로봇 축구 시스템의 개발(MIRAGE I) 2

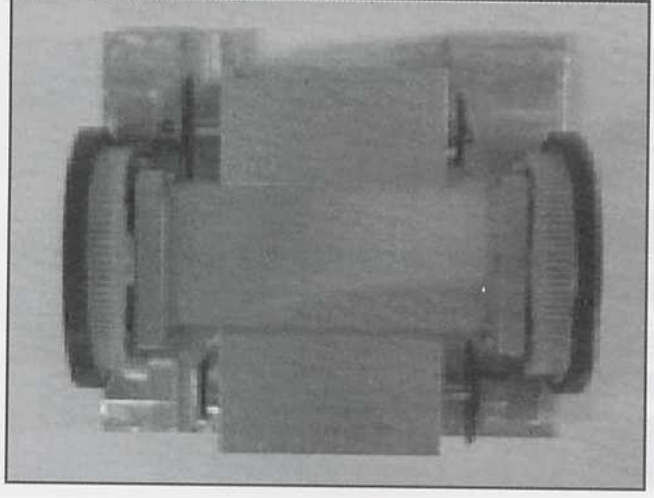
한국현, 최정이, 최필순, 이세중 | KAIST 전기 및 전자공학과 동아리 '미라지'

지능적이고 안전하며 튼튼한 로봇을 만들겠다는 일념으로 밤낮없이 매진한 결과 공식적인 대회에서의 수상을 통해 그 우수성을 검증 받은 축구로봇을 중심으로 하여 본 팀이 구현했던 로봇과 비젼 시스템, 그리고 전략 알고리즘 등에 대해 정리를 해 보고자 하며, 이 글이 로봇을 연구하고 시스템 설계를 공부하는 전자 관련 학과 학생들에게 많은 도움이 되길 바란다.

로봇의 몸체

그림 7은 MIRAGE I 로봇의 몸체를 위에서 본 모습이다. 가운데에 있는 구멍은 충전지가 들어가는 부분이고 앞에 있는 홈은 공이 빠져나가지 못하도록 만든 것이다. 앞에서도 언급했지만 로봇의 기계구조는 절대로 무시해서는 안 된다. 로봇을 만드는 사람들은 아마도 대부분이 전자공학이나 전산학을 전공하는 사람들일 것이다. 따라서 기계부를 간과하는 경우가 상당히 있는 것 같은데 큰 코 다치기 십상이다.

〈그림 7〉 로봇의 몸체



필자는 96년 국내대회용으로 로봇을 만들 때 정밀하게 만들어야 한다는 개념이 없었기 때문에 일반 mm단위의 자를 사용했었다. 완성된 로봇이 어떠했는지는 말하지 않아도 잘 알 것이다. 필자가 이렇게 강조를 하는 이유는 그 만큼 중요하다는 것을 명심하라는 뜻이다.

- 바퀴는 마찰력이 커야 미끄러짐을 막을 수 있어 제어의 오차를 줄일 수 있다.
- 충전지는 착탈이 쉽도록 로봇의 구조를 설계해야 한다.

MIRAGE I 로봇은 몸체를 Autocad로 설계하고 알루미늄을 깎아 만들었다. 알루미늄이 아닌 다른 재질을 사용하게 되면 로봇이 너무 무거워져 나중에 문제가 될 것이다. 사실 MIRAGE 로봇처럼 몸체를 꼭 금속으로 만들어야 하는 것은 아니다.

다른 참가팀들 중에는 PCB를 이용하여 몸체를 만든 팀, 알루미늄 판을 구부려서 몸체를 만든 팀, 레고를 이용하여 만든 팀, 알루미늄을 깎아 만든 팀 등 여러 종류의 로봇들이 있었다. 어떤 방법이 가장 좋다고 말을 할 수는 없지만 각각의 경우가 모두 장단점을 가지고 있다.

바퀴를 구현하는 방법에는 크게 두 가지가 있다. 하나는 두 개의 바퀴를 사용하고 중심을 잡을 수 있도록 보조바퀴를 앞뒤로 달아주는 방법이고 또 하나는 자동차와 같이 네 개의 바퀴를 사용하는 방법이다. 네 개의 바퀴를 사용할 경우에는 무한궤도 방식도 사용할 수 있을 것이다. 두 개의 바퀴를 사용할 경우에는 바닥과의 접점이 분명하기 때문에 로봇의 움직임을 정확히 계산할 수 있고 제자리에서 정확히 회전하는 것이 가능하다는 장점이 있지만 쉽게 미끄러질 수 있다는 단점이 있다.

네 개의 바퀴를 사용할 경우에는 쉽게 미끄러지지는 않겠지만 무한궤도 방식이 아니라면 회전을 할 때 문제가 생길 것이다. 무한궤도 방식인 경우에는 회전 시 중심점을 구하기가 애매하기 때문에 움직임 계산 시에 오차가 생길 것이다. 바퀴 역시 각각의 경우 장단점이 있는 것이다.

MIRAGE I 로봇은 그림에서 보는 것과 같이 두 개의 바퀴를 사용하고 중심을 잡기

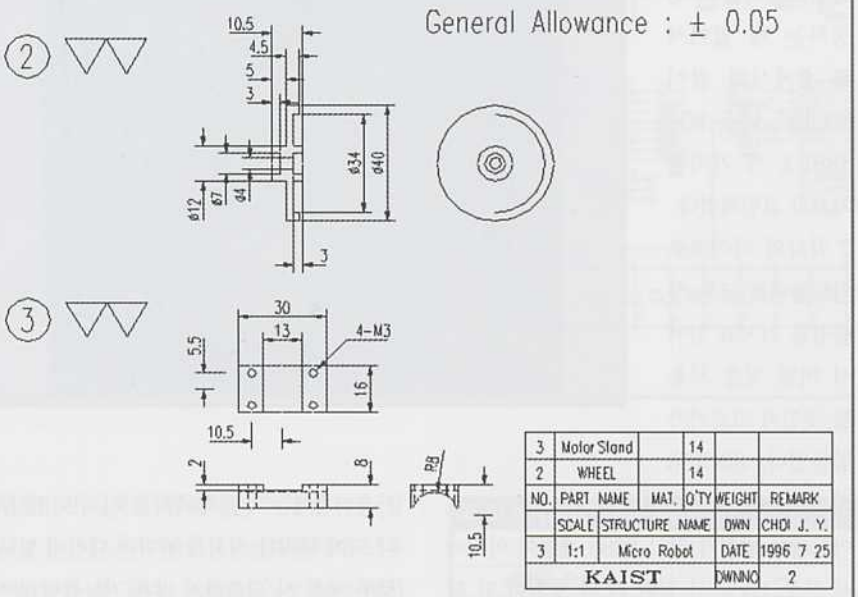
위해 보조바퀴로 작은 볼 케스터 두 개를 사용하였다. 또한 바퀴의 미끄러짐을 방지하기 위해서 모 타이어 회사 연구소로부터 마찰력이 큰 재질의 고무를 구하여 타이어로 사용하였다.

MIRAGE I 로봇의 몸체와 바퀴, 그리고 기어 등의 도면을 **그림 8, 그림 9**에 나타내었다.

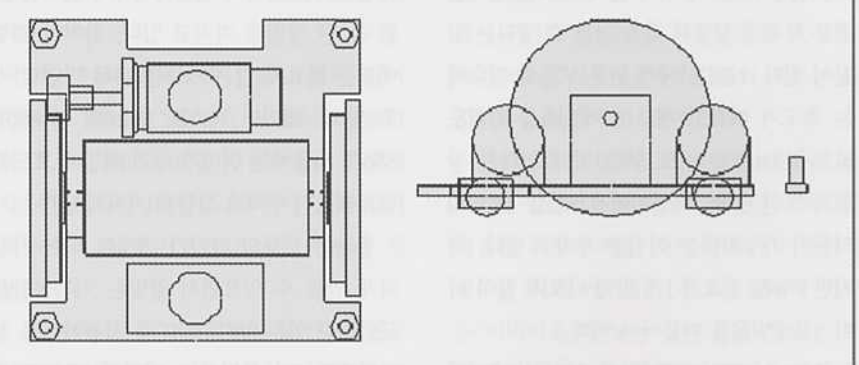
• 축구로봇용 소스 코드(일부만 공개)

```
/* MIRAGE I version */
/* KAIST, EE */
/* Programmed by Kuk-Hyun Han */
/* 1997. 5. 29 */
/* For MIROSOT '97 */

#pragma model(kc)
#include <stdio.h>
#include <80c196.h>

#include "\\8096\\gookie\\tools\\gookie97.h"

#define MY_ID 1
#define YOUR_ID 2
#define SOFTWARE_TIMER_PERIOD 25000
/* 20 msec */
#define SENSOR_INTERRUPT 5
/* 100 msec */
#define COMM_MAX 11
#define PENALTY_DEGREE 5

#define WHEEL_RIGHT
#define DB_COFX 5
#define DB_COFY 8
#define GO_BACK 0x01
#define LEFT_WALL 0x02
#define RIGHT_WALL 0x04
#define DRIBBLE 0x08
#define PRE_DRIBBLEL 0x10
#define PRE_DRIBBLER 0x20

#define MID_LINE 50
#define MID_WIDTH 0
#define MAX_SPD 127
#define MIN_SPD -127

#define OP_INST 0xfa
#define OP_DATA 0xf5
#define OP_MODE 0xf3
#define START_F 0x0f      /* first-half */
#define START_S 0x07   /* second-half */
#define RESET 0xff
#define EMER (0xf0+MY_ID-1)
#define PENAL (0x55+MY_ID-1)
#define R30 (0x10+MY_ID-1)
#define L30 (0x15+MY_ID-1)
#define R90 (0x1a+MY_ID-1)
#define L90 (0x20+MY_ID-1)
#define F10 (0x25+MY_ID-1)
#define F50 (0x2a+MY_ID-1)
#define SHOOT (0x30+MY_ID-1)

void init_robot(void);
void receive_interrupt(void);
void software_timer(void);
void set_data(void);
void reset(void);
BYTE arctangent(int data);
void cosine(void);
void sine(void);
void penalty(void);
void emergence(void);
void vector_calc(void);
void state(void);
void target(void);
void delta_dis_deg(void);
void moving(void);
void look(int data);
void lookb(BYTE data);

BYTE half;
int ball_x, ball_y;
int ball_xp, ball_yp;
```

〈그림 8〉 로봇의 바퀴와 모터 지지대



〈그림 9〉 로봇의 몸체 도면



```
int ball_mov_x, ball_mov_y;
int my_x, my_y;
int my_degree;
int delta_x, delta_y;
int target_x, target_y;
int target_degree;
BYTE my_mode;
BYTE my_state;
BYTE start_sig;
int goal_x, goal_y;
int delta_distance;
int delta_degree;
BYTE comm_data[COMM_MAX];
int cos[37];
int sin[37];
int signx, signy;
BYTE position;
BYTE keeper;
BYTE sensor_data;
int count_sen;
BYTE back_quit;
```

마이크로컨트롤러 보드

- 마이크로컨트롤러 보드 등은 항상 확장성을 고려하여 설계하는 것이 유리하다.
- 74시리즈의 칩을 사용할 경우 LS, HC, F 등 여러 타입 중에서 적당한 것으로 사용해야 한다.
- 마이크로컨트롤러의 선정도 상당히 중요하다. 먼저 자신이 만들려는 로봇의 기능을 정하고 그것에 가장 적합한 마이크로컨트롤러를 선정한다. 기능, 크기, 속도 등을 고려하여 자신이 구현하려는 시스템과의 적합성을 따져봐야 한다.

로봇의 두뇌에 해당하는 부분이 바로 마이크로컨트롤러 보드이다. MIRAGE I 로봇의 마이크로컨트롤러 보드에서 하는 일은 모든 센서들의 상태를 읽어들이고 인코더의 카운터 값을 읽는다. 호스트로부터 날아오는 데이터는 RF통신 회로를 거쳐 받아들인다.

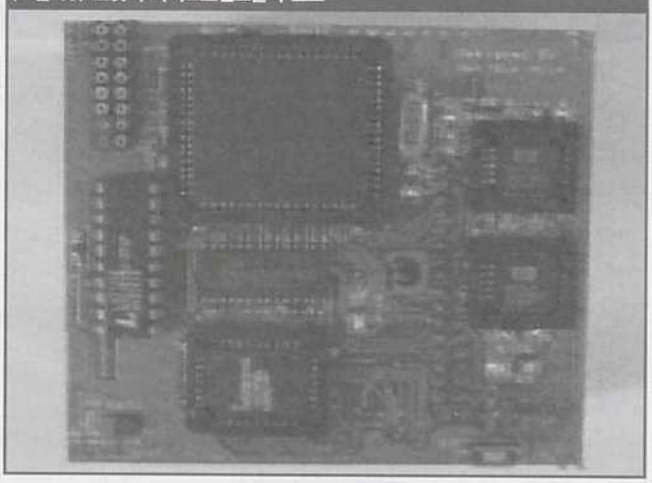
이렇게 읽은 데이터들을 종합하여 자신의 움직임을 결정하게 되고 스스로 제어를 하여 움직이게 하는 기능을 한다. 즉 로봇의 가장 중요한 부분 중의 하나이다. 물론 호스트 중심 시스템인 경우에는 큰 기능은 필요하지 않겠지만 로봇 중심 시스템을 구현하려 한다면 신중히 고려하여 설계해야 할 것이다.

따라서 마이크로컨트롤러 선정에 있어서도 신중해야 한다. 마이크로컨트롤러의 종류에 따라서 로봇의 연산속도, 기능, 보드의 크기, 특징 등이 결정된다. 필자는 마이크로컨트롤러를 선정하는 데 있어서 **표 2**에서와 같이 80C196, V55, 80-386EX 세 가지를 가지고 고민하였다.

〈표 2〉 마이크로컨트롤러의 특성

관련부분	80C196KC	V55	80386EX
Clock	(20MHz)	(25MHz)	(25MHz)
Data Bus	16/8~16	16/8~16	32/8~16
Mem. Addr.	64Kbytes	16Mbytes	64Mbytes
PWM	3PWM	1PWM	-
16bit Timer	2ch	5ch	3ch (8254)
Serial	1ch	2ch	2ch
ADC	8~10bits/8ch	8bits/4ch	-
I/O Port	5(0~1)	8(1)	3(0)
계열	8096	8086	8086
OS	모니터 프로그램	ROM-DOS가능	ROM-DOS가능
핀 수	68핀	120핀	120핀
보드 크기	小	中	大

〈그림 10〉 로봇의 마이크로컨트롤러 보드



각각의 마이크로컨트롤러가 모두 장단점을 가지고 있어서 어떤 것을 사용할 것인지 고르기가 힘들었다. 80C196의 경우 마이크로컨트롤러 속도가 느리고 어드레싱 범위가 작고 8086 계열이 아니어서 프로그래밍 시 일반 PC와 호환이 안 된다는 단점이 있지만 모터를 구동할 수 있는 PWM 신호가 3개나 출력되고 ADC가 8채널이나 있어 센서 등의 확장에 용이하다.

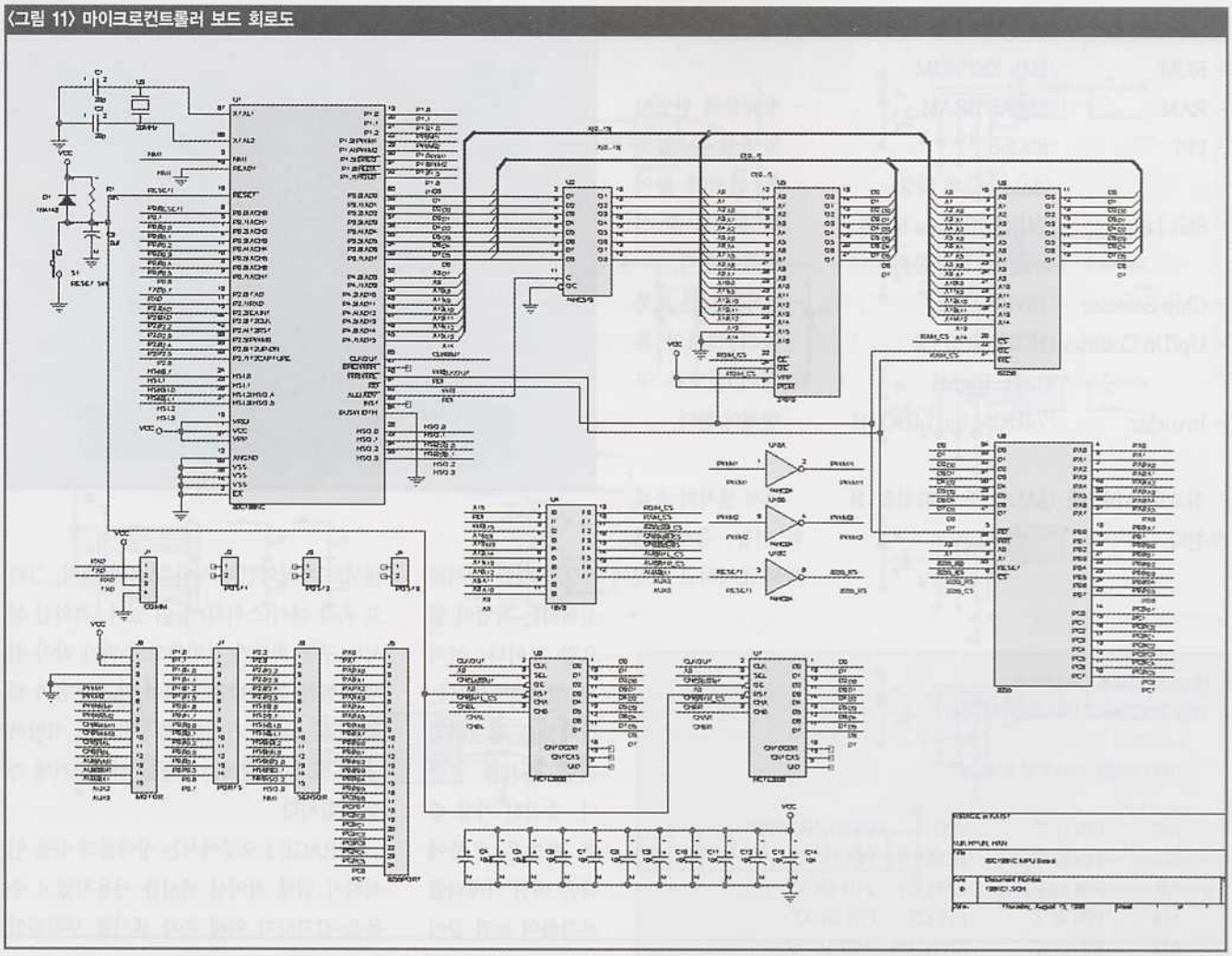
또한 68핀 밖에 되지 않고 보드를 설계할 경우 부피를 상당히 작게 만들 수 있다는 장점이 있다. NEC사에서 만든 V55의 경우에는 속도가 빠르고 메모리 어드레싱 범위도 넓고 16bit 타이머도 5채널이나 있다는 장점과 또한 8086 계열이어서 ROM-DOS의 사용이 가능하다는 이점을 가지고 있다. 하지만 PWM 신호가 1개 밖에 나오지 않아 타이머로 PWM을 만들어야 한다.

또한 ADC도 4채널밖에 없어서 센서에 사용하기에는 약간 부족하고 핀 수도 120개나 되어 부피가 커지게 된다는 단점이 있다. 로봇 제작 시 필요하지 않은 기능들이 많이 내장되어 있는 것이다.

마지막으로 모든 사람들이 잘 아는 80386EX의 경우, 빠른 속도와 넘치는 어드레싱 범위, 그리고 일반 PC와의 완벽한 호환 등 큰 장점을 가지고 있다. 하지만 로봇에게는 필요치 않은 기능을 너무 많이 가지고 있고 데이터 버스도 16bit를 사용해야 386을 사용하는 의미가 있기 때문에 보드를 설계해보면 부피가 상당히 커지게 된다.

필자는 로봇의 크기가 걱정이 되어 가장 작게 만들 수 있으면서 알맞는 기능들만을 포함하고 있는 80C196KC를 사용하기로 결정하였다. 더 설명하기 이전에 80C196을

〈그림 11〉 마이크로컨트롤러 보드 회로도



선택하여 로봇을 완성시킨 후의 생각을 말해보면, 세 가지 마이크로컨트롤러가 모두 부적합한 것 같다. 80C196은 기능 면에서는 가장 알맞지만 속도가 느리다는 단점이 치명적이다. 하지만 다른 마이크로컨트롤러를 사용했다면 현재 MIRAGE I 로봇의 기능을 모두 가지려면 크기 제한을 넘었을 것이다.

그림 11의 회로도에서 보는 바와 같이 ROM은 1Mbits 28F010, 120ns를 사용하여 그 중 256Kbits만을 사용하였다. 이렇게 사용한 이유는 1M가 오히려 더 값이 쌌기 때문이다. RAM은 256Kbits로 70ns를 사용하였다.

칩 셀렉터로는 GAL(16V8)을 사용하였고 인코더용 UP/DN 카운터로는 HCTL2020을 사용하였다. 센서와 모터를 구동하기 위해서 PPI로 82C55를 사용하였고 20MHz 크리스탈을 사용하였다. 8255의 포트는 **표 3**과 같이 설정하여 사용하였다.

〈표 3〉 마이크로컨트롤러 보드 I/O

	MotorDriver	Infrared sensor	Touch sensor
8255 Port A	–	6 bits enable	–
8255 Port B	–	5 bits sensing data	–
8255 Port C	2 bits enable	–	2 bits sensing data

마이크로컨트롤러 보드 설계 시 다른 보드와의 입출력을 고려하여 적절한 커넥터를 적절한 위치에 빼는 것이 중요하고 만일의 경우를 위해 어드레스 버스와 데이터 버스는 커넥터로 빼 놓는 것이 좋을 것이다. 또한 안정된 보드를 설계하기 위해서는 각 칩의 VCC, GND 사이에 콘덴서를 달아주고 외부와의 전원이 입출력되는 단자에 큰 콘덴서를 달아주는 것이 좋다.

80C196을 사용할 경우 회로도에서 보는 바와 같이 상당히 간단하다는 것을 알 수 있다. 단지 GAL을 구울 때와 프로그래밍을 할 때 주소 할당에만 신경을 쓰면 다른 것은 문제될 것이 없다.

- Microcontroller : 80C196KC 20MHz
- ROM : 1Mb EEPROM
- RAM : 256Kb SRAM
- PPI : 82C55 (data bus의 확장)
- 8bit Latch : 74HC573 (data bus와 address bus 분리)
- Chip Selector : 16V8 (GAL)
- Up/Dn Counter : HCTL2020 * 2 (Left, Right)
- Inverter : 74HC14 (or 74HC04)

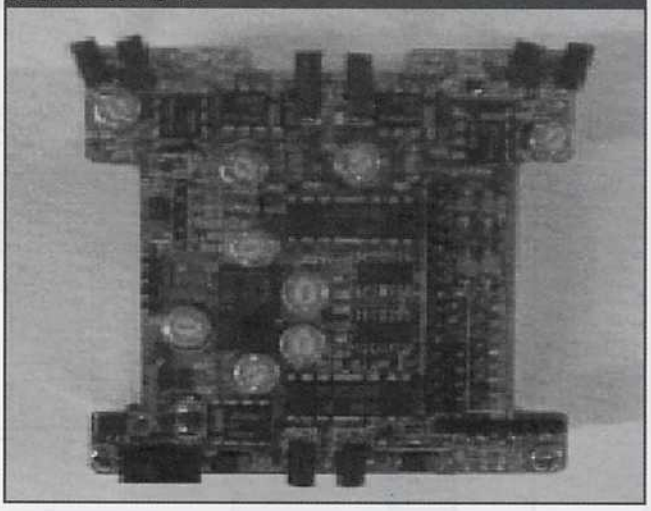
참고로 ABEL용 GAL의 소스 파일을 첨부한다.

```
Module BD196_GAL1
Title '80C196KC BOARD GAL(U4)';

    DECODER  DEVICE 'P16V8C';

    A15      PIN 1; "I"        VCC       PIN 20; "POWER"
    RD       PIN 2; "I"        ROM_CS    PIN 19; "O"
    WR       PIN 3; "I"        RAM_CS    PIN 18; "O"
    A14      PIN 4; "I"        PPI_CS    PIN 17; "O"
    A13      PIN 5; "I"        CNTL_CS   PIN 16; "O"
    A12      PIN 6; "I"        CNTR_CS   PIN 15; "O"
    A11      PIN 7; "I"        AUX1      PIN 14;
    A10      PIN 8; "I"        AUX2      PIN 13;
    A9       PIN 9; "I"        AUX3      PIN 12;
    GND      PIN 10; "POWER"   A8        PIN 11; "I"

    "RAM ENABLE (LOW ACTIVE)  = 8000H - FFFFH
    "PPI ENABLE (LOW ACTIVE) = 7D00H ( 0111 1101 0000 0000 )
    "CNTL ENABLE (LOW ACTIVE) = 7E00H ( 0111 1110 0000 0000 )
    "CNTR ENABLE (LOW ACTIVE) = 7F00H ( 0111 1111 0000 0000 )
    "ROM ENABLE (LOW ACTIVE)  = OTHERWISE
    X,C,L,H = .X.,.C.,0,1;

EQUATIONS
    !RAM_CS = A15;
    !PPI_CS = !A15 & A14 & A13 & A12 & A11 & A10 & !A9 & A8;
    !CNTL_CS = !A15 & A14 & A13 & A12 & A11 & A10 & A9 & !A8
    & !RD;
    !CNTR_CS = !A15 & A14 & A13 & A12 & A11 & A10 & A9 & A8
    & !RD;
    !ROM_CS = !A15 & PPI_CS & CNTL_CS & CNTR_CS;

END BD196_GAL1
```

센서

- 장애물의 판별이 완벽할 수 있도록 센서의 높이와 위치를 잘 선정해야 한다.
- 센서는 다른 로봇들의 센서와 혼선이 없도록 구현해야 한다.

〈그림 12〉 로봇의 센서 보드



먼저 센서의 종류와 기능, 특성들에 대해 조사하고 로봇에 가장 적합한 센서를 선택하는 작업이 필요할 것이다. 여기서 말하는 센서는 장애물을 감지하기 위한 센서를 말한다. 솔직히 가장 좋은 방법은 로봇에 작은 비젼 카메라를 부착하여 눈과 같이 사용할 수 있도록 해 주는 것이 최선이겠지만 카메라를 달 공간이 있는 것도 아니고 있다고 해도 화상을 분석하려면 시간이 너무 걸리기 때문에 불가능하다고 볼 수 있다. 따라서 최소한 전후방의 장애물은 감지를 해야 하기에 센서가 필요한 것이다. 공도 센서를 이용해 찾는 것이 가능할 것이다.

필자가 생각하기에 로봇에 적합한 센서로는 적외선 센서와 초음파 센서, 그리고 촉각 센서(스위치) 등이 있다. 적외선 센서는 구동 회로가 간편하고 부피가 작아 쉽게 구현할 수 있다는 장점이 있고 초음파 센서는 장애물의 거리를 측정하기에 적합하다. 또한 촉각 센서는 충돌을 감지하기에 가장 적합하다.

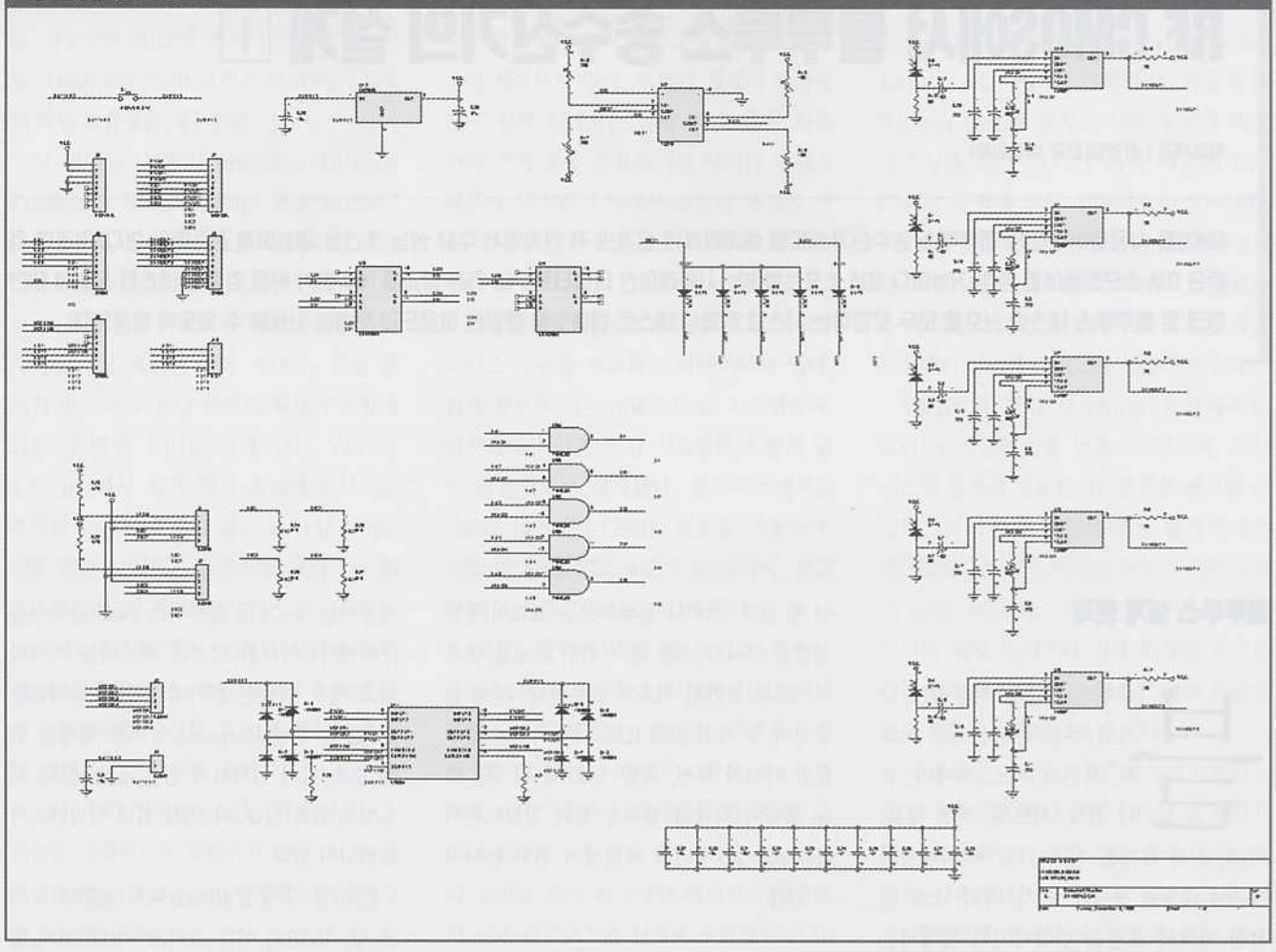
MIRAGE I 로봇에서는 장애물과 공을 인식하기 위해 적외선 센서를 사용하였고 충돌을 감지하기 위해 촉각 센서를 부착하였다. 적외선 센서는 전방에 4조, 후방에 1조를 달았다.

MIRAGE I 로봇에서 구현한 센서의 기능은 외부의 간섭에 영향을 받지 않도록 하고 전력 소모를 최소화하기 위해 약 50KHz의 캐리어 주파수를 실어 적외선을 쏘고 LM-567 Tone Detector를 필터로 사용하여 캐리어 주파수 성분만을 받아들이도록 회로를 설계하였다.

또한 각각의 센서는 프로그램으로 제어가 가능하도록 하기 위해서 모든 센서의 enable 신호를 마이크로컨트롤러 보드의 8255와 모두 연결시켰다. 또한 프로그램으로 센싱하는 거리를 두 가지로 바꿀 수 있도록 회로를 구현하였다.

- 적외선 LED : EL-7L * 5 (발광부)
- 적외선 Diode : ST-7L * 5 (수광부)

〈그림 13〉 센서 보드 회로도



- Relay : TQ2-5 (센서 거리 변환에 이용)
- Power TR : ULN2064 * 2 (전류 증폭)
- Regulator : LM2940
- Tone Detector : LM567 * 5 (센서의 캐리어 주파수 설정에 이용)

SN