소프트웨어 제출

소스 코드 설명 순서

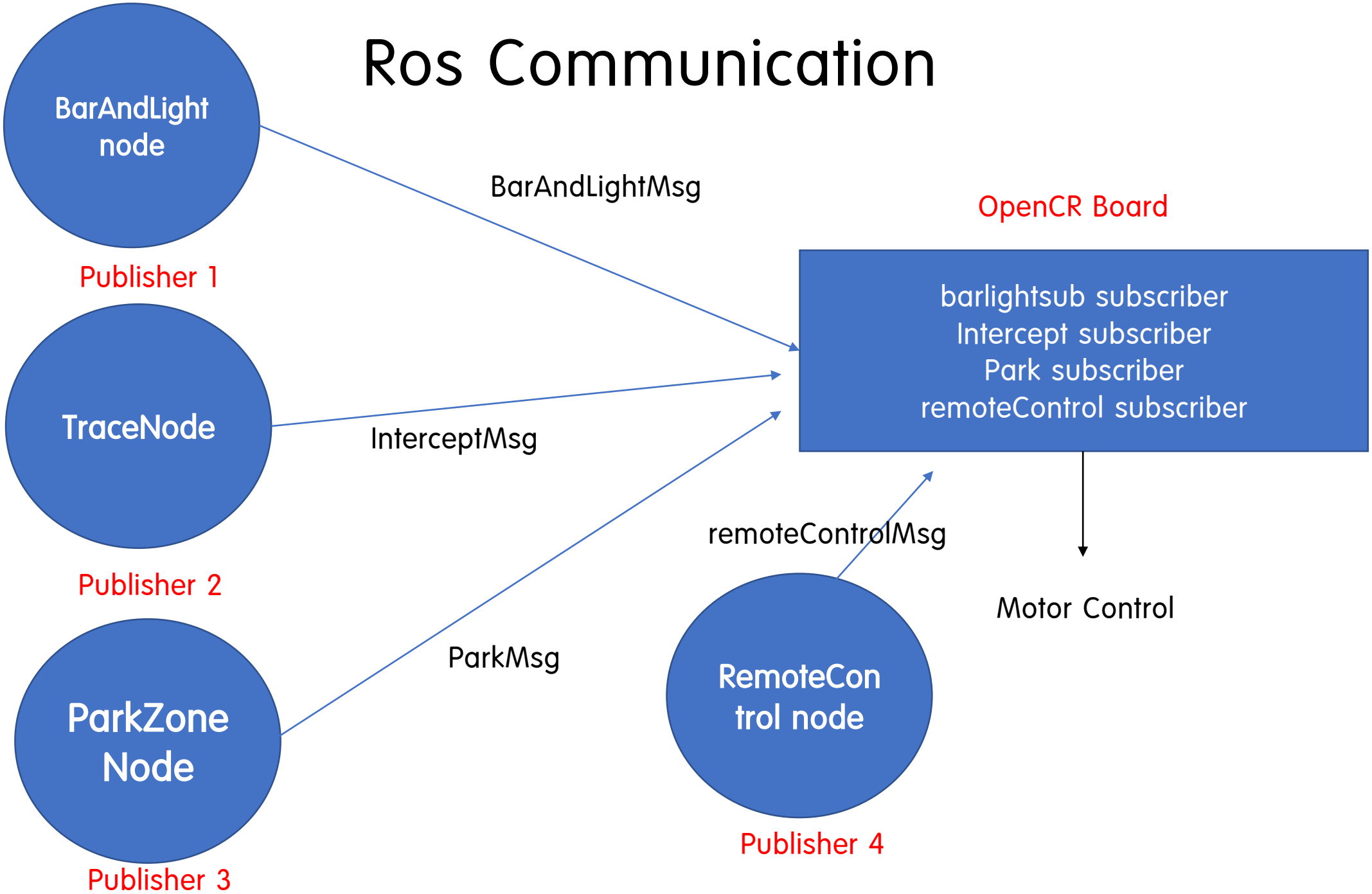
0. Ros Communication
1. 라인 트레이싱
2. 주차 미션
3. 터널 탈출 미션
4. 차단바 미션
5. 신호등 미션

국민대학교 보노보노팀

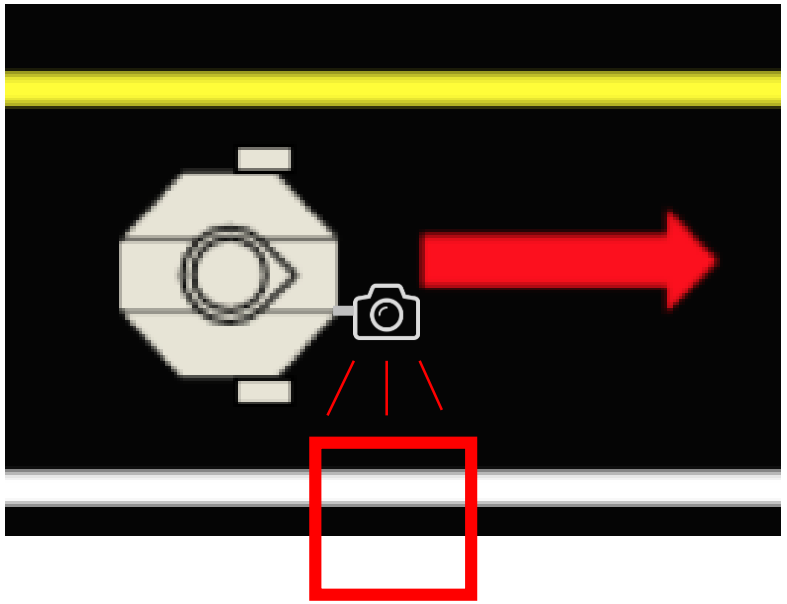
Github link

: https://github.com/Gaeul/BonobonoTurtlebot

Ros Communication



1. 라인 트레이싱

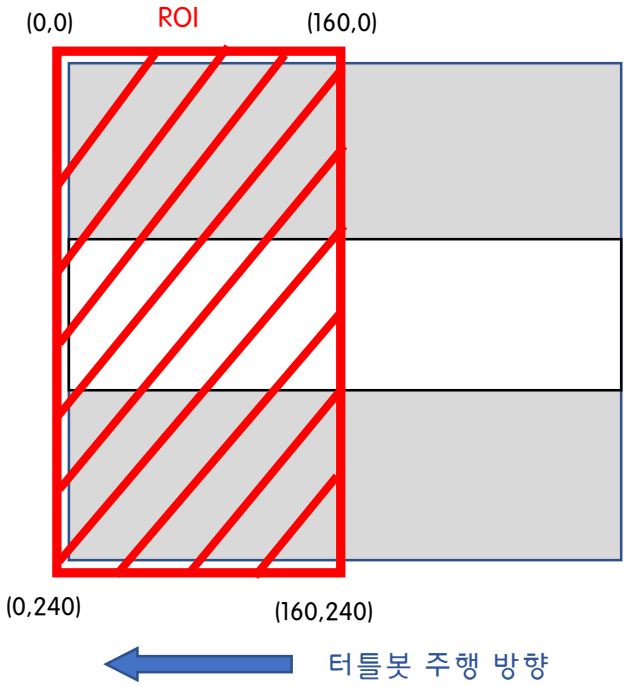


(1) 카메라 장착의 제한 사항

대회측에서 터널 입구 높이를 25cm로 제한하여 이미 20cm의 크기를 갖는 터틀봇에 카메라를 위로 장착하는데는 한계가 있다.

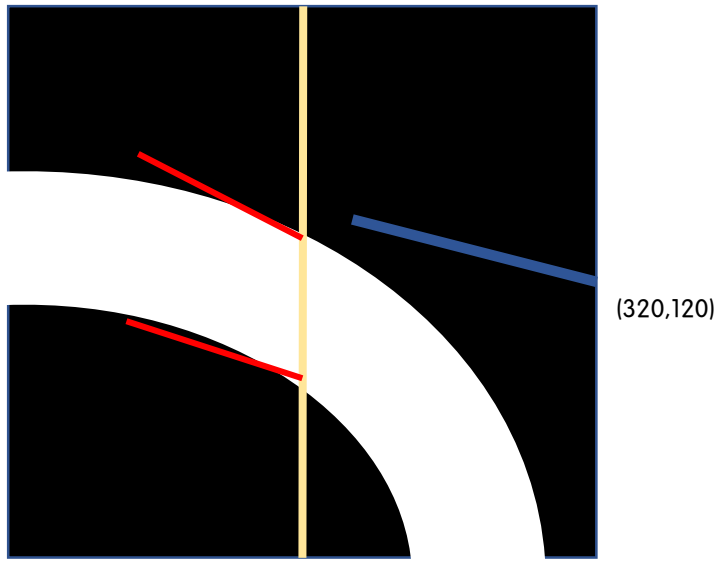
그리하여 두개의 선을 카메라 한개로 보고 가기에는 무리가 있다고 판단하여 몸체의 오른쪽 흰색 라인 하나만 사용하여 움직이게 하였다.

1. 라인 트레이싱



(1) 관심 영역 설정

320x240의 프레임 화면 중 터틀봇 진행 방향에서 조금 앞쪽을 사용하여 미리 판단하기 위하여
앞쪽(0,0 ~ 160,240)을 관심영역으로 설정하였다.



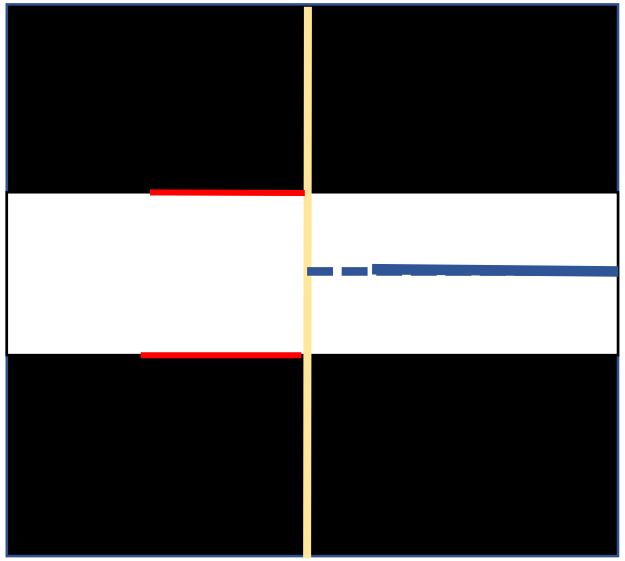
(2) 기울기를 이용한 알고리즘

OpenCV의 HoughLineP 함수를 이용하여
관심 영역 내의 모든 검출될 수 있는 직선들을
Vector에 넣어놓고, x좌표가 160인 지점을 지나는
직선들의 평균 기울기(avg)를 구한다.

그 후 (320,120)을 지나고 avg을 기울기로 갖는
직선을 긋는다. y = avg (x- 320) + 120

이 직선이 x = 160을 지날 때의
y 값(avg *(-160) + 120))을 기준으로 터틀봇의
진행 방향을 결정한다.

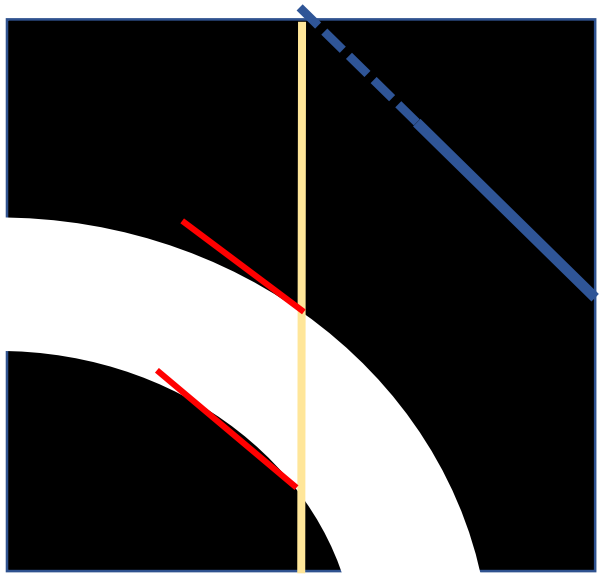
1. 라인 트레이싱



y value = 120

120에서 +- 10인 경우

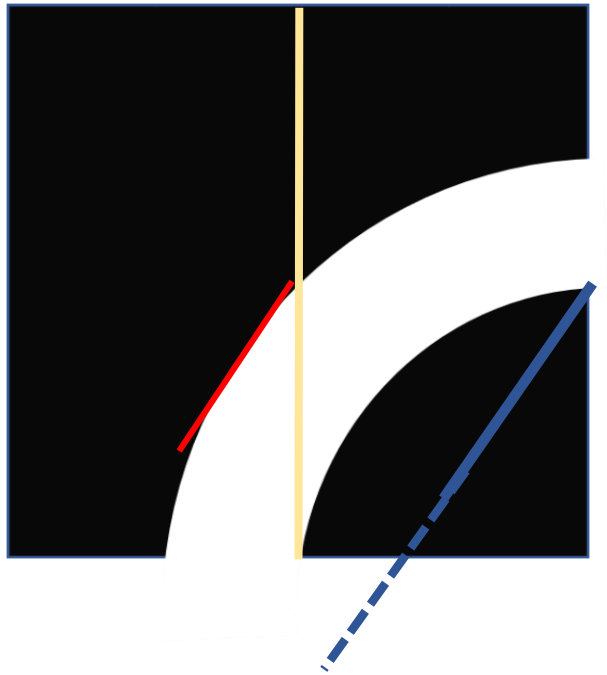
직진



y value = 0

0보다 작은 경우
0으로 취급

좌회전



y value = 240

240보다 큰 경우
240으로 취급

우회전

1. 라인 트레이싱

모터 제어 : PID 제어 방식

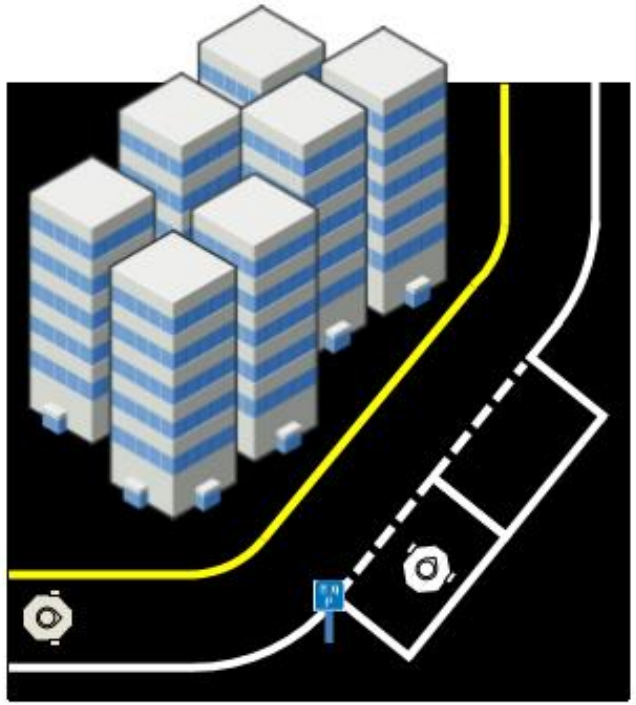
터틀봇의 모터를 제어하기 위해서 자동제어 방식 중 pid제어방식을 이용하였다.
차선이 직선일때 y value의 값이 120이 있고 차선의 기울기가 변할때 y value가 변하는걸 이용해서 목표값인 120과 현재 값을 빼서 오류량을 계산하고 이를 이용하여 모터의 조향을 주었다.
또한 잔류편차를 누적하여 조작량을 증가하여 편차를 없애는 i제어를 이용하였다.
또한 직선의 기울기의 변화에 빠르게 대처하기 위해서 미분제어인 d제어를 이용하였다.

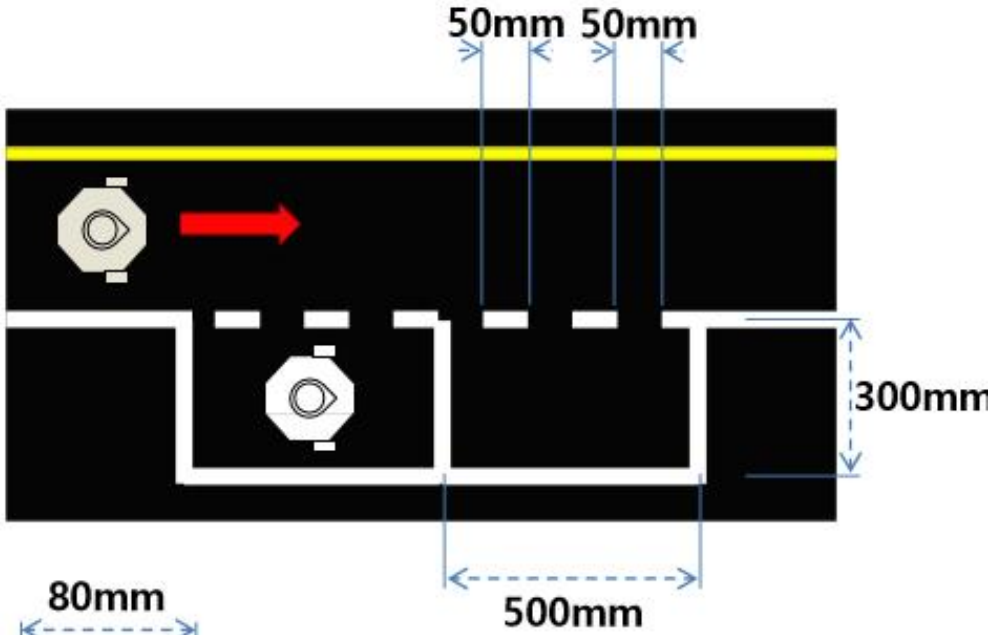
turtlebot

```
/*
 pid 제어를 위한 각종 변수 선언.
 */
double Kp = 1;
double Ki = 2.5;
double Kd = 3.2;
double error;
double error_previous;
double desired_value = 120;
double current_value = Intercept;
double Time = 0.004;


double P_control, I_control, D_control;
double PID_control;

/*
 모터를 차선의 y절편의 값에 따라, pid제어를 통해서 도로를 주행하게 해주는 함수
 */
void pid(){
  current_value = Intercept;
  error = desired_value - current_value;
  P_control = Kp * error;
  I_control = I_control + Ki * error * Time;
  D_control = Kd * (error - error_previous) / Time;


  PID_control = P_control + I_control + D_control;
  PID_control = constrain(PID_control, 0, 250);
  controlMotor(130+PID_control,130-PID_control);
  error_previous = error;
}
```

2. 주차 미션

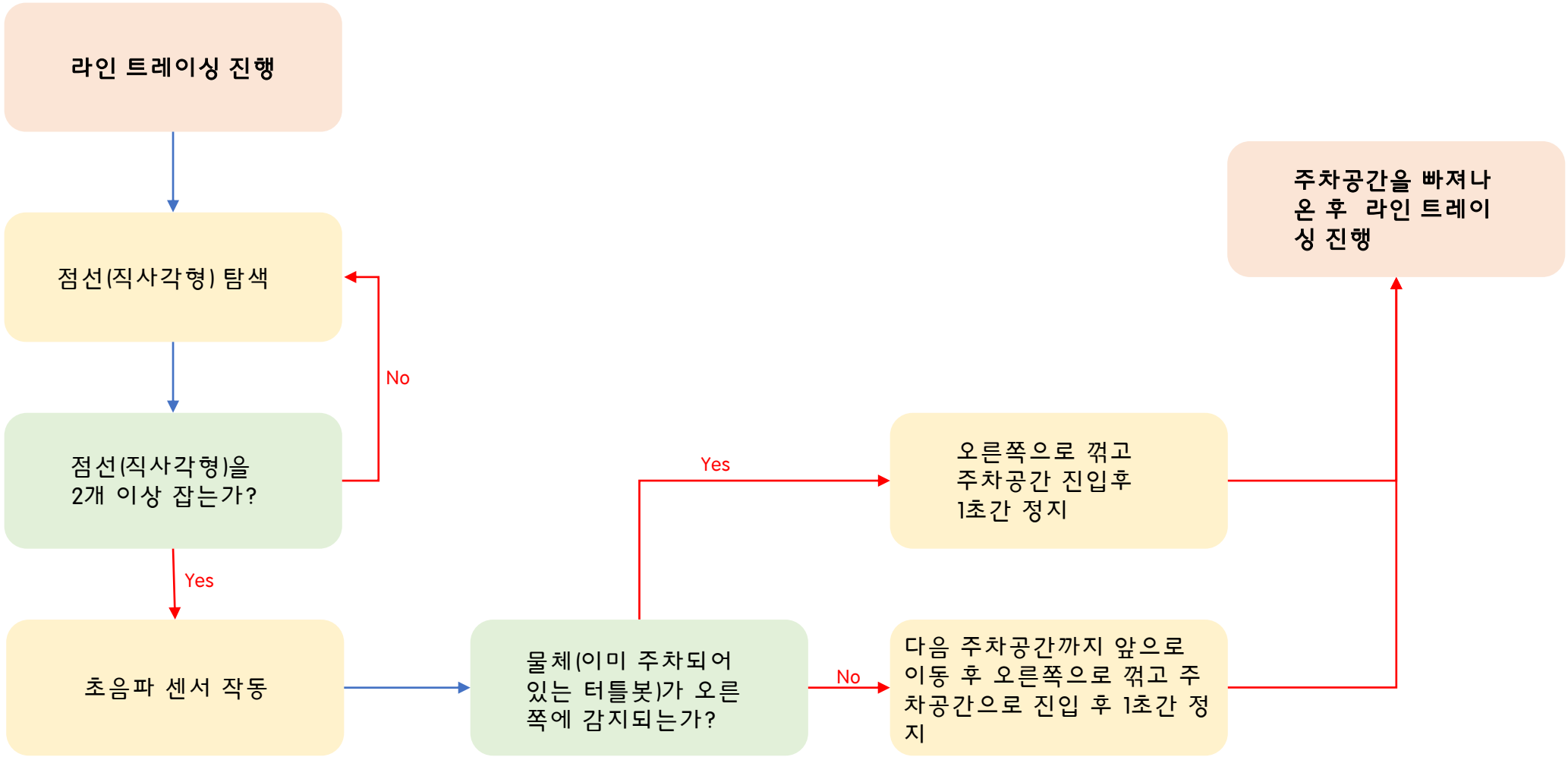
(1) 주차 공간 예시





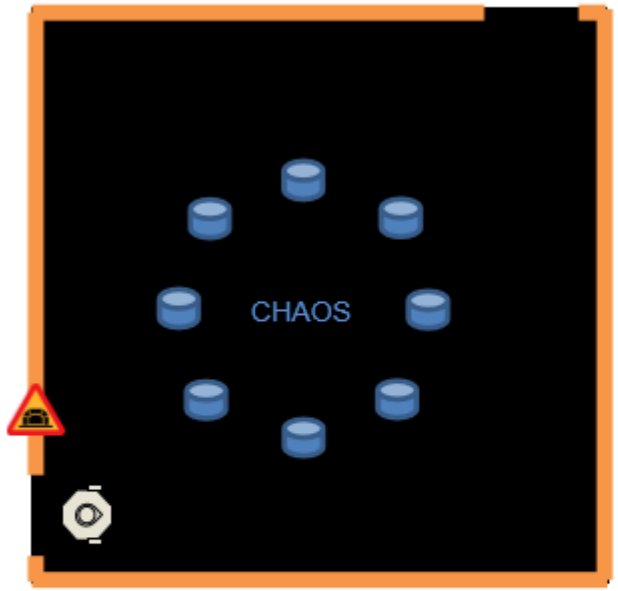
2. 주차 미션

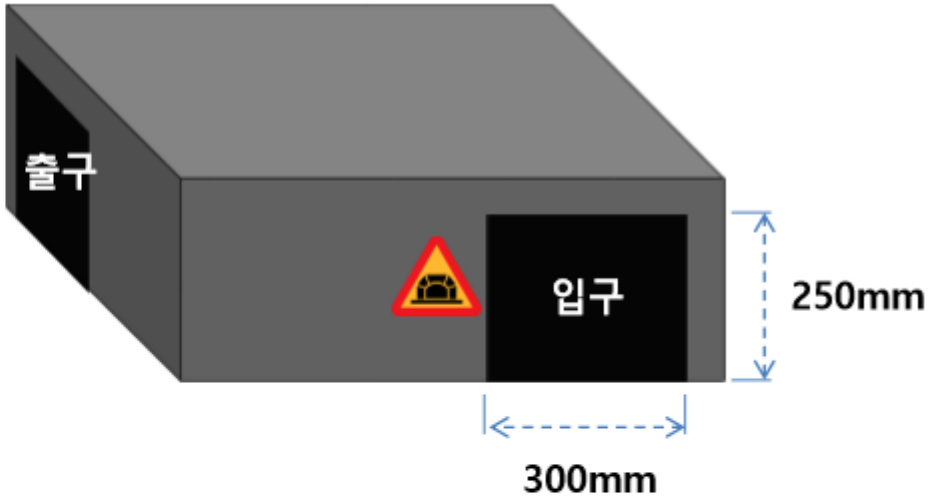
(2) 알고리즘


라인 트레이싱 진행
점선(직사각형) 탐색
점선(직사각형)을
2개 이상 잡는가?
No
Yes
초음파 센서 작동
물체(이미 주차되어
있는 터틀봇)가 오른
쪽에 감지되는가?
Yes
오른쪽으로 꺾고
주차공간 진입후
1초간 정지
No
다음 주차공간까지 앞으로
이동 후 오른쪽으로 꺾고 주
차공간으로 진입 후 1초간 정
지
주차공간을 빠져나
온 후 라인 트레이
싱 진행

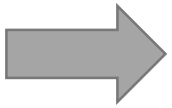
3. 터널 탈출 미션

(1) 터널안에 장애물





3. 터널 탈출 미션

(2) 현재 사용 알고리즘  미로탈출 알고리즘 우수법 사용

미로를 탈출하는 방법 중 가장 대표적인 방법은 우수법과 좌수법이 있다. 미로를 탈출할 때 오른쪽 벽이나 왼쪽 벽을 손으로 짚고 나간다면 미로를 탈출할 수 있다는 방법이다. 이것이 가능한 이유는 입구와 출구가 연결되어있기 때문이다. 입구와 출구가 같은 벽면에 연결되어있는 경우 한쪽 벽면을 따라가게 된다면 그 벽면에 연결된 모든 장소를 탐색하게 되어 출구가 있는 장소도 알아낼 수 있다.

예를들어 우수법에 의한 교차로에서 이동방향의 우선순위는

1. 오른쪽 2. 직진 3. 왼쪽 순이 된다.

3. 터널 탈출 미션

(2) 현재 사용 알고리즘 → 미로탈출 알고리즘 우수법 사용

1. 터틀봇이 오른쪽으로 이동할 수 있는 경로를 만나면 오른쪽으로 90도 회전 후 앞으로 이동

2. 터틀봇이 왼쪽과 정면으로 이동할 수 있을 경우 앞으로 이동

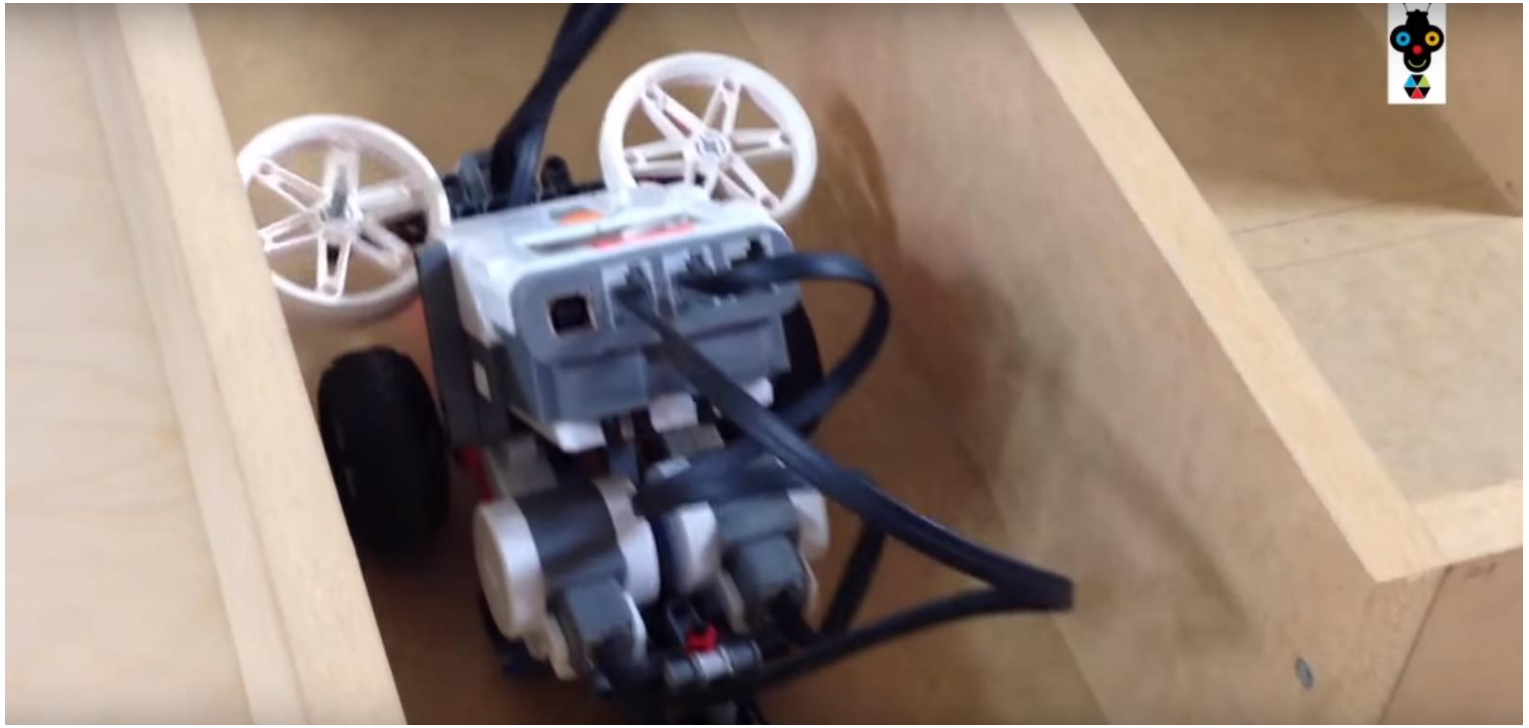
3. 터틀봇이 왼쪽으로만 움직일수 있을 경우 왼쪽으로 90도 회전 후 앞으로 이동

4. 터틀봇이 막다른 길을 만날경우 180도 회전 후 왔던길을 되돌아감

5. 조도센서 값이 올라가면 터널밖이라고 인식하고 라인트레이싱 실행

3. 터널 탈출 미션

(3)-α. 추가 하드웨어 장착



정확한 90도 회전이 안되므로, 가이드휠을 부착하여 이동을 원활하게함

3. 터널 탈출 미션

(3)-b. 추가 하드웨어 장착

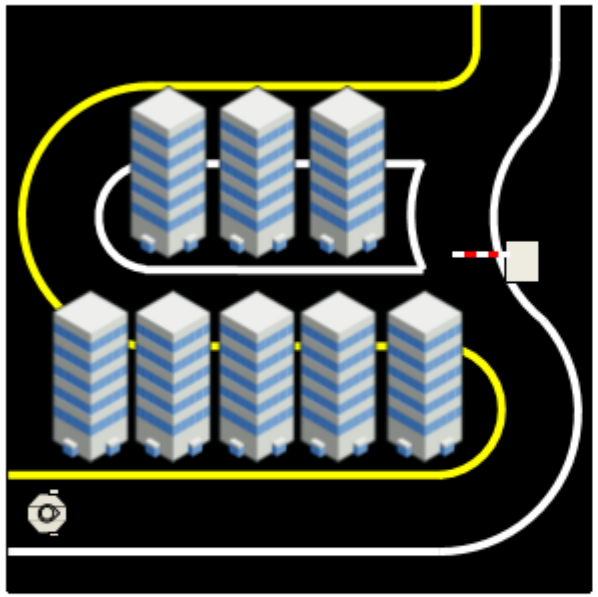
조도센서

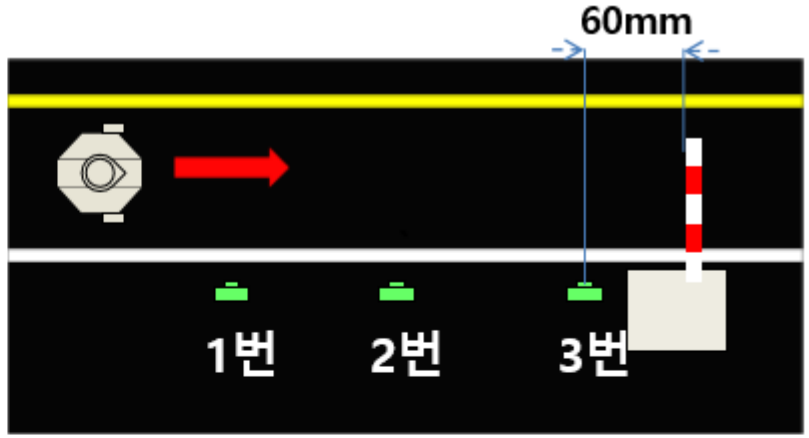
LED

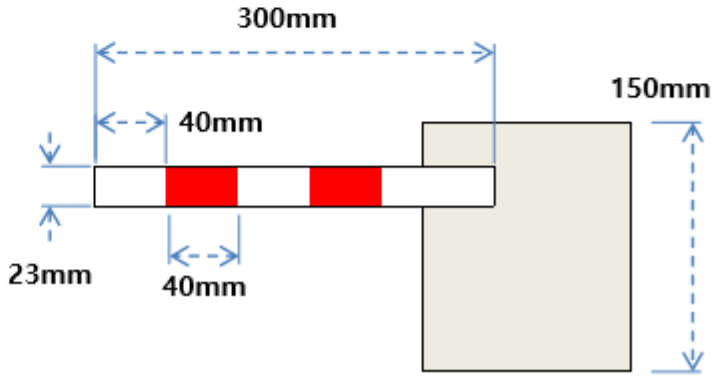
조도센서를 이용하여 터널에 진입여부를 판단하고 진입 시 LED의 불을 키고, 터널을 탈출하면 LED의 불을 끈다.

4. 차단바 미션

(1) 차단바 예시





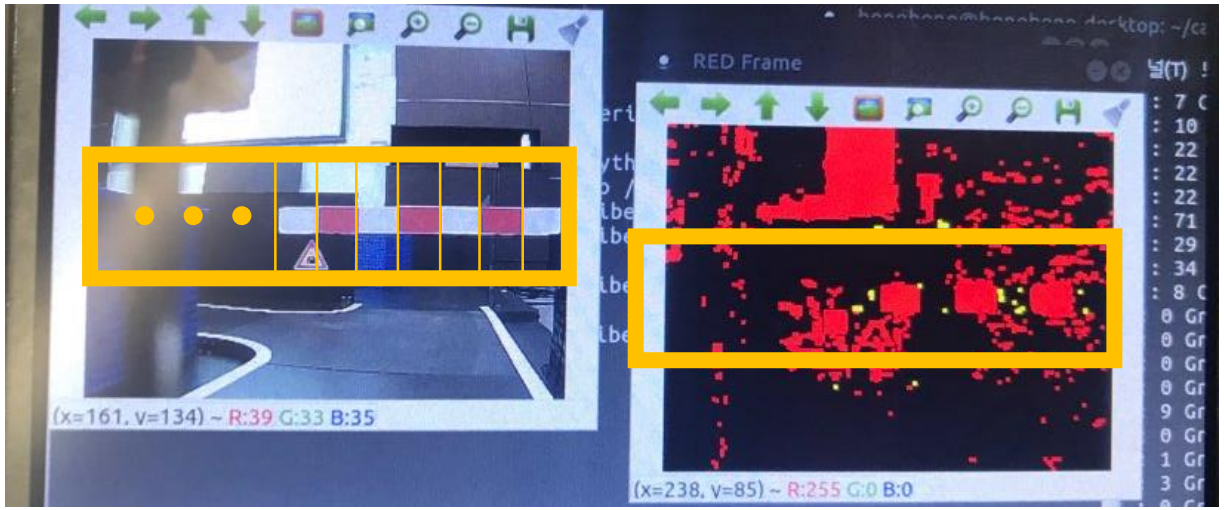


4. 차단바 미션

(2) 현재 사용 알고리즘

1. 차단바를 감지한다.

- 영상을 빨간색 영상으로 이진화
- 이 때 차단바는 화면 중앙에 가로로 길게 위치해 있으므로 관심영역을 노란색 그림과 같이 잡는다
- 차단바는 '흰-빨' 이 연속하여 있으므로 그 특성을 이용하여 정해진 관심영역을 16등분한다.



(A) 실제 주행 사진.

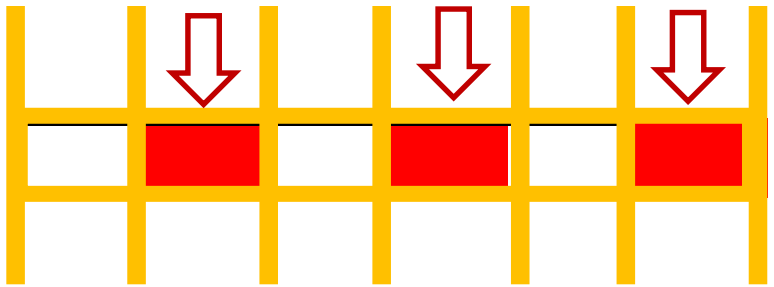
4. 차단바 미션

(2) 현재 사용 알고리즘

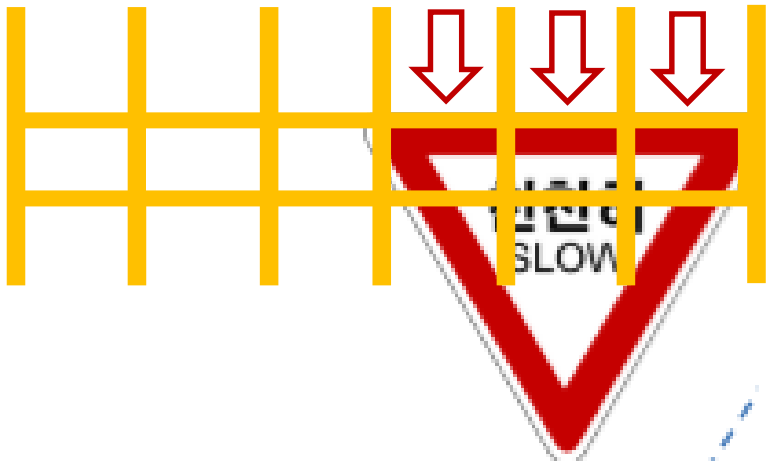
2. 차단바가 있다면 멈춰있는다.
- 16개로 나뉘어진 영역 중 7, 9, 11 번 째 혹은 8, 10, 12 번 째 와 같이 2개 걸러 1개 씩 검출이 3번 이상 이루어진 경우, 차단바라고 생각하고 속도를 0으로 한다.

3. 차단바가 사라지면 RED_COUNT가 급격히 감소하므로 다시 속도를 높이며 라인트레이싱 알고리즘을 실행한다.

(B) 차단바 – PASS

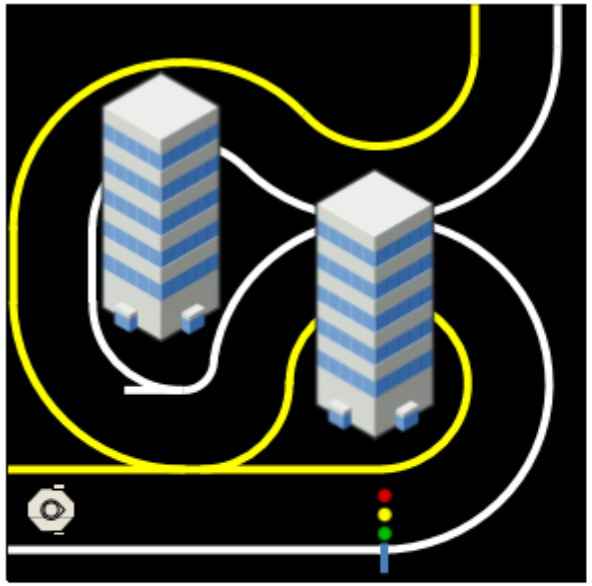


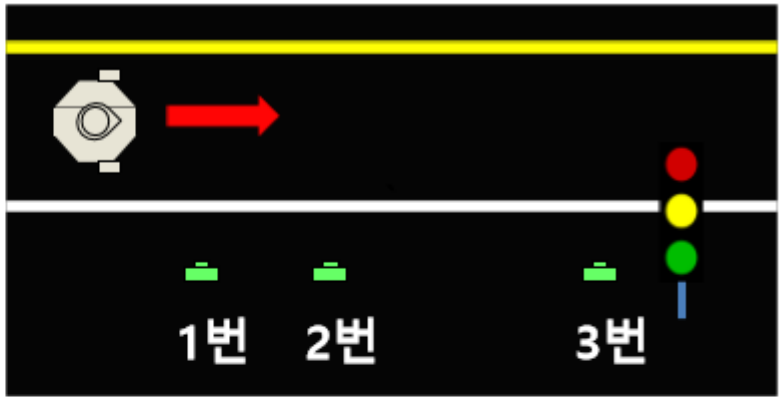
(C) 차단바 - Not PASS



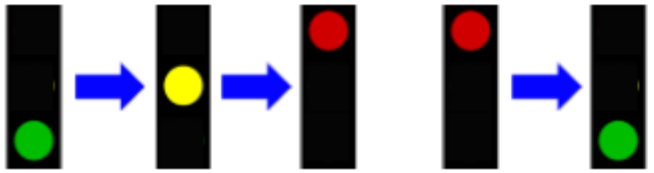
5. 신호등 미션

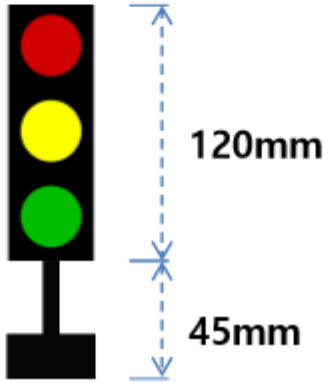
(1) 신호등 예시





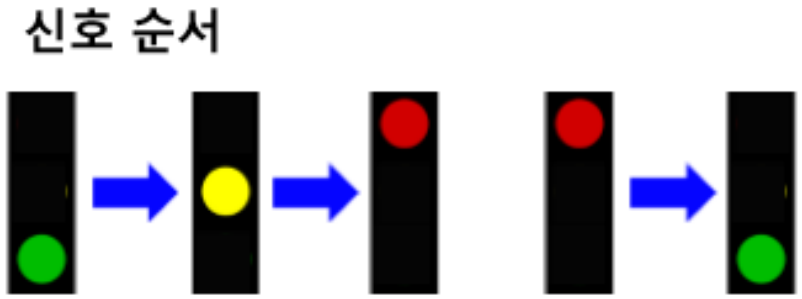
신호 순서





5. 신호등 미션

(2) 현재 사용 알고리즘



1. 신호등이 있는지 살핀다.
- 이 때 관심영역을 오른쪽으로 두어 영상을 받는 속도를 빨리 한다.

(왼쪽과 오른쪽의 위치는 정해지지 않았으므로 당일 날 관심 영역은 바꿀 수 있도록 한다.)

2. 신호등 알고리즘을 실행한다.
- 이 알고리즘은 초록불이 감지된 후 시작한다.
- 초록불이 감지되었다면 (bool green_flag = 1)로 하여 신호등 미션 중 초록불 상태를 감지했음을 알린다.
- (Green_flag == 1) 인 경우, 노란불 상태를 감지했으면 yellow_flag = 1 로 바꾼다.
- (Green_flag == 1 && yellow_flag == 1) 인 경우, 초록불과 노란불이 없어졌음을 감지하고 빨간불 또한 감지한다. 역시 red_flag = 1 로 바꾼다.
- (Green_flag == 1 && yellow_flag == 1 &&red_flag) 인 경우, 빨간불이 사라지고 다시 초록불이 보이면 원래의 속도로 돌아간다.

3. 다시 라인 트레이싱 알고리즘을 이용한다.

감사합니다.