
 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

공학설계입문

프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇
팀 명	장 인 정 신
문서 제목	기밀 보고서

Version	2.0
Date	2011-NOV-30

팀원	20093276 김태원 (조장)
	20113247 강예진
	20113248 강인구
	20113250 강현빈
	20113251 권영훈
	20113252 김경필
지도교수	최준수 교수님

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 공학설계입문 수강 학생 중 프로젝트 “장애물 인식과 회피가 가능한 라인트레이싱 로봇”을 수행하는 팀 “장인정신”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “장인정신”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	기밀보고서 - 장애물 인식과 회피가 가능한 라인트레이싱 로봇.hwp
원안작성자	강 인구
수정작업자	김 태원 , 강 예진

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2011-11-27	강 인구	1.0	최초 작성	
2011-11-28	강 예진	1.1	내용 수정	문제점 및 해결 방안 수정
2011-11-29	김 태원	1.2	내용 수정	도전 과제 제안 내용 추가
2011-11-30	강 인구	1.3	내용 수정	결론 수정

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

목 차

1	서론	4
2	기본 아이디어	5
2.1	H/W 디자인 방향	5
2.2	S/W 디자인 방향	6
3	수행 내용	8
3.1	직교형 장애물 인식을 이용한 직각 코스 주행	8
3.2	장애물 인식 및 회피	8
3.3	트랙중첩 인지 및 처리	8
3.4	급격한 곡률 코스의 처리	8
3.5	막다른 골목 인식 처리	8
4	향후 추진계획	9
4.1	문제점 및 해결방안	9
4.2	도전과제 제안	9
5	결론	10
6	참고문헌	10
7	전체 소스코드	11

 국민대학교 컴퓨터공학부 공학설계입문	기말보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

2 기본 아이디어

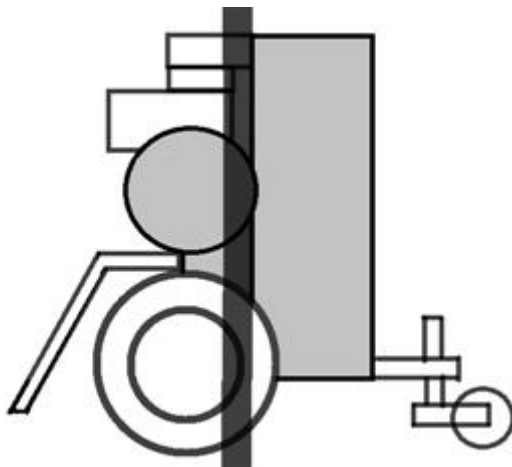
H/W는 크기는 작게, 부품 무게는 최대한 회전 중심축에 가까이 배치되도록 하여 제자리 회전과 장애물 통과가 쉽도록 직립형으로 제작한다.

S/W는 트랙 위에서는 직진하고, 벗어났을 경우에는 왼쪽으로 벗어났는지, 오른쪽으로 벗어났는지를 검사하여 항상 바른 방향으로 가게 한다.

2.1 H/W 디자인 방향

앞선 과제에서, 로봇이 회전할 때 반동을 줄이려면 무게가 회전축에 가까이 놓이는 것이 좋다는 점을 발견하여 이 점을 핵심적인 디자인 방향으로 잡았다.

그에 따라, Intelligent brick과 모터가 차지하는 무게가 가장 크므로 이 둘을 본체의 중심(회전축)에 놓이도록 하였다. 그에 따라 Intelligent brick은 수직으로 세웠고, 모터 역시 수직으로 세워 Intelligent brick과 결합하였다.



또한 앞선 과제에서 모터가 본체에 고정되는 부분이 너무 간결할 경우 강도가 너무 약해 모터와 바퀴 축이 기울어지는 현상을 겪었기 때문에 모터가 단단히 고정되도록 하는 것을 많이 고려하였다. 그와 함께 빛 센서를 달기 위한 고정 부분이 필요하여 이 두 가지 역할을 통합한 고정 뼈대를 만들고자 하였다.

그에 따라, 페그와 축, 직각 커넥터 등을 이용하여 양쪽 모터와 Intelligent brick, 빛 센서를 서로 연결하는 구조물을 제작함으로써 강도와 센서 장착부, 짧은 앞뒤 길이를 모두 충족하도록 하였다.

그리고 로봇이 중심을 유지하도록 하면서, 작동시에 자유도에 영향을 받지 않도록 무게가 약간 더 쏠려 있는 뒤쪽으로는 자유롭게 방향이 바뀌는 보조바퀴를 장착하였고, 정지시에 앞으로 쏠리는 것을 방지하기 위해 직진/회전시에는 땅에 잘 닿지 않는 지지대를 센서 앞부분으로 장착하였다.

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

2.2 S/W 디자인 방향

라인 트레이싱을 위한 빛 센서 수치를 측정을 통해 결정하는 부분, 장애물이 없고 라인 위에 있을 때 직진하는 부분, 라인을 벗어났을 경우 라인을 찾는 부분, 장애물을 판단하고 회피하는 부분으로 기능을 나누어 설계하였다.

빛 센서 수치를 측정하고 평균값을 정하는 부분의 알고리즘은 다음과 같다.

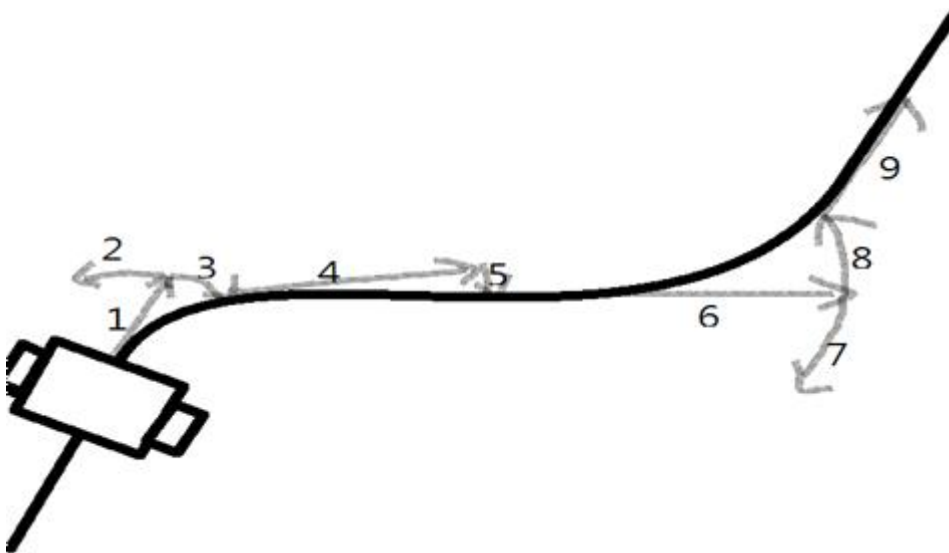
- 1) 제자리에서 약 180도 가량을 회전하면서 최대, 최소 밝기를 확인한다.
- 2) 원래 위치로 복귀하고, 최대, 최소값을 더하여 평균을 낸다. 이를 기준값으로 한다.


장애물이 없고 라인 위에 있을 경우에 직진하는 부분의 알고리즘은 다음과 같다.

- 1) 장애물이 가까이 있는가? -> 정지하고 장애물을 판단, 회피한다.
- 2) 1번이 아니고 검은 선 위에 있는가? -> 직진한다.
- 3) 2번이 아니고 검은 선 위에 있지 않은가? -> 정지하고 라인을 찾는다.
- 4) 그 외의 기타 예외가 발생할 경우는 직진한다.

라인을 찾는 알고리즘은 간단히 나타내면 다음과 같다.

- 1) 라인 위에 있을 경우(기준 수치보다 어두울 때)에는 직진한다.
- 2) 벗어났을 경우(기준 수치보다 밝을 때)에는 그 자리에서 정지한다.
- 3) 직진하던 방향으로부터 왼쪽, 오른쪽으로 90도 가량을 회전하면서 라인을 찾는다.
- 3) 라인이 발견되면 그 방향에서 정지한 후 다시 직진한다.



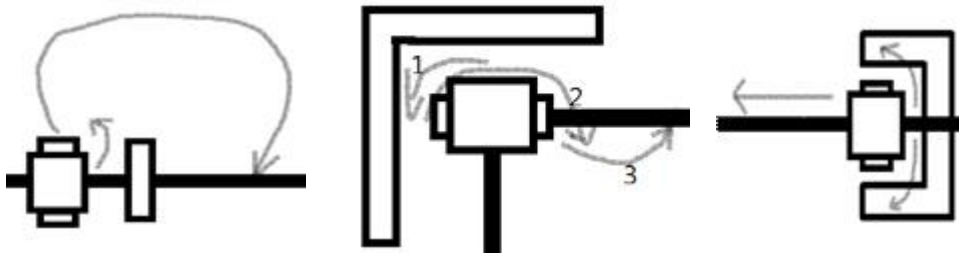
 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

장애물이 발견되었을 때 판단, 회피하는 알고리즘은 다음과 같다.

전진하던 방향에서 장애물이 감지되었을 경우

- 1) 왼쪽으로 본체를 돌려 거리를 측정한다. 막혀있지 않다면 벽을 돌아서 회피한다.
- 2) 1번에서 막혀있다면 180도 회전(원래 진행하던 방향의 오른쪽)하여 거리를 측정한다. 막혀있다면 ㄷ자형 벽이므로 원래 방향으로 복귀한 후 후진한다.
- 3) 오른쪽이 막혀있지 않다면 벽을 돌아서 회피한다.

* 벽을 돌아서 회피하는 경우에는 수직으로 틀어 회피하지 않고 수직보다 더 틀어 회피하도록 하여, 직각 구간에서는 라인에 복귀하도록 한다.



 국민대학교 컴퓨터공학부 공학설계입문	기말보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

3 수행 내용

핵심 부품은 모터 2개, 거리 센서 1개, 빛 센서 1개만을 사용하였으며, C언어 형식의 NXT 프로그래밍용 언어인 NXC(Not Exactly C)를 사용하여 프로그램을 작성하였다.

시험 동작을 했을 때 약 80%의 완주 성공률을 보였으며, 완주시 기록은 약 1분 45초 전후를 기록하였다.

3.1 직교형 장애물 인식을 이용한 직각 코스 주행

- 직각코스의 장애물은 왼쪽/오른쪽을 확인함으로써 장애물을 인식한다.
- 직진 방향 장애물 인식과 동일한 프로그램으로 직각벽도 통과하도록 하였다.
- 원형을 그리면서 회피하고, 라인을 만났을 때 멈추므로 같은 알고리즘으로 두 가지 기능을 할 수 있었다.

3.2 장애물 인식 및 회피

- 장애물에 접근했을 때 왼쪽이 트여 있다면 원을 그리면서 회피하고 라인에 복귀하도록 하였다.
- 직각벽과 정면 장애물을 같은 알고리즘으로 동작 가능하게 설계하여 로봇이 판단해야 할 경우의 수와 오류를 줄였다.

3.3 트랙중첩 인지 및 처리


- 우리 조 로봇의 라인트레이싱 알고리즘 자체가 검은 선 위에서는 직진하도록 하였기 때문에 트랙이 중첩되었을 경우에도 직진하여 지나가게 된다.

3.4 급격한 곡률 코스의 처리

- 급격한 곡률이 있는 곳에서는, 라인에서 벗어났을 때 마지막으로 발견된 방향을 우선 검사하게 하여 속도를 높이도록 하였다.
- 라인트레이싱 알고리즘의 특성상 직각 이상의 각도에서는 코스를 이탈할 확률이 극히 낮다.

3.5 막다른 골목 인식 처리

- ㄷ자형 막다른 골목에서는 거리센서를 통해 검사하여 왼쪽, 오른쪽이 모두 막혀있다고 확인되면 골목을 후진하여 빠져나간 후 일정시간 후에 멈추게 하여 END라인에서 끝내도록 하였다.

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

4 향후 추진계획

4.1 문제점 및 해결방안

현재 로봇은 직진 구간에서의 속도가 다른 라인트레이싱 방식의 로봇에 비해 현저히 느리다. 이것은 연속 곡선 구간에서 속도를 높이기 위해 라인에서 벗어났을 경우 바로 이전에 벗어났을 때 라인을 발견한 방향을 우선으로 검사하도록 하였기 때문인데, 이로 인해 직진 구간에서 직진 후 검사 시간이 현저히 길어진다.

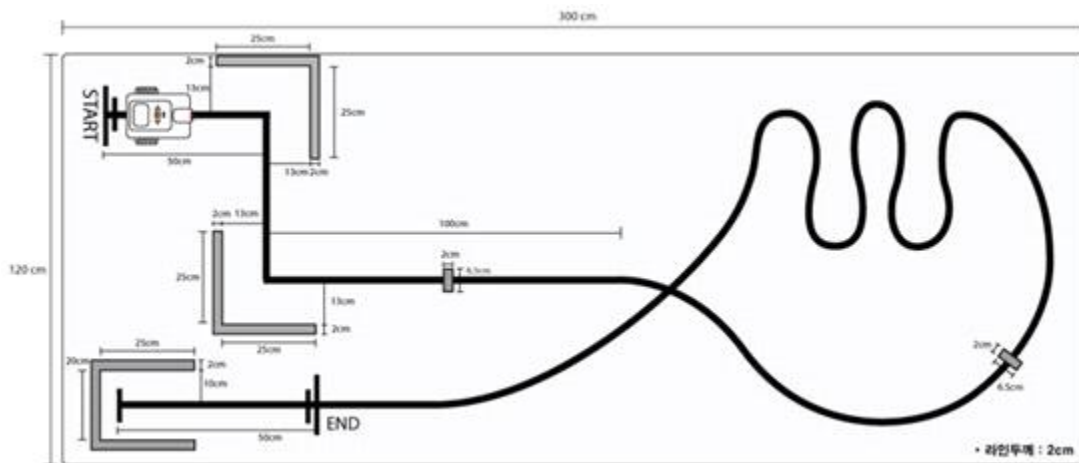
-> 직진 구간에서 왼쪽 오른쪽을 번갈아가며 검사하게 하면 빨라지게 된다.

-> 연속 곡선 구간은 2번 이상 같은 방향에서 선이 발견되었을 경우에 그 방향으로 우선 검사하게끔 바꾸어야 한다.

4.2 도전과제 제안

- 한쪽 벽을 계속 따라가는 방법을 통하여 미로를 탈출하는 로봇
- 미로에서 공을 발견하면 집어서 들고 나오도록 한다.
- 직각 미로로 하되 직진성을 유지하기 위해 라인을 따라 직진할 수 있도록 한다.

5 결론



시험 주행을 30회 가량 시행하였으며, 시험 주행에서는 80% 이상의 성공률을 보였다. 개선된 알고리즘을 적용해보지는 못하였으나 적용할 경우 기존의 1분 50초대의 기록에서 20초가량 단축될 것으로 예상되었다.

시행된 실제 대회에서는 2번 주행 중 첫 번째는 반사광의 영향으로 성공적으로 완주하지 못했고, 두 번째 주행 시에는 비교적 제대로 작동하였으나 센서 고장으로 인하여 장애물과 충돌하였다.

대회에서는 성공적이지 않았으나 기존 시험 주행에서의 성공률과 고려할 수 없었던 변수를 고려했을 때 성공적인 프로젝트였다고 볼 수 있다.

6 참고문헌

Br i c x C C , N X C H o m e p a g e

<http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/index.html>

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

7 전체 소스코드

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

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

#define DIST_WALL 21 // 장애물을 감지할 거리
#define STDPWR 55 // 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
char k; //
unsigned int min, max, avg, tmp; // 빛센서 수치 계산을 위한 변수들
int dist;
char i;

/* 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 j;

    if(pivotDegree < 0)
    {
        j = -100;
    }
    else
    {
        j = 100;
    }
}
```

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

```
RotateMotorExPID(OUT_BC, PivotPower, (pivotDegree * 130 / 100), j, 1, 1, PID_2, PID_2,
PID_2);
```

```
// 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()
```

```
{
```

```
    // 오른쪽 우선 검사
```

```
    if(i == 0)
```

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

```

{
    k = 0;

    // 오른쪽으로 약 90 도 돌면서 라인이 있는지 찾는다.
    t = CurrentTick() + 450;
    MotorControl(30, -30);

    while(t > CurrentTick())
    {
        //라인을 찾았다면 멈추고 k 를 1 로 만든다.
        if(avg <= SensorRaw(S3) + (avg / 7))
        {
            // Wait(50);
            Off(OUT_BC);
            k = 1;
            break;
        }

        /*
        MotorControl(40, -40);
        Wait(150);
        Off(OUT_BC);
        */
    }

    // 오른쪽에서 라인을 못 찾았다면
    if(k != 1)
    {
        // 왼쪽으로 약 90 도 돌면서 라인을 찾는다.
        t = CurrentTick() + 900;
        MotorControl(-30, 30);

        while(t > CurrentTick())
        {
            if(avg <= SensorRaw(S3) + (avg / 7))
            {
                // Wait(50);
                Off(OUT_BC);
                k = -1;
                break;
            }
        }
    }

    /*
    // 방향 보정
    MotorControl(-40, 40);
    Wait(150);

```

 국민대학교 컴퓨터공학부 공학설계입문	기말보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

```

        Off(OUT_BC);
        */
    }

    if(k == 1)
        i = 0;

    if(k != 1)
        i = 1;
}

// 왼쪽 우선 검사
if(i == 1)
{
    k = 0;

    // 왼쪽으로 약 90 도 돌면서 라인이 있는지 찾는다.
    t = CurrentTick() + 450;
    MotorControl(-30, 30);

    while(t > CurrentTick())
    {
        //라인을 찾았다면 멈추고 k 를 1 로 만든다.
        if(avg <= SensorRaw(S3) + (avg / 7))
        {
            // Wait(50);
            Off(OUT_BC);
            k = 1;
            break;
        }

        /*
        MotorControl(-40, 40);
        Wait(150);
        Off(OUT_BC);
        */
    }

    // 왼쪽에서 라인을 못 찾았다면
    if(k != 1)
    {
        // 오른쪽으로 약 90 도 돌면서 라인을 찾는다.
        t = CurrentTick() + 900;
        MotorControl(30, -30);
    }
}

```

 국민대학교 컴퓨터공학부 공학설계입문	기밀보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

```

while(t > CurrentTick())
{
    if(avg <= SensorRaw(S3) + (avg / 7))
    {
        // Wait(50);
        Off(OUT_BC);
        k = -1;
        break;
    }
}

// 방향 보정
/*
MotorControl(40, -40);
Wait(150);
Off(OUT_BC);
*/
}

if(k == 1)
    i = 1;

if(k != 1)
    i = 0;
}

// Off(OUT_BC);
}

/* 장애물을 피해서 지나간 다음 라인으로 복귀하는 함수 */
sub goByWall()
{
    // 왼쪽으로 방향을 튼다
    PivotBody(-105);
    Wait(350);

    if(SensorUS(S2) > DIST_WALL + 6)
    {
        PivotBody(-15);
        // 장애물을 돌아가면서 라인으로 접근한다
        OnFwdSyncPID(OUT_BC, -70, 15, PID_1, PID_1, PID_1);
        while(SensorRaw(S3) < avg + (avg / 10));
        while(SensorRaw(S3) > avg); // 라인을 살짝 지나치도록 한다.
        Off(OUT_BC);
        Wait(150);
    }
}

```

 국민대학교 컴퓨터공학부 공학설계입문	기말보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

```

// 방향 복귀
MotorControl(-30, 30);
while(SensorRaw(S3) < avg + (avg / 10));
Wait(150);
// 라인에 복귀 완료

i = 1;
}

// 왼쪽이 막힌 경우
else
{
    PivotBody(190);

    // ㄷ 모양 구간(끝)일 경우
    if(SensorUS(S2) < DIST_WALL + 6)
    {
        PivotBody(-95);
        OnFwdSyncPID(OUT_BC, 40, 0, PID_1, PID_1, PID_1);
        Wait(2500);
        Off(OUT_BC);
        Stop(true);
        //StopAllTasks();
    }

    else
    {
        PivotBody(20);
        OnFwdSyncPID(OUT_BC, -70, -17, PID_1, PID_1, PID_1);
        while(SensorRaw(S3) < avg + (avg / 10));
        while(SensorRaw(S3) > avg); // 라인을 살짝 지나치도록 한다.
        Off(OUT_BC);
        Wait(150);

        // 방향 복귀
        OnFwdSyncPID(OUT_BC, -40, 100, PID_1, PID_1, PID_1);
        while(SensorRaw(S3) < avg + (avg / 10));
        Wait(150);
    }
}

}

task main()
{

```


 국민대학교 컴퓨터공학부 공학설계입문	기말보고서		
	프로젝트 명	장애물 인식과 회피가 가능한 라인트레이싱 로봇	
	팀 명	장인정신	
	Confidential Restricted	Version 1.3	2011-Nov-30

```

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

i = 0;

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

    // 장애물이 없고 검은 선 위에 있다면 직진한다.
    else if(avg < SensorRaw(S3))
    {
        OnFwdSyncPID(OUT_BC, -STDPWR, -2, PID_1, PID_1, PID_1);
    }

    // 검은 선 위에 있지 않다면 선을 찾아서 그 방향으로 튼다.
    else if(avg > SensorRaw(S3))
    {
        LineFollow();
        OnFwdSyncPID(OUT_BC, -STDPWR, 0, PID_1, PID_1, PID_1);
    }

    else
    {
        OnFwdSyncPID(OUT_BC, -STDPWR, -2, PID_1, PID_1, PID_1);
    }
}
}

```