

# 햄스터로 배우는 파이선

광운대학교 로봇학부  
박광현

# 파이선 설치

## python.org

The image shows a browser window at python.org. The navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The main header features the Python logo, a search bar, and a 'Donate' button. A secondary navigation bar contains links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The 'Downloads' menu is open, listing options like All releases, Source code, Windows, Mac OS X, Other Platforms, License, and Alternative Implementations. The 'Windows' option is selected, displaying a 'Download for Windows' section with a 'Python 3.8.0' button. An orange arrow labeled '1' points to the 'Downloads' link, and another orange arrow labeled '2' points to the 'Python 3.8.0' button. Below the navigation bar, there is a code snippet and a promotional message: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

```
# Python 3: Lists
>>> fruits = ['apple', 'banana', 'orange']
>>> loud_fruits = [fruit.upper() for fruit in fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'ORANGE']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'apple'), (2, 'orange')]
```

**Download for Windows**

Python 3.8.0

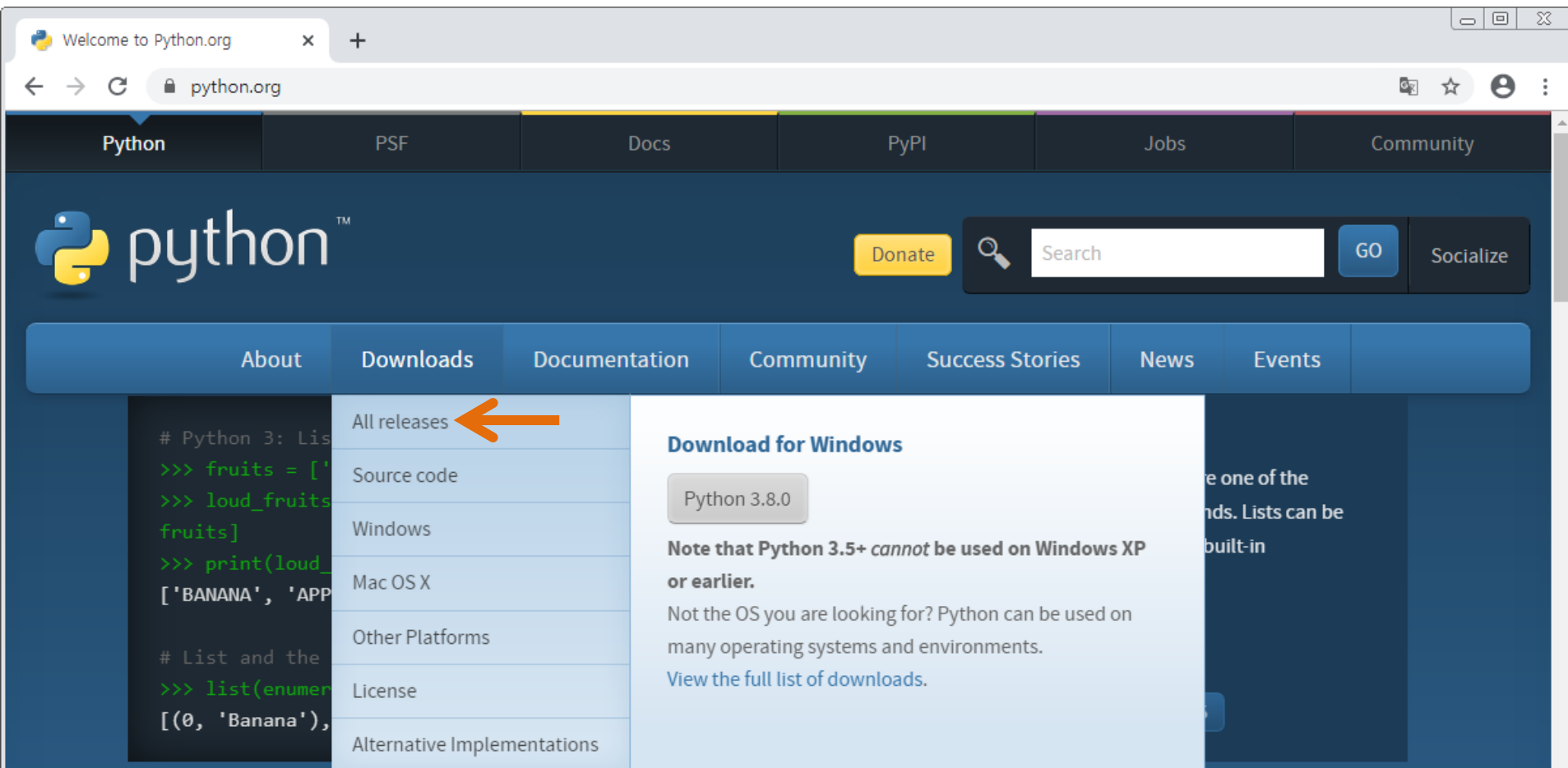
**Note that Python 3.5+ cannot be used on Windows XP or earlier.**

Not the OS you are looking for? Python can be used on many operating systems and environments. [View the full list of downloads.](#)

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

<https://www.python.org/downloads/>

OS에 맞는 파이선이 자동으로 선택되지 않는 경우



Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)

아래로 스크롤 → Python 3.8.0의 Download 클릭

The screenshot shows the Python.org download page. At the top, there is a navigation bar with the Python logo and the text "Download Python | Python.org". Below this is a search bar with the text "Looking for Python 2.7? See below for specific releases". A large yellow banner contains the text "Join the official 2019 Python Developers Survey" and a button labeled "Start the survey!". Below the banner, the text "Looking for a specific release?" is followed by "Python releases by version number:". A table lists various Python releases with columns for "Release version", "Release date", and "Click for more". The row for "Python 3.8.0" is highlighted in orange, and an orange arrow points to the "Download" link in that row. Below the table, there is a link that says "View older releases".

Release version	Release date	Click for more
<a href="#">Python 3.5.9</a>	Nov. 2, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>
<a href="#">Python 3.5.8</a>	Oct. 29, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>
<a href="#">Python 2.7.17</a>	Oct. 19, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>
<a href="#">Python 3.7.5</a>	Oct. 15, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>
<b><a href="#">Python 3.8.0</a></b>	Oct. 14, 2019	<b><a href="#">Download</a></b> <a href="#">Release Notes</a>
<a href="#">Python 3.7.4</a>	July 8, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>
<a href="#">Python 3.6.9</a>	July 2, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>
<a href="#">Python 3.7.3</a>	March 25, 2019	<a href="#">Download</a> <a href="#">Release Notes</a>

[View older releases](#)

아래로 스크롤 → OS에 맞는 설치 파일 내려 받기

The screenshot shows the Python 3.8.0 download page. The browser address bar shows the URL: [python.org/downloads/release/python-380/](https://python.org/downloads/release/python-380/). The page title is "Files". Below the title is a table with the following columns: Version, Operating System, Description, MD5 Sum, File Size, and GPG. The table lists various download options for Python 3.8.0. Two orange arrows point to the "Windows x86-64 executable installer" and "Windows x86 executable installer" rows.

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

# 햄스터 라이브러리 설치

# 라이브러리 설치

8

## hamster.school

Hamster School

주의 요함 | hamster.school/ko/

햄스터 스쿨

한국어

이제 작고 앙증맞은 햄스터 로봇과 함께  
책상 위에서 로봇을 프로그래밍하세요!

새 소식

강의 신청

구입 방법

페이스북 그룹

교육 자료

문제 해결

다운로드

레퍼런스



## 바로가기에서 파이썬 클릭

The screenshot shows a web browser window with the URL `hamster.school/ko/download/`. The page title is '햄스터 스쿨' (Hamster School). The navigation menu includes '새 소식', '다운로드', '교육자료', '강의 신청', '문제해결', '레퍼런스', and '구입방법'. The '다운로드' (Downloads) section is highlighted in orange and contains a sub-section '소프트웨어 다운로드' (Software Downloads). Under this section, there are two main categories: '바로가기' (Quick Links) and '그래픽 언어' (Graphic Languages). The '바로가기' category lists '로봇 코딩 소프트웨어 스택' (Robot Coding Software Stack) and '로보이드 스튜디오' (Robo-Id Studio). The '그래픽 언어' category lists '스크립트 언어' (Scripting Languages), '프로세싱' (Processing), '파이썬' (Python), and '자바스크립트' (JavaScript). A red arrow points to '파이썬' (Python). The '스크립트 언어' category lists '프로세싱' (Processing), '파이썬' (Python), and '자바스크립트' (JavaScript). The '고급 언어' (Advanced Languages) category lists 'C 라이브러리' (C Libraries) and 'C++ 라이브러리' (C++ Libraries). The '파이썬' (Python) link is highlighted with a red arrow. The main content area shows a section for '로봇 코딩 소프트웨어: 스크래치 + 엔트리 + 플레이봇 + 자바스크립트' (Robot Coding Software: Scratch + Entri + PlayBot + JavaScript) with the current version '1.7.2' and a release date of '2019.06.06'. Below this, there are three bullet points: '로봇 코딩 SW를 설치하면 스크래치와 엔트리, 플레이봇, 자바스크립트를 사용할 수 있습니다.' (After installing the robot coding SW, you can use Scratch, Entri, PlayBot, and JavaScript.), 'USB 동글의 디바이스 드라이버는 설치 파일에 포함되어 있으며, 설치 과정에서 디바이스 드라이버도 같이 설치됩니다.' (The USB dongle's device driver is included in the installation file, and the device driver is also installed during the installation process.), and 'USB 동글을 PC에 연결하기 전에 로봇 코딩 소프트웨어를 먼저 설치해야 합니다.' (Before connecting the USB dongle to the PC, you must first install the robot coding software.). Below the bullet points, there is a section '내려 받기' (Download) with text: '윈도우/OSX용 로봇 코딩 SW에는 스크래치 3 오프라인 에디터(버전 3.3.0)가 포함되어 있습니다. 엔트리 오프라인에서 햄스터S를 사용하려면 엔트리 오프라인 포함 버전을 설치해야 합니다. 엔트리 온라인에서 햄스터S를 사용하려면 엔트리 오프라인 포함 버전과 미포함 버전 어느 것을 설치해도 됩니다.' (The robot coding SW for Windows/OSX includes Scratch 3 offline editor (version 3.3.0). To use HamsterS in the Entri offline version, you must install the version with Entri offline. To use HamsterS in the Entri online version, you can install either the version with Entri offline or the version without Entri offline.)

## 파이썬 라이브러리 내려 받기

The screenshot shows a web browser window with the URL `hamster.school/ko/download/`. The page title is "소프트웨어 다운로드" (Software Download). The main content area is titled "파이썬" (Python) and includes the text "지원하는 하드웨어: 햄스터, 햄스터S, 거북이" (Supported hardware: Hamster, HamsterS, Turtle). The current version is listed as "현재 버전: 1.5.2 (공개 날짜: 2019.11.28)". A list of links for downloading Python versions is provided:

- 파이썬 2.7.x 버전용 라이브러리 및 예제 파일 [zip](#) (442 KB)
- 파이썬 3.5.x 버전용 라이브러리 및 예제 파일 [zip](#) (456 KB)
- 파이썬 3.6.x 버전용 라이브러리 및 예제 파일 [zip](#) (443 KB)
- 파이썬 3.7.x 버전용 라이브러리 및 예제 파일 [zip](#) (443 KB)
- 파이썬 3.8.x 버전용 라이브러리 및 예제 파일 [zip](#) (444 KB)

On the left side, there is a navigation menu with categories: "바로가기" (Quick Links), "그래픽 언어" (Graphic Languages) including "로봇 코딩 소프트웨어 스택" and "로보이드 스튜디오", "스크립트 언어" (Script Languages) including "프로세싱", "파이썬", and "자바스크립트", and "고급 언어" (Advanced Languages) including "C 라이브러리" and "C++ 라이브러리".

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

13

hamster.school

Hamster School

주의 요함 | hamster.school/ko/

햄스터 스쿨

한국어

이제 작고 양증맞은 햄스터 로봇과 함께  
책상 위에서 로봇을 프로그래밍하세요!

새 소식

강의 신청

구입 방법

페이스북 그룹

교육 자료

문제 해결

다운로드

레퍼런스

## 바로가기에서 디바이스 드라이버 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



#### 그래픽 언어



로봇 코딩 소프트웨어: 스크래치 + 엔트리 + 플레이봇 + 자바스크립트

현재 버전: 1.7.2 (공개 날짜: 2019.06.06)

- 로봇 코딩 SW를 설치하면 스크래치와 엔트리, 플레이봇, 자바스크립트를 사용할 수 있습니다.
- USB 동글의 디바이스 드라이버는 설치 파일에 포함되어 있으며, 설치 과정에서 디바이스 드라이버도 같이 설치됩니다.
- **USB 동글을 PC에 연결하기 전에 로봇 코딩 소프트웨어를 먼저 설치해야 합니다.**

#### 내려 받기

윈도우/OSX용 로봇 코딩 SW에는 스크래치 3 오프라인 에디터(버전 3.3.0)가 포함되어 있습니다.

엔트리 오프라인에서 햄스터S를 사용하려면 엔트리 오프라인 포함 버전을 설치해야 합니다.

엔트리 온라인에서 햄스터S를 사용하려면 엔트리 오프라인 포함 버전과 미포함 버전 어느 것을 설치해도 됩니다.

#### 엔트리 오프라인 포함 버전

지원하는 하드웨어: 햄스터, 햄스터S, 거북이

- **윈도우 32비트용 설치 파일** (475.2 MB) 윈도우 7 이상
- **윈도우 64비트용 설치 파일** (489.2 MB) 윈도우 7 이상  
윈도우용 로봇 코딩 소프트웨어는 C:\₩RobotCoding 폴더에 설치됩니다.
- **OSX 64비트용 설치 파일** (478.6 MB)

## 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

되어 있어야 합니다. [고급 클라이언트에서 로보이드 구성 설치하기](#)

#### 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.

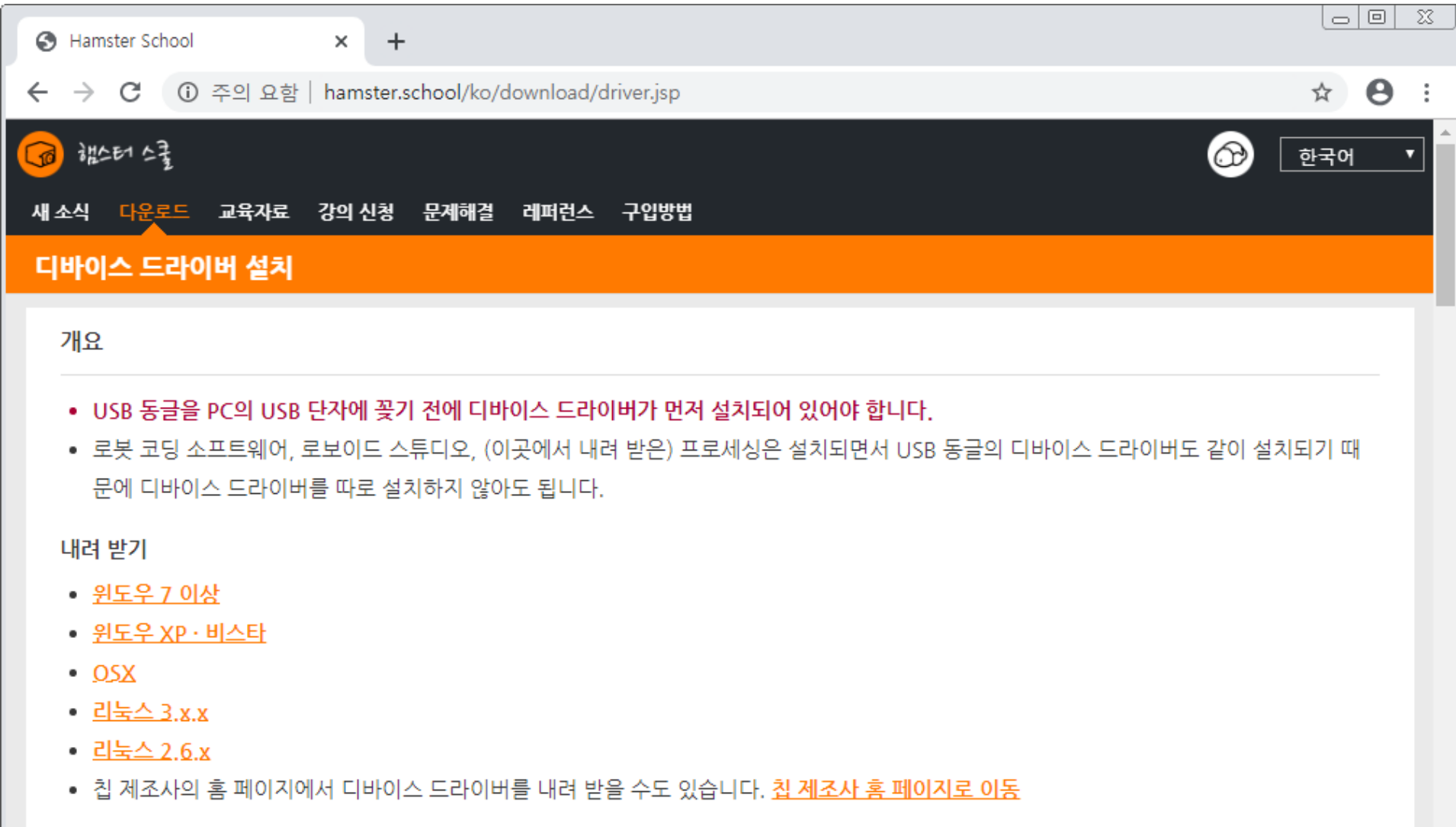
[디바이스 드라이버 내려 받기 및 설치 방법](#)



Copyright 로봇SW교육원 All rights reserved.

어려운 일이 있으면 광운대학교 로봇학부 박광현 교수(akaii@kw.ac.kr)에게 연락하세요.

## OS에 맞는 디바이스 드라이버 내려 받아 설치



Hamster School

주의 요함 | hamster.school/ko/download/driver.jsp

햄스터 스쿨

한국어

새 소식 다운로드 교육자료 강의 신청 문제해결 레퍼런스 구입방법

### 디바이스 드라이버 설치

#### 개요

- USB 동글을 PC의 USB 단자에 꽂기 전에 디바이스 드라이버가 먼저 설치되어 있어야 합니다.
- 로봇 코딩 소프트웨어, 로보이드 스튜디오, (이곳에서 내려 받은) 프로세싱은 설치되면서 USB 동글의 디바이스 드라이버도 같이 설치되기 때문에 디바이스 드라이버를 따로 설치하지 않아도 됩니다.

#### 내려 받기

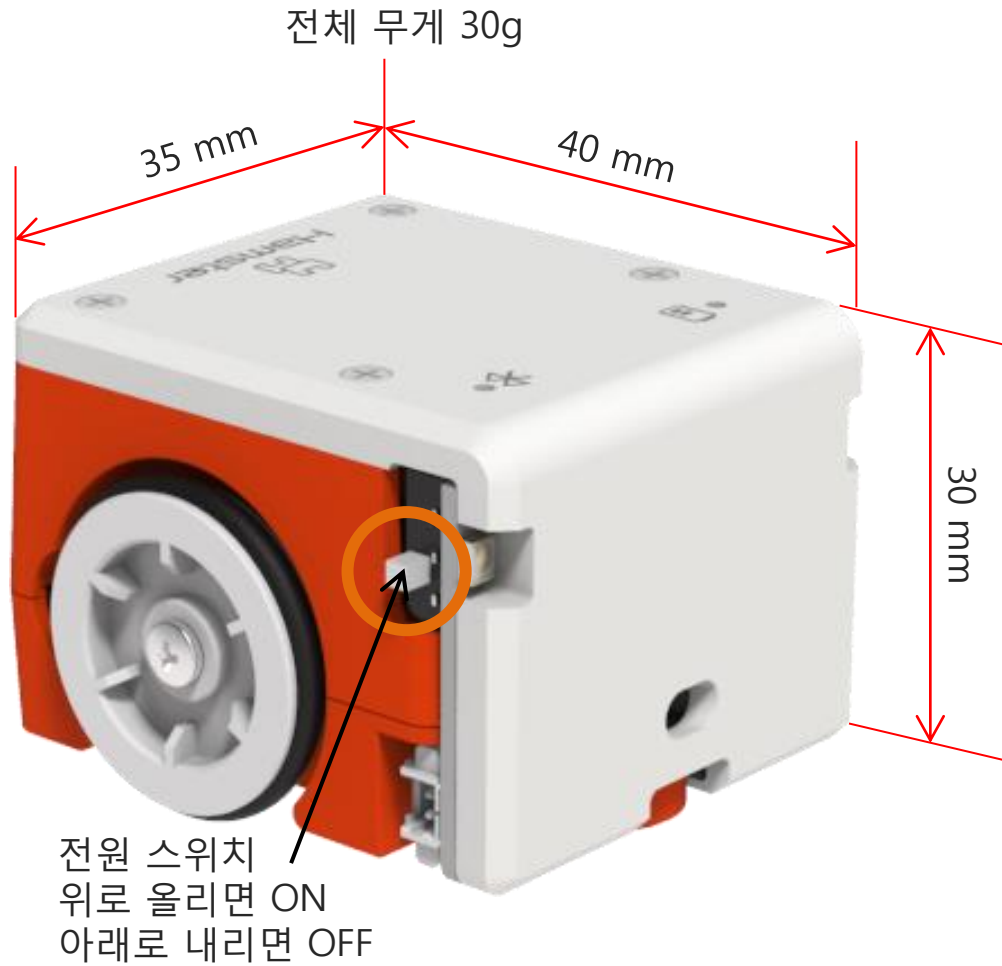
- [윈도우 7 이상](#)
- [윈도우 XP · 비스타](#)
- [OSX](#)
- [리눅스 3.x.x](#)
- [리눅스 2.6.x](#)
- 칩 제조사의 홈 페이지에서 디바이스 드라이버를 내려 받을 수도 있습니다. [칩 제조사 홈 페이지로 이동](#)

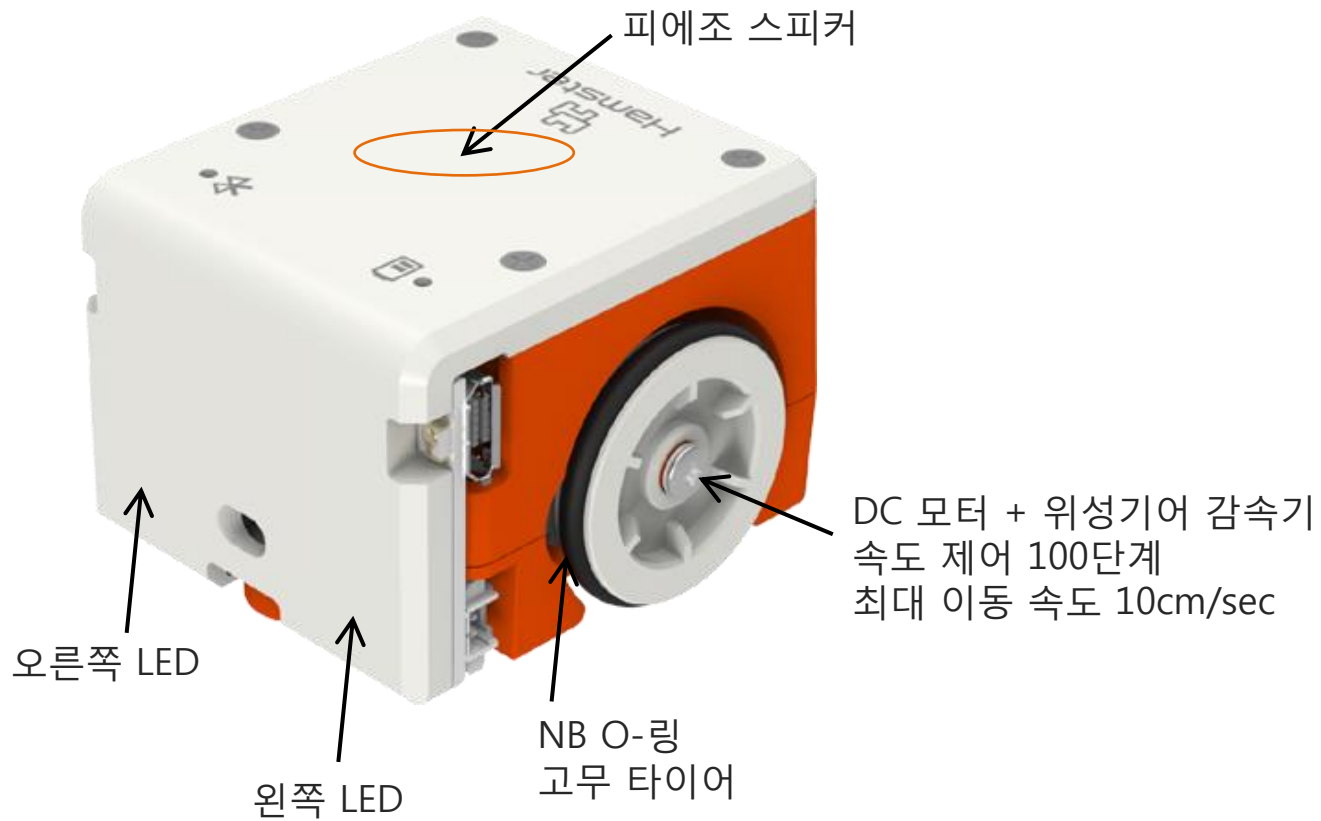


# 햄스터 로봇 소개

# 크기 · 무게 · 전원 스위치

18





# 입력 장치

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

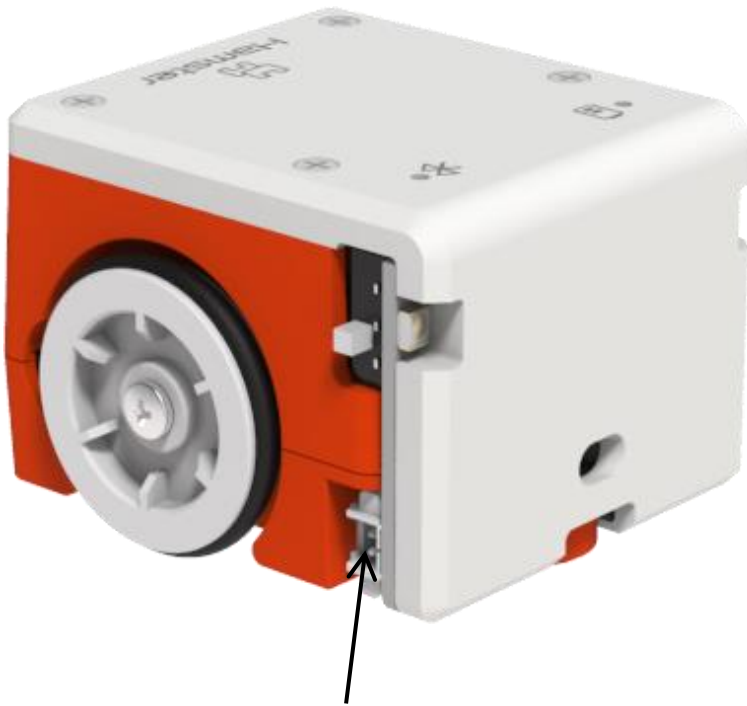


왼쪽 바닥 센서  
(적외선 센서)  
0~255단계 감지





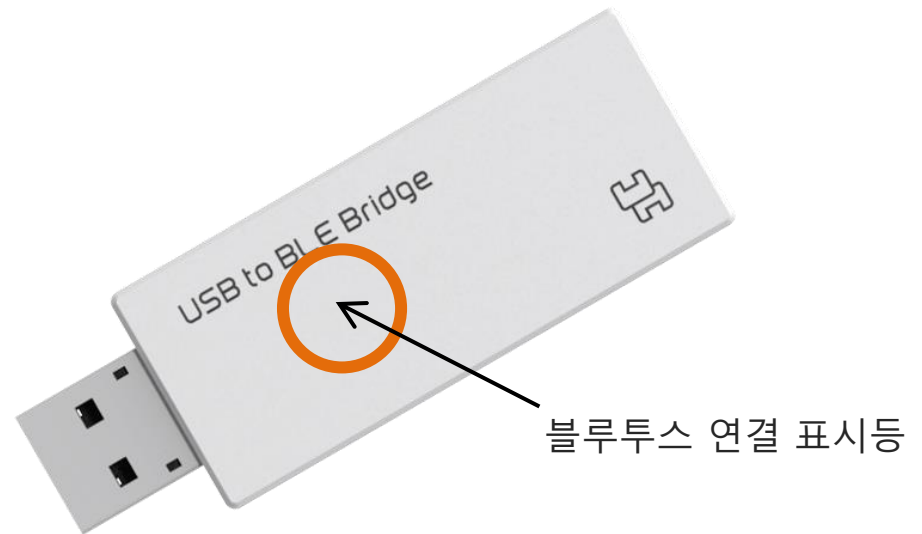
내장 리튬 배터리 3.7V, 120mA  
충전 약 30분  
연속 동작 평균 1시간  
대기 최대 12시간

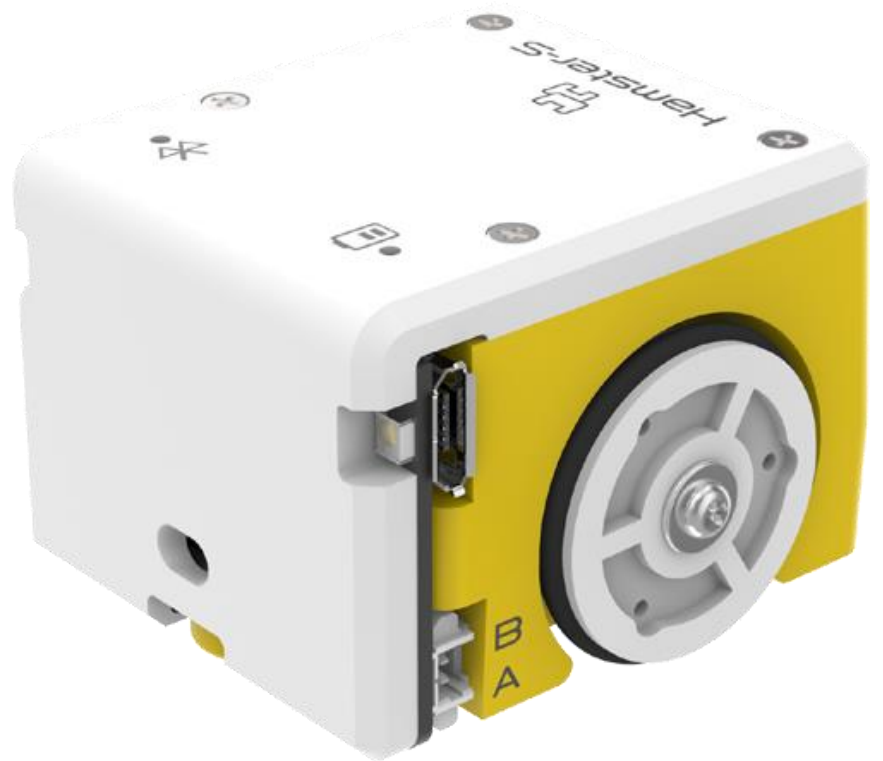


보조 전원 단자  
3.7V 리튬 폴리머 전지



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

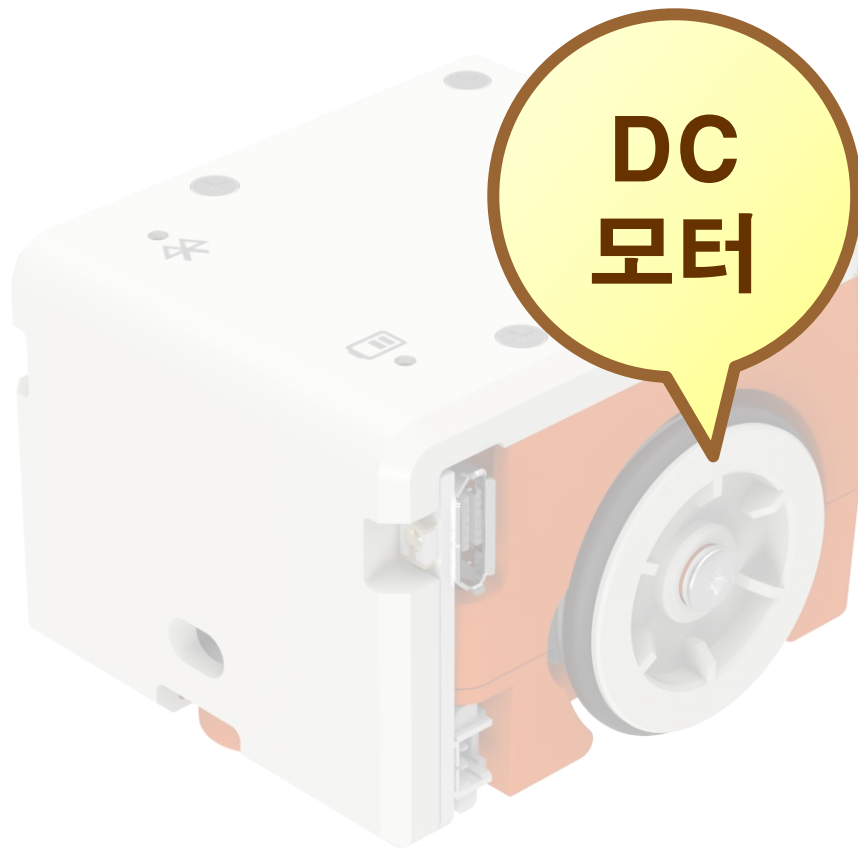




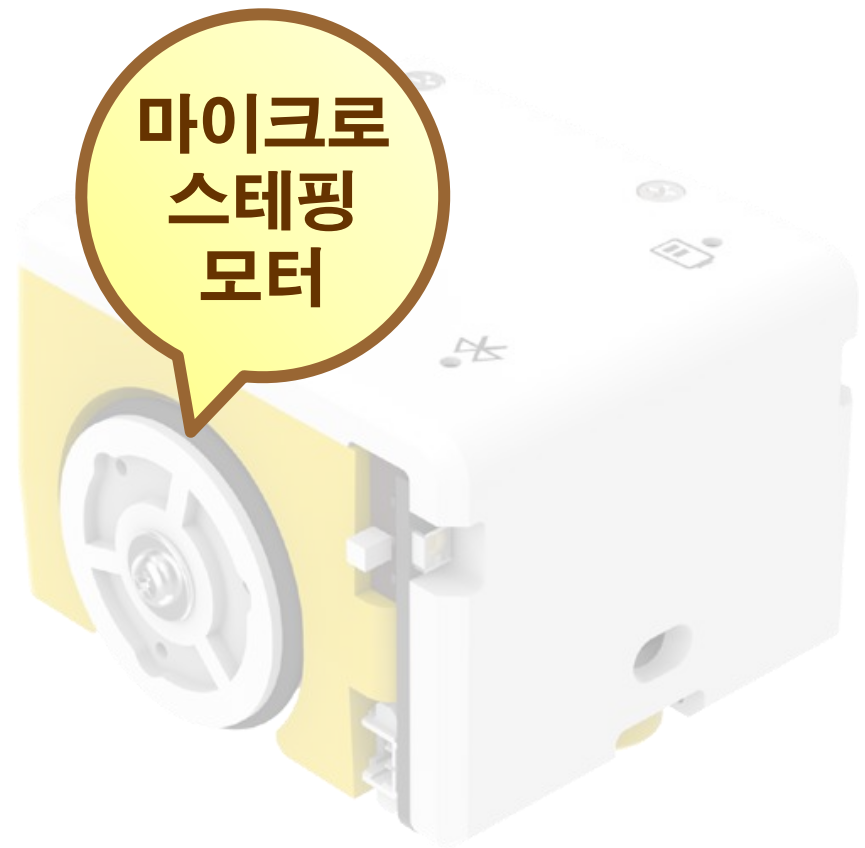




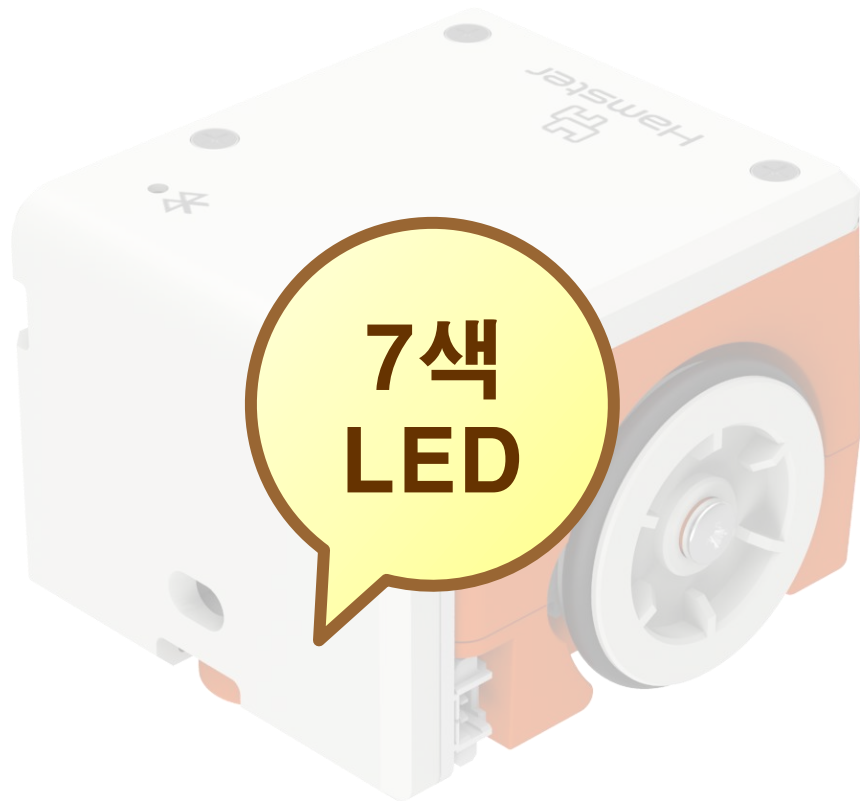
## HAMSTER



## HAMSTER-S



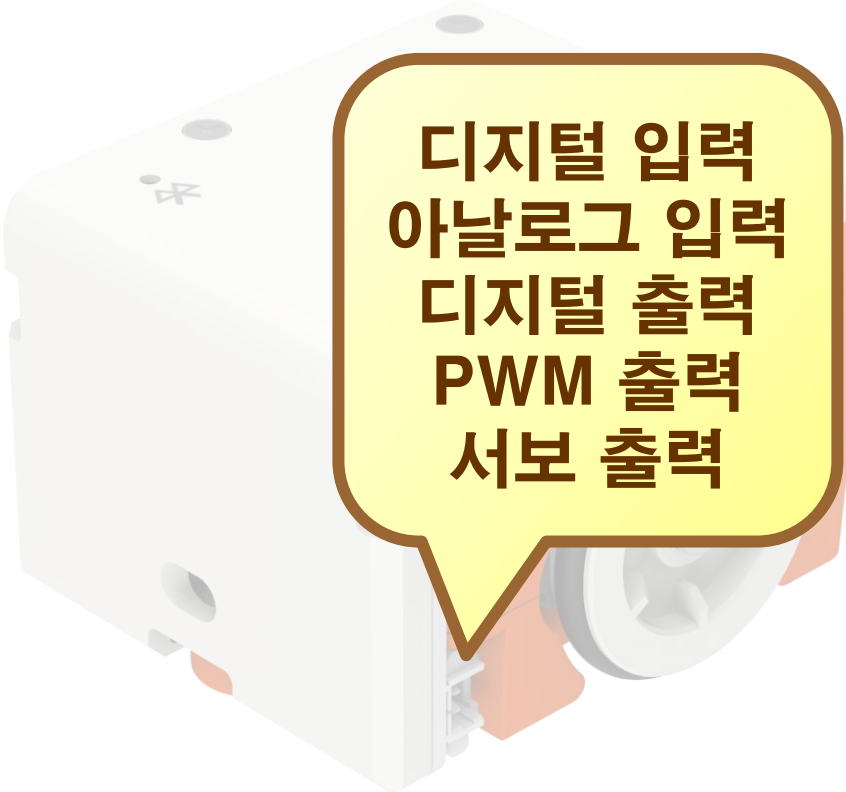
## HAMSTER



## HAMSTER-S



## HAMSTER

A white and orange HAMSTER robot with a single wheel and a servo motor. A yellow speech bubble is overlaid on the robot.

디지털 입력  
아날로그 입력  
디지털 출력  
PWM 출력  
서보 출력

## HAMSTER-S

A white and yellow HAMSTER-S robot with two wheels and a servo motor. A yellow speech bubble is overlaid on the robot.

+ 풀업/풀다운  
+ 시리얼 통신

# 햄스터 + 거북이 = 햄스터S



# 햄스터 로봇 연결

- USB 동글을 PC의 USB 단자에 연결



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





```
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

왼쪽 바닥 센서  
(적외선 센서)  
0~100단계 감지

오른쪽 바닥 센서  
(적외선 센서)  
0~100단계 감지



```
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)



```
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



```
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

41



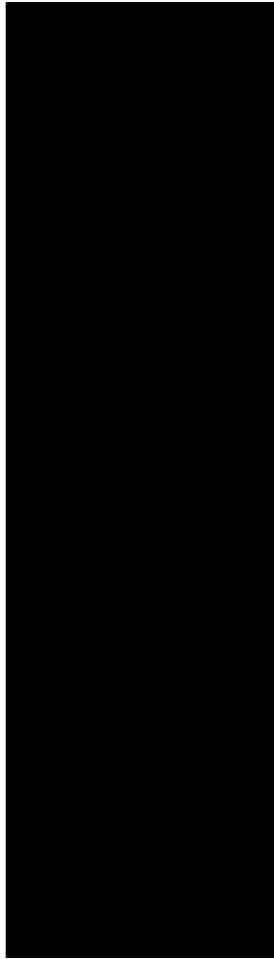
```
from roboid import *

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.
```



```
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



```
from roboid import *

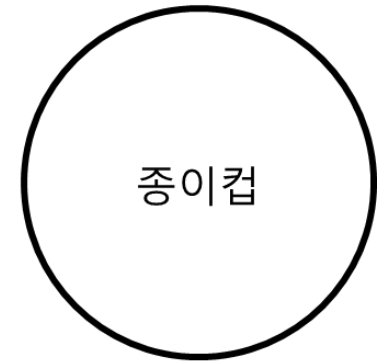
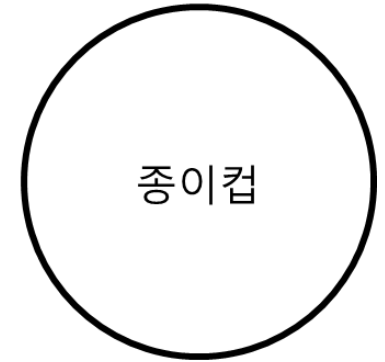
hamster = Hamster()

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
```





# 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.
```

# 해스터 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



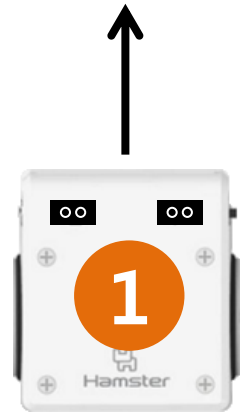
```
from roboid import *

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

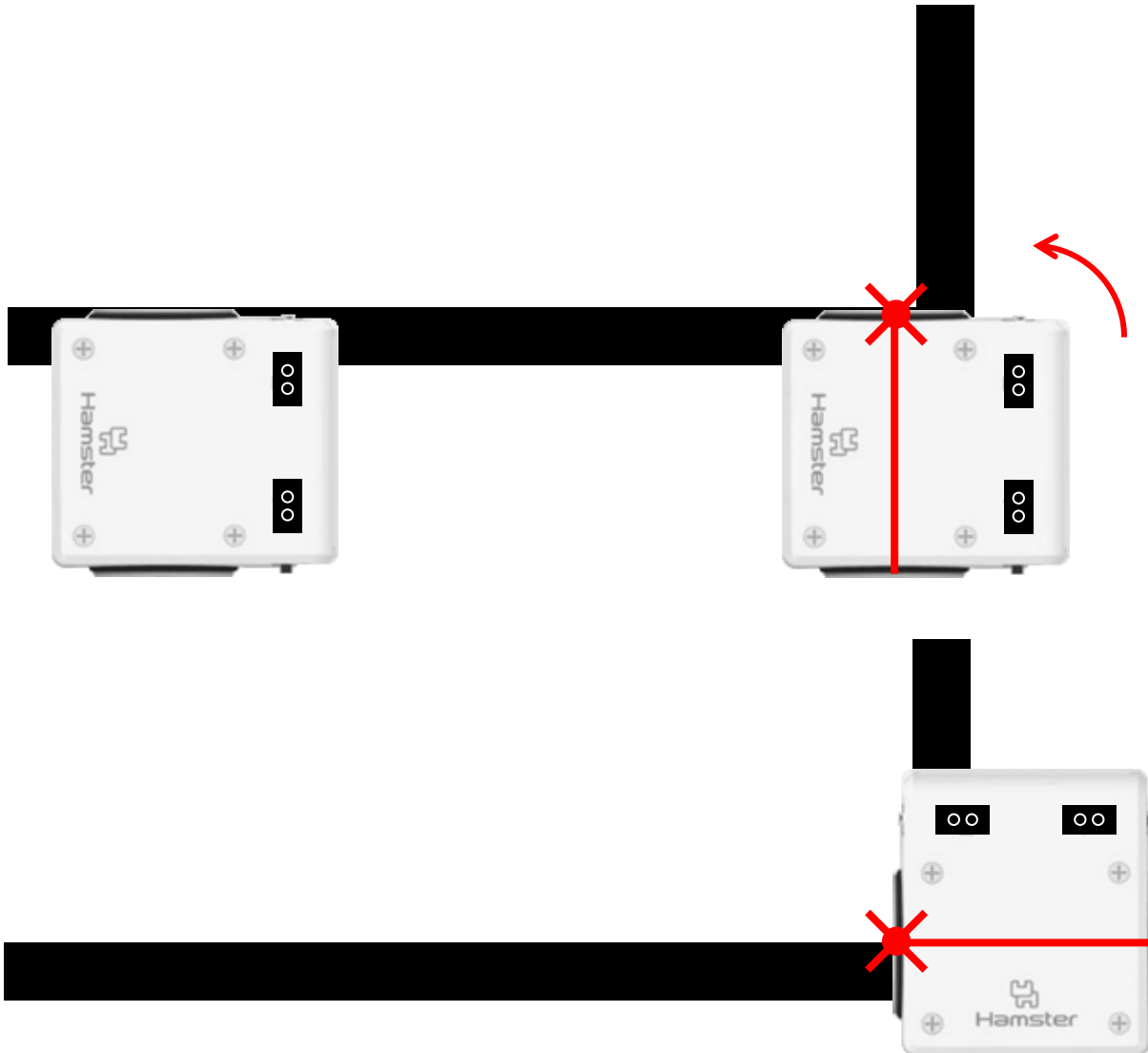
56





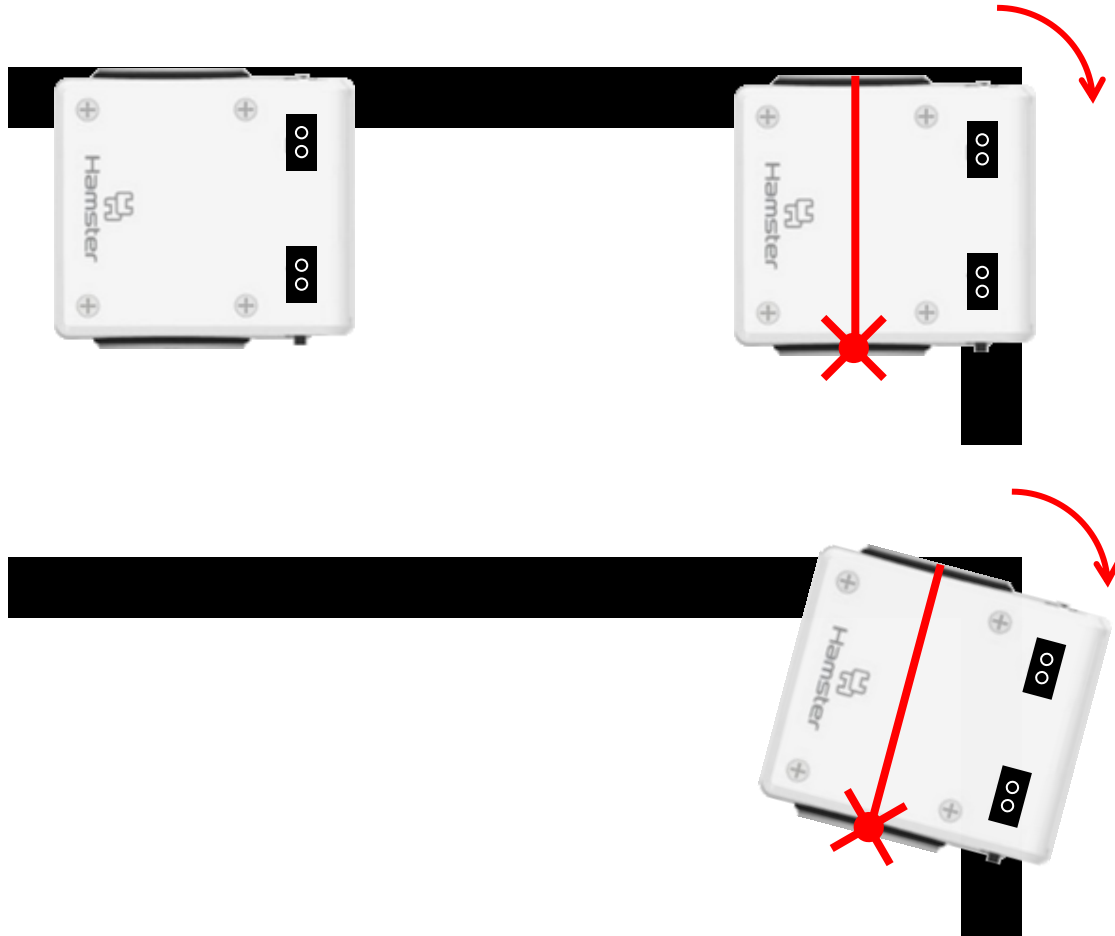
# 왼쪽 센서 + 오른쪽 가장자리

57



# 왼쪽 센서 + 오른쪽 가장자리

58



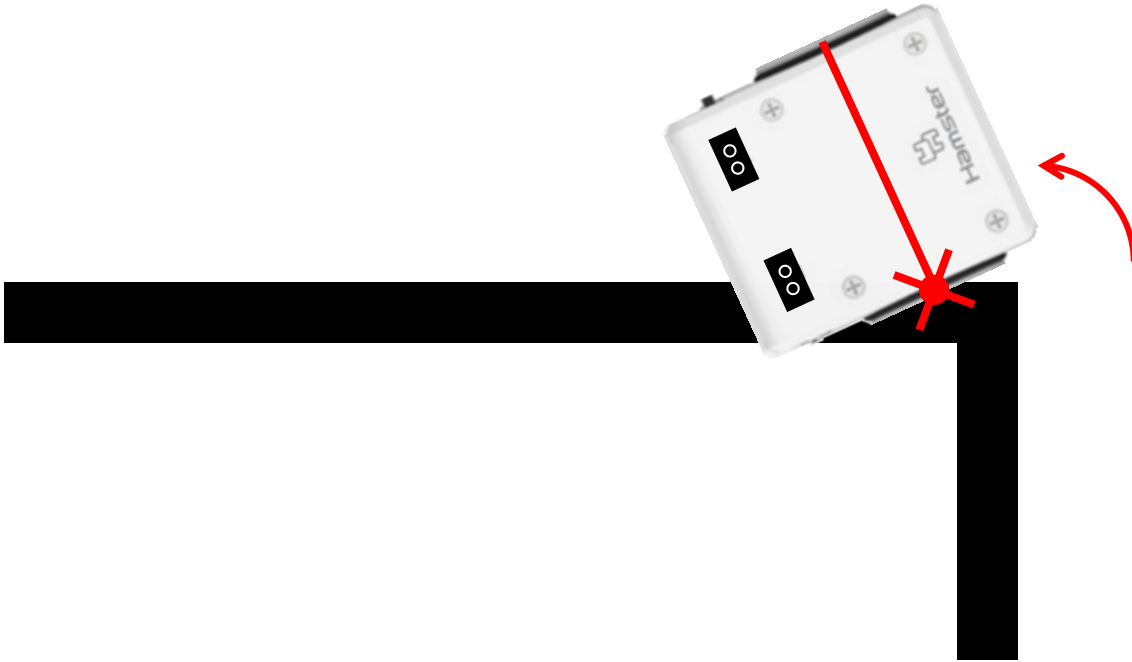
# 왼쪽 센서 + 오른쪽 가장자리

59



# 왼쪽 센서 + 오른쪽 가장자리

60



# 왼쪽 센서 + 왼쪽 가장자리

61



```
from roboid import *

hamster = Hamster()

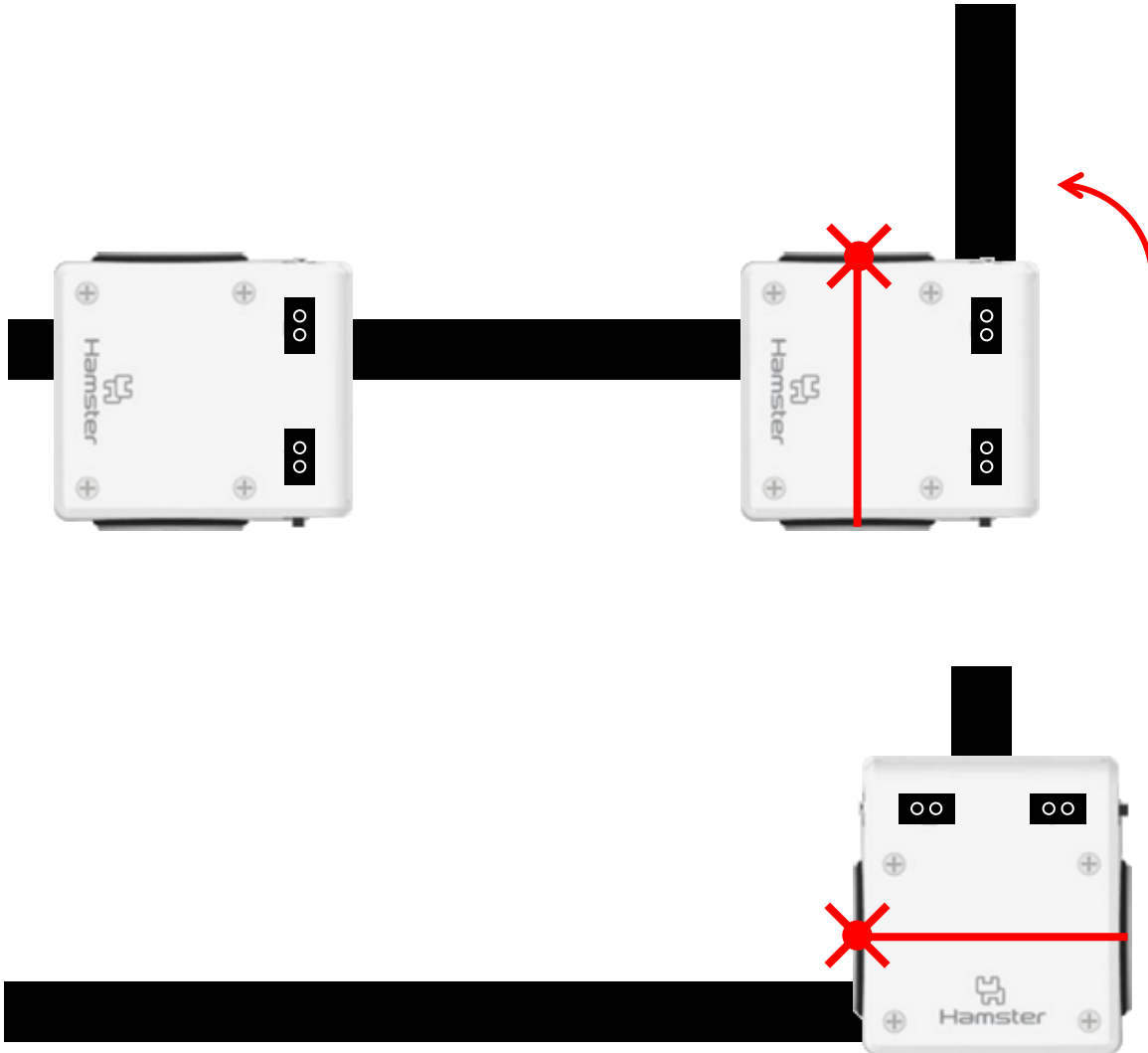
while True:
    if hamster.left_floor() > 50:
        hamster.wheels(30, 0)
    else:
        hamster.wheels(0, 30)
    wait(10) # 너무 빨리 반복하지 않도록 한다.
```

# 왼쪽 센서 + 왼쪽 가장자리

63



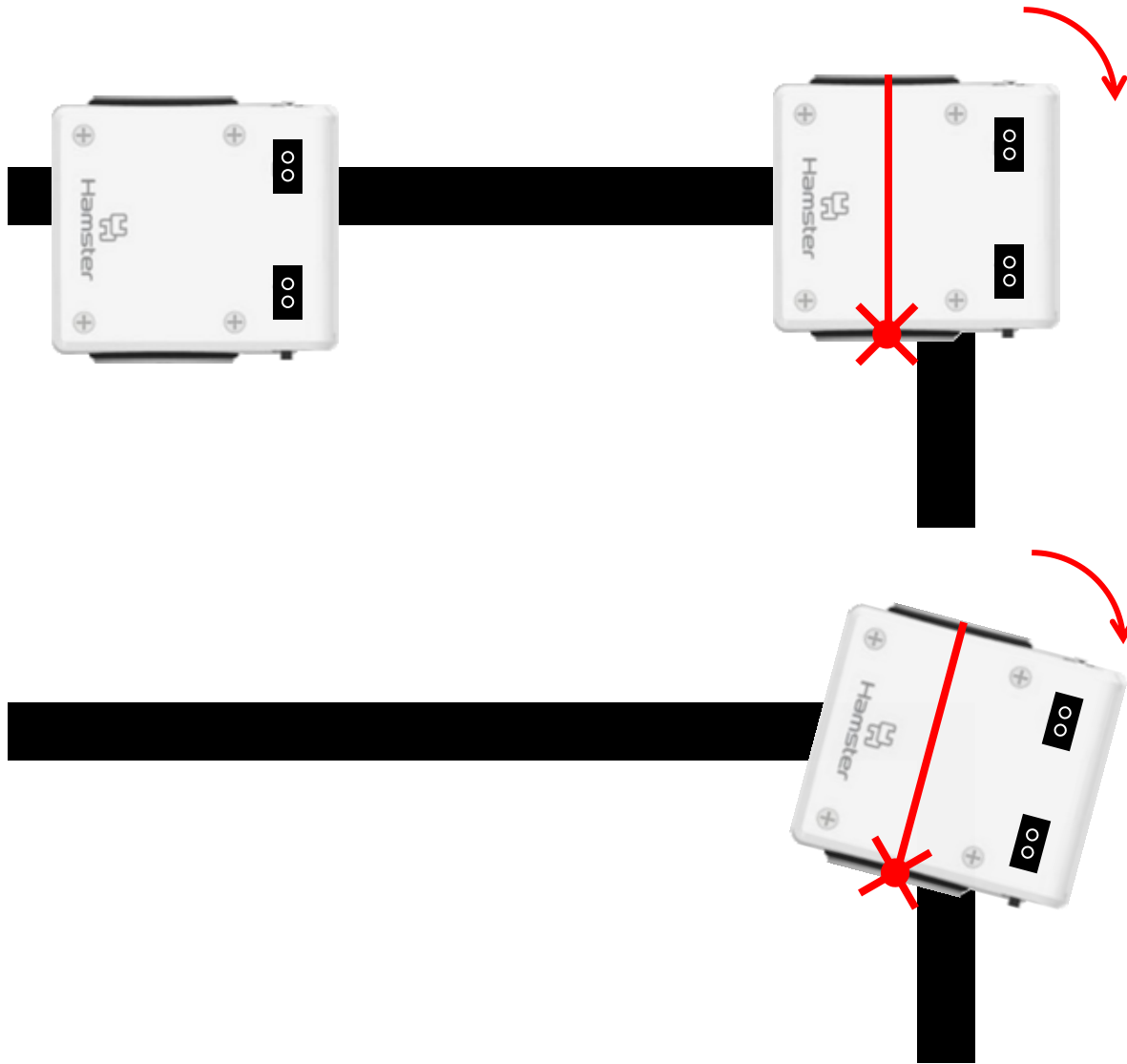
# 왼쪽 센서 + 왼쪽 가장자리





# 왼쪽 센서 + 왼쪽 가장자리

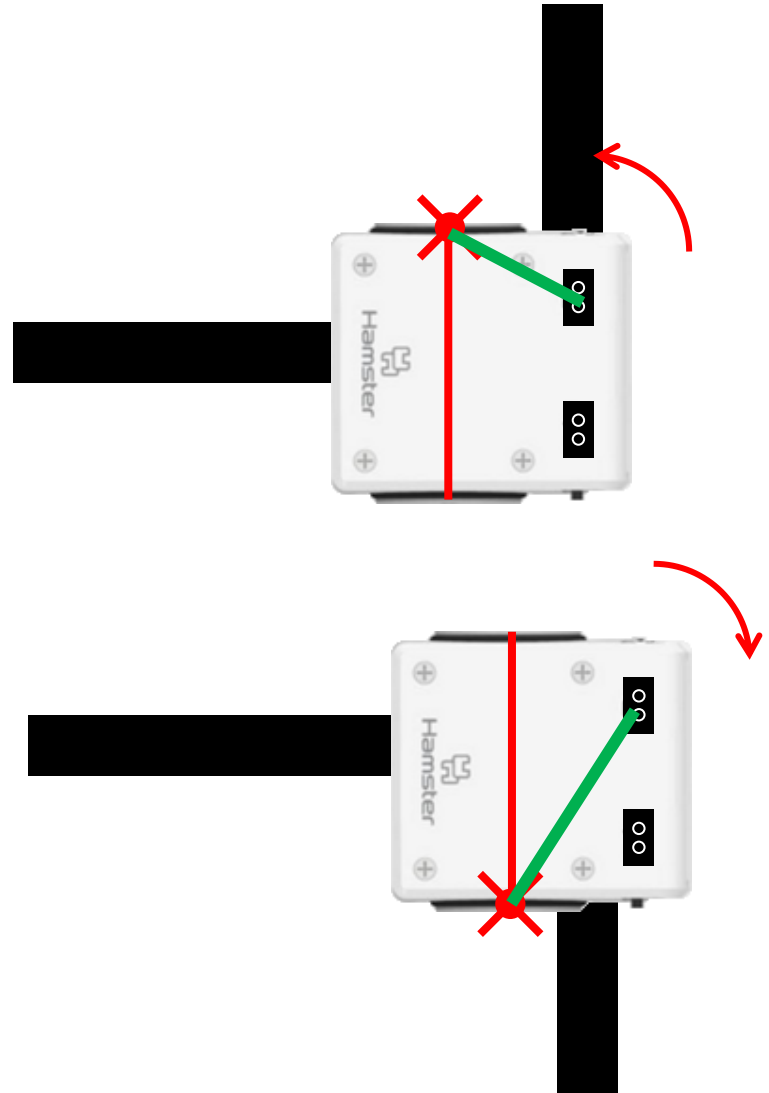
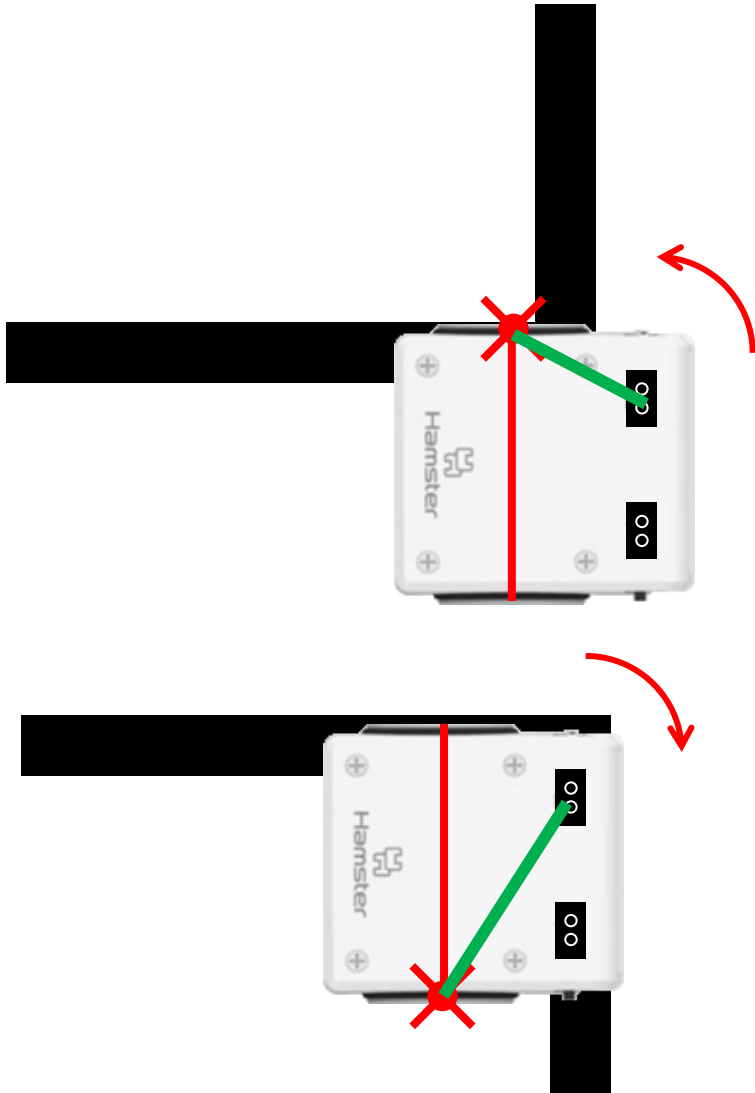
65





# 오른쪽 가장자리

# 왼쪽 가장자리





```
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.
```



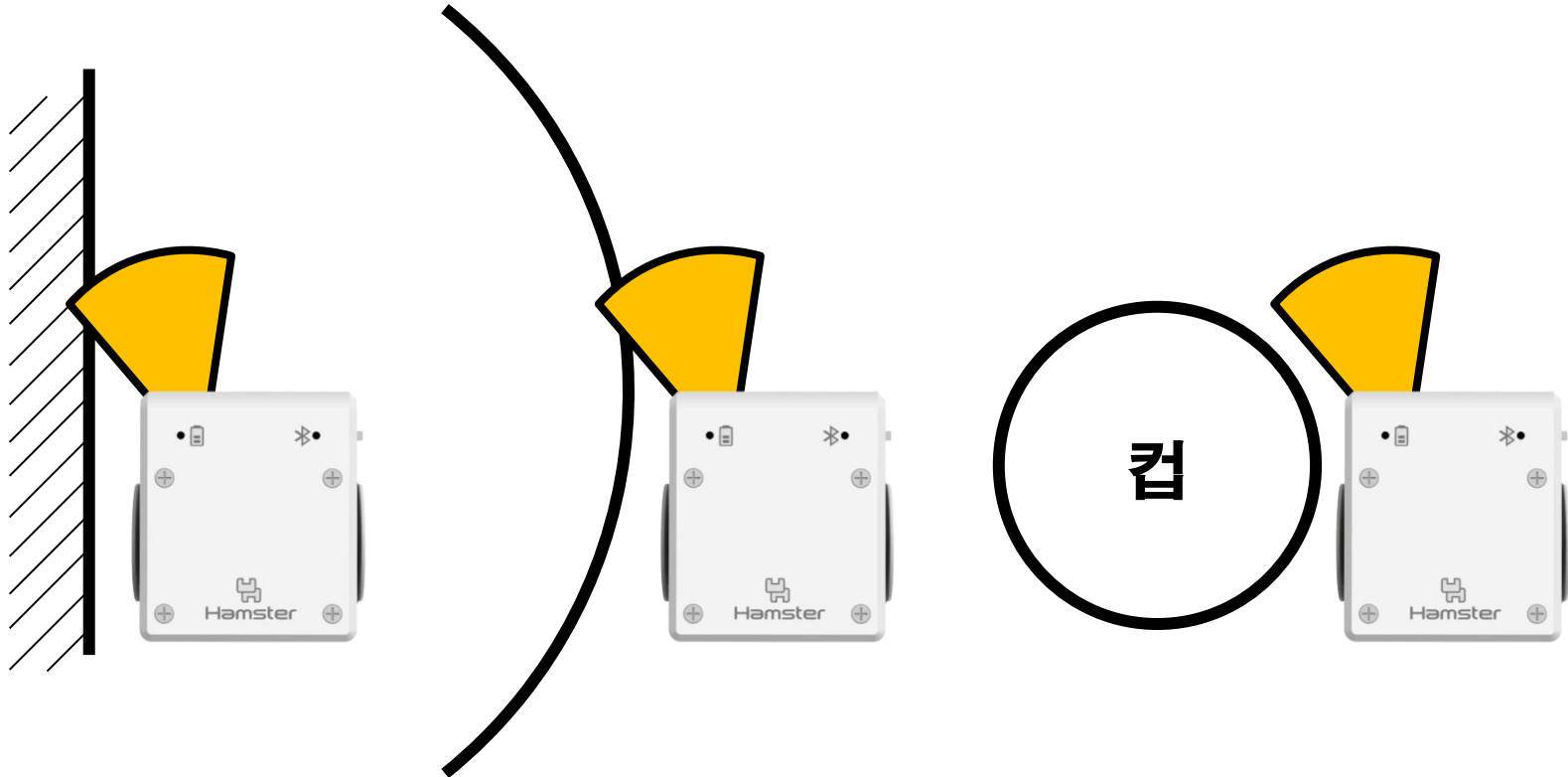
```
from roboid import *

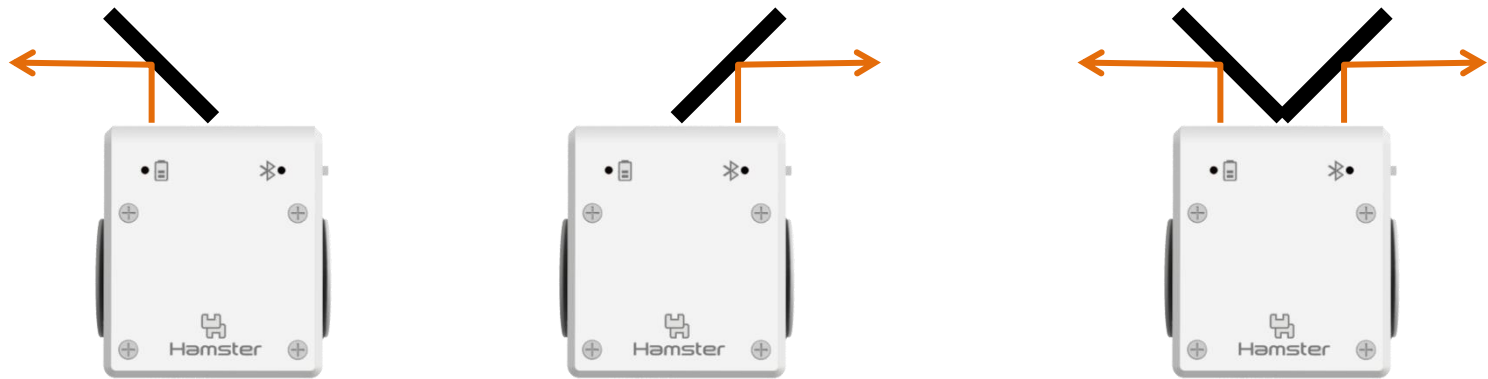
hamster = Hamster()

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.
```

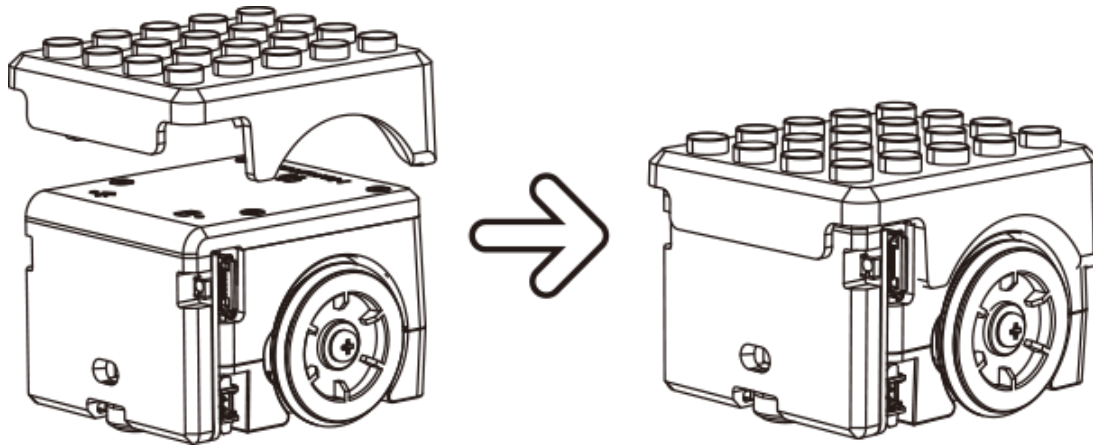
**컵 따라 돌기**



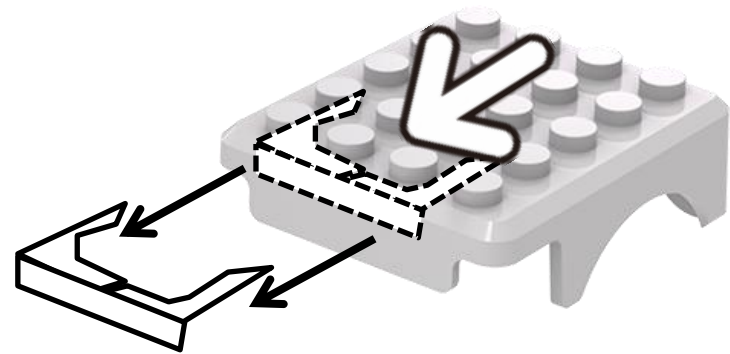
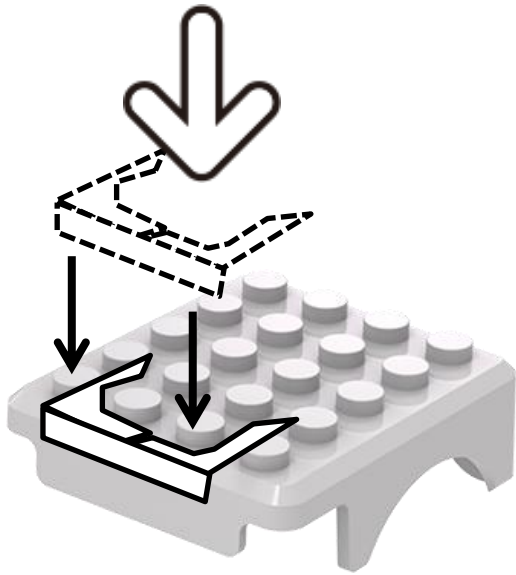




**양쪽 반사판을 끼워 주세요 !!**

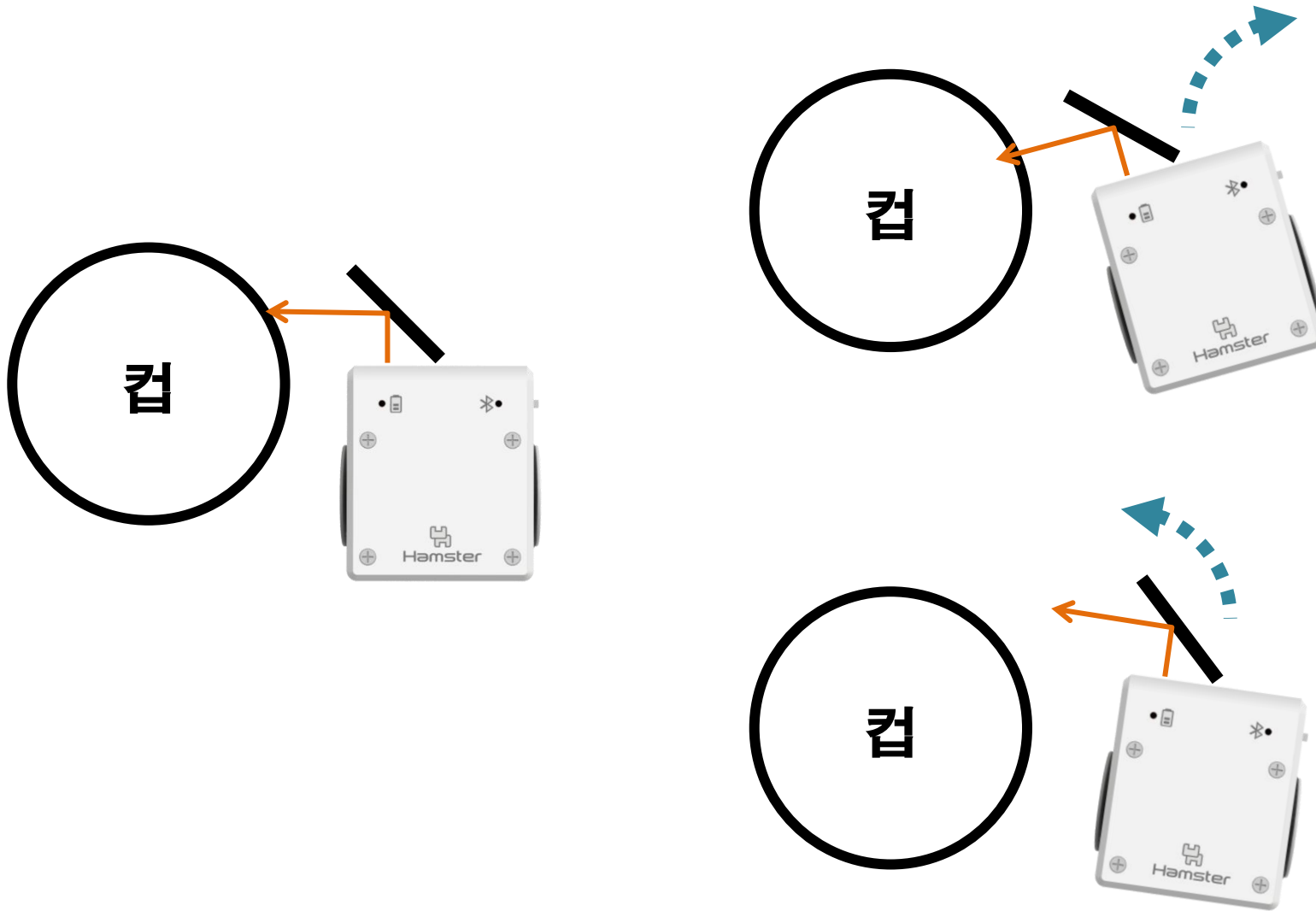


위에서 누르면 끼울 수 있어요



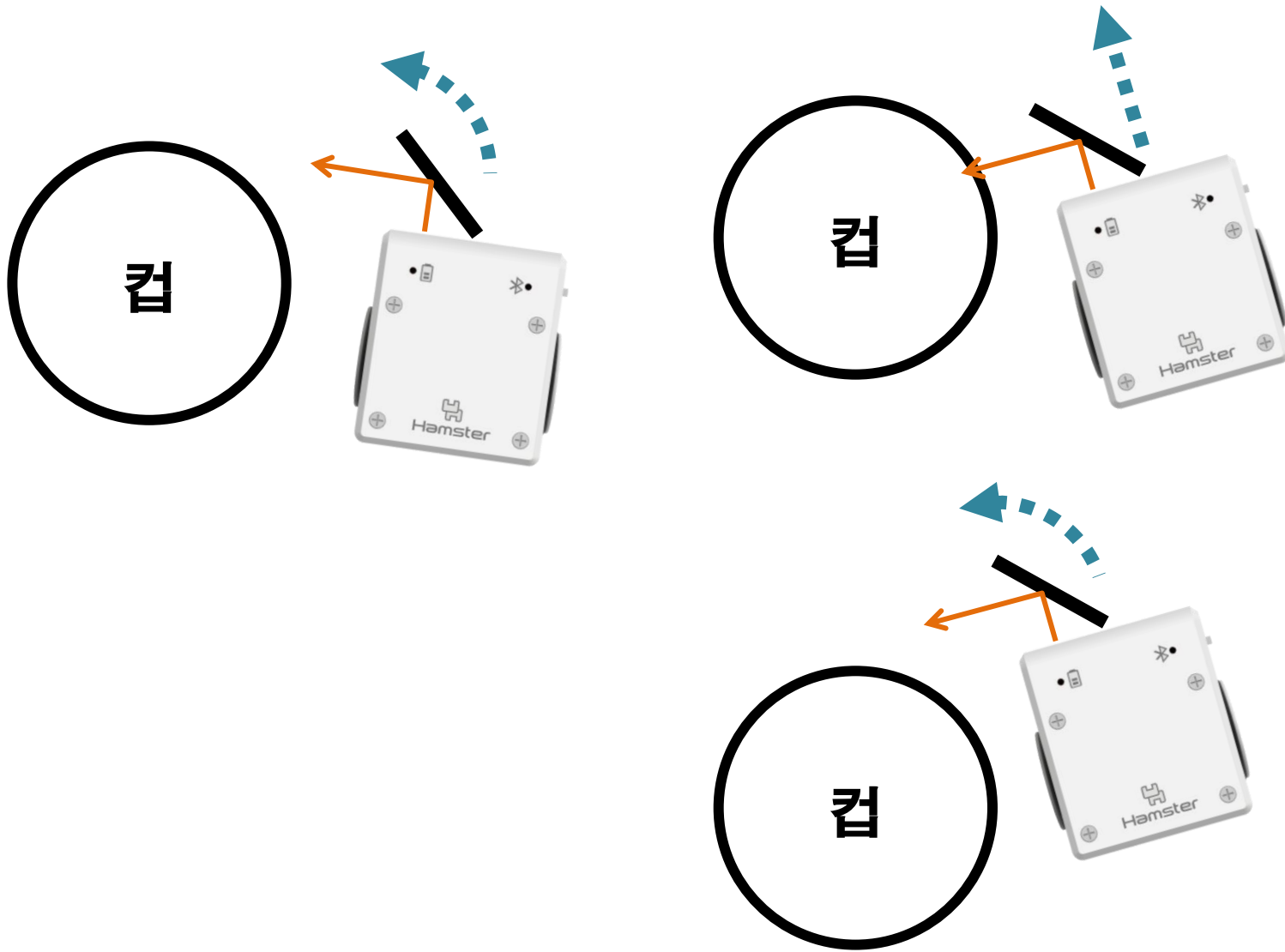
뒤에서 앞으로 밀면 뺄 수 있어요

# 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()`



```
from roboid import *

hamster = Hamster()

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



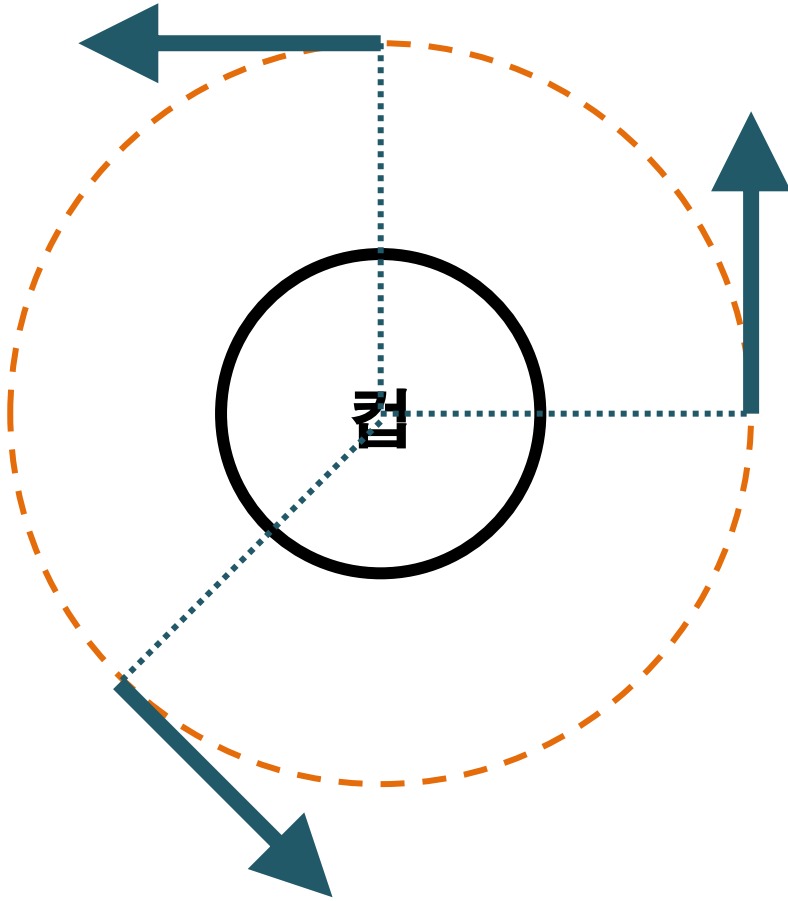
```
from roboid import *

hamster = Hamster()

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

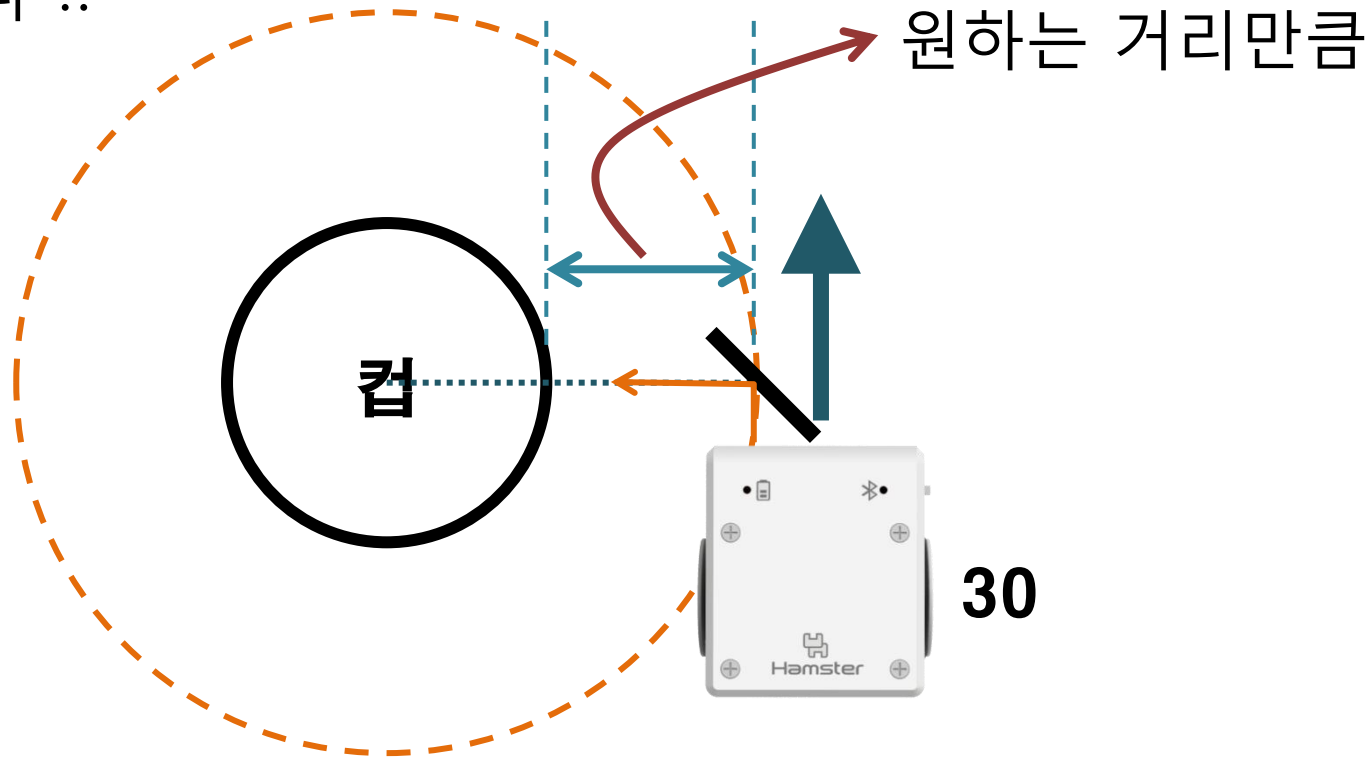
근접 센서 값이 커지면  
바퀴의 속도도 커진다 ?

```
from roboid import *  
  
hamster = Hamster()  
  
while True:  
    hamster.wheels(hamster.left_proximity(), 30)  
    wait(10)
```

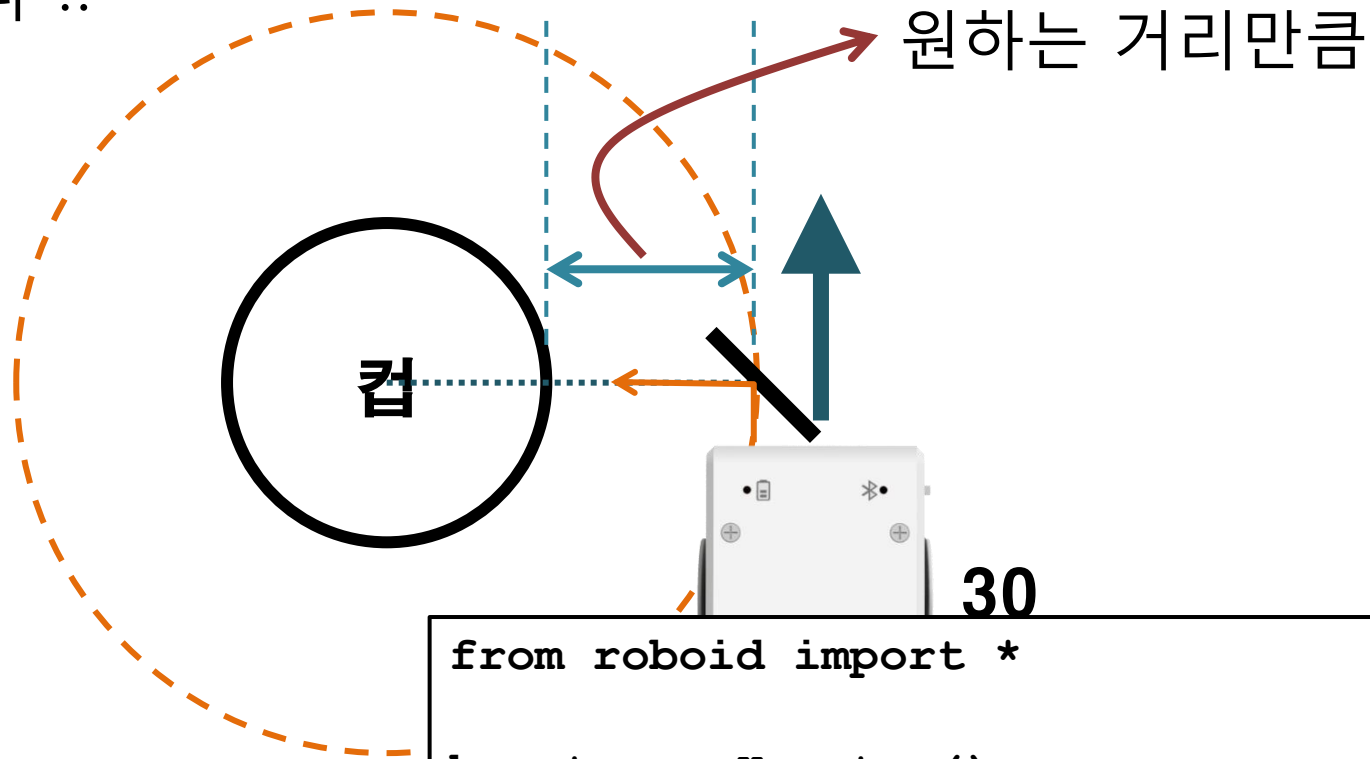


# 원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에  
맞춘다 !!



센서 위치를 컵의 중심에  
맞춘다 !!

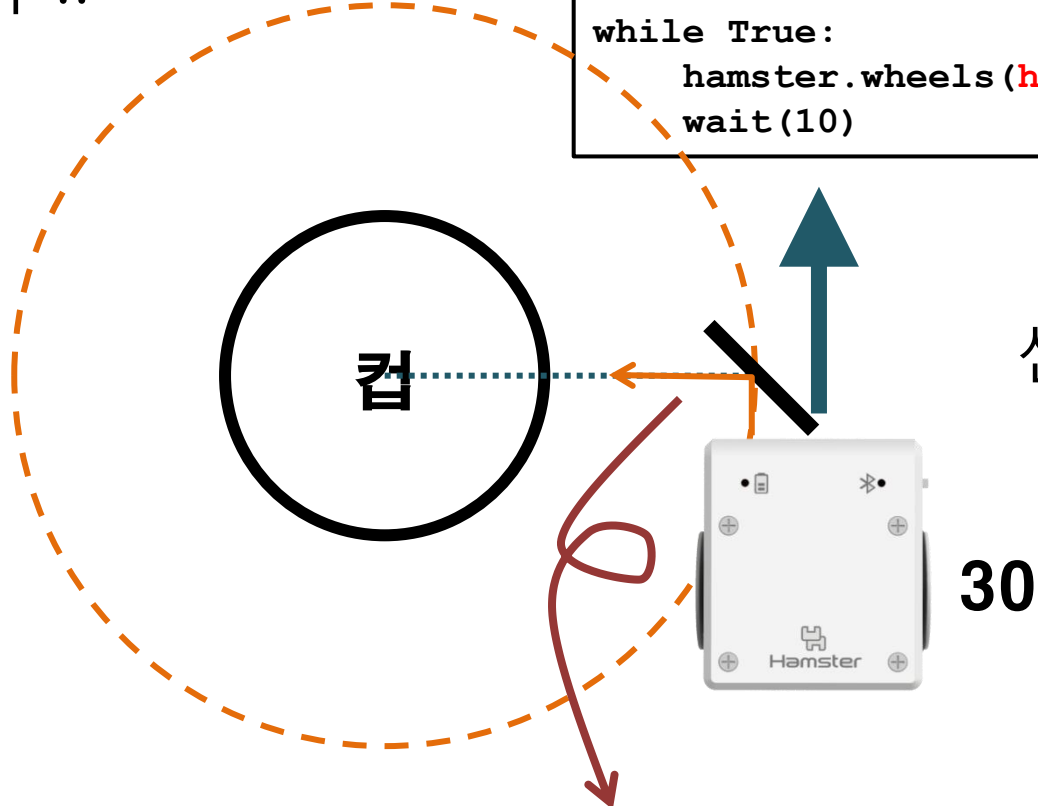


```
from roboid import *  
  
hamster = Hamster()  
  
while True:  
    print(hamster.left_proximity())  
    wait(20) # 너무 빨리 반복하지 않도록 한다.
```

# 원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에  
맞춘다 !!

```
from roboid import *  
  
hamster = Hamster()  
  
while True:  
    hamster.wheels(hamster.left_proximity(), 30)  
    wait(10)
```



센서 값: 16

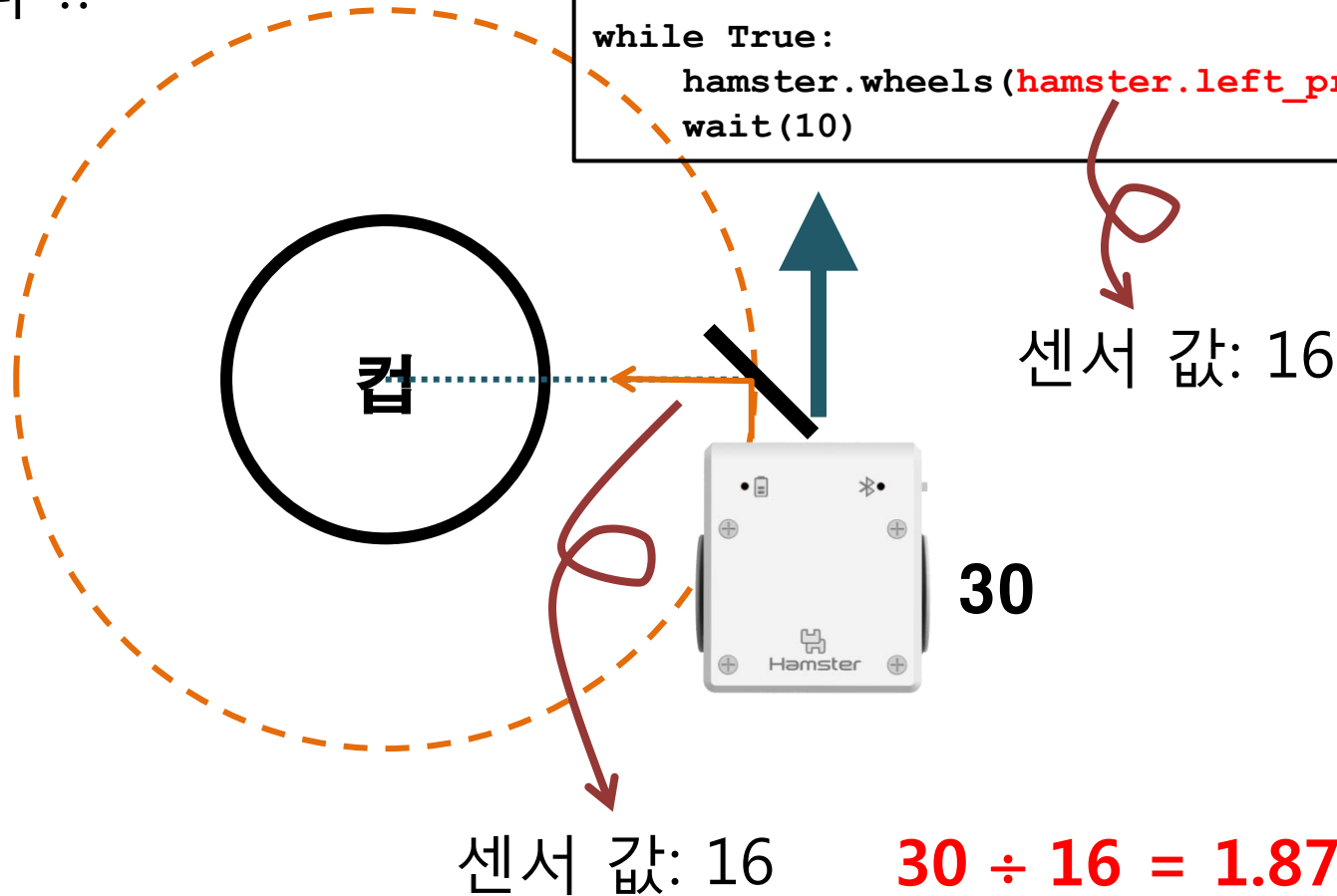
센서 값: 16

# 원하는 거리만큼 떨어져서 돌기

87

센서 위치를 컵의 중심에  
맞춘다 !!

```
from roboid import *  
  
hamster = Hamster()  
  
while True:  
    hamster.wheels(hamster.left_proximity(), 30)  
    wait(10)
```

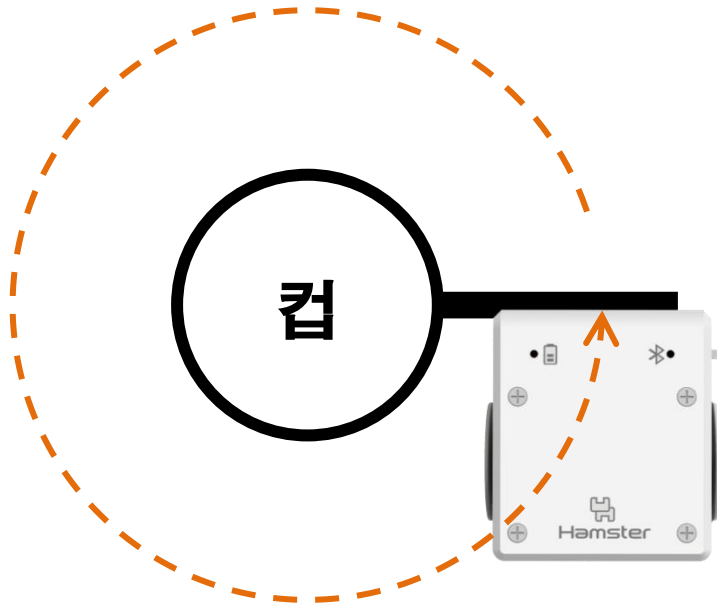


```
from roboid import *  
  
hamster = Hamster()  
  
while True:  
    hamster.wheels(hamster.left_proximity() * 1.875, 30)  
    wait(10)
```



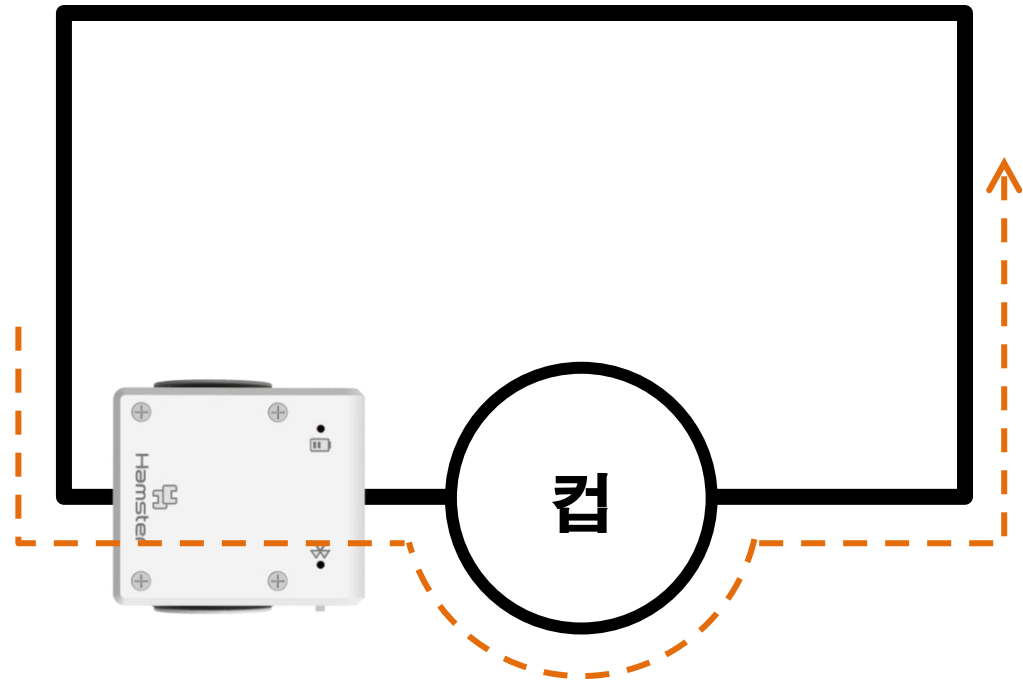
일정 횟수만큼 돌고 정지하기

중급



고급

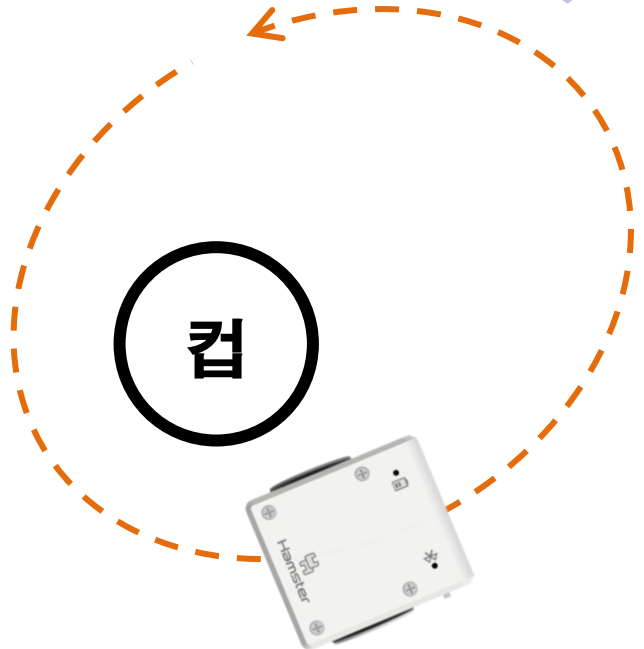
라인 트레이싱 + 컵 따라 돌기



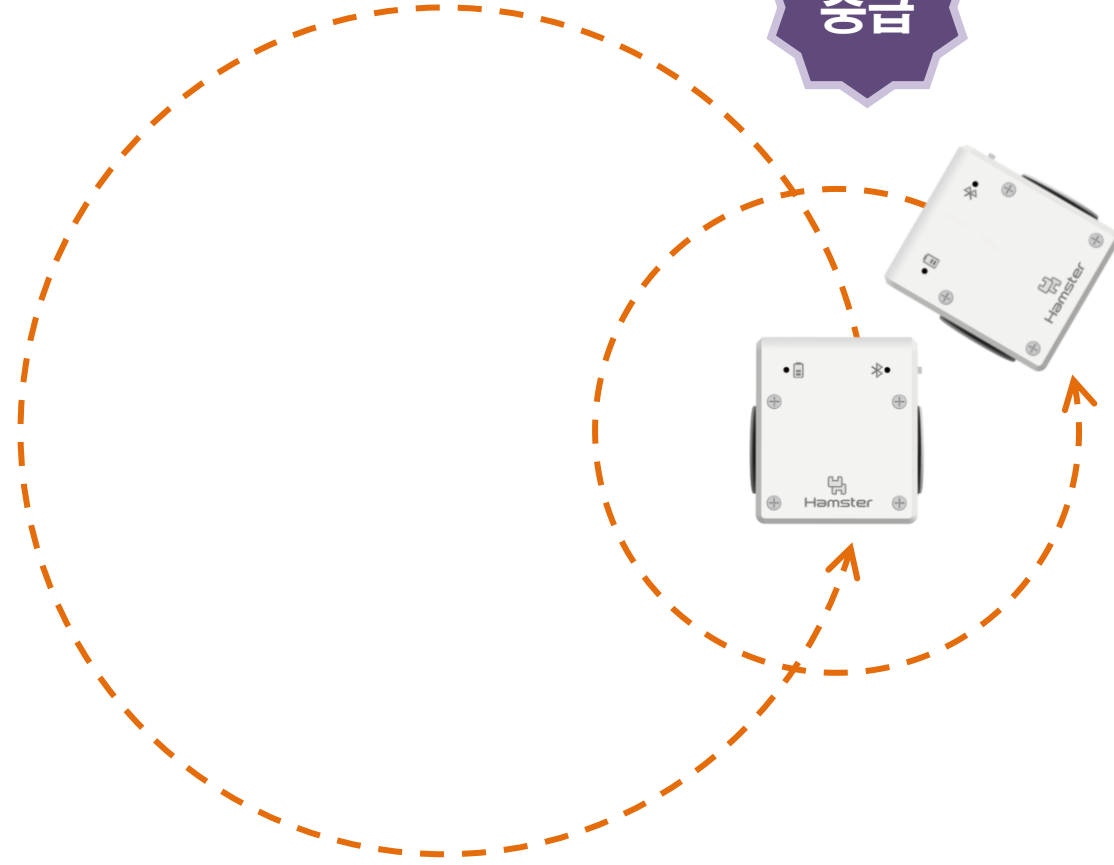
# 여러 가지 활동들

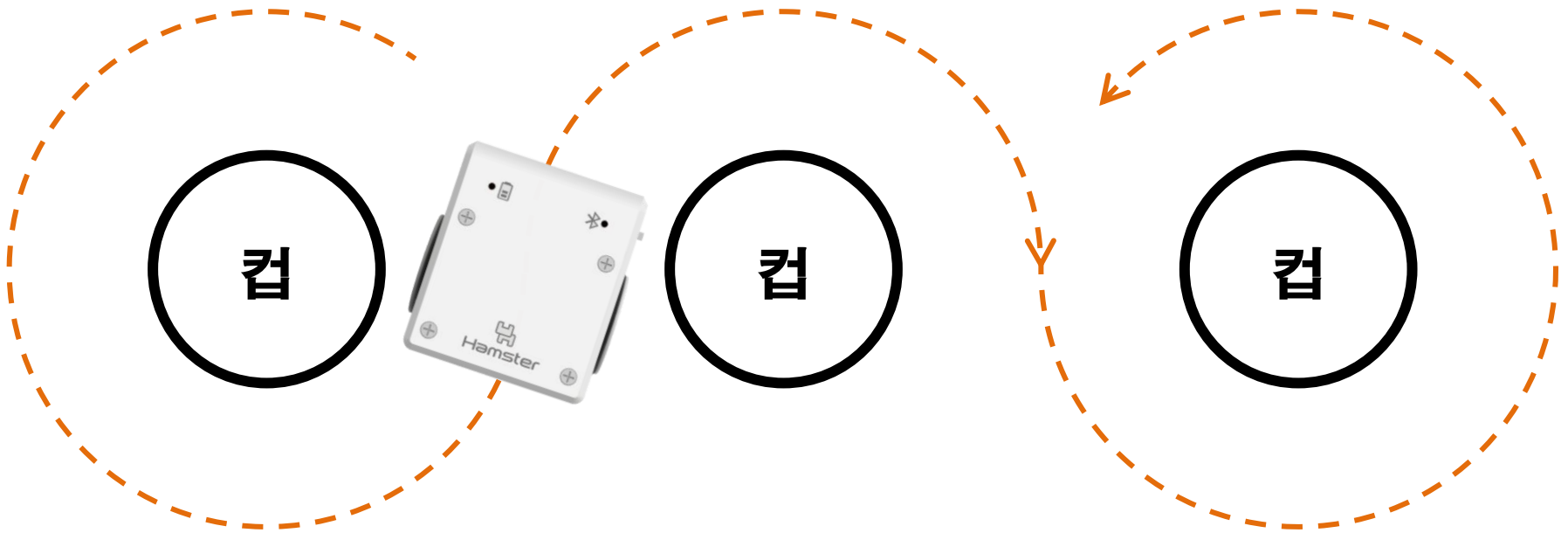
햄스터 플래닛

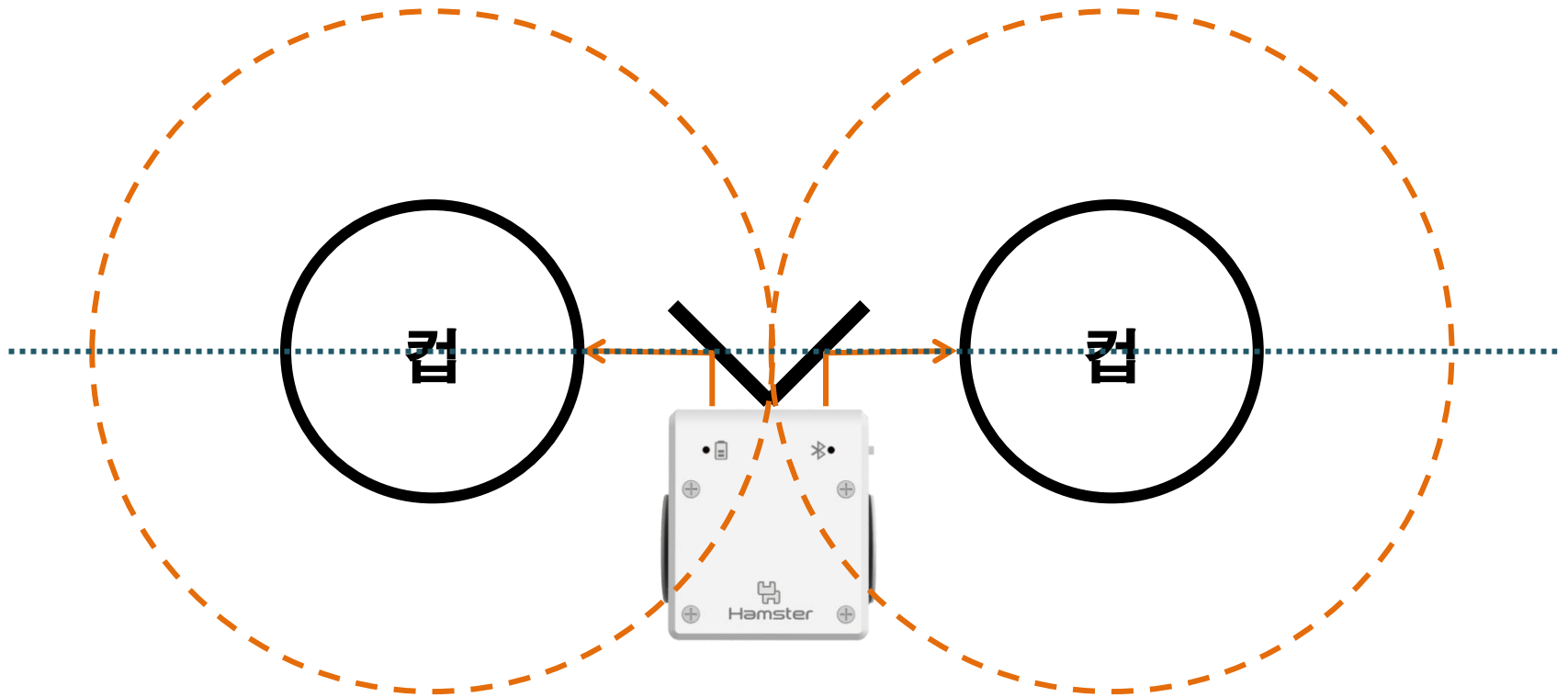
중급  
고급



중급



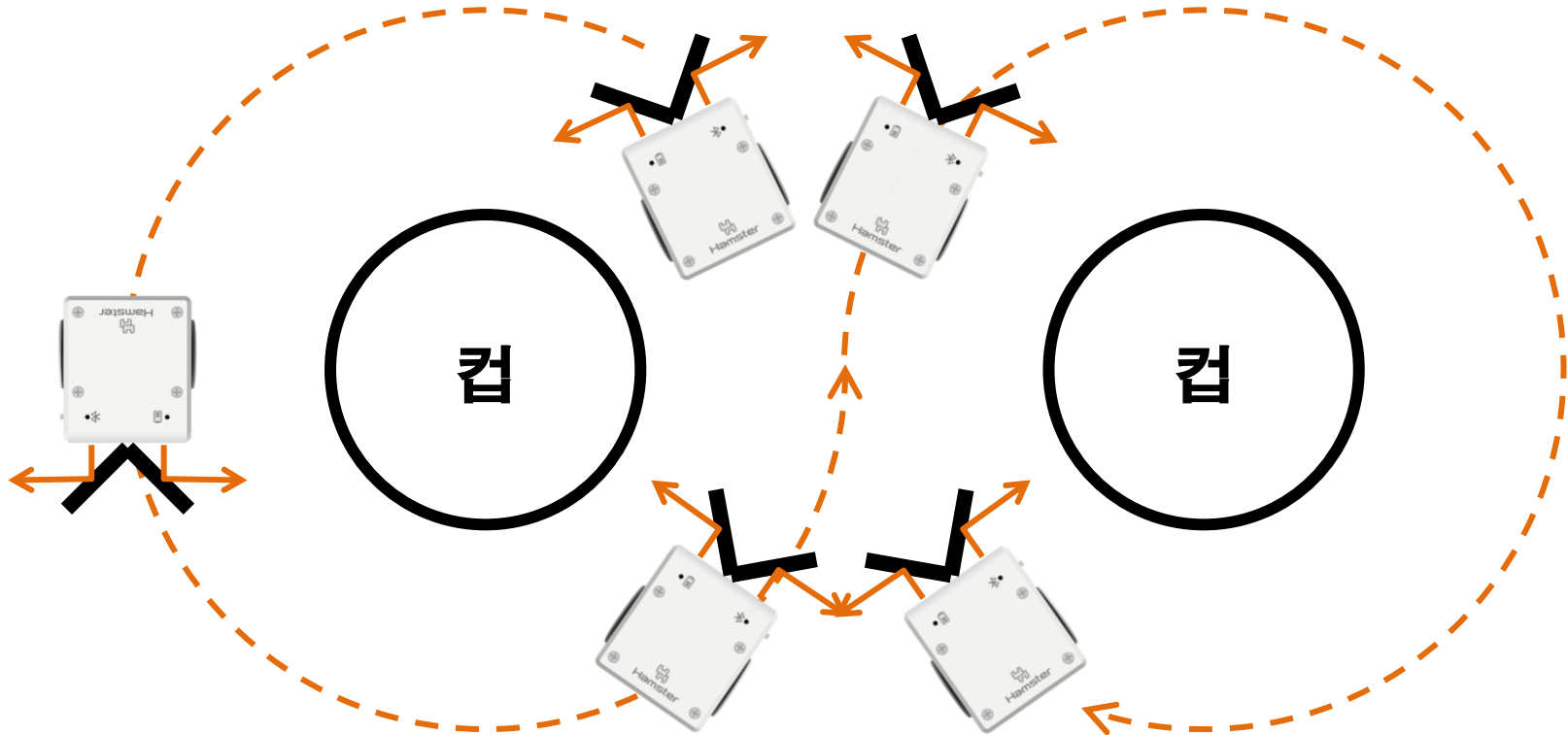




**컵 중간으로 지나가게 하는 방법은?**

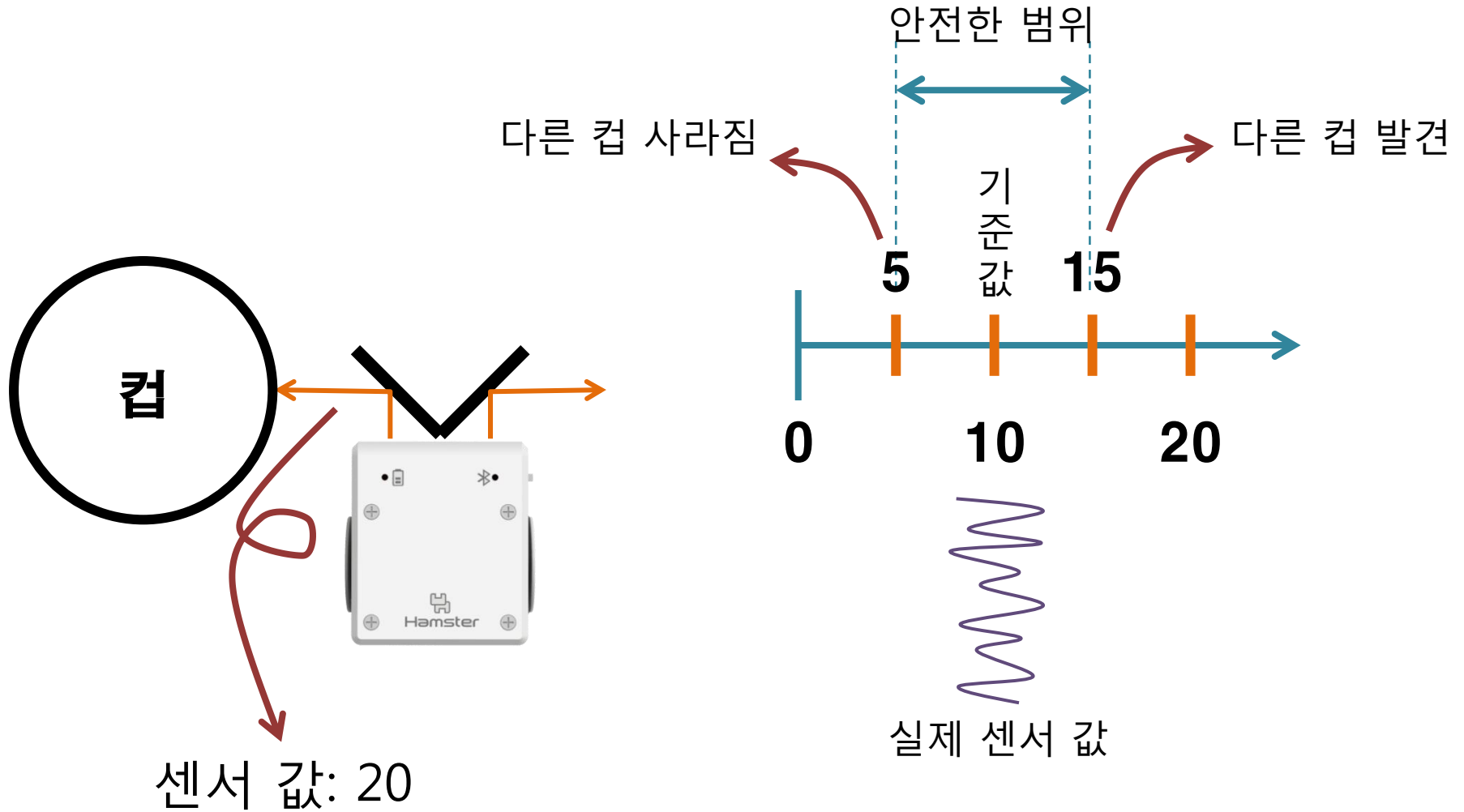
4 오른쪽 컵이 사라질 때까지  
왼쪽 컵 따라 돌기

2 왼쪽 컵이 사라질 때까지  
오른쪽 컵 따라 돌기



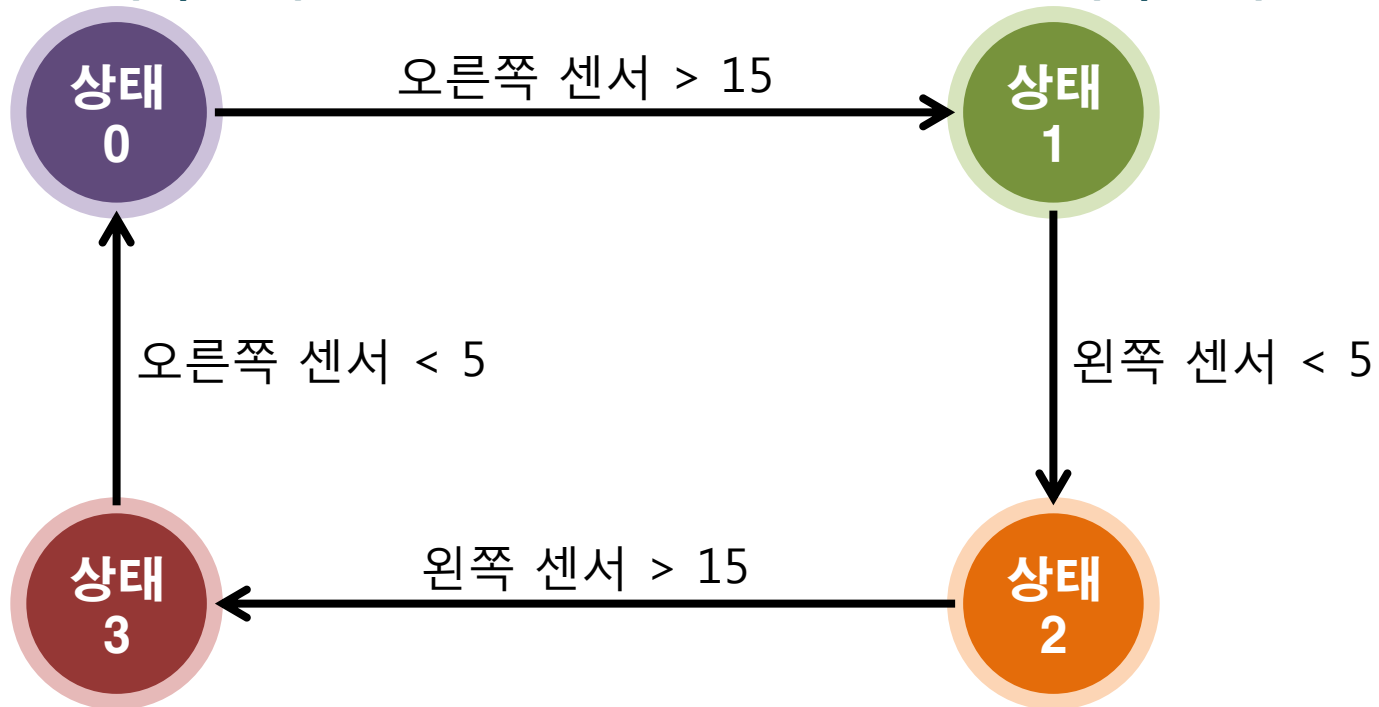
1 오른쪽 컵을 발견할 때까지  
왼쪽 컵 따라 돌기

3 왼쪽 컵을 발견할 때까지  
오른쪽 컵 따라 돌기



왼쪽 컵 따라 돌기

오른쪽 컵 따라 돌기



왼쪽 컵 따라 돌기

오른쪽 컵 따라 돌기

# 잘 안 되는 코드... 왜?

96

```
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

97

```
from roboid import *

hamster = Hamster()

state = 0
count = 0
while True:
    left = hamster.left_proximity()
    right = hamster.right_proximity()
    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)
```

```
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

```
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"

```
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)
```

```
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.
```

```
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)
```

```
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)
```



```
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](mailto:akaii@kw.ac.kr)**