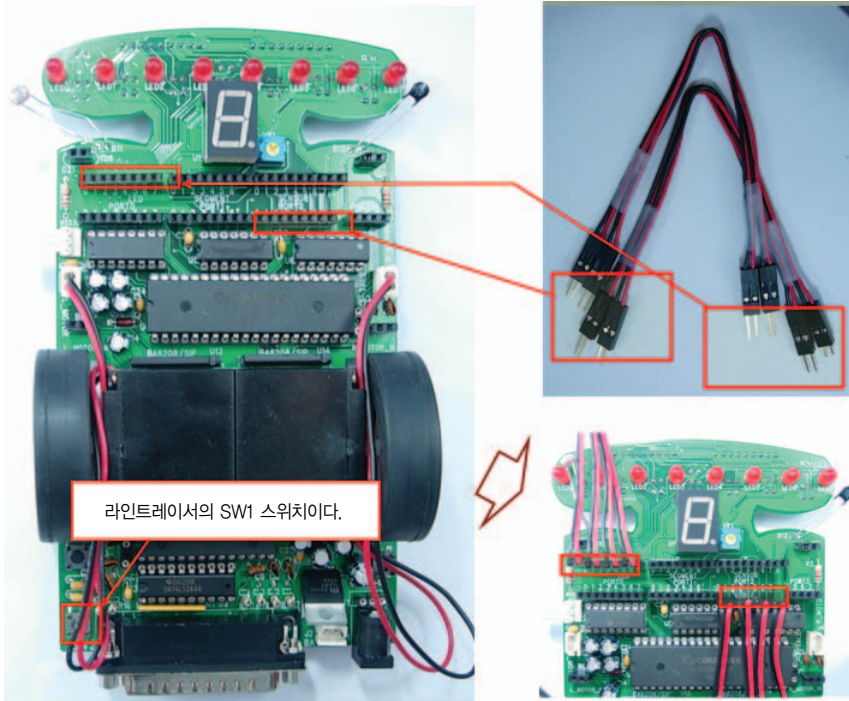
| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |

### 라인트레이서 핀 연결 방법

| LED 포트 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|--------|---|---|---|---|---|---|---|---|
| PORT2  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |



[그림 4.4-3] SW1을 이용한 LED 제어 회로도



[그림 4.4-4] SW1을 이용하여 LED 제어 사진

### 해답 소스

```
#include <GC89T51.h> // 89T51의 입출력 포트 정의 헤더 파일

#define SW1 P3.2 // 포트 3의 3번째 비트를 SW1 입력으로 지정

void delay(int time) // 시간 지연을 위한 함수 정의
{
    unsigned int i=0,j=0; // 시간지연을 위한 매개변수를 선언하고 초기화

    for(i=0;i<time;i++){
        for(j=0;j<=1500;j++);
    }
}

void main(void)
{
    char SW_Value=0; // SW1 입력 값을 저장하는 변수를 선언하고 초기화
```

```

for(;;)
{
    if (SW1==0) {SW_Value=1;} //SW1이 눌리면 SW_Value값을 1로 변화시킴
    if(SW_Value==1){        // SW1이 눌렸으면 수행
        P2=0xFF;           // PORT2에 연결된 LED 전체를 끄
        delay(100);        //시간을 약 1초 지연
        P2=0x00;           //PORT2에 연결된 LED 전체를 켜
        delay(100);        //시간을 약 1초 지연
    }
}
}
}

```

② [그림 4.4-5], [그림 4.4-6]과 같이 P2 포트에 7-세그먼트를 연결하고 SW1, SW2스위치와 7-세그먼트를 이용하여 상승/하강 계수를 시키는 데, SW1 스위치를 누르면 7-세그먼트의 표시 숫자가 1씩 증가하고, SW2 스위치를 누르면 7-세그먼트의 표시 숫자가 1씩 감소하도록 하는 프로그램을 작성한다. 단, 7-세그먼트의 초기 값은 5이며, 0미만의 숫자와 9를 초과하지 않는 숫자 범위 내에서 구동해야 한다.

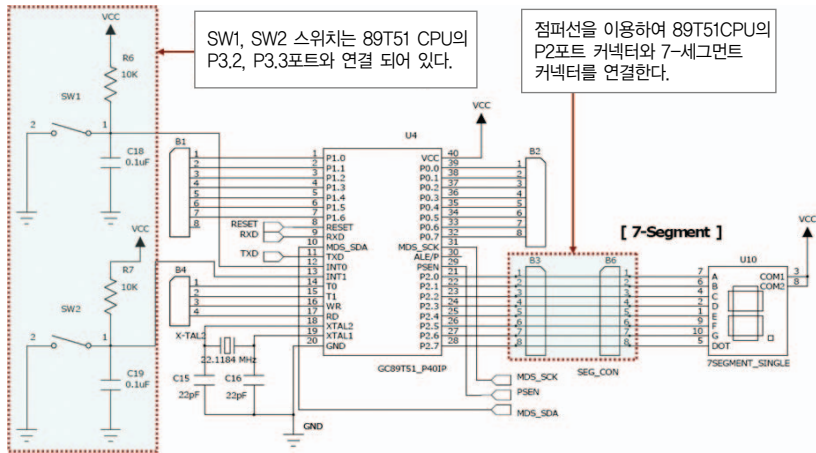
SW1 스위치 누름

⇒ 7-세그먼트 숫자가 1씩 증가되어짐  
(최대 9까지만 증가)

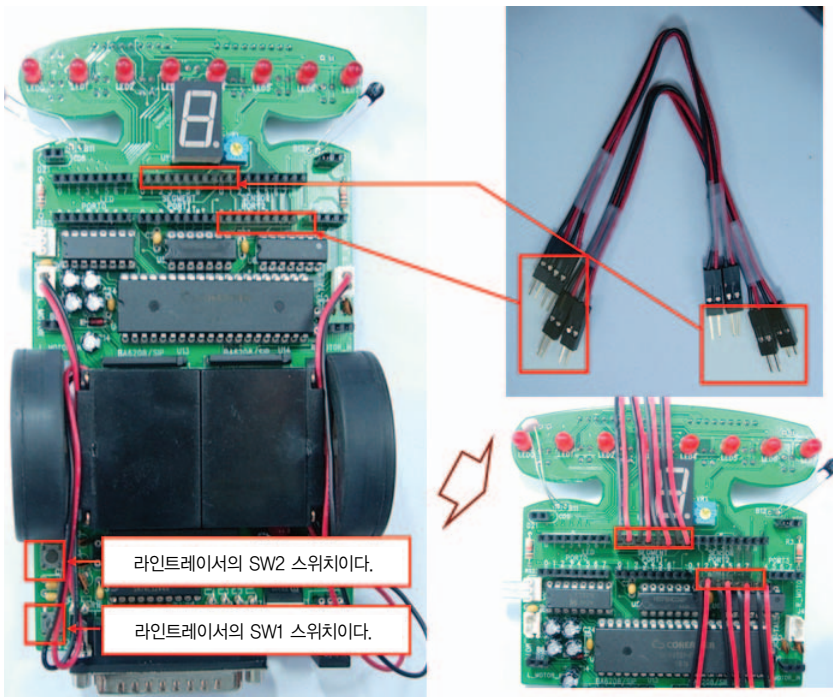
SW2 스위치 누름

⇒ 7-세그먼트 숫자가 1씩 감소되어짐  
(최소 0까지만 감소)

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SEGMENT 포트     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |



[ 그림 4.4-5] P2 포트에 7-세그먼트를 연결하고 SW1, SW2로 상승/하강 제어 회로도



[그림 4.4-6] P2 포트에 7-세그먼트를 연결하고 SW1, SW2로 상승/하강 제어 사진



### 해답 소스

```
#include <GC89T51.H> // 89T51의 입출력 포트 정의의 헤더 파일
#define SW1 P3.2 // 포트 3의 3번째 비트를 SW1 입력으로 지정
#define SW2 P3.3 // 포트 3의 4번째 비트를 SW2 입력으로 지정
char SEG_Font[10] = { 0xc0, 0xf9, 0xa4, // '0', '1', '2'
                    0xb0, 0x99, 0x92, // '3', '4', '5'
                    0x82, 0xd8, 0x80, // '6', '7', '8'
                    0x90}; // '9'
// 7-세그먼트 폰트를 배열로 지정해 놓는다.
void delay(int time) // 시간지연을 위한 함수 정의
{
    unsigned int i=0, j=0; // 시간지연을 위한 매개변수를 선언하고 초기화



    for(i=0; i<time; i++){
        for(j=0; j<=1500; j++){
        }
    }
}
void main(void)
{
    char SW_Count=5; // 7-세그먼트 상태를 나타내는 매개변수

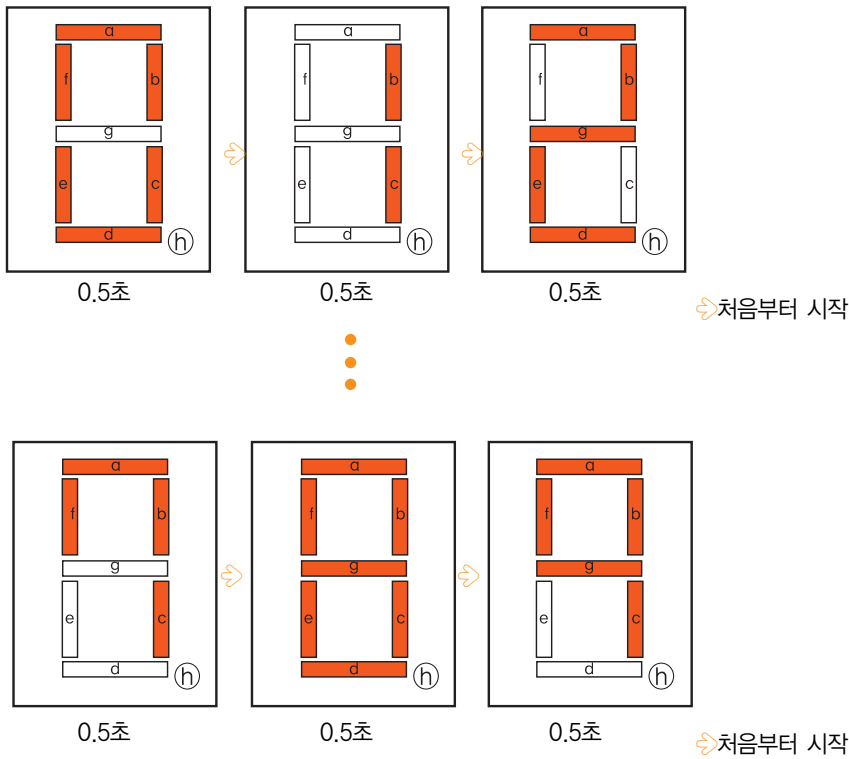
    while(1)
    {
        if(SW1==0) { // SW1이 눌리면 다음을 수행.
            SW_Count++; // SW_Count를 1 증가
            if(SW_Count>9) SW_Count=9; // SW_Count는 최대 9.
        }
        if(SW2==0) { // SW2가 눌리면 다음을 수행
            SW_Count--; // SW_Count를 1 감소
            if(SW_Count<0) SW_Count=0; // SW_Count는 최소 0
        }
        P2=SEG_Font[SW_Count]; // 포트2로 7-세그먼트에 숫자를 출력.
        delay(10); // SW_Count 변화에 시간지연을 줌
    }
}
```



### 3. 7-세그먼트 제어 실습

① [그림 4.4-7], [그림 4.4-8]과 같이 P2 포트에 7-세그먼트를 연결하고 다음과 같이 7-세그먼트에 약 0.5초 간격으로 0~9까지 숫자가 표시되도록 하는데, 숫자 9를 표시하고 나면 다시 0부터 반복적으로 숫자를 표시하도록 하자.

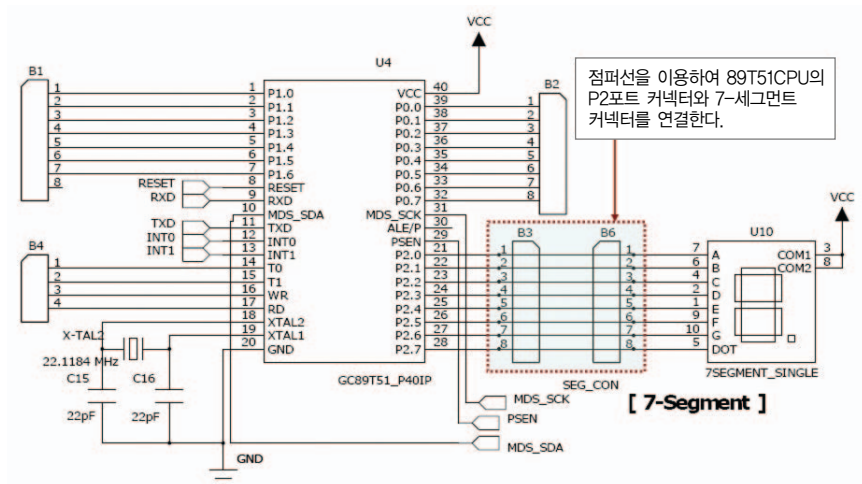
7-세그먼트 LED :  → 꺼짐 (OFF)  → 켜짐 (ON)



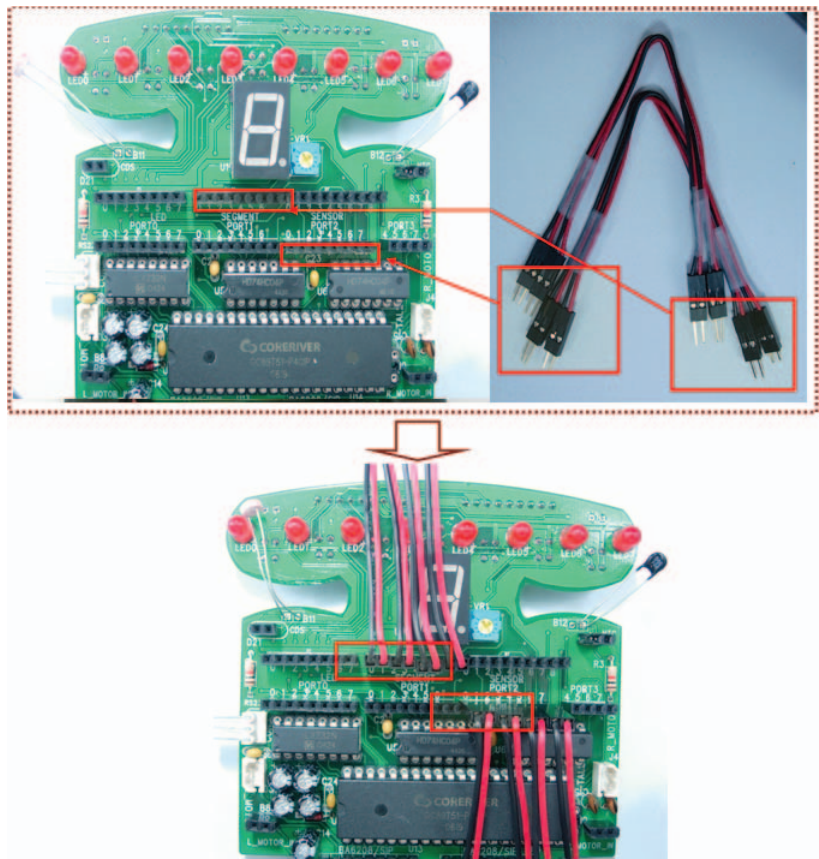
라인트레이서 핀 연결 방법

|           |   |   |   |   |   |   |   |   |
|-----------|---|---|---|---|---|---|---|---|
| SEGMENT포트 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

● 라인트레이서 회로 결선 방법



[그림 4.4-7] P2 포트에 7-세그먼트 연결 회로도



[그림 4.4-8] P2 포트에 7-세그먼트 연결 사진



## 해답 소스

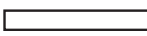

```
#include <GC89T51.H>           // 89T51의 입출력 포트 정의 헤더 파일

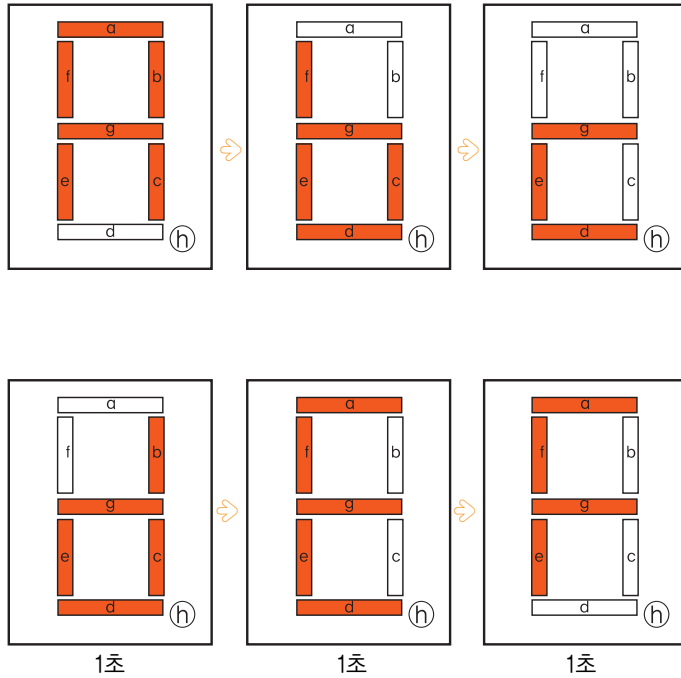
void delay(int time)          // 시간 지연을 위한 함수 정의
{
    unsigned int i=0,j=0;     // 시간 지연을 위한 매개 변수를 선언하고 초기화

    for(i=0;i<time;i++){
        for(j=0;j<=1500;j++);
    }
}

void main(void)
{
    while(1)
    {
        P2=0xc0;              // 7-세그먼트에 숫자 '0'을 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0xf9;              // 7-세그먼트에 숫자 '1'을 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0xa4;              // 7-세그먼트에 숫자 '2'를 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0xb0;              // 7-세그먼트에 숫자 '3'을 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0x99;              // 7-세그먼트에 숫자 '4'를 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0x92;              // 7-세그먼트에 숫자 '5'를 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0x82;              // 7-세그먼트에 숫자 '6'을 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0xd8;              // 7-세그먼트에 숫자 '7'를 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0x80;              // 7-세그먼트에 숫자 '8'을 출력
        delay(50);            // 시간을 약 0.5초 지연
        P2=0x90;              // 7-세그먼트에 숫자 '9'를 출력
        delay(50);            // 시간을 약 0.5초 지연
    }
}
```

- ② [그림 4.4-9, 4.4-10과 같이 P2 포트에 7-세그먼트를 연결하고 다음과 같이 7-세그먼트로 영문 알파벳 A, b, c, d, E, F를 약 0.5초 간격으로 하나씩 표시하는 프로그램을 작성한다. 단, 알파벳은 대소문자를 구분하여 표시한다. 7-세그먼트 LED 각각의 초기 상태는 모두 꺼진 상태이다.

7-세그먼트 LED :  → 꺼짐 (OFF)  → 켜짐 (ON)



라인트레이서 핀 연결 방법

| SEGMENT포트 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----------|---|---|---|---|---|---|---|---|
| PORT2     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |



## 해답 소스

```
#include <GC89T51.H> // 89T51의 입출력 포트 정의 헤더파일

void delay(int a) // a/1000초 시간지연 함수
{
    unsigned int i=0,j=0; // 시간지연을 위한 매개변수를 선언하고 초기화
    for (i=0;i<a;i++) {
        for(j=0;j<=550;j++)
        }
    }

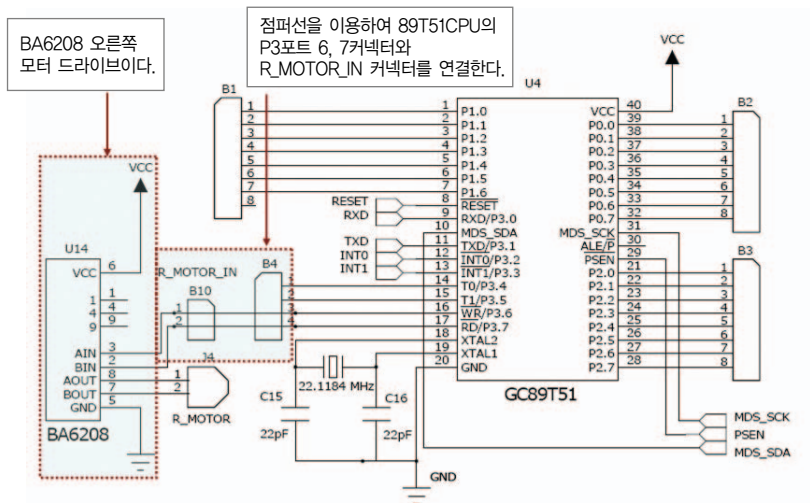
void main(void)
{
    while(1)
    {
        P2=0x88; // 7-세그먼트에 'A'가 출력
        delay(1000); // 시간을 약1초 지연
        P2=0x83; // 7-세그먼트에 'b'가 출력
        delay(1000); // 시간을 약1초 지연
        P2=0xa7; // 7-세그먼트에 'c'가 출력
        delay(1000); // 시간을 약1초 지연
        P2=0xa1; // 7-세그먼트에 'd'가 출력
        delay(1000); // 시간을 약1초 지연
        P2=0x86; // 7-세그먼트에 'f'가 출력
        delay(1000); // 시간을 약1초 지연
        P2=0x8e; // 7-세그먼트에 'f'가 출력
        delay(1000); // 시간을 약1초 지연
    }
}
```

## 4. DC 모터 제어 실습

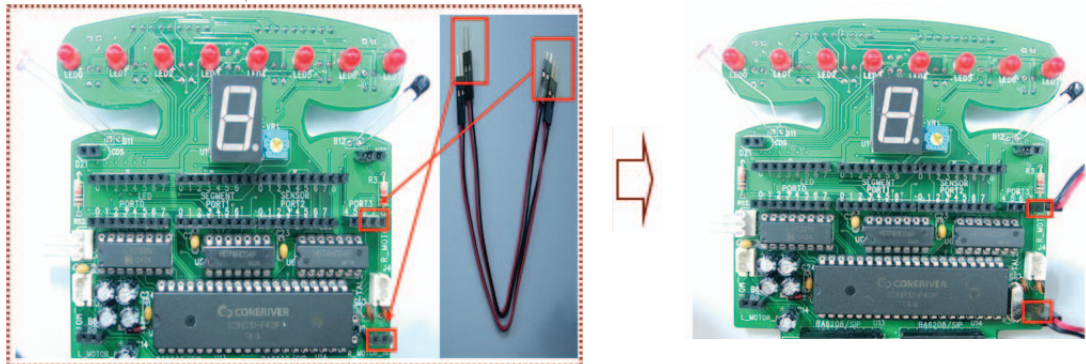
- ① [그림 4.4-9], [그림 4.4-10]와 같이 P3.6, P3.7 포트에 오른쪽 모터 드라이버를 연결하고 라인트레이서의 오른쪽 바퀴만 정방향 회전하도록 하는 프로그램을 작성하여 보자.

| 라인트레이서 핀 연결 방법 |                 |                 |
|----------------|-----------------|-----------------|
| MOTOR 포트       | R_MOTOR_IN(1포트) | R_MOTOR_IN(2포트) |
| PORT3          | 6               | 7               |

### ● 라인트레이서 회로 결선 방법



[그림 4.4-9] P3.6, P3.7 비트에 오른쪽 모터 드라이버 연결 회로도



[그림 4.4-10] P3.6, P3.7 비트에 오른쪽 모터 드라이버 연결 사진





### 해답 소스

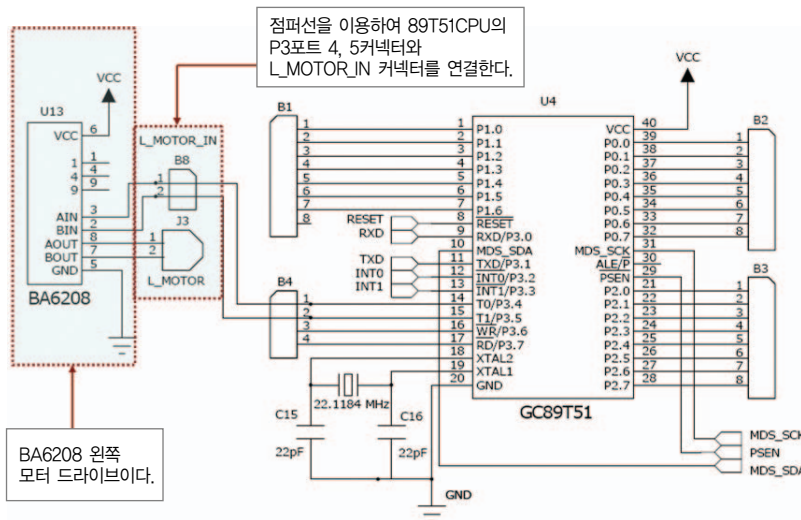
```
#include <GC89T51.H> // 89T51의 입출력 포트 정의 헤더파일
#define Right_Motor_1 P3.6 // P3의 비트 6을 우측 모터 역방향 비트로 정의
#define Right_Motor_0 P3.7 // P3의 비트 7을 우측 모터 정방향 비트로 정의

void main(void)
{
    while(1)
    {
        Right_Motor_1=0; // 오른쪽 모터가 정방향으로 회전하도록 역방향 비트를 0으로 설정.
        Right_Motor_0=1; // 오른쪽 모터가 정방향으로 회전하도록 정방향 비트를 1로 설정.
    }
}
```

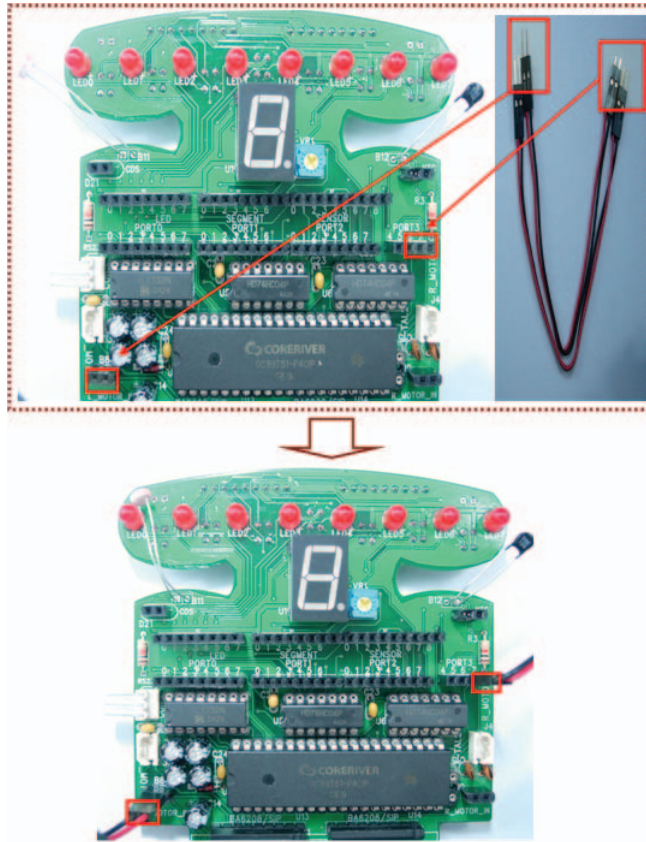
② [그림 4.4-11], [4.4-12]와 같이 P3.4, P3.5 포트에 왼쪽 모터 드라이버를 연결하고 라인트레이서의 왼쪽 바퀴만 역 방향 회전하도록 하는 프로그램을 작성하여 보자.

| 라인트레이서 핀 연결 방법 |                 |                 |
|----------------|-----------------|-----------------|
| MOTOR 포트       | L_MOTOR_IN(1포트) | L_MOTOR_IN(2포트) |
| PORT3          | 4               | 5               |

### ● 라인트레이서 회로 결선 방법



[그림 4.4-11] P3.4, P3.5 비트에 왼쪽 모터 드라이버를 연결한 회로도



[그림 4.4-12] P3.4, P3.5 비트에 왼쪽 모터 드라이버를 연결한 사진

### 해답 소스

```
#include <GC89T51.H>    // 89T51의 입출력 포트 정의의 헤더 파일

#define Left_Motor_0 P3.5 // P3의 비트 5를 왼쪽 모터 정방향 비트로 정의
#define Left_Motor_1 P3.4 // P3의 비트 4를 왼쪽 모터 정방향 비트로 정의

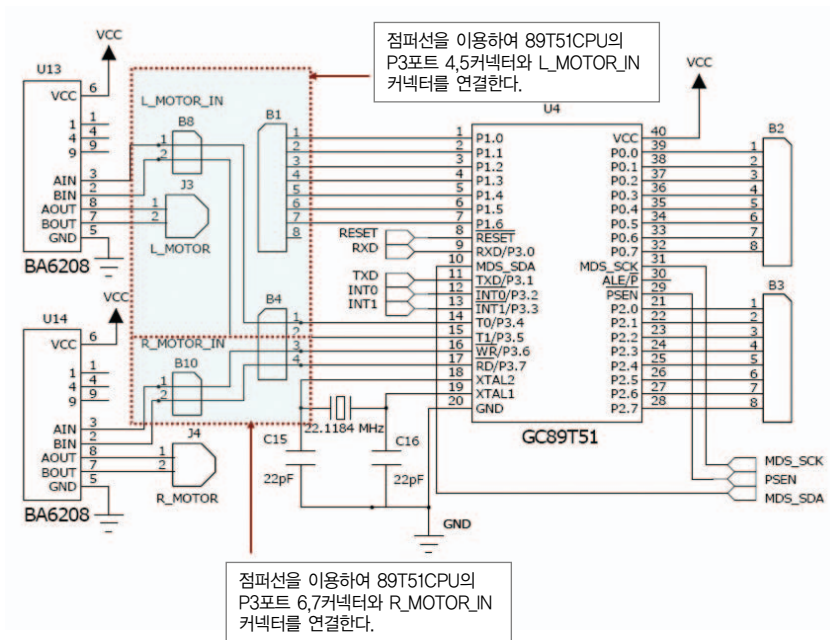
void main(void)
{
    while(1)
    {
        Left_Motor_0=1; // 왼쪽 모터가 역방향으로 회전하도록 정방향 비트를 1로 설정.
        Left_Motor_1=0; // 왼쪽 모터가 역방향으로 회전하도록 정방향 비트를 0으로 설정.
    }
}
```



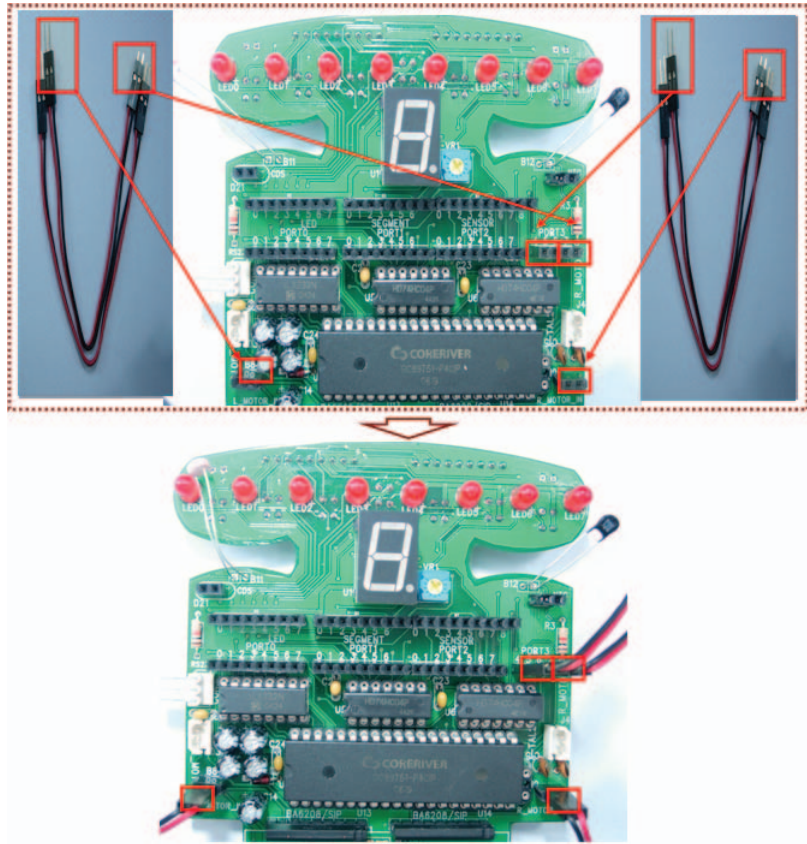
- ③ [그림 4.4-13], [그림 4.4-14]과 같이 P3.4 ~ P3.7에 왼쪽과 오른쪽 모터 드라이버를 연결하고 라인트레이서가 직선 방향으로 전진하도록 하는 프로그램을 작성하여 보자.

| 라인트레이서 핀 연결 방법 |                  |                  |                  |                  |
|----------------|------------------|------------------|------------------|------------------|
| MOTOR 포트       | L_MOTOR_IN (1포트) | L_MOTOR_IN (2포트) | R_MOTOR_IN (1포트) | R_MOTOR_IN (2포트) |
| PORT3          | 4                | 5                | 6                | 7                |

● 라인트레이서 회로 결선 방법



[그림 4.4-13] P3.4 ~ P3.7에 왼쪽과 오른쪽 모터 드라이버를 연결한 회로도



[그림 4.4-14] P3.4 ~ P3.7에 왼쪽과 오른쪽 모터 드라이버를 연결한 사진

### 해답 소스

```
#include <GC89T51.H> // 89T51의 입출력 포트 정의 헤더 파일

#define Left_Motor_1 P3.4 // P3의 비트 4를 왼쪽 모터 역방향 비트로 정의
#define Left_Motor_0 P3.5 // P3의 비트 5를 왼쪽 모터 정방향 비트로 정의
#define Right_Motor_1 P3.6 // P3의 비트 6을 오른쪽 모터 역방향 비트로 정의
#define Right_Motor_0 P3.7 // P3의 비트 7을 오른쪽 모터 정방향 비트로 정의

void main(void)
{
    while(1) {

        /* 라인트레이서의 양쪽 모터를 정방향으로 회전시킨다. */
        Left_Motor_1 = 0;
        Left_Motor_0 = 1;
```



```

Right_Motor_1 = 0;
Right_Motor_0 = 1;

    }
}

```

- ④ [그림 4.4-15], [그림 4.4-16]과 같이 P3.4 ~ P3.7에 왼쪽과 오른쪽 모터 드라이버를 연결하고 라인트레이서가 직선 방향으로 후진하도록 하는 프로그램을 작성하여 보자.

| 라인트레이서 핀 연결 방법 |                     |                     |                     |                     |
|----------------|---------------------|---------------------|---------------------|---------------------|
| MOTOR 포트       | L_MOTOR_IN<br>(1포트) | L_MOTOR_IN<br>(2포트) | R_MOTOR_IN<br>(1포트) | R_MOTOR_IN<br>(2포트) |
| PORT3          | 4                   | 5                   | 6                   | 7                   |

### 해답 소스

```

#include <GC89T51.H>           // 89T51의 입출력 포트 정의 헤더 파일

#define Left_Motor_1  P3.4 // P3의 비트 4를 왼쪽 모터 역방향 비트로 정의
#define Left_Motor_0  P3.5 // P3의 비트 5를 왼쪽 모터 정방향 비트로 정의
#define Right_Motor_1 P3.6 // P3의 비트 6을 오른쪽 모터 역방향 비트로 정의
#define Right_Motor_0 P3.7 // P3의 비트 7를 오른쪽 모터 정방향 비트로 정의

void main(void)
{
    while(1) {

        /* 라인트레이서의 양쪽 모터를 역방향으로 회전한다. */
        Left_Motor_0 =0;
        Left_Motor_1 =1;
        Right_Motor_0 =0;
        Right_Motor_1 =1;

    }
}

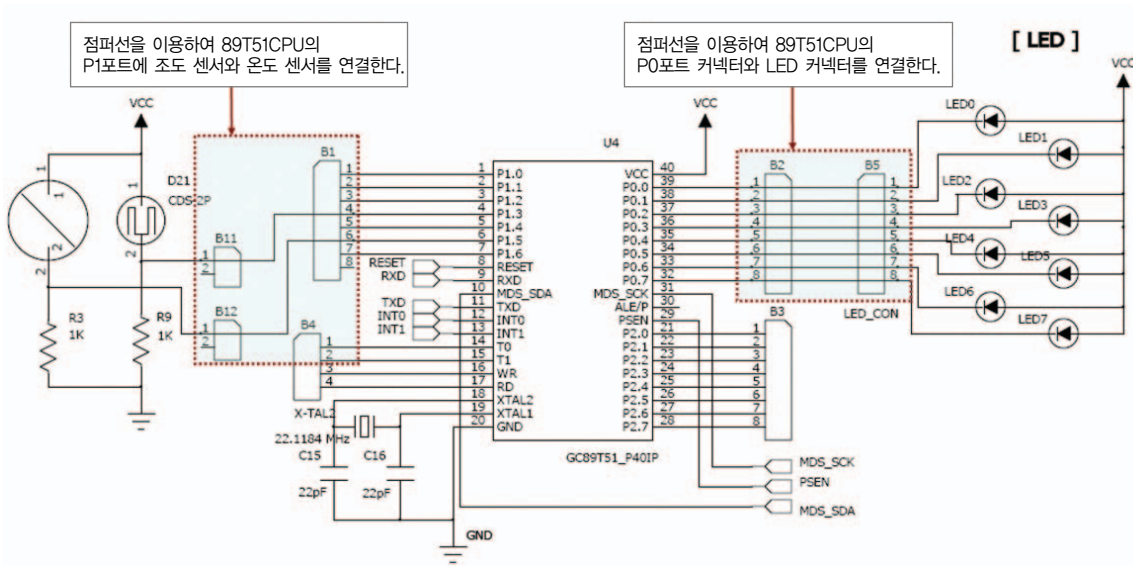
```

## 5. A/D 변환, 조도/온도 센서 제어 실습

- ① [그림 4.4-15], [그림 4.4-16]과 같이 P1.3 포트에 CDS 조도 센서를 연결하고 P1.5 포트에 NTC 조도 센서를 연결하고 라인트레이서에 탑재된 CDS 조도 센서가 감지하는 조도 값을 A/D 변환하고 8bit 2진수 값으로 환산하여 LED로 표시하는 프로그램을 작성하여 보자.

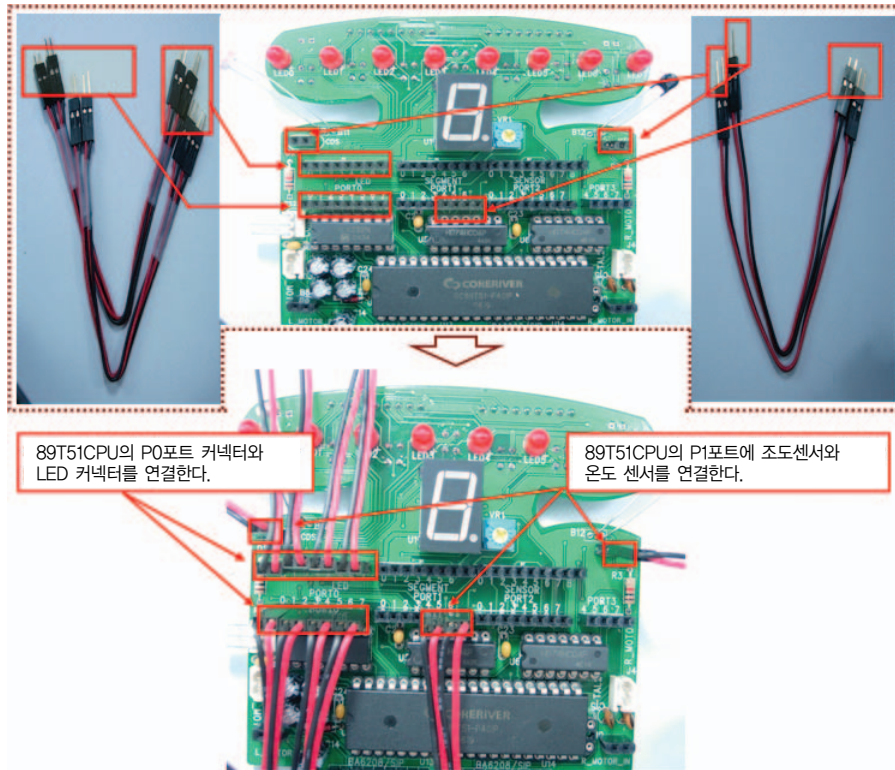
| 라인트레이서 핀 연결 방법 |           |   |   |   |           |   |   |   |
|----------------|-----------|---|---|---|-----------|---|---|---|
| 센서 단자          | CDS 조도 센서 |   |   |   | NTC 온도 센서 |   |   |   |
| PORT1          | 3         |   |   |   | 5         |   |   |   |
| SEGMENT포트      | 0         | 1 | 2 | 3 | 4         | 5 | 6 | 7 |
| PORT2          | 0         | 1 | 2 | 3 | 4         | 5 | 6 | 7 |

### ● 라인트레이서 회로 결선 방법



[그림 4.4-15] 조도센서와 온도센서 연결 회로도





[그림 4.4-16] 조도센서와 온도센서 연결 사진

### 해답 소스

```
#include <GC89T51.H>           // 89T51의 입출력 포트 정의 헤더 파일

#define LED_data  P0           // LED 인터페이스 포트를 P0으로 정의
#define NTC      P1.3         // NTC 온도 센서의 입력 핀을 포트 1.3으로 설정
#define CDS      P1.5         // CDS 조도 센서의 입력 핀을 포트 1.5으로 설정

// 센서로부터 받는 아날로그 신호를 A/D 변환
int Operate_ADC(unsigned char ch)
{
    unsigned int value;

    // A/D 변환 입력 채널 3 또는 5 선택
    if ((ch == 3) || (ch == 5))
    {
        ADCSEL &= 0xF8;        // A/D 변환 입력 채널 선택 비트 초기화
        ADCSEL |= ch;         // A/D 변환 입력 채널 선택
    }
}
```

```

}
else return(0);

ADCON |= 0x40;    // AD_REQ = 1, A/D 변환 요청
while (ADCON & 0x40); // A/D 변환 완료 확인

value = ADCR * 4;    // 상위 8 비트 값 입수
value += ADCON & 0x03; // 하위 2 비트 값 합산
return(value);      // A/D 변환 값 전달
}

// A/D 변환에 필요한 레지스터를 초기화
void Init_ADC(void)
{
    ADCSEL &= 0x1F;
    ADCSEL |= 0x20;    // A/D 클럭을 2 분주
    ADCHEN |= 0x08;    // A/D 채널 3 입력 허용
    ADCHEN |= 0x20;    // A/D 채널 5 입력 허용

    EA = 1;            // 전체 인터럽트 허용
    EADC = 1;          // A/D 변환 인터럽트 허용
    ADCON |= 0x80;     // AD_EN = 1, A/D 변환 허용
}

void adc_int(void) interrupt ADC_VECTOR
{
    ADCON &= 0xEF;    // ADCR = 0, A/D 변환 인터럽트 발생표시 소거
}

void main(void)
{
    long Advalue=0;    //A/D 변환값를 저장할 변수를 long 형으로 선언하고 초기화

    Init_ADC();
    while(1)
    {
        // P1.3에서 받는 조도 센서 신호를 A/D 변환
        Advalue = Operate_ADC(3); // Advalue 변수를 통해 조도 센서 A/D 값을 받는다.

        Advalue = (Advalue*256)/1024; // 10bit입력 값을 8bit로 변환 한다.
        LED_data= Advalue;           // LED에 Advalue 값을 출력한다.
    }
}

```





② [그림 4.4-15], [그림 4.4-16]과 같이 P1.3 포트에 CDS 조도센서를 연결하고 P1.5 포트에 NTC 조도센서를 연결하고 NTC 온도센서의 온도 값을 A/D 변환한 후 이를 0~5사이의 정수 값으로 환산하였을 때, 결과 값이 2 이상이면 아래의 그림과 같이 LED 전체를 켜고 끄기를 약 1초 간격으로 반복하는 프로그램을 작성하여 보자.

NTC 서미스터 온도 센서의 온도 값이 2이상



| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |

♡ (0.2초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

라인트레이서 핀 연결 방법

| 센서 단자  | CDS 조도 센서 |   |   |   | NTC 온도 센서 |   |   |   |
|--------|-----------|---|---|---|-----------|---|---|---|
| PORT1  | 3         |   |   |   | 5         |   |   |   |
| LED 포트 | 0         | 1 | 2 | 3 | 4         | 5 | 6 | 7 |
| PORT2  | 0         | 1 | 2 | 3 | 4         | 5 | 6 | 7 |

### 해답 소스

```
#include <GC89T51.H> // 89T51의 입출력 포트 정의 헤더 파일
#define LED_data P0 // SEGMENT 인터페이스 포트를 P0으로 정의
#define NTC P1.3 // NTC 온도 센서의 입력 핀을 포트 1.3으로 설정
#define CDS P1.5 // CDS 조도 센서의 입력 핀을 포트 1.5으로 설정
```

// 센서로부터 받는 아날로그 신호를 A/D 변환

```
int Operate_ADC(unsigned char ch)
```

```
{
```

```
    unsigned int value;
```

```

// A/D 변환 입력 채널 3 또는 5 선택
if ((ch == 3) || (ch == 5))
{
    ADCSEL &= 0xF8;      // A/D 변환 입력 채널 선택 비트 초기화
    ADCSEL |= ch;       // A/D 변환 입력 채널 선택
}
else return(0);

ADCON |= 0x40;        // AD_REQ = 1, A/D 변환 요청
while (ADCON & 0x40); // A/D 변환 완료 확인

value = ADCR * 4;     // 상위 8 비트 값 입수
value += ADCON & 0x03; // 하위 2 비트 값 합산
return(value);       // A/D 변환 값 전달
}

// A/D 변환에 필요한 레지스터를 초기화
void Init_ADC(void)
{
    ADCSEL &= 0x1F;
    ADCSEL |= 0x20;    // A/D 클럭을 2 분주
    ADCHEN |= 0x08;   // A/D 채널 3 입력 허용
    ADCHEN |= 0x20;   // A/D 채널 5 입력 허용

    EA = 1;           // 전체 인터럽트 허용
    EADC = 1;        // A/D 변환 인터럽트 허용
    ADCON |= 0x80;   // AD_EN = 1, A/D 변환 허용
}

void adc_int(void) interrupt ADC_VECTOR
{
    ADCON &= 0xEF;    // ADCR = 0, A/D 변환 인터럽트 발생 표시 소거
}

void delay(int time) //시간 지연을 위한 함수 정의
{
    unsigned int i=0,j=0; //시간 지연을 위한 매개 변수를 선언하고 초기화

    for(i=0;i<time;i++){
        for(j=0;j<=1500;j++);
    }
}

```



```

    }
}

void main(void)
{
    long NTC_Advalue=0; //A/D 변환값을 저장할 변수

    Init_ADC();
    while(1)
    {
        // P1.5에서 받는 온도 센서 신호를 A/D 변환
        NTC_Advalue = Operate_ADC(5); //A/D 변환 값을 NTC_Advalue로 받는다.

        NTC_Advalue=(NTC_Advalue*5)/1024;

        if(NTC_Advalue>2){ //온도 센서 값이 2이상이면 LED를ON/OFF
            LED_data=0x00;
            delay(100);
            LED_data=0xff;
            delay(100);
        }
        else LED_data=LED_data; //온도 센서 값이 2 이상이 아니면 LED는 그대로
    }
}

```

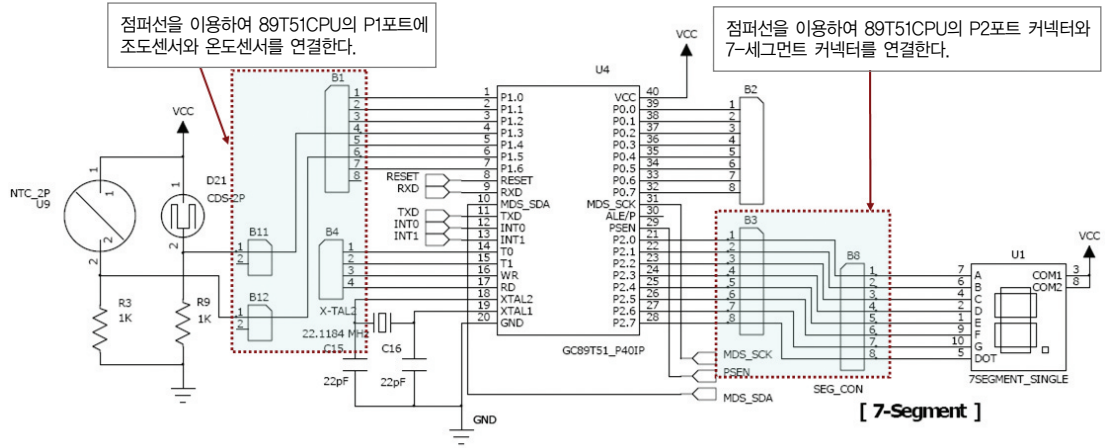
③ [그림 4.4-17], [그림 4.4-18]과 같이 조도 센서, 온도 센서, 7-세그먼트를 연결하고 라인트레이서에 탑재된 NTC 서미스터 온도 센서를 이용하여 온도 센서가 감지하는 온도 값을 A/D 변환한 후 7-세그먼트에 0~5V 사이의 정수 값으로 표현하는 프로그램을 작성하여 보자.



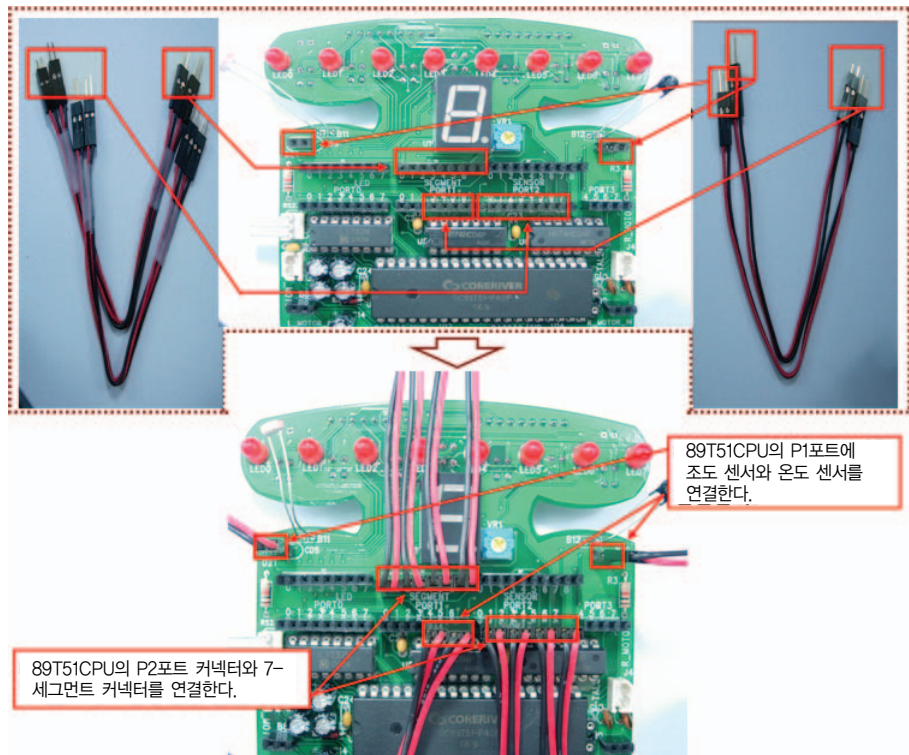
**서미스터 (thermistor)**

전자부품으로 사용하기에 알맞은 저항값과 온도 특성을 가진 반도체 디바이스. 온도가 높아지면 저항값이 내려가는 NTC(negative temperature coefficient thermistor), 온도가 상승하면 저항값이 올라가는 PTC(positive temperature coefficient thermistor), 특정 온도에서 저항값이 급변하는 CTR(critical temperature resistor)로 분류된다.

| 라인트레이서 핀 연결 방법 |           |   |   |   |           |   |   |   |
|----------------|-----------|---|---|---|-----------|---|---|---|
| 센서 단자          | CDS 조도 센서 |   |   |   | NTC 온도 센서 |   |   |   |
| PORT1          | 3         |   |   |   | 5         |   |   |   |
| Segment 포트     | 0         | 1 | 2 | 3 | 4         | 5 | 6 | 7 |
| PORT2          | 0         | 1 | 2 | 3 | 4         | 5 | 6 | 7 |



[그림 4.4-17] 조도센서, 온도 센서, 7-세그먼트 연결 회로도



[그림 4.4-18] 조도센서, 온도센서, 7-세그먼트 연결 사진



## 해답 소스

```
#include <GC89T51.H>    // 89T51의 입출력 포트 정의 헤더 파일

#define SEGMENT_data P2 // SEGMENT 인터페이스 포트를 P0으로 정의
#define NTC    P1.3    // NTC 온도센서의 입력 핀을 포트 1.3으로 설정한다.
#define CDS    P1.5    // CDS 조도센서의 입력 핀을 포트 1.5으로 설정한다.

// 센서로부터 받는 아날로그 신호를 A/D 변환한다.
int Operate_ADC(unsigned char ch)
{
    unsigned int value;

    // A/D 변환 입력 채널 3 또는 5 선택
    if ((ch == 3) || (ch == 5))
    {
        ADCSEL &= 0xF8;    // A/D 변환 입력 채널 선택 비트 초기화
        ADCSEL |= ch;    // A/D 변환 입력 채널 선택
    }
    else return(0);

    ADCON |= 0x40;    // AD_REQ = 1, A/D 변환 요청
    while (ADCON & 0x40); // A/D 변환 완료 확인

    value = ADCR * 4;    // 상위 8 비트 값 입수
    value += ADCON & 0x03; // 하위 2 비트 값 합산
    return(value);    // A/D 변환 값 전달
}

// A/D 변환에 필요한 레지스터를 초기화한다.
void Init_ADC(void)
{
    ADCSEL &= 0x1F;
    ADCSEL |= 0x20;    // A/D 클럭을 2 분주
    ADCHEN |= 0x08;    // A/D 채널 3 입력 허용
    ADCHEN |= 0x20;    // A/D 채널 5 입력 허용

    EA = 1;    // 전체 인터럽트 허용
    EADC = 1;    // A/D 변환 인터럽트 허용
    ADCON |= 0x80;    // AD_EN = 1, A/D 변환 허용
}
```

```

}

void adc_int(void) interrupt ADC_VECTOR
{
    ADCON &= 0xEF;    // ADCR = 0, A/D 변환 인터럽트 발생표시 소거
}

void main(void)
{
    long Advalue=0; // A/D 변환값을 저장할 변수를 long 형으로 설정하고 초기화

    Init_ADC();    // A/D 변환 모듈을 초기화
    while(1)
    {
        // P1.5에서 받는 온도 센서 신호를 A/D 변환한다.
        Advalue = Operate_ADC(5); // A/D 변환 값을 Advalue로 받는다.

        Advalue = (Advalue*5)/1024; // 10bit 입력 값을 0~5 사이 정수로 변환

        if(Advalue==0) SEGMENT_data=0xC0; // 7-세그먼트에 숫자 '0'을 출력
        else if(Advalue==1) SEGMENT_data=0xF9; // 7-세그먼트에 숫자 '1'을 출력
        else if(Advalue==2) SEGMENT_data=0xA4; // 7-세그먼트에 숫자 '2'를 출력
        else if(Advalue==3) SEGMENT_data=0xB0; // 7-세그먼트에 숫자 '3'을 출력
        else SEGMENT_data=0x99; // 7-세그먼트에 숫자 '4'를 출력
    }
}

```



## 6. 적외선 센서 제어 실습

- ① [그림 4.4-19], [그림 4.4-20]와 같이 연결하고 라인트레이서의 P3포트를 이용하여 적외선 발광센서 8개 모두 켜고, P2포트를 통해 적외선 수광 센서 8개의 값을 모두 입력받아 이를 LED에 표시한다. 다음의 표와 같이 P2.0과 연결된 수광 센서가 적외선을 감지하면 LED0를 켜고, P2.1과 연결된 수광 센서가 적외선을 감지하면 LED1을 켜는 방법으로 LED0

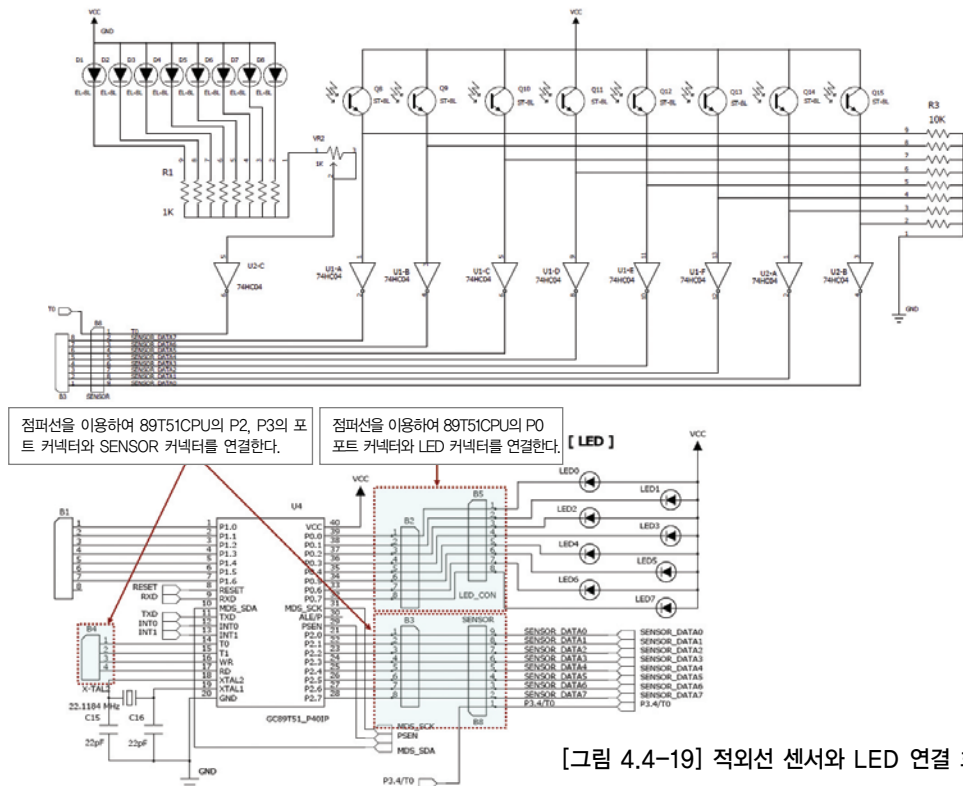


~ LED7을 이용해 수광 센서가 감지한 값을 모두 표시하여 보자. 단, 적외선 수광 센서가 적외선을 감지 못하면 LED는 꺼져야 한다.

| P2.0 감지 | P2.1 감지 | P2.2 감지 | P2.3 감지 | P2.4 감지 | P2.5 감지 | P2.6 감지 | P2.7 감지 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| LED0 ON | LED1 ON | LED2 ON | LED3 ON | LED4 ON | LED5 ON | LED6 ON | LED7 ON |

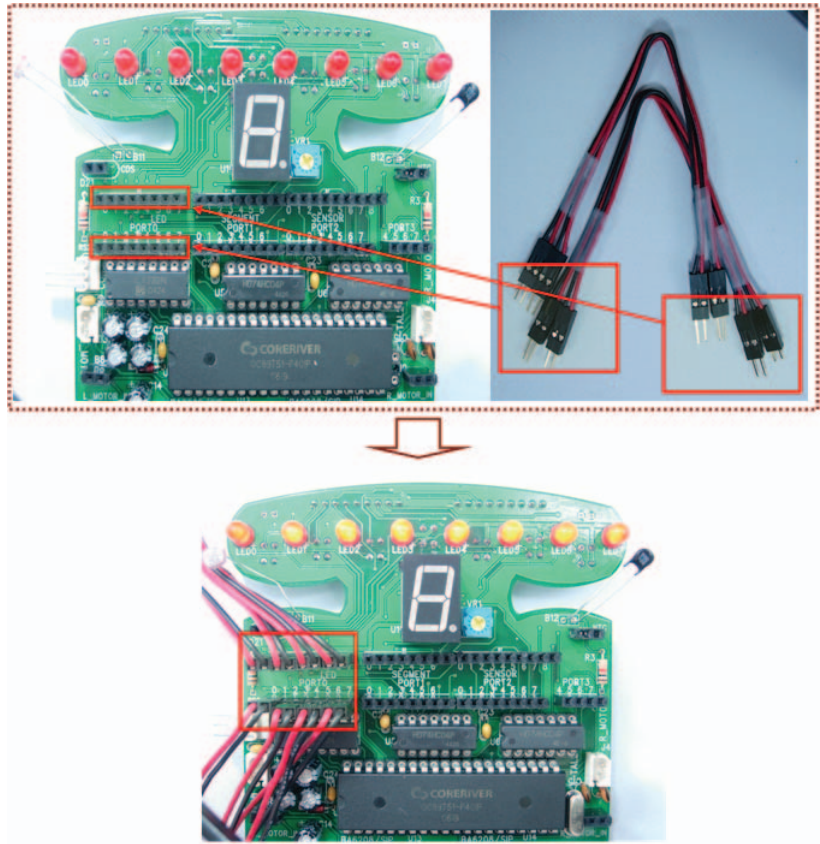
| 라인트레이서 핀 연결 방법 |   |   |                                                               |   |   |   |   |   |
|----------------|---|---|---------------------------------------------------------------|---|---|---|---|---|
| SENSOR         | 0 | 1 | 2                                                             | 3 | 4 | 5 | 6 | 7 |
| PORT 2         | 0 | 1 | 2                                                             | 3 | 4 | 5 | 6 | 7 |
| SENSOR         | E |   |                                                               |   |   |   |   |   |
| PORT3          | 4 | 1 | 2                                                             | 3 | 4 | 5 | 6 | 7 |
| LED 포트         | 0 | 1 | 2                                                             | 3 | 4 | 5 | 6 | 7 |
| PORT0          | 0 | 1 | 2 </td <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> | 3 | 4 | 5 | 6 | 7 |

● 라인트레이서 회로 결선 방법



[그림 4.4-19] 적외선 센서와 LED 연결 회로도





[그림 4.4-20] 적외선 센서와 LED 연결 사진

### 해답 소스

```
#include <GC89T51.H>

#define SENSOR    P2    // 적외선 센서 인터페이스 포트를 P2로 정의
#define LEDdata   P0    // LED 인터페이스 포트를 P0으로 정의
#define SENSOR_EN P3.4  // 적외선 발광 센서 ON/OFF 신호선으로 지정

void main(void)
{
    SENSOR_EN=1;    //발광 센서를 켜 준다.

    while(1)
    {
        LEDdata=SENSOR; //적외선 입력 값을 바로 LED 로 표현한다.
    }
}
```



② [그림 4.4-21], [그림 4.4-22]과 같이 연결하고 라인트레이서의 P3포트를 이용하여 적외선 발광센서 8개를 모두 켜고, P2포트를 이용하여 적외선 수광 센서 8개의 값을 모두 입력받아 7-세그먼트에 표시한다. 다음의 표와 같이 P2.0과 연결된 수광 센서가 적외선을 감지하면 7-세그먼트의 숫자는 0, P2.1에 연결되어진 수광 센서가 적외선을 감지하면 7-세그먼트에 숫자는 1, 같은 방법으로 수광 센서의 적외선 감지 값을 7-세그먼트에 0 ~ 7까지 숫자로 표시하는 프로그램을 작성한다. 단, 1개의 수광 센서도 적외선을 입력받지 못하면 7-세그먼트 값은 8, 2개 이상의 수광센서가 동시에 적외선을 감지하면 7-세그먼트의 숫자는 9로 표시한다.

|                 |                 |                 |                 |                 |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| P2.0 감지         | P2.1 감지         | P2.2 감지         | P2.3 감지         | P2.4 감지         |
| 7-세그먼트<br>숫자는 0 | 7-세그먼트<br>숫자는 1 | 7-세그먼트<br>숫자는 2 | 7-세그먼트<br>숫자는 3 | 7-세그먼트<br>숫자는 4 |
| P2.5 감지         | P2.6 감지         | P2.7 감지         | 적외선 감지 못함       | 2개 이상 동시감지      |
| 7-세그먼트<br>숫자는 5 | 7-세그먼트<br>숫자는 6 | 7-세그먼트<br>숫자는 7 | 7-세그먼트<br>숫자는 8 | 7-세그먼트<br>숫자는 9 |

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SENSOR         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT 2         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| SENSOR         | E |   |   |   |   |   |   |   |
| PORT3          | 4 |   |   |   |   |   |   |   |
| SEGMENT 포트     | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT0          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |





## 해답 소스

```
#include <GC89T51.H>           // 89T51의 입출력 포트 정의 헤더 파일

#define SENSOR      P2        // 적외선 센서 인터페이스 포트를 P2로 정의
#define SEGMENTdata P0        // 7-세그먼트 인터페이스 포트를 P0으로 정의
#define SENSOR_EN   P3.4     // 발광 센서 ON/OFF제어를 위해 지정

const char Font[10] = { 0xc0, 0xf9, 0xa4, // '0', '1', '2'
                        0xb0, 0x99, 0x92, // '3', '4', '5'
                        0x82, 0xd8, 0x80, // '6', '7', '8'
                        0x90};         // '9'
                                // 7-세그먼트 숫자 폰트를 배열로 지정해 놓는다.

unsigned char SENSOR_DATA(void) // 센서의 값에 따라 7-세그먼트값을 출력하는 함수
{
    if(SENSOR==0xff){return 8;} // 적외선 센서 입력이 없으면 8값이 출력
    else if(SENSOR==0xfe){return 0;} // 적외선 센서 Q0이 입력받으면 출력 값은 0
    else if(SENSOR==0xfd){return 1;} // 적외선 센서 Q1이 입력받으면 출력 값은 1
    else if(SENSOR==0xfb){return 2;} // 적외선 센서 Q2가 입력받으면 출력 값은 2
    else if(SENSOR==0xf7){return 3;} // 적외선 센서 Q3이 입력받으면 출력 값은 3
    else if(SENSOR==0xef){return 4;} // 적외선 센서 Q4이 입력받으면 출력 값은 4
    else if(SENSOR==0xdf){return 5;} // 적외선 센서 Q5가 입력받으면 출력 값은 5
    else if(SENSOR==0xbf){return 6;} // 적외선 센서 Q6이 입력받으면 출력 값은 6
    else if(SENSOR==0x7f){return 7;} // 적외선 센서 Q7이 입력받으면 출력 값은 7
    else {return 9;} // 적외선 센서가 동시에 입력 받으면 9값이 출력
}

void delay(int time) //시간 지연을 위한 함수 정의
{
    unsigned int i=0,j=0; //시간 지연을 위한 매개 변수를 선언하고 초기화

    for(i=0;i<time;i++){
        for(j=0;j<=1500;j++){
        }
    }
}

void main(void)
{
    SENSOR_EN=1; // 발광 센서를 켜다.

    while(1)
    {
```

```

/* 적외선 센서의 값을 입력받아 비교한 후 세그먼트에
적외선 센서 값이 인지된 부분을 숫자로 출력한다. */
SEGMENTdata=Font[SENSOR_DATA()];
delay(10); //시간을 지연해 준다.
}
}

```

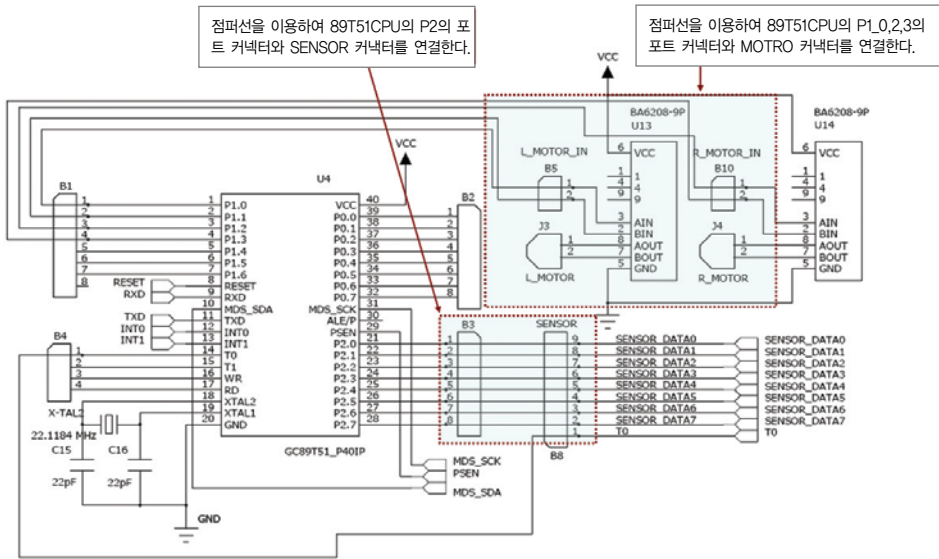
③ [그림 4.4-23], [그림 4.4-24]과 같이 연결하고 라인트레이서의 P3포트를 이용하여 적외선 발광센서 8개를 모두 켜고, P2포트를 이용하여 적외선 수광 센서의 값을 모두 입력받아 라인트레이서의 바퀴를 구동한다. 적외선 수광 센서들 중 Q0이 적외선을 감지하면 라인트레이서의 왼쪽 바퀴가 정방향으로 회전하고, Q0이 감지하지 못하면 라인트레이서의 왼쪽 바퀴는 정지되며, 적외선 수광 센서 Q7이 적외선을 감지하면 오른쪽 바퀴가 정방향으로 회전하고, Q7이 감지하지 못하면 라인트레이서의 오른쪽 바퀴가 정지하도록 하는 프로그램을 작성한다.



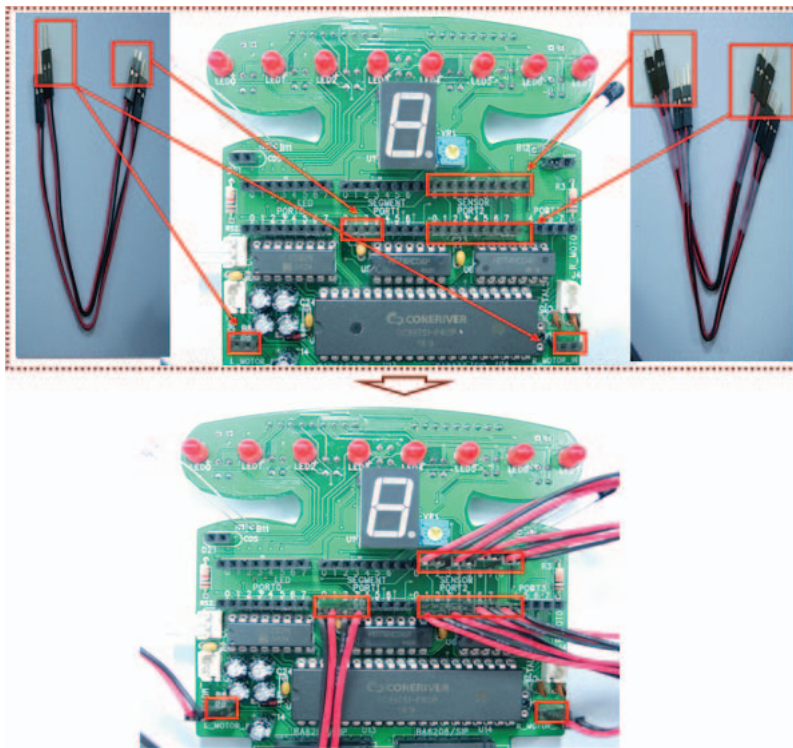
| 라인트레이서 핀 연결 방법 |                 |   |                 |   |                 |   |                 |   |
|----------------|-----------------|---|-----------------|---|-----------------|---|-----------------|---|
| SENSOR 포트      | 0               | 1 | 2               | 3 | 4               | 5 | 6               | 7 |
| PORT2          | 0               | 1 | 2               | 3 | 4               | 5 | 6               | 7 |
| SENSOR 포트      | E               |   |                 |   |                 |   |                 |   |
| PORT3          | 4               |   |                 |   |                 |   |                 |   |
| MOTOR 포트       | L_MOTOR_IN(1포트) |   | L_MOTOR_IN(2포트) |   | R_MOTOR_IN(1포트) |   | R_MOTOR_IN(2포트) |   |
| PORT2          | 0               |   | 1               |   | 2               |   | 3               |   |



### ● 라인트레이서 회로 결선 방법



[그림 4.4-23] 적외선 센서와 오른쪽과 왼쪽 모터 드라이버 연결 회로도



[그림 4.4-24] 적외선 센서와 오른쪽과 왼쪽 모터 드라이버 연결 사진

### 해답 소스

```
#include <GC89T51.H> // 89T51의 입출력 포트 정의 헤더 파일

#define SENSOR0    P2.0 // 적외선 센서 Q0의 값을 입력받는다.
#define SENSOR7    P2.7 // 적외선 센서 Q7의 값을 입력받는다.

#define SENSOR_EN  P3.4 // 적외선 발광 센서를 켜고/끄는다.

#define RightM_0   P1.1 // P1의 비트 1을 오른쪽 모터 정방향 비트로 정의.
#define RightM_1   P1.0 // P1의 비트 0를 오른쪽 모터 역방향 비트로 정의.
#define LeftM_0    P1.3 // P1의 비트 3을 왼쪽 모터 정방향 비트로 정의.
#define LeftM_1    P1.2 // P1의 비트 2를 왼쪽 모터 역방향 비트로 정의.

#define ON         0
#define OFF        1

void MOTOR_STOP(void) // 라인트레이서가 정지하도록 하는 함수이다.
{
    RightM_0 = 1;
    RightM_1 = 1;
    LeftM_0 = 1;
    LeftM_1 = 1;
}

void main(void)
{
    SENSOR_EN=1; // 적외선 센서를 발광시킨다.

    while(1)
    {

        if(SENSOR0==ON) // 적외선 센서 Q0이 입력받으면 오른쪽 모터가 정 회전한다.
        {
            RightM_1 =0;
            RightM_0 =1;
        }
        else if(SENSOR7==ON) // 적외선 센서 Q7이 입력받으면 왼쪽 모터가 정 회전한다.
        {
            LeftM_1 =0;
            LeftM_0 =1;
        }
        else MOTOR_STOP(); // 라인트레이서를 정지한다.
    }
}
```





## 7. 외부인터럽트 제어 실습

- ① [그림 4.4-19], [그림 4.4-20]와 같이 연결하고 라인트레이서의 SW1, SW2 스위치를 이용하여 적외선 센서를 제어하는데, SW1 스위치를 누르면 적외선 발광 센서가 켜지고, SW2 스위치를 누르면 적외선 발광센서가 꺼지는 프로그램을 작성하여 보자. 이때 P2 포트를 이용하여 적외선 수광 센서의 값을 입력받아 이를 LED를 사용하여 아래의 표와 같이 수광 센서의 센서 값을 모두 표시한다. 단, 적외선 수광 센서가 적외선을 감지 못하면 LED는 꺼지고, 스위치 입력은 외부인터럽트를 사용한다.

| P2.0 감지 | P2.1 감지 | P2.2 감지 | P2.3 감지 | P2.4 감지 | P2.5 감지 | P2.6 감지 | P2.7감지  |
|---------|---------|---------|---------|---------|---------|---------|---------|
| LED0 ON | LED1 ON | LED2 ON | LED3 ON | LED4 ON | LED5 ON | LED6 ON | LED7 ON |

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SENSOR 포트      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT 2         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| SENSOR 포트      | E |   |   |   |   |   |   |   |
| PORT3          | 4 |   |   |   |   |   |   |   |
| LED 포트         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT0          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

### 해답 소스

```
#include <GC89T51.H>

#define SENSOR    P2 //적외선 센서를 포트 2로 입력받도록 지정
#define LED_data  P0 //세그먼트의 출력 포트를 지정
#define SENSOR_EN P3.4 //발광 센서 ON/OFF 제어를 위해 지정

char SW=2; //어느 스위치가 눌렸는지 알려주는 매개 변수, 선언 후 초기화.
```



```

void main(void)
{
    TCON=0x05;           //외부 인터럽트 0,1을 신호가 하강할 때 받는다.
    EA=1;
    EX0=1;
    EX1=1;
    SENSOR_EN=0;

    while(1)
    {
        if(SW==1) { SENSOR_EN=1; } //SW변수가 1이면 발광 센서를 켜다.
        else { SENSOR_EN=0; } //SW변수가 1이 아니면 발광 센서를 끈다.

        LED_data=SENSOR; //적외선 센서 값을 LED로 디스플레이 해준다.
    }
}

//외부 인터럽트 0이 입력 받으면 SW값이 1로 된다.
void EX0_int(void) interrupt IE0_VECTOR
{
    SW=1;
}

//외부 인터럽트 1이 입력 받으면 SW값이 2로 된다.
void EX1_int(void) interrupt IE1_VECTOR
{
    SW=2;
}

```



## 8. 타이머/카운터 제어 실습

- 1 [그림 4.4-13], [그림 4.4-14]과 같이 P3.4 ~ P3.7에 왼쪽과 오른쪽 모터 드라이버를 연결하고 라인트레이서의 전진 속도가 0단계부터 3단계 까지 변화되도록 구동하는데, 각 단계별 전진 속도의 변화 시간은 약 3초



간격으로 하며, 전진 속도를 0단계에서 3단계까지 단계적으로 증가시키자. 단, 최초의 라인트레이서 상태는 정지 상태(0단계)이며, 전진 속도를 3단계까지 변화시킨 후에는 라인트레이서를 정지시킨다.

라인트레이서 핀 연결 방법

| MOTOR 포트 | L_MOTOR_IN<br>(1포트) | L_MOTOR_IN<br>(2포트) | R_MOTOR_IN<br>(1포트) | R_MOTOR_IN<br>(2포트) |
|----------|---------------------|---------------------|---------------------|---------------------|
| PORT3    | 4                   | 5                   | 6                   | 7                   |

해답 소스

```
#include <GC89T51.H>

#define RightM_0 P3.5 // P3.5를 우측 모터의 정방향 비트로 설정
#define RightM_1 P3.4 // P3.4를 우측 모터의 역방향 비트로 설정
#define LeftM_0 P3.7 // P3.7를 좌측 모터의 정방향 비트로 설정
#define LeftM_1 P3.6 // P3.6를 좌측 모터의 역방향 비트로 설정

char speed_up=0; //모터 속도설정을 위한 매개 변수를 선언하고 초기화
unsigned int count=0; //타이머/카운터에서 시간알림을 위한 매개 변수, 선언하고 초기화

void delay (unsigned int p) //시간 지연을 위한 함수 정의
{
    int i, j; //시간 지연을 위한 매개 변수를 선언하고 초기화 한다.
    for (j = 0 ; j < p ; j++)
        for (i = 0 ; i < 1000; i++);
}

void speed_go(char speed) //모터를 전진하는 스피드를 정해주는 함수 이다.
{
    LeftM_1 = 0; LeftM_0 = 1; //왼쪽, 오른쪽 전진
    RightM_1 = 0; RightM_0 = 1;
    delay(speed*10); //스피드 값만큼 전진한다.

    LeftM_0 = 0; LeftM_1 = 0; //관성 운전
    RightM_0 = 0; RightM_1 = 0;
    delay(6);
}
```

```

LeftM_0 = 1;   LeftM_1 = 1;       // 왼쪽, 오른쪽 정지
RightM_0 = 1;  RightM_1 = 1;
delay(10-speed);
}

void MOTOR_STOP(void)           //라인트레이서 정지
{
    LeftM_0 = 1;   LeftM_1 = 1;
    RightM_0 = 1;  RightM_1 = 1;
}

void main(void)
{

    /* Timer/Counter 0만 사용하나 TMOD로 Timer/Counter 1도 설정한다.
    일단 TMOD의 상위 4비트를 0으로 해놓자.
    Timer/Counter 0의 GATE=0 : 외부 핀의 영향을 받지 않는다.
    C/T=0 : 내부 클록을 계수한다. M1,M0=0: 모드 0 13비트 타이머이다 */
    TMOD = 0x00;

    /* 계산한 값을 설정한다 */
    TH0 = 0x19;
    TL0 = 0x13;

    /* 타이머 0 카운터 시작 (TR0는 GC89T51.H에 1비트로 선언되었다.
    TR0는 TCON의 비트이므로 TCON = 0x10도 사용할 수 있다) */
    TR0 = 1;

    /* EA, ET도 GC89T51.H에 1 비트 변수로 선언되었다.
    EA와 ET는 IE 레지스터의 비트이므로 IE = 0x82로 사용할 수 있다. */

    /*인터럽트를 사용하려면 먼저 EA 비트를 1로 설정하고*/
    EA = 1;

    /* Timer/Counter 0의 오버플로우 인터럽트를 사용한다. */
    ET0 = 1;

    while(1)
    {
        if(speed_up==0) speed_go(speed_up); // speed_up값이 0이면 전진 속도를
  0으로 한다.
        else if(speed_up==1) speed_go(speed_up); // speed_up값이 1이면 전진 속도
    }
}

```



```

        // speed_up값이 2이면 전진 속도를 1으로 한다.
        else if(speed_up==2)speed_go(speed_up); // speed_up값이 2이면 전진 속도를 2으로 한다.
        else if(speed_up==3)speed_go(speed_up); // speed_up값이 3이면 전진 속도를 3으로 한다.
        else MOTOR_STOP(); // speed_up값이 0,1,2,3이 아니면 정지한다.
    }
}

void T0_int(void) interrupt TF0_VECTOR
{
    /* 일단 오버플로우가 되면 TH0와 TL0가 0이 되므로 처음값을 다시 입력시킨다 */
    TH0 = 0x19;
    TL0 = 0x13;
    count++;

    /* 0.004초마다 인터럽트 루틴을 750번 호출했다면 즉 이 뜻은 0.004*750 = 3초가 된다 */
    if(count >=750){
        count=0; // 3초가 되면 count값을 0으로 초기화한다.
        speed_up++; // 3초마다 speed_up값을 1씩 증가시킨다.
    }
}

```

② [그림 4.4-25], [4.4-26]과 같이 연결하고 다음과 같이 라인트레이서의 구동 속도를 제어하는데, 라인트레이서의 전진 속도를 0단계부터 7단계로 구분하여 약 1초 간격의 라인트레이서 속도를 증가시키고, 속도 변화 단계에 따라 LED0부터 시작하여 LED7까지 약 1초 간격으로 옮겨 가면서 켜지는 프로그램을 작성하여 보자. 단, LED7이 켜지면 라인트레이서를 정지시킨다. 단 최초의 라인트레이서는 정지 상태에 있는 것으로 하며 이때를 0단계로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 1

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 2

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ●    | ○    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 3

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 4

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ●    | ○    | ○    | ○    |

라인트레이서 전진 속도는 5

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ●    | ○    | ○    |

라인트레이서 전진 속도는 6

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

라인트레이서 전진 속도는 7

♡ (약 1초 후)

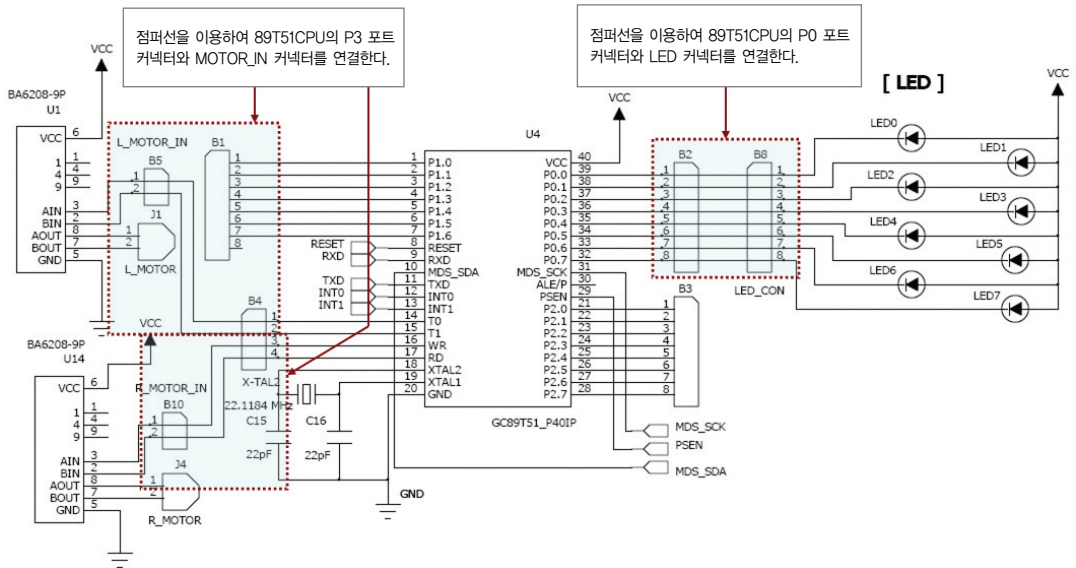


| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |

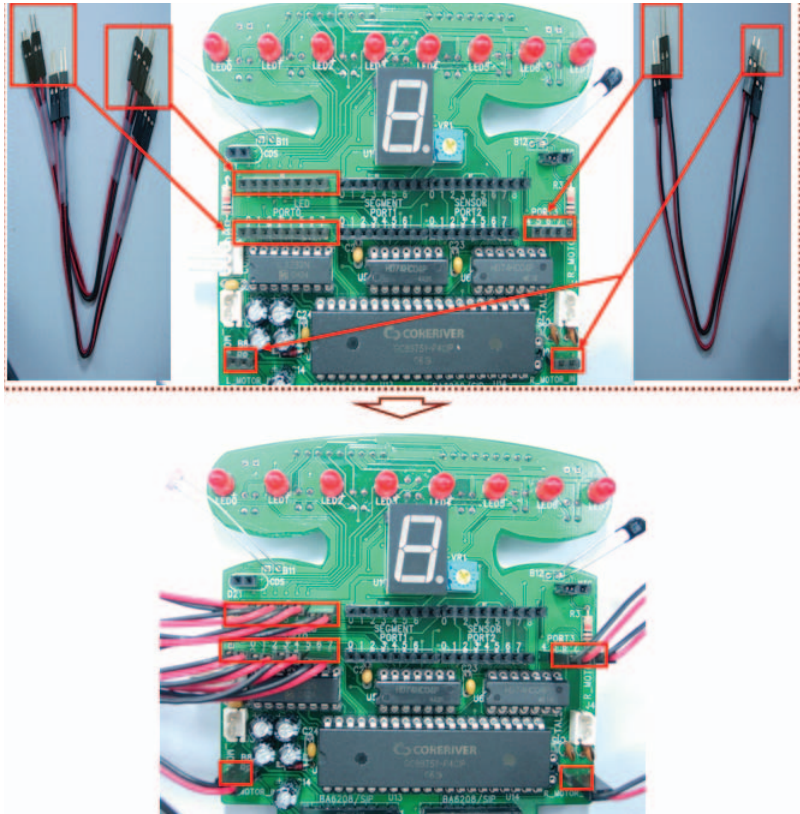
라인트레이서 정지

| 라인트레이서 핀 연결 방법 |                 |   |                 |   |                 |   |                 |   |
|----------------|-----------------|---|-----------------|---|-----------------|---|-----------------|---|
| LED포트          | 0               | 1 | 2               | 3 | 4               | 5 | 6               | 7 |
| PORT0          | 0               | 1 | 2               | 3 | 4               | 5 | 6               | 7 |
| MOTOR 포트       | L_MOTOR_IN(1포트) |   | L_MOTOR_IN(2포트) |   | R_MOTOR_IN(1포트) |   | R_MOTOR_IN(2포트) |   |
| PORT3          | 4               |   | 5               |   | 6               |   | 7               |   |

● 라인트레이서 회로 결선 방법



[그림 4.4-25] 왼쪽과 오른쪽 모터 드라이버와 LED 연결 회로도



[그림 4.4-26] 왼쪽과 오른쪽 모터 드라이버와 LED 연결 사진

### 해답 소스

```
#include <GC89T51.H>           // 89T51의 입출력 포트 정의 헤더 파일

#define LEDdata  P0 // LED 인터페이스 포트를 P0으로 정의
#define RightM_0 P3.5 // P3의 비트 5를 왼쪽 모터 정방향 비트로 정의
#define RightM_1 P3.4 // P3의 비트 4를 왼쪽 모터 역방향 비트로 정의
#define LeftM_0  P3.7 // P3의 비트 7을 오른쪽 모터 정방향 비트로 정의
#define LeftM_1  P3.6 // P3의 비트 6을 오른쪽 모터 역방향 비트로 정의

char speed_up=0; // 모터의 속도 명령을 전달하는 전역 변수를 선언하고 초기화.
unsigned int count=0; // 타이머/카운트에서 시간을 알리는 변수를 선언하고 초기화.
const char Font[8] = {0xfe, 0xfd, 0xfb, 0xf7, 0xef, 0xdf, 0xbf, 0x7f }; // LED 패턴

void delay (unsigned int p) // 시간 지연을 위한 함수 정의
{
    int i, j; // 시간 지연을 위한 매개 변수를 선언하고 초기화
```



```
for (j = 0 ; j < p ; j++)
for (i = 0 ; i < 1000; i++); // 약 0.01초 시간 지연을 갖는다.
}

void speed_go(char speed) // 모터의 전진 속도를 정해주는 함수이다.
{
    LeftM_1 = 0;   LeftM_0 = 1; // 왼쪽, 오른쪽 모터 전진
    RightM_1 = 0;   RightM_0 = 1;

    delay(speed*10); // 스피드 값만큼 전진한다.

    LeftM_0 = 0;   LeftM_1 = 0; // 관성 운전
    RightM_0 = 0;   RightM_1 = 0;
    delay(6);

    LeftM_0 = 1;   LeftM_1 = 1; // 왼쪽, 오른쪽 모터 정지
    RightM_0 = 1;   RightM_1 = 1;
    delay(7-speed);
}

void MOTOR_STOP(void) // 라인트레이서가 정지하도록 하는 함수이다.
{
    LeftM_0 = 1;   LeftM_1 = 1;
    RightM_0 = 1;   RightM_1 = 1;
}

void main(void)
{
    /* Timer/Counter 0만 사용하나 TMOD로 Timer/Counter 1도 설정한다.
    일단 TMOD의 상위 4비트를 0으로 해놓자.
    Timer/Counter 0의 GATE=0 : 외부 핀의 영향을 받지 않는다.
    C/T=0 : 내부 클럭을 계수한다. M1,M0=0: 모드 0 13비트 타이머이다 */
    TMOD = 0x00;

    /* Timer/Counter 0 주기를 0.004초로 설정한다. */
    TH0 = 0x19;
    TL0 = 0x13;

    /* 타이머 0 카운터 시작 (TR0는 GC89T51.H에 1비트로 선언되었다.
    TR0는 TCON의 비트이므로 TCON = 0x10도 사용할 수 있다) */
    TR0 = 1;
```



```

/* EA, ET도 GC89T51.H에 1 비트 변수로 선언되었다.
EA와 ET는 IE 레지스터의 비트이므로 IE = 0x82로 사용할 수 있다. */

```

```

/*인터럽트를 사용하기 위하여 먼저 EA 비트를 1로 설정하고*/
EA = 1;

```

```

/* Timer/Counter 0의 오버플로우 인터럽트를 사용한다. */
ET0 = 1;

```

```

while(1)
{
    if(speed_up==7)
    {
        MOTOR_STOP();
        EA=0;
        break;
        // speed_up이 7이면 정지하고 while문을 탈출한다.
    }
    speed_go(speed_up);
    // 전진 속도 명령을 입력받아 전진한다.
}
}

```

```

/* Timer/Counter 0 함수이다 */

```

```

void TO_int(void) interrupt TF0_VECTOR
{

```

```

/* 오버플로우가 되면 TH0와 TL0가 0이 되므로 처음 값을 다시 입력시킨다 */
TH0 = 0x19;
TL0 = 0x13;
count++;           //시간 계산을 위해 변수를 1 증가시킨다.

```

```

/* 만약 0.004초마다 인터럽트 루틴을 250번 호출했다면 0.004*250 = 1초가 된다. */
if(count >=250)    // 1초마다 라인트레이서 전진 속도를 증가
{
    count=0;        // count 변수를 초기화시킨다.
    speed_up ++;    // speed_up을 1초에 1씩 증가시킨다.
    LEDdata=Font[speed_up]; // LED를 사용해서 speed_up 값을 출력한다.
}
}

```

# 단원 학습 정리

01. 마이크로프로세서로 로봇을 구동할 때는 무한 루프를 사용한다.
02. LED를 마이크로프로세서의 입출력 포트에 직접 연결하고 출력 값을 조정하여 켜거나 끌 수 있다.
03. 7-세그먼트도 마이크로프로세서의 입출력 포트에 직접 연결하고 출력 값을 조정하면 원하는 값을 나타낼 수 있다.
04. 모터를 제어하는 방법과 적외선 센서의 동작은 5단원 로봇 길 찾기에서 설명한다.
05. 라인트레이서에 설치된 푸시 스위치 SW1, SW2를 외부 인터럽트 입력으로 사용할 수 있다. 외부 인터럽트 입력이 들어오면 프로그램은 지금까지 하던 작업을 멈추고 인터럽트에 할당된 작업을 마치고 원래하던 작업으로 돌아간다.
06. 마이크로세서에서는 시간을 계산하기 위하여 내장된 타이머를 사용한다. 타이머의 역할은 프로그램에서 정해진 수만큼 크리스털에 의하여 생성되는 펄스의 수를 계수하는 것이다. 그 결과로 (펄스의 주기\*펄스의 수)만큼의 시간이 경과하게 된다. 이것은 마이크로프로세서를 전자기기의 제어에 사용할 때 시간 계산의 기초가 된다.

# 단원 종합 문제

01

라인트레이서의 P1포트를 이용하여 아래의 그림과 같은 패턴으로 LED가 구동되도록 프로그래밍하되 약 0.5초 간격으로 아래와 같은 패턴 변화가 무한정 반복되도록 하시오. 단, 최초의 LED는 모두 OFF 상태에 있는 것으로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ●    | ○    | ●    | ○    | ●    |

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ●    | ○    | ●    | ○    | ●    | ○    |

| 라인트레이서 핀 연결 부위 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| LED포트          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT1          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

02

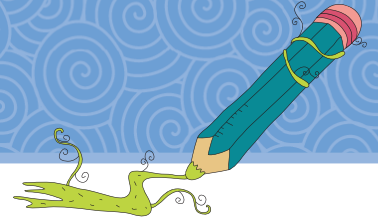
라인트레이서의 P2포트를 이용하여 아래의 그림과 같은 패턴으로 LED가 구동되도록 프로그래밍하되, 약 1초 간격으로 아래와 같은 패턴 변화가 무한정 반복되도록 하시오. 단, 최초의 LED는 모두 OFF 상태에 있는 것으로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ●    | ●    | ●    | ●    |

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ○    | ○    | ○    | ○    |

| 라인트레이서 핀 연결 부위 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| LED포트          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |



03

라인트레이서의 P1포트를 이용하여 아래의 그림과 같이 LED0부터 시작하여 LED7까지 약 0.5초 간격으로 시프트 ON 되는 프로그램을 작성하되, LED7까지의 시프트가 모두 이루어지면 LED0부터 시프트 동작을 다시 시작하도록 하고, 이와 같은 패턴이 무한정 반복되도록 하시오. 단, 최초의 LED는 모두 OFF 상태에 있는 것으로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

⤵ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

⤵ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⋮

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

⤵ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |

⇒ 다시 처음부터 시작한다.

| 라인트레이서 핀 연결 부위 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| LED포트          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT1          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

# 단원 종합 문제

04

라인트레이서의 SW1 스위치를 누르면 아래의 그림과 같은 패턴으로 LED가 동작하도록 하는 프로그램을 작성하되, LED 패턴의 변화 간격은 약 0.5초로 하시오. 단, 최초의 LED는 모두 OFF 상태에 있는 것으로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

SW1 스위치 누름



| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ○    | ○    | ○    | ○    |

↕ (약 0.5초 간격)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ●    | ●    | ●    | ●    |

라인트레이서 핀 연결 부위

|       |   |   |   |   |   |   |   |   |
|-------|---|---|---|---|---|---|---|---|
| LED포트 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

05

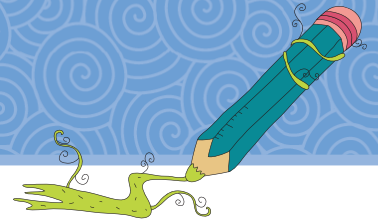
라인트레이서의 SW1과 LED를 이용한 비트 시프트 프로그램을 작성하되, 아래의 그림과 같이 SW1 스위치를 누를 때마다 LED0에서부터 LED7까지 1칸씩 우측 방향으로 시프트 ON 되는 프로그램을 작성하시오. 이때 SW1 스위치 누름에 따라 LED7까지의 시프트가 모두 이루어지면, 다시 SW1 스위치 누름에 따라 LED0부터 시프트가 이루어지도록 하시오. 단, 최초의 LED는 모두 OFF 상태에 있는 것으로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

SW1 스위치 누름





| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

SW1 스위치 누름



| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⋮

SW1 스위치 누름



| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

SW1 스위치 누름



| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |

⇒ 다시 처음부터 시작한다.

| 라인트레이서 핀 연결 부위 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| LED포트          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

# 단원 종합 문제

06

SW1, SW2스위치와 7-세그먼트를 이용한 업/다운 카운팅 프로그램을 작성하되, SW1 스위치를 누르면 7-세그먼트의 표시 숫자가 1씩 증가하고, SW2스위치를 누르면 7-세그먼트의 표시 숫자가 1씩 감소하도록 하는 프로그램을 작성하시오. 단, 7-세그먼트의 초기 값은 5이며, 0미만의 숫자와 9를 초과하지 않는 숫자 범위 내에서 구동 되도록 하시오.

SW1 스위치 누름 ⇨ 7-세그먼트 숫자가 1씩 증가되어짐(최대 9까지만 증가)

SW1 스위치 누름 ⇨ 7-세그먼트 숫자가 1씩 감소되어짐(최소 0까지만 감소)

| 라인트레이서 핀 연결 부위 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SEGMENT포트      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT2          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

07

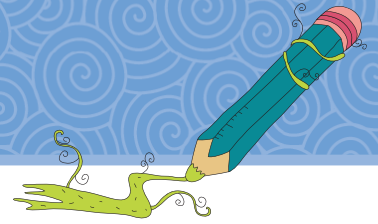
라인트레이서가 직선 방향으로 전진하도록 하는 프로그램을 작성하시오.

| 라인트레이서 핀 연결 부위 |                |                 |                |                 |
|----------------|----------------|-----------------|----------------|-----------------|
| MOTOR          | L_MOTOR_IN(왼쪽) | L_MOTOR_IN(오른쪽) | R_MOTOR_IN(왼쪽) | R_MOTOR_IN(오른쪽) |
| PORT3          | 6              | 7               | 4              | 3               |

08

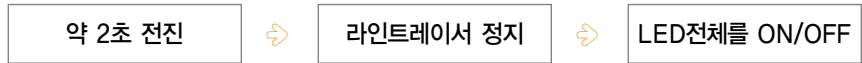
라인트레이서의 오른쪽 바퀴는 정 방향 회전, 왼쪽바퀴는 역 방향 회전하도록 하는 프로그램을 작성하시오.

| 라인트레이서 핀 연결 부위 |                |                 |                |                 |
|----------------|----------------|-----------------|----------------|-----------------|
| MOTOR          | L_MOTOR_IN(왼쪽) | L_MOTOR_IN(오른쪽) | R_MOTOR_IN(왼쪽) | R_MOTOR_IN(오른쪽) |
| PORT3          | 6              | 7               | 4              | 3               |



09

라인트레이서가 약 2초간 직선 방향의 전진 동작을 수행한 후 정지하며, LED전체를 1초 간격으로 계속 ON/OFF 동작하도록 하는 프로그램을 작성하시오.



| 라인트레이서 핀 연결 부위 |                |   |   |                 |   |                |   |                 |  |
|----------------|----------------|---|---|-----------------|---|----------------|---|-----------------|--|
| MOTOR          | L_MOTOR_IN(왼쪽) |   |   | L_MOTOR_IN(오른쪽) |   | R_MOTOR_IN(왼쪽) |   | R_MOTOR_IN(오른쪽) |  |
| PORT3          | 6              |   |   | 7               |   | 4              |   | 5               |  |
| SEGMENT포트      | 0              | 1 | 2 | 3               | 4 | 5              | 6 | 7               |  |
| PORT0          | 0              | 1 | 2 | 3               | 4 | 5              | 6 | 7               |  |

10

아래의 그림과 같이 LED0에서부터 LED7까지 0.5초 간격으로 우측 시프트한 후 LED7이 ON되면 라인트레이서가 직선 방향의 전진 동작을 수행하도록 하는 프로그램을 작성하시오. 단, 최초의 LED는 모두 OFF 상태인 것으로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

♡ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

♡ (0.5초)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

⋮

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

♡ (0.5초)



# 단원 종합 문제

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |



라인트레이서 전진

| 라인트레이서 핀 연결 부위 |                |   |                 |   |                |   |                 |   |
|----------------|----------------|---|-----------------|---|----------------|---|-----------------|---|
| LED포트          | L_MOTOR_IN(왼쪽) |   | L_MOTOR_IN(오른쪽) |   | R_MOTOR_IN(왼쪽) |   | R_MOTOR_IN(오른쪽) |   |
| PORT3          | 6              |   | 7               |   | 4              |   | 5               |   |
| SEGMENT포트      | 0              | 1 | 2               | 3 | 4              | 5 | 6               | 7 |
| PORT0          | 0              | 1 | 2               | 3 | 4              | 5 | 6               | 7 |

11

라인트레이서의 SW1 스위치를 누르면 라인트레이서가 전진하도록 하는 프로그램을 작성하시오.

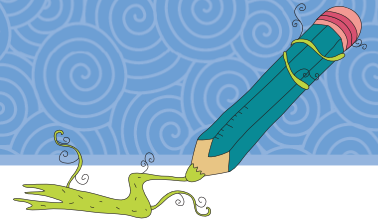
SW1 스위치 누름 라인트레이서 전진

| 라인트레이서 핀 연결 부위 |                |  |                 |  |
|----------------|----------------|--|-----------------|--|
| MOTOR          | L_MOTOR_IN(왼쪽) |  | R_MOTOR_IN(오른쪽) |  |
| PORT3          | 6              |  | 7               |  |
|                | 4              |  | 5               |  |

12

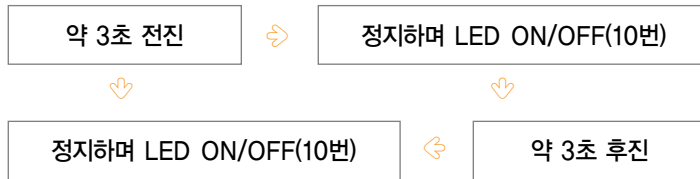
라인트레이서의 SW1 스위치를 누를 때마다 7-세그먼트의 표시 숫자가 0에서부터 9까지 1씩 증가되고, 이와 동시에 7-세그먼트 표시 숫자를 2진수로 변환하여 LED로 표시 되도록 하는 프로그램을 작성하시오. 단, 7-세그먼트의 초기 상태는 0이고, 9까지 증가되면 다시 0으로 초기화하시오.

| 라인트레이서 핀 연결 부위 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SEGMENT포트      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT3          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| LED포트          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT0          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |



13

라인트레이서가 약 3초간 전진 동작을 수행한 후 정지하면서 전체 LED를 10번 ON/OFF한 다음, 다시 약 3초간 후진한 후 전체 LED를 10번 ON/OFF 하도록 하는 프로그램을 작성하되 이러한 동작이 무한정 반복 되도록 하는 프로그램을 작성하시오.



| 라인트레이서 핀 연결 부위 |                |   |                 |   |                |   |                 |   |
|----------------|----------------|---|-----------------|---|----------------|---|-----------------|---|
| MOTOR          | L_MOTOR_IN(왼쪽) |   | L_MOTOR_IN(오른쪽) |   | R_MOTOR_IN(왼쪽) |   | R_MOTOR_IN(오른쪽) |   |
| PORT3          | 6              |   | 7               |   | 4              |   | 5               |   |
| LED포트          | 0              | 1 | 2               | 3 | 4              | 5 | 6               | 7 |
| PORT0          | 0              | 1 | 2               | 3 | 4              | 5 | 6               | 7 |

14

라인트레이서의 P3포트를 이용하여 적외선 발광센서 8개 모두를 ON시키고, P2포트를 통해 적외선 수광 센서 8개의 값을 모두 입력받아 이를 LED에 표시하는 프로그램을 작성하되, 아래의 표와 같이 P2.0과 연결된 수광 센서가 적외선을 감지하면 LED0를 ON 시키고, P2.1과 연결된 수광 센서가 적외선을 감지하면 LED1을 ON, 그리고 같은 방법으로 LED2 ~ LED7을 이용해 수광 센서의 센싱 값을 모두 표시하도록 하시오. 단, 적외선 수광 센서가 적외선을 감지 못하면 LED는 OFF상태로 하시오.

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| P2.0 감지 | P2.1 감지 | P2.2 감지 | P2.3 감지 | P2.4 감지 | P2.5 감지 | P2.6 감지 | P2.7감지  |
| LED0 ON | LED1 ON | LED2 ON | LED3 ON | LED4 ON | LED5 ON | LED6 ON | LED7 ON |

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SENSOR 포트      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT 2         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| SENSOR 포트      | E |   |   |   |   |   |   |   |
| PORT3          | 4 |   |   |   |   |   |   |   |
| LED 포트         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT0          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

# 단원 종합 문제

15

아래의 그림과 같이 라인트레이서의 구동 속도 제어 프로그램을 작성하되, 라인트레이서의 전진 속도를 0단계부터 7단계로 구분하여 약 1초 간격의 라인트레이서 속도 증가 프로그램을 작성하고, 각각의 속도 변화 단계에 따라 LED0부터 시작하여 LED7까지 약 1초 간격으로 시프트 ON 되는 프로그램을 작성하시오. 단, LED7이 ON 되면 라인트레이서를 정지시키도록 하시오. 단 최초의 라인트레이서는 정지 상태에 있는 것으로 하며 이때를 0단계로 한다.

LED : ●→켜짐(ON), ○→꺼짐(OFF)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ○    | ○    | ○    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 1

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ●    | ○    | ○    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 2

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ●    | ○    | ○    | ○    | ○    | ○    |

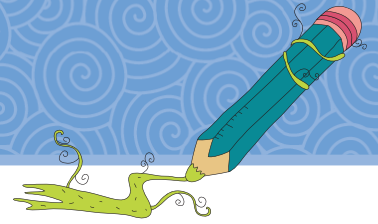
라인트레이서 전진 속도는 3

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ○    | ○    | ○    | ○    |

라인트레이서 전진 속도는 4

♡ (약 1초 후)



| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ●    | ○    | ○    | ○    |

라인트레이서 전진 속도는 5

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ●    | ○    | ○    |

라인트레이서 전진 속도는 6

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ●    | ○    |

라인트레이서 전진 속도는 7

♡ (약 1초 후)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ○    | ○    | ○    | ○    | ●    |

라인트레이서 정지

| 라인트레이서 핀 연결 부위 |                |   |                 |   |                |   |                 |   |
|----------------|----------------|---|-----------------|---|----------------|---|-----------------|---|
| LED포트          | 0              | 1 | 2               | 3 | 4              | 5 | 6               | 7 |
| PORT0          | 0              | 1 | 2               | 3 | 4              | 5 | 6               | 7 |
| MOTOR 포트       | L_MOTOR_IN(왼쪽) |   | L_MOTOR_IN(오른쪽) |   | R_MOTOR_IN(왼쪽) |   | R_MOTOR_IN(오른쪽) |   |
| PORT3          | 6              |   | 7               |   | 4              |   | 5               |   |

# 단원 종합 문제

16

라인트레이서의 P3포트를 이용하여 적외선 발광 센서 8개 모두를 ON시키고, P2포트를 이용하여 적외선 수광 센서의 값을 입력받되, 적외선 수광 센서들 중 Q3 혹은 Q4 센서가 ON 되면 라인트레이서가 전진하도록 하는 프로그램을 작성하시오. 단, 적외선 수광 센서들 중 Q3, Q4가 동시에 OFF 되면 라인트레이서를 정지하도록 하시오.

| 라인트레이서 핀 연결 부위 |                |                 |                |                 |   |   |   |   |
|----------------|----------------|-----------------|----------------|-----------------|---|---|---|---|
| SENSOR포트       | 0              | 1               | 2              | 3               | 4 | 5 | 6 | 7 |
| PORT2          | 0              | 1               | 2              | 3               | 4 | 5 | 6 | 7 |
| SENSOR포트       | E              |                 |                |                 |   |   |   |   |
| PORT3          | 4              |                 |                |                 |   |   |   |   |
| MOTOR          | L_MOTOR_IN(왼쪽) | L_MOTOR_IN(오른쪽) | R_MOTOR_IN(왼쪽) | R_MOTOR_IN(오른쪽) |   |   |   |   |
| PORT1          | 2              | 3               | 0              | 1               |   |   |   |   |

17

라인트레이서의 SW1, SW2 스위치를 이용한 적외선 센서 제어 프로그램을 작성하되, SW1 스위치를 누르면 적외선 발광 센서가 ON되고, SW2 스위치를 누르면 적외선 발광센서가 OFF 되는 프로그램을 작성하시오. 그리고 이때 P2포트를 이용하여 적외선 수광 센서의 값을 입력받아 이를 LED에 표시하되 아래의 표와 같이 수광 센서의 센싱 값을 모두 표시하도록 하시오. 단, 최초의 적외선 발광 센서는 OFF 상태이며, 적외선 수광 센서가 적외선을 감지 못하면 LED는 OFF 상태로 하시오.

|         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|
| P2.0 감지 | P2.1 감지 | P2.2 감지 | P2.3 감지 | P2.4 감지 | P2.5 감지 | P2.6 감지 | P2.7감지  |
| LED0 ON | LED1 ON | LED2 ON | LED3 ON | LED4 ON | LED5 ON | LED6 ON | LED7 ON |

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SENSOR 포트      | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT 2         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| SENSOR 포트      | E |   |   |   |   |   |   |   |
| PORT3          | 4 |   |   |   |   |   |   |   |
| LED 포트         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT0          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |







# ROBOT CONTROL SYSTEM

# V 로봇! 길을 찾다.

1. 모터를 구동해서 로봇 움직이기
2. LED를 사용해서 적외선 센서 값 확인
3. 갈림길을 감지하고 길 선택하기
4. 로봇의 미로 찾기

3장에서 GC89T51을 응용한 라인트레이서를 조립했다. 그리고 4장에서는 C 언어의 기초와 프로그램 실행 파일을 만들고, 로봇에 전송하여 구동시키는 방법을 학습하였다. 5장에서는 라인트레이서에서 가장 핵심적인 기능인 적외선 센서를 이용하여 길을 찾고 모터 구동 프로그램을 작성하는 방법을 학습한다.



# 01. 모터를 구동해서 로봇 움직이기

## ROBOT CONTROL SYSTEM

### 학습목표



- 모터를 구동해서 로봇을 움직이는 방법을 설명할 수 있다.
- 센서를 이용해서 로봇의 위치와 길을 찾는 방법을 설명할 수 있다.
- 로봇이 갈림길을 감지하고 갈 길을 선택하는 방법을 설명할 수 있다.
- 길 상황에 따라서 로봇이 대처하는 프로그램을 실습한다.
- 우회전, 좌회전, 직진, 상황을 조합하여 로봇이 미로에서 길 찾는 프로그램을 완성하고 실습한다.

라인트레이서에서 모터는 사람의 다리 역할을 한다. 이제부터 사람의 다리 역할을 하는 모터를 작동해서 로봇을 움직여 보자. 모터를 움직이기 위하여 우선 MPU 모듈의 헤더 커넥터와 모터 드라이버의 커넥터를 다음과 같이 연결한다.

[표 5-1] 모터 구동을 위한 라인트레이서의 연결

| 라인트레이서의 연결 방법 |                             |                             |                              |                              |
|---------------|-----------------------------|-----------------------------|------------------------------|------------------------------|
| MOTOR 포트      | L_MOTOR_IN (1포트)<br>Left_M1 | L_MOTOR_IN (2포트)<br>Left_M0 | R_MOTOR_IN (1포트)<br>Right_M1 | R_MOTOR_IN (2포트)<br>Right_M0 |
| PORT 3        | 4                           | 5                           | 6                            | 7                            |

모터의 회전은 드라이버 BA6208에 연결된 핀 2, 3에 연결된 두 제어신호에 의하여 결정된다. 제어신호 Left\_M0, Left\_M1의 조합에 의하여 왼쪽 모터는 다음과 같이 제어된다.



[표 5-2] 왼쪽 모터의 구동을 위한 제어 신호

| Left_M0 | Left_M1 | 동 작              |
|---------|---------|------------------|
| 1       | 0       | 왼쪽 모터 전진 방향으로 회전 |
| 0       | 1       | 왼쪽 모터 후진 방향으로 회전 |
| 1       | 1       | 왼쪽 모터 정지         |
| 0       | 0       | 왼쪽 모터 관성 운전      |

제어 신호 Right\_M0, Right\_M1의 조합에 의하여 오른쪽 모터는 다음과 같이 제어된다.

[표 5-3] 오른쪽 모터의 구동을 위한 제어 신호

| Right_M0 | Right_M1 | 동 작               |
|----------|----------|-------------------|
| 1        | 0        | 오른쪽 모터 전진 방향으로 회전 |
| 0        | 1        | 오른쪽 모터 후진 방향으로 회전 |
| 1        | 1        | 오른쪽 모터 정지         |
| 0        | 0        | 오른쪽 모터 관성 운전      |

이제 [표 5-2]의 제어 신호표를 이용하여 왼쪽 바퀴를 움직여 보자.



### 예 제 5-1

#### 왼쪽 바퀴만 전진하기

```
#include <GC89T51.H>
// 모터 제어 신호가 연결된 CPU 출력 포트를 매크로 상수로 선언한다.
#define Left_M0 P3_4
#define Left_M1 P3_5

void main(void)
{
    IOCFG |= 0x01; // P1 포트 사용을 위한 초기화

    for (;;) {
        Left_M0 = 1; // 약 1.5초 대기
        Left_M1 = 0;
    }
}
```

### 설명

```
#define Left_M0 P3_4
```

```
#define Left_M1 P3_5
```

왼쪽 모터를 구동을 제어하는 신호를 출력하는 MPU 포트를 매크로 상수로 정의하였다.

```
for (;) {.....}
```

중괄호 안의 실행문을 무한 반복하여 실행한다.

```
Left_M0 = 1; Left_M1 = 0;
```

이제 제어 신호표를 이용하여 왼쪽 모터를 (전진→관성 운전→정지→후진→관성 운전→정지)의 과정을 반복해 보자. 모터의 제어 신호를 모두 끊어도 얼마간 계속 움직이는데 이것을 관성 운전이라고 한다.



### 예제 5-2

#### 왼쪽 바퀴의 복합 구동

```
#include <GC89T51.H>
```

```
// 모터 제어신호가 연결된 CPU 출력 포트를 매크로 상수로 선언한다.
```

```
#define Left_M0 P3_4
```

```
#define Left_M1 P3_5
```

```
void main(void)
```

```
{
```

```
    long i;
```

```
    IOCFG |= 0x01; // P1 포트 사용을 위한 초기화
```

```
    for (;) {
```

```
        //왼쪽 바퀴 전진
```

```
        Left_M0 = 1; Left_M1 = 0;
```

```
        for (i=0; i<740,000; i++); // 약 1.5초 대기
```

```
        //왼쪽 바퀴 관성 운전
```

```
        Left_M0 = 0; Left_M1 = 0;
```

```
        for (i=0; i<740,000; i++); // 약 1.5초 대기
```

```
        //왼쪽 바퀴 정지
```

```
        Left_M0 = 1; Left_M1 = 1;
```

```
        for (i=0; i<740,000; i++); // 약 1.5초 대기
```

```
        //왼쪽 바퀴 후진
```



```

Left_M0 = 0;   Left_M1 = 1;
for (i=0 ; i<90000 ; i++); // 약 1.5초 대기
//왼쪽 바퀴 관성 운전
Left_M0 = 0;   Left_M1 = 0;
for (i=0 ; i<90000 ; i++); // 약 1.5초 대기
//왼쪽 바퀴 정지
Left_M0 = 1;   Left_M1 = 1;
for (i=0 ; i<90000 ; i++); // 약 1.5초 대기
}
}

```

이제 왼쪽 바퀴를 움직인 것 같이 오른쪽 바퀴를 (전진→관성 운전→정지→후진→관성운전→정지)의 과정을 반복해 보자.



### 예제 5-3

#### 오른쪽 바퀴의 복합 구동

```

#include <GC89T51.H>
// 모터 제어신호가 연결된 CPU 출력 포트를 매크로 상수로 선언한다.
#define Right_M0 P3_6
#define Right_M1 P3_7

void main(void)
{
    long i;

    IOCFG |= 0x01; // P1 포트를 사용하기 위한 초기화
    for ( ; ) {
        // 오른쪽 바퀴 전진
        Right_M0 = 1;   Right_M1 = 0;
        for (i=0 ; i<740,000 ; i++); // 약 1.5초 대기
        // 오른쪽 바퀴 관성 운전
        Right_M0 = 0;   Right_M1 = 0;
        for (i=0 ; i<740,000 ; i++); // 약 1.5초 대기
        // 오른쪽 바퀴 정지
        Right_M0 = 1;   Right_M1 = 1;
        for (i=0 ; i<740,000 ; i++); // 약 1.5초 대기
        // 오른쪽 바퀴 후진
    }
}

```

```

Right_M0 = 0;   Right_M1 = 1;
for (i=0 ; i<90000 ; i++); // 약 1.5초 대기
// 오른쪽 바퀴 관성 운전
Right_M0 = 0;   Right_M1 = 0;
for (i=0 ; i<90000 ; i++); // 약 1.5초 대기
// 오른쪽 바퀴 정지
Right_M0 = 1;   Right_M1 = 1;
for (i=0 ; i<90000 ; i++); // 약 1.5초 대기
}
}

```

지금까지는 오른쪽과 왼쪽 바퀴를 따로따로 구동했다. 이제부터는 왼쪽과 오른쪽 바퀴를 같이 움직여 로봇을 구동하는 프로그램을 작성하여 보자.



#### 예제 5-4 로봇의 전진, 후진

```

#include <GC89T51.H>

#define Left_M0   P3_4
#define Left_M1   P3_5
#define Right_M0  P3_6
#define Right_M1  P3_7

void main(void)
{
    long i;

    IOCFG |= 0x01; // P1 포트를 사용하기 위한 초기화
    for (;) {
        // 로봇 전진
        Left_M0 = 1; Left_M1 = 0; // 왼쪽 바퀴 전진
        Right_M0 = 1; Right_M1 = 0; // 오른쪽 바퀴 전진
        for (i = 0 ; i < 740,000 ; i++); // 약 1.5초 대기
        Left_M0 = 0; Left_M1 = 0; // 왼쪽 바퀴 관성 운전
        Right_M0 = 0; Right_M1 = 0; // 오른쪽 바퀴 관성 운전
        for (i = 0 ; i < 740,000 ; i++); // 약 1.5초 대기
        Left_M0 = 1; Left_M1 = 1; // 왼쪽 바퀴 정지
        Right_M0 = 1; Right_M1 = 1; // 오른쪽 바퀴 정지
        for (i = 0 ; i < 740,000 ; i++); // 약 1.5초 대기
    }
}

```



```

// 로봇 후진
Left_M0 = 0; Left_M1 = 1; // 왼쪽 바퀴 후진
Right_M0 = 0; Right_M1 = 1; // 오른쪽 바퀴 후진
for ( i = 0 ; i < 90000 ; i++); // 약 1.5초 대기
Left_M0 = 0; Left_M1 = 0; // 왼쪽 바퀴 관성 운전
Right_M0 = 0; Right_M1 = 0; // 오른쪽 바퀴 관성 운전
for ( i = 0 ; i < 90000 ; i++); // 약 1.5초 대기
Left_M0 = 1; Left_M1 = 1; // 왼쪽 바퀴 정지
Right_M0 = 1; Right_M1 = 1; // 오른쪽 바퀴 정지
for ( i = 0 ; i < 90000 ; i++); // 약 1.5초 대기
}
}

```

지금까지 로봇을 전진시키고 후진시키는 프로그램을 학습했다. 이제부터는 왼쪽 바퀴와 오른쪽 바퀴를 서로 반대 방향으로 구동하여 로봇이 제자리에서 회전하는 프로그램을 작성해 보여 보자.



### 예제 5-5 로봇의 회전

```

#include <GC89T51.H>

#define Left_M0 P3_4
#define Left_M1 P3_5
#define Right_M0 P3_6
#define Right_M1 P3_7

void main(void)
{
    long i;

    IOCFG |= 0x01; // P1 포트를 사용하기 위한 초기화
    for ( ; ) {
        Left_M0 = 1; Left_M1 = 0; // 왼쪽 바퀴 전진
        Right_M0 = 0; Right_M1 = 1; // 오른쪽 바퀴 후진
        for ( i = 0 ; i < 740,000 ; i++); // 약 1.5초 대기
        Left_M0 = 0; Left_M1 = 0; // 왼쪽 바퀴 관성 운전
        Right_M0 = 0; Right_M1 = 0; // 오른쪽 바퀴 관성 운전
    }
}

```





## 02. LED를 사용해서 적외선 센서 값 확인

### ROBOT CONTROL SYSTEM

#### 학습목표



- 센서를 이용해서 로봇의 위치와 길을 찾는 방법을 설명할 수 있다.
- 외부 환경 변화에 따른 센서감도를 조절할 수 있다.

로봇은 여러 가지 센서를 이용하여 위치와 장애물을 판단한다. 여기서는 적외선 센서로 선을 감지하여 로봇이 움직이는 방향을 선택하는 프로그램을 작성하자.

로봇의 눈 역할을 하는 적외선 센서는 발광 센서에서 빛을 방출하면 반사된 빛을 수광 센서가 인식해서 동작을 한다. 라인트레이서에 장착된 LED를 활용하여 적외선 센서가 읽어 들인 값을 눈으로 확인한다. 이를 위하여 라인트레이서를 다음과 같이 연결한다.

[표 5-4] 적외선 센서 값을 LED로 확인하기 위한 연결

| 라인트레이서 핀 연결 방법 |   |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
| SENSOR포트       | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT 2         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| SENSOR포트       | E |   |   |   |   |   |   |   |
| PORT 1         | 6 |   |   |   |   |   |   |   |
| LED포트          | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| PORT 0         | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

적외선 센서 0이 감지되면 LED 0이 켜지고, 적외선 센서 1이 감지되면 LED 1이 켜지는 방식으로 센서에 감지된 결과가 LED에 나타나는 프로그램을 작성하자.



### 예제 5-6

#### 적외선 센서를 읽어 로봇 LED에 표시하기



```
#include <GC89T51.H>

#define SENSOR    P2    // 적외선 센서와 연결된 포트를 P2로 정의.
#define LEDdata   P0    // LED와 연결된 포트를 P0으로 정의.
#define SENSOR_EN P1_6  // 적외선 발광센서 켜고 끄는 신호선으로 지정.

void main(void)
{
    IOCFG |= 0x01; // P1 포트 사용을 위한 초기화
    SENSOR_EN=1;   // 적외선 발광센서를 켜준다.
    while(1)
    {
        LEDdata=~SENSOR; // 적외선 센서 회로 출력을 반전시켜서 LED로 전송.
    }
}
```

[예제 5-6]의 동작을 확인하려면 검정색 바닥에 흰색 종이를 로봇 아래 두고 움직이면서 하면 편리하다. 적외선 센서의 감도가 좋지 않으면 가변 저항을 돌려서 감도를 향상시킬 수 있다. 적외선 센서에서 출력단에는 인버터가 있어서 빛을 받으면 출력전압이 낮아지고 빛을 받지 못하면 출력 전압이 높아진다. 그래서 적외선 센서가 빛을 받은 비트에 해당하는 LED가 켜지게 하려면 출력을 반전시켜 LED에 전달해야 한다.

라인트레이서가 바닥에 그려진 길을 정확히 찾으려면 흰색과 검정색이 반사하는 빛의 차이를 정확하게 판단해야 한다. 그렇게 하려면 다음의 방법으로 적외선 센서의 감도를 조정해야 한다.



### 라인트레이서의 적외선 센서 감도 조정 방법

- 1 리셋 스위치를 누르면서 검정색 바닥위에 센서를 위치시킨다.
- 2 드라이버를 이용해서 가변저항을 좌우로 돌려 LED에 불이 전부 켜지지 않도록 조정한다.
- 3 흰색 바닥 위에 센서를 위치시킨다.
- 4 드라이버를 이용해서 가변저항을 좌우로 돌려 LED가 전부 켜지도록 조정한다.
- 5 적외선 센서가 위로 약간 올라가게 라인트레이서의 뒷부분을 약간 누른다.
- 6 적외선 센서가 위로 약간 올라간 상태로 가변저항을 좌우로 돌려 LED에 불이 전부 켜지도록 조정한다.



.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

# 03. 갈림길을 감지하고 길 선택하기

## ROBOT CONTROL SYSTEM

### 학습목표



- 로봇이 갈림길을 감지하고 갈 길을 선택하는 방법을 설명할 수 있다.
- 길 상황에 따라서 로봇이 대처하는 프로그램을 실습한다.

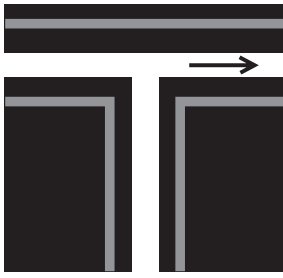
이제부터는 라인트레이서가 적외선 센서 값을 이용하여 길을 감지하고 모터를 구동해서 길을 찾아가는 프로그램을 작성해 보자.

로봇이 길을 올바르게 찾는지 확인하려면 간단한 미로판이 필요하다. 이때 검정색 밑판은 무광택이어야 하면 흰색선은 광택이어야 한다. 적외선 센서는 빛을 반사하는 흰색과 반사하지 않는 검정색을 비교해서 길을 찾으므로, 미로를 제작할 때 꼭 지켜야 한다. 로봇이 만나는 갈림길의 형태를 파악해서 로봇이 갈림길에서 우회전, 좌회전, 직진을 할 수 있도록 프로그램해 보자

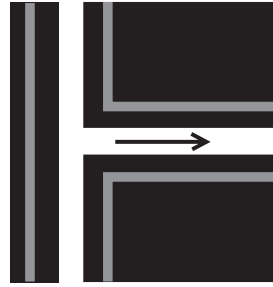


### 1. 로봇의 갈림길에서 우회전

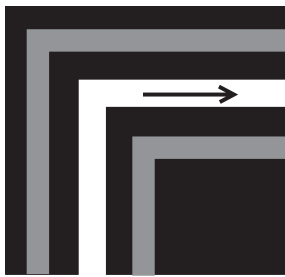
로봇이 오른쪽으로 회전하는 경우는 다음 그림들과 같은 4가지이다.



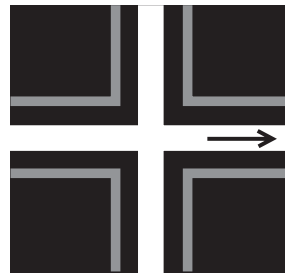
[그림 5-1] T갈림 길에서 우회전



[그림 5-2] T갈림 길에서 우회전



[그림 5-3] ㄱ갈림 길에서 우회전



[그림 5-4] +갈림 길에서 우회전

각각의 회전하는 경우가 발생 할 때 센서의 동작은 다음과 같다.

⇒ ㄱ 갈림 길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ●    | ●    | ●    | ●    |

⇒ T갈림 길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |

⇒ T갈림 길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ●    | ●    | ●    | ●    |





## 실습 5-1

앞에서 설명한 4가지의 경우처럼 갈림길에서 오른쪽으로 회전하는 프로그램을 작성해 보자.

```
#include <GC89T51.H>

#define Left_M0      P3_4
#define Left_M1      P3_5
#define Right_M0     P3_6
#define Right_M1     P3_7
#define SENSOR       P2           // 센서 포트
#define LEDdata      P0           // LED 포트
#define SENSOR_EN    P1_6        // 적외선 센서 켜고 끄기

//   제어보드 위에 적외선 센서의 비트별 위치
//       좌측 0 1 2 3 4 5 6 7   우측

//   2진수 표시와 반대로 제어보드에서는 좌측 비트가 LSB이다
//   적외선 센서가 직진해야 하는 상태일 때
//       센서 상태의 2진수 표시
#define CSensor      0xE7 // 1110 0111
#define CSensor1     0xEF // 1110 1111
#define RCSensor     0xE3 // 1110 0011
#define RCSensor1    0xCF // 1100 1111
#define LCSensor     0xC7 // 1100 0111
#define LCSensor1    0xF3 // 1111 0011

//   로봇이 오른쪽으로 조금 치우침
#define LCSensor2     0xEB // 1110 1011
#define LCSensor3     0xFB // 1111 1011
#define LCSensor4     0xE9 // 1110 1001

//   로봇이 왼쪽으로 조금 치우침
#define RCSensor2     0xD7 // 1101 0111
#define RCSensor3     0xDF // 1101 1111
```

```

#define RCSensor4      0x8F    // 1000 1111

// 우회전 갈림길 발견      ㄱ, ㅏ, ㅓ, ㅕ
#define RSensor        0x3F    // 0011 1111
#define RSensor1      0x7F    // 0111 1111
#define RSensor2      0x77    // 0111 0111
#define RSensor3      0x83    // 1000 0011
#define RSensor4      0x17    // 0001 0111
#define RSensor5      0x9F    // 1001 1111
#define RSensor6      0x03    // 0000 0011

// ㅏ 또는 T자 교차로 발견
#define CROSS          0x00    // 0000 0000
#define CROSS1        0x3C    // 0011 1100
#define CROSS2        0xBD    // 1011 1101
#define CROSS3        0x18    // 0001 1000
#define CROSS4        0x08    // 0000 1000
#define CROSS5        0x81    // 1000 0001
#define CROSS6        0xC3    // 1100 0011

void run(void);
void turn_right90(void);

unsigned char ReadData;      // 적외선 센서가 인지한 값
unsigned char speed;
unsigned char smooth_speed;
unsigned char turn_delay;
unsigned char stop_delay;

// 루프를 사용하는 시간 지연 함수
void delay(unsigned char num) {
    unsigned char j;
    for(j=0; j<num; j++);
}

```





// 적외선 센서의 상태를 조회해서 리턴함

```
unsigned char SensorScan() {
    unsigned char read_data, scan_data;

    scan_data = 0x00;
    read_data = P2;
    delay(1);
    read_data = read_data & P2;          // 센서값 받아 오기

    scan_data = 0x3F | read_data;       // 0011 1111
    if (scan_data == 0x3F) {           // 우회전 갈림길 상태 리턴
        return 0x3F;
    }
    return P2;                          // 오른쪽을 제외한 다른 값 리턴
}
```

// 정지

```
void move_stop(unsigned char stop_num) {
    Left_M0 = 1;           Right_M0 = 1; // 왼쪽 바퀴 정지
    Left_M1 = 1;           Right_M1 = 1; // 오른쪽 바퀴 정지
    delay(stop_num);
}
```

// 직진

```
void forward(unsigned char forward_num) {
    Left_M0 = 1;           Right_M0 = 1; // 왼쪽 바퀴 직진
    Left_M1 = 0;           Right_M1 = 0; // 오른쪽 바퀴 직진
    delay(forward_num);
    move_stop(stop_delay);
}
```

// 왼쪽으로 조금 방향 수정

```

void turn_left(unsigned char turn_left_num) {
    // (Smooth Turn Left Moving)
    Left_M0 = 1;           Left_M1 = 1;    // 왼쪽 바퀴 정지
    Right_M0 = 1;          Right_M1 = 0;   // 오른쪽 바퀴 직진
    delay(turn_left_num);
    move_stop(stop_delay);
}

// 오른쪽으로 조금 방향 수정
/

void turn_right(unsigned char turn_right_num) {
    // (Smooth Turn Right Moving)
    Right_M0 = 1;          Right_M1 = 1;   // 오른쪽 바퀴 정지
    Left_M0 = 1;           Left_M1 = 0;    // 왼쪽 바퀴 직진
    delay(turn_right_num);
    move_stop(stop_delay);
}

// 90도 우회전

void turn_right90(void) {
    unsigned int num1, num2;

    Left_M0 = 1;Right_M0 = 1;              // 왼쪽 바퀴 전진
    Left_M1 = 0;Right_M1 = 0;              // 오른쪽 바퀴 전진
    for(num1=0; num1<200; num1++)
        for(num2=0; num2<115; num2++);
    move_stop(stop_delay);
    move_stop(stop_delay);

    while(!((SensorScan()&0x18)==0x18)) { // 중앙 센서가 반응할 때까지
        Left_M0 = 1;Right_M0 = 0;         // 왼쪽 바퀴 전진
        Left_M1 = 0;Right_M1 = 1;         // 오른쪽 바퀴 후진
        delay(turn_delay);
        move_stop(stop_delay);
    }
}

```



```
while(!(SensorScan()==CSensor)) { // 중앙 센서만 반응할 때까지
    Left_M0 = 1;Right_M0 = 0; // 왼쪽 바퀴 전진
    Left_M1 = 0;Right_M1 = 1; // 오른쪽 바퀴 후진
    delay(smooth_speed);
    move_stop(stop_delay);
}
}

// 초기화
void init(void) {
    IOCFG |= 0x01; // P1 포트 사용을 위한 초기화
    P0 = 0xFF;    P1 = 0xFF;
    P2 = 0xFF;    P3 = 0xFF;

    ReadData  = 0; // 센서값 저장
    speed      = 28; // 전진 진행 속도
    stop_delay = 4; // 멈춤 속도
    turn_delay = 18; // 좌,우 90°회전할 때
    smooth_speed = 18; // 좌,우 조금씩 이동할 때
}

// 주함수
void main(void) {
    init();

    SENSOR_EN = 1; // SENSOR Enable

    while(1) {
        ReadData = SensorScan(); // 센서값 Read
        LEDdata = ReadData; // LED에 센서값 표시
        run(); // 센서값에 따라 이동 지시
    }
} // Motor Move End
```

```

void run(void) {
    // LED On is Sensor action
    // Low Signal : LED ON,   White Line   reflection
    // High Signal : LED OFF, Black Line   Not reflection
    //                               0 1 2 3 4 5 6 7
    // LSB   0000 0000           MSB
    switch(ReadData) {
        // 로봇이 중앙에 있는 경우, 직진
        // 오른쪽 바퀴 : 전진,           왼쪽 바퀴 : 전진
        case CSensor:
        case CSensor1:
        case RCSensor:
        case LCSensor:
        case RCSensor1:
        case LCSensor1:
            forward(speed);
            break;

        // 로봇이 오른쪽으로 치우쳐져 있는 경우,
        // 왼쪽으로 조금이동
        // 오른쪽 바퀴 : 전진, 왼쪽 바퀴 : 후진
        case RCSensor2:
        case RCSensor3:
        case RCSensor4:
            turn_right(smooth_speed-5);
            break;

        // 로봇이 왼쪽으로 치우쳐져 있는 경우,
        // 오른쪽으로 조금 이동
        // 오른쪽 바퀴 : 후진, 왼쪽 바퀴 : 전진
        case LCSensor2:
        case LCSensor3:
        case LCSensor4:
            turn_left(smooth_speed-5);
            break;
    }
}

```



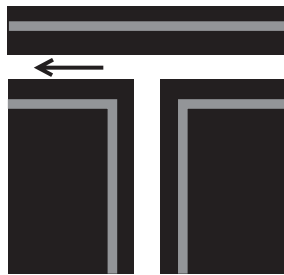
```
// 갈림길에서 오른쪽 센서가 반응한 경우, 빠르게 회전
// 오른쪽 바퀴 : 후진, 왼쪽 바퀴 : 전진
case RSensor:           // 3F, 0011 1111
case RSensor1:         // 7F, 0111 1111
case RSensor2:         // 77, 0111 0111
case RSensor3:         // 83, 1000 0011
case RSensor4:         // 17, 0001 0111
case RSensor5:         // 9F, 1001 1111
case RSensor6:         // 03, 0000 0011
    turn_right90();
    break;

// 십자 교차로
// 전진
case CROSS:
case CROSS1:
case CROSS2:
case CROSS3:
case CROSS4:
case CROSS5:
    turn_right90();
    break;

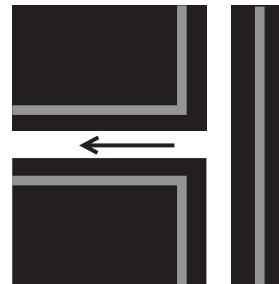
// 기타 상황에 대해서는 모두 전진
// 오른쪽 바퀴 : 전진, 왼쪽 바퀴 : 후진
default:
    forward(speed);
    forward(speed);
}
}
```

## 2. 로봇의 갈림길에서 좌회전

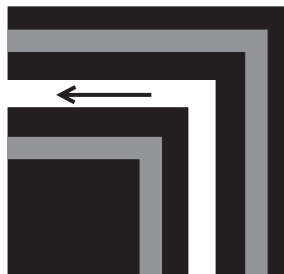
로봇이 좌회전하는 경우도 우회전과 같이 다음 그림과 같이 4가지다. 센서의 동작 상태는 우회전하는 경우와 반대로 동작한다. T자 갈림길과 +갈림길에서는 똑같이 동작을 하지만, 이때는 방향을 임의로 선택하도록 프로그램하면 된다.



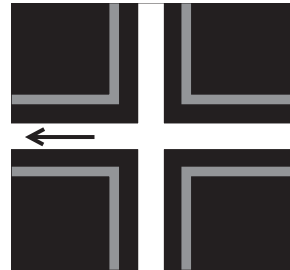
[그림 5-6] T갈림 길에서 좌회전



[그림 5-7] L갈림 길에서 좌회전



[그림 5-8] L갈림 길에서 좌회전



[그림 5-9] +갈림 길에서 좌회전

각각의 좌회전하는 경우가 발생할 때 센서의 동작을 보면 다음과 같다.

⇒ L갈림 길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ○    | ○    | ○    |





## 실습 5-2

이번 실험에서는 앞에서 설명한 4가지의 경우처럼 갈림길에서 왼쪽으로 회전하는 프로그램을 작성하자. 간단하게 왼쪽으로 회전하는 함수만 추가하면 된다.

```
// 90도 좌회전
//
void turn_left90(void) {
    unsigned int num1, num2;

    Left_M0 = 1; Right_M0 = 1;        // 왼쪽 바퀴 전진
    Left_M1 = 0; Right_M1 = 0;       // 오른쪽 바퀴 전진
    for(num1=0; num1<200; num1++)
        for(num2=0; num2<115; num2++);
    move_stop(stop_delay);
    move_stop(stop_delay);

    while(!((SensorScan()&0x18)==0x18)) { // 중앙 센서가 반응할 때까지
        Left_M0 = 0; Right_M0 = 1;    // 왼쪽 바퀴 후진
        Left_M1 = 1; Right_M1 = 0;    // 오른쪽 바퀴 전진
        delay(turn_delay);
        move_stop(stop_delay);
    }

    while(!(SensorScan()==CSensor)) { // 중앙 센서만 반응할 때까지
        Left_M0 = 0; Right_M0 = 1;    // 왼쪽 바퀴 후진
        Left_M1 = 1; Right_M1 = 0;    // 오른쪽 바퀴 전진
        delay(smooth_speed);
        move_stop(stop_delay);
    }
}
```

이때의 센서 감지 문은,

```
while(!(SensorScan()==CSensor)) { // 중앙 센서만 반응할 때까지
    Left_M0 = 0; Right_M0 = 1; // 왼쪽 바퀴 후진
    Left_M1 = 1; Right_M1 = 0; // 오른쪽 바퀴 전진
    delay(smooth_speed);
    move_stop(stop_delay);
}
```

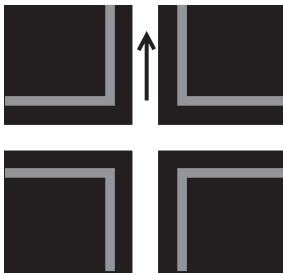
가 된다.



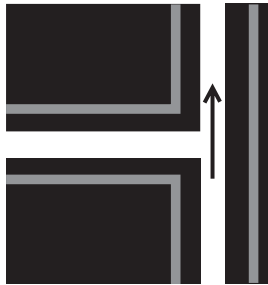


### 3. 로봇의 갈림길에서 직진

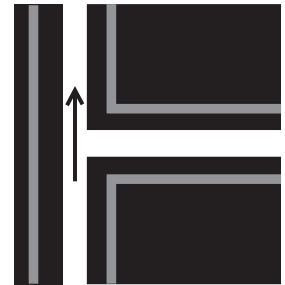
갈림길에서 로봇을 오른쪽 또는 왼쪽으로 회전하는 프로그램 작성하는 것을 실습을 통해 학습했다. 이제 갈림길에서 로봇이 직진을 하는 경우를 알아보자. 갈림길에서 직진하는 경우는 다음의 3가지다.



[그림 5-11] + 갈림길에서 직진



[그림 5-12] - 갈림길에서 직진



[그림 5-13] † 갈림길에서 직진

세 가지의 갈림 길에서 센서의 동작 상태를 보면 다음과 같다.

⇒ † 갈림길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ○    | ○    | ○    | ●    | ●    | ●    | ●    | ●    |

⇒ - 갈림길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ○    | ○    | ○    |

⇒ + 갈림길에서 오른쪽으로 회전하는 경우(꺼짐 ○, 켜짐 ●)

| LED0 | LED1 | LED2 | LED3 | LED4 | LED5 | LED6 | LED7 |
|------|------|------|------|------|------|------|------|
| ●    | ●    | ●    | ●    | ●    | ●    | ●    | ●    |



### 실습 5-3

갈림길에서 직진하는 프로그램을 작성하자. 일정 시간(감지된 길을 빠져 나갈 정도) 동안 직진하도록 하자.

```

// 직진
void forward(unsigned char forward_num) {
    Left_M0 = 1;           Right_M0 = 1;    // 왼쪽 바퀴 직진
    Left_M1 = 0;           Right_M1 = 0;    // 오른쪽 바퀴 직진
    delay(forward_num);
    move_stop(stop_delay);
}

```



### 실습 5-4

로봇이 길을 따라 이동하다가 길이 없을 경우 U-턴하는 함수를 만들어 보자.

```

// 라인이 없는 경우 U-Turn 시도
//
void uturn(void) {
    unsigned int num1, num2;
    Left_M0 = 1; Right_M0 = 1;           // 왼쪽 바퀴 정지
    Left_M1 = 1; Right_M1 = 1;           // 오른쪽 바퀴 정지
    for(num1=0; num1<500; num1++)
        for(num2=0; num2<400; num2++);

    while(!(SensorScan()==CSensor)) { // 중앙 센서만 반응할 때까지
        Left_M0 = 0; Right_M0 = 1; // 왼쪽바퀴 후진
        Left_M1 = 1; Right_M1 = 0; // 오른쪽바퀴 전진
        delay(turn_delay);
        move_stop(stop_delay);
    }
}

```

이때의 센서 감지 문은,

```

// 라인이 없는 경우
// U 턴 하는 경우
case END_LINE:
    uturn();
    break;

```



가 된다.



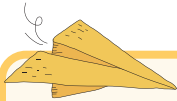
### 실습 5-5

로봇이 길을 따라 이동하다가 정지하는 경우를 만들어 보자.

```

// 정지
//
void move_stop(unsigned char stop_num) {
    unsigned char snum;
    for(snum=0; snum<(stop_num/2); snum++) {
        P3 = 0x00; // 왼쪽과 오른쪽 바퀴 모두 관성운동
    }
    for(snum=0; snum<(stop_num/2); snum++) {
        P3 = 0xF0; // 왼쪽과 오른쪽 바퀴 모두 정지
    }
}

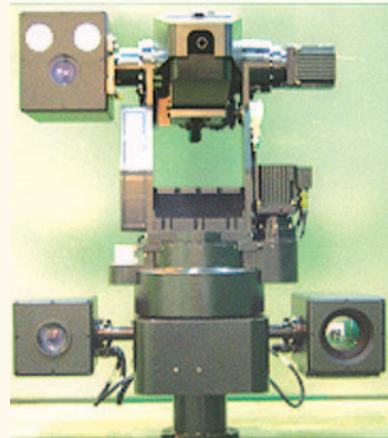
```



### 쉬어가기

#### 휴전선 감시로봇 2007년 배치

야간에도 물체 자동 탐지 -국내 연구진이 사람을 대신해 침입자를 찾아내고 제압까지 할 수 있는 지능형 감시 경계 로봇을 개발했다. 이 로봇은 전방 휴전선이나 인천국제공항 등에도 배치될 전망이다. 낮에는 4Km, 밤에는 2Km 거리에 있는 움직이는 물체를 자동으로 탐지할 수 있는 능력을 갖춘 게 특징이다. 특히 주간 2Km 야간 1Km 거리 내에서는 사람, 나무, 차량 등을 구분할 수 있다. 10m 이내 근거리에서는 군대에서 사용하는 암구호 등을 통해 피아 식별을 할 수 있고 기관총을 탑재해 적을 사살하거나 제압하는 것도 가능하다.



# 04. 로봇의 미로 길 찾기

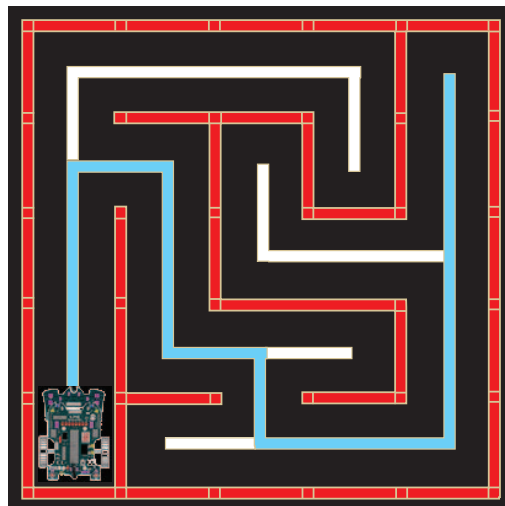
## ROBOT CONTROL SYSTEM

### 학습목표



- 우회전, 좌회전, 직진, 상황 대처를 조합하여 로봇이 미로에서 길 찾는 프로그램을 완성하고 실습한다.

여기서는 갈림길에서 센서 상태를 판단하여 미로에서 길을 찾아 로봇이 주행하는 프로그램을 작성하자.



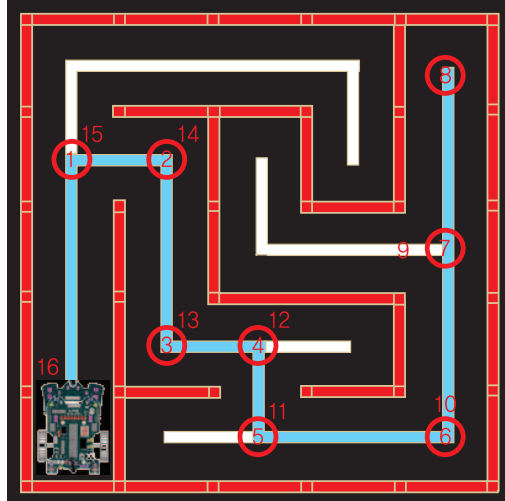
Starting Point

[그림 5-14] 5x5 미로 판에서 로봇이 따라갈 길

로봇이 [그림 5-14]의 5x5 미로 판에서 파란색으로 주어진 경로를 따라



이동해서 다시 출발점까지 돌아오는 과정에서 15번의 갈림길을 만난다.



Starting Point

[그림 5-15] 로봇이 길을 따라가다 만나는 갈림길

적외선 센서의 상태에 따라 갈림길이 나온 횟수를 고려하여 갈림길을 판단하여 로봇이 가야할 방향을 프로그램에서 지정을 해주면서 지정된 경로를 따라 이동하는 프로그램을 작성해 보자.



### 실습 5-6

[그림 5-12]처럼 경로를 설정하고 로봇이 경로를 따라 이동해서 다시 출발점으로 돌아오는 프로그램을 작성해 보자.

```
#include <GC89T51.H>

#define Left_M0      P3_4
#define Left_M1      P3_5
#define Right_M0     P3_6
#define Right_M1     P3_7
#define SENSOR       P2           // 센서 포트
```

```

#define LEDdata      P0      // LED 포트
#define SENSOR_EN    P16     // 적외선 센서 켜고 끄기

// 제어 보드 위에 적외선 센서의 비트별 위치
// 좌측 0 1 2 3 4 5 6 7 우측
//
// 2진수 표시와 반대로 제어 보드의 좌측 비트가 LSB이다

// 적외선 센서가 직진해야 하는 상태일 때
// 센서 상태의 2진수 표시
#define CSensor      0xE7    // 1110 0111
#define CSensor1     0xEF    // 1110 1111
#define RCSensor     0xE3    // 1110 0011
#define RCSensor1    0xCF    // 1100 1111
#define LCSensor     0xC7    // 1100 0111
#define LCSensor1    0xF3    // 1111 0011

// 로봇이 오른쪽으로 조금 치우침
#define LCSensor2    0xEB    // 1110 1011
#define LCSensor3    0xFB    // 1111 1011
#define LCSensor4    0xE9    // 1110 1001

// 로봇이 왼쪽으로 조금 치우침
#define RCSensor2    0xD7    // 1101 0111
#define RCSensor3    0xDF    // 1101 1111
#define RCSensor4    0x8F    // 1000 1111

// 좌회전 갈림길 발견
#define LSensor      0xFC    // 1111 1100
#define LSensor1     0xFE    // 1111 1110
#define LSensor2     0xF6    // 1111 0110
#define LSensor3     0xC1    // 1100 0001
#define LSensor4     0xE8    // 1110 1000
#define LSensor5     0xF9    // 1111 1001
#define LSensor6     0xC0    // 1100 0000

```



```
// 우회전 갈림길 발견          ㄱ, ㄴ, ㄷ, ㄹ
#define RSensor          0x3F // 0011 1111
#define RSensor1        0x7F // 0111 1111
#define RSensor2        0x77 // 0111 0111
#define RSensor3        0x83 // 1000 0011
#define RSensor4        0x17 // 0001 0111
#define RSensor5        0x9F // 1001 1111
#define RSensor6        0x03 // 0000 0011

// ㅊ 또는 T자 교차로 발견
#define CROSS          0x00 // 0000 0000
#define CROSS1        0x3C // 0011 1100
#define CROSS2        0xBD // 1011 1101
#define CROSS3        0x18 // 0001 1000
#define CROSS4        0x08 // 0000 1000
#define CROSS5        0x81 // 1000 0001
#define CROSS6        0xC3 // 1100 0011

/// 길이 없음, U-턴 길 발견
#define END_LINE      0xFF // 1111 1111

void run(void);
void turn_right90(void);
void turn_left90(void);
void turn_cross_right90(void);
void turn_cross_left90(void);
void right_forward(void);

unsigned char ReadData;          // 적외선 센서가 인지한 값
unsigned char speed;
unsigned char smooth_speed;
unsigned char turn_delay;
unsigned char stop_delay;
```

```

unsigned char flag;                // 몇 번째 갈림길인지를 기억함
unsigned char delay_num;
unsigned char uturn_chk;

// 루프를 사용하는 시간 지연 함수
void delay(unsigned char num) {
    unsigned char j;
    for(j=0; j<num; j++) ;
}

// 적외선 센서의 상태를 조회해서 리턴함
//
unsigned char SensorScan() {
    unsigned char read_data, scan_data;

    scan_data = 0x00;
    read_data = P2;
    delay(1);
    read_data = read_data & P2;

    scan_data = 0x3C | read_data;
    if (scan_data == 0x3C) return 0x3C;        // 십자로 리턴

    scan_data = 0x00;
    scan_data = 0xFC | read_data;
    if (scan_data == 0xFC) {                  // 좌회전 갈림길 리턴
        return 0xFC;
    }

    scan_data = 0x00;
    scan_data = 0x3F | read_data;
    if (scan_data == 0x3F) {                  // 우회전 갈림길 리턴
        return 0x3F;
    }
    return P2;
}

```





```
}

// 정지
//
void move_stop(unsigned char stop_num) {
    unsigned char snum;
    for(snum=0; snum<(stop_num/2); snum++) {
        P3 = 0x00;      // 왼쪽과 오른쪽 바퀴 모두 관성 운전
    }
    for(snum=0; snum<(stop_num/2); snum++) {
        P3 = 0xF0;      // 왼쪽과 오른쪽 바퀴 모두 정지
    }
}

// 전진
//
void forward(unsigned char forward_num) {
    if ((P2 | 0x3F) == 0x3F) {      // 우회전 갈림길 발견
        switch(flag) {
            case 0x01:
            case 0x04:
            case 0x0A:
                turn_right90();
                break;
        }
    } else if ((P2 | 0xFC) == 0xFC) {      // 좌회전 갈림길 발견
        switch(flag) {
            case 0x05:
            case 0x0B:
            case 0x0D:
                turn_left90();
                break;
        }
    } else {      // 직진 상황
```

```

        P3 = 0x50;           // 왼쪽과 오른쪽 바퀴 모두 전진
        delay(forward_num);
        move_stop(stop_delay);
    }
}

// ┆ 형태의 갈림길에서 직진
void right_forward(void) {
    unsigned int j;
    SENSOR_EN = 0;
    for (j=0; j<500; j++) {
        P3 = 0x50;           // 왼쪽과 오른쪽 바퀴 모두 전진
        delay(speed);
        move_stop(stop_delay);
    }
    flag++;
    SENSOR_EN = 1;
}

// 왼쪽으로 조금 방향 수정
//
void turn_left(unsigned char turn_left_num) {
    P3 = 0x40;           // 왼쪽 바퀴 관성 운전, 오른쪽 바퀴 전진
    delay(turn_left_num);
    move_stop(stop_delay);
}

// 오른쪽으로 조금 방향 수정
//
void turn_right(unsigned char turn_right_num) {
    P3 = 0x10;           // 오른쪽 바퀴 관성 운전과 왼쪽 바퀴 전진
    delay(turn_right_num);
    move_stop(stop_delay);
}

```



```
// 90도 우회전
//
void turn_right90(void) {
    unsigned int num1, num2;

    if (flag == 0x07) {          // ┆ 형태의 갈림길 (직진 선택)
        right_forward();
        return;
    }

    else if (flag == 0x05) {     // T형태 갈림길 (좌회전 선택)
        turn_left90();
        return;
    }

    else if (flag == 0x0D) {    // T 형태 갈림길 (좌회전 선택)
        turn_left90();
        return;
    }

    else if (flag == 0x0A) {    // T 형태 갈림길 (좌회전 선택)
        turn_left90();
        return;
    }

    else {

        P3 = 0x50;              // 왼쪽과 오른쪽 바퀴 모두 전진
        for(num1=0; num1<200; num1++)
            for(num2=0; num2<delay_num; num2++);
        move_stop(stop_delay);
        move_stop(stop_delay);

        P3 = 0x90;              // 왼쪽 바퀴 전진, 오른쪽 바퀴 후진
        delay(smooth_speed);
        move_stop(stop_delay);
    }
}
```

```

        while(!(SensorScan()==CSensor)) { // 중앙 센서가 반응할 때까지
            P3 = 0x90;// 왼쪽 바퀴 전진, 오른쪽 바퀴 후진
            delay(smooth_speed);
            move_stop(stop_delay);

        }
        flag++;
    }
}

// 90도 좌회전
//
void turn_left90(void) {
    unsigned int num1, num2;

    if (flag == 0x07) {
        forward(speed);
        return;
    } else {
        P3 = 0x50; // 왼쪽과 오른쪽 바퀴 모두 전진
        for(num1=0; num1<200; num1++)
            for(num2=0; num2<delay_num; num2++);
        move_stop(stop_delay);

        P3 = 0x60; // 왼쪽 바퀴에 후진, 오른쪽 바퀴 전진
        delay(turn_delay);

        while(!(SensorScan()==CSensor)) { // 중앙 센서 반응할 때까지
            P3 = 0x60; // 왼쪽 바퀴 후진, 오른쪽 바퀴에 전진
            delay(smooth_speed);
            move_stop(stop_delay);

        }
        flag++;
    }
}
}

```



```
// 조금 전진하고 우측으로 회전
// 중앙 센서 인식 상태 확인
//
void uturn(void) {
    unsigned int num1, num2;

    P3 = 0x50;      // 왼쪽과 오른쪽 바퀴 모두 전진
    for(num1=0; num1<350; num1++)
        for(num2=0; num2<delay_num+7; num2++);
    move_stop(stop_delay);
    move_stop(stop_delay);

    while(!(SensorScan()==CSensor)) {      // 중앙 센서의 반응 확인
        P3 = 0x60;      // 왼쪽 바퀴 후진, 오른쪽 바퀴 전진
        delay(turn_delay+7);
        move_stop(stop_delay);
    }
    uturn_chk = 1;
}

// 초기화
void init(void) {
    IOCFG |= 0x01;  // P1 포트 사용을 위한 초기화
    P0 = 0xFF;     P1 = 0xFF;
    P2 = 0xFF;     P3 = 0xFF;

    ReadData  = 0;      // 센서 값 초기화
    speed     = 31;     // 진행 속도
    stop_delay = 8;     // 멈춤 속도
    turn_delay = 24;    // 좌,우 90° 회전할 때
    smooth_speed = 24;  // 좌,우 조금씩 이동할 때
    delay_num  = 60;
    flag = 0x01;
}
```

```

    uturn_chk = 0;
}

// 주함수
void main(void) {
    init();

    SENSOR_EN = 1;      // 적외선 발광 센서 가동

    while(1) {
        ReadData = SensorScan();
        LEDdata = ~flag;
        run();
    }
}

void run(void) {
    // LED를 사용하여 적외선 센서의 인지 값을 표시
    // 흰색 선에서 반사되는 빛을 받으면 수광 센서가 작용해서 LED가 켜진다.
    // 검정 선은 빛을 반사하지 않으므로 수광 센서가 작용하지 않고 LED가 꺼
    진다.
    switch(ReadData) {
        // 로봇이 중앙에 있는 경우, 직진
        // 오른쪽 바퀴: 전진, 왼쪽 바퀴: 전진
        case CSensor:
        case CSensor1:
        case RCSensor:
        case LCSensor:
            forward(speed);
            break;

        // 로봇이 오른쪽으로 치우쳐져 있는 경우,
        // 왼쪽으로 조금 이동
        // 오른쪽 바퀴: 전진, 왼쪽 바퀴: 후진

```



```
case RCSensor1:
case RCSensor2:
case RCSensor3:
case RCSensor4:
    turn_right(smooth_speed);
    break;

// 로봇이 왼쪽으로 치우쳐져 있는 경우,
// 오른쪽으로 조금이동
// 오른쪽 바퀴 : 후진, 왼쪽 바퀴 : 전진
case LCSensor1:
case LCSensor2:
case LCSensor3:
case LCSensor4:
    turn_left(smooth_speed);
    break;

// 갈림길에서 오른쪽 센서가 반응한 경우, 빠르게 회전
// 오른쪽 바퀴 : 후진, 왼쪽 바퀴 : 전진
case RSensor: // 3F, 0011 1111
case RSensor1: // 7F, 0111 1111
case RSensor2: // 77, 0111 0111
case RSensor3: // 83, 1000 0011
case RSensor4: // 17, 0001 0111
case RSensor5: // 9F, 1001 1111
case RSensor6: // 03, 0000 0011
    turn_right90();
    break;

// 갈림길에서 왼쪽 센서가 반응한 경우, 빠르게 회전
// 오른쪽 바퀴 : 전진, 왼쪽 바퀴 : 후진
case LSensor: // FC, 1111 1100
case LSensor1: // FE, 1111 1110
case LSensor2: // F6, 1111 0110
case LSensor3: // C1, 1100 0001
```

```

case LSensor4:           // E8, 1110 1000
case LSensor5:           // F9, 1111 1001
case LSensor6:           // C0, 1100 0000
    turn_left90();
    break;

// 십자 교차로
// 전진
case CROSS:              // 00, 0000 0000
case CROSS1:             // 3C, 0011 1100
case CROSS2:             // BD, 1011 1101
case CROSS3:             // 18, 0001 1000
case CROSS4:             // 08, 0000 1000
case CROSS5:             // 81, 1000 0001
    turn_left90();
    break;

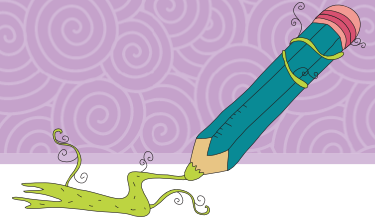
// 길이 없는 경우
// U 턴 하는 경우
case END_LINE:
    uturn();
    break;

// 기타 상황에 대해서는 모두 전진
// 오른쪽 바퀴 : 전진, 왼쪽 바퀴 : 후진
default:
    forward(speed);
}
}

```



# 단원 학습정리



01. 모터 드라이버에 두 개의 입력 단자를 사용해서 모터의 회전, 정지, 회전 방향, 관성운전을 제어한다.
02. 적외선 센서의 상태를 LED로 전달해서 확인한다.
03. 로봇이 갈림길에서 우회전할 조건은 LED 0이 켜질 때, 즉 가장 우측의 적외선 센서가 감지될 때이다.
04. 로봇이 갈림길에서 좌회전할 조건은 LED 7이 켜질 때, 즉 가장 좌측의 적외선 센서가 감지될 때이다.
05. 로봇이 LED 0와 LED 7이 동시에 켜질 때, 즉 가장 우측과 좌측의 적외선 센서가 켜질 때, 갈림길을 확인하고 직진한다.
06. 로봇이 LED 0이 켜질 때, 즉 가장 우측 적외선 센서가 켜질 때, 갈림길을 확인하고 직진한다.
07. 로봇이 LED 7이 켜질 때, 즉 가장 좌측 적외선 센서가 켜질 때, 갈림길을 확인하고 직진한다.



# 단원 종합문제

01

8개의 적외선 센서 중 어느 센서가 감지될 때 로봇이 우회전 하게 되는가?

- ① 센서 0      ② 센서 3      ③ 센서 4      ④ 센서 7

02

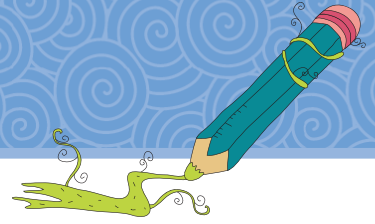
라인트레이서가 갈림길이 아닌 직선 길을 진행할 경우 감지되는 적외선 센서는?

- ① 센서 0와 센서 7      ② 센서 3과 센서 4  
③ 센서 1과 센서 6      ④ 센서 0와 센서 6

03

라인트레이서가 더 이상 전진할 길이 없는 경우를 인식하는 감지 조건은?

- ① 센서 3과 4만 감지된다.  
② 모든 센서가 감지된다.  
③ 센서 5와 6이 감지된다.  
④ 센서 3만 감지되다가 모두 감지되지 않는다.



04

라인트레이서가 갈림길에서 직진할 때 적외선 센서의 상태는?

- ① 센서 1과 2가 감지된다.
- ② 센서 3과 4가 감지된다.
- ③ 모든 센서가 감지된다.
- ④ 센서 1과 6이 감지된다.

05

정지선만 여러 개 평행하게 그어 놓은 상황에서 라인트레이서는 정지선에서 수직 방향으로 진행하고자 한다. 정지선을 발견할 때마다 약 1.5초 정지하고 재출발하는 프로그램을 작성해 보자.

06

라인트레이서가 SW1을 누르면 정지했다가 약 1.5초 후에 재출발하도록 프로그램을 작성해 보자.

07

라인트레이서가 SW1을 누르면 전진하고, SW2를 누르면 후진하는 프로그램을 작성해 보자.

# 부록

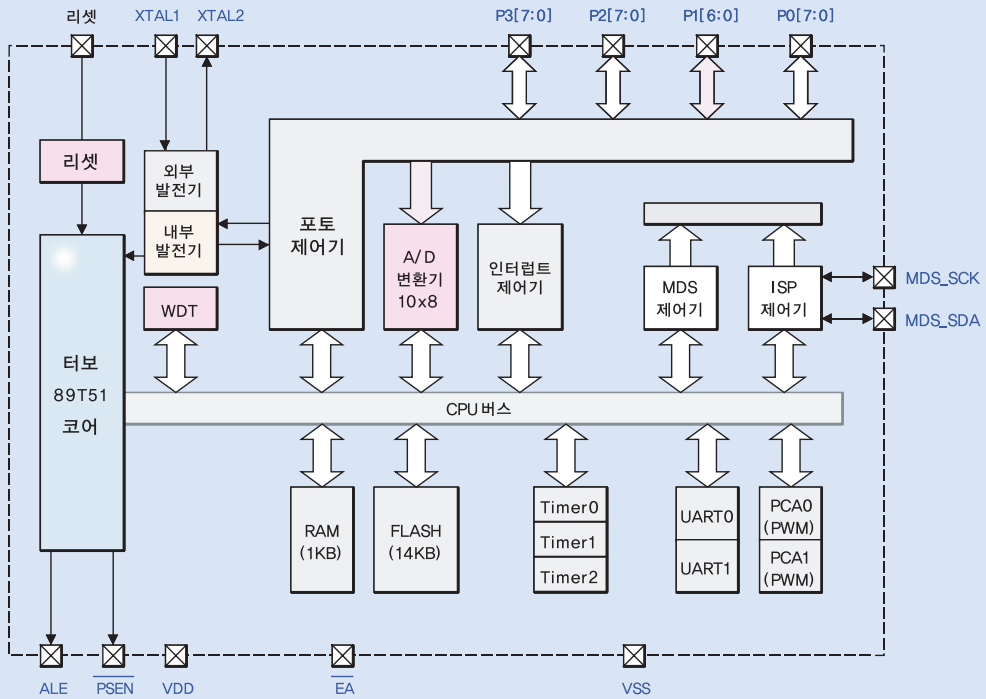
## ● GC89T51 마이크로프로세서

GC89T51은 인텔 8051과 호환성을 갖는 최신형 마이크로프로세서로, 국내의 마이크로프로세서 전문회사인 코아리버에서 개발했다. 원래의 8051이 명령어 하나를 수행하기 위하여 시스템 클럭 12주기를 사용하는데 비하여 GC89T51은 4주기를 사용하므로 명령어 수행속도가 3배 정도 빠르다. 14[KB]의 플래시 메모리를 내장하고 있기 때문에 별도의 장치가 없이 ISP 방식으로 프로그램을 마이크로프로세서에 기록할 수 있다. 또한 A/D 변환 모듈과 PWM(Pulse Width Modulator) 등 다양한 기능의 주변회로들을 포함하고 있기 때문에 시스템을 편리하게 구성하면서 비용도 절감할 수 있다. 또한 256 바이트의 내부 데이터 메모리 외에도 시스템을 구성할 때 필요한 데이터 메모리를 추가로 내장하여 외부 메모리처럼 MOVX로 접근하도록 구성하였다. 그 결과 대부분의 시스템에서 외부 데이터 메모리를 장착해서 P0 포트와 P2 포트를 사용하여 읽을 필요가 없으므로 일반 입출력에 더 많은 포트를 할당할 수 있다.

### GC89T51과 8051의 비교

| 비교항목           | GC89T51                                  | 8051              |
|----------------|------------------------------------------|-------------------|
| 클럭             | 40MHz                                    | 11.0592MHz        |
| 명령어 속도         | 명령어 1개 실행 → 4 클럭                         | 명령어 1개 실행 → 12 클럭 |
| Flash 메모리      | 14 킬로바이트 내장                              | 없음                |
| A/D 변환 모듈      | 10비트 A/D 변환 모듈 내장                        | 없음                |
| PWM(펄스 폭 변조)출력 | 12개까지 생성 가능                              | 없음                |
| 16 비트 타이머      | 3개 내장                                    | 2개 내장             |
| 직렬 통신 포트       | 2개                                       | 1개                |
| 외부 인터럽트        | 5개                                       | 2개                |
| Watchdog 타이머   | 내장                                       | 없음                |
| 내장 데이터 메모리     | 일반 내부 메모리 256 바이트<br>MOVX 접근 메모리 768 바이트 | 일반 내부 메모리 256 바이트 |

## 89T51 내부 구조



|                    |    |    |                    |
|--------------------|----|----|--------------------|
| ADC0 / T2 / P1.0   | 1  | 40 | VDD                |
| ADC1 / TZEX / P1.1 | 2  | 39 | P0.0 / AD0 / C1EX0 |
| ADC2 / P1.2        | 3  | 38 | P0.1 / AD1 / C1EX1 |
| ADC3 / P1.3        | 4  | 37 | P0.2 / AD2 / C1EX2 |
| ADC4 / INT2 / P1.4 | 5  | 36 | P0.3 / AD3 / C1EX3 |
| ADC5 / INT3 / P1.5 | 6  | 35 | P0.4 / AD4 / C1EX4 |
| ADC6 / INT4 / P1.6 | 7  | 34 | P0.5 / AD5 / C1EX5 |
| RESET              | 8  | 33 | P0.6 / AD6 / EC11  |
| RXD / P3.0         | 9  | 32 | P0.7 / AD7 / EC10  |
| MDS_SDA            | 10 | 31 | MDS_SCK            |
| TXD / P3.1         | 11 | 30 | ALE / PROG         |
| INT0 / P3.2        | 12 | 29 | PSEN               |
| INT1 / P3.3        | 13 | 28 | P2.7 / A15 / TXD1  |
| T0 / P3.4          | 14 | 27 | P2.6 / A14 / RXD1  |
| T1 / P3.5          | 15 | 26 | P2.5 / A13 / COEX5 |
| WR / P3.6          | 16 | 25 | P2.4 / A12 / COEX4 |
| RD / P3.7          | 17 | 24 | P2.3 / A11 / COEX3 |
| XTAL2              | 18 | 23 | P2.2 / A10 / COEX2 |
| XTAL1              | 19 | 22 | P2.1 / A9 / COEX1  |
| VSS                | 20 | 21 | P2.0 / A8 / COEX0  |

[ 40-PDIP ]

## ● 2006년 로봇 관련 대회

- 2006 제5회 전국 청소년 로봇 경진대회
- 2006 EBS 로봇파워 ON
- 2006 서울산업대학교 로봇 페스티벌
- 제8회 한양대학교 라인트레이서 대회
- 2006년 제 4회 오프로드 라인트레이서 대회
- 제8회 한양대학교 라인트레이서 대회
- 제1회 전국 휴머노이드 3종 경기대회
- 2006 전국 대학생 자율 로봇 경진대회
- 2006 ACE용인 전국배틀로봇축구대회
- 제2회 모듈형 지능로봇 경진대회 안내문
- 2006 국제 로봇 컨테스트(IRC 2006)
- 제24회 한국 마이크로 로봇 경연대회
- 2006년 유비쿼터스 로봇 경진대회(분야2)
- 제5회 단국대학교 마이크로마우스 경연대회
- 2006년 유비쿼터스 로봇 경진대회(분야1)
- 2006년 아주대 라인트레이서 대회
- 2006년 한국 지능로봇 경진대회(포항공대)
- 제2회 전국 지능로봇 대회 (경남)
- 제3회 대한민국 로봇대전
- 제4회 광주과기원 기전공학과 메카트로닉스 경진대회
- 제5회 로봇 항공기 경연대회
- 제12회 아주대 총장배 지능로봇 레이싱 대회
- 2006년 KT배 국제 로보원 대회
- 지능형 SoC Robot War 2006
- 2006년 제3회 대한민국 로봇대전

## ● 참고 문헌

- 로봇. 현대정보문화사. ISAAC ASIMOV. 2001
- 인터넷 다음은 로봇이다. 동아시아. 배일한. 2006
- 헬로우, ROBOT. 을파소. 로버트 말론. 2006
- 로봇과 인간. 한국로봇공학회. 변증남. 2006
- 로봇기술. 한국종합기술. 김영철. 2006
- 8051을 이용한 마이크로 마우스. 동일출판사. 차영배. 2001
- 마이크로프로세서 응용로봇 제작. 도서출판세화. 정상봉 외3인. 2003
- 마이크로 로봇 로비II. 복두출판사. 정기철 외3인. 2005
- 메카트로닉스 공학기술 2. 성안당. 선권석. 2005
- 메카트로닉스 공학기술 3. 성안당. 선권석. 2005
- 프로그래밍. 서울교과서. 성종국. 2006
- 마이크로 로봇. 복두출판사. 정기철 외1인. 2005
- 8051기초플러스알파. 성안당. 정용원. 2003
- 공학도를 위한 전기전자공학. 미전사이언스. 김영춘 외1인. 2006
- CORERIVER, “8051 호환 MCU: GC89T51 사용자 매뉴얼.”
- 정기철, 신동욱, 김도우, 권용세, 배종홍, “GC89T51(8051 호환) 설계 및 프로그램.”
- CORERIVER, “GENTOS(C-compiler를 포함한 설계환경) 사용자 매뉴얼.”
- Intel, “MCS 51 Microcontroller Family User's Manual.”

## ● 참고 사이트

- 휠체어로봇 : ( Wheelesley Robot ) :[www.ai.mit.edu](http://www.ai.mit.edu) (AI Lab., MIT)
- 화성탐사로봇 스피릿 : [www.nasa.gov](http://www.nasa.gov)
- 경비로봇 D1 : [www.alsok.co.jp](http://www.alsok.co.jp)
- 애완로봇 I-ROBO(I-Cybie) : <http://www.tigertoys.com> (Tiger Electronics)
- 정찰로봇 ARV : <http://www.darpa.mil> (Defense Advanced Research Projects Agency)
- 간호로봇 Pearl : <http://www-2.cs.cmu.edu/>





고등학교

# 로봇 제어 시스템



|           |     |           |
|-----------|-----|-----------|
| ○ 연구위원    | 조성준 | 경기도교육청    |
|           | 김태영 | 경기도교육청    |
|           | 김태갑 | 경기도교육청    |
| ○ 집필위원    | 성종국 | 양명디지털고등학교 |
|           | 김응석 | 한라대학교     |
|           | 박영복 | 양명디지털고등학교 |
|           | 송정환 | 양명디지털고등학교 |
|           | 송일재 | 원택오토메이션   |
| ○ 심의위원    | 정기철 | 대덕대학      |
|           | 송필재 | 동서울대학     |
|           | 이훈구 | 청평공업고등학교  |
|           | 김송경 | 서울로봇고등학교  |
|           | 김수용 | 원광디지털대학교  |
| ○ 검토·협의위원 | 양경택 | 대림대학      |
|           | 김종형 | 서울 산업대학교  |
|           | 윤혜영 | 양명디지털고등학교 |
|           | 조용형 | 양명디지털고등학교 |
|           | 이충기 | 양명디지털고등학교 |
|           | 신윤철 | 양명디지털고등학교 |
| ○ 삽화      | 강은하 | 서울교과서     |
| ○ 표지·편집   | 정유정 | 서울교과서     |

2007년 3월 1일 초판 발행 / 2007년 2월 1일 인쇄 / 2007년 3월 1일 발행

저작권자 경기도교육청

편찬자 로봇 제어 시스템 인정도서 편찬위원회

발행 및 인쇄인 서울교과서 / 서울시 마포구 서교동 448-38 한일빌딩 2층 TEL : 02-322-1350