


공학설계입문

프로젝트 명	장애물 회피 라인트레이싱
팀 명	갈피를 못잡쥬~
문서 제목	장애물 회피 라인트레이싱 보고서

Version	1.3
Date	2017.11.26

팀원	박 정 규 (조장)
	남 경 태
	박 태 범
지도교수	한광수 교수

	중간보고서		
	프로젝트 명	장애물회피라인트레이싱	
	팀 명	갈피를 못잡조~	
	Confidential Restricted	Version 1.3	2017-Nov-16


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 공학설계입문 수강 학생 중 프로젝트 “장애물 회피 라인트레이싱”를 수행하는 팀 “갈피를 못잡조~”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “갈피를 못잡조~”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역


Filename	중간보고서- 장애물회피라인트레이싱 보고서.doc
원안작성자	박정규, 박태범, 남경태, 연개소문
수정작업자	박정규, 박태범, 남경태

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2017-11-15	박정규	1.0	최초 작성	과제 분석 및 알고리즘 작성
2017-11-20	박정규	1.1	내용 수정	수정된 코드내용 추가
2017-11-22	박정규	1.2	내용 수정	평가결과 추가
2017-11-26	박태범	1.3	개인별 수정	느낀점 추가

	국민대학교 컴퓨터공학부 공학설계입문	중간보고서		
		프로젝트 명	장애물회피라인트레이싱	
		팀 명	갈피를 못잡조~	
		Confidential Restricted	Version 1.3	2017-Nov-16


목 차

1	프로젝트 개요.....	4
2	프로젝트 분석.....	5
2.1	요구사항 분석.....	5
2.2	모듈 구조.....	5
3	설계.....	6
3.1	메서드 설계.....	6
4	구현 및 평가결과.....	7
4.1	프로그램 구현.....	7
4.2	평가결과.....	8
5	결론 및 건의사항.....	8

	중간보고서		
	프로젝트 명	장애물회피라인트레이싱	
	팀 명	갈피를 못잡조~	
	Confidential Restricted	Version 1.3	2017-Nov-16

1 프로젝트 개요

1. 흰 바닥에 검정색 테이프로 이뤄진 라인을 따라 주행한다.
 - 1-1 테이프의 너비는 5 방향 추적 센서의 LED 두 개에서 세 개 길이이다.
 - 1-2 주행 라인은 4 개의 직선 부분과 4 개의 곡선 부분으로 이루어져 있고, 직선 코스와 곡선 코스가 번갈아가며 나오는 코스이다.
2. 장애물을 회피하여 주행한다.
 - 2-1. 장애물은 뽀빠로 상자정도의 크기이며, 직선 코스의 중앙 부분에 놓여 있다.
3. 가능한 한 빠른 시간 내로 완주한다.

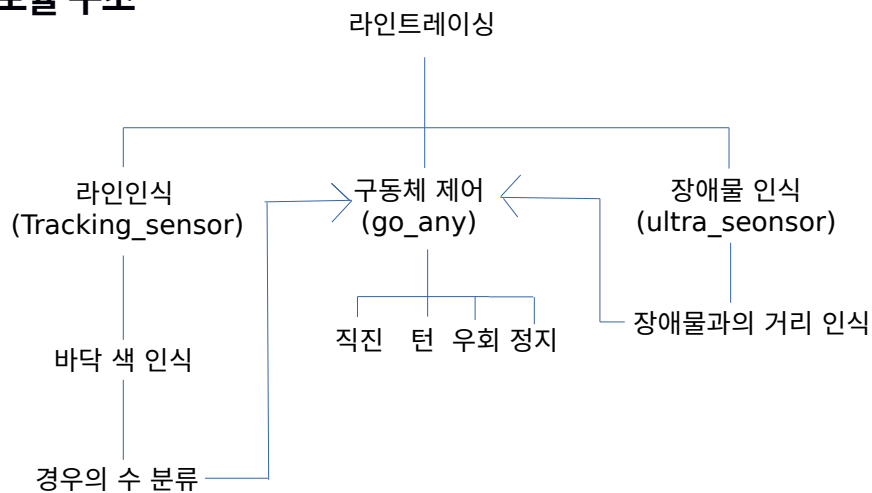
	중간보고서		
	프로젝트 명	장애물회피라인트레이싱	
	팀 명	갈피를 못잡죠~	
	Confidential Restricted	Version 1.3	2017-Nov-16


2 프로젝트 분석

2.1 요구사항 분석

- 라인 인식이 가능한 모듈 구현
 - 센서로 구동체가 어느 쪽으로 위치하는지 파악
- 직진, 회전 등 구동체 이동이 가능한 모듈 구현
 - 파악된 위치에 따라 구동체가 라인을 따라 이동할 수 있도록 움직임 제어
- 전/후진 시 장애물 인식이 가능한 모듈 구현
 - 장애물을 인식하면 장애물 회피 후, 다시 라인을 따라 주행

2.2 모듈 구조



	국민대학교 컴퓨터공학부 공학설계입문	중간보고서		
		프로젝트 명	장애물회피라인트레이싱	
		팀 명	갈피를 못잡조~	
		Confidential Restricted	Version 1.3	2017-Nov-16

3 설계

3.1 메서드 설계

1. 라인 인식 모듈 (Tracking_Sensor)

List를 사용하여 listcalc라는 변수에 현재 바닥 라인의 대한 정보를 표시하도록 하였다.
List에 대한 경우의 수는 다음과 같다.

```

case1) [0, 0, 0, 0, 0]
    → listcalc = Last
case2) [1, 1, 0, 1, 1]
    → listcalc = Center
case3) [1, 0, 0, 0, 1]
    → listcalc = Center
case4) [1, 0, 0, 1, 1]
    → listcalc = Left
case5) [1, 1, 0, 0, 1]
    → listcalc = Right
case6) [1, 0, 1, 1, 1]
    → listcalc = Left
case7) [1, 1, 1, 0, 1]
    → listcalc = Right
case8) 첫번째 원소가 0
    → listcalc = MostLeft
case9) 마지막 원소가 0
    → listcalc = MostRight
case10) [1, 1, 1, 1, 1]
    → listcalc = Out

```


2. 구동체 이동 및 회전 모듈 (go_any)

구동체를 이동, 회전 시키기 위해 총 4개의 함수를 작성하였다.

- 직진하는 함수를 2가지 작성하였고, 둘의 차이는 구동시간의 제한여부이다.
- 회전에 관한 함수는 방향을 받게 하여 어떤 방향인지에 따라 오른쪽, 왼쪽 모터의 속도를 조절하여 회전을 하도록 만들었다.
- 위에 있는 라인인식 모듈의 함수를 가져와 트랙의 상태를 읽은 다음에 구동체의 좌우 바퀴 속도를 적절히 변경하여 라인을 제대로 따라가게끔 구현하였다.

3. 장애물 인식 모듈(Ultrawave_Sensor)

초음파 센서를 이용해서 장애물과의 거리를 측정한 후 그 값을 반환하여 주는 함수를 작성하였다.

	중간보고서		
	프로젝트 명	장애물회피라인트레이싱	
	팀 명	갈피를 못잡죠~	
	Confidential Restricted	Version 1.3	2017-Nov-16

4 구현 및 평가결과

4.1 프로그램 구현

```
from go_any import *
from Tracking_sensor import *
from ultrawave_sensor import *
```

try:

초기 셋팅

```
LeftPwm.start(0)
RightPwm.start(0)
d = getDistance()
s = getTrack()
```

위에 작성해 놓은 모듈을 이용하기 위해 import 하고 원활한 실행을 위해 초기환경을 설정해 주었다.

주행 시작

```
while (d > 18):
    d = getDistance()
    print (d)
    s = getTrack()
    print (s)
    linetrace(forward,s,50)
    time.sleep(0.001)
```

장애물을 만나기 전까지는 라인을 따라 주행하기 위해 while 문을 사용했고, 장애물과의 거리를 반복적으로 확인하게 하였다.

장애물 감지.

```
avoid('left', 0.7, 40, 0.7, 40, 0.7, 40)
s = getTrack()
```

장애물 회피.


```
while (s == 'out'):
    go_any(forward,40,40)
    s = getTrack()
    time.sleep(1)
```

장애물과의 거리가 줄어들어 반복문을 빠져나가게 된다면 구현해둔 avoid 모듈(아래 참고)을 통해 장애물을

회피하게 하였다. 또한 장애물을 피한 후에 구동체가 다시 라인에 돌아올 수 있도록 만들어주었다.

```
# =====
def avoid(direction, ftime, ftspeed, gotime, gospeed, stime, stspped):
    ### 회전방향, 첫번째 회전의 시간,속도, 직진의 시간,속도, 두번째 회전의 시간, 속도를 인자로 받는다.
    ### 회전방향대로 첫번째 회전의 조건대로 회전을 받고, 두번째 직진 시간만큼 직진 한 뒤, 두번째 회전의 조건대로 회전을 실행한다.
    if direction == 'right':
        PointTurn('right', ftspeed, ftime)
        go(forward, gospeed, gospeed, gotime)
        stop()
        time.sleep(1)
        PointTurn('left', stspped, stime)
    elif direction == 'left':
        PointTurn('left', ftspeed, ftime)
        go(forward, gospeed, gospeed, gotime)
        stop()
        time.sleep(1)
        PointTurn('right', stspped, stime)
# =====
```

참고-avoid 모듈

	중간보고서		
	프로젝트 명	장애물회피라인트레이싱	
	팀 명	갈피를 못잡쥬~	
	Confidential Restricted	Version 1.3	2017-Nov-16

4.2 평가결과

라인트레이싱은 성공적으로 완수했으나, 장애물 회피 이후 라인으로 되돌아 오는 과정에서 속도가 급격히 올라가는 문제가 발생, 이를 해결해야 한다고 생각함.

장애물을 회피할 때 있어 회전의 정도가 계속해서 바뀌는 문제가 발생하였다.

5 결론 및 건의사항

이번에도 저번과 마찬가지로 환경에 따라 결과가 많이 달라졌다. 판에 테이프를 붙이는 것부터 수작업이다보니 판에 따라서 결과가 다르게 나왔다. 실제로 연습용 판에서 정상작동을 확인하고 시험용 판에서 작동을 시켜보니 장애물을 피하는 부분에서 연습과 다른 결과가 나타났다. 하지만 모든 결과가 다르게 나온 것은 아니었다. 라인트레이싱 같은 경우는 연습용과 시험용 판 다를 것 없이 정상실행이 되었다. 올해에는 주변환경에 많은 영향을 받는 회전을 많이 사용했지만 다음에는 이런 것을 줄이고 환경에 영향을 적게 받는 라인트레이싱같은 테스트 항목을 늘려주었으면 하는 바람이 있다. 또한 구동체가 완벽하게 작동을 하는 것을 평가하는 것이 아니라 어느정도 융통성을 갖고 평가하되 주된 평가가 구동체가 아닌 코드의 완성도로 평가되었으면 좋겠다. 시도때도 없이 달라지는 결과보다는 비교적 객관적으로 평가할 수 있지 않을까? 하는 생각이 들었다. 이미 완성된 코드에서 모터의 속도와 시간을 조금씩 변경하며 평가하는 것보다는 효율적으로 코드를 짜는 법을 알려주고 그 완성도에 따라 평가를 하는 것이 맞다고 생각한다.