






- 4차 과제 상세 보고서 (기기, 프로그램 상세보고서, 사용부품상세보고서)

프로젝트 명	4차 과제 : 라인 트레이싱을 하며 장애물 회피가 가능한 로봇 개발		
팀명	장인정신	용도	프로젝트 설명
작성	김태원		상세설명서(기기/프로그램)
검토	강예진, 김경필, 권영훈, 강인구, 강현빈	작성일자	2011년 11월 9일

단계	사진	설명	사용 재료명
0		0. 완성모습 구성 : 빛 센서 파트 x1 동력모터바퀴 x2 초음파센서파트 x1 후방 바퀴 파트 x1 기울임 방지 지지대x1	-
1		1. 빛 센서파트 사진 차례 대로 1-1, 1-2 라인 트레이싱을 위해 바닥의 명암을 판단하는 빛 센서파트입니다. 바닥 쪽에 밀착하게 제작하여 정확도를 높이려고 했습니다. 두 동력 모터 사이에 지지대 형식으로 장착하여 동력모터 사이의 양쪽으로 무게 쏠림을 완화 했습니다.	빛센서 x1 빔 5 x2 앵글커넥터 6 x4 연결패그 1 x 2 연결패그 5 x 5 연결패그 6 x2 연결패그 2 x3 축 2 x2 축 3 x3 직각축 커넥터 x1 앵글 커넥터 1 x3

단계	사진	설명	사용 재료명
2	 	<p>2. 동력 기어 바퀴</p> <p>사진 차례대로 2-1 : 모터에 기어사용 사진 2-2 : 모터에 기울임방지 지지대 장착</p> <p>기기를 움직이는 동력을 공급하는 동력 바퀴 부분입니다. 기어를 사용하여 모터 회전의 효율을 높이고 바퀴 자체에 기어 장착하고 앞쪽 기울임 방지 지지대까지 한 파트에 장착하여 작은 부피와 튼튼한 구조를 구성했습니다.</p>	<p>모터 x2 12 기어 x4 24 기어 x4 연결 페그1 x2 부시 x6 바퀴와 휠 x2 축7 x1</p>
3	  	<p>3. 앞쪽 기울짐 방지 지지대</p> <p>사진 차례대로 3-1,3-2,-3-3</p> <p>3-1 : 지지대의 정면 모습 3-2 : 모터에 지지대 장착 3-3 : 지지대 장착 모습</p> <p>빠른 속도나 방향 전환을 위한 정지 시 기기가 앞으로 넘어지는 것을 방지하기 위해 앞쪽에 기울임 방지 지지대를 장착 하였습니다. 실제로 바닥에는 닿지 않아서 속도에 영향을 주지 않으며 별도의 프레임 없이 동력 모터에 장착함으로써 기기 효율을 높이하고자 하였습니다.</p>	<p>벨트 휠 x1 연결 페그5 x4 연결 페그9 x2 연결 페그6 x2 각 빔11 x2 각 빔5 x2 축6 x2 연결 페그2 x4</p>

단계	사진	설명	사용 재료명
4	 	<p>4. 초음파 센서 파트</p> <p>사진 차례대로 4-1, 4-2</p> <p>4-1 : 초음파 센서 파트</p> <p>4-2 : 센서파트 본체 장착</p> <p>전방의 장애물을 판별하기 위해 전방을 바라보는 초음파 센서를 장착 합니다.</p> <p>전방 확인만을 목적으로 하므로 회전 모터 없이 본체에 직접 고정합니다.</p>	<p>초음파센서 x1</p> <p>3x3 빔 x1</p>
5	  	<p>5. 후방 바퀴 파트</p> <p>사진 차례 대로 5-1, 5-2, 5-3</p> <p>본체(인텔리전트 블록)이 서있는 모양이고 바퀴가 양쪽으로 구성 됨에 따라 앞뒤 기울어짐을 방지해줄 보조 바퀴를 장착합니다. 고정되어 회전 없던 전 보조바퀴와 다르게 전방위로 회전이 가능하도록 유연하게 제작하여 필요 없는 마찰을 줄이도록 하였습니다.</p>	<p>연결패그1 x20</p> <p>부시 x1</p> <p>축 8 x1</p> <p>빔11 x2</p> <p>각빔7 x2</p> <p>h빔 x1</p> <p>빔9 x2</p> <p>1/2부시 x1</p> <p>빔5 x2</p> <p>직각축 x1</p> <p>커넥터3 x1</p> <p>연결 페그9 x1</p> <p>벨트휠 x1</p>

단계	사진	설명	사용 재료명
6	 	<p>6. 본체 장착 사진 차례대로 6-1,6-2</p> <p>동력모터에 6-1과 같이 빔을 연결하고 본체 연결합니다. 연결 모습은 6-2와 같습니다.</p> <p>다른 파트를 모두 결합하여 완성합니다.</p>	-

완성품			
	정면	상면	측면
사진			
특징	1. 본체가 서있는 기립형 구조 2. 작은 부피, 튼튼한 구조		
장점	- 전진 시 불필요한 마찰이 매우 적다. - 센서밀착구조로 안정적 인식		
단점	- 미완성 된 듯한 부품 배치		

- 프로그램 상세 보고서

프로젝트	4차 과제 : 라인 트레이싱을 하며 장애물 회피가 가능한 로봇 개발		
팀명	장인정신	용도	상세명세서소프트웨어-01
작성	강인구		상세명세서소프트웨어-02
검토	강예진, 김경필, 권영훈, 김태원, 강현빈	작성일자	2011. 10. 11

프로그램 코드	
메인 함수 task main() { SetSensorType(S3, SENSOR_TYPE_LIGHT_ACTIVE); // 빛센서를 송광부를 켜 상태로 초기화 SetSensorMode(S3, SENSOR_MODE_RAW); // 빛센서 모드는 Raw(값 범위 (unsigned int) 0~1024) SetSensorLowspeed(S2); // UltraSonic 거리 센서 초기화 CalibrateLightSensor(); // 라인트레이싱 센서값 조정 while(true) { if(SensorUS(S2) < DIST_WALL) { // 장애물이 가까울 때에는 피해간다. Off(OUT_BC); goByWall(); } else if(avg < SensorRaw(S3)) { // 장애물이 없고 검은 선 위에 있다면 직진한다. OnFwd(OUT_BC, -STDPWR); } else if(avg > SensorRaw(S3)) { // 검은 선 위에 있지 않다면 선을 찾아서 그 방향으로 튼다. Off(OUT_BC); LineFollow(); } } }	
코드 설명	
<p>라인을 확인하여 라인을 따라 진행하다가 장애물을 만나면 피해서 가는 프로그램이다.</p> <ul style="list-style-type: none"> - 전진 방향에 장애물이 있으면 돌아서 감 - 검은 선 위에 있을 때에는 전진 - 흰색으로 벗어나면 멈추고 오른쪽, 왼쪽으로 선을 찾아 회전 	
프로그램 환경	
언어 : NXC (Not Exactly C)	
개발용 프로그램 : BricxCC (Bricx Command Center)	

프로그램 소스코드와 설명 주석

(좌->우, 상->하 순서)

```
// 센서 포트 배치
// Light      S3
// UltraSonic S2

// 모터 포트 배치
// Left      OUT_B
// Right     OUT_C
// 모터 역방향 회전이 전진임

#define DIST_WALL 22 // 장애물을 감지할 거리
#define STDPWR 40 // 전진 모터 구동 파워
#define PivotPower 30 // 방향 전환시 모터 구동 파워

/* 최대/최소값 조정 코드 */
#define ScanMinMax(value,min,max) W
    if(value < min) W
        min = value; W
    if(value > max) W
        max = value;

long t; // 명령 반복 시간을 나타내기 위한 변수
int k; //
unsigned int min, max, avg, tmp; // 빛센서 수치 계산을 위한
변수들
int dist;
/* 모터 수치를 조절하는 함수 */
sub MotorControl(int l_pwr, int r_pwr)
{
    OnFwd(OUT_B, -l_pwr);
    OnFwd(OUT_C, -r_pwr);
}
/* 지정한 각도만큼 본체를 회전시키는 함수 */
sub PivotBody(long pivotDegree)
{
    char i;
    if(pivotDegree < 0)
    {
        i = -100;
    }
    else {
        i = 100;
    }
    RotateMotorEx(OUT_BC, PivotPower, (pivotDegree * 130 /
100), i, 1, 1);

    // 모터의 회전 각도는 실험적 수치에 의해 비례상수를 결정하여
    곱한다.
    // 본 로봇에서는 상수가 130 / 100 이다.
    // 수학적으로 계산했을 때의 예상값과 생각보다 많이 다르다.

    Off(OUT_BC);
}
```

```
/* 라인트레이싱을 위한 빛센서 수치 기준값을 계산하기 위한 함수
*/
sub CalibrateLightSensor()
{
    max = 0;
    min = 1023;
    t = CurrentTick() + 1500;
    // 한 쪽으로 돌면서 최대/최소 센서 수치 저장
    MotorControl(PivotPower, -PivotPower);
    while(t > CurrentTick())
    {
        tmp = SensorRaw(S3);

        ScanMinMax(tmp, min, max);
    }
    t = CurrentTick() + 1500;
    //반대로 돌면서도 최대/최소 센서 수치를 저장
    MotorControl(-PivotPower, PivotPower);
    while(t > CurrentTick())
    {
        tmp = SensorRaw(S3);

        ScanMinMax(tmp, min, max);
    }
    Off(OUT_BC); // 원위치로 돌아오면 일단 정지

    avg = (min + max) / 2;
}

/* 라인에서 벗어났을 때 라인 방향으로 본체를 돌리는 함수 */
sub LineFollow()
{
    k = 0;
    // 오른쪽으로 약 90도 돌면서 라인이 있는지 찾는다.
    t = CurrentTick() + 600;
    MotorControl(40, -40);

    while(t > CurrentTick())
    {
        //라인을 찾았다면 멈추고 k를 1로 만든다.
        if(avg < SensorRaw(S3))
        {
            Off(OUT_BC);
            k = 1;
            break;
        }
    }
}
```

프로그램 소스코드와 설명 주석

```
// 오른쪽에서 라인을 못 찾았다면
if(k != 1)
{
    // 왼쪽으로 약 90도 돌면서 라인을 찾는다.
    t = CurrentTick() + 1200;
    MotorControl(-40, 40);

    while(t > CurrentTick())
    {
        if(avg < SensorRaw(S3))
        {
            Off(OUT_BC);
            k = -1;
            break;
        }
    }
}

if(k == -1)
{
    // 왼쪽으로 돌았을 때에는 조금 더 돌아주면 알고리즘
    // 특성상 진행속도 향상에 좋다.
    MotorControl(-40, 40);
    Wait(100);
}

Off(OUT_BC);
}

/* 장애물을 피해서 지나간 다음 라인으로 복귀하는 함수 */
sub goByWall()
{
    // 왼쪽으로 방향을 튼다(라인에서 벗어나는 방향)
    PivotBody(-90);
    Wait(350);

    // 직진(라인에서 벗어난다)
    RotateMotorEx (OUT_BC, PivotPower, -250, 0, 0, 1);
    Wait(350);

    // 오른쪽으로 방향을 튼다 (장애물 옆으로 지나갈 방향)
    PivotBody(90);
    Wait(350);

    // 직진(장애물을 지나가게 된다)
    RotateMotorEx(OUT_BC, PivotPower, -400, 0, 0, 1);
    Wait(350);

    // 오른쪽으로 방향을 튼다(라인 방향으로)
    PivotBody(90);
    Wait(350);
}
```

```
// 라인으로 접근한다
MotorControl(20, 20);
while(SensorRaw(S3) < avg);
while(SensorRaw(S3) > avg); // 라인을 살짝 지나치도록 한다.
RotateMotorEx(OUT_BC, PivotPower, -5, 0, 0, 1);
Wait(350);

// 라인에 복귀 완료
PivotBody(-100);
Wait(350);
}

task main()
{
    SetSensorType(S3, SENSOR_TYPE_LIGHT_ACTIVE); //
    // 빛센서를 송광부를 켜 상태로 초기화
    SetSensorMode(S3, SENSOR_MODE_RAW); // 빛센서 모드는
    // Raw(값 범위 (unsigned int) 0~1024)
    SetSensorLowspeed(S2); // UltraSonic 거리 센서 초기화

    CalibrateLightSensor(); // 라인트레이싱 센서값 조정

    while(true)
    {
        // 장애물이 가까울 때에는 피해간다.
        if(SensorUS(S2) < DIST_WALL)
        {
            Off(OUT_BC);
            goByWall();
        }

        // 장애물이 없고 검은 선 위에 있다면 직진한다.
        else if(avg < SensorRaw(S3))
        {
            OnFwd(OUT_BC, -STDPWR);
        }

        // 검은 선 위에 있지 않다면 선을 찾아서 그 방향으로 튼다.
        else if(avg > SensorRaw(S3))
        {
            Off(OUT_BC);
            LineFollow();
        }
    }
}
```


-상세 부품보고서

- 동력모터바퀴 + 기울임방지 지지대

형태	이름	개수	확인	용도
	모터	2		
	12 기어	4		
	24 기어	4		
	연결 페그1	2		
	부시	6		
	바퀴와 휠	2		
	축7	1		
	벨트 휠	1		
	연결 페그5	4		
	연결 페그9	2		
	연결 페그6	2		
	각 빔11	2		
	각 빔5	2		
	축6	2		
	연결 페그2	4		

(위 사진은 모터 부분의 1/2부분)

- 본체 파트(초음파센서 + 후방 보조바퀴파트)

형태	이름	개수	용도
	연결패그1	20	  
	NXT 본체	1	
	부시	1	
	축 8	1	
	초음파 센서	1	
	빔11	2	
	각빔 7	2	
	h빔	1	
	빔9	2	
	1/2 부시	1	
	3x3 빔	1	
	빔5	2	
	직각축 커넥터3	1	
	연결 페그9	1	
	벨트 휠	1	

- 빛센서 파트

형태	이름	개수	용도
	빛센서	1	
	빔 5	2	
	앵글커넥터 6	4	
	연결패그 1	2	
	연결패그 5	5	
	연결패그 6	2	
	연결패그 2	3	
	축 2	2	
	축 3	3	
	직각축 컨넥터	1	
	앵글 컨넥터 1	3	