- 2.2 Gyroscope
자이로스코프

자이로 스코프 센서는 방향 변화를 감지하기 위해 사용하는 작은 칩입니다. 로봇의 경우 회전 속도를 측정하는 장치입니다.

자이로 스코프는 Zumi의 회전 각도를 계산하기 위해 회전 속도가 필요하기 때문에 중요합니다.

zumi.update_angles()[0]

GyroX

zumi.update_angles()[1]

GyroY

zumi.update_angles()[2]

GyroZ

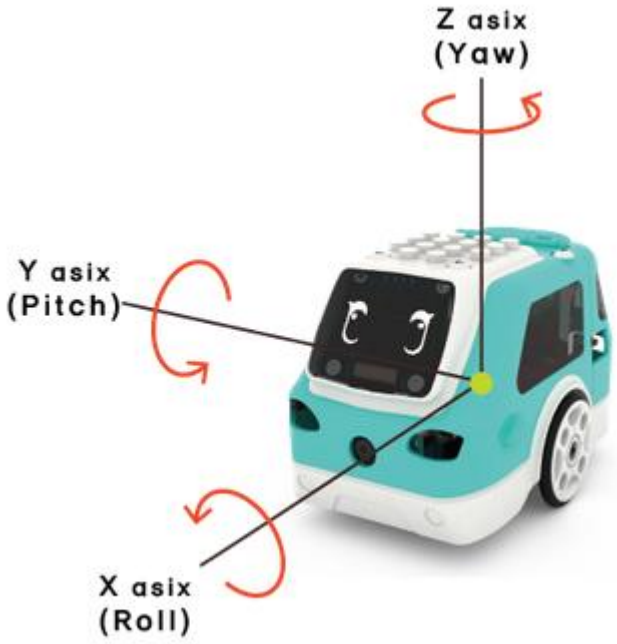
- 2.2 Gyroscope
자이로스코프

X,Y,Z 축에 해당하는 방향을 이해하셨나요?

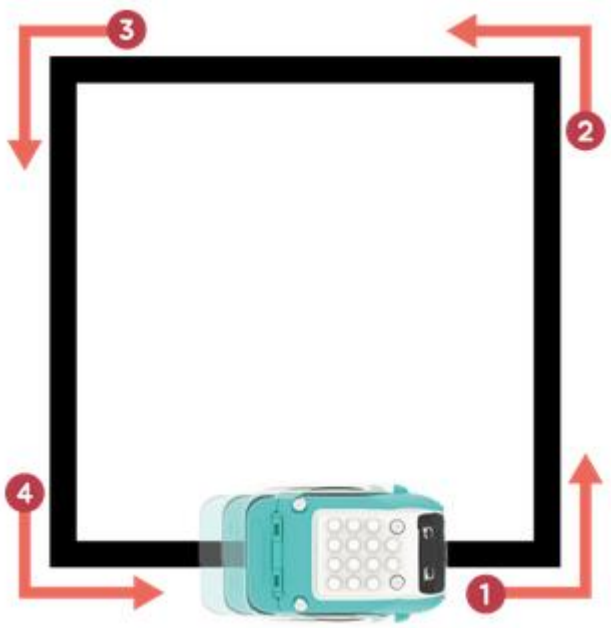
첫 번째 X 축 : 이것은 다른 이름으로 롤(roll)이라고 불립니다.
롤은 오른쪽 또는 왼쪽 기울기를 담당합니다.

두 번째 Y축 :이것은 다른 이름으로 피치(pitch)라고 불립니다.
피치는 앞 또는 뒤 기울기를 담당합니다.

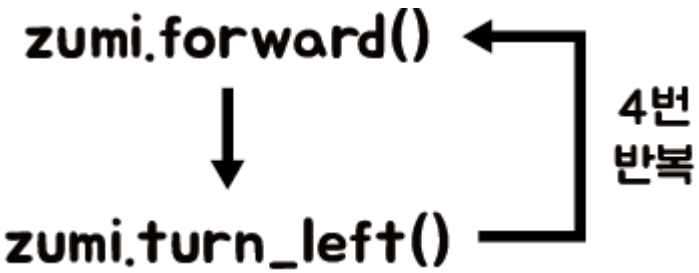
마지막 Z 축: 이것은 다른 이름으로 요(yaw)라고 불립니다.
요는 좌우 선회를 담당합니다.



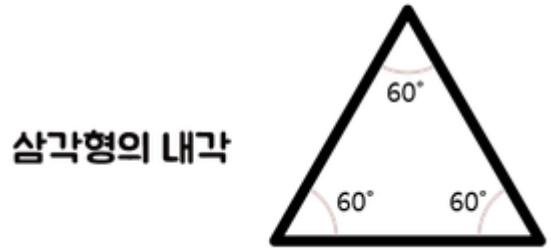
- 2.3 shapes with loops
반복문 사용하기

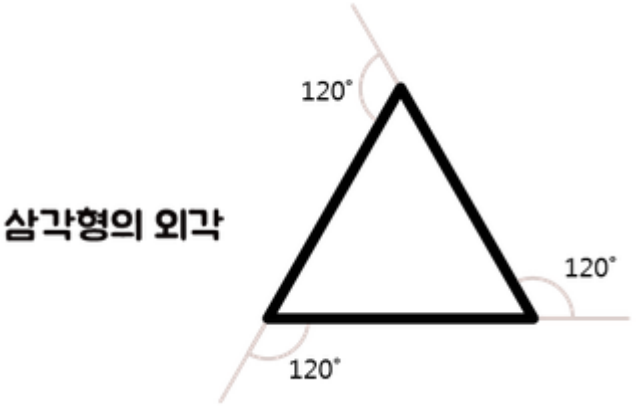


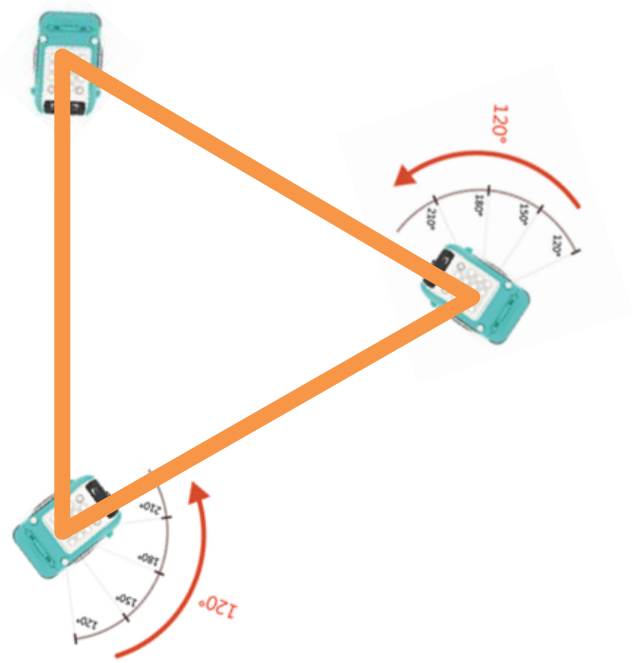
정사각형 모양으로 주행을 하려면 오른쪽 또는 왼쪽으로 4번을 회전해야 합니다.



- 2.3 shapes with loops
반복문 사용하기








```
# 삼각형 그리기

initial_angle = zumi.read_z_angle()

for angle in range(0, 271, 120):
        zumi.forward(40,1,initial_angle+angle)
```

- 2.3 shapes with loops
반복문 사용하기


```
# 오각형 그리기
initial_angle = zumi.read_z_angle()

for angle in range(0, 289, 72):
    zumi.forward(40,1,initial_angle+angle)
```

```
# 다각형 그리기

polygon = 5

angle = int(360/polygon)

initial_angle = zumi.read_z_angle()
for angle in range(0, (angle*polygon-1), angle):
    zumi.forward(40,1,initial_angle+angle)
```

- 1.3 light
라이트

- zumi.all_lights_on() : 모든 라이트 켜기
- zumi.all_lights_off() : 모든 라이트 끄기

- zumi.headlights_on() : 전면 라이트 켜기
- zumi.headlights_off() : 전면 라이트 끄기

- zumi.brake_lights_on() : 후면 라이트 켜기
- zumi.brake_lights_off() : 후면 라이트 끄기

- 1.3 light
라이트

- zumi.hazard_lights_on() : 모든 라이트 깜빡이기
- zumi.hazard_lights_off() : 모든 라이트 깜빡이 끄기

- zumi.signal_left_on() : 왼쪽 라이트 깜빡이기
- zumi.signal_left_off() : 왼쪽 라이트 깜빡이 끄기

- zumi.signal_right_on() : 오른쪽 라이트 깜빡이기
- zumi.signal_right_off() : 오른쪽 라이트 깜빡이 끄기

- 1.3 light
라이트

#주미의모든LED를3초동안켜고끄기

```
zumi.all_lights_on()
time.sleep(3)
zumi.all_lights_off()
```

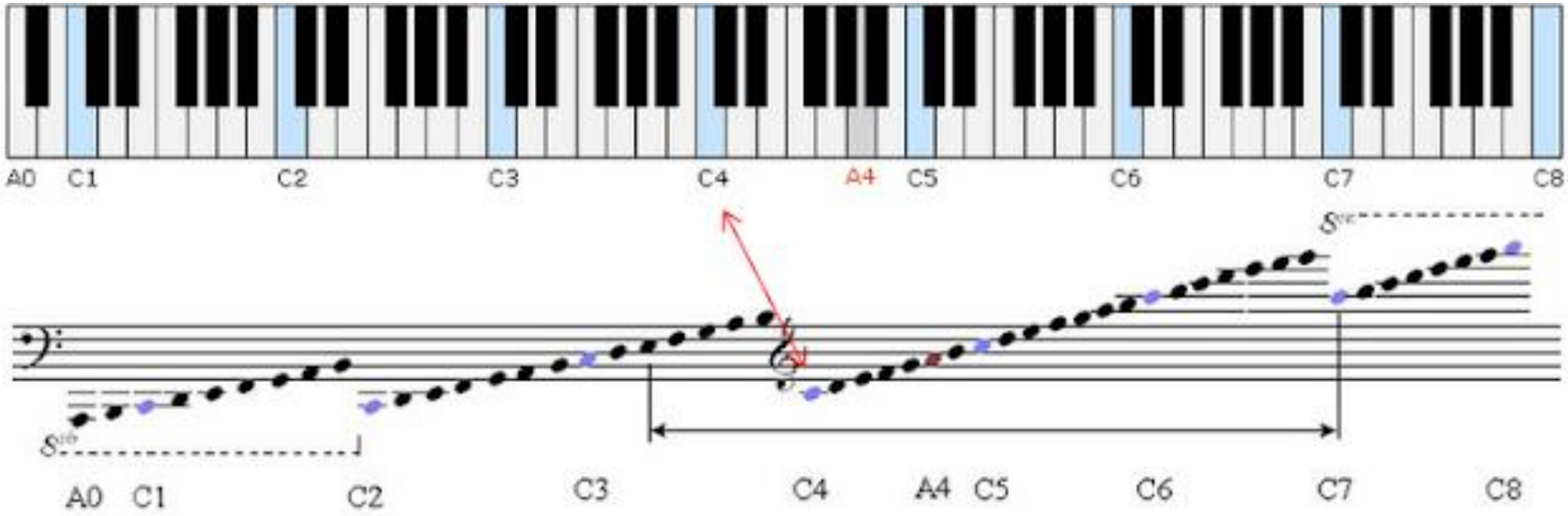
#주미의 모든 LED를 3번 끄고 켜기 반복

```
for angle in range(0, 3):
    zumi.all_lights_on()
    time.sleep(1)
    zumi.all_lights_off()
    time.sleep(1)
```

#signal_right_on() / signal_right_off()
명령을 이용하여 주미가 우회전을 할 때 신호를 나타내도록 해보세요.

#전진하고 멈추면서 브레이크등으로 신호를 나타내도록 해보세요.
brake_lights_on() / brake_lights_off() /
time.sleep() 명령을이용하세요.

- 5.2 Robot Emotions
감정 표현 프로그래밍



음계
class Note:

C2 = 1 CS2 = 2 D2 = 3 DS2 = 4 E2 = 5 F2 = 6 FS2 = 7 G2 = 8 GS2 = 9 A2 = 10 AS2 = 11 B2 = 12

C3 = 13 CS3 = 14 D3 = 15 DS3 = 16 E3 = 17 F3 = 18 FS3 = 19 G3 = 20 GS3 = 21 A3 = 22 AS3 = 23 B3 = 24

C4 = 25 CS4 = 26 D4 = 27 DS4 = 28 E4 = 29 F4 = 30 FS4 = 31 G4 = 32 GS4 = 33 A4 = 34 AS4 = 35 B4 = 36

C5 = 37 CS5 = 38 D5 = 39 DS5 = 40 E5 = 41 F5 = 42 FS5 = 43 G5 = 44 GS5 = 45 A5 = 46 AS5 = 47 B5 = 48

C6 = 49 CS6 = 50 D6 = 51 DS6 = 52 E6 = 53 F6 = 54 FS6 = 55 G6 = 56 GS6 = 57 A6 = 58 AS6 = 59 B6 = 60

- 5.2 Robot Emotions
감정 표현 프로그래밍

#불쌍해보이는 모습

```
for i in range(3):
    screen.sad()
    time.sleep(0.5)
    screen.close_eyes()
    time.sleep(0.5)
    zumi.reverse(5, 0.1)
    zumi.reverse(5, 0.1)
```

- 5.2 Robot Emotions
감정 표현 프로그래밍

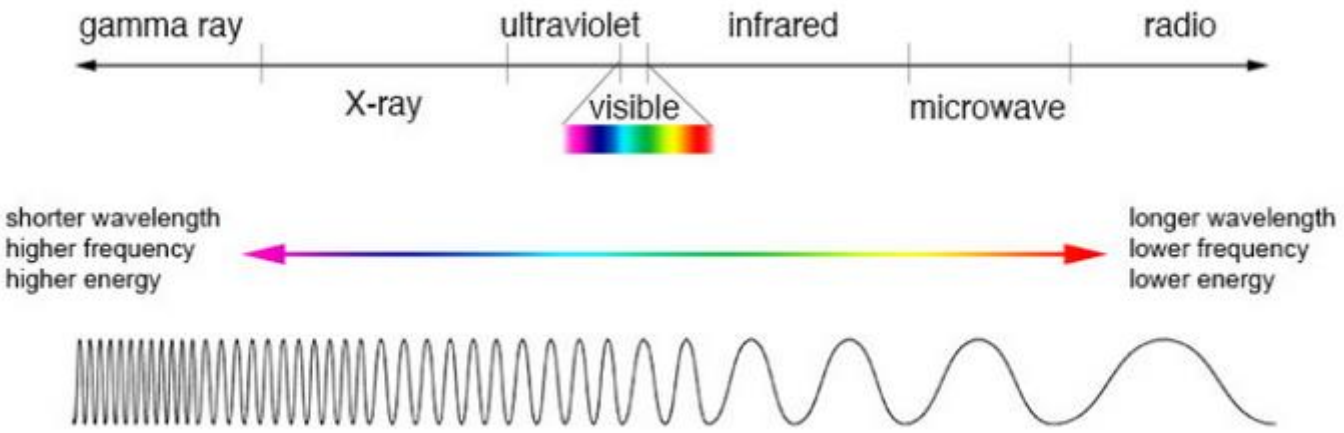
#놀란 모습

```
for i in range(3):
    screen.blink()
    zumi.play_note(49, 20)
    zumi.play_note(50, 20)
    zumi.play_note(51, 20)
    zumi.forward(10, 0.1)
```

- 5.2 Robot Emotions
감정 표현 프로그래밍

```
# 다양한 표현

tempo = 80
zumi.play_note(Note.C4, tempo)
zumi.play_note(Note.E4, tempo)
zumi.play_note(Note.G4, tempo)
zumi.play_note(Note.C5, tempo)
screen.sleeping()

tempo = 100
zumi.play_note(Note.B5, tempo)
zumi.play_note(Note.AS5, tempo)
zumi.play_note(Note.A5, tempo)
zumi.play_note(Note.GS5, tempo)
zumi.reverse(10, 0.1)
screen.glimmer()
time.sleep(0.5)

zumi.play_note(Note.C6,tempo)
screen.sleepy_eyes()
```

- 3.1 IR Sensor
- 적외선 센서



적외선(줄여서 IR)은 인간의 눈으로는 볼 수 없는 전자기 방사선의 한 형태입니다. 이는 또한 에너지의 한 형태입니다.

우리가 볼 수 있는 작은 부분도 있는데 이를 가시광선이라고 합니다.
우리가 보는 여러 색상은 다양한 파장에 의해 정해집니다.

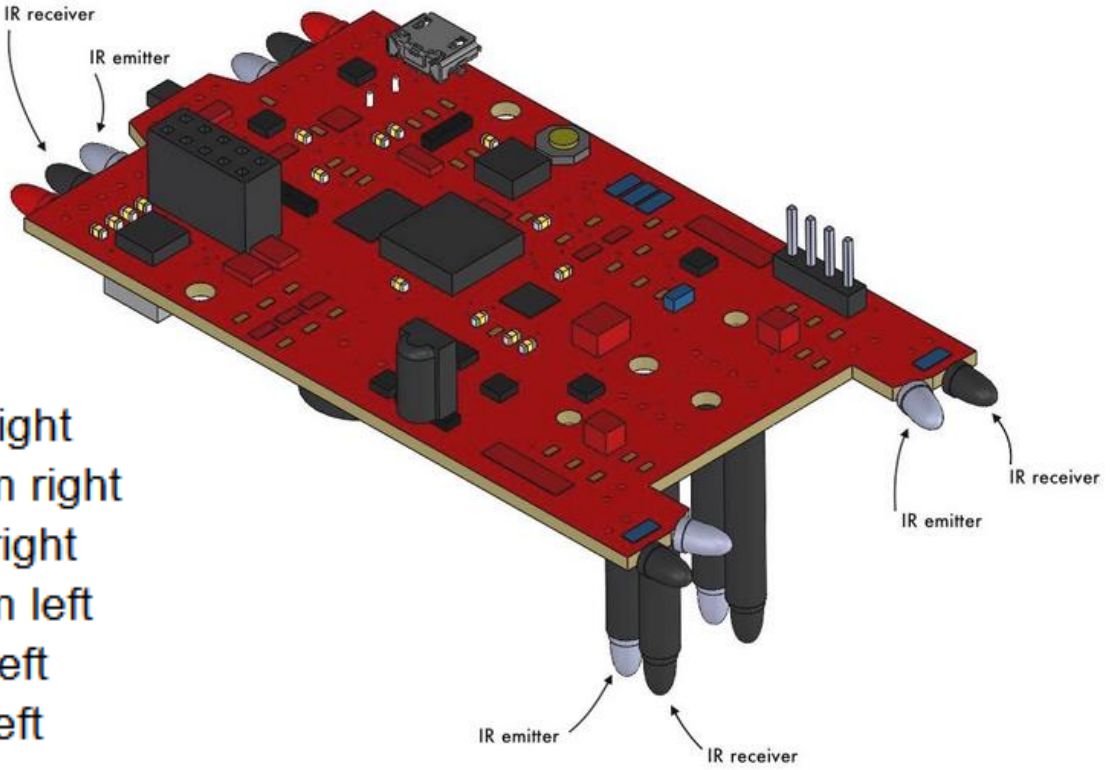
파장이 길어질수록, 사람들에게 보이지 않습니다.
그러나, 특수 센서들은 이러한 파장을 감지할 수 있습니다.

Zumi의 적외선센서(IR Sensor)를 사용하여 센서에 닿는 물체들을 감지할 수 있습니다.

- 3.1 IR Sensor
- 적외선 센서

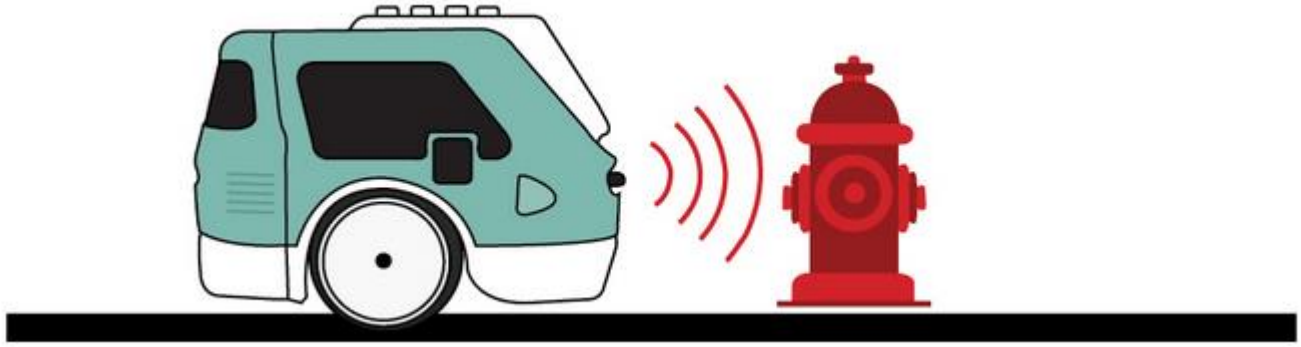
Zumi에는 6 개의 IR 센서가 장착되어 있습니다

(전면 2 개, 후면 2 개, 하단 2 개). 그들은 모두 번호를 가지고 있어 데이터를 읽을 수 있습니다.



- IR 0 = front right
- IR 1 = bottom right
- IR 2 = back right
- IR 3 = bottom left
- IR 4 = back left
- IR 5 = front left

- 3.2 obstacle Avoidance
- 장애물 회피



```
# 전방 센서 사용하기

for i in range(20):

    ir_readings = zumi.get_all_IR_data()
    front_right_ir = ir_readings[0]
    front_left_ir = ir_readings[5]

    if front_left_ir < 100:
        print("something on left")

    elif front_right_ir < 100:
        print("something on right")
```

- 3.2 obstacle Avoidance
- 장애물 회피

```
# 장애물 회피하기
for i in range(50):
    ir_readings = zumi.get_all_IR_data()
    front_right_ir = ir_readings[0]
    front_left_ir = ir_readings[5]

    if front_left_ir < 100 and front_right_ir < 100:
        print("something ahead")
        zumi.stop(0)
        time.sleep(0.5)
        zumi.reverse()
        zumi.stop(0)

    elif front_left_ir < 100:
        print("something on left")
        # 여기에 코딩해 보세요.

    elif front_right_ir < 100:
        print("something on right")
        # 여기에 코딩해 보세요.

    # 여기에 코딩해 보세요.
    time.sleep(0.05)

zumi.stop(0)
```

- 3.2 obstacle Avoidance
- 장애물 회피

```
# 따라오는 반려 동물 만들기

for i in range(50):
    ir_readings = zumi.get_all_IR_data()
    front_right_ir = ir_readings[0]
    front_left_ir = ir_readings[5]

    if front_left_ir < 100 and front_right_ir < 100:
        print("something ahead")

    elif front_left_ir < 100:
        print("something on left")
        # 여기에 코딩해 보세요.

    elif front_right_ir < 100:
        print("something on right")
        # 여기에 코딩해 보세요.

    # 여기에 코딩해 보세요.
    time.sleep(0.05)

zumi.stop(0)
```

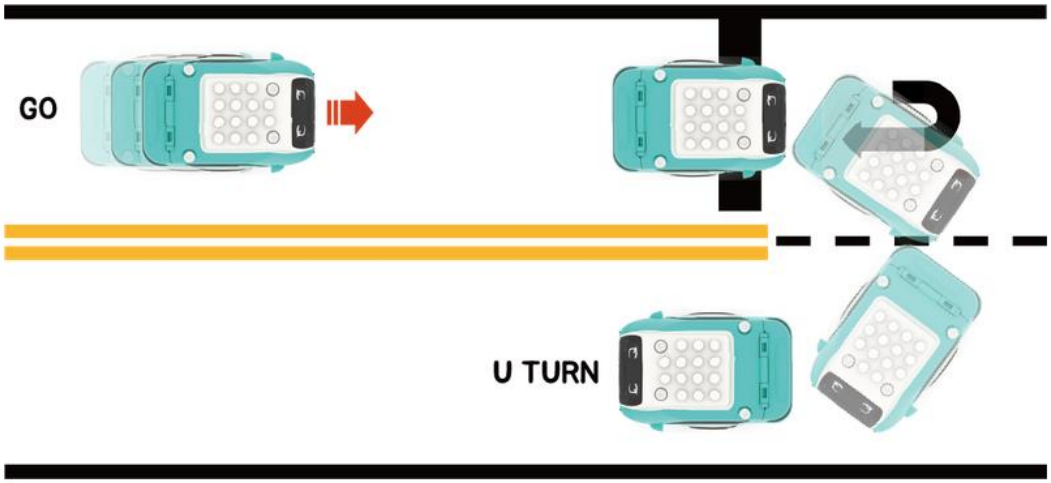
- 3.2 obstacle Avoidance
- 라인트레이서


```
# 도로 주행 중 멈추기
while True:

    ir_readings = zumi.get_all_IR_data()
    bottom_right = ir_readings[1]
    bottom_left = ir_readings[3]

    zumi.go_straight(10, 0)

    if bottom_right > 100  and bottom_left > 100:
        zumi.stop(0)
        print('stop')
        break

time.sleep(3)
```

- 3.2 obstacle Avoidance
- 라인트레이서

실제 자동차처럼 Zumi가 정지선을 감지하면 제동 등을 켜도록 코딩해보세요.

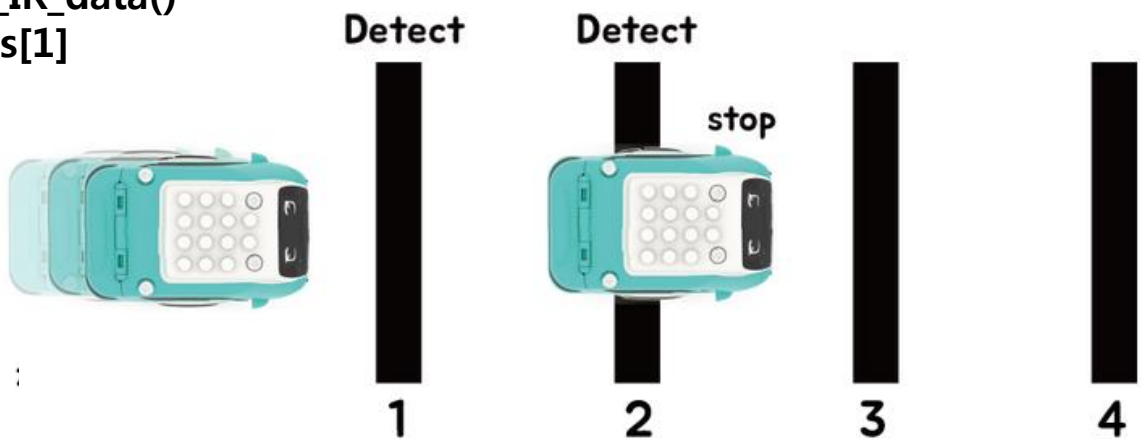
- 3.2 obstacle Avoidance
- 라인트레이서




```
# 도로 주행 중  u턴

zumi.reset_gyro()
while True:
    ir_readings = zumi.get_all_IR_data()
    bottom_right = ir_readings[1]
    bottom_left = ir_readings[3]

    zumi.go_straight(10, 0)
    if bottom_right > 100  and bottom_left > 100:
        zumi.stop(0)
        print('U-Turn')
        zumi.right_u_turn(step=6)
        print('stop')
        break
time.sleep(3)
```

- 3.2 obstacle Avoidance
- 라인트레이서

```
# 입력한 라인 갯수 만큼 이동

zumi.reset_gyro()

count = int(input("input count : "))
blackFlag = False

while count > 0:
    ir_readings = zumi.get_all_IR_data()
    bottom_right = ir_readings[1]

    if blackFlag == False:
        if bottom_right > 100:
            blackFlag = True

    elif blackFlag == True:
        if bottom_right < 50:

            count = count - 1
            blackFlag = False

    zumi.go_straight(20, 0)

zumi.stop(0)
```



- 3.2 obstacle Avoidance
- 라인트레이서

```
try:
    zumi.reset_gyro()
    init_ang_z = zumi.read_z_angle()

    # 선을 벗어나면 threshold 값을 조금씩 늘려보세요.
    threshold = 80   # 기준 센서 값
    turnSpeed = 3    # 회전시 속도
    turnAngle = 3    # 회전 각도

    while True:

        # 센서 읽기
        ir_readings = zumi.get_all_IR_data()
        bottom_right = ir_readings[1]
        bottom_left = ir_readings[3]

        # 왼쪽색 & 오른쪽 센서 검정색 감지 된 경우   -> 직진
        if bottom_left > threshold and bottom_right > threshold :
            zumi.go_straight(turnSpeed, init_ang_z)

        # 왼쪽색 센서 검정색 감지 된 경우   -> 오른쪽으로 방향 회전
        elif bottom_left > threshold :
            init_ang_z+= turnAngle
            zumi.go_straight(turnSpeed, init_ang_z)

        # 오른쪽 센서 검정색 감지 된 경우 -> 왼쪽으로 방향 회전
        elif bottom_right > threshold :
            init_ang_z-= turnAngle
            zumi.go_straight(turnSpeed, init_ang_z)


except KeyboardInterrupt:
    zumi.stop(0)
    print("The interrupt button was pressed.")
```

- 3.2 obstacle Avoidance
- 라인트레이서

#Line Tracer (모터 컨트롤을 사용한 라인트레이서)

```
try:
    threshold = 100
    turnSpeed = 10
    forwardSpeed = 10

    while True:

        ir_readings = zumi.get_all_IR_data()
        bottom_right = ir_readings[1]
        bottom_left = ir_readings[3]

        if bottom_left < threshold and bottom_right < threshold :
            zumi.control_motors(forwardSpeed,forwardSpeed,0)

        elif bottom_left > threshold and bottom_right < threshold :
            zumi.control_motors(turnSpeed,0,0)

        elif bottom_left < threshold and bottom_right > threshold:
            zumi.control_motors(0,turnSpeed,0)

        elif bottom_left > threshold :
            zumi.control_motors(turnSpeed,0,0)

        elif bottom_right > threshold :
            zumi.control_motors(0,turnSpeed,0)

except KeyboardInterrupt:
    zumi.stop(0)
    print("The interrupt button was pressed.")
```