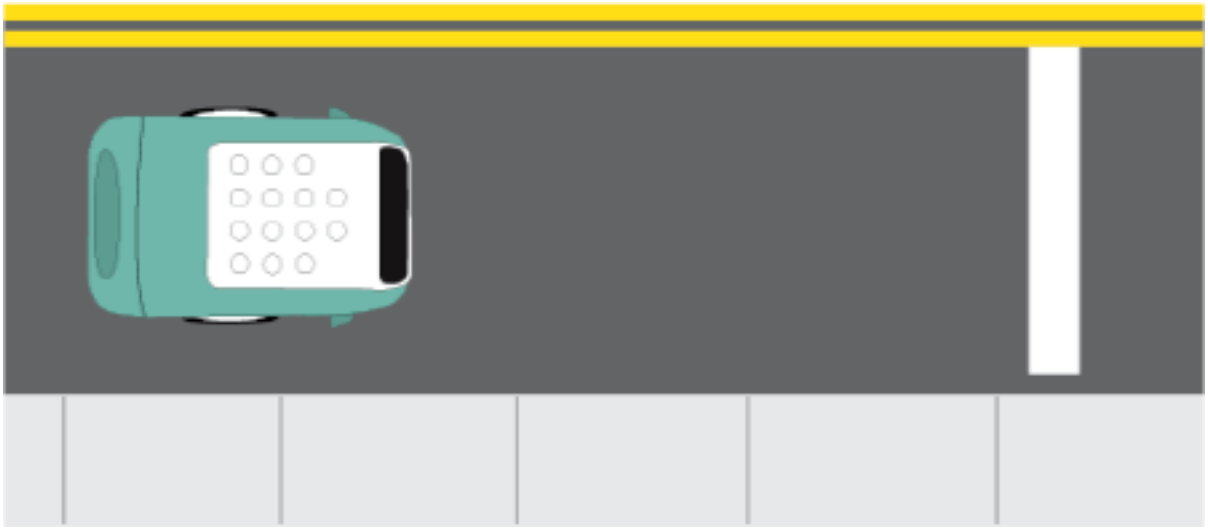
횡단보도 및 라인 따라가기(하단 IR)

자동차는 보행자 및 자전거 이용자와 도로를 공유해야 합니다.
이것이 교차로에서 자동차가 멈출 수 있는 안전 거리가 있는 이유입니다.
이 레슨에서는 하단 IR 센서를 사용하여 횡단보도에서 정지하고 선을 따라 이동합니다.



라이브러리 가져오기

```
from zumi.zumi import Zumi import time

zumi = Zumi()
```

바닥 센서 값을 확인하기

우선 횡단보도와 도로(흰색과 검은색)의 바닥에 있을 때의 하단 왼쪽과 오른쪽 센서 값을 확인해 봅시다.

In [ ]:

```
for i in range(0,30):
    ir_readings = zumi.get_all_IR_data()
    bottom_right_ir = ir_readings[1]
    bottom_left_ir = ir_readings[3]

    message = "    IR readings        "
    message = message + str(bottom_right_ir) + ", " + str(bottom_left_ir)
    print(message)
    time.sleep(0.1)

print("완료!")
```

- 흰색 바닥에 주미가 놓여 있을 때의 주미의 하단 왼쪽 오른쪽 센서 값은 얼마인가요?
- 검은색 바닥에 주미가 놓여 있을 때의 주미의 하단 왼쪽 오른쪽 센서 값은 얼마인가요?

우리는 임계값이라고 하는 센서의 기준 값을 정해야만 프로그램에서 판단을 내릴 수 있습니다.
이 임계값은 측정한 센서 값에 따라서 정할 수 있습니다.

예를 들어, 검은색에서 센서 값이 **200**이고 흰색에서 센서 값이 **50** 이라면, 우리는 얼마의 값을 임계값으로 하면 좋을까요?
이때 우리는 **200과 50의 사이 값**을 사용하면 됩니다.

기본적인 주미의 예제 및 함수에는 기본으로 50 혹은 100정도의 값을 임계값으로 사용하고 있습니다.
하지만 어떤 주미는 센서의 임계 값이 다른 경우도 존재합니다.

만약 여러분의 주미가 검은색에서 센서 값이 **180**이고 흰색에서 센서 값이 **120** 이라면, 우리는 얼마의 값을 임계 값으로 하면 좋을까요?
예제에서 사용되는 임계 값인 100으로 입력할 때 작동할까요? (이런 경우 **180과 120의 사이 값**을 사용하면 됩니다.)

이처럼 우리가 센서를 사용하기 위해서는 센서의 동작 범위 값이 얼마인지 파악해야만 알맞은 동작을 할 수 있도록 사용할 수 있습니다.
또한 바닥의 재질이나 환경에 따라서 값이 달라지므로, 그때마다 현재 센서 상황을 확인해야합니다.

횡단보도에서 정지

이전 강의에서는 장애물을 피하기 위해 주미의 전면 IR 센서를 사용했습니다.
이제 하단 IR 센서를 사용하여 도로의 흰색 또는 검은색 선을 감지하고 정지합니다.

정지할 선을 감지하려면 주미가 운전하는 도로의 색이 반대여야 합니다.

예를 들어, 흰색 선에서 멈추려면 도로가 검은색이어야 합니다. 그렇지 않으면 주미가 멈추지 않습니다.

의사 코드

장애물 회피 수업에서 했던 것처럼 의사 코드의 각 줄을 **주석**으로 작성해봅시다. 우리는 이것을 코드로 변환할 것입니다.

In [ ]:

```
# 여기에 의사 코드를 작성해보세요.
```

해결 방안

[해결 방안을 보려면 클릭하세요!]

```
# 하단 왼쪽과 오른쪽의 센서 값을 읽습니다.
# 하단 왼쪽과 오른쪽의 센서 값이 50보다 작다면 주미가 정지합니다.
# 전방 왼쪽과 오른쪽의 센서 값이 50보다 작지 않다면 주미는 전진합니다.
# 300번 반복합니다.
```

테스트 해보기!

의사 코드를 만들었다면 각 코드 줄을 파이썬으로 바꾸고 아래 for 반복문을 채우세요.
오른쪽 하단 IR의 **인덱스** 값은 1이고, 왼쪽 하단 IR은 3입니다.

작성한 코드는 부드러운 바닥에 전기 테이프를 붙이거나 종이에 검은색 라인을 그린 후 인쇄하여 코드를 테스트합니다.

In [ ]:

```
zumi.reset_gyro() # 현재위치로 방향을 초기화 합니다.
try:
    for x in range(300):
        #여기에 코드를 쓰세요.

finally:
    zumi.stop() # 프로그램이 모두 종료되면 주미는 반드시 멈춰야 합니다.
    print("완료!")
```

해결 방안

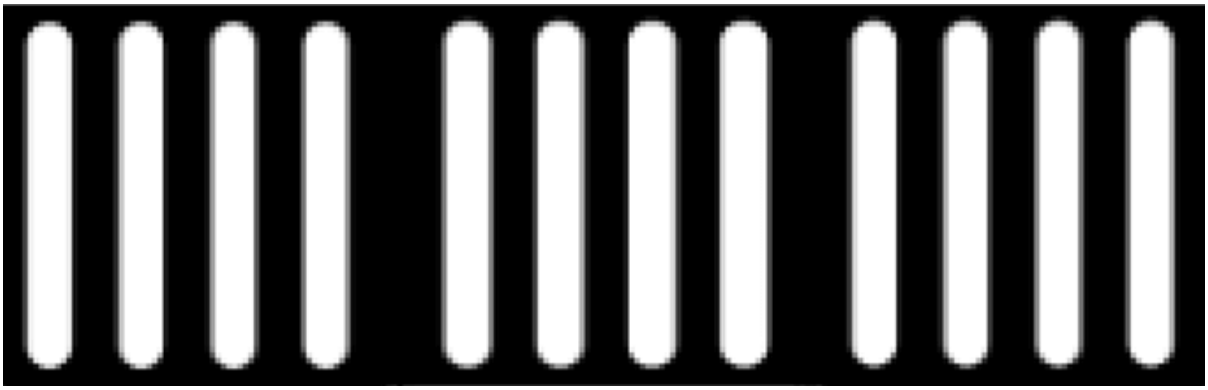
[해결 방안을 보려면 클릭하세요!]

```
zumi.reset_gyro()
try:
    for x in range(300):
        ir_readings = zumi.get_all_IR_data()
        bottom_right_ir = ir_readings[1]
        bottom_left_ir = ir_readings[3]
        if bottom_right_ir < 50 and bottom_left_ir < 50:
            print("흰색 감지 됨")
            zumi.stop()
        else:
            zumi.forward_step(40, 0)
```

```
finally:
    zumi.stop()
print("완료!")
```

라인 갯수 만큼 지나가기

우리는 이러한 도로를 지나칠 때, 원하는 만큼의 라인만을 지나치고 싶을 것입니다.
앞서 센서를 사용하는 방법을 배웠으므로, 바닥 센서를 읽으며 직진을 하도록 반복문을 사용하면서 변수를 기록하면 원하는 만큼이동할 수 있습니다.



다음 함수를 사용해봅시다. 쉽게 이동할수 있나요?

In [ ]:

```
zumi.drive_over_markers(road_markers=3,speed=30,time_out=6)
```

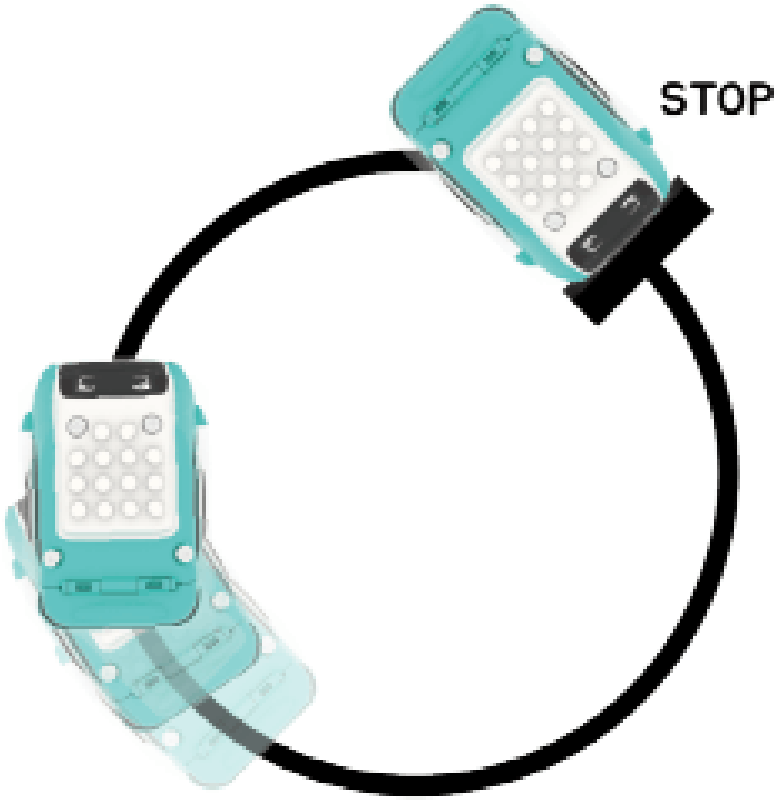
drive_over_markers(road_markers=10,speed=30, ir_threshold = 100, time_out=6)

- road_markers: 흰색 라인의 갯수
- speed: 속도 0 ~ 80
- ir_threshold: 센서가 흰색을 감지하는 임계 값
- time_out: 함수가 작동하는 시간(초)

drive_over_marker() 함수는 입력한 갯수만큼의 라인을 지나치고 멈추게 됩니다.
markers 는 원하는 흰색 라인의 갯수입니다. 이 갯수를 지나치면 주미는 멈추게 됩니다.
speed는 이동할 때의 속도입니다.
IR_threshold는 센서가 흰색을 감지하는 임계 기준 값입니다. 주미의 센서의 상태나 바닥의 상태에 따라 값을 변경합시다.
time_out 은 입력한 시간이 지나면 주미를 자동으로 멈추게 합니다. 갯수를 다 읽지 못했더라도 이 시간이 지나면 주미는 멈추게 됩니다.

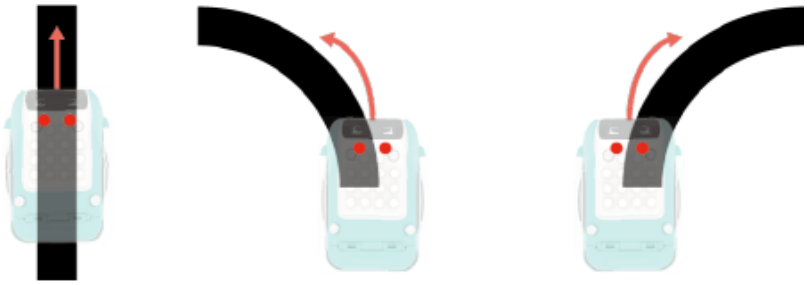
라인 따라가기

이제 주미는 선을 볼 때 멈추지 않고 선을 따라갈 것입니다.
도로에서 차선을 유지하듯이, 함수를 호출하여 주미가 검은 선에 머물도록 합니다.
먼저 작동 방식을 알아보겠습니다.



주미는 바닥의 두 센서를 모두 사용하여 라인을 확인하며 이동합니다.

바닥에 있는 테이프 위를 걷는 다고 생각해봅시다.
오른발이 라인을 벗어나면 어느 방향으로 회전해야할까요? 왼발은 어떤가요?
IR 센서는 동일한 논리를 사용합니다.



line_follow_gyro_assist() 함수 사용해보기

함수를 테스트해봅시다!

전기 테이프 또는 주미 맵을 사용하고 주미를 검은색 선위에 정렬합니다.

`zumi.line_follower()` 함수에는 적어도 하나의 매개변수(작동시간)는 꼭 필요합니다.

예를 들어 프로그램을 3초 동안 실행해 보겠습니다.

In [ ]:

```
zumi.line_follow_gyro_assist(duration=3)
```

주미가 라인을 잘 감지하지 못한다고 생각이 들거나, 바닥의 어둡고 밝음 차이가 충분하지 않다고 생각되면 센서의 임계값을 조절해야합니다. 기본센서 임계값은 100으로 설정되어 있습니다. IR 센서를 테스트하고 임계값을 파악하려면 센서 값 확인하기 부분을 참조하세요.

line_follow_gyro_assist(speed=20, duration=1.0)
line_follow_gyro_assist(speed=20, duration=1, angle=None, angle_adj=2, l_th=100, r_th=100)

- speed: 속도 (0 ~ 80)
- duration(지속 시간): 함수가 작동하는 시간(초)
- angle: 원하는 각도(기본값은 주미의 현재 각도)
- angle_adj(각도 보정): 하나의 IR 센서가 흰색을 감지하면 Zumi가 회전하는 각도입니다.
- l_th: 왼쪽 하단 IR 센서의 임계값
- r_th: 우측 하단 IR 센서의 임계값

<참고>

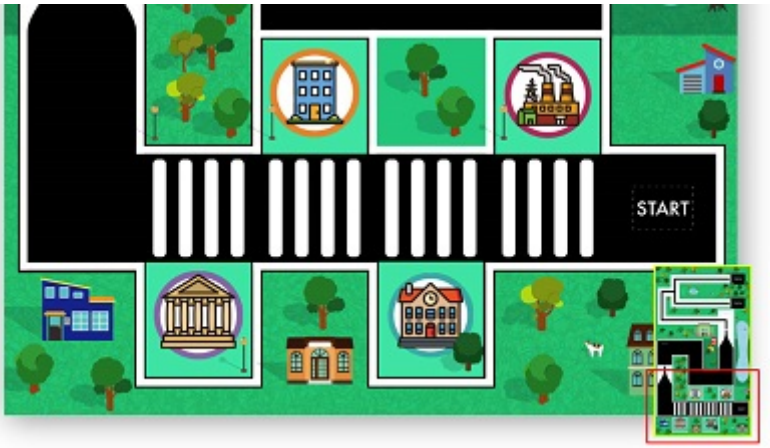
주미가 현재 향하고 있는 방향으로 하단 IR 센서가 검은 선을 감지하면 주미는 계속 주행합니다.
한쪽 센서에서 흰색이 검출되면 주미는 자동으로 조정되어 라인을 유지합니다.
입력된 지속 시간이 지나면 정지합니다.
두 센서 모두 흰색을 감지하면 지속 시간이 충족되지 않았더라도 주미는 정지합니다.

다른 값들도 입력하여 테스트 해봅시다.

In [ ]:

```
zumi.line_follow_gyro_assist(speed=5,duration=7, angle_adj=2.2,l_th =150, r_th=150)
```

대회 : 어떻게 사용되나요?

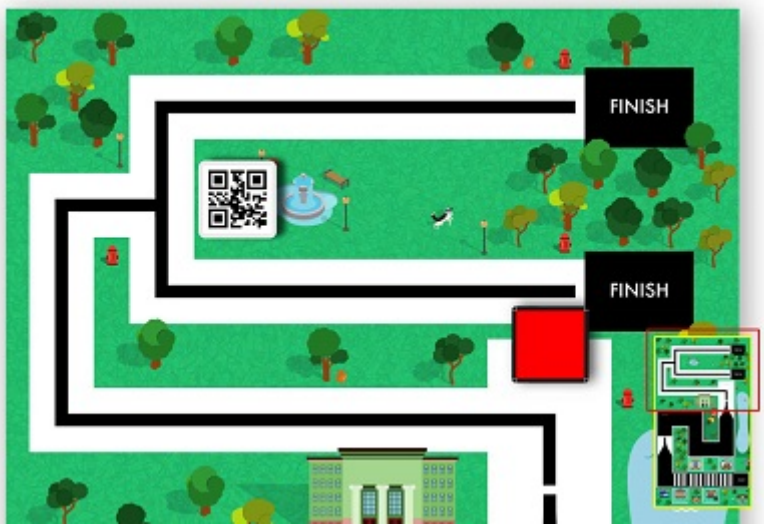


우리는 앞서 drive_over_markers()함수를 사용하는 방법을 배웠습니다. 따라서, 바닥 센서를 읽으며 직진을 하도록 매개변수 값을 조정해주면 됩니다.
맵에서는 4개의 목적지가 있습니다.
QR 코드에서 읽은 메시지에 따라서 해당 목적지로 이동합니다.
각각의 목적지까지 이동하기 위한 라인의 갯수는 다르므로, 이에 따라서 프로그램을 작성해야합니다.

다음 예를 한번 볼까요? 이것은 목적지에 따라 이동하는 코드 예시의 일부분입니다.

In [ ]:

```
qr_message = 'factory' # 예를 들어 QR 코드에서 읽은 메시지 factory 였다고 가정합니다.

if qr_message == 'factory':
    print("go factory")
    zumi.drive_over_markers(3,speed=30,time_out=6)

# 다른 목적지의 조건문도 추가해봅시다.
```



대회 맵에서 마지막 구역의 경로가 라인으로만 구성되어 있음을 눈치채셨을 것입니다.
이 섹션에서는 line_follow_gyro_assist() 함수를 사용하여 주미가 경로를 제대로 따르고 있는지 확인할 수 있습니다.

라인이 직각으로 꺾이는 부분에서 line_follow_gyro_assist() 함수는 멈추게 됩니다.
그럴 때는 turn_left() 와 turn_right() 를 사용하여 진행합니다.
주미가 맵에서 라인을 통과하는 방법을 보려면 아래 코드를 확인하세요!

In [5]:

```
zumi.line_follow_gyro_assist(speed=20,duration=2, angle_adj=0.6)
zumi.turn_left()
zumi.line_follow_gyro_assist(speed=20,duration=7, angle_adj=0.6)
zumi.turn_right()
```

line_follow_gyro_assist() 함수를 사용할 때는 속도가 너무 빠르면 라인을 이탈할 확률이 높습니다.
지속 시간이 너무 짧으면 라인을 이동하기전에 주미가 멈춰버릴 수 있습니다.
각도 보정 값이 너무 크거나 작으면 라인을 이탈할 수 있습니다.

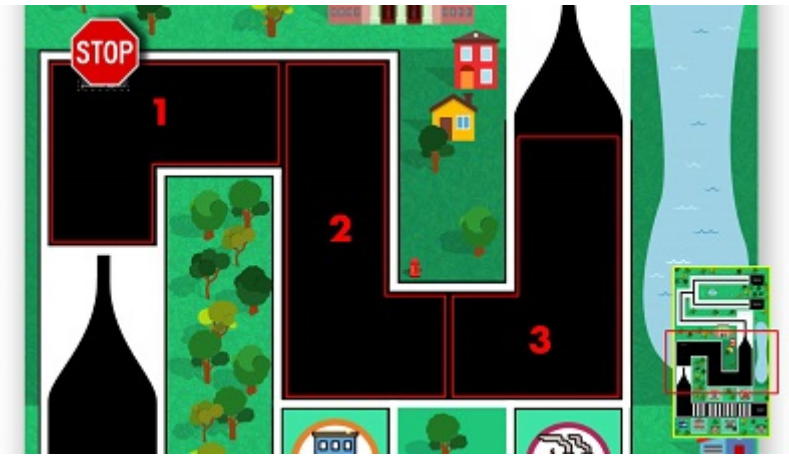
테스트를 진행할 때는 지속 시간은 넉넉히 주고, 속도는 느리게, 각도 보정 값은 작은 값으로 테스트를 시작하며 점점 늘려가세요.
주미에 따라서 센서의 임계 값도 추가로 입력해 주어야 할수도 있습니다.

마지막 구간에서는 QR코드에 따라서 이동하는 방향이 달라집니다. 다음 코드를 확인해 보세요.

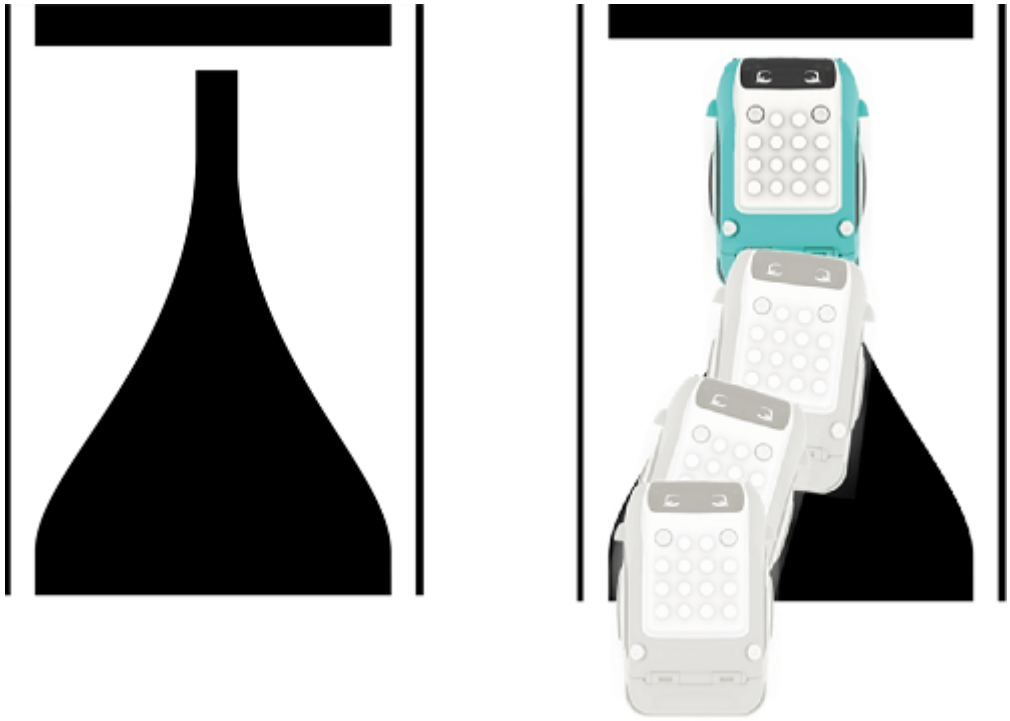
In [ ]:

```
# 이런식으로 작동합니다.

qr_message = 'left' # 예를 들어 QR 코드에서 읽은 메시지 left 였다고 가정합니다.

if decision == "left":
    print("go left")
    zumi.turn_left()
    zumi.line_follow_gyro_assist(speed=10,duration=5, angle_adj=angle_adjustment)
```

깔때기 모양의 도로 (funnel_align)



대회에서 일부 구간에는 깔때기 모양의 경로가 있는 것을 보았을 것입니다.
이 구간의 경우 Zumi의 IR 센서를 사용하여 Zumi를 안내할 수 있습니다.
그 방법이 궁금하다면 IR 센서를 반복문 및 조건문과 결합하여 깔때기를 따라가도록 하세요!



주미가 출발할때는 z축의 각도가 정확하였지만, 긴 코스에서 주행을 오래하다보면 오차가 조금씩 생기게 됩니다.
이 부분을 돕기위해 주미의 맵에는 다음과 같은 장소가 존재합니다.
깔때기 모양의 도로로 이 부분을 지나면 주미의 방향이 깔때기의 끝을 향하도록 모이게 됩니다.
완전하지는 않지만, 최대한 보정을 해주는 역할을 합니다.

해당 장소에서 "funnel_align()" 함수를 사용하여 코스를 통과할 수 있습니다.

`funnel_align(speed=20, duration=1, angle=None, angle_adj=2, l_th=100, r_th=100)`

- `speed` : 0 ~80 속도의 양의 정수값
- `duration` : 지속시간, 함수가 작동하는 시간(초)
- `angle` : 원하는 각도(기본값은 주미의 현재 각도)
- `angle_adj` (각도 보정) :적외선 센서로 흰색을 감지할 때 주미가 회전할 각도
- `l_th`: 왼쪽 하단 IR 센서의 임계값
- `r_th`: 우측 하단 IR 센서의 임계값

속도"speed"는 맵안에서의 이동속도이며 빠를 필요는 없습니다.
지속시간 "duration"은 주 미가 깔때기 끝까지 가서 완료 동작을 하지 않아도 해당 시간이 지나면 자동으로 함수가 종료되게 됩니다.
주미가 바닥의 양쪽 센서에 감지 될때마다 중앙으로 회전하게 되는데, 이 회전하는 각도가 너무 크거나 작다면 각도 조정 "angle_adj" 값을 수정합니다.
l_th, r_th 값은 각각의 주미의 센서 상황에 따라서 입력하도록 합니다. (주미에 따라 값이 다를수 있습니다.)

아래 코드 예제에서 Zumi는 흰색 선에 도달할 때까지 계속 앞으로 운전한 다음 멈출 것입니다.

In [ ]:

```
#funnel_align(speed=20, duration=1, angle=None, angle_adj=2, l_th=100, r_th=100)

zumi.funnel_align(speed=5,duration=7,angle_adj=1)
```