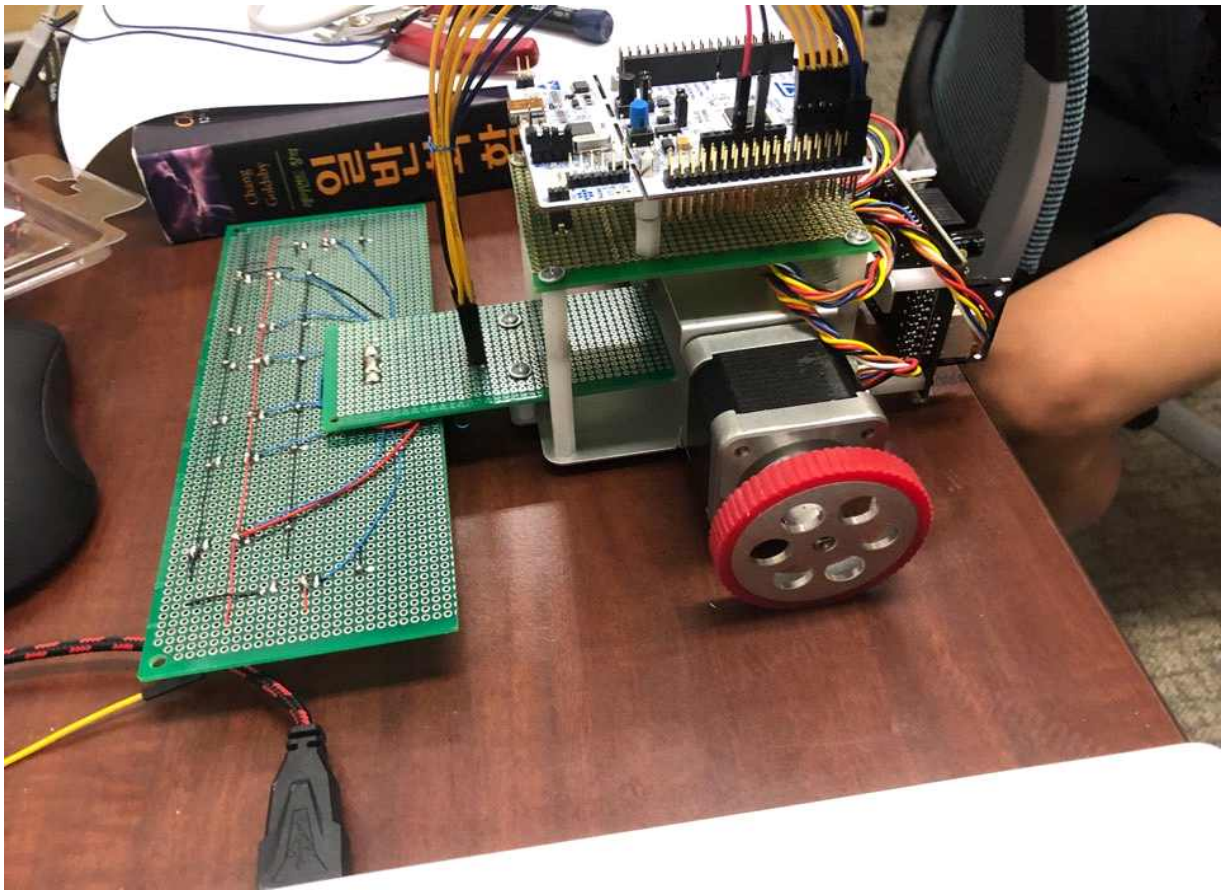
Ajou University.

2018년 1학기 하계 Project

#1차 프로젝트 - Line Tracer



작 성 자 : 남기현
작성일자 : 2018년 9월 13일
최종 수정일 : -----

Line Tracer 제작 결과 보고서

과 제 명	Line tracer
개발자명	남기현
과제기간	하계 방학
지 역	아주대학교
총팀원수	1

팀 원	학번	학 부	학 년	비 고
남기현	201820949	전자공학과	1	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-
-	-	-	-	-

- 개발 목적 및 동기
적성을 찾기 위한 탐구활동

-총 예 산

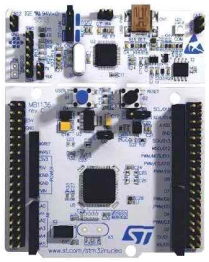
제품명	가격	수량	계
NUCLEO-F103RB	20000	1	20000
Stepping모터구동모듈3A (AM-CS2P)	27500	1	27500
SteppingMotor (103H5205-0473)	35980	1	70000
ST-1KLA(수광부)	0	8	0
EL-1KL2(발광부)	0	8	0
합계			117500

1. 프로젝트 설계

1.1 제품 설명

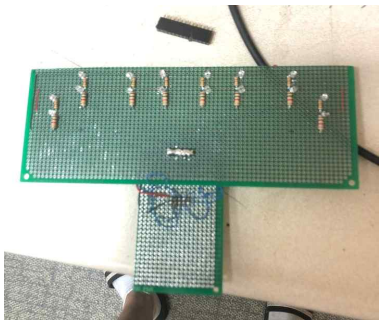
-main 보드

NULCEO-F103RB



-Arduino™와 공유되는 사용자 LED 1개
-사용자 1명 및 재설정 푸시 버튼 1명
-2.768kHz LSE 크리스탈 오실레이터

-적외선 감지 센서

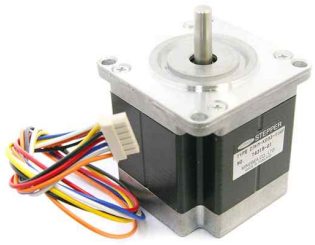


-8개의 EL-KL2(발광)와 8개의 수광(ST-1KLA)이 달려있음
-발광부에서 발산한 빛이 바닥에 반사되어 돌아와 수광부에서 인식
-검은색과 흰색에서 반사되는 빛의 차이로 라인을 인식

-steppingMotor 구동 모듈AM-CS2P

-라인트레이서용 스테핑 모터 구동보드
-스테핑 모터 2개 구동
-소프트웨어적으로 A, /A, B, /B 신호를 인가하여 제어
-10Pin Cable 과 12V 전원 공급 커넥터 연결
-모터에 흐르는 전류량을 조절할 수 있음 (가변저항 사용)

-Step Motor



-모터의 한 바퀴를 여러 개의 스텝으로 나누어 제어
-전류의 방향이 항상 한쪽으로만 흐르는 방식
(유니폴라 스텝모터)
-Ticker을 사용해 모터를 돌리는 함수의 주기를 바꿔가며 속도 조절

1.2 프로그램 설계

-1차 주행
주어진 맵의 라인을 따라 차체가 이동하도록 해야 한다.

해결해야할 문제
1) 검은 선과 흰선(라인) 판단.
2) 스텝모터 제어
3) 좌회전 우회전 등 전반적인 차체 제어
4) 교차로 판단
5) 정지

-2차 주행
1차 주행에서의 길을 외워 직선코스에서는 더 빠르게 주행하여야 한다.

해결해야할 문제
1) 1차 주행에서의 코스 저장
2) 직선 코스 판단

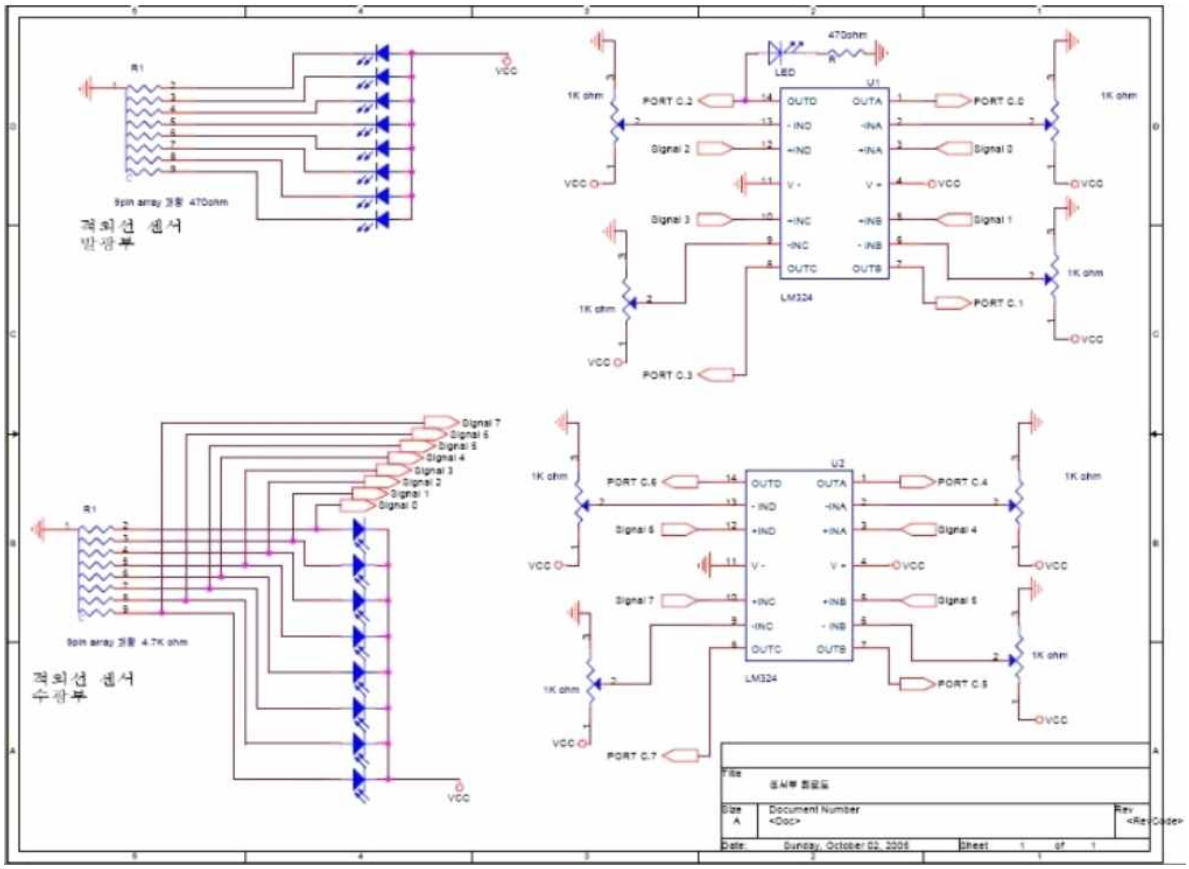
2. 제작 과정

2.1 하드웨어

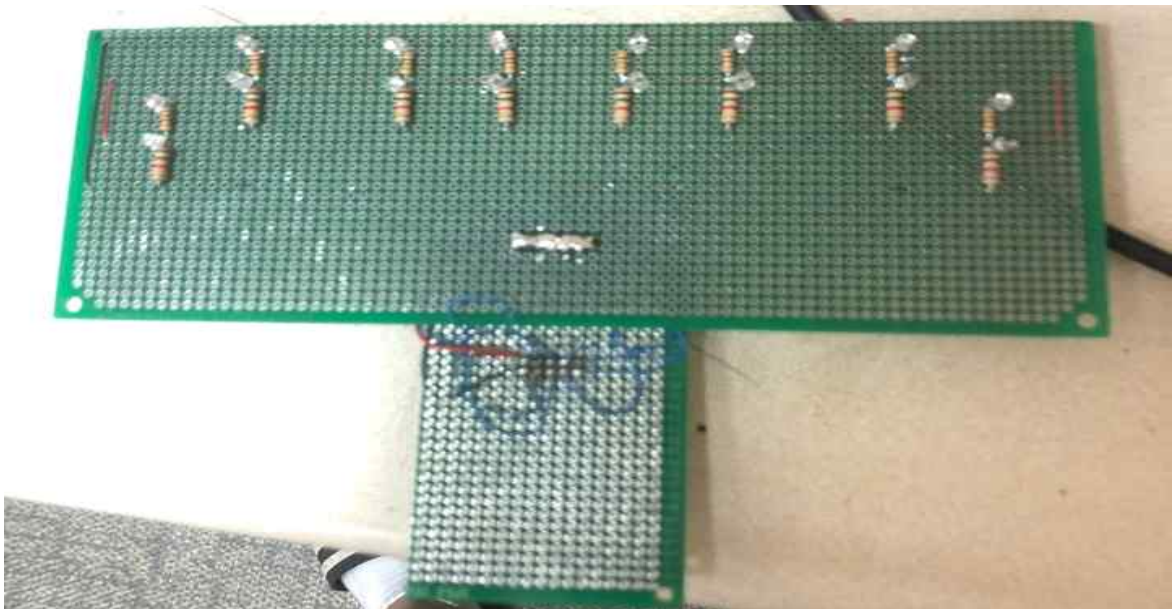
– 센서부

발광부에서 빛을 보내면 센서부에서 빛을 받아 빛의 양을 인식.

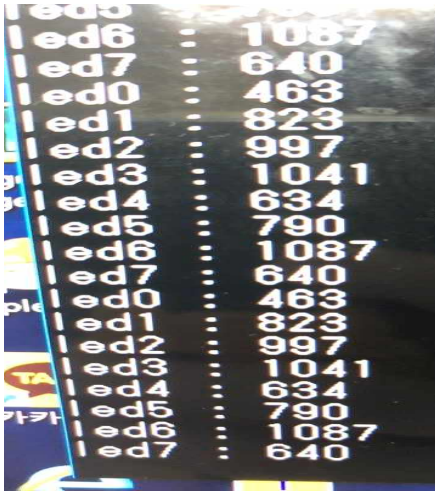
회로도



완성본



센서가 제대로 작동하는 지 확인.

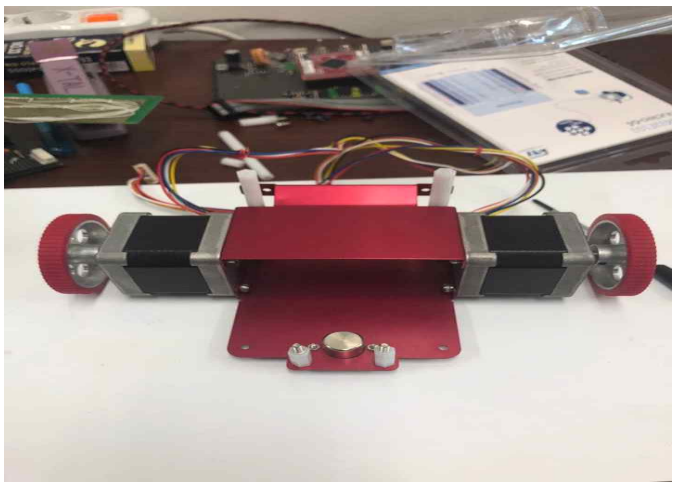


각 센서의 빛의 양을 출력하여 확인함
(센서에 따라 빛의 양을 판단하는 정도가 다름.)

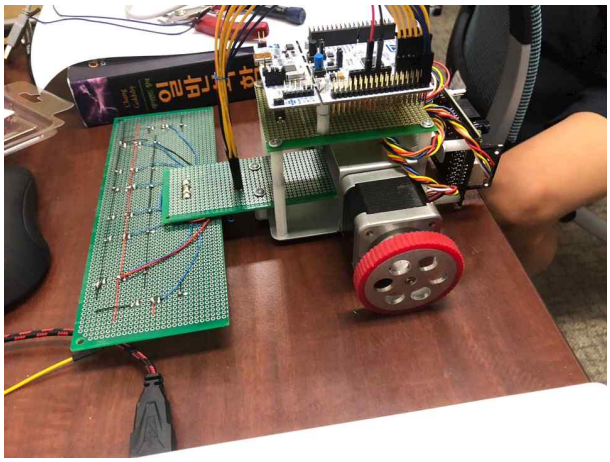
- 모터부

모터를 차체에 고정.

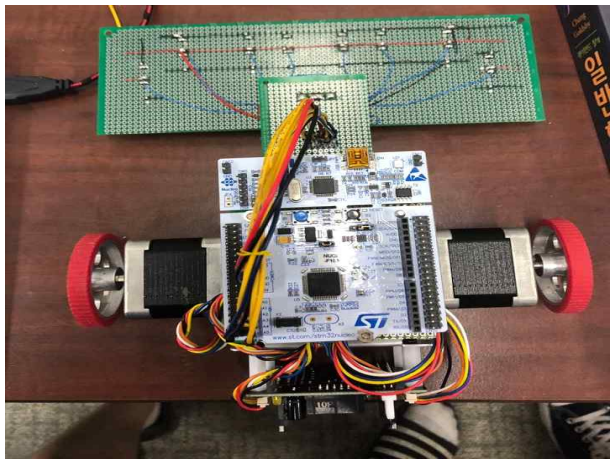
완성본



- 최종



옆에서 본 모습

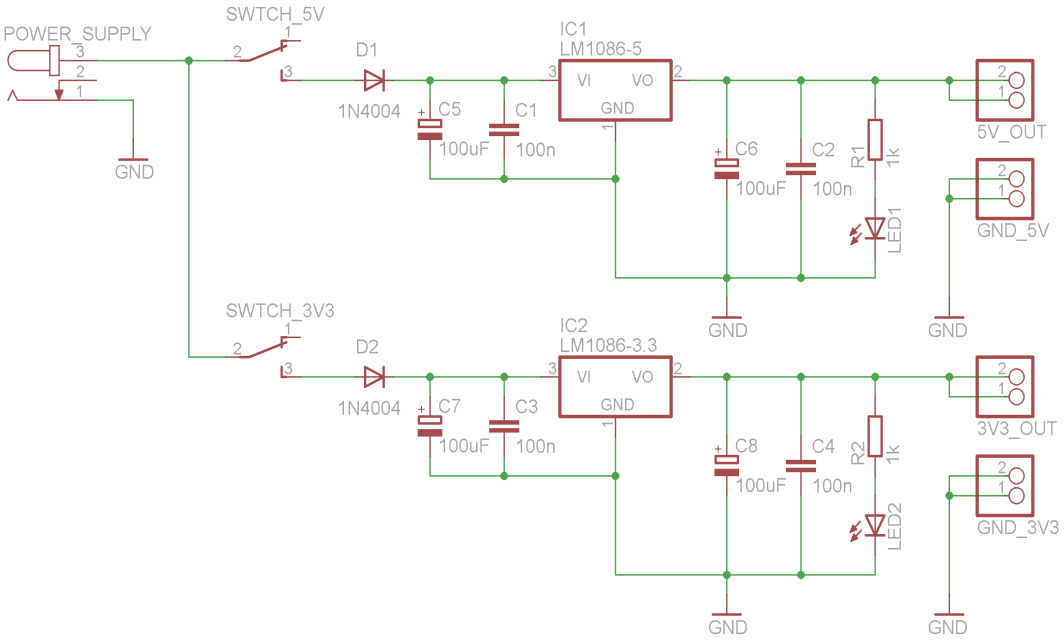


앞에서 본 모습

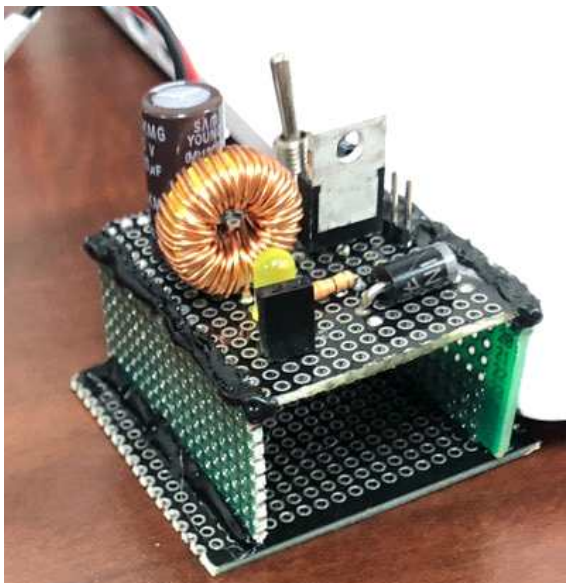
– 레귤레이터

건전지로 트레이서를 작동시키려면 메인보드에 5V를 넣어줘야하므로 12V를 3.3V로 바꿔주는 기능을 하는 레귤레이터가 필요함.

회로도



완성본

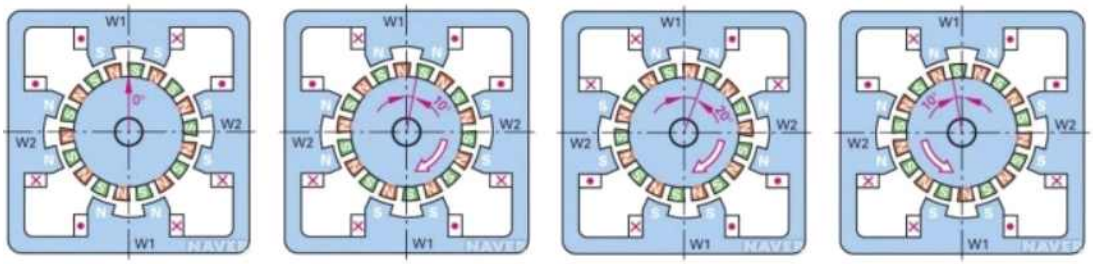


2.2 소프트웨어

-1차 주행

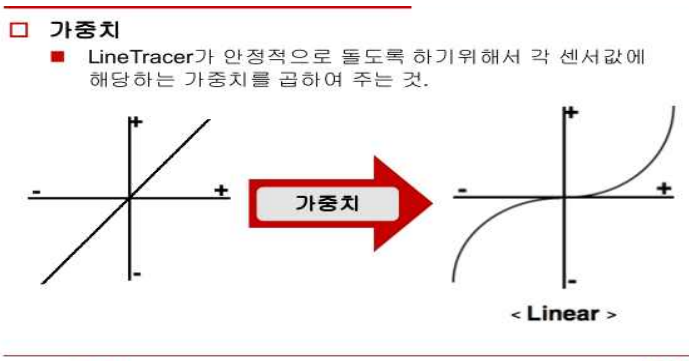
1) 검은 선과 흰선(라인) 판단.
출발하기 전 흰선과 검은선을 계속해서 읽으며 최솟값과 최대값을 업데이트 한 뒤 최대값과 최솟값의 평균을 낸 뒤 그 평균을 기준으로 평균보다 높으면 흰선 평균보다 낮으면 검은선으로 판단.

2) 스텝모터 제어



위의 그림보다 더 세분화시켜 위상을 8단계로 나눈 후 순차적으로 비트연산자를 이용하여 위상을 바꿔주어 모터를 움직인다. 위의 위상을 움직이는 것을 함수에 넣어 tikertimer를 통해 단위 시간당 함수를 호출하는 수를 조절하여 속도를 제어함.

3) 전반적인 차체 제어
-가중치 제어



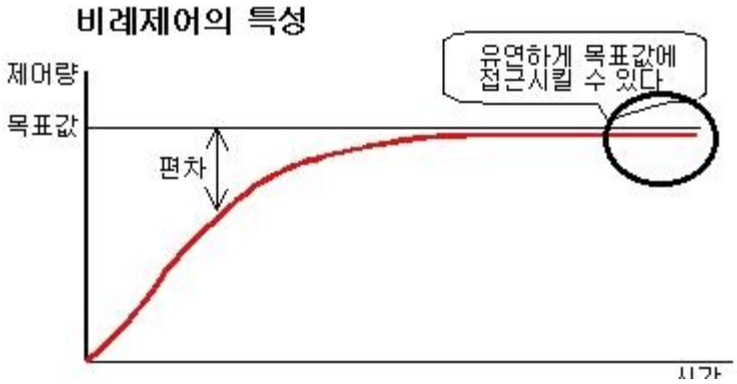
왼쪽센서가 흰선에 닿으면 왼쪽으로 오른쪽 센서가 흰선에 닿으면 오른쪽으로 이동하게 하며 함수의 호출 횟수에 적절한 가중치 상수를 곱해주어 제어한다.

단점: 운행이 불안전하며 급하게 꺽이거나 속도가 빠를 때는 경로를 이탈함.

-PID 제어

p제어 이용

왼쪽과 오른쪽에서 받은 값을 각 센서의 최댓값, 최솟값, 현재의 센서값을 이용하여 현재의 센서를 백분위로 환산한 뒤 그 차이를 직접 함수 호출 횟수에 곱해준다.



장점: 운행이 가중치보다 상대적으로 부드러우며 좌우 4쌍의 차이를 모두 이용하여 가중치를 다르게 적용하면 급하게 꺾인 구간에서도 이탈하지 않는다.

4) 교차로 판단 & 정지

가중치 제어에서는 흰선을 왼쪽과 오른쪽 모두 흰선을 인식하면 count를 하나 올리고 사용자가 정한 시간 안에 흰선을 만나면 종착지점으로 인식해 정지, 그렇지 않으면 다시 count를 내려줌.

P제어에서는 교차로에 대한 구현을 따로 하지 않아도 되며 위와 같은 방식으로 종착지점을 인식함.
(센서쌍을 여러개 사용할수록 효율이 증대.)

-2차 주행

코스 좌우에 흰 선이 한쪽에만 붙어 있는 경우 왼쪽이 연속 두 번 혹은 오른쪽이 연속 두 번 나오면 직진.

1) 1차 주행에서의 코스 저장

흰 선이 한쪽에만 붙어 있는 경우 흰 선을 읽을 때마다 선의 위치(몇 번째 위치에 있는지)를 받아들이는 동시에 왼쪽이 두 번 나오거나 오른쪽이 두 번 나올 때 뒤의 흰 선의 위치를 저장한다. 또한 그 때의 바퀴가 몇 번 돌아갔는지를 저장한다.

2) 직선 코스 판단

흰선의 위치를 계속해서 읽으면서 1차에 저장한 위치와 같은 위치에 있다면 속도를 올린다. 여기서 속도가 너무 빠르면 다시 회전 구간이 나오면 경로를 이탈 할 수 있으므로 돌아간 바퀴수의 0.8배만큼만 속도를 올리고 그 뒤는 속도를 다시 줄인다.

4. 최종 결과

4.1 주행
동영상 첨부...해야하는데 동영상이 없다

4.2 한계

PID중 P제어만 사용하여 속도나 안정성 면에서 약간 뒤떨어지고 정지나 교차를 잘못 인식하는 경우가 속도에 따라 간혹 발생한다. 속도나 안정성을 올리기 위해서는 p제어의 비례상수의 적절한 값을 제대로 찾아내거다 I제어 D제어를 함께 사용하면 된다. 또한 정지나 교차로면은 트레이서의 속도에 따라 인식 시간을 바꿔줘야하기 때문에 효율적인 알고리즘이 아니기 때문에 새로운 알고리즘을 고찰 해봐야한다.

코드

P제어 1차&2차

```
#include "mbed.h"

AnalogIn led0(A0);
AnalogIn led1(A1);
AnalogIn led2(A2);
AnalogIn led3(A3);
AnalogIn led4(A4);
AnalogIn led5(A5);
AnalogIn led6(PC_2);
AnalogIn led7(PC_3);
DigitalOut ledL(PC_5);
DigitalOut ledR(PC_6);
DigitalOut ledS(PC_8);

Ticker timer1;
Ticker timer2;
Ticker timer3;
PortOut Motor(PortA, 0xffffffff);
InterruptIn button(USER_BUTTON);
Serial pc(SERIAL_TX, SERIAL_RX);

int count1 = 0;
int count2 = 0;
int count3 = 0;
int go_number=0;
int go_number2=0;
int tmp;
int sensor[8],Max[8], Min[8],avg[8],LED[8];
int percent_L1,percent_R1,percent_L2,percent_R2;
int K=5,M=12;
int diff_1,diff_2;
int Velocity_R=1500;
int Velocity_L=1500;
int sum;
int turnmark_L=0,turnmark_R=0,turnmark_cntR,turnmark_cntL,led_cntR,led_cntL,a=0,b=0;
int num[100],num2[100],arr[50],q,x,y;
int finish=1,speed=0;
int wheelcnt[100]={0,},wheelcnt_L[100]={0,},k;
int jinoo=0,remember,u=0;

void MinMax();
void pressed();
void linetracing();
void speedlinetracing();
void calculate();
void go_R();
void go_L();
void change_R();
void change_L();
```

```
void end();
void readturnmark();
void turnmarkcalculate();
void printf();
void wheelcalculate();
void starttracing();

int main(){
    num[0]=0;
    q=-2;
    button.fall(&pressed);
    while(count2==0){
        MinMax();
        ledS=1;
        ledL=1;
        ledR=1;
        }
        pc.printf("a\n\ra\n\ra\n\r");
    ledL=0,ledR=0,ledS=0;
        Velocity_R=700,Velocity_R=700;
    timer2.attach_us(&go_R,Velocity_R);
    timer3.attach_us(&go_L,Velocity_L);
    while(u<3000){
        starttracing();
        }
    while(count2==1){
        linetracing();
        readturnmark();
            }
    speed=1;
    while(count2==2){
        ledL=1;
        ledR=1;
        ledS=1;
        q=0;
        y=0;
        }
        ledL=0;
        ledR=0;
        ledS=0;
        u=0;
        Velocity_R=700,Velocity_R=700;
    timer2.attach_us(&go_R,Velocity_R);
    timer3.attach_us(&go_L,Velocity_L);
    while(u<3000){
        starttracing();
        }
    while(count2==3){
        readturnmark();
        linetracing();
            }
```

```
        }
void MinMax(){
    sensor[0]=led0.read()*3300;
    sensor[1]=led1.read()*3300;
    sensor[2]=led2.read()*3300;
    sensor[3]=led3.read()*3300;
    sensor[4]=led4.read()*3300;
    sensor[5]=led5.read()*3300;
    sensor[6]=led6.read()*3300;
    sensor[7]=led7.read()*3300;
    if(count1==0){
        Max[0]=sensor[0];
        Max[1]=sensor[1];
        Max[2]=sensor[2];
        Max[3]=sensor[3];
        Max[4]=sensor[4];
        Max[5]=sensor[5];
        Max[6]=sensor[6];
        Max[7]=sensor[7];
        count1++;
        }
    else if(count1==1){
        Min[0]=sensor[0];
        Min[1]=sensor[1];
        Min[2]=sensor[2];
        Min[3]=sensor[3];
        Min[4]=sensor[4];
        Min[5]=sensor[5];
        Min[6]=sensor[6];
        Min[7]=sensor[7];
            for (int i = 0; i < 8; i++) {
        if (Max[i] < Min[i]) {
            tmp = Max[i];
            Max[i] = Min[i];
            Min[i] = tmp;
            }
        }
        count1++;
        }
    else{
    for (int i = 0; i < 8; i++) {
        if (sensor[i] < Min[i]) {
            Min[i] = sensor[i];
        }
        else if (sensor[i] > Max[i]) {
            Max[i] = sensor[i];
                    }
                }
            }
    for(int i=0;i<8;i++){
        avg[i]=(Min[i]+Max[i])/2;
        }
```

```
        }

void pressed(){
    count2=count2++;
    }

void linetracing(){
    calculate();
    if(speed==1&&jinoo==1&&wheelcnt_L[q]<wheelcnt[q]*0.8){
        ledS=1;
        if(sum>=0){
        Velocity_L=320;
        Velocity_R=320+sum;
        }
    else{
        Velocity_L=320-sum;
        Velocity_R=320;
        }
        }
    else{
    if(sum>=0){
        ledS=0;
        Velocity_L=450;
        Velocity_R=450+sum;
        }
    else{
        Velocity_L=450-sum;
        Velocity_R=450;
        }
    }

    LED[1]=led1.read()*3300;
    LED[2]=led2.read()*3300;
    LED[3]=led3.read()*3300;
    LED[4]=led4.read()*3300;
    LED[5]=led5.read()*3300;
    LED[6]=led6.read()*3300;

    if(LED[1]>avg[1]&&LED[2]<avg[2]&&LED[5]<avg[5]&&LED[6]>avg[6]){
            count3++;
            if(count3==30){
                end();
                count3=0;
                turnmarkcalculate();
                }
            }
    if(LED[1]<avg[1]&&LED[6]<avg[6]){
        count3=0;
        }
        }

void calculate(){
```

```
    percent_L1=(led3.read()*3300-Min[3])/(Max[3]-Min[3])*100;
    percent_R1=(led4.read()*3300-Min[4])/(Max[4]-Min[4])*100;
    percent_L2=(led2.read()*3300-Min[2])/(Max[2]-Min[2])*100;
    percent_R2=(led5.read()*3300-Min[5])/(Max[5]-Min[5])*100;
    diff_1=(percent_L1-percent_R1)*K;
    diff_2=(percent_L2-percent_R2)*M;
    sum=diff_1+diff_2;
    }
void go_L(){
    change_R();
    if(go_number==0){
        Motor=(0xfffffe1f&Motor)|0x120;
        }
    else if(go_number==1){
        Motor=(0xfffffe1f&Motor)|0x100;
        }
    else if(go_number==2){
        Motor=(0xfffffe1f&Motor)|0x180;
        }
    else if(go_number==3){
        Motor=(0xfffffe1f&Motor)|0x80;
        }
    else if(go_number==4){
        Motor=(0xfffffe1f&Motor)|0xc0;
        }
    else if(go_number==5){
        Motor=(0xfffffe1f&Motor)|0x40;
        }
    else if(go_number==6){
        Motor=(0xfffffe1f&Motor)|0x60;
        }
    else if(go_number==7){
        Motor=(0xfffffe1f&Motor)|0x20;
        }
    go_number++;
    if(go_number==8){
        go_number=0;
        }
    }

void go_R() {
    change_L();
    if (go_number2 == 0) {
        Motor = (0xffffe1ff & Motor) | 0x1200;
    }
    else if (go_number2 == 1) {
        Motor = (0xffffe1ff & Motor) | 0x1000;
    }
    else if (go_number2 == 2) {
        Motor = (0xffffe1ff & Motor) | 0x1800;
    }
    else if (go_number2 == 3) {
```

```
        Motor = (0xffffe1ff & Motor) | 0x800;
    }
    else if (go_number2 == 4) {
        Motor = (0xffffe1ff & Motor) | 0xc00;
    }
    else if (go_number2 == 5) {
        Motor = (0xffffe1ff & Motor) | 0x400;
    }
    else if (go_number2 == 6) {
        Motor = (0xffffe1ff & Motor) | 0x600;
    }
    else if (go_number2 == 7) {
        Motor = (0xffffe1ff & Motor) | 0x200;
    }
    go_number2++;
    if (go_number2 == 8) {
        go_number2 = 0;
    }
    if(speed==0){
    if(k==1){
    wheelcnt[q]++;
    }
    }
    if(speed==1){
    if(k==1){
    wheelcnt_L[q]++;
        }
        }
}

void change_L() {
    timer2.detach();
    timer2.attach_us(&go_R,Velocity_R);
    }

void change_R(){
    timer3.detach();
    timer3.attach_us(&go_L,Velocity_L);
    }

void end(){
    timer2.detach();
    timer3.detach();
    }

void readturnmark(){
    LED[0]=led0.read()*3300;
    LED[1]=led1.read()*3300;
    LED[6]=led6.read()*3300;
    LED[7]=led7.read()*3300;
    if(LED[2]>avg[2]&&LED[5]>avg[5]){
        b=500;//700절대안됨
```

```
        }
    if(b==0){
    if(LED[7] > avg[7]&&LED[0] < avg[0]){
        turnmark_cntR=100;
        ledR=1;
        k=0;
        }
    if(LED[0] > avg[0]&&LED[7] < avg[7]){
        turnmark_cntL=100;
        ledL=1;
        k=0;
        }
    if(turnmark_cntR>0&&turnmark_cntR<101){
        turnmark_cntR--;
        }
    if(turnmark_cntL>0&&turnmark_cntL<101){
        turnmark_cntL--;
        }
    if(turnmark_cntR==0){
        q++;
        num[q]=q;
        k=1;
        if(speed==1){
            if(q==arr[y]){
                jinoo=1;
                y++;
                }
            else{
                jinoo=0;
                }
            }
        num2[q]=2;
        pc.printf("       num : %d      R\n\r",num[q]);
        turnmark_cntR=101;
        ledR=0;
        }
    if(turnmark_cntL==0){
        q++;
        num[q]=q;
        k=1;
        if(speed==1){
            if(q==arr[y]){
                jinoo=1;
                pc.printf("%d ==  %d",q,arr[y]);
                y++;
                }
            else{
                jinoo=0;
                }
            }
        num2[q]=1;
        pc.printf("       num : %d      L\n\r",num[q]);
```

```
            turnmark_cntL=101;
            ledL=0;
            }
        }
        if(b!=0){
            b--;
            }
}
void turnmarkcalculate(){
        x=1,y=0;
        while(x<43){
        if(num2[x]+num2[x+1]==3){
        x=x+1;
        }
        else{
        arr[y]=x+1;
        y++;//y=15
        x=x+2;
        }
        }
        for(int w=0;w<y;w++){//15개
            printf("stop mark : %d \n\r",arr[w]);
            }
        remember=y;
        }
void printf(){
        }
void starttracing(){
        calculate();
                if(sum>=0){
            Velocity_L=600;
            Velocity_R=600+sum;
            }
        else{
            Velocity_L=600-sum;
            Velocity_R=600;
            u++;
        }
}
```