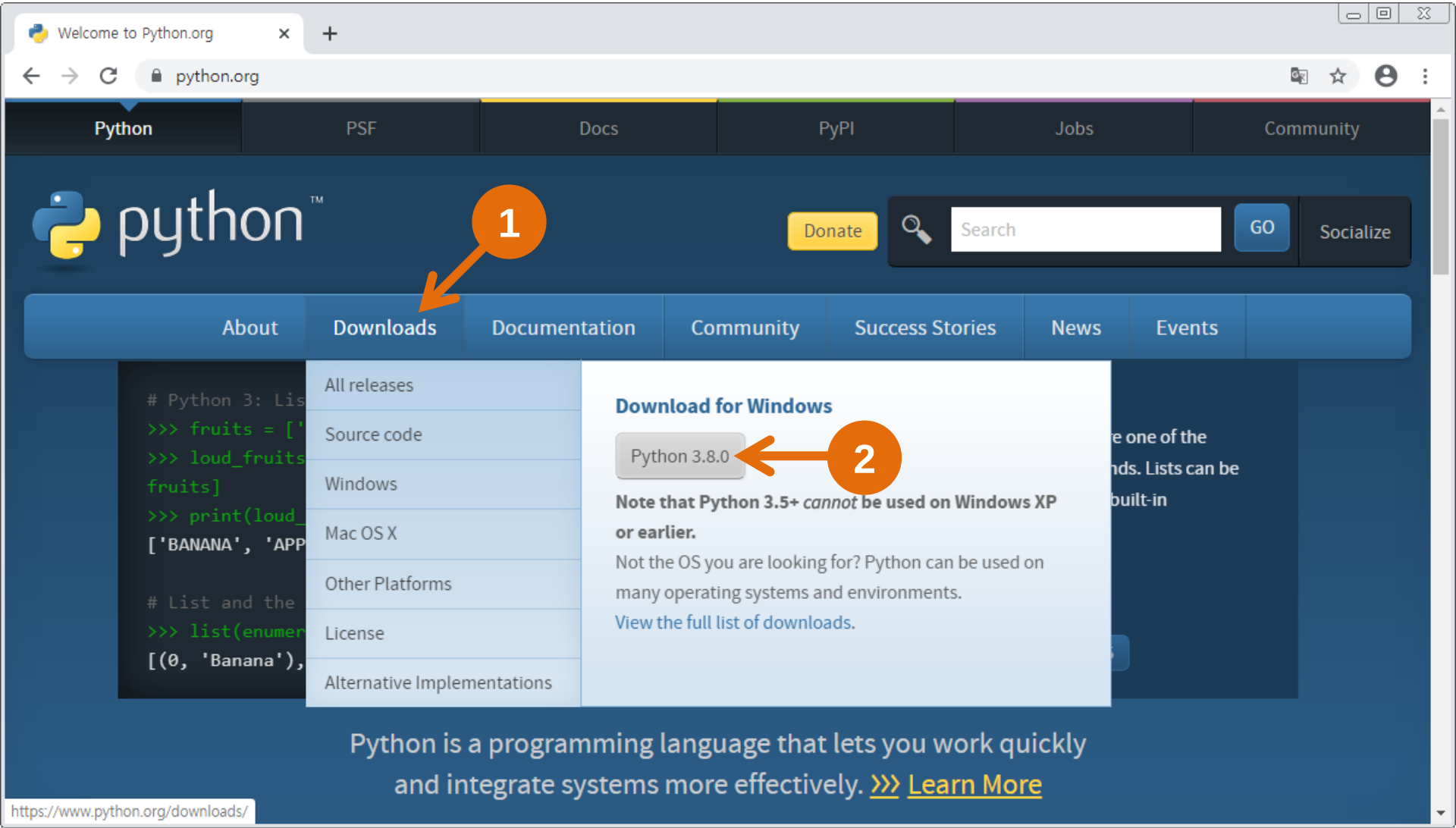
햄스터로 배우는 파이선

광운대학교 로봇학부
박광현

파이선 설치

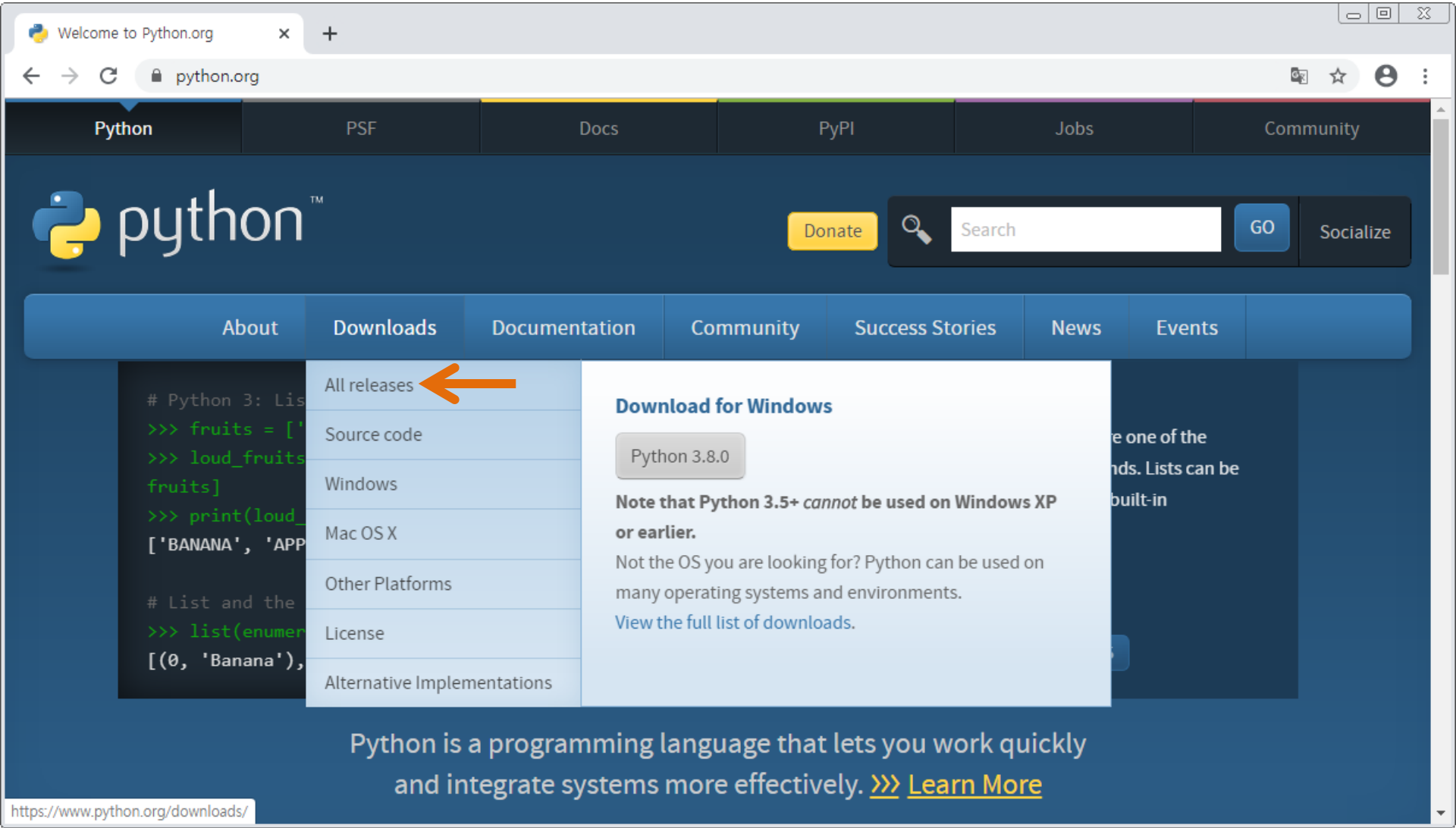
파이선 설치

python.org



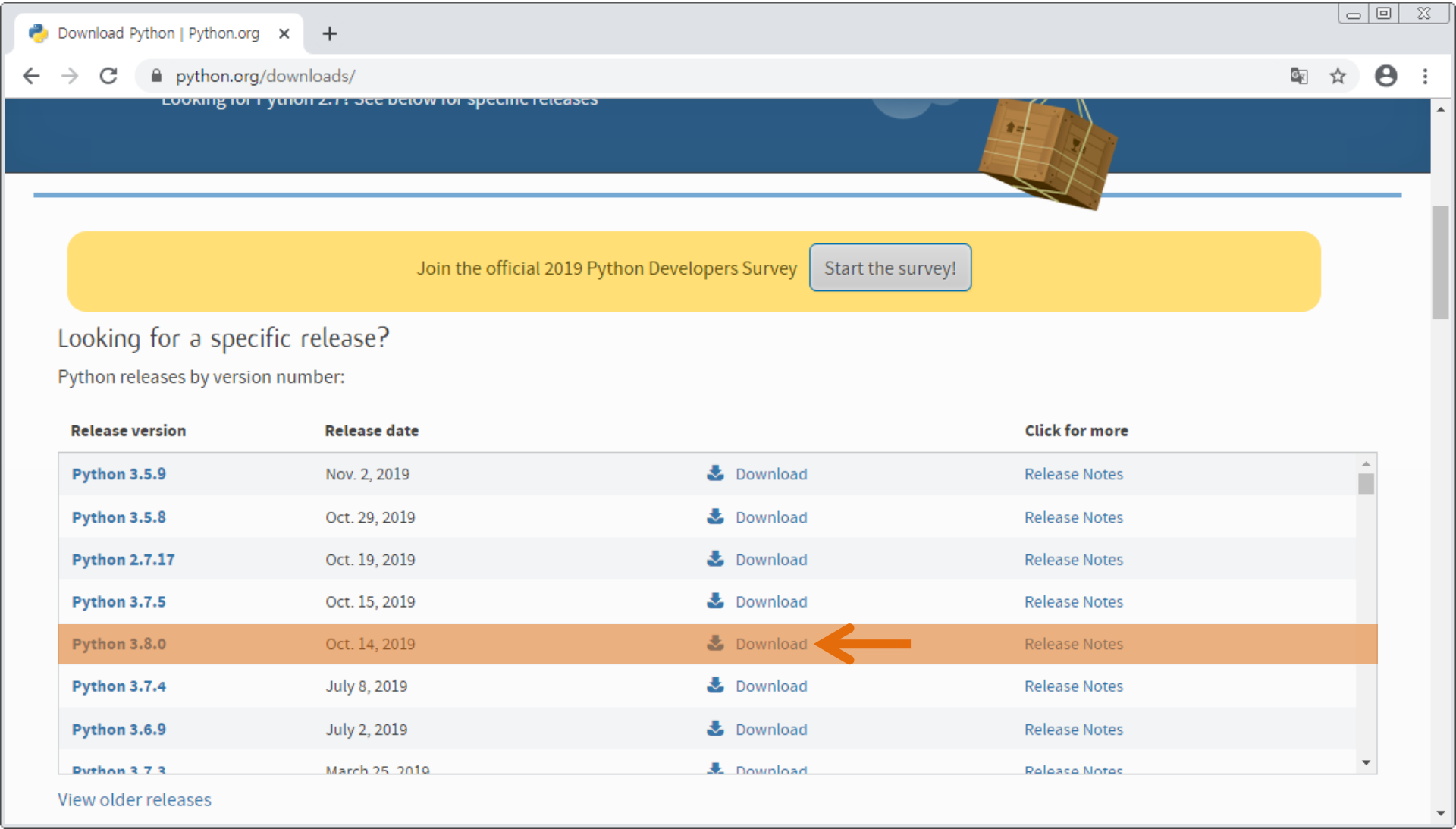
파이선 설치

OS에 맞는 파이선이 자동으로 선택되지 않는 경우



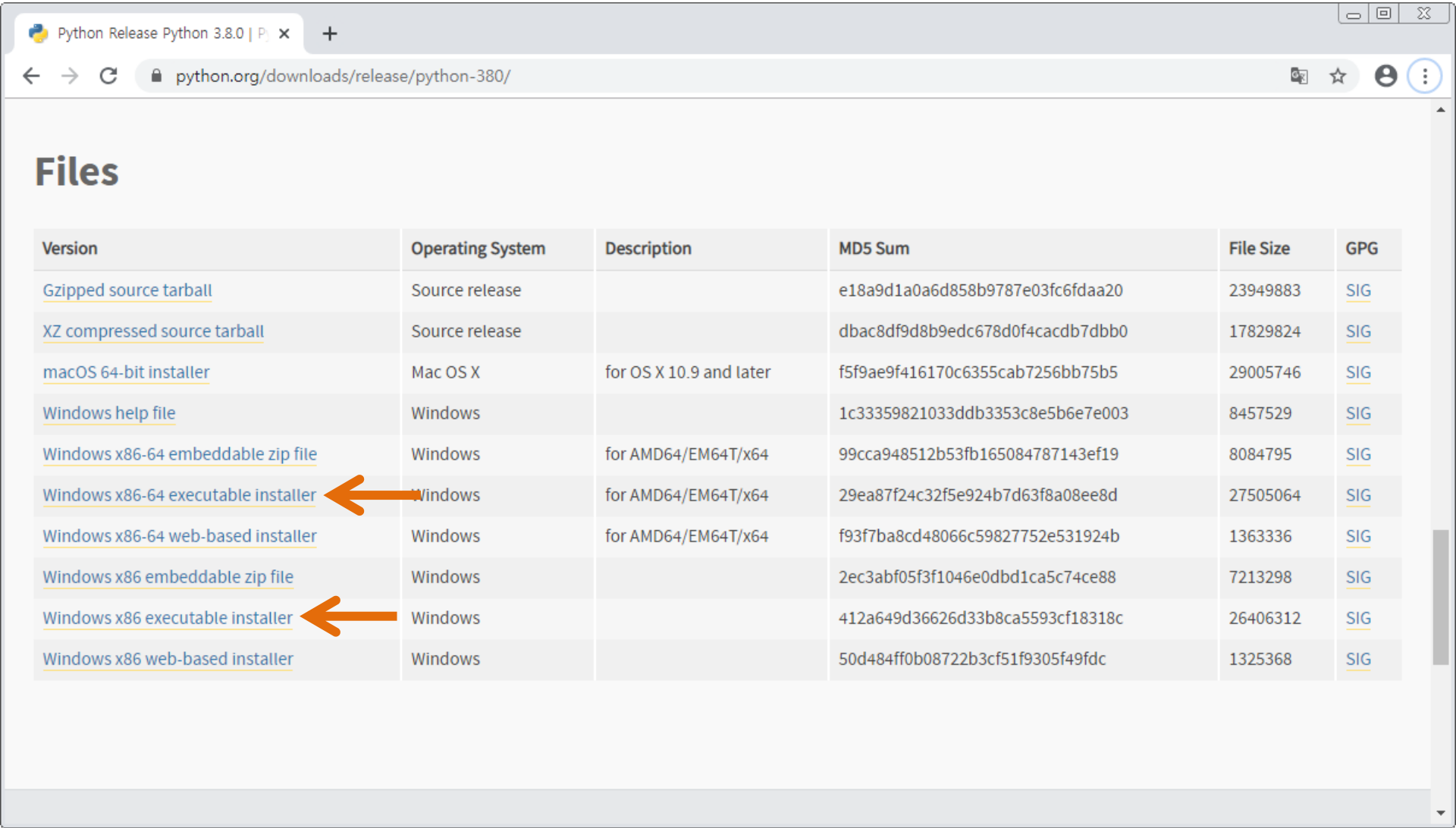
파이선 설치

아래로 스크롤 → Python 3.8.0의 Download 클릭



파이선 설치

아래로 스크롤 → OS에 맞는 설치 파일 내려 받기



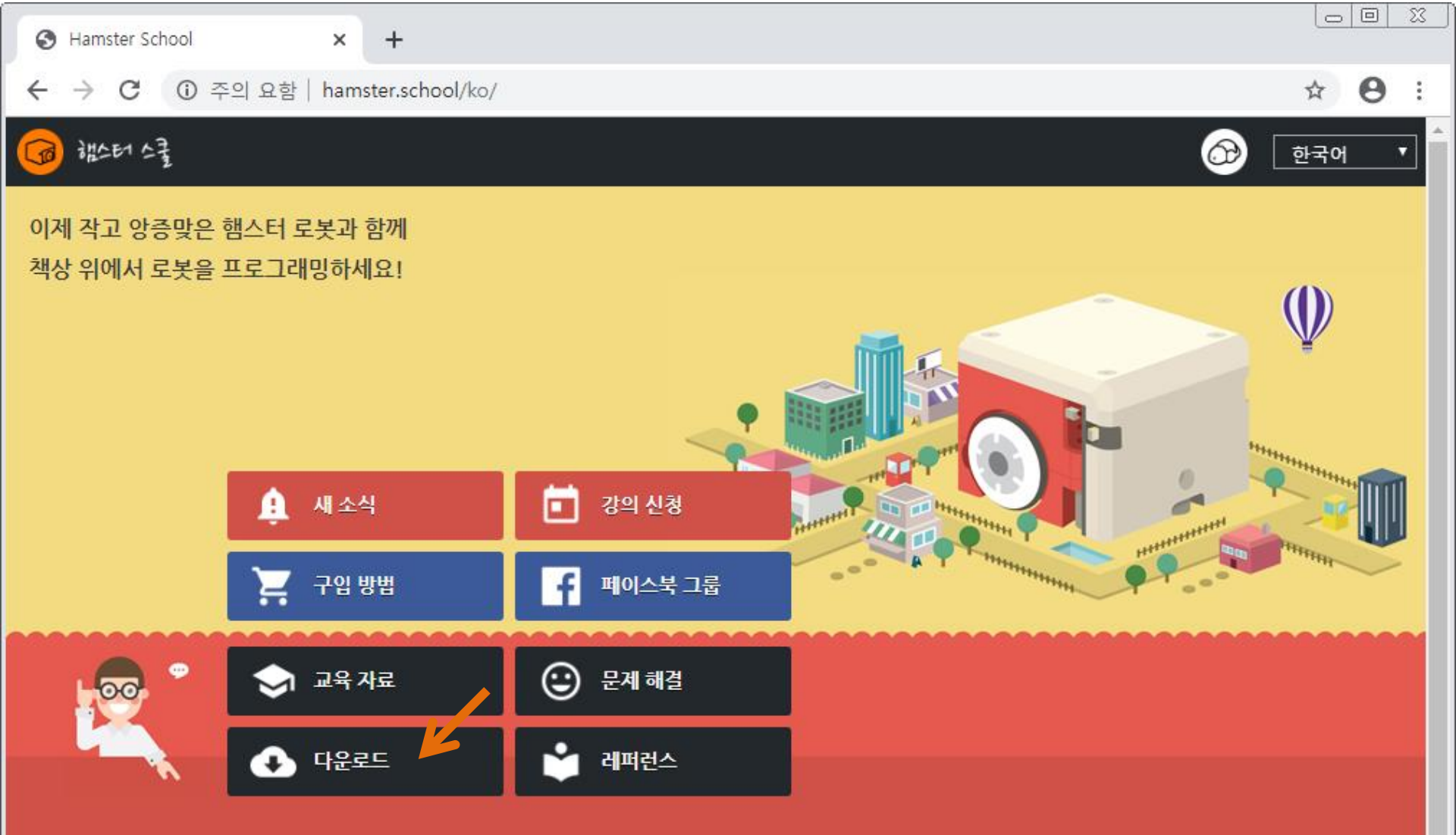
Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		e18a9d1a0a6d858b9787e03fc6fdaa20	23949883	SIG
XZ compressed source tarball	Source release		dbac8df9d8b9edc678d0f4cacdb7dbb0	17829824	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	f5f9ae9f416170c6355cab7256bb75b5	29005746	SIG
Windows help file	Windows		1c33359821033ddb3353c8e5b6e7e003	8457529	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	99cca948512b53fb165084787143ef19	8084795	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	29ea87f24c32f5e924b7d63f8a08ee8d	27505064	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	f93f7ba8cd48066c59827752e531924b	1363336	SIG
Windows x86 embeddable zip file	Windows		2ec3abf05f3f1046e0dbd1ca5c74ce88	7213298	SIG
Windows x86 executable installer	Windows		412a649d36626d33b8ca5593cf18318c	26406312	SIG
Windows x86 web-based installer	Windows		50d484ff0b08722b3cf51f9305f49fdc	1325368	SIG

햄스터 라이브러리 설치

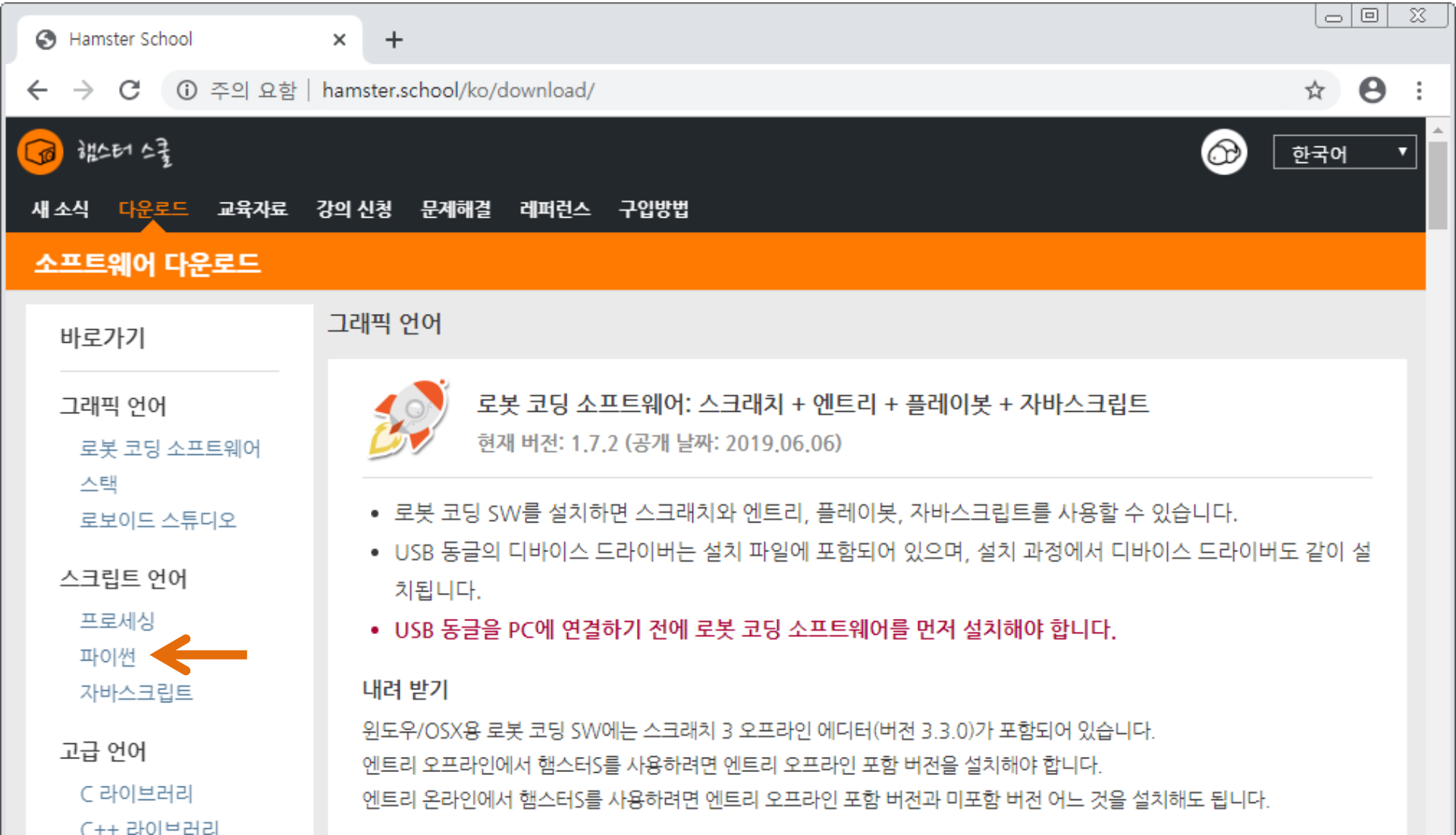
라이브러리 설치

hamster.school



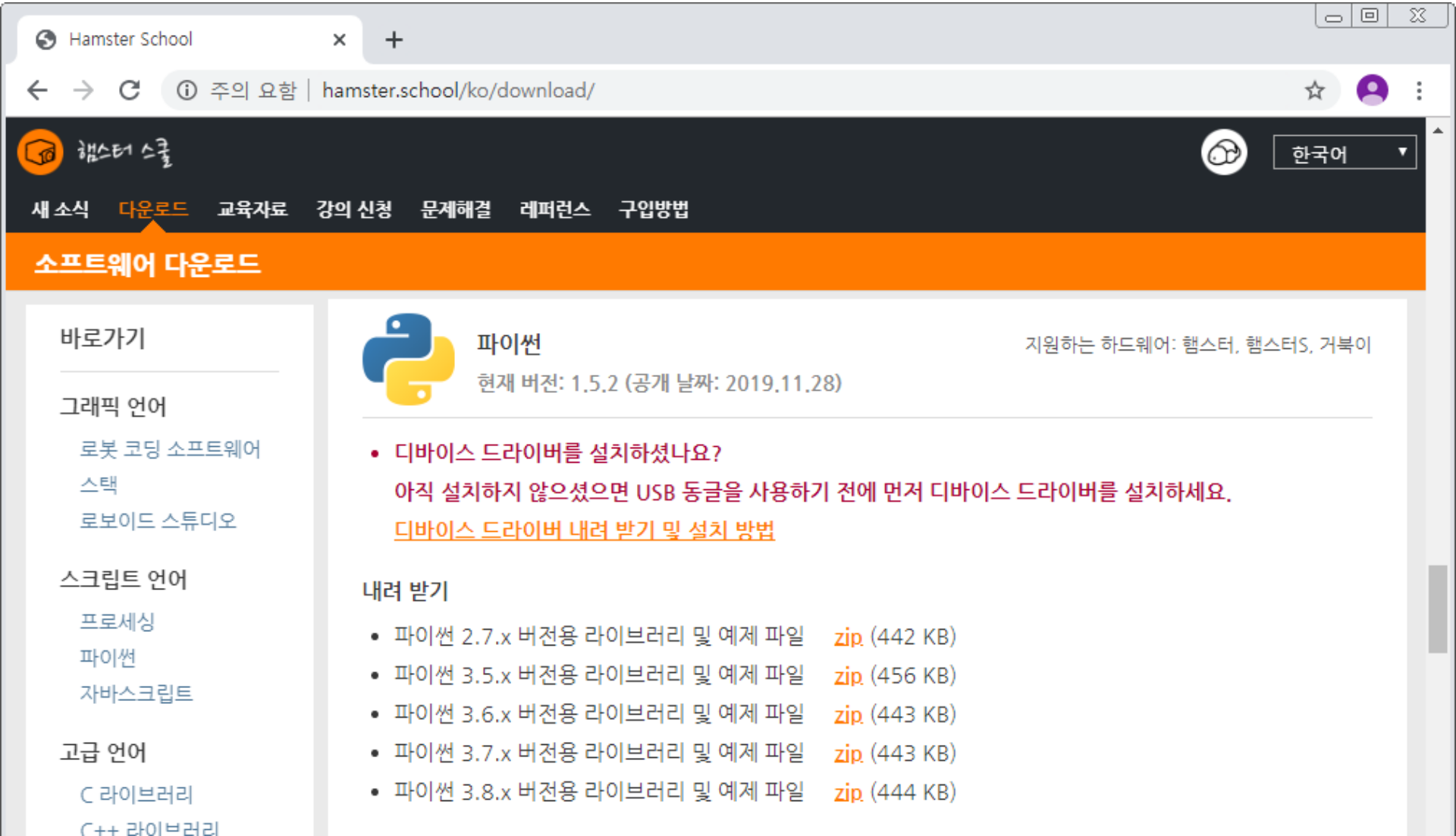
라이브러리 설치

바로가기에서 파이썬 클릭



라이브러리 설치

파이선 라이브러리 내려 받기



라이브러리 설치

압축을 풀고 win-setup.cmd 더블 클릭하여 설치

라이브러리 사용 시 아래 메시지 출력되면
라이브러리가 설치되지 않은 것임

```
ModuleNotFoundError: No module named 'roboid'
```

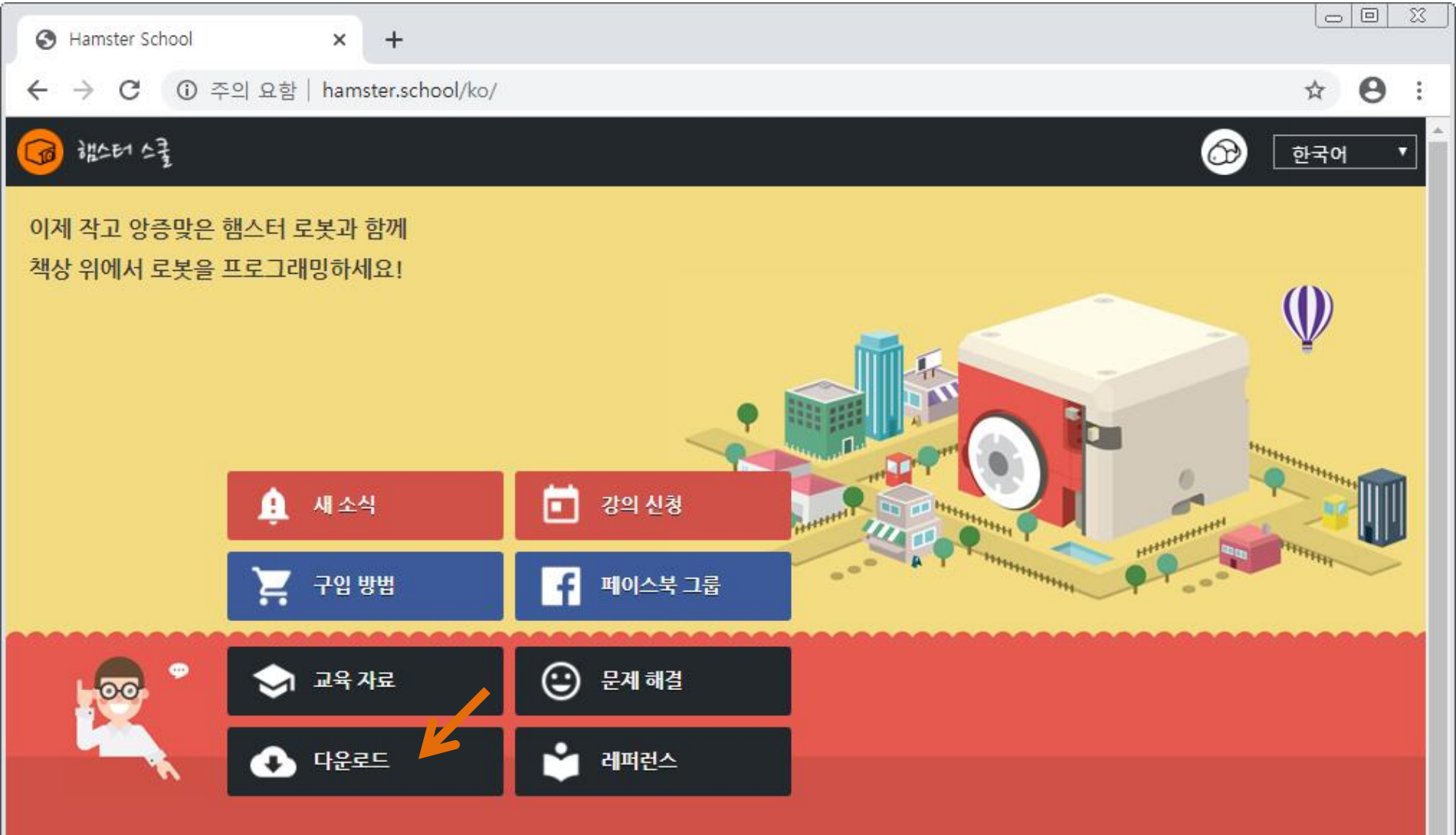
이 경우에는 압축을 푼 폴더의 site-packages 폴더에 있는 roboid와 serial 폴더를 아래 위치로 복사

[파이선 설치된 폴더]/Lib/site-packages

디바이스 드라이버 설치

디바이스 드라이버 설치

hamster.school



디바이스 드라이버 설치

바로가기에서 디바이스 드라이버 클릭

디바이스 드라이버 설치

디바이스 드라이버 내려 받기 및 설치 방법 클릭

소프트웨어 다운로드

바로가기

그래픽 언어
- 로봇 코딩 소프트웨어 스택
- 로보이드 스튜디오

스크립트 언어
- 프로세싱
- 파이썬
- 자바스크립트

고급 언어
- C 라이브러리
- C++ 라이브러리
- 자바 라이브러리
- 안드로이드 라이브러리

기타
- 디바이스 드라이버

내려 받기
- 안드로이드 라이브러리 (이클립스용)

지원하는 OS
- 안드로이드 4.3 젤리빈 이상

설치 및 실행 방법

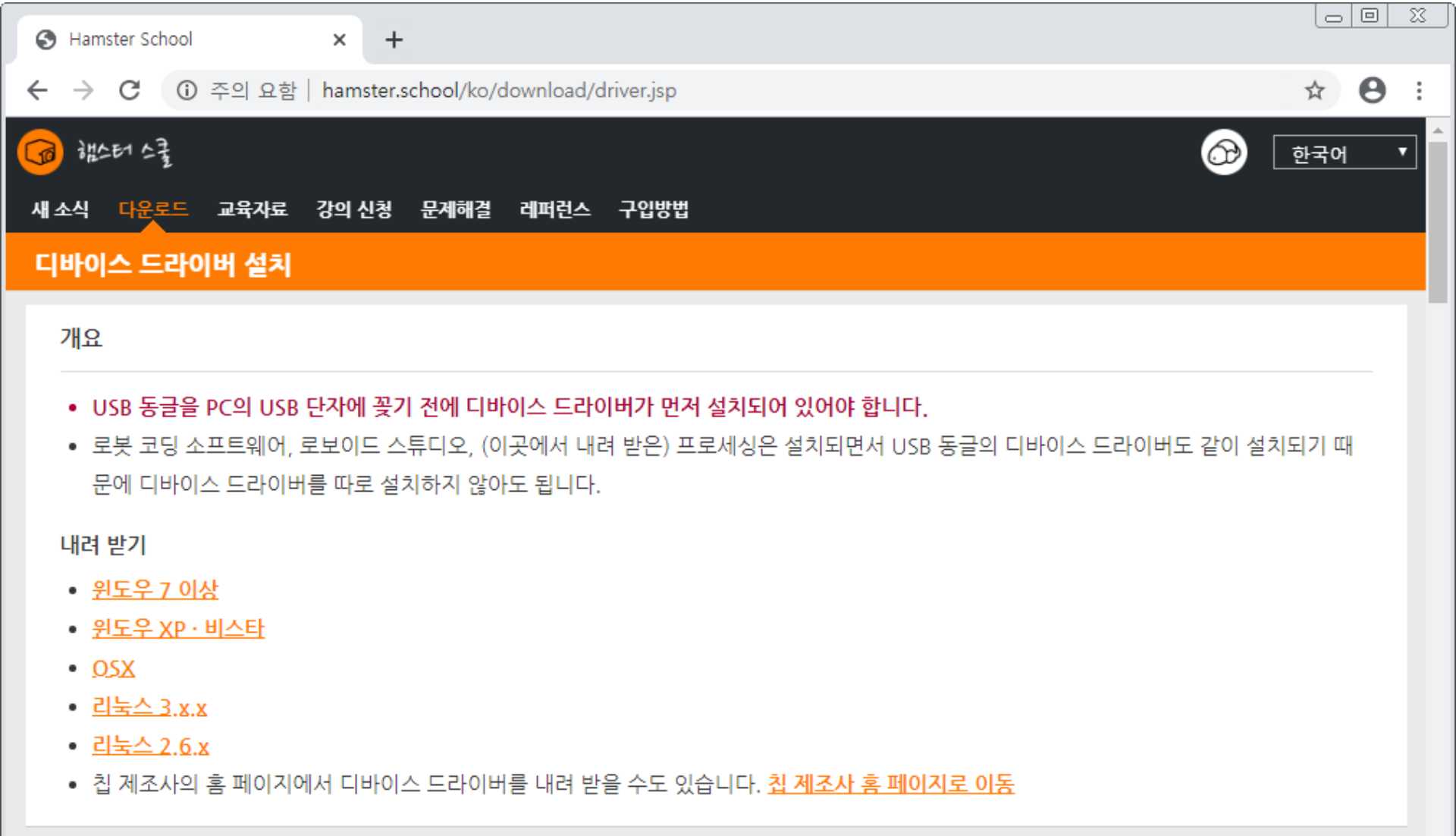
기타

USB 동글 디바이스 드라이버

- USB 동글의 디바이스 드라이버를 설치합니다.
  디바이스 드라이버 내려 받기 및 설치 방법

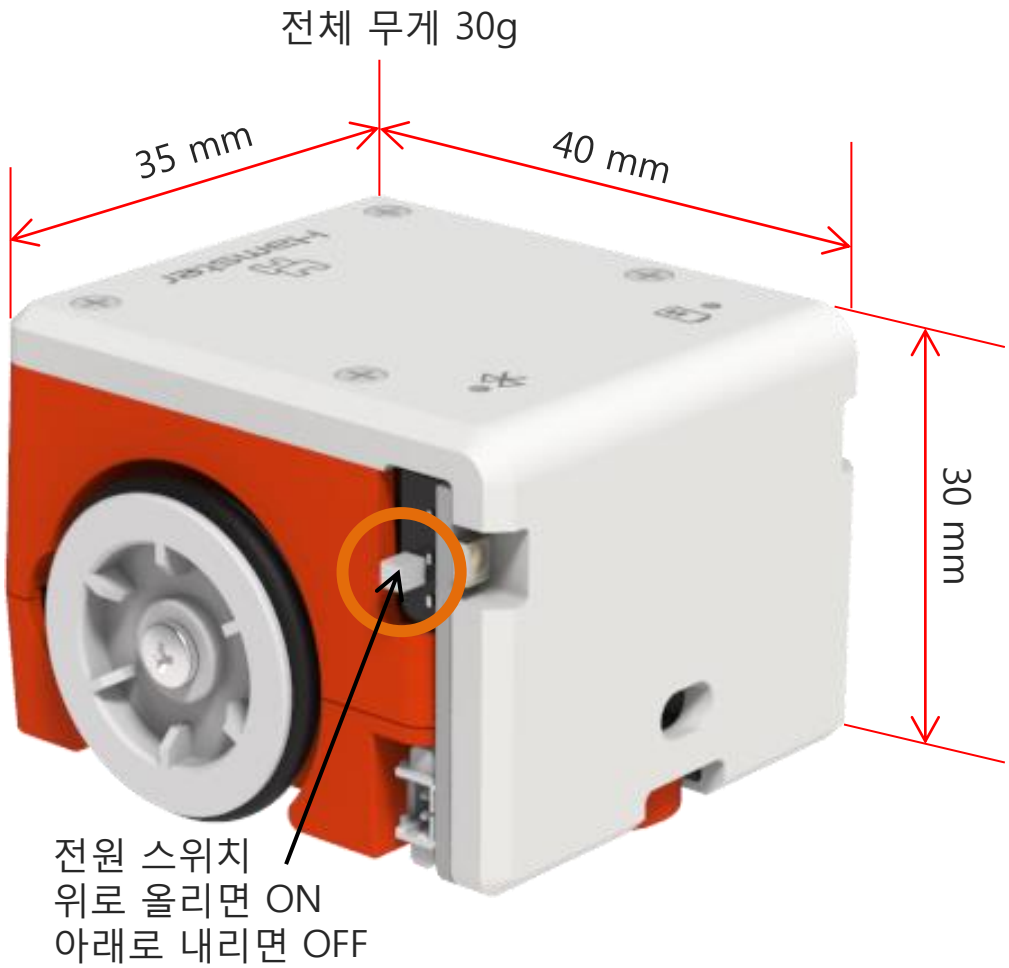
디바이스 드라이버 설치

OS에 맞는 디바이스 드라이버 내려 받아 설치

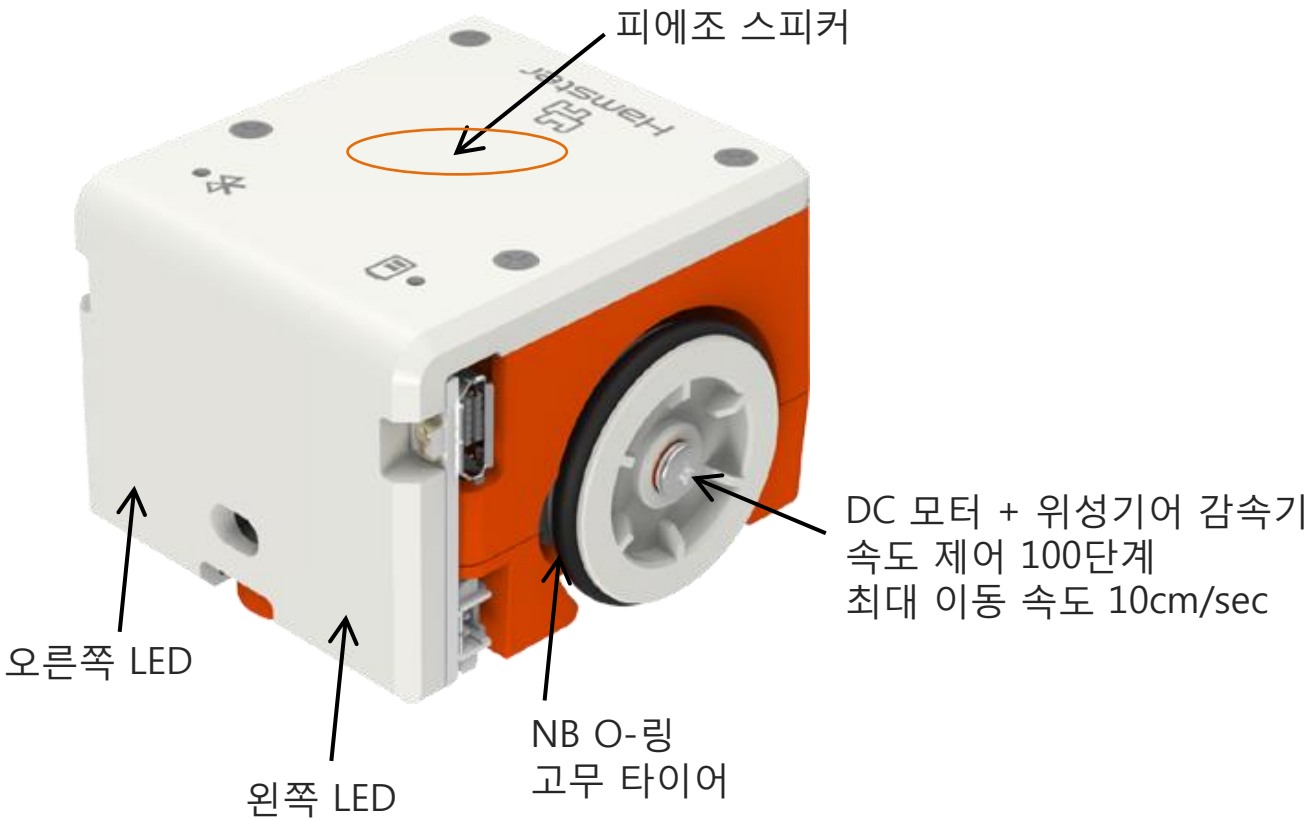


햄스터 로봇 소개

크기 · 무게 · 전원 스위치



출력 장치


피에조 스피커
DC 모터 + 위성기어 감속기
속도 제어 100단계
최대 이동 속도 10cm/sec
오른쪽 LED
왼쪽 LED
NB O-링
고무 타이어

입력 장치

근접 센서 (적외선 센서)
1~30cm, 1mm 정밀도
햇빛 아래 동작 가능

오른쪽 근접 센서
(송신)

왼쪽 근접 센서
(송신)

수신

밝기 센서
0~65,535룩스 감지

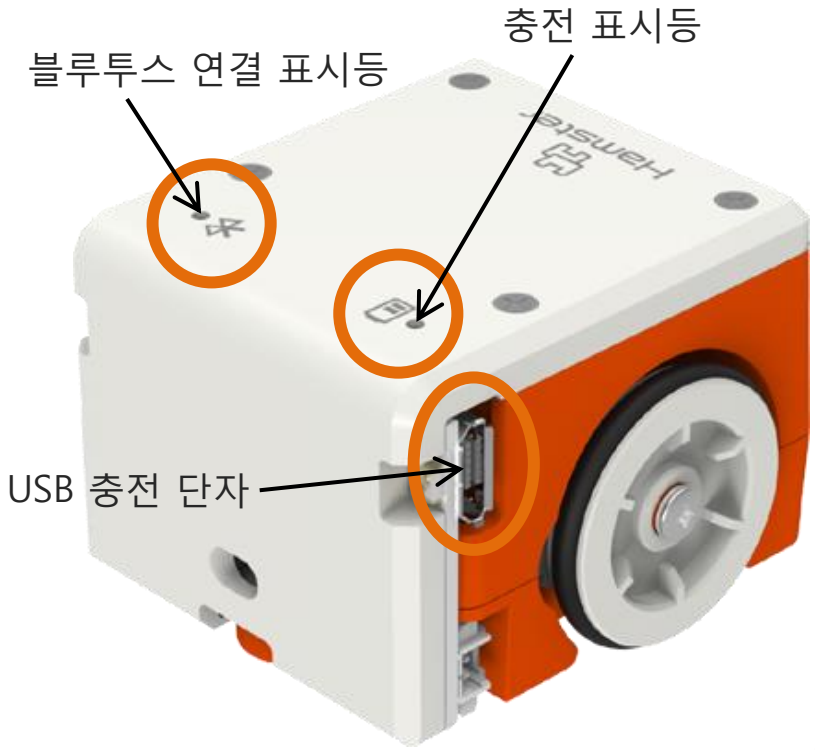
왼쪽 바닥 센서
(적외선 센서)
0~255단계 감지

3축 가속도 센서
2g, 4g, 8g, 16g 범위 설정
16비트 정밀도

내부 온도 센서
–40도 ~ +87.5도
0.5도 정밀도

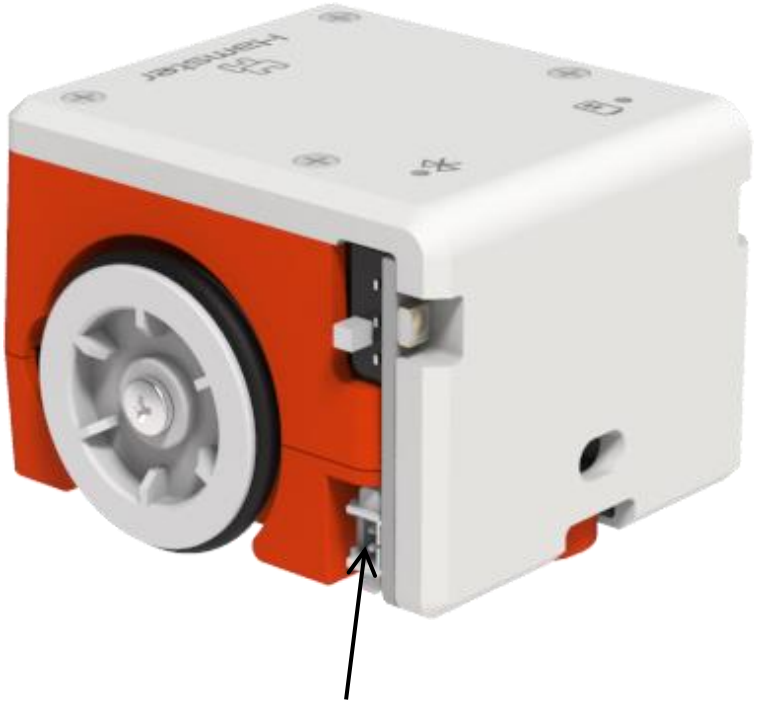
오른쪽 바닥 센서
(적외선 센서)
0~255단계 감지

표시등



내장 리튬 배터리 3.7V, 120mA
충전 약 30분
연속 동작 평균 1시간
대기 최대 12시간

외부 확장 포트

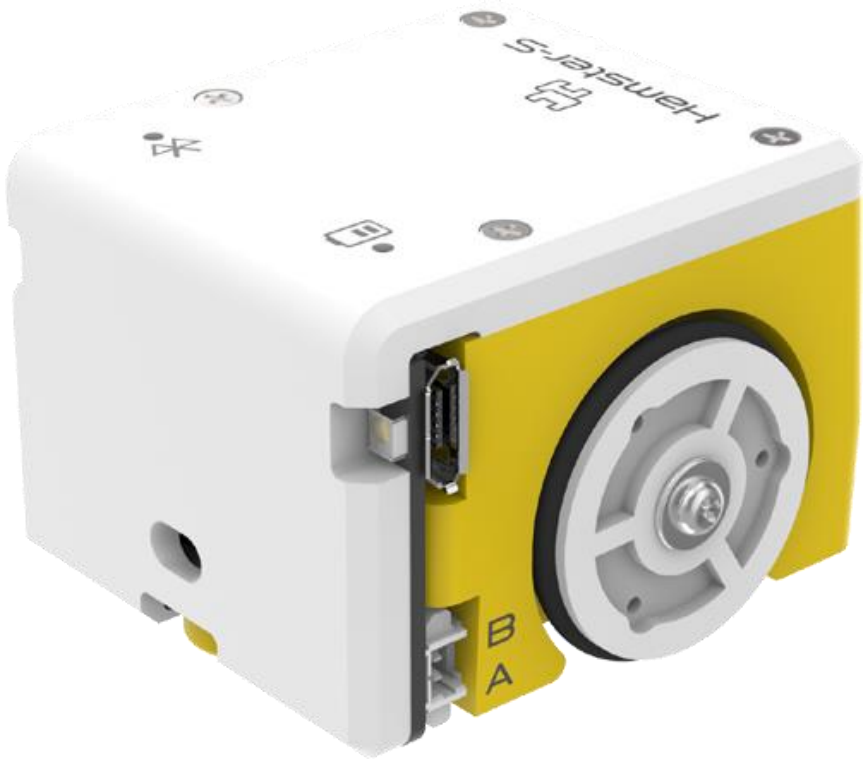


보조 전원 단자
3.7V 리튬 폴리머 전지

외부 입출력 단자 (포트A, 포트B)
디지털 입력, ADC 입력
디지털 출력, 아날로그(PWM) 출력
아날로그 서보 제어 출력

USB 동글

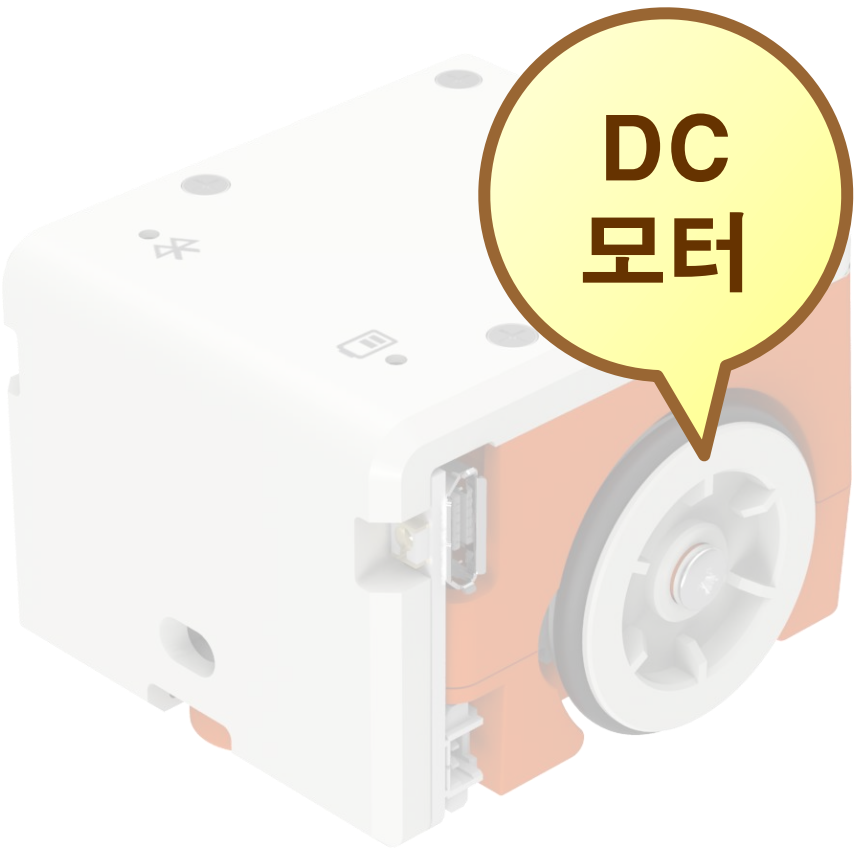
외관 비교



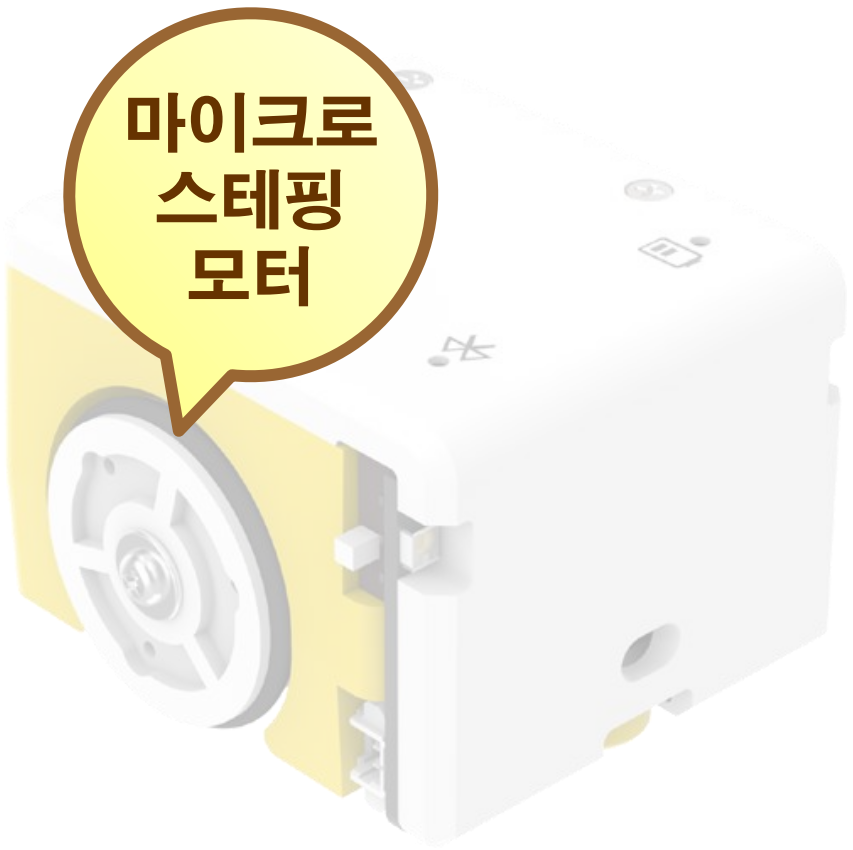
외관 비교

주요 차이점

HAMSTER

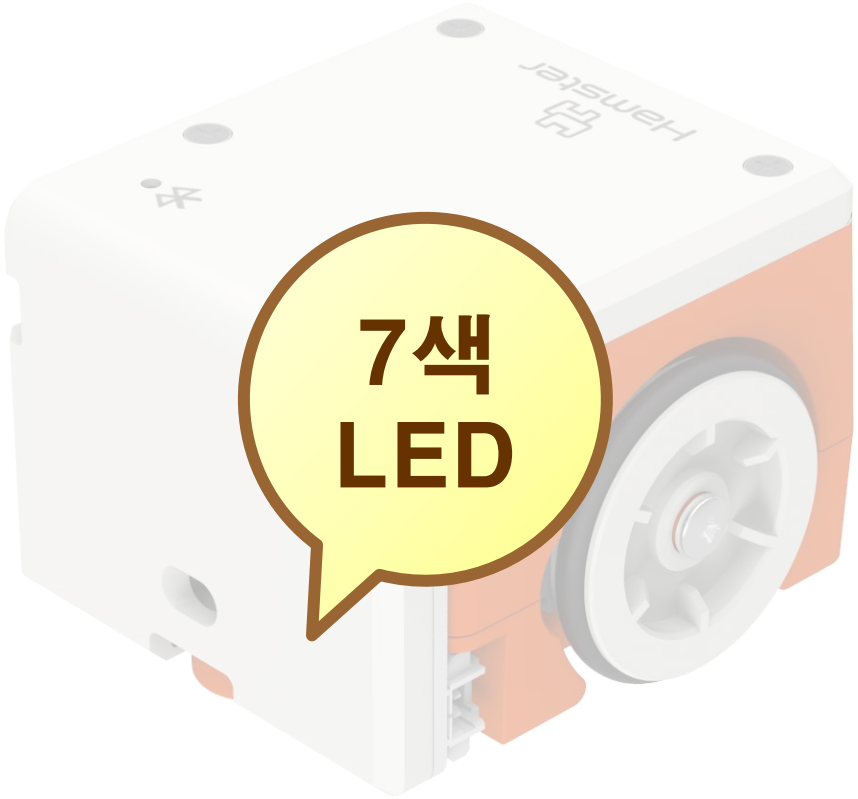


HAMSTER-S



주요 차이점

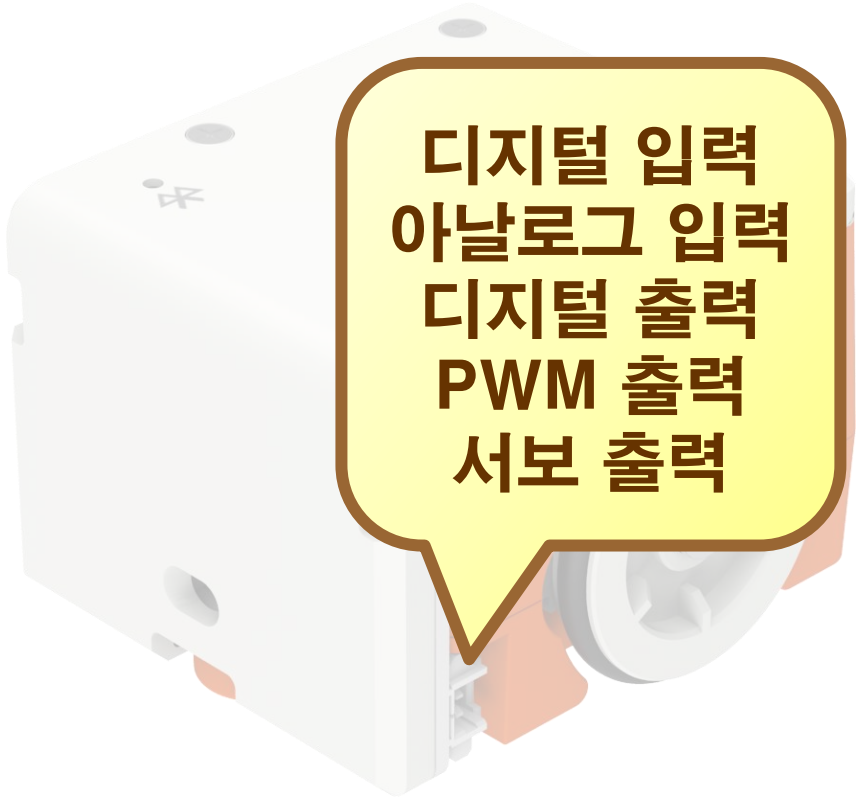
HAMSTER



HAMSTER-S

주요 차이점

HAMSTER



HAMSTER-S

햄스터 + 거북이 = 햄스터S

햄스터 로봇 연결

햄스터 로봇 연결

- USB 동글을 PC의 USB 단자에 연결

햄스터 로봇 연결

- 햄스터 로봇의 전원을 켜고
  USB 동글 가까이 가져 감 (15cm 이내)
- 삑 소리가 나면 연결된 것임

햄스터 연결


```
from roboid import *

hamster = Hamster()
```

- 실제 연결된 하드웨어가 햄스터 로봇이든 햄스터S 로봇이든 모두 Hamster의 메소드 사용

```
from roboid import *

hamster = HamsterS()
```

- 실제 연결된 하드웨어가 햄스터S 로봇이고 HamsterS의 메소드 사용

즉, 실제 하드웨어가 햄스터S 로봇인 경우
Hamster의 메소드를 사용할 수도 있고
HamsterS의 메소드를 사용할 수도 있다.

바닥 센서 사용하기

바닥 센서

왼쪽 바닥 센서
(적외선 센서)
0~100단계 감지

오른쪽 바닥 센서
(적외선 센서)
0~100단계 감지

센서 값 관찰


```
from roboid import *

hamster = Hamster()

while True:
    print(hamster.left_floor(), hamster.right_floor())
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

- hamster.left_floor()
- hamster.right_floor()
- wait(20)

정지선 지키기

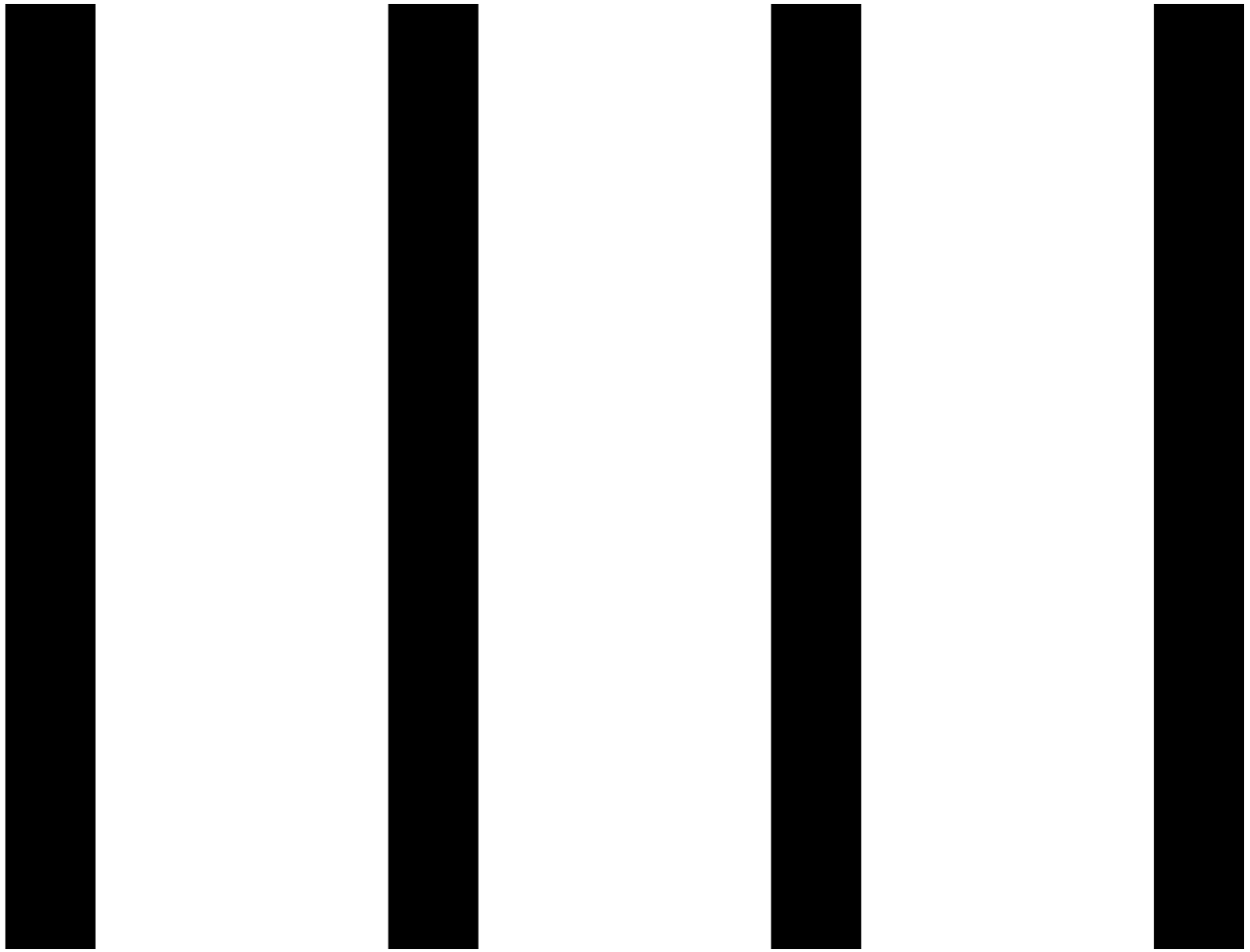
[redacted]

정지선 지키기


```
from roboid import *

hamster = Hamster()

while hamster.left_floor() > 20 and hamster.right_floor() > 20:
    hamster.wheels(30, 30) # 앞으로 이동한다.
    wait(20) # 너무 빨리 반복하지 않도록 한다.

hamster.stop() # 정지한다.
```

- **hamster.wheels(30,30)**
- **hamster.wheels(30)**
- **hamster.left_wheel(30)**
- **hamster.right_wheel(30)**
- **hamster.stop()**

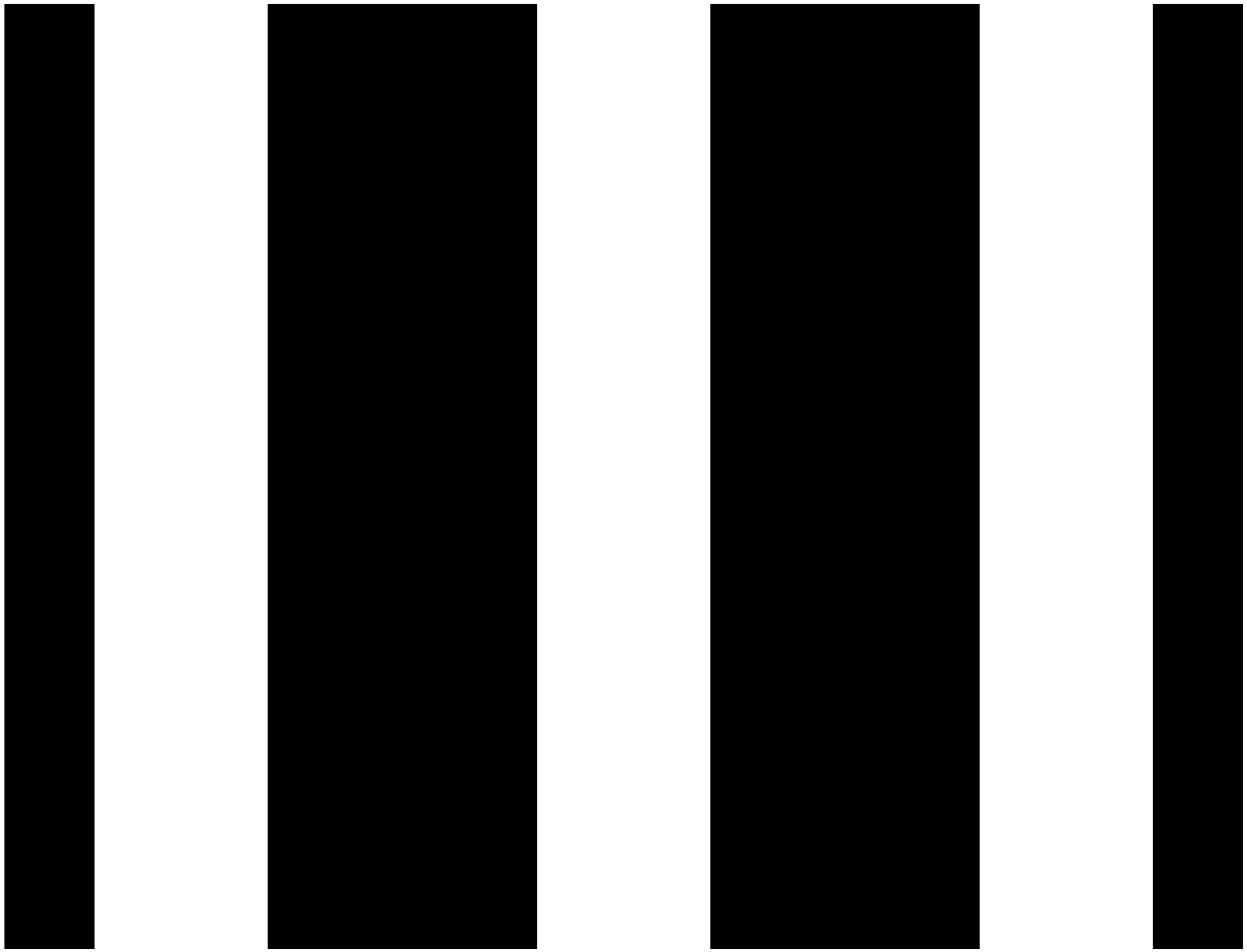
일정 간격의 검은색 선 개수 세기



일정 간격의 검은색 선 개수 세기


```
from roboid import *

hamster = Hamster()

hamster.wheels(30, 30) # 앞으로 이동한다.

count = 0
while True:
    if hamster.left_floor() < 20 or hamster.right_floor() < 20:
        count += 1
        print('count:', count)

        hamster.wheels(30, 30) # 1초 동안 앞으로 이동한다.
        wait(1000)
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

임의 간격 및 굵기의 검은색 선 개수 세기



임의 간격 및 굵기의 검은색 선 개수 세기


```
from roboid import *

hamster = Hamster()

hamster.wheels(30, 30) # 앞으로 이동한다.

count = 0
white = False
while True:
    if hamster.left_floor() > 80 and hamster.right_floor() > 80:
        white = True # 하얀색 종이 위에 있다.
    elif white and (hamster.left_floor() < 20 or hamster.right_floor() < 20):
        white = False
        count += 1
        print('count:', count)

        hamster.wheels(30, 30) # 1초 동안 앞으로 이동한다.
        wait(1000)
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

바코드 읽기

[redacted]

바코드 읽기


```
from roboid import *

hamster = Hamster()

hamster.wheels(30, 30) # 앞으로 이동한다.
tick = 0
code = ''
while True:
    if hamster.left_floor() > 80 and hamster.right_floor() > 80:
        if tick > 0:
            if tick > 40: # 선이 굵다.
                code += '-'
            else: # 선이 가늘다.
                code += '.'
            print(code) # 바코드 확인
            if code == '.--.':
                print('peanut')
                code = ''
            elif code == '-..-':
                print('cheese')
                code = ''
        tick = 0
    elif hamster.left_floor() < 20 or hamster.right_floor() < 20:
        tick += 1
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

키보드 사용하기

오리걸음 경주

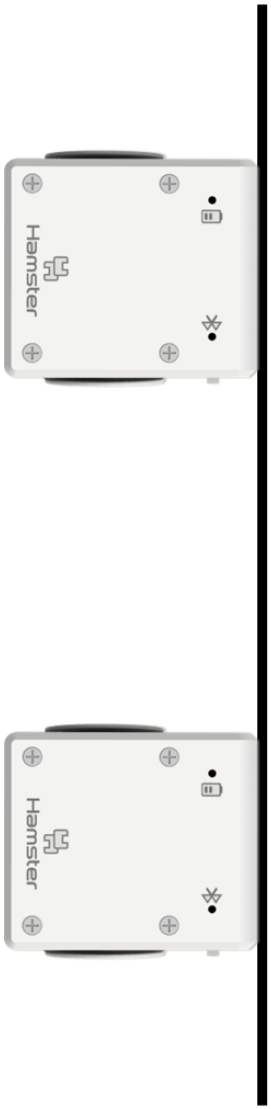
오리걸음 경주


```
from roboid import *

hamster = Hamster()

left = False
while True:
    key = Keyboard.read() # 키보드 이벤트를 얻는다.
    if key == ' ': # 스페이스 키를 눌렀으면
        left = not left # 방향을 반대로 한다.
        if left: # 왼쪽 방향이면
            hamster.wheels(0, 50)
        else: # 오른쪽 방향이면
            hamster.wheels(50, 0)

    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

- Keyboard.read()

```
>> python test.py
```

오리걸음 경주



종이컵

종이컵

햄스터 피아노

- `hamster.note("C4")`
- `hamster.note("D#4")`
- `hamster.note("Db4")`
- `hamster.note("C4", 0.5)`
- `hamster.note("D#4", 0.5)`
- `hamster.note("Db4", 0.5)`

- `hamster.note("OFF")`
- `hamster.note("OFF", 0.5)`

음표 길이를 정해 주지 않으면 소리가 계속 남

음표 길이 (박)

```
"OFF"
"C1"
...
"C4"
"C#4"
"Db4"
"D4"
"D#4"
"Eb4"
"E4"
"F4"
"F#4"
"Gb4"
"G4"
"G#4"
"Ab4"
"A4"
"A#4"
"Bb4"
"B4"
...
"B7"
```

햄스터 피아노

```
from roboid import *
hamster = Hamster()
while True:
    key = Keyboard.read() # 키보드 이벤트를 얻는다.
    if key: # 키보드 이벤트가 있으면
        if key == 'a':
            hamster.note("C4") # 도
        elif key == 's':
            hamster.note("D4") # 레
        elif key == 'd':
            hamster.note("E4") # 미
        elif key == 'f':
            hamster.note("F4") # 파
        elif key == 'g':
            hamster.note("G4") # 솔
        elif key == 'h':
            hamster.note("A4") # 라
        elif key == 'j':
            hamster.note("B4") # 시
        elif key == 'k':
            hamster.note("C5") # 도
        elif key == 'l':
            hamster.note("D5") # 레
        elif key == ';':
            hamster.note("E5") # 미
        elif key == "'":
            hamster.note("F5") # 파
        elif key == ' ': # 스페이스 키
            hamster.note("OFF") # 소리를 끈다.
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

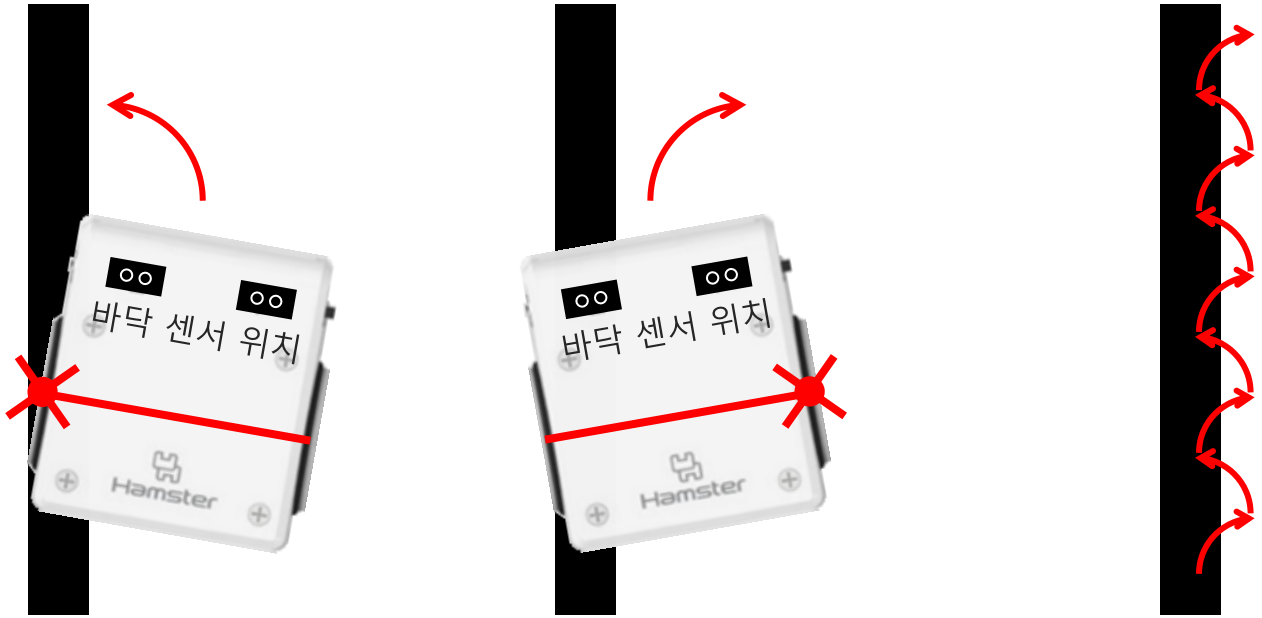
햄스터 피아노

```
from roboid import *
hamster = Hamster()
notes = {
    " ": "OFF", # 소리를 끈다.
    "a": "C4", # 도
    "s": "D4", # 레
    "d": "E4", # 미
    "f": "F4", # 파
    "g": "G4", # 솔
    "h": "A4", # 라
    "j": "B4", # 시
    "k": "C5", # 도
    "l": "D5", # 레
    ";": "E5", # 미
    "'": "F5" # 파
}
while True:
    key = Keyboard.read() # 키보드 이벤트를 얻는다.
    if key and key in notes: # 키보드 이벤트가 딕셔너리에 있으면
        hamster.note("OFF", 0.05) # 0.05 박자로 잠시 쉰다.
        hamster.note(notes[key])
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

```
from roboid import *
hamster = Hamster()
notes = {
    " ": "OFF", # 소리를 끈다.
    "a": "C4", # 도
    "w": "C#4", # 도# (레b)
    "s": "D4", # 레
    "e": "Eb4", # 미b (레#)
    "d": "E4", # 미
    "f": "F4", # 파
    "t": "F#4", # 파# (솔b)
    "g": "G4", # 솔
    "y": "G#4", # 솔# (라b)
    "h": "A4", # 라
    "u": "Bb4", # 시b (라#)
    "j": "B4", # 시
    "k": "C5", # 도
    "o": "C#5", # 도# (레b)
    "l": "D5", # 레
    "p": "Eb5", # 미b (레#)
    ";": "E5", # 미
    "'": "F5" # 파
}
while True:
    key = Keyboard.read() # 키보드 이벤트를 얻는다
```

라인 트레이싱

왼쪽 센서 + 오른쪽 가장자리

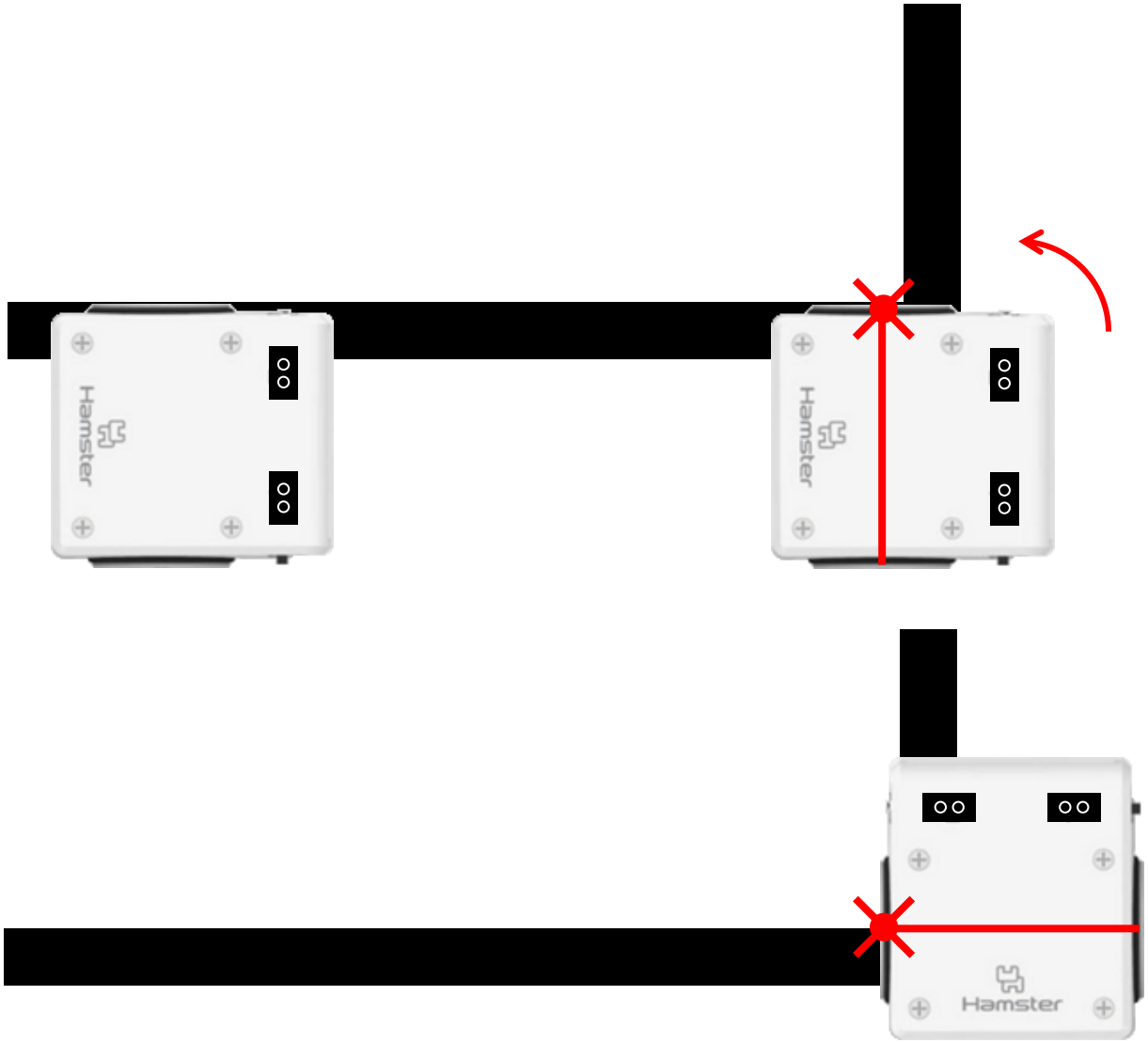


왼쪽 센서 + 오른쪽 가장자리

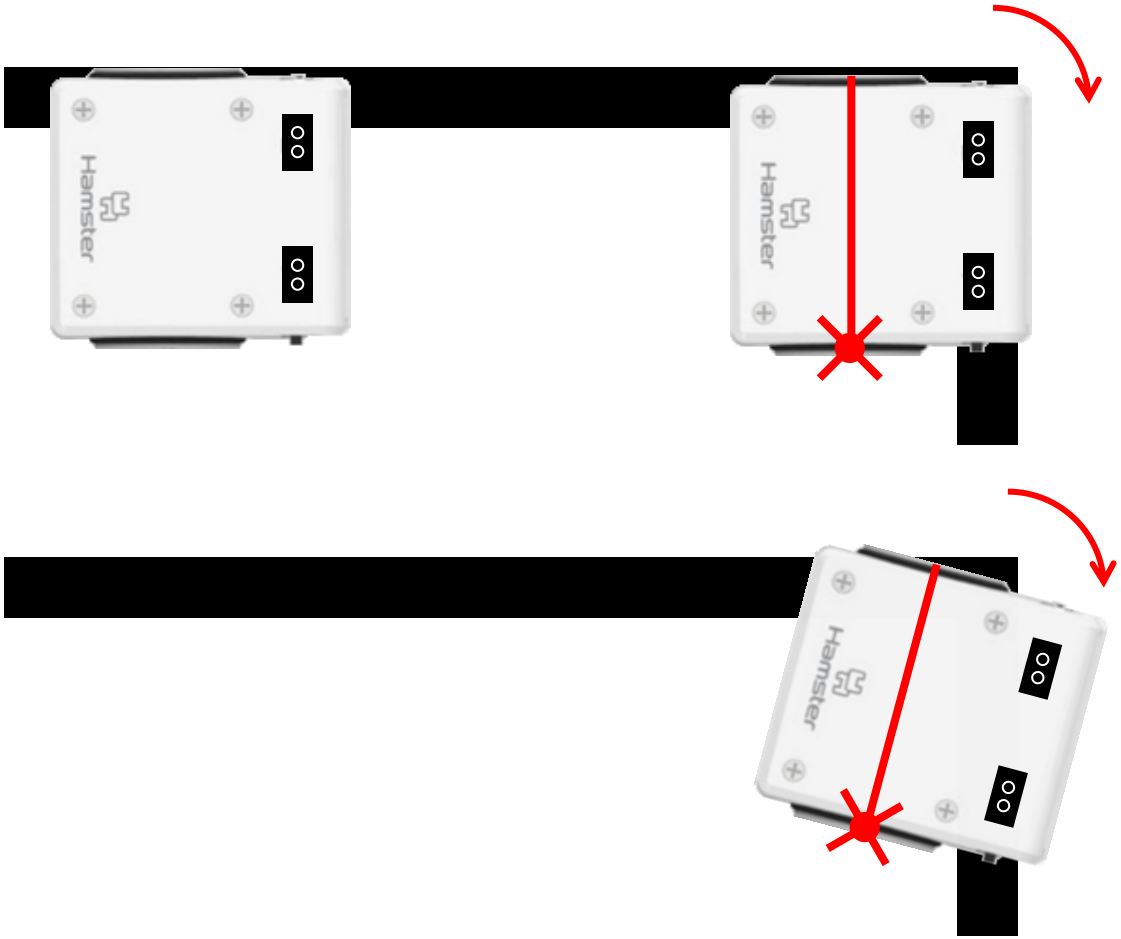

```
from roboid import *

hamster = Hamster()

while True:
    if hamster.left_floor() > 50:
        hamster.wheels(0, 30)
    else:
        hamster.wheels(30, 0)
    wait(10) # 너무 빨리 반복하지 않도록 한다.
```

왼쪽 센서 + 오른쪽 가장자리

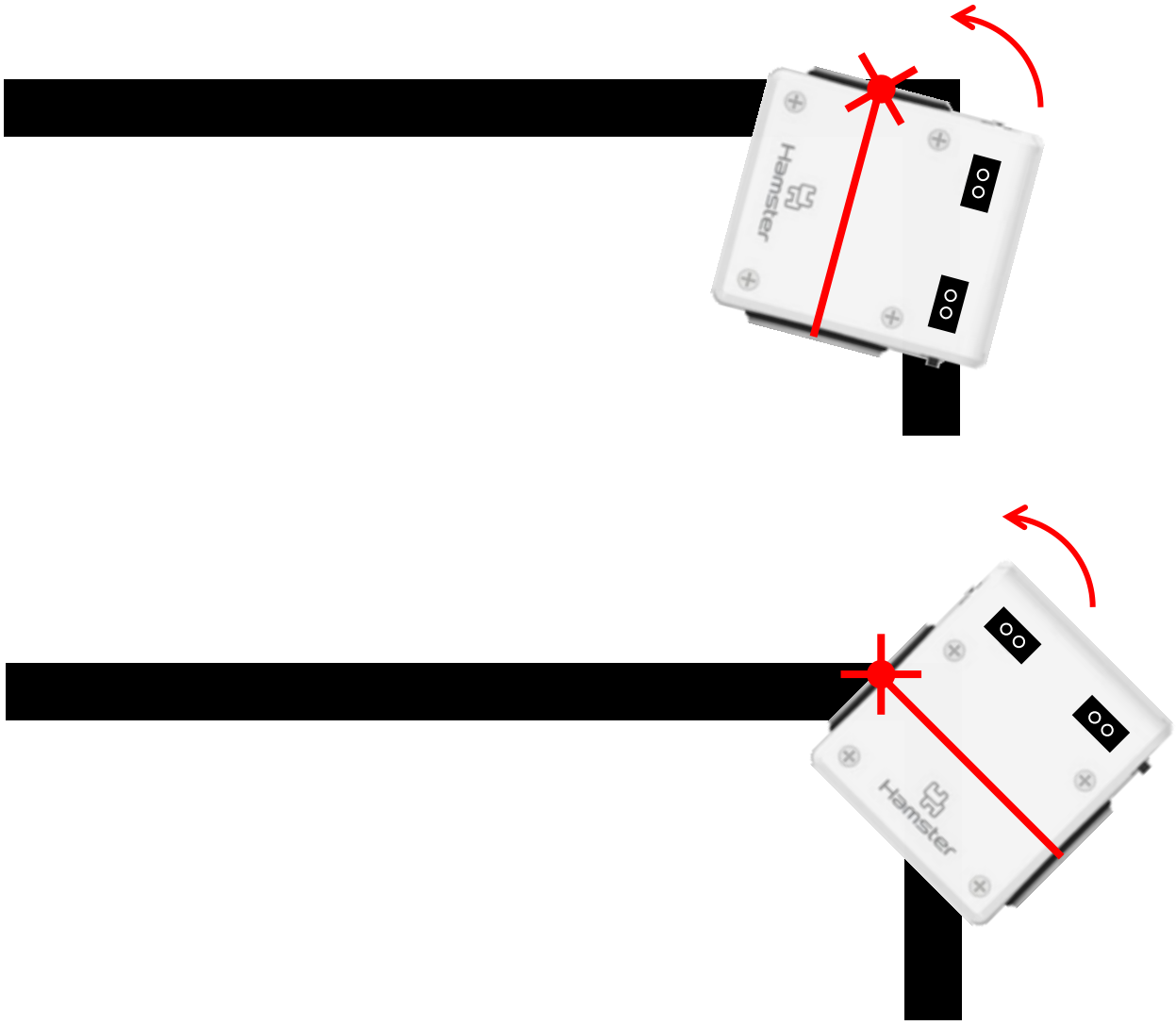
왼쪽 센서 + 오른쪽 가장자리



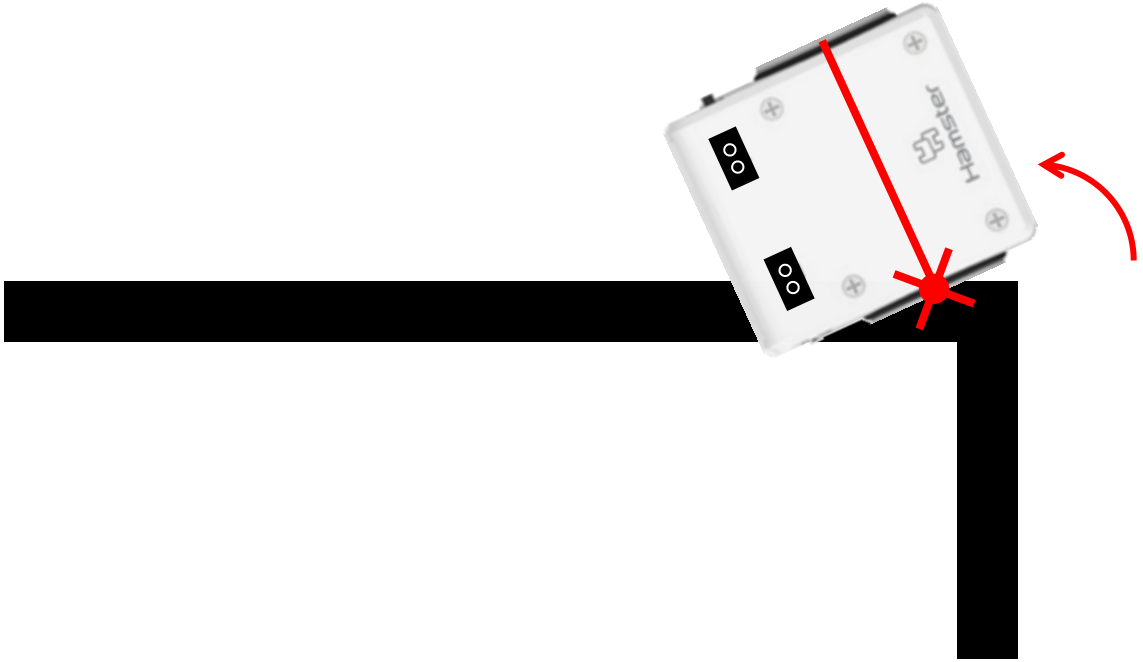
왼쪽 센서 + 오른쪽 가장자리



왼쪽 센서 + 오른쪽 가장자리



왼쪽 센서 + 오른쪽 가장자리

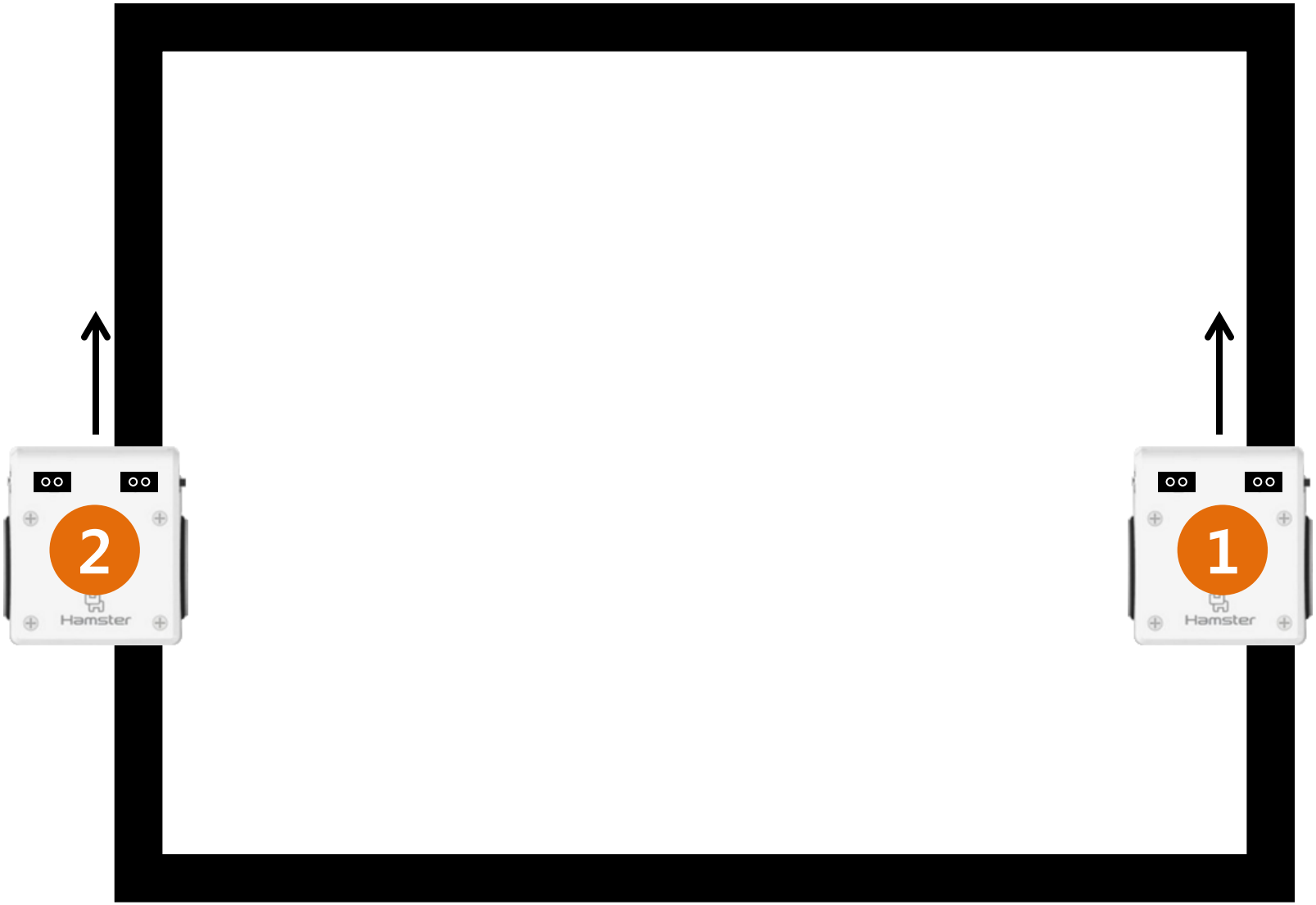


왼쪽 센서 + 왼쪽 가장자리

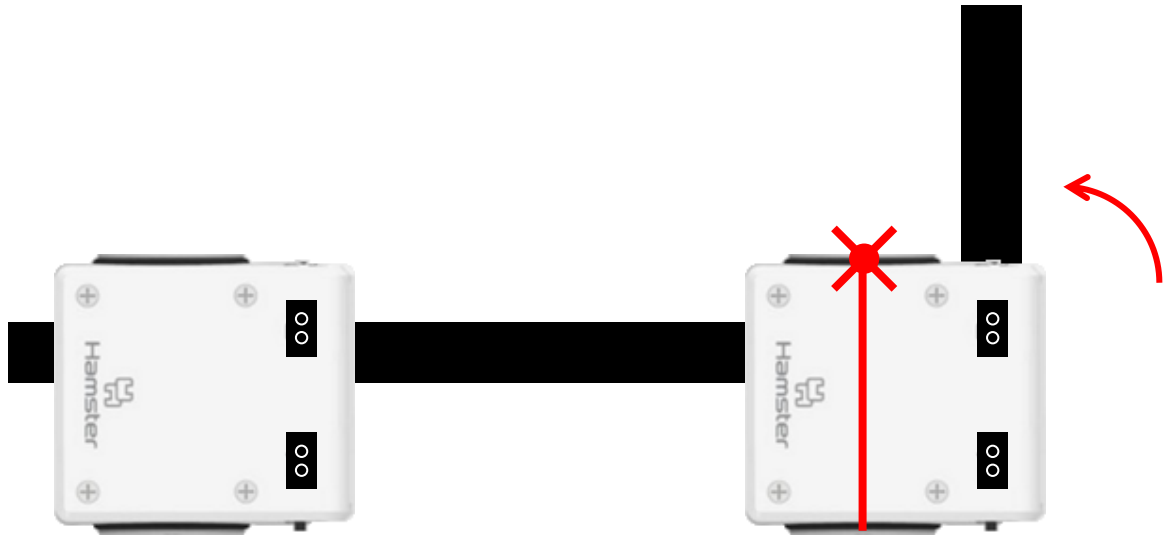
왼쪽 센서 + 왼쪽 가장자리


```
from roboid import *

hamster = Hamster()

while True:
    if hamster.left_floor() > 50:
        hamster.wheels(30, 0)
    else:
        hamster.wheels(0, 30)
    wait(10) # 너무 빨리 반복하지 않도록 한다.
```

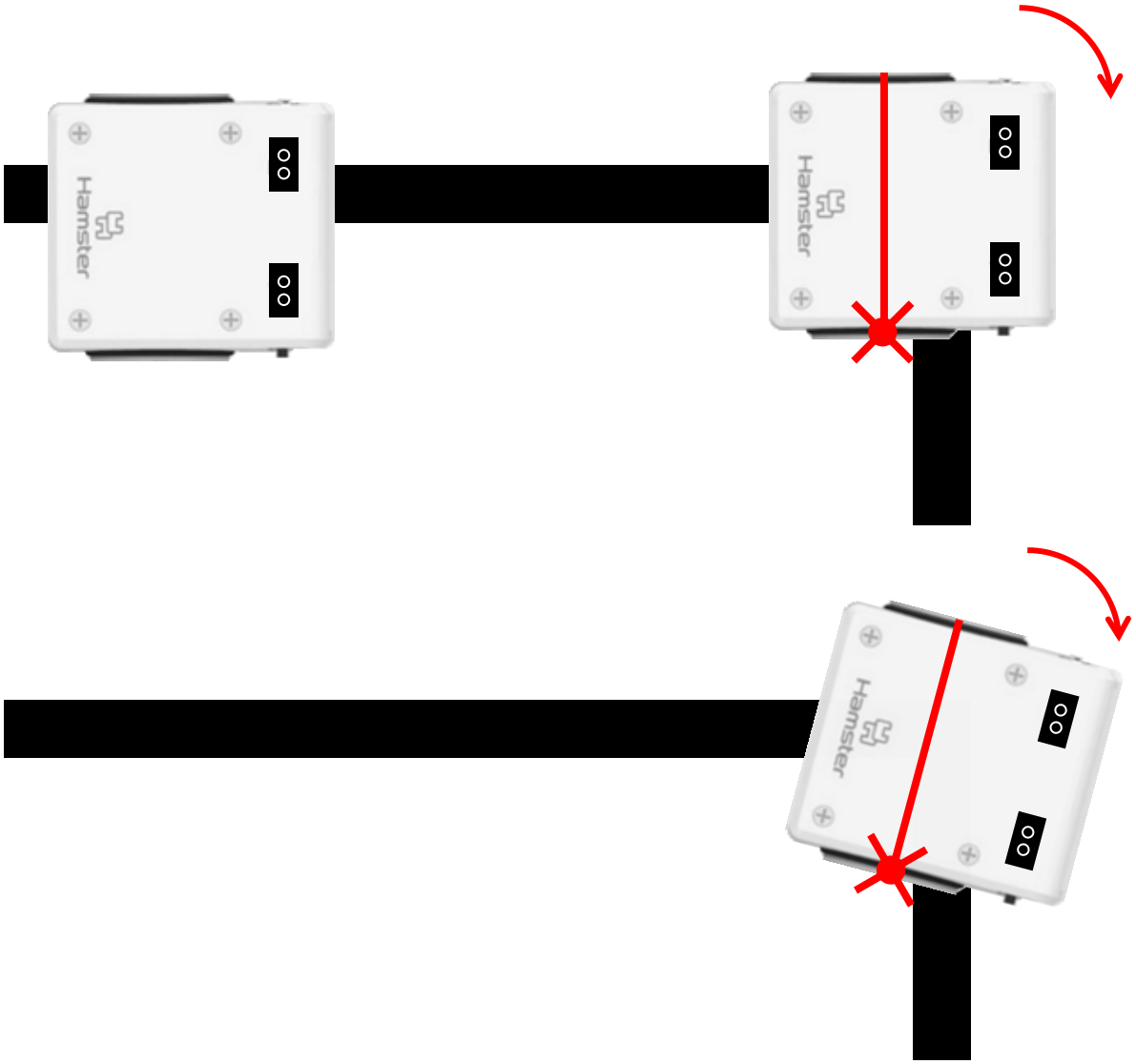
왼쪽 센서 + 왼쪽 가장자리



왼쪽 센서 + 왼쪽 가장자리

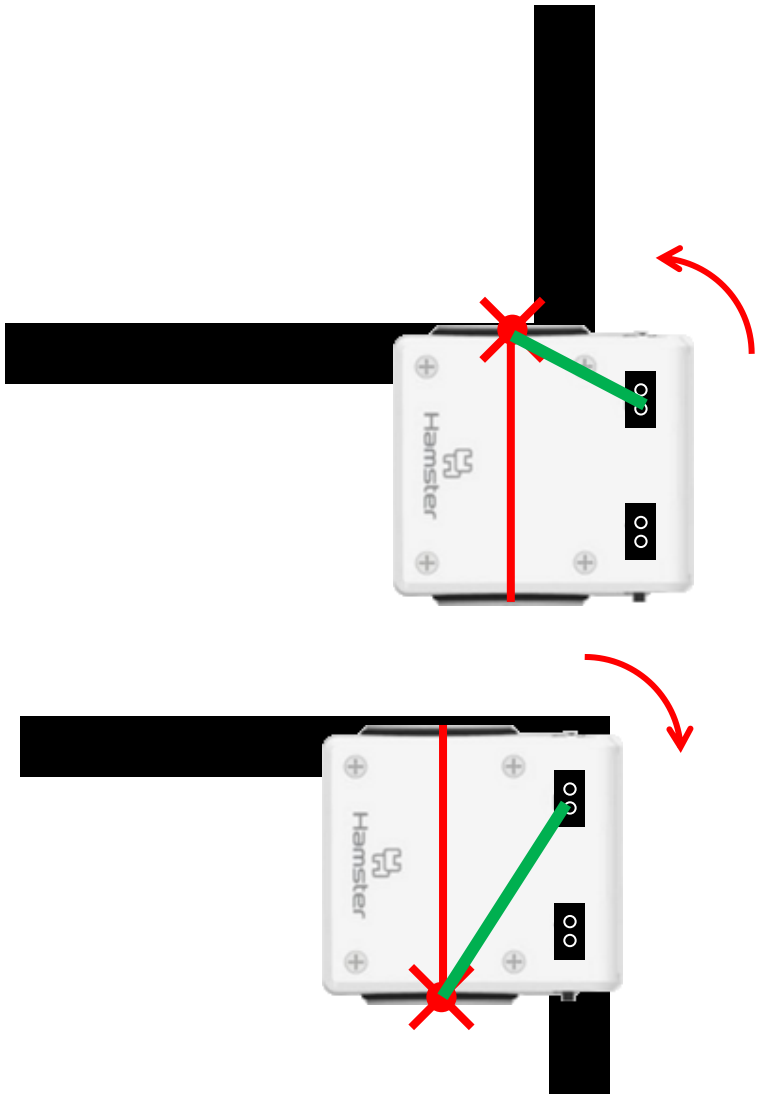


왼쪽 센서 + 왼쪽 가장자리

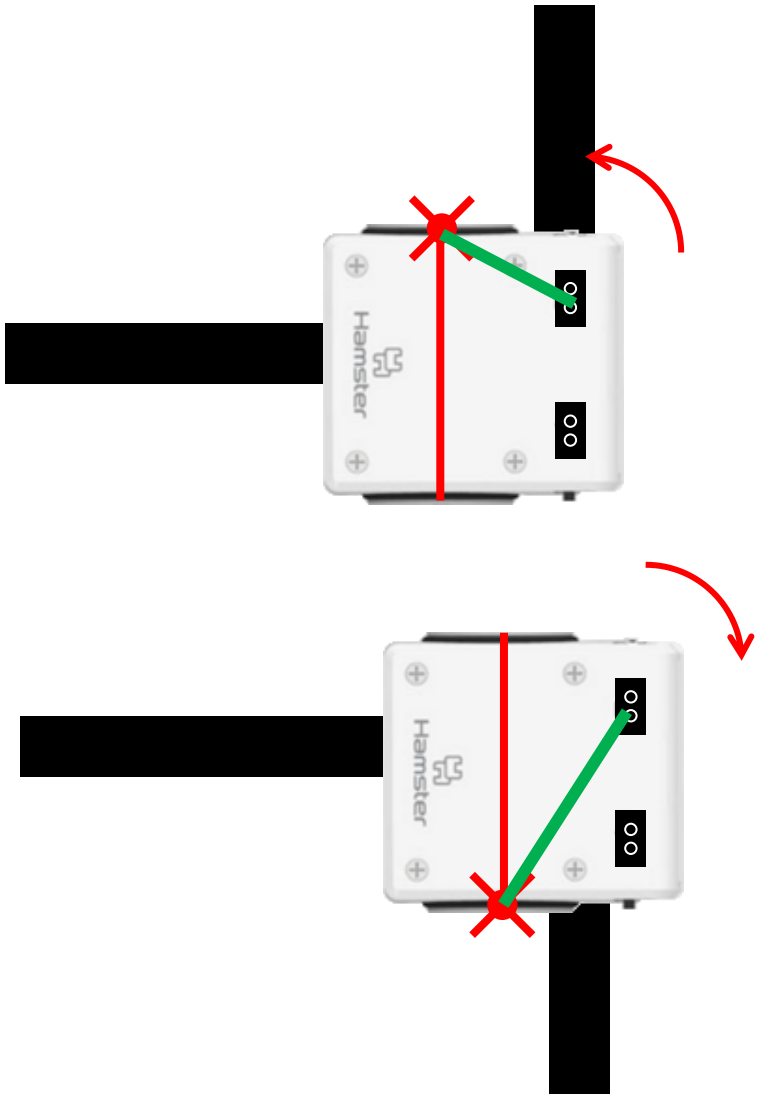


왼쪽 센서 + 왼쪽 가장자리

오른쪽 가장자리



왼쪽 가장자리



양쪽 바닥 센서

양쪽 바닥 센서


```
from roboid import *

hamster = Hamster()

while True:
    hamster.wheels(30, 30)
    if hamster.left_floor() < 50:
        hamster.wheels(-30, 30)
    elif hamster.right_floor() < 50:
        hamster.wheels(30, -30)
    wait(10) # 너무 빨리 반복하지 않도록 한다.
```

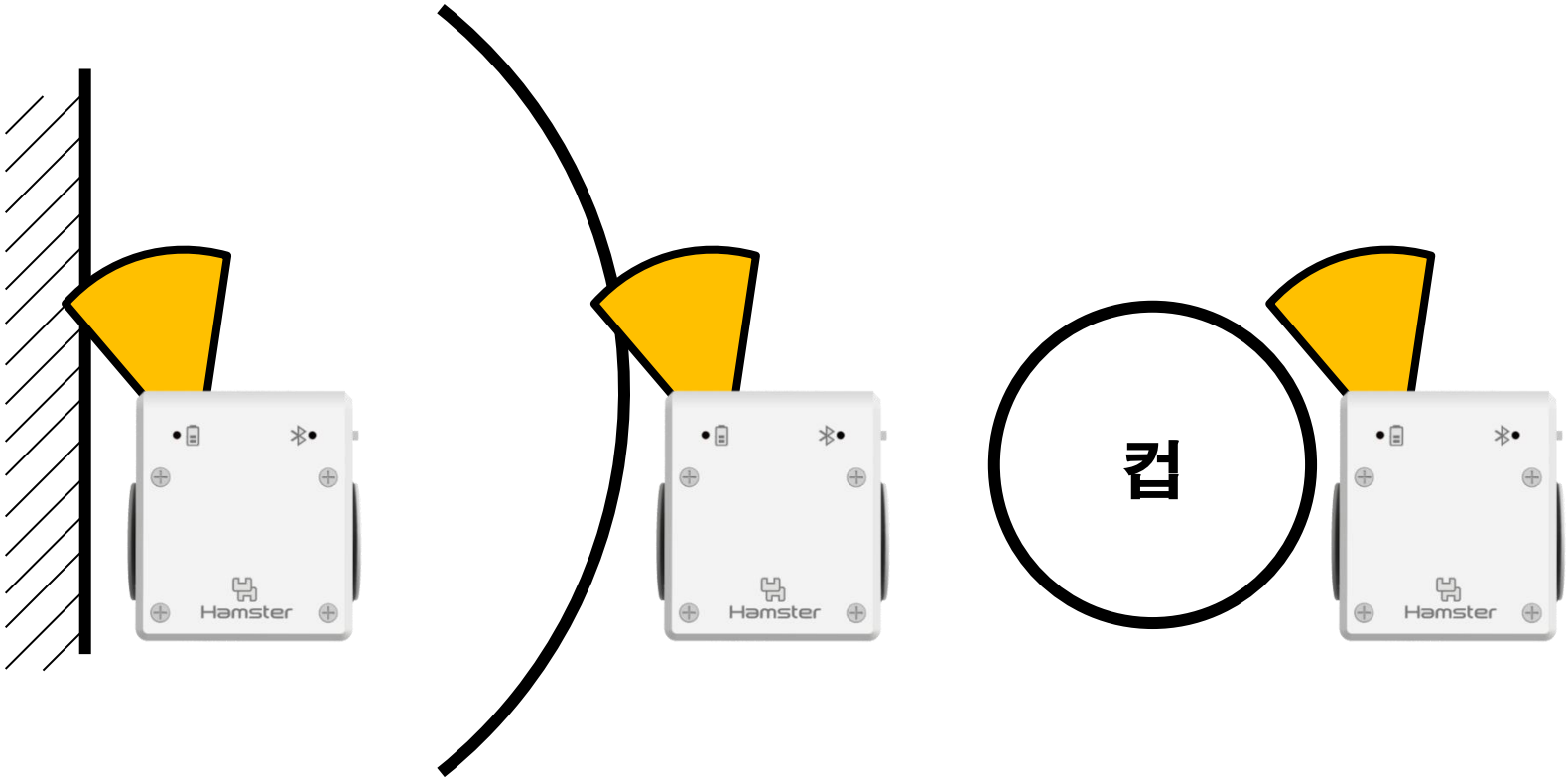
양쪽 바닥 센서

양쪽 바닥 센서

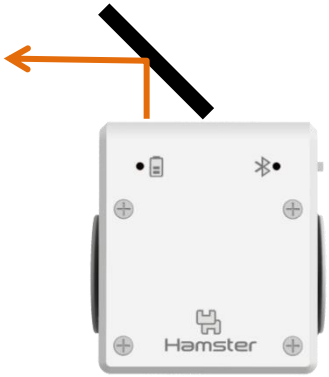

```
from roboid import *

hamster = Hamster()

while True:
    diff = hamster.left_floor() - hamster.right_floor()
    hamster.wheels(30 + diff * 0.4, 30 - diff * 0.4)
    wait(10) # 너무 빨리 반복하지 않도록 한다.
```

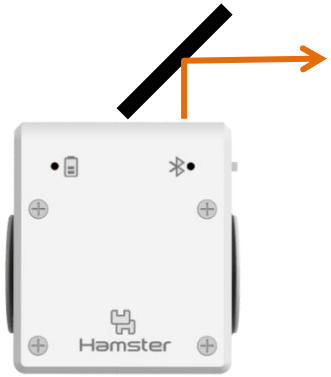
컵 따라 돌기

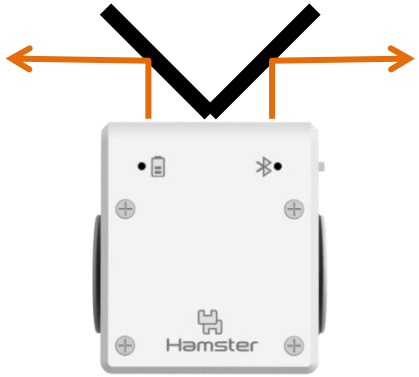
근접 센서



반사판

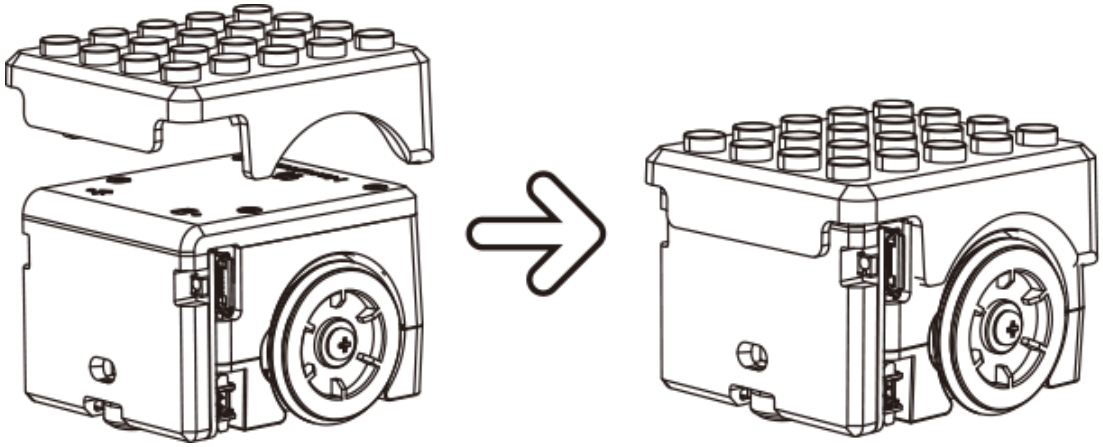






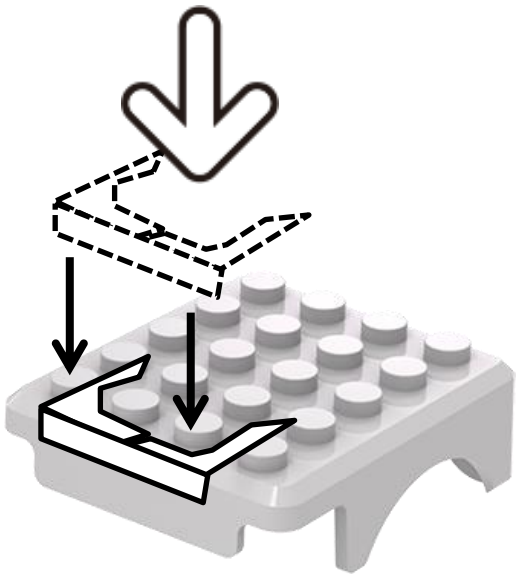
양쪽 반사판을 끼워 주세요 !!

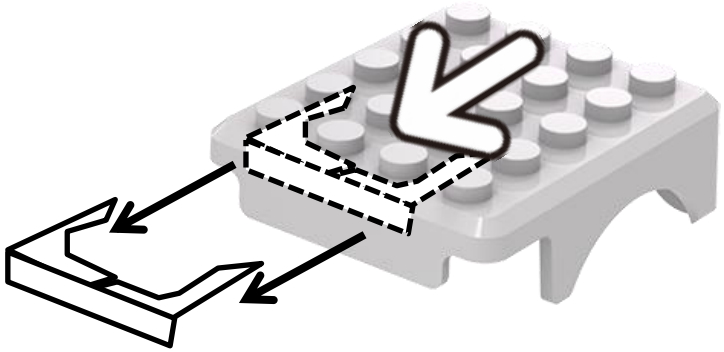
커버 조립



반사판 조립

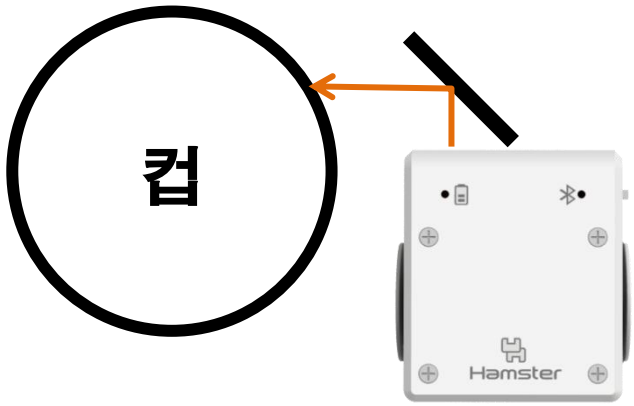
위에서 누르면 끼울 수 있어요

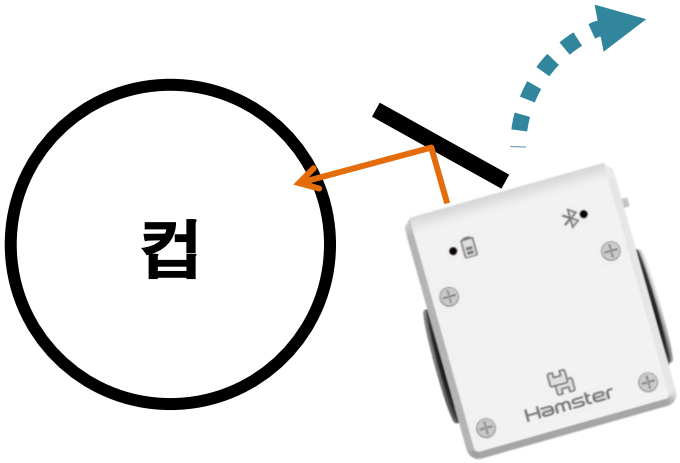


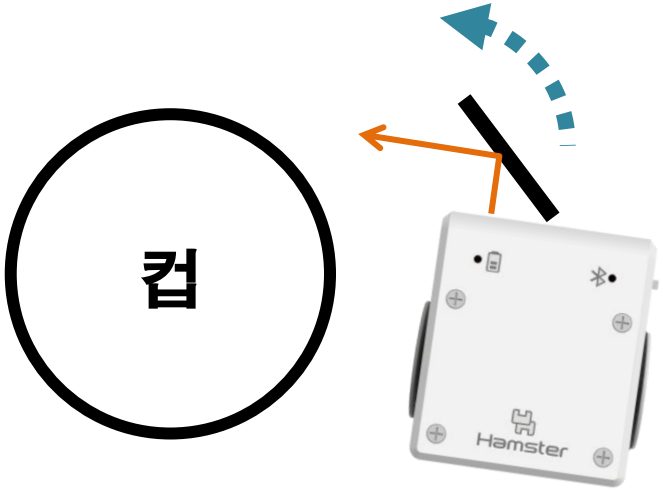


뒤에서 앞으로 밀면 뺄 수 있어요

컵 따라 돌기





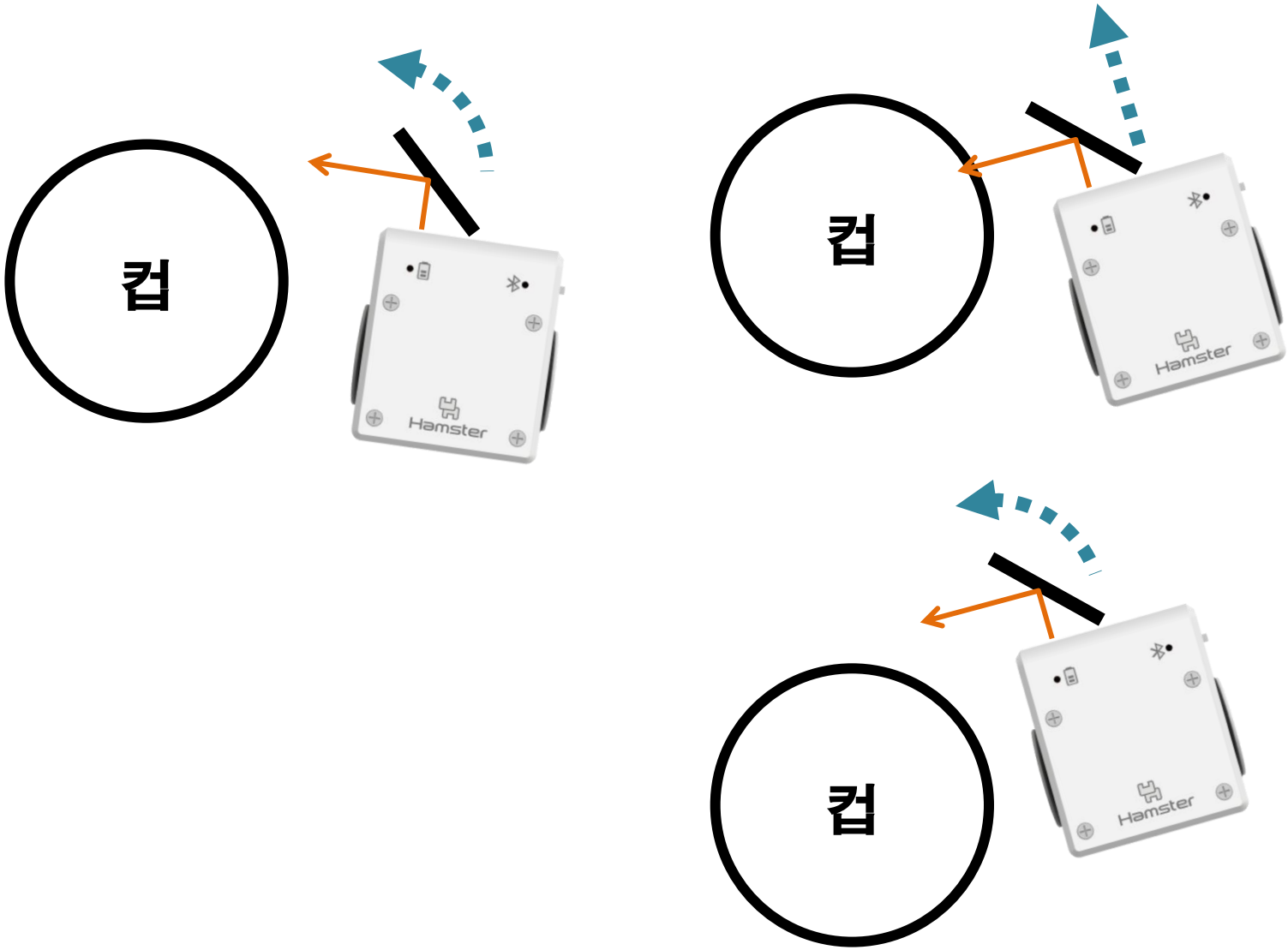


컵 따라 돌기


```
from roboid import *

hamster = Hamster()

while True:
    if hamster.left_proximity() > 10:
        hamster.wheels(30, 0)
    else:
        hamster.wheels(0, 30)
    wait(10)
```

- hamster.left_proximity()
- hamster.right_proximity()

개선된 컵 따라 돌기



개선된 컵 따라 돌기


```
from roboid import *

hamster = Hamster()

while True:
    if hamster.left_proximity() > 10:
        hamster.wheels(30, 30)
    else:
        hamster.wheels(0, 30)
    wait(10)
```

좀 더 개선된 컵 따라 돌기

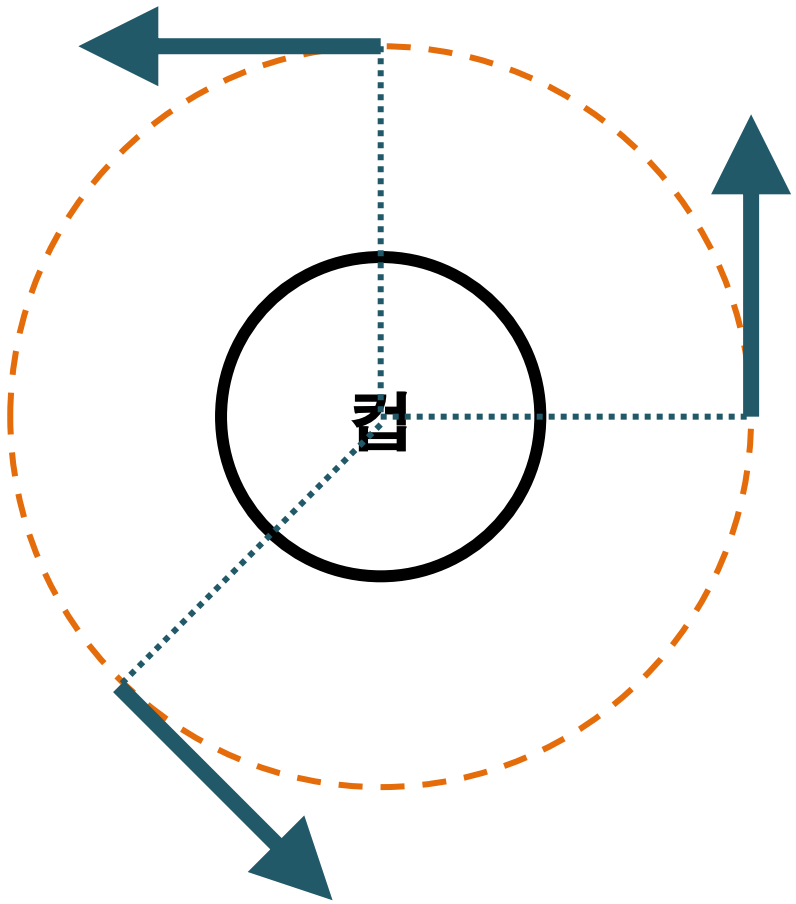

```
from roboid import *

hamster = Hamster()

while True:
    if hamster.left_proximity() > 10:
        hamster.wheels(30, 30)
    else:
        hamster.wheels(0, 30)
    wait(10)
```

근접 센서 값이 커지면
바퀴의 속도도 커진다 ?

좀 더 개선된 컵 따라 돌기

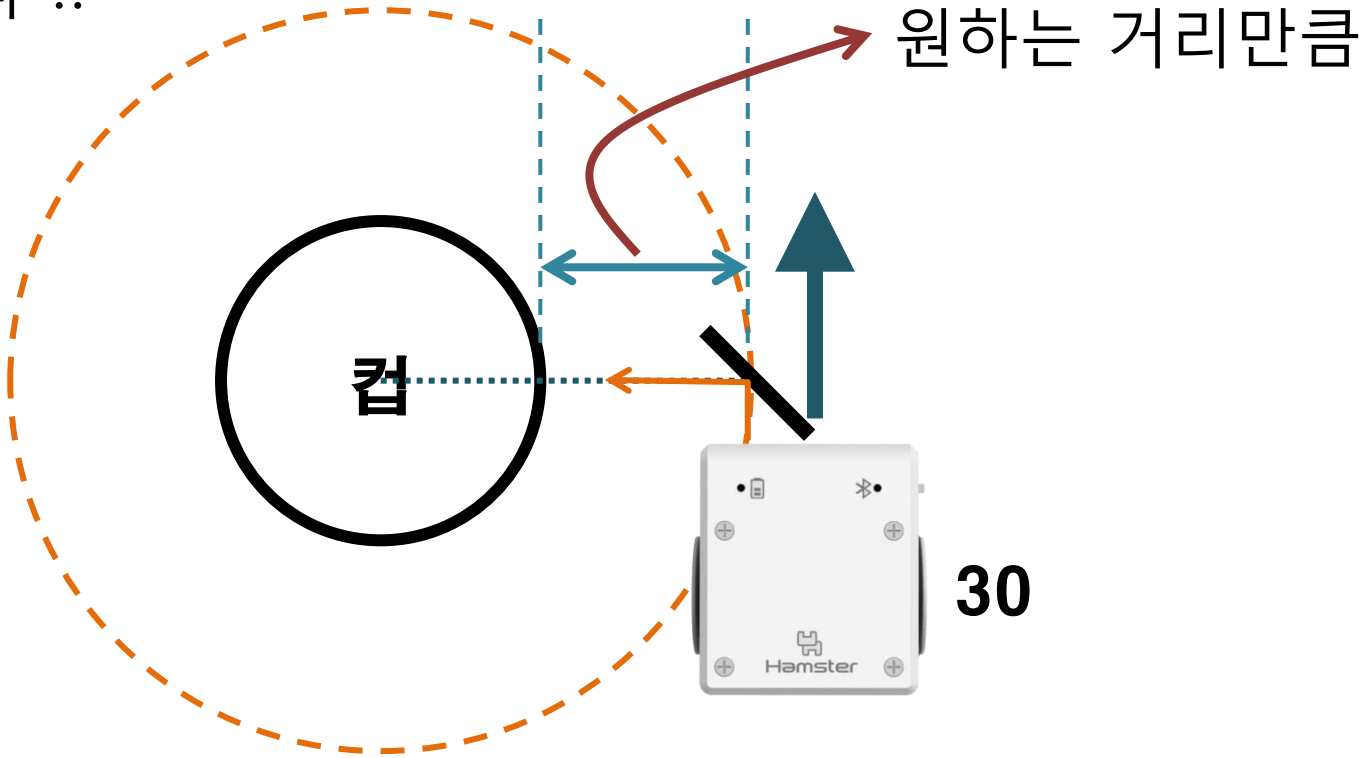

```
from roboid import *

hamster = Hamster()

while True:
    hamster.wheels(hamster.left_proximity(), 30)
    wait(10)
```

원 운동



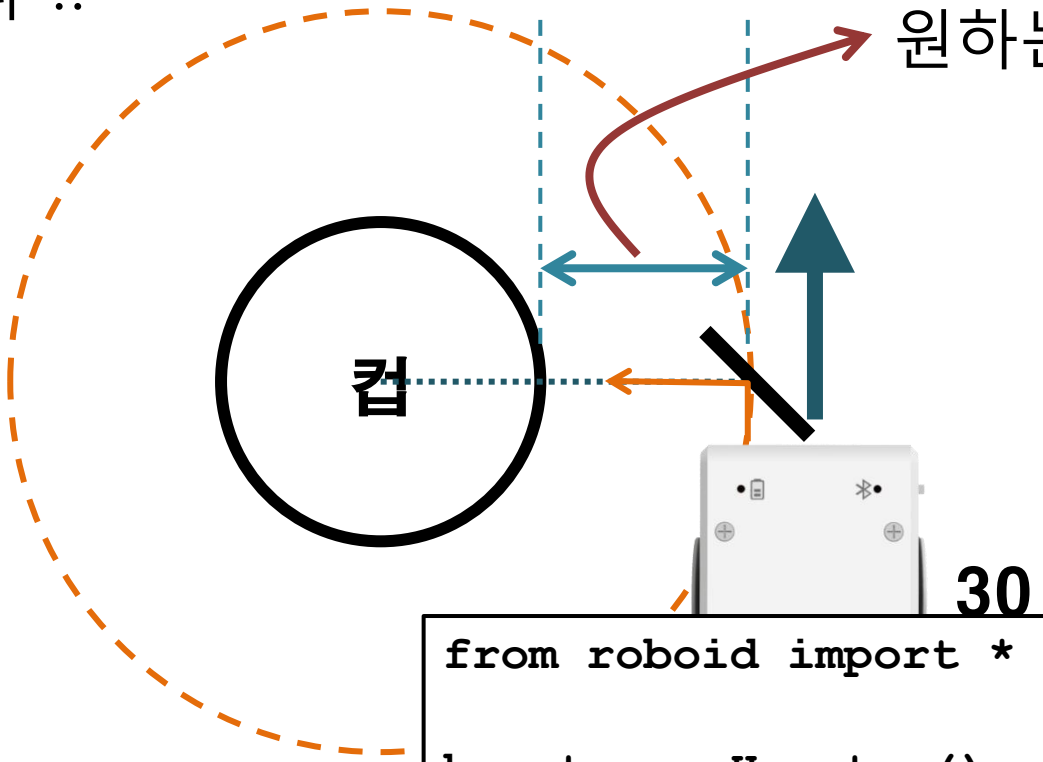
원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에
맞춘다 !!



원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에
맞춘다 !!

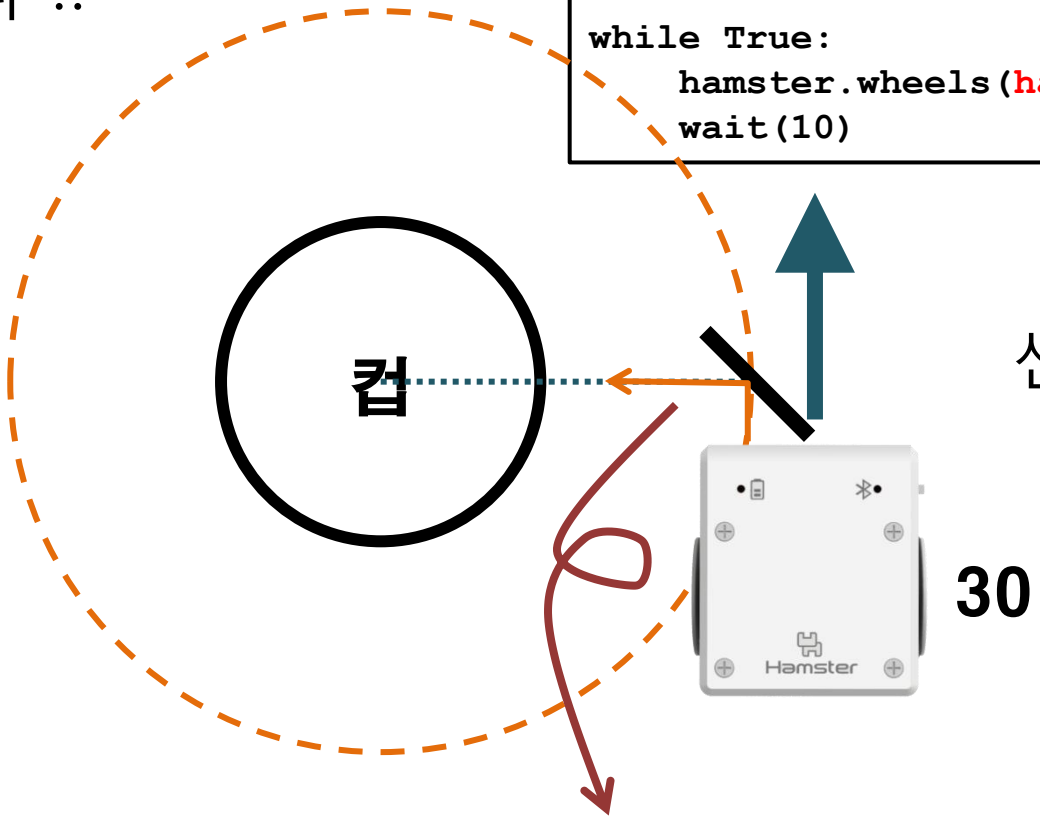



```
from roboid import *

hamster = Hamster()

while True:
    print(hamster.left_proximity())
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에 맞춘다 !!

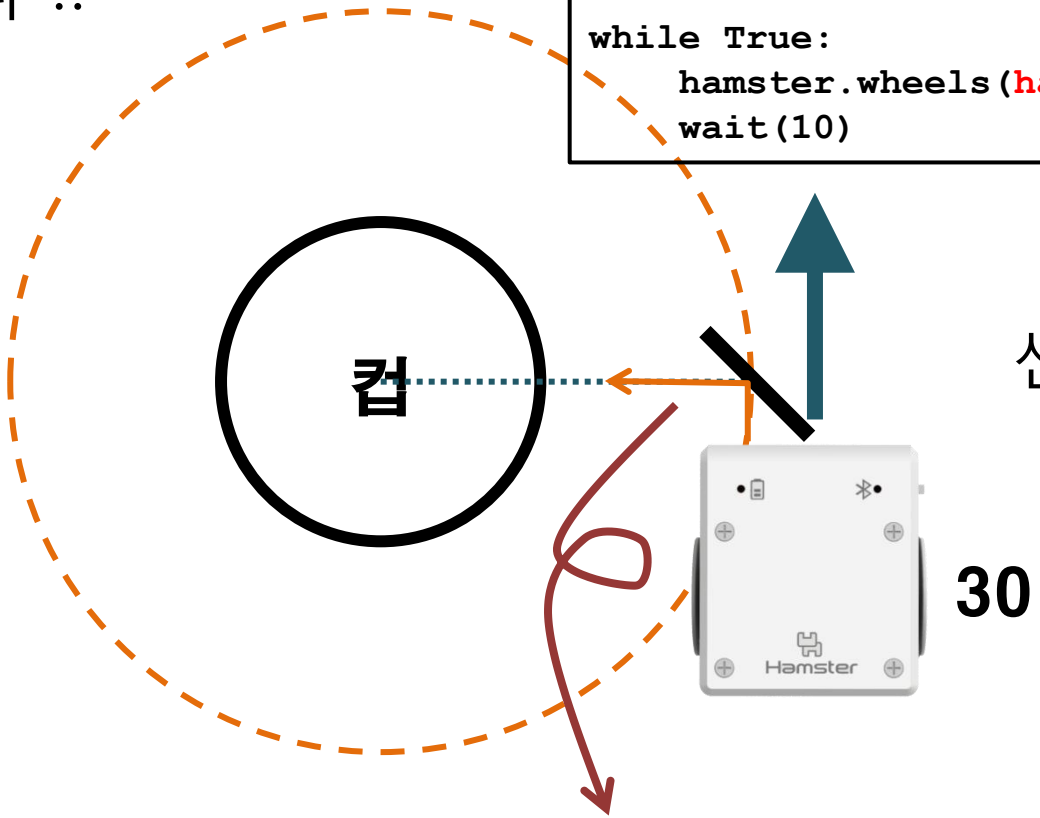
```
from roboid import *

hamster = Hamster()

while True:
    hamster.wheels(hamster.left_proximity(), 30)
    wait(10)
```



원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에 맞춘다 !!

```
from roboid import *

hamster = Hamster()

while True:
    hamster.wheels(hamster.left_proximity(), 30)
    wait(10)
```



센서 값: 16

30

센서 값: 16

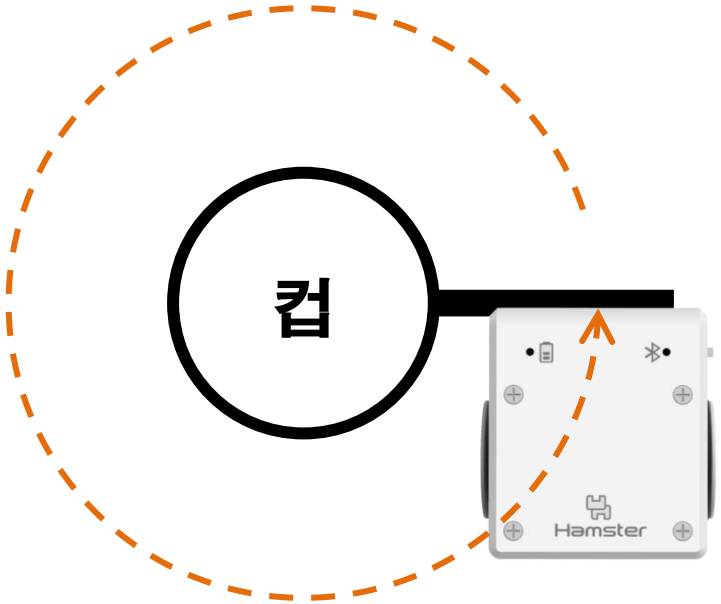
30 ÷ 16 = 1.875

원하는 거리만큼 떨어져서 돌기


```
from roboid import *

hamster = Hamster()

while True:
    hamster.wheels(hamster.left_proximity() * 1.875, 30)
    wait(10)
```

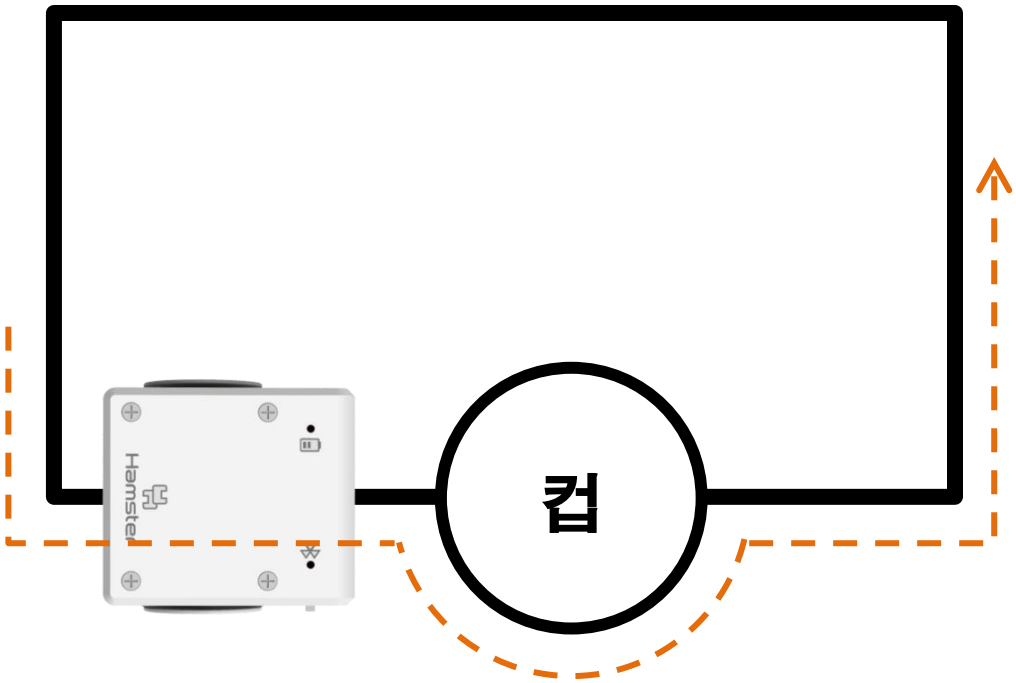
여러 가지 활동들

일정 횟수만큼 돌고 정지하기 중급



고급

라인 트레이싱 + 컵 따라 돌기



여러 가지 활동들

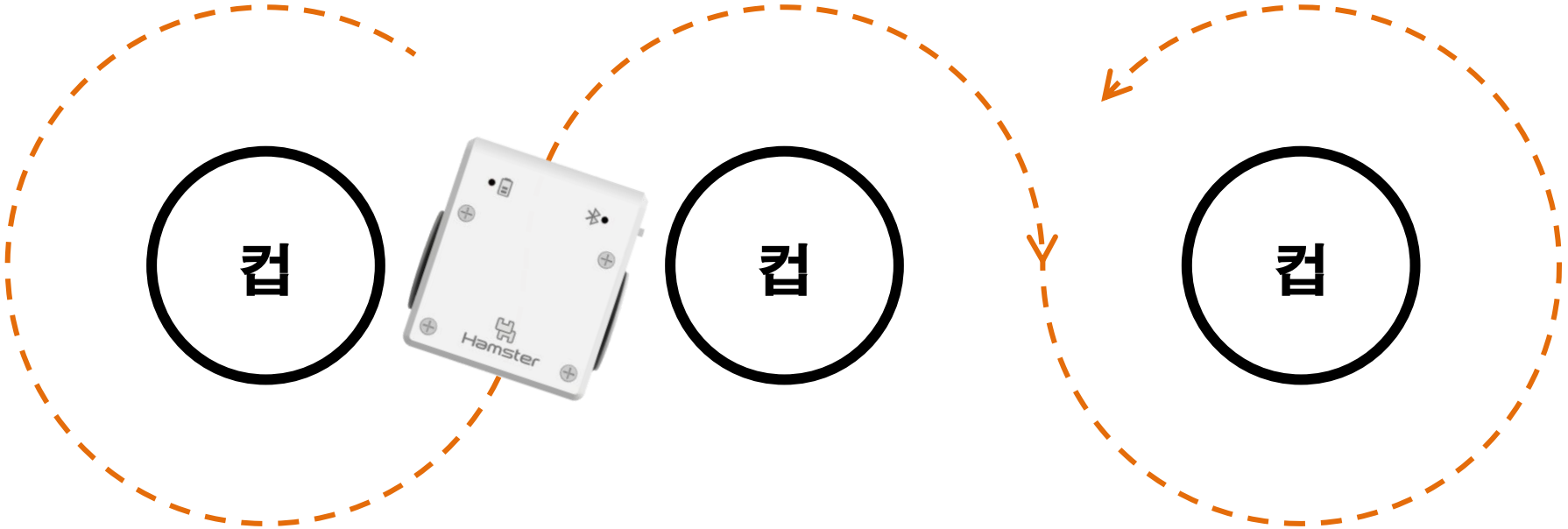
햄스터 플래닛

중급
고급

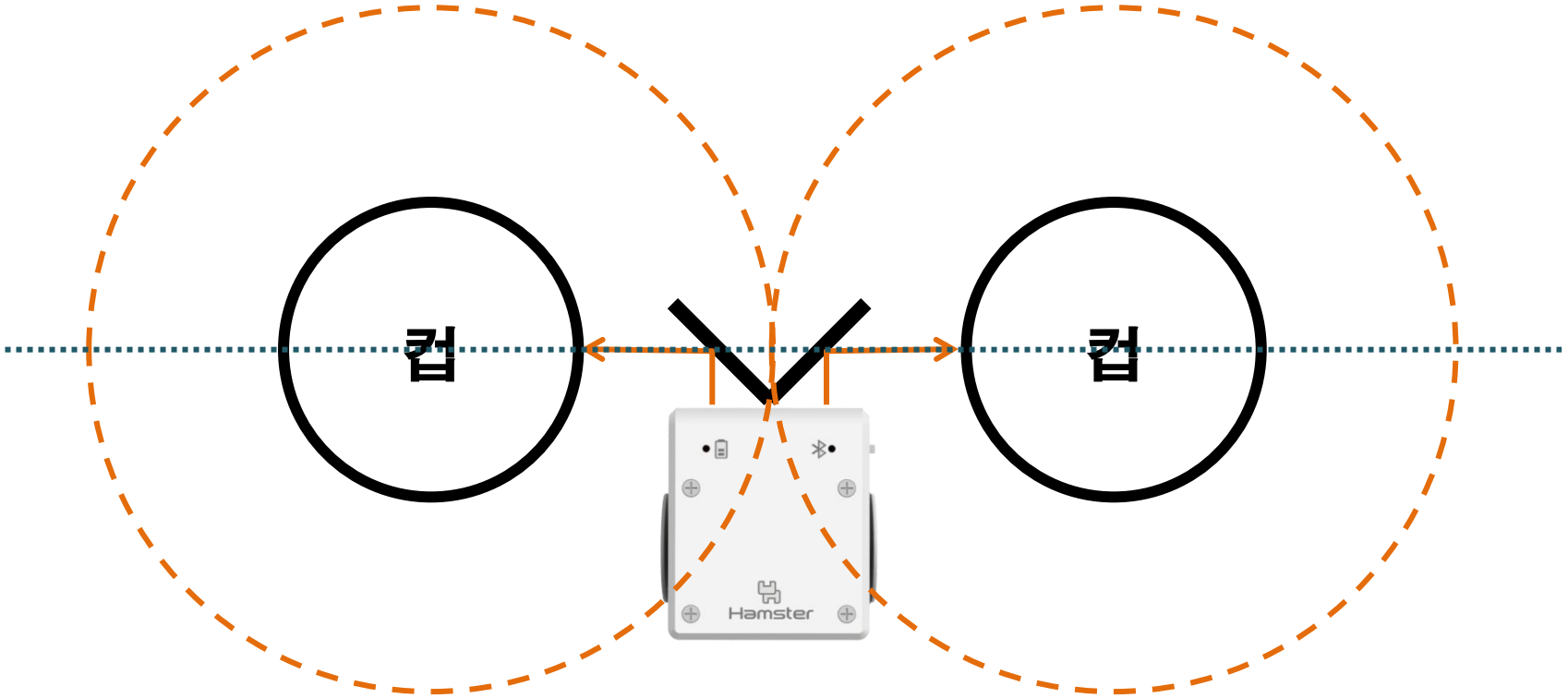
컵

중급

슬라럼 경주

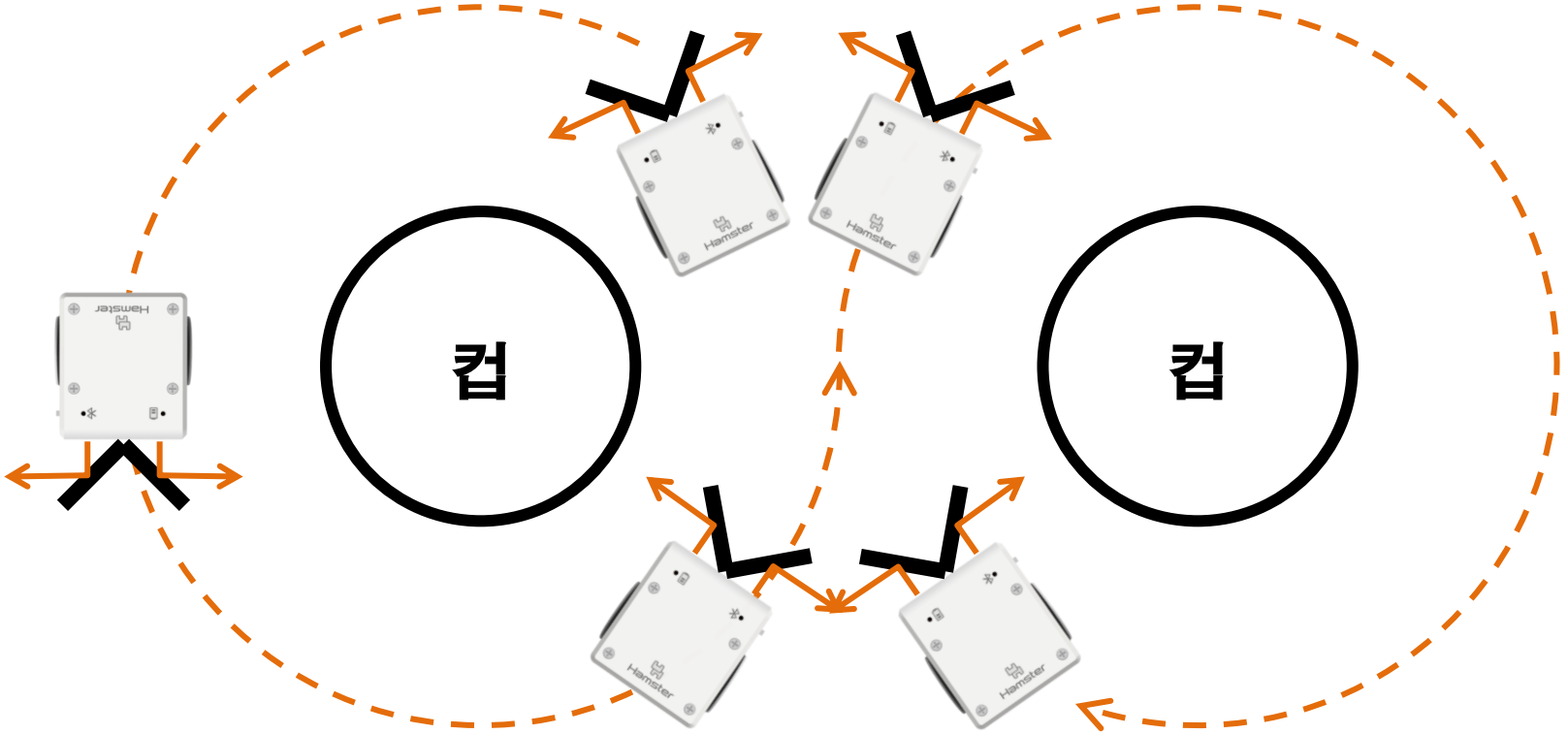


슬라럼 경주

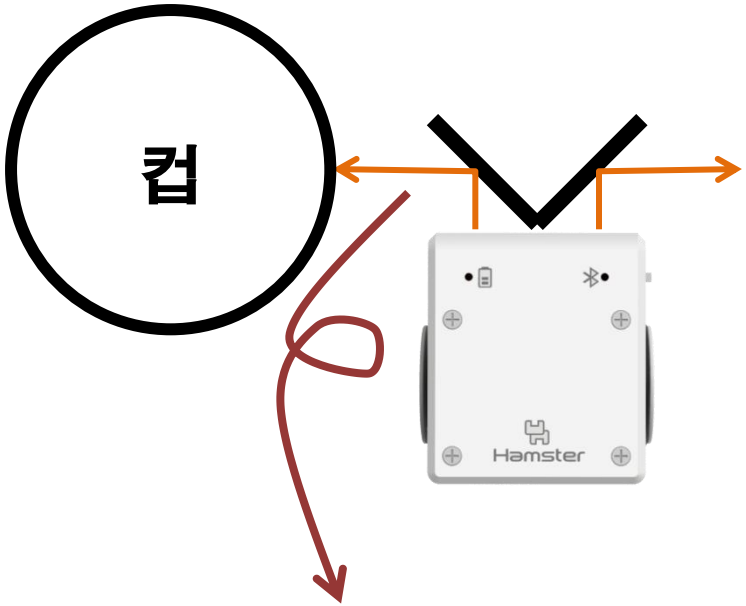


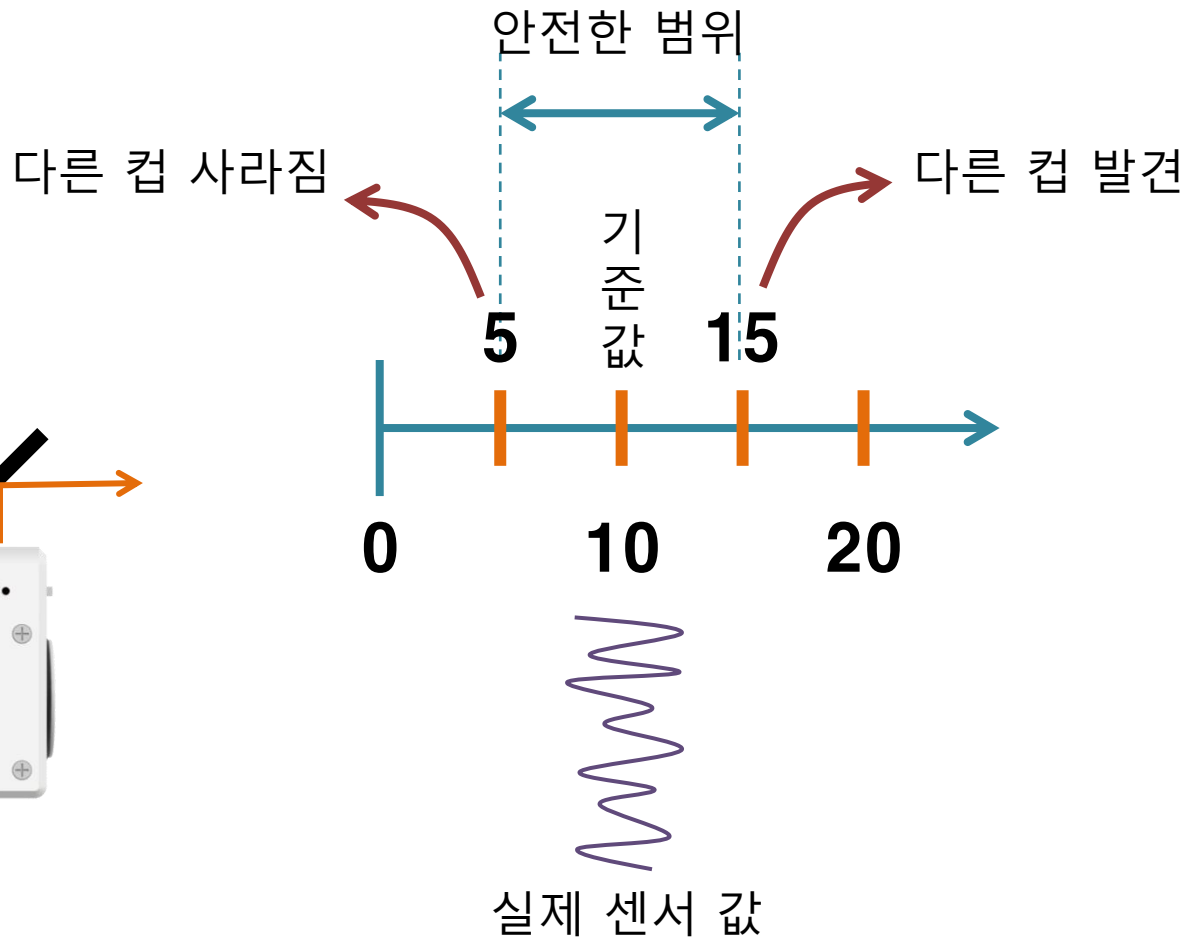
컵 중간으로 지나가게 하는 방법은?

슬라럼 경주

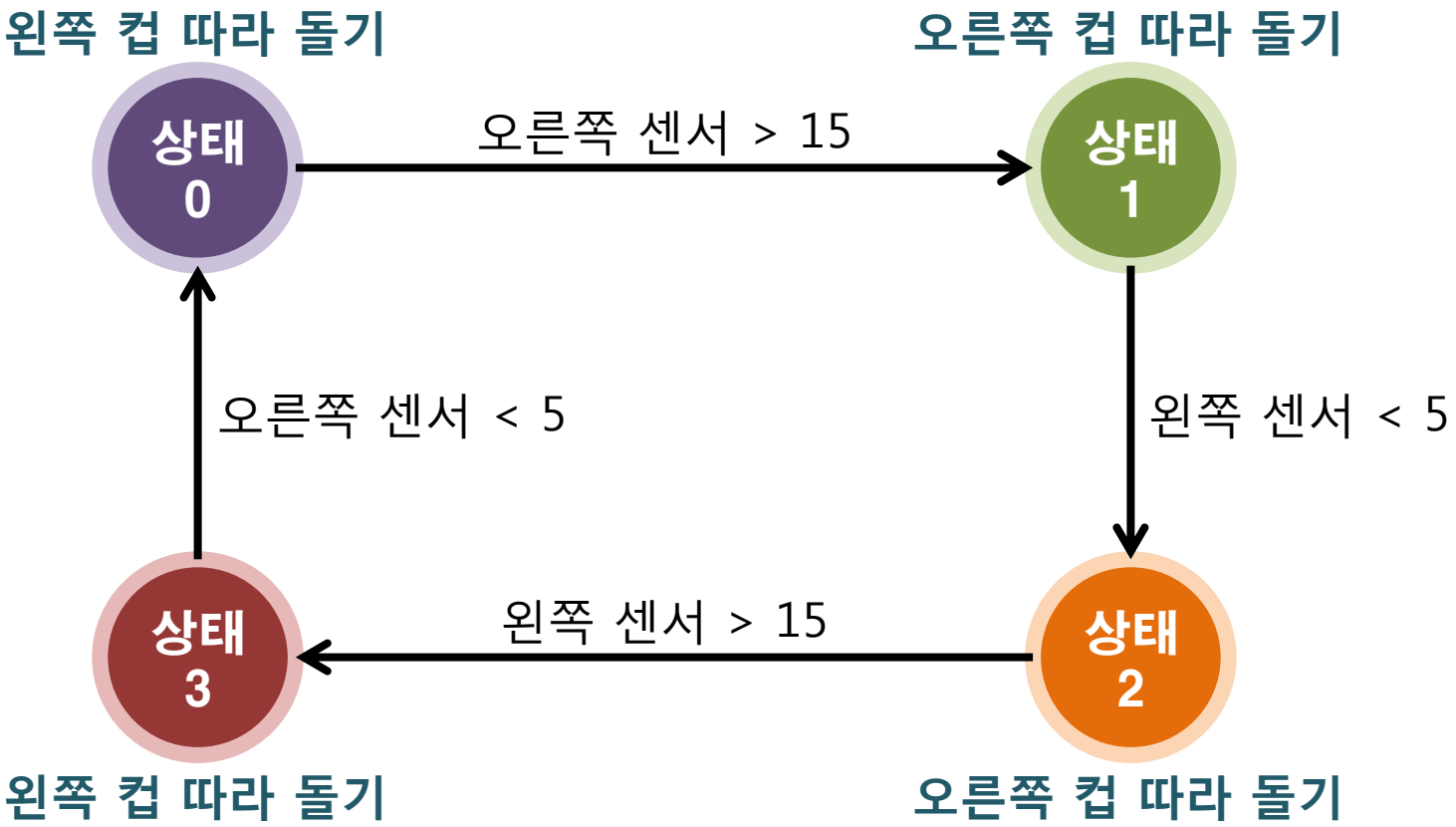

4 오른쪽 컵이 사라질 때까지
왼쪽 컵 따라 돌기
2 왼쪽 컵이 사라질 때까지
오른쪽 컵 따라 돌기
컵
컵
1 오른쪽 컵을 발견할 때까지
왼쪽 컵 따라 돌기
3 왼쪽 컵을 발견할 때까지
오른쪽 컵 따라 돌기

판단 기준





다이어그램


왼쪽 컵 따라 돌기
오른쪽 컵 따라 돌기
상태 0
오른쪽 센서 > 15
상태 1
오른쪽 센서 < 5
왼쪽 센서 < 5
상태 3
왼쪽 센서 > 15
상태 2
왼쪽 컵 따라 돌기
오른쪽 컵 따라 돌기

잘 안 되는 코드... 왜?


```
from roboid import *

hamster = Hamster()

state = 0
while True:
    left = hamster.left_proximity()
    right = hamster.right_proximity()
    if state == 0:
        hamster.wheels(left * 2.5, 50)
        if right > 15: state = 1
    elif state == 1:
        hamster.wheels(50, right * 2.5)
        if left < 5: state = 2
    elif state == 2:
        hamster.wheels(50, right * 2.5)
        if left > 15: state = 3
    else:
        hamster.wheels(left * 2.5, 50)
        if right < 5: state = 0
    wait(10)
```

순간적으로 튀는 센서 값 제거하기


```
from roboid import *

hamster = Hamster()

state = 0
count = 0
while True:
    left = hamster.left_proximity(
    right = hamster.right_proximit
    if state == 0:
        hamster.wheels(left * 2.5,
        if right > 15: count += 1
        else: count = 0
        if count > 10:
            count = 0
            state = 1
    elif state == 1:
        hamster.wheels(50, right * 2.5)
        if left < 5: count += 1
        else: count = 0
        if count > 10:
            count = 0
            state = 2
    elif state == 2:
        hamster.wheels(50, right * 2.5)
        if left > 15: count += 1
        else: count = 0
        if count > 10:
            count = 0
            state = 3
    else:
        hamster.wheels(left * 2.5, 50)
        if right < 5: count += 1
        else: count = 0
        if count > 10:
            count = 0
            state = 0
    wait(10)
```

다르게 작성해 보기


```
from roboid import *

hamster = Hamster()

def slalom(left_speed, right_speed, check):
    count = 0
    while count < 10:
        left = hamster.left_proximity()
        right = hamster.right_proximity()
        hamster.wheels(left_speed(left, right), right_speed(left, right))
        if check(left, right): count += 1
        else: count = 0
        wait(10)

while True:
    slalom(lambda l, r: l * 2.5, lambda l, r: 50, lambda l, r: r > 15)
    slalom(lambda l, r: 50, lambda l, r: r * 2.5, lambda l, r: l < 5)
    slalom(lambda l, r: 50, lambda l, r: r * 2.5, lambda l, r: l > 15)
    slalom(lambda l, r: l * 2.5, lambda l, r: 50, lambda l, r: r < 5)
```

병렬 처리 및 이벤트

병렬 처리


```
for i in range(2):
    hamster.note("C4", 0.5)
    hamster.note("E4", 0.5)
    hamster.note("G4", 0.5)
for j in range(3):
    hamster.note("A4", 0.5)
hamster.note("G4", 1)
hamster.note(0, 0.5)
```

```
for i in range(5):
    hamster.leds("RED", "RED")
    wait(500)
    hamster.leds("OFF", "OFF")
    wait(500)
```

- hamster.leds("RED", "YELLOW")
- hamster.leds("OFF", "OFF")
- hamster.leds("GREEN")
- hamster.leds("OFF")
- hamster.left_led("SKY BLUE")
- hamster.left_led("OFF")
- hamster.right_led("BLUE")
- hamster.right_led("OFF")

```
"OFF"
"RED"
"YELLOW"
"GREEN"
"SKY BLUE"
"BLUE"
"PURPLE"
"WHITE"
```

병렬 처리


```
from roboid import *

hamster = Hamster()

def play_music():
    for i in range(2):
        hamster.note("C4", 0.5)
        hamster.note("E4", 0.5)
        hamster.note("G4", 0.5)
    for j in range(3):
        hamster.note("A4", 0.5)
    hamster.note("G4", 1)
    hamster.note(0, 0.5)

def blink():
    for i in range(5):
        hamster.leds("RED", "RED")
        wait(500)
        hamster.leds("OFF", "OFF")
        wait(500)

parallel(play_music, blink)
```

이벤트 처리


```
from roboid import *

hamster = Hamster()

while True:
    if hamster.left_proximity() > 30 or hamster.right_proximity() > 30:
        hamster.wheels(-30) # 뒤로 이동한다.
        wait(1000) # 1초
        hamster.stop() # 정지한다.
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

```
rom roboid import *

hamster = Hamster()

while True:
    if hamster.left_floor() < 20 or hamster.right_floor() < 20:
        hamster.beep()
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

이벤트 처리


```
from roboid import *

hamster = Hamster()

def when_hand_found():
    return hamster.left_proximity() > 30 or hamster.right_proximity() > 30

def do_move_backward():
    hamster.wheels(-30)
    wait(1000)
    hamster.stop()

def when_on_black_line():
    return hamster.left_floor() < 20 or hamster.right_floor() < 20

def do_sound():
    hamster.beep()

when_do(when_hand_found, do_move_backward)
when_do(when_on_black_line, do_sound)

# Ctrl+C 키를 누를 때까지 계속 기다립니다.
wait(-1)
```

WHILE-DO


```
from roboid import *

hamster = Hamster()

def while_hand_found():
    return hamster.left_proximity() > 30 or hamster.right_proximity() > 30

def do_move_backward():
    hamster.wheels(-30)
    wait(1000)
    hamster.stop()

def when_on_black_line():
    return hamster.left_floor() < 20 or hamster.right_floor() < 20

def do_sound():
    hamster.beep()

while_do(while_hand_found, do_move_backward)
when_do(when_on_black_line, do_sound)

# Ctrl+C 키를 누를 때까지 계속 기다립니다.
wait(-1)
```

WAIT UNTIL


```
from roboid import *

hamster = Hamster()

def hand_found():
    return hamster.left_proximity() > 30 or hamster.right_proximity() > 30

wait_until(hand_found)
hamster.wheels(-30)
wait(1000)
hamster.stop()
```

수고하셨습니다.

http://hamster.school

akaii@kw.ac.kr