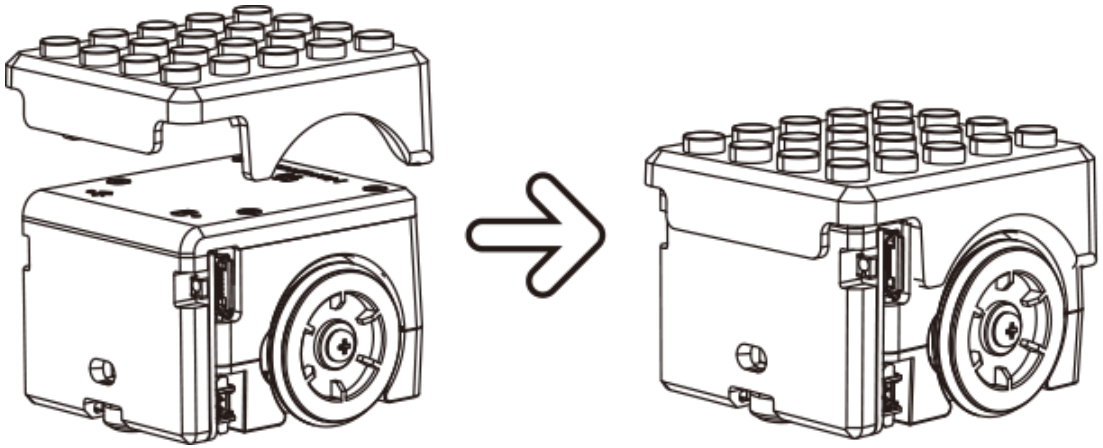
햄스터 미로찾기

광운대학교 로봇학부
박광현

블록 커버

레고블록



나노블록

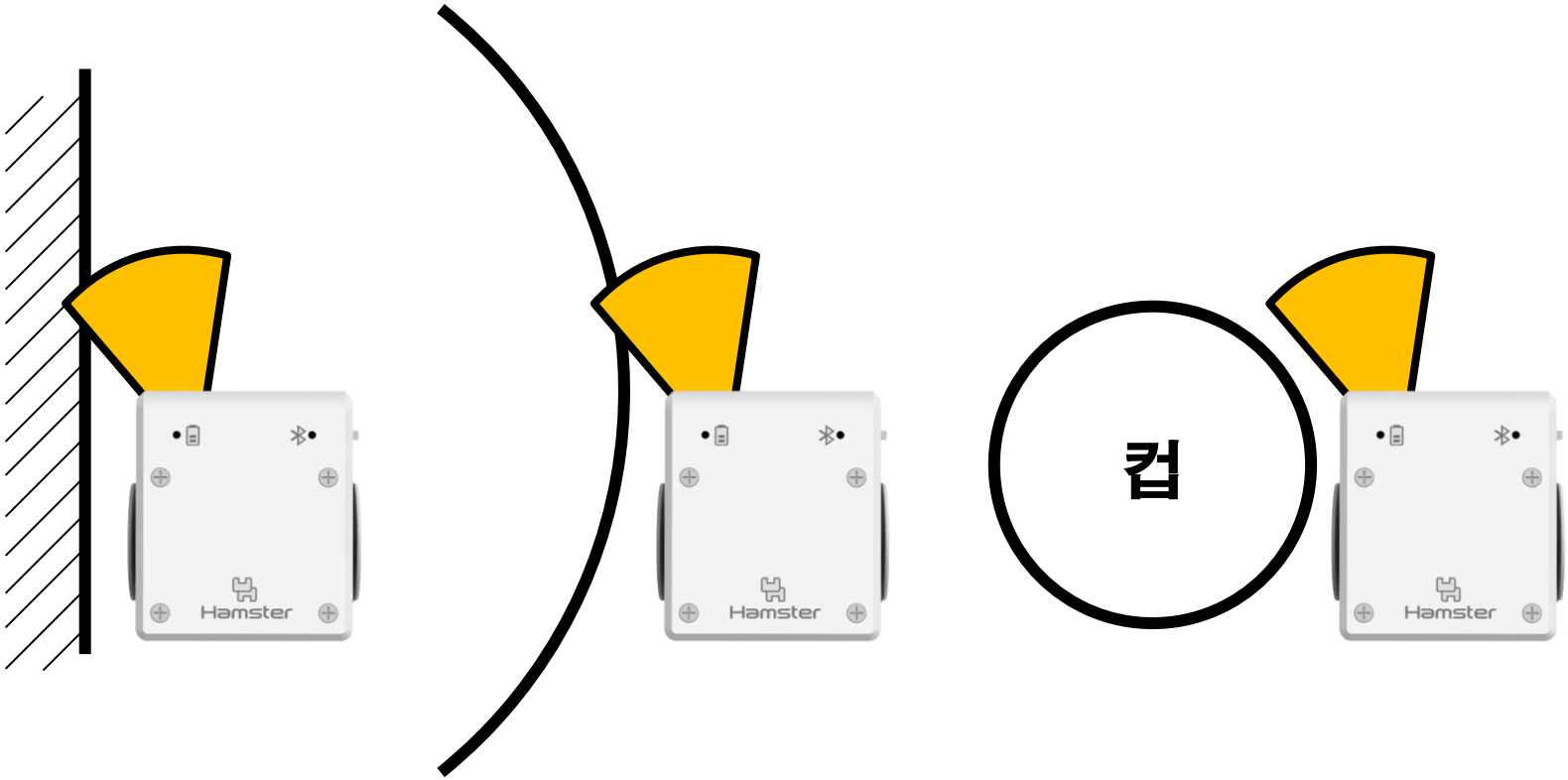
블록 커버

레고블록

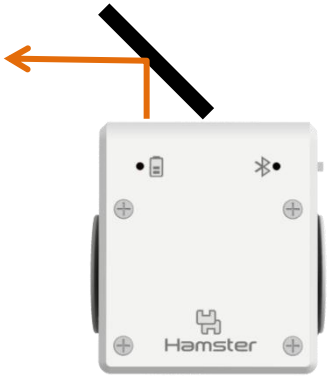
나노블록

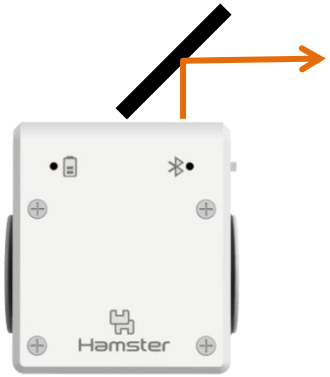
컵 따라 돌기

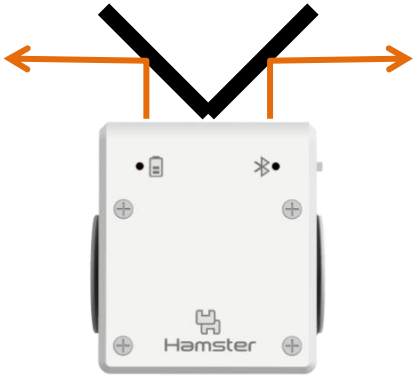
근접 센서



반사판



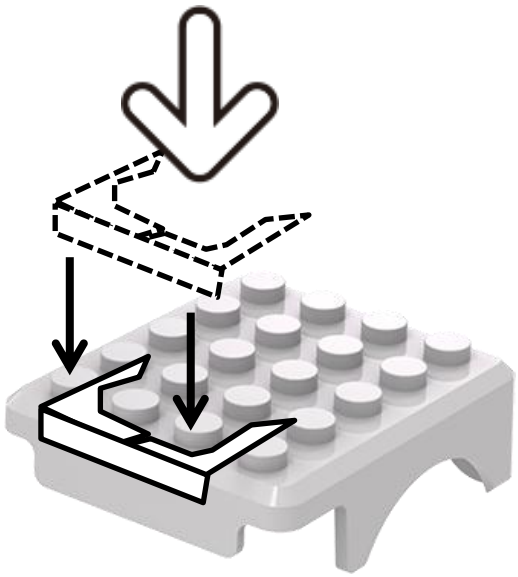


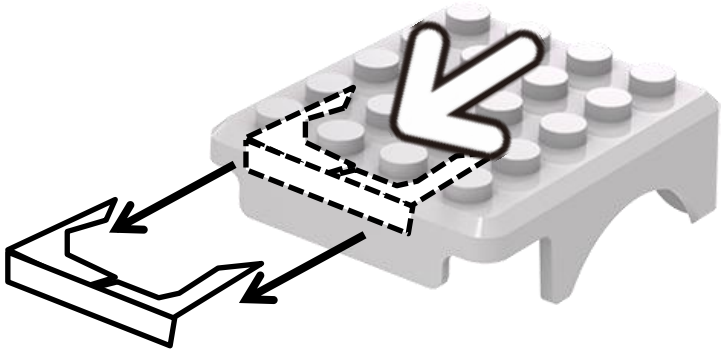


양쪽 반사판을 끼워 주세요 !!

미로찾기용 부품 조립 방법

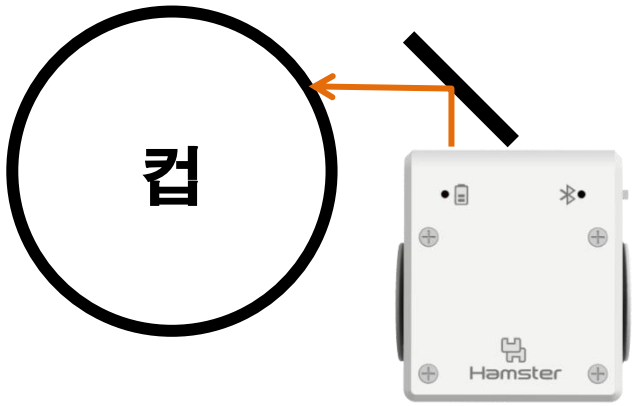
위에서 누르면 끼울 수 있어요

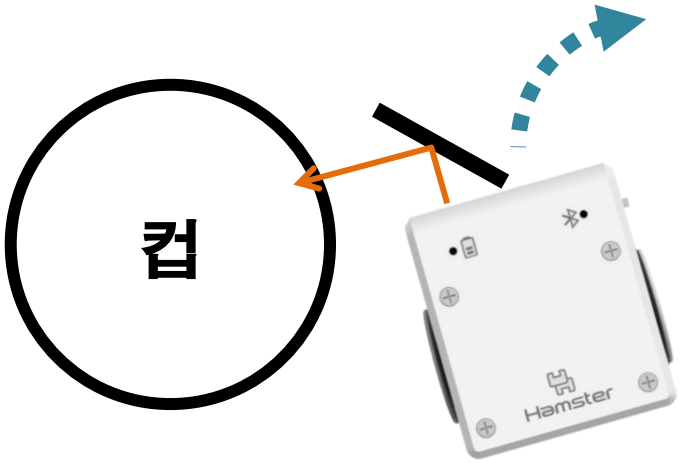


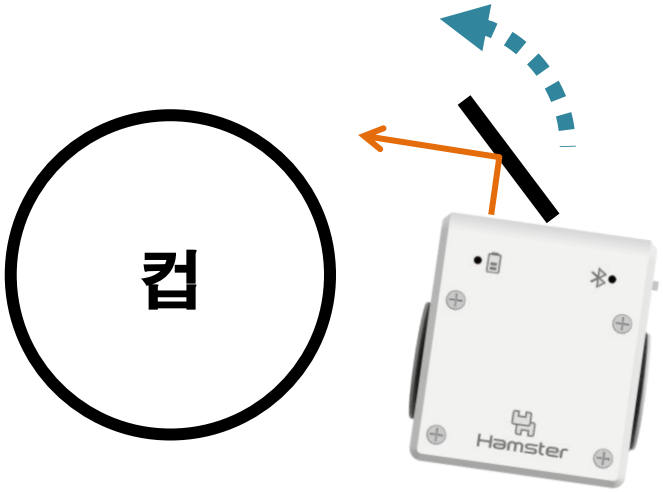


뒤에서 앞으로 밀면 뺄 수 있어요

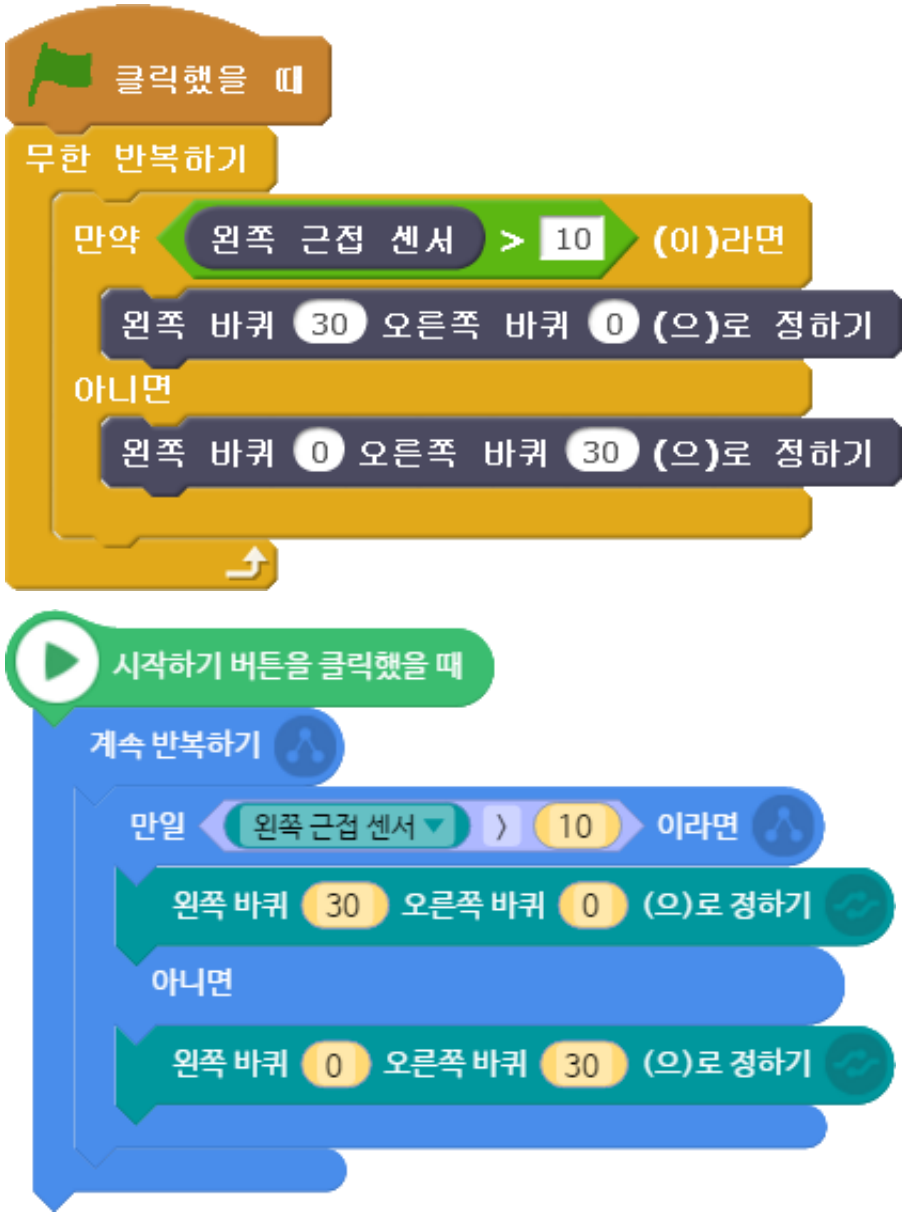
컵 따라 돌기







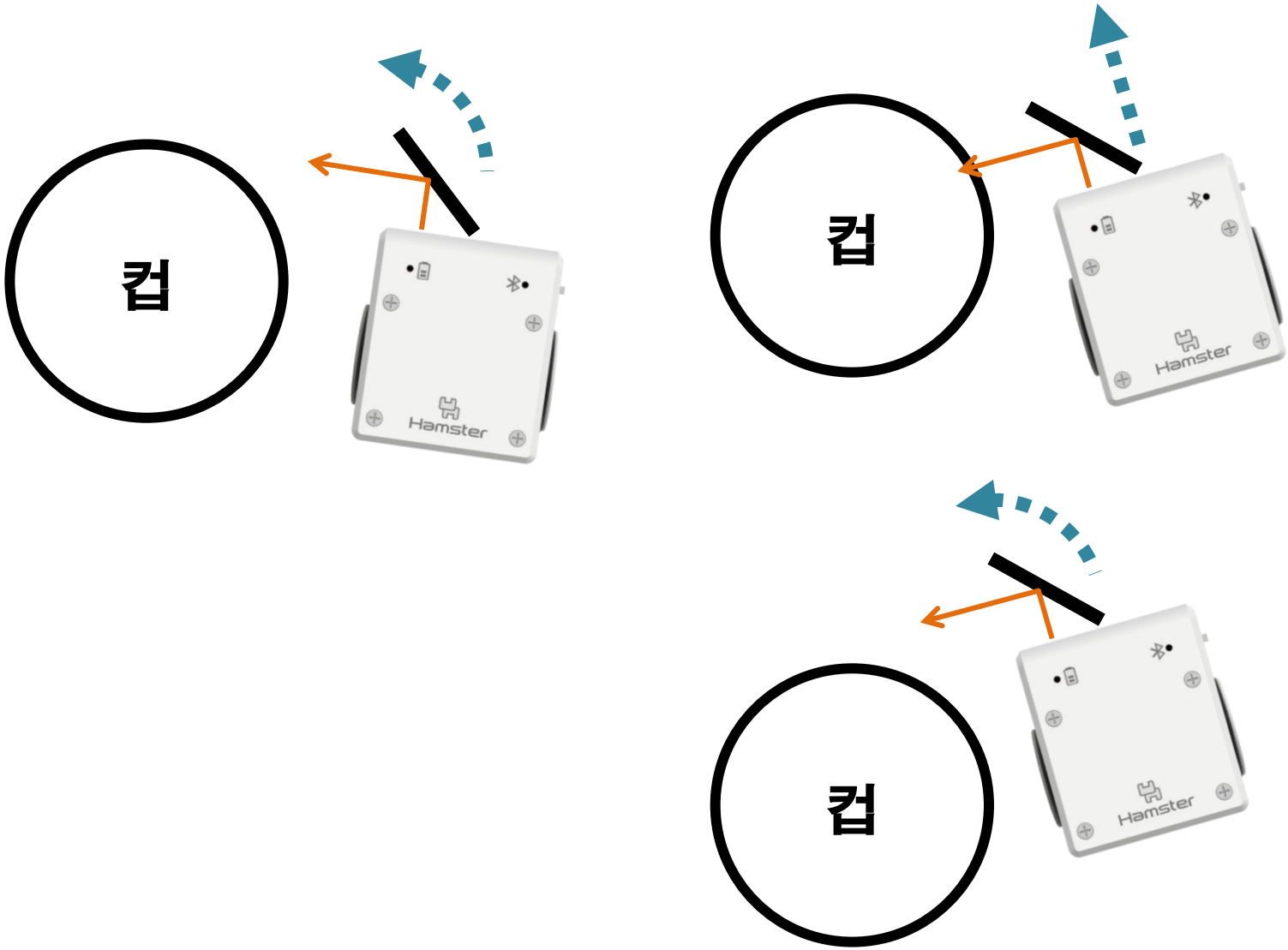
컵 따라 돌기



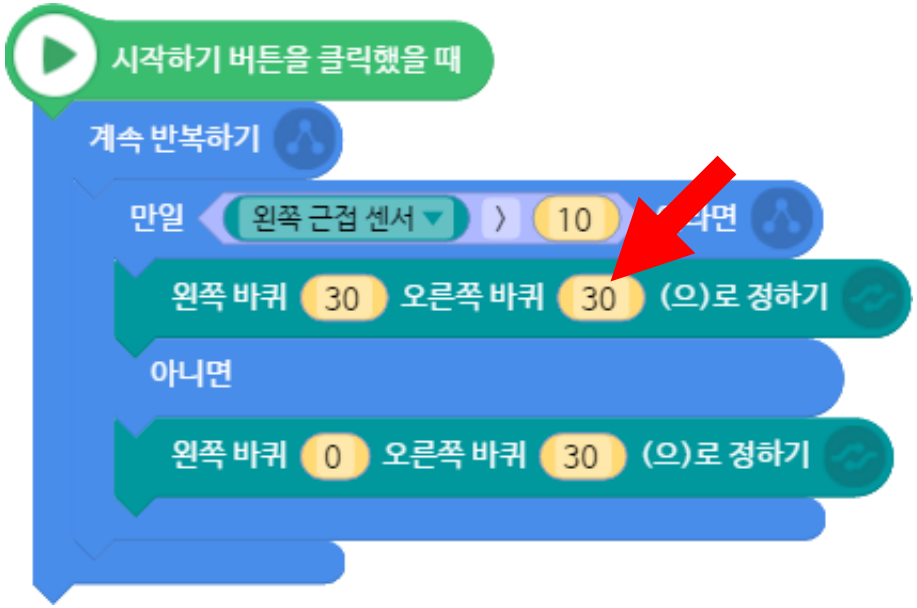

```python
from roboid import *

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

```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    hamster_create();

    while(1) {
        if(hamster_left_proximity() > 10)
            hamster_wheels(30, 0);
        else
            hamster_wheels(0, 30);
        wait(10);
    }
    return 0;
}
```

개선된 컵 따라 돌기



개선된 컵 따라 돌기




```python
from roboid import *

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

```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    hamster_create();

    while(1) {
        if(hamster_left_proximity() > 10)
            hamster_wheels(30, 30);
        else
            hamster_wheels(0, 30);
        wait(10);
    }
    return 0;
}
```

좀 더 개선된 컵 따라 돌기

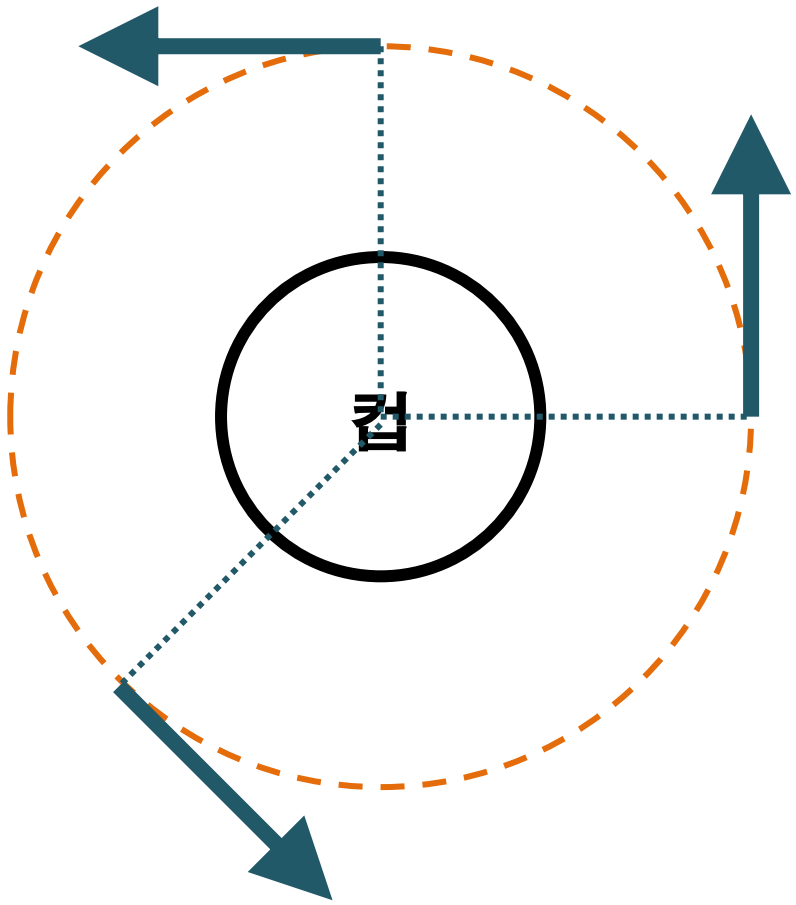
근접 센서 값이 커지면
바퀴의 속도도 커진다 ?

좀 더 개선된 컵 따라 돌기


```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    hamster_create();

    while(1) {
        hamster_wheels(hamster_left_proximity(), 30);
        wait(10);
    }
    return 0;
}
```

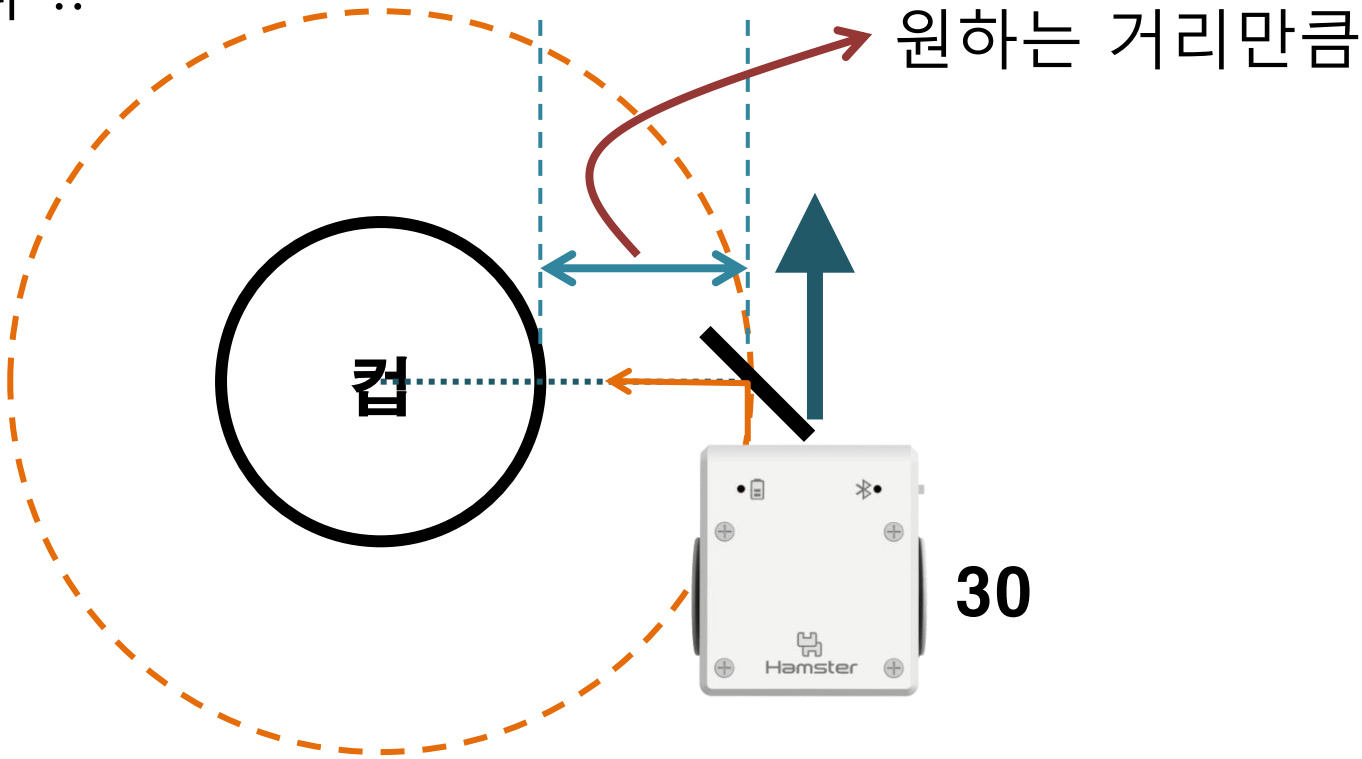
```python
from roboid import *

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

원 운동



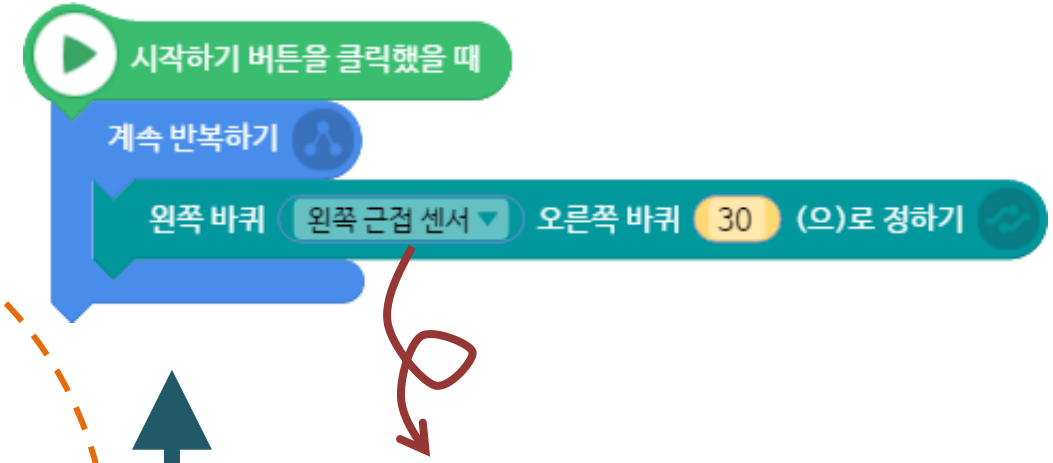
원하는 거리만큼 떨어져서 돌기

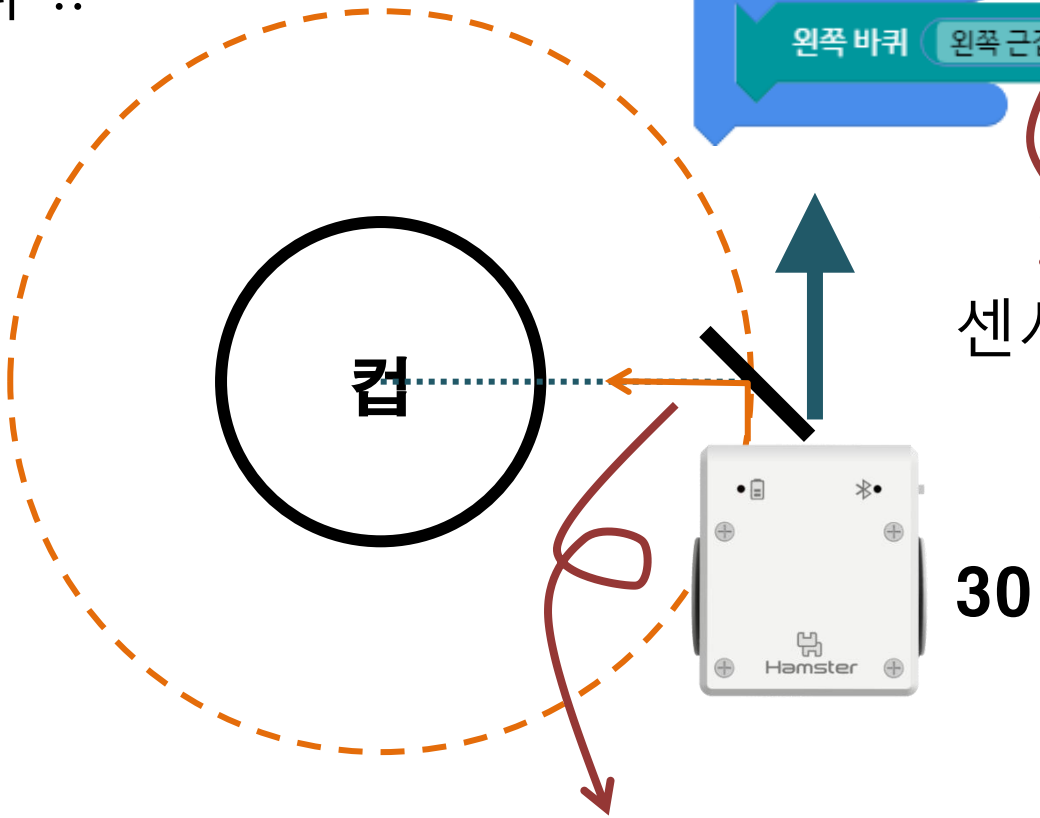
센서 위치를 컵의 중심에
맞춘다 !!



원하는 거리만큼 떨어져서 돌기

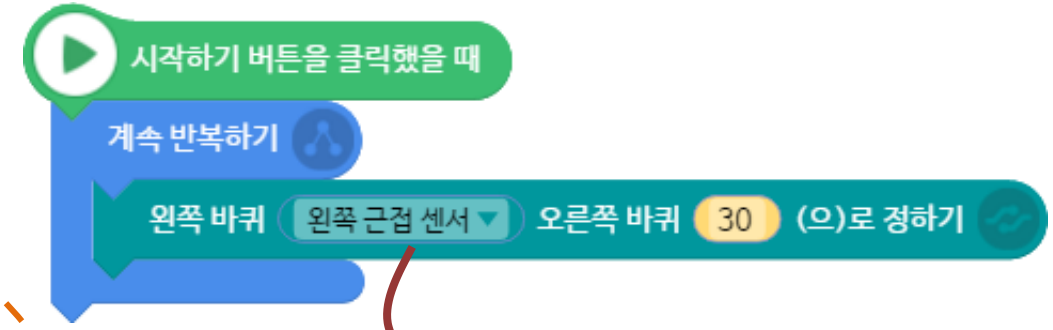
센서 위치를 컵의 중심에 맞춘다 !!

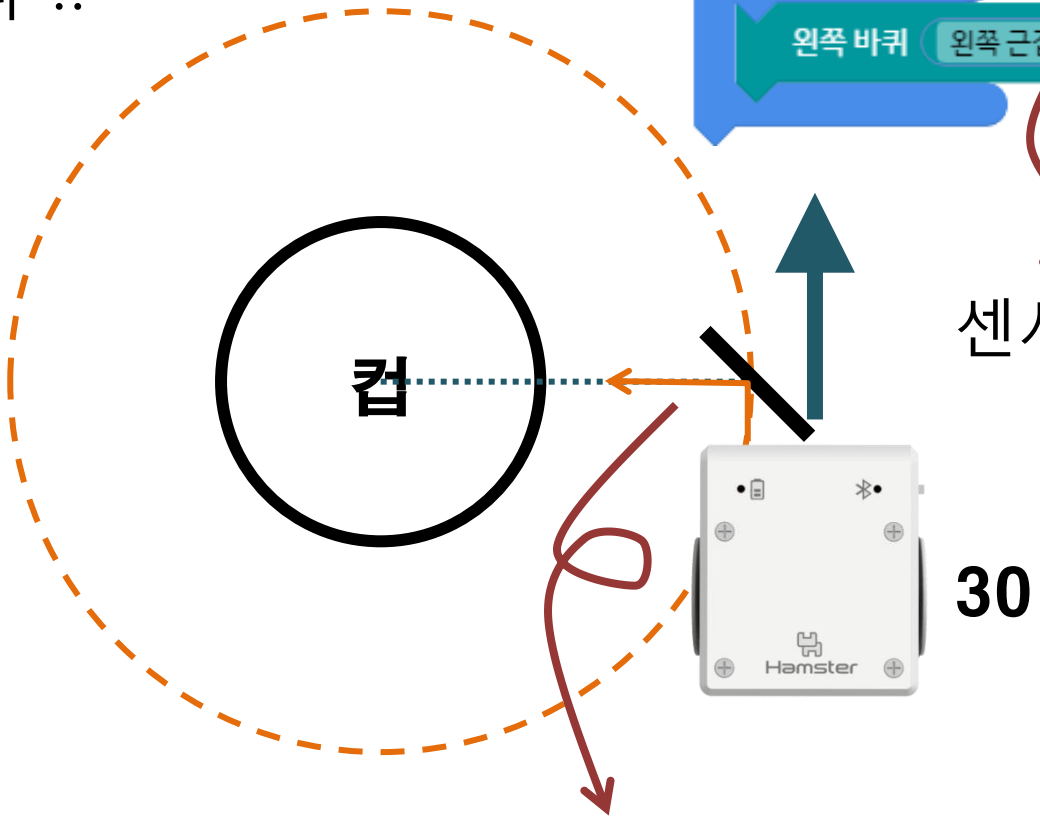




원하는 거리만큼 떨어져서 돌기

센서 위치를 컵의 중심에 맞춘다 !!





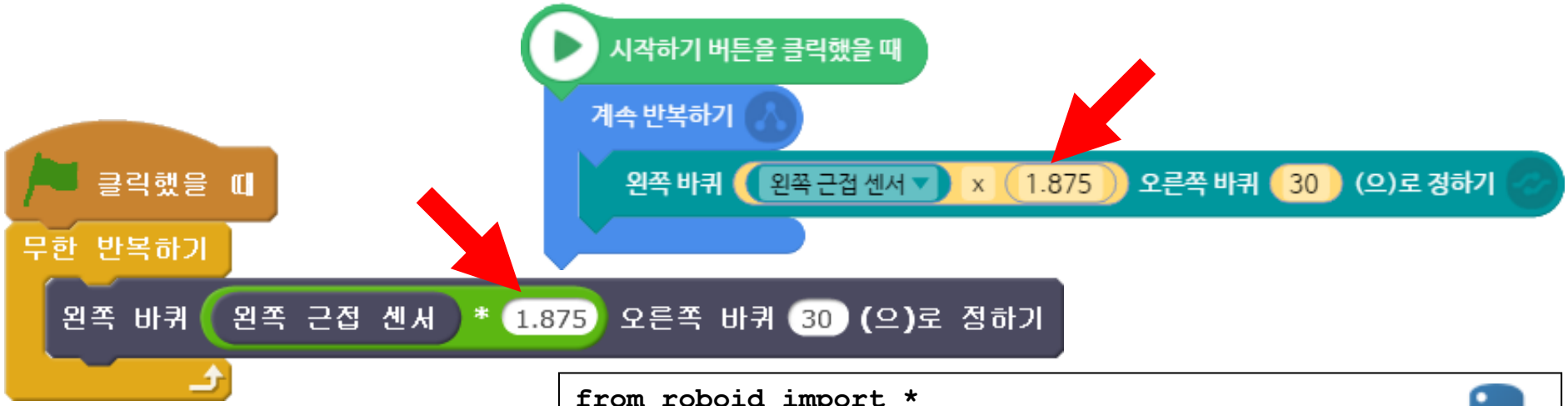
센서 값: 16

30

센서 값: 16

30 ÷ 16 = 1.875

원하는 거리만큼 떨어져서 돌기

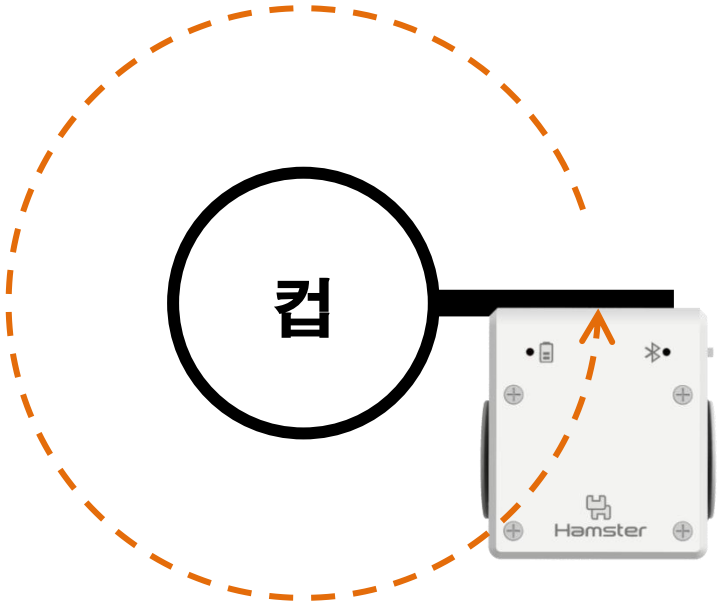



```python
from roboid import *

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

```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    hamster_create();

    while(1) {
        hamster_wheels(hamster_left_proximity() * 1.875, 30);
        wait(10);
    }
    return 0;
}
```

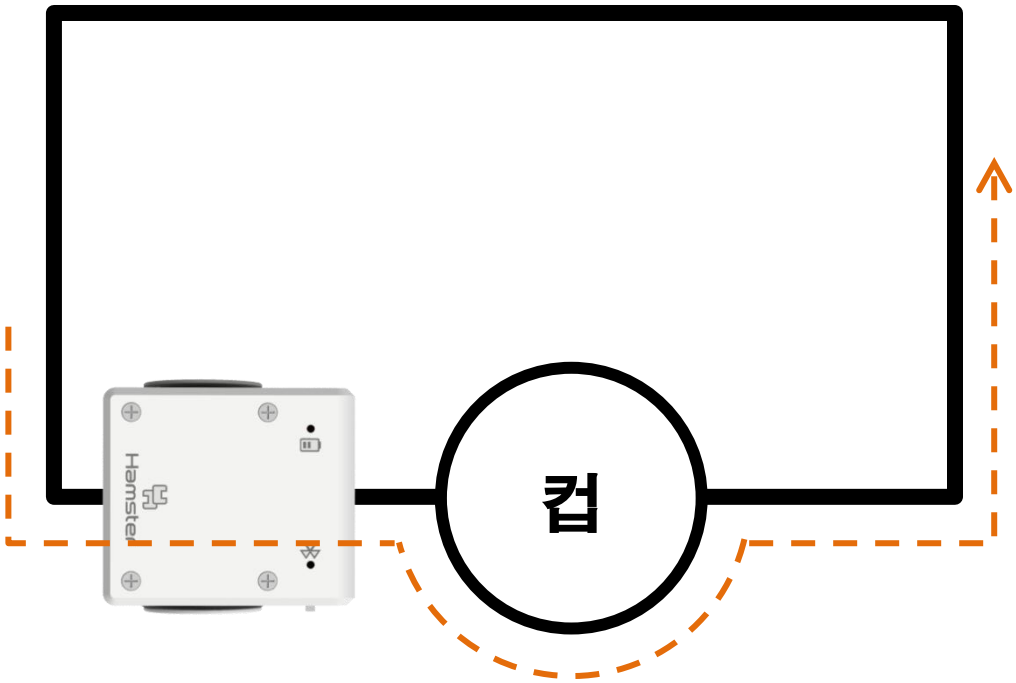
여러 가지 활동들

일정 횟수만큼 돌고 정지하기 중급



고급

라인 트레이싱 + 컵 따라 돌기



여러 가지 활동들

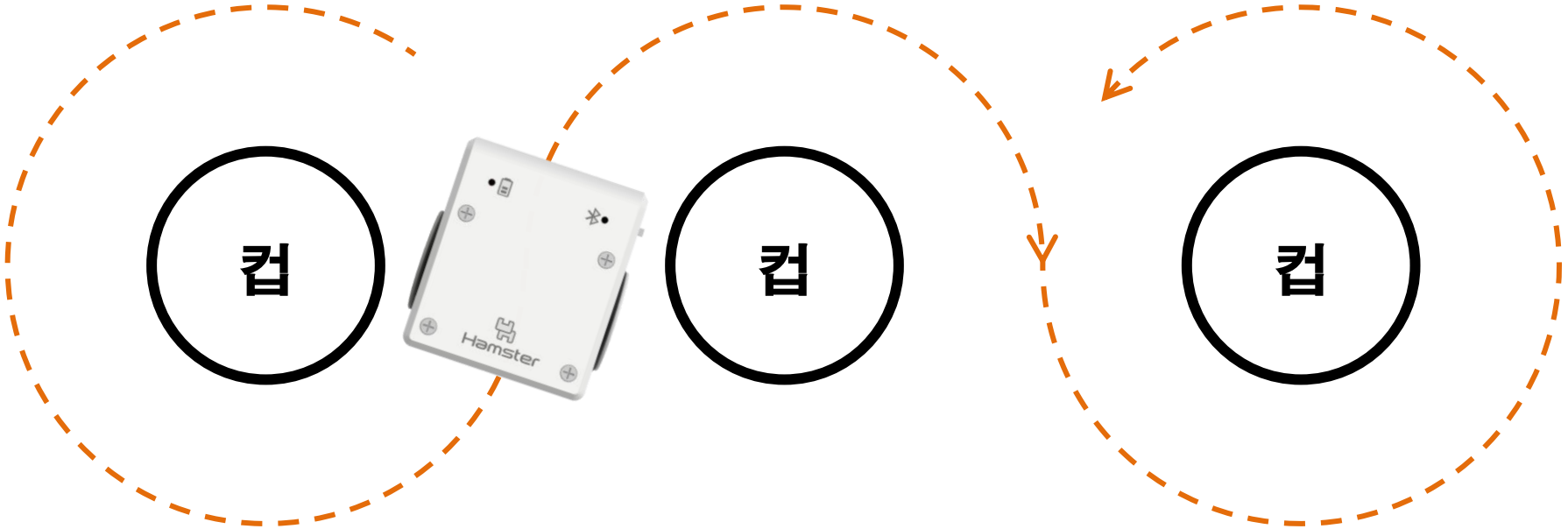
햄스터 플래닛

중급
고급

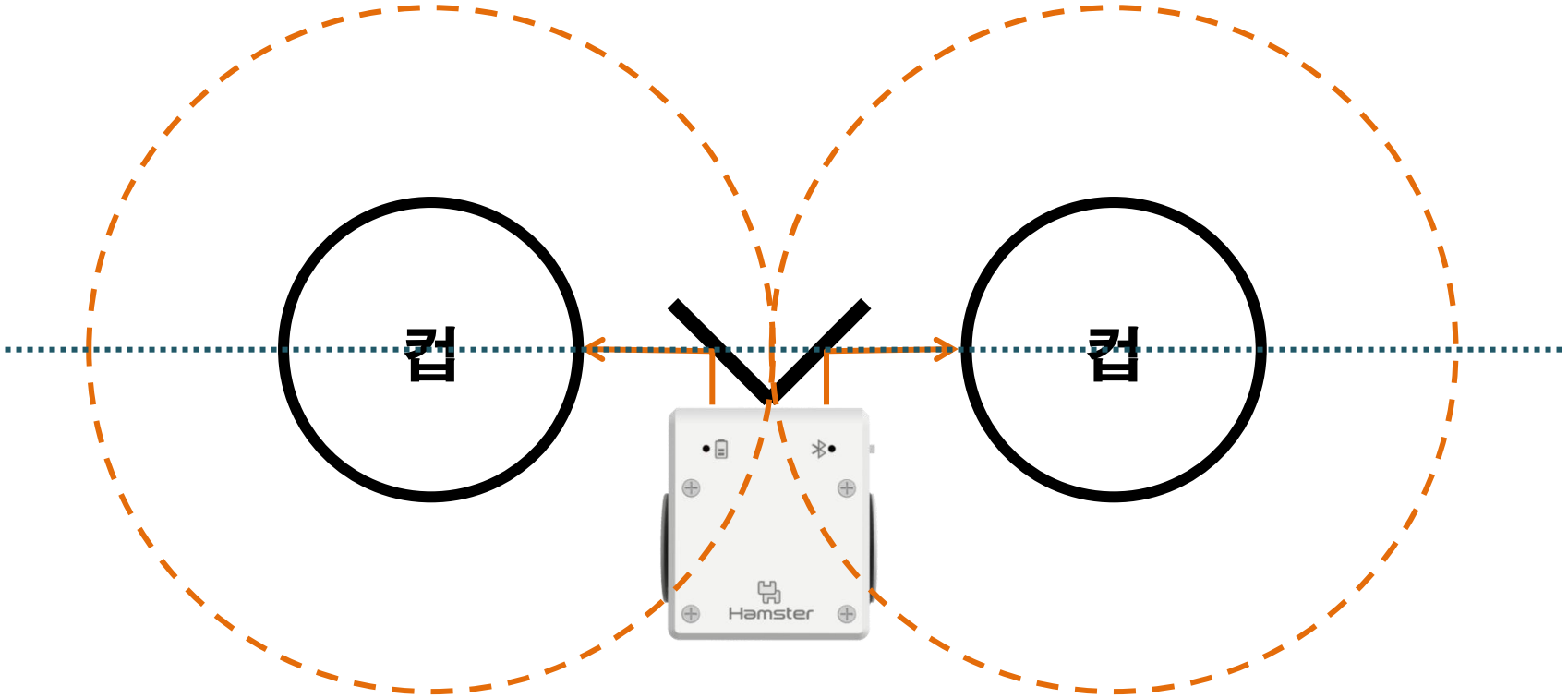
컵

중급

슬라럼 경주

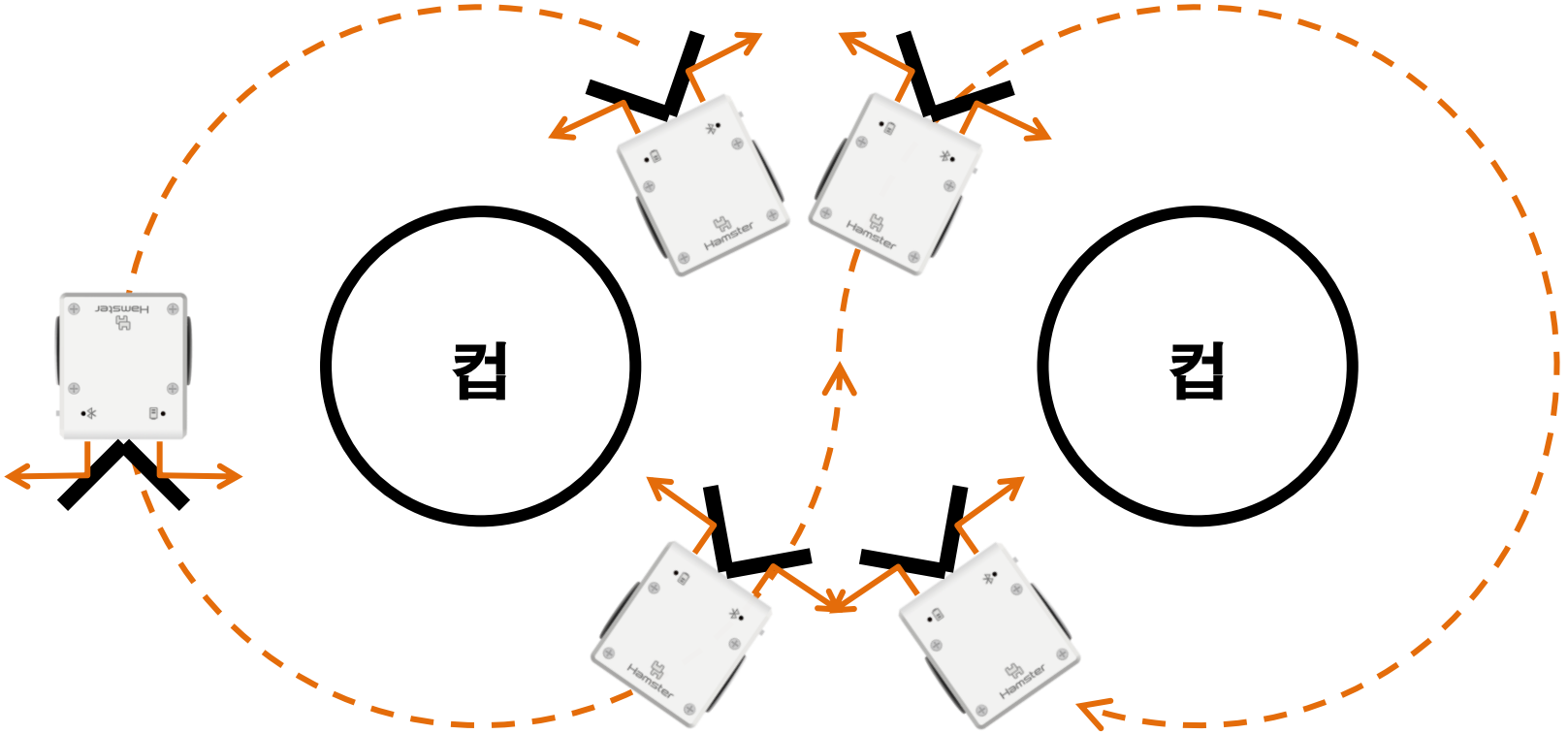


슬라럼 경주

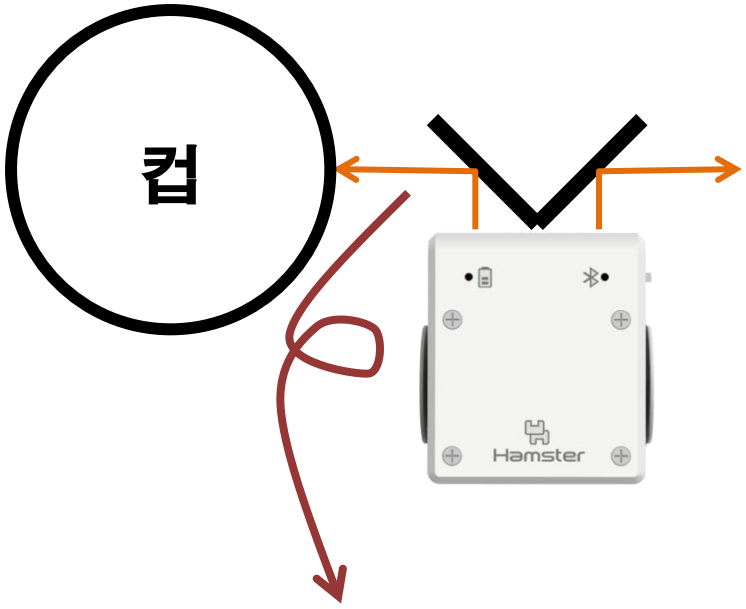


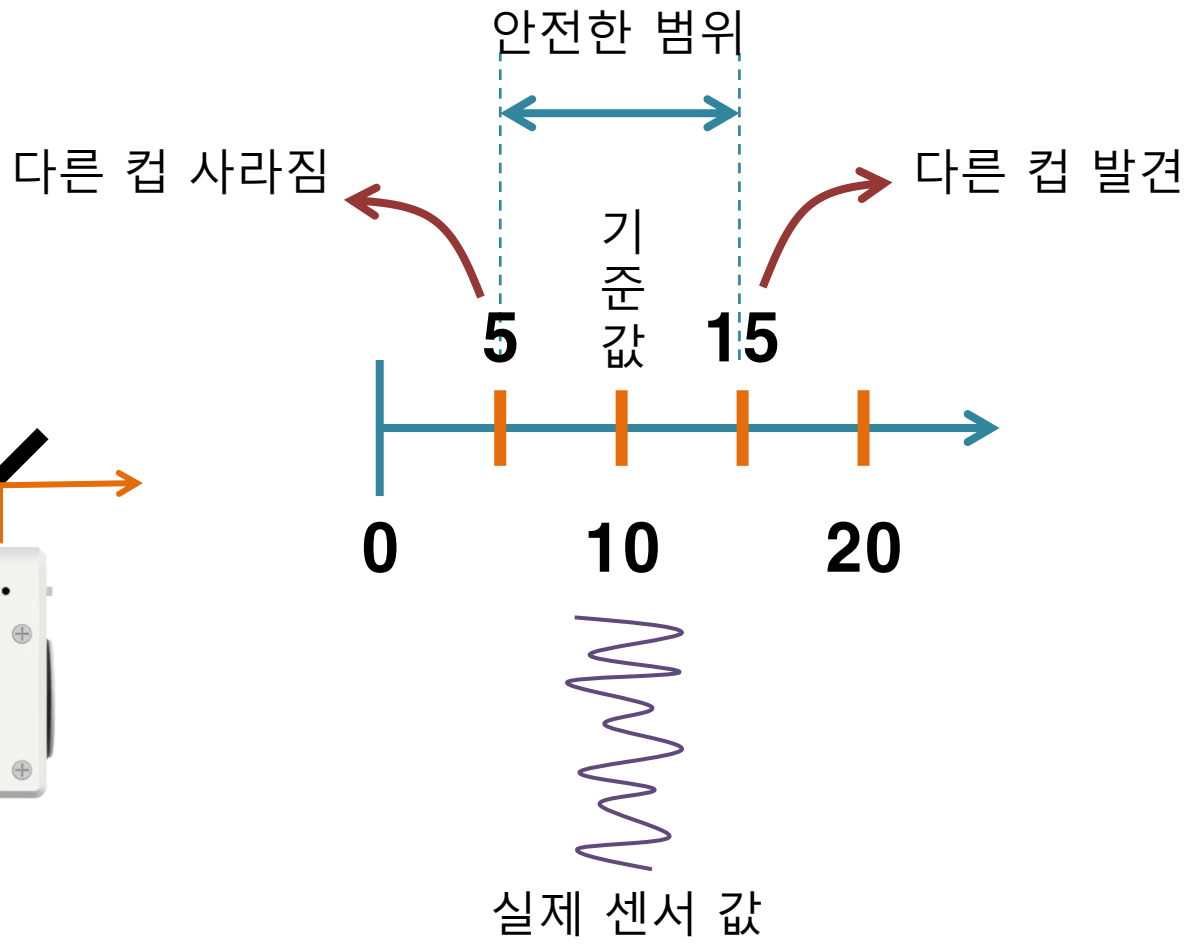
컵 중간으로 지나가게 하는 방법은?

슬라럼 경주

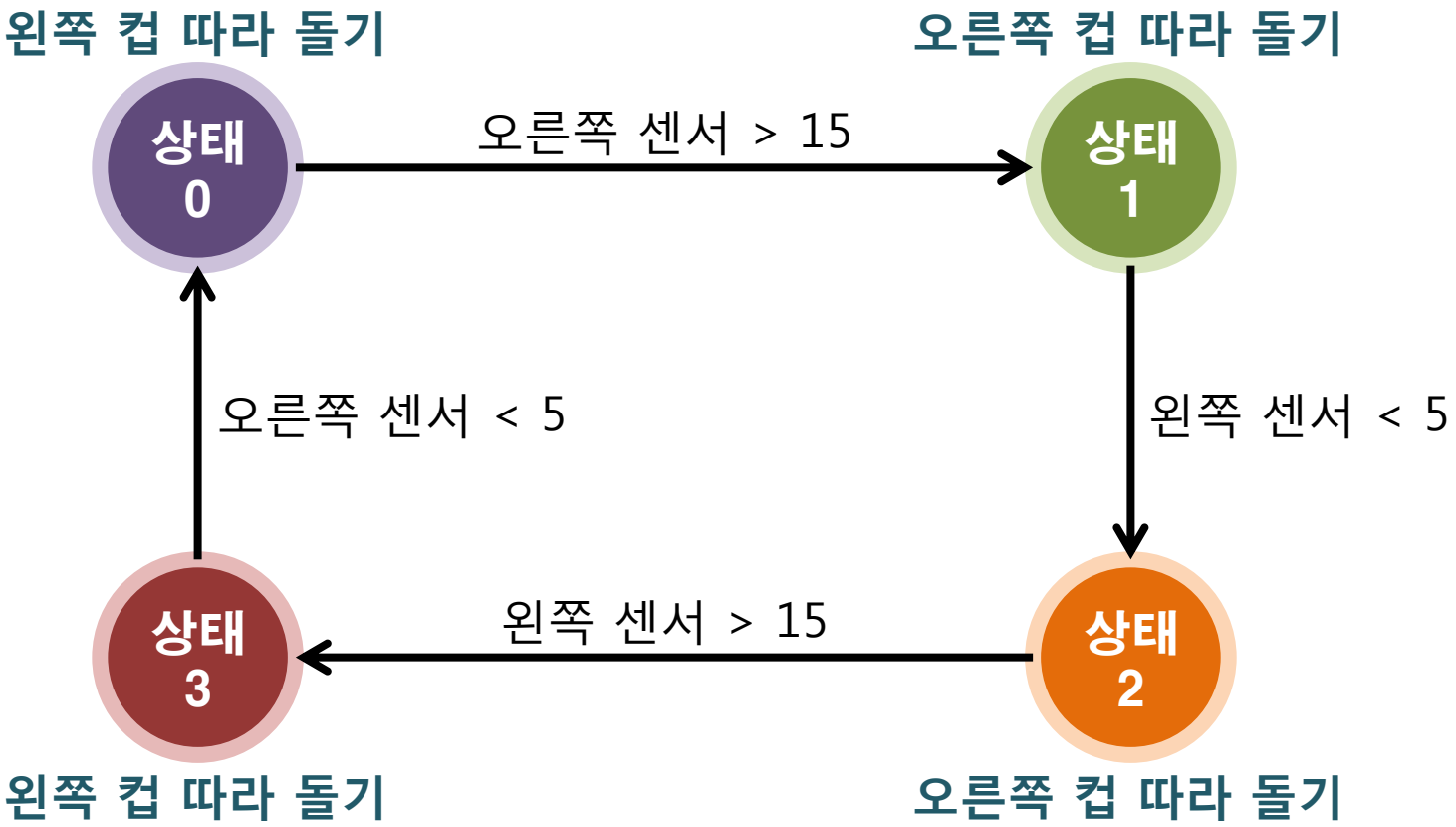

4 오른쪽 컵이 사라질 때까지
왼쪽 컵 따라 돌기
2 왼쪽 컵이 사라질 때까지
오른쪽 컵 따라 돌기
컵
컵
1 오른쪽 컵을 발견할 때까지
왼쪽 컵 따라 돌기
3 왼쪽 컵을 발견할 때까지
오른쪽 컵 따라 돌기

판단 기준



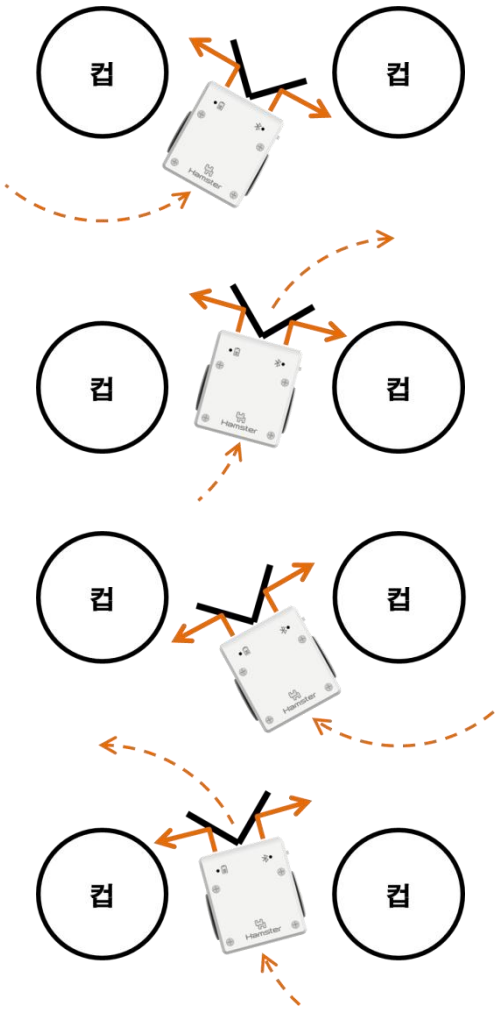


다이어그램


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

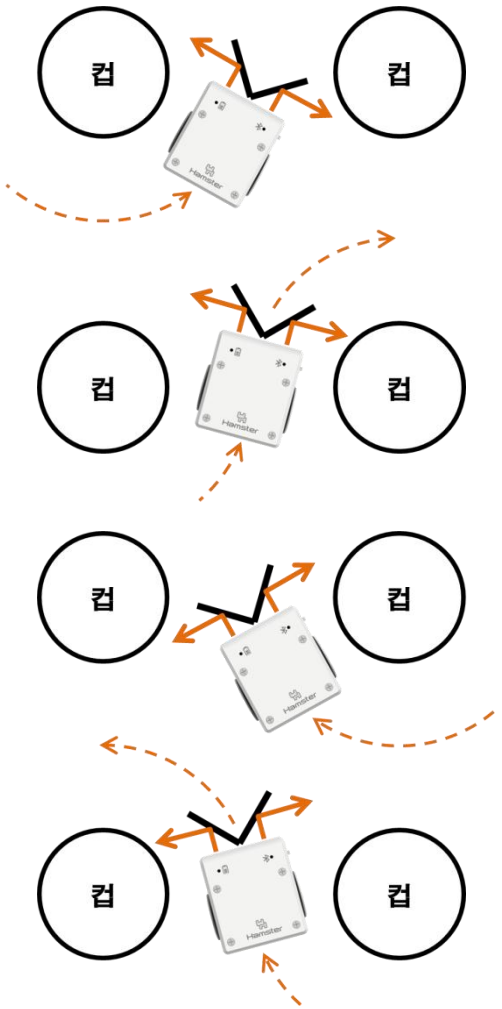
잘 안 되는 코드... 왜?


클릭했을 때
무한 반복하기
오른쪽 근접 센서 > 15 까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 * 2.5 오른쪽 바퀴 50 (으)로 정하기
왼쪽 근접 센서 < 5 까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 * 2.5 (으)로 정하기
왼쪽 근접 센서 > 15 까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 * 2.5 (으)로 정하기
오른쪽 근접 센서 < 5 까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 * 2.5 오른쪽 바퀴 50 (으)로 정하기


컵
컵
컵
컵
컵
컵
컵
컵

잘 안 되는 코드... 왜?


시작하기 버튼을 클릭했을 때
계속 반복하기
오른쪽 근접 센서 〉 15 이 될 때까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 x 2.5 오른쪽 바퀴 50 (으)로 정하기
왼쪽 근접 센서 〈 5 이 될 때까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 x 2.5 (으)로 정하기
왼쪽 근접 센서 〉 15 이 될 때까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 x 2.5 (으)로 정하기
오른쪽 근접 센서 〈 5 이 될 때까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 x 2.5 오른쪽 바퀴 50 (으)로 정하기


컵
컵
컵
컵
컵
컵
컵
컵

잘 안 되는 코드... 왜?

```python
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)
```

```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    int state = 0, left, right;

    hamster_create();

    while(1) {
        left = hamster_left_proximity();
        right = hamster_right_proximity();
        switch(state) {
            case 0:
                hamster_wheels(left * 2.5, 50);
                if(right > 15) state = 1;
                break;
            case 1:
                hamster_wheels(50, right * 2.5);
                if(left < 5) state = 2;
                break;
            case 2:
                hamster_wheels(50, right * 2.5);
                if(left > 15) state = 3;
                break;
            case 3:
                hamster_wheels(left * 2.5, 50);
                if(right < 5) state = 0;
                break;
        }
        wait(10);
    }
    return 0;
}
```

순간적으로 튀는 센서 값 제거하기

10번 연속으로 조건 만족하면...

순간적으로 튀는 센서 값 제거하기

10번 연속으로 조건 만족하면...

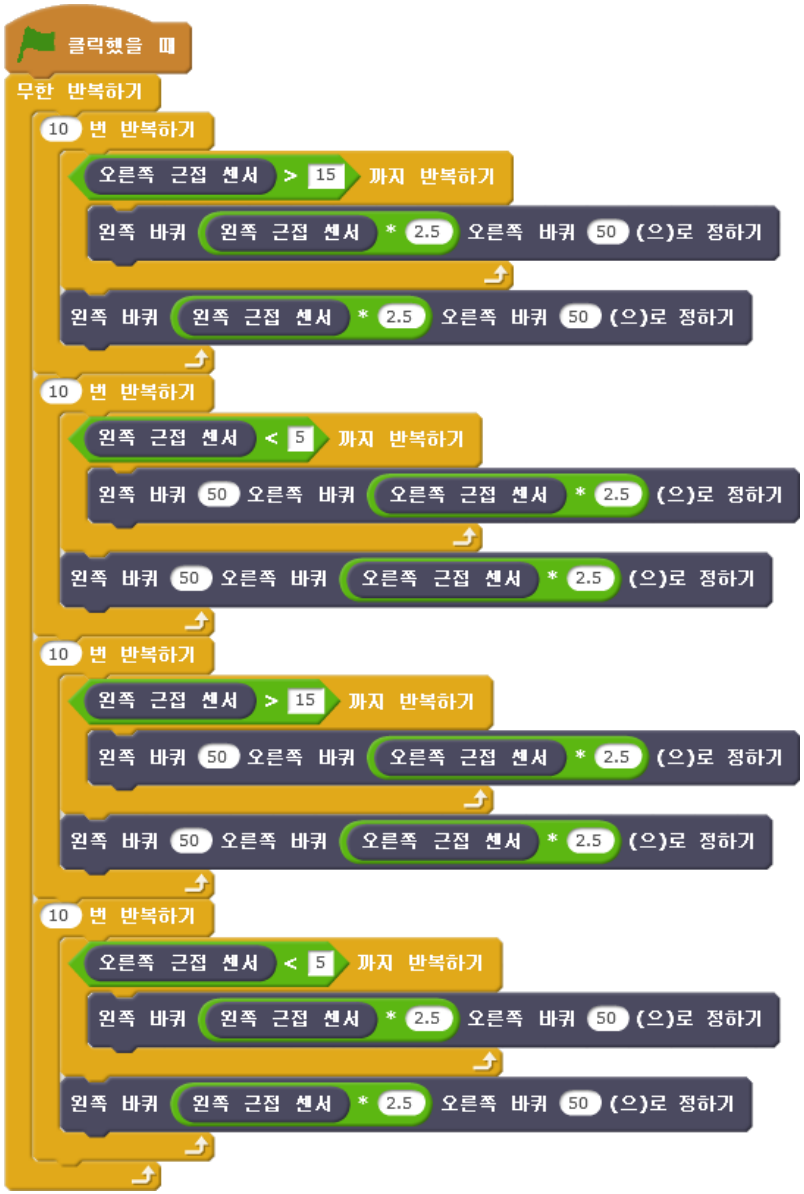
다르게 작성해 보기


클릭했을 때
상태0 방송하기
상태0 을(를) 받았을 때
횟수 을(를) 0 로 정하기
횟수 > 10 까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 * 2.5 오른쪽 바퀴 50 (으)로 정하기
만약 오른쪽 근접 센서 > 15 (이)라면
횟수 을(를) 1 만큼 바꾸기
아니면
횟수 을(를) 0 로 정하기
상태1 방송하기
상태1 을(를) 받았을 때
횟수 을(를) 0 로 정하기
횟수 > 10 까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 * 2.5 (으)로 정하기
만약 왼쪽 근접 센서 < 5 (이)라면
횟수 을(를) 1 만큼 바꾸기
아니면
횟수 을(를) 0 로 정하기
상태2 방송하기
상태3 을(를) 받았을 때
횟수 을(를) 0 로 정하기
횟수 > 10 까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 * 2.5 오른쪽 바퀴 50 (으)로 정하기
만약 오른쪽 근접 센서 < 5 (이)라면
횟수 을(를) 1 만큼 바꾸기
아니면
횟수 을(를) 0 로 정하기
상태0 방송하기
상태2 을(를) 받았을 때
횟수 을(를) 0 로 정하기
횟수 > 10 까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 * 2.5 (으)로 정하기
만약 왼쪽 근접 센서 > 15 (이)라면
횟수 을(를) 1 만큼 바꾸기
아니면
횟수 을(를) 0 로 정하기
상태3 방송하기

다르게 작성해 보기


시작하기 버튼을 클릭했을 때
상태0 신호 보내기
상태0 신호를 받았을 때
회수 를 0 로 정하기
회수 값 > 10 이 될 때까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 x 2.5 오른쪽 바퀴 50 (으)로 정하기
만일 오른쪽 근접 센서 > 15 이라면
회수 에 1 만큼 더하기
아니면
회수 를 0 로 정하기
상태1 신호 보내기
상태1 신호를 받았을 때
회수 를 0 로 정하기
회수 값 > 10 이 될 때까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 x 2.5 (으)로 정하기
만일 왼쪽 근접 센서 < 5 이라면
회수 에 1 만큼 더하기
아니면
회수 를 0 로 정하기
상태2 신호 보내기
상태3 신호를 받았을 때
회수 를 0 로 정하기
회수 값 > 10 이 될 때까지 반복하기
왼쪽 바퀴 왼쪽 근접 센서 x 2.5 오른쪽 바퀴 50 (으)로 정하기
만일 오른쪽 근접 센서 < 5 이라면
회수 에 1 만큼 더하기
아니면
회수 를 0 로 정하기
상태0 신호 보내기
상태2 신호를 받았을 때
회수 를 0 로 정하기
회수 값 > 10 이 될 때까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 오른쪽 근접 센서 x 2.5 (으)로 정하기
만일 왼쪽 근접 센서 > 15 이라면
회수 에 1 만큼 더하기
아니면
회수 를 0 로 정하기
상태3 신호 보내기

2% 부족한... 그러나 좀 더 간단한



2% 부족한... 그러나 좀 더 간단한

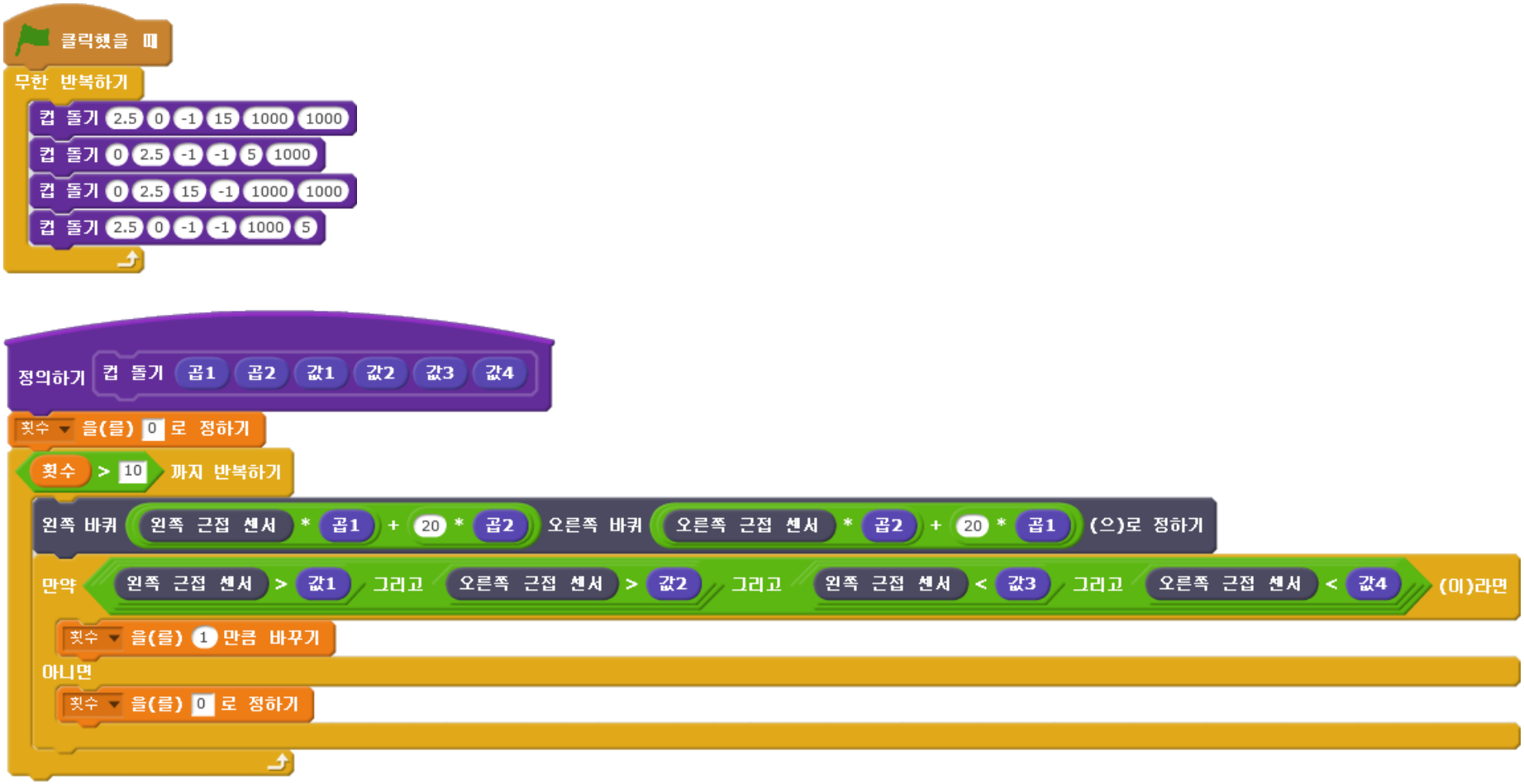
순간적으로 튀는 센서 값 제거하기

```
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, 50)
        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)
```

이렇게 작성할 수도 있지만 좀 그러한...

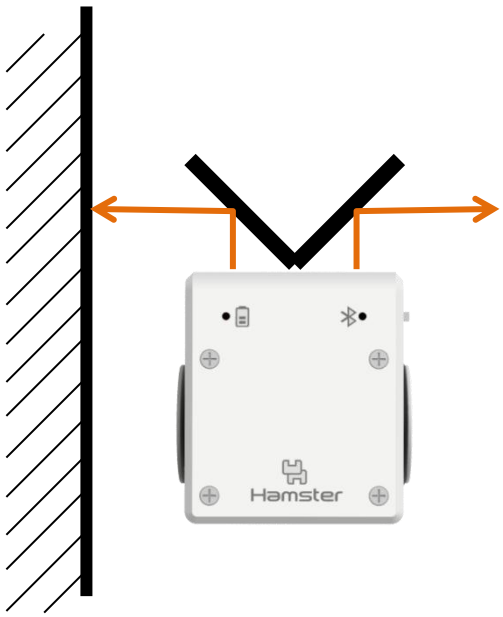


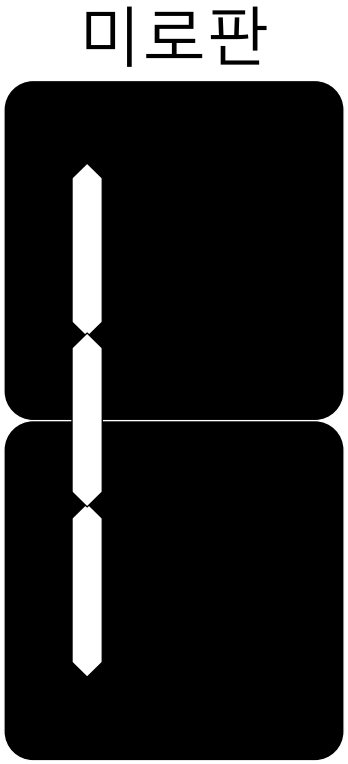
순간적으로 튀는 센서 값 제거하기


```c
#include "roboid.h"

int main(int argc, char *argv[]) {
  int state = 0, count = 0, left, right;

  hamster_create();

  while(1) {
    left = hamster_left_proximity();
    right = hamster_right_proximity();
    switch(state) {
      case 0:
        hamster_wheels(left * 2.5, 50);
        if(right > 15) count ++;
        else count = 0;
        if(count > 10) {
          count = 0;
          state = 1;
        }
        break;
      case 1:
        hamster_wheels(50, right * 2.5);
        if(left < 5) count ++;
        else count = 0;
        if(count > 10) {
          count = 0;
          state = 2;
        }
        break;
      case 2:
        hamster_wheels(50, right * 2.5);
        if(left > 15) count ++;
        else count = 0;
        if(count > 10) {
          count = 0;
          state = 3;
        }
        break;
      case 3:
        hamster_wheels(left * 2.5, 50);
        if(right < 5) count ++;
        else count = 0;
        if(count > 10) {
          count = 0;
          state = 0;
        }
        break;
    }
    wait(10);
  }
  return 0;
}
```

미로 찾기

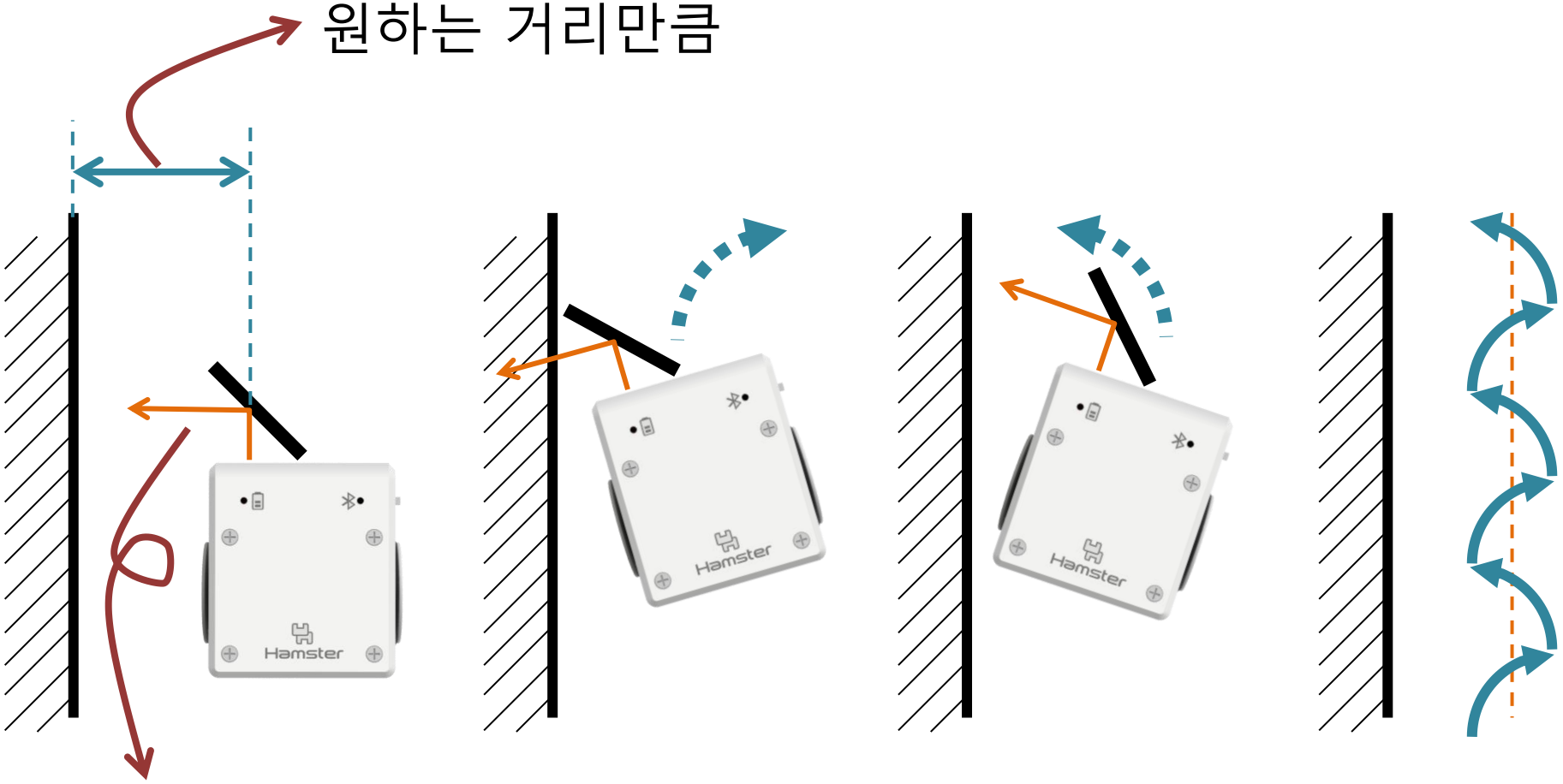
한 쪽 센서 사용하여 벽 따라 가기



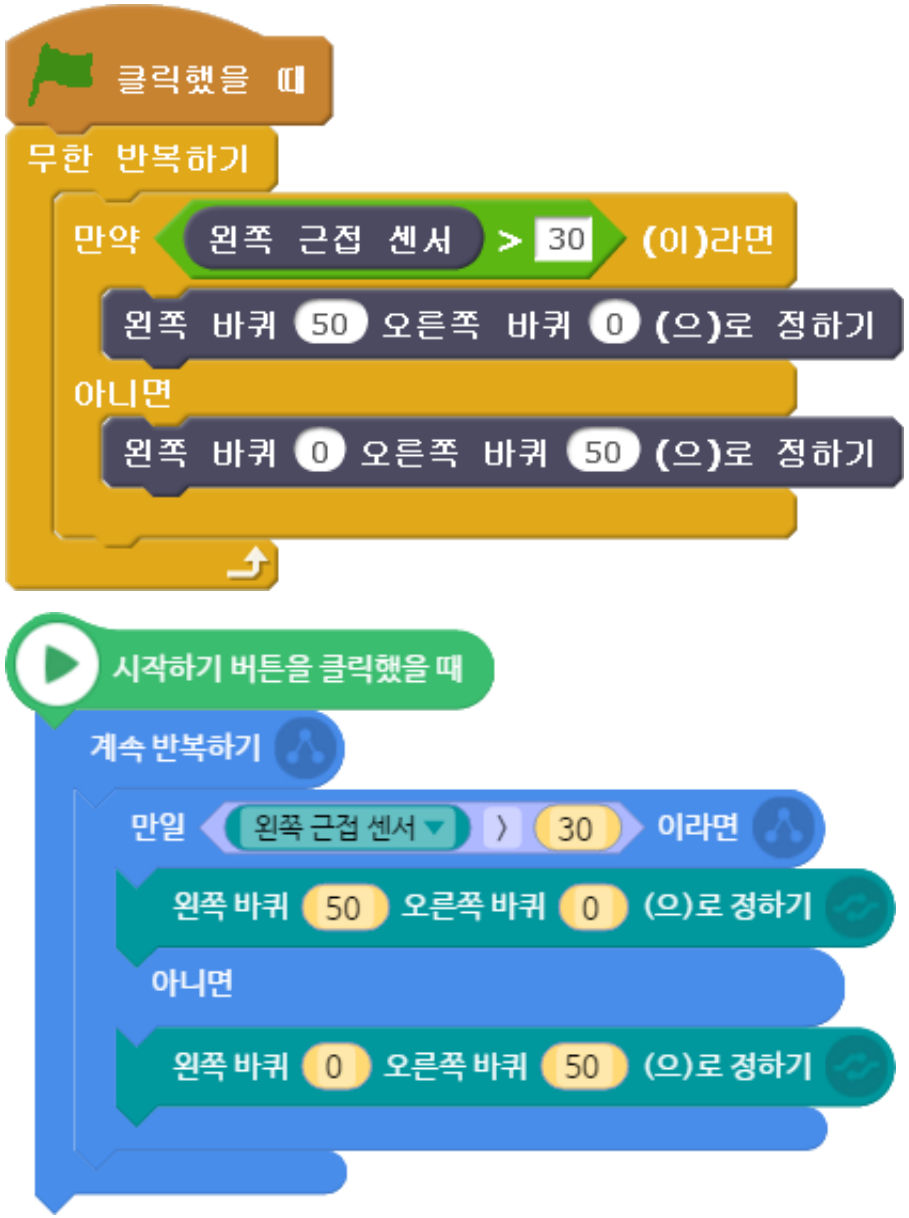


왼쪽 벽을 따라 주행

한 쪽 센서 사용하여 벽 따라 가기



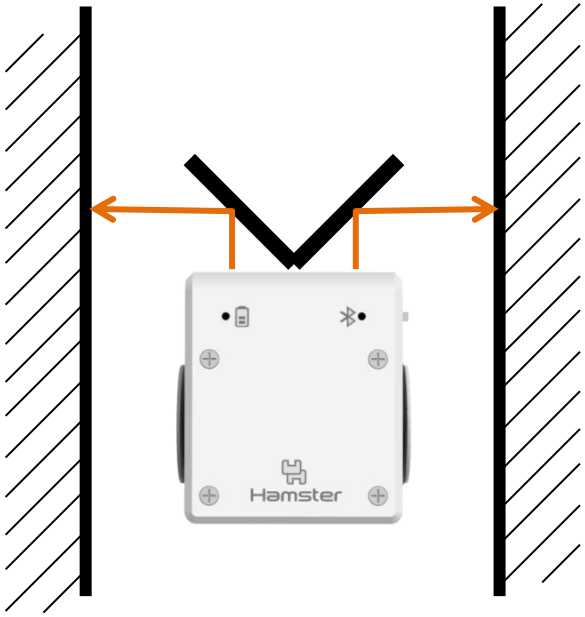
한 쪽 센서 사용하여 벽 따라 가기



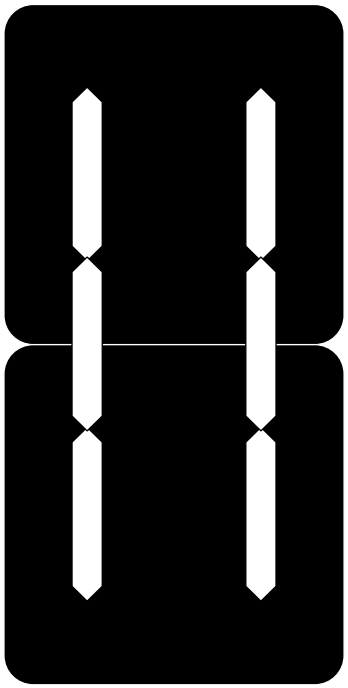

```python
from roboid import *

hamster = Hamster()
while True:
    if hamster.left_proximity() > 30:
        hamster.wheels(50, 0)
    else:
        hamster.wheels(0, 50)
    wait(10)
```

```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    hamster_create();

    while(1) {
        if(hamster_left_proximity() > 30)
            hamster_wheels(50, 0);
        else
            hamster_wheels(0, 50);
        wait(10);
    }
    return 0;
}
```

양쪽 센서 사용하여 복도 주행

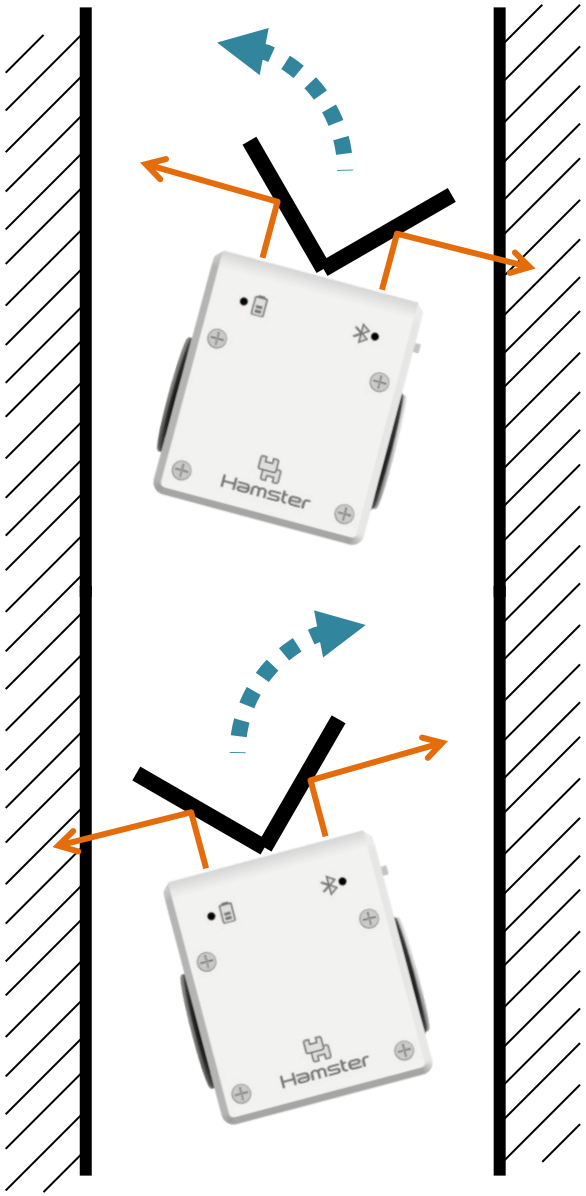


미로판



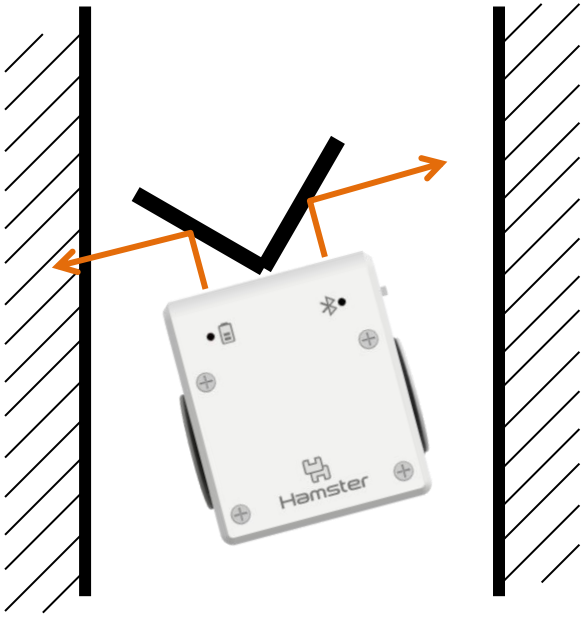
복도 중앙으로 주행

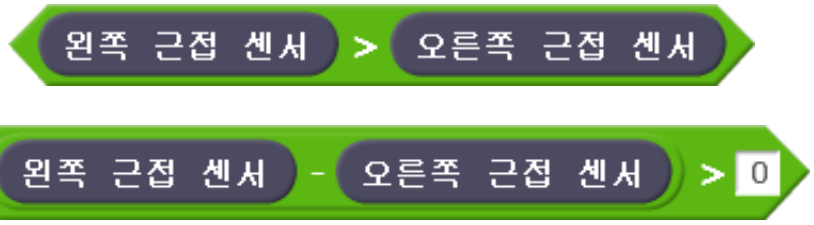
양쪽 센서 사용하여 복도 주행



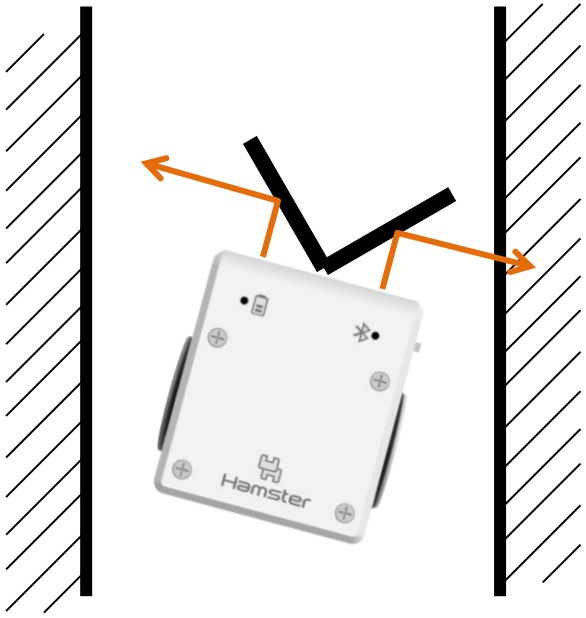
양쪽 센서 사용하여 복도 주행

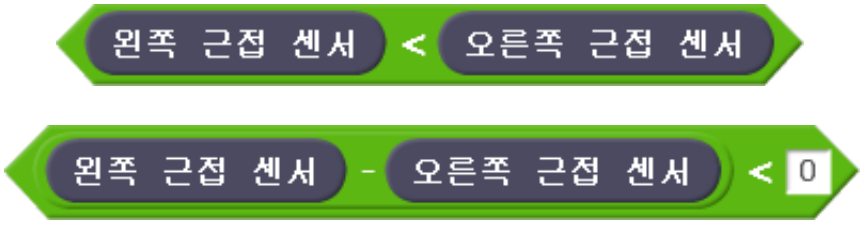
왼쪽 센서 값 > 오른쪽 센서 값



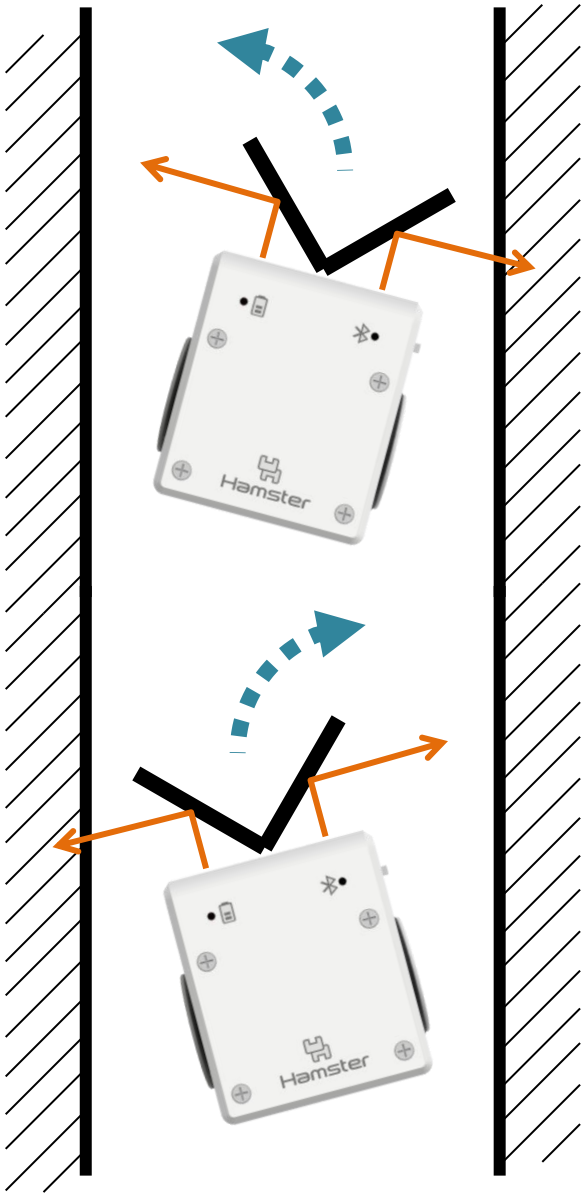


왼쪽 센서 값 < 오른쪽 센서 값



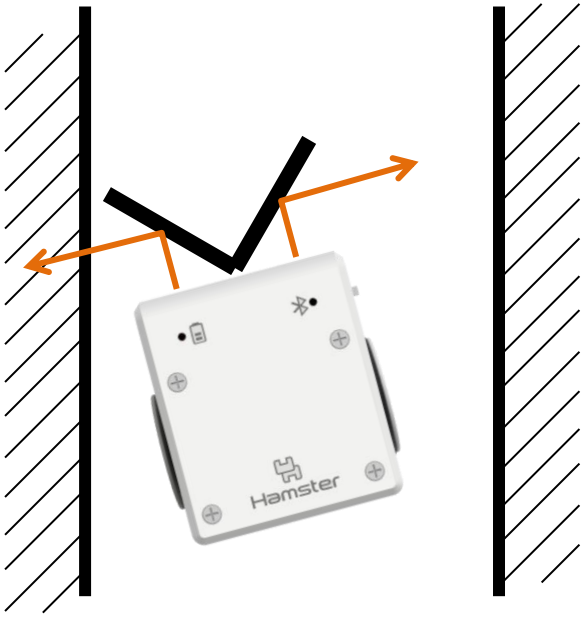


양쪽 센서 사용하여 복도 주행

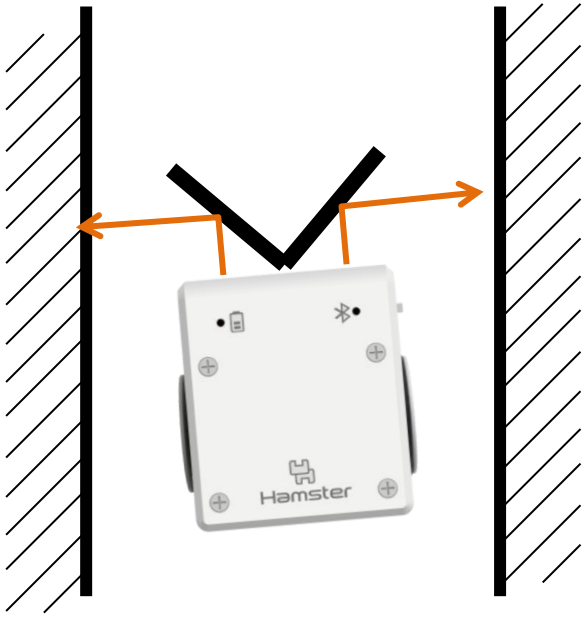


양쪽 센서 사용하여 복도 주행

오른쪽으로 **많이** 움직여야...



오른쪽으로 **조금** 움직이면...



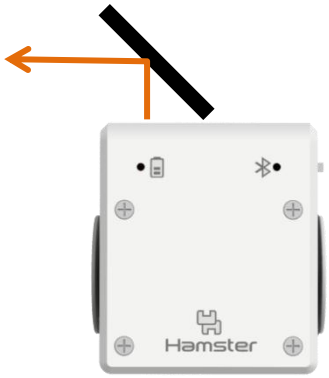
양쪽 센서 사용하여 복도 주행


```python
from roboid import *

hamster = Hamster()
while True:
    diff = hamster.left_proximity() - hamster.right_proximity()
    hamster.wheels(50 + diff * 1.3, 50 - diff * 1.3)
    wait(10)
```

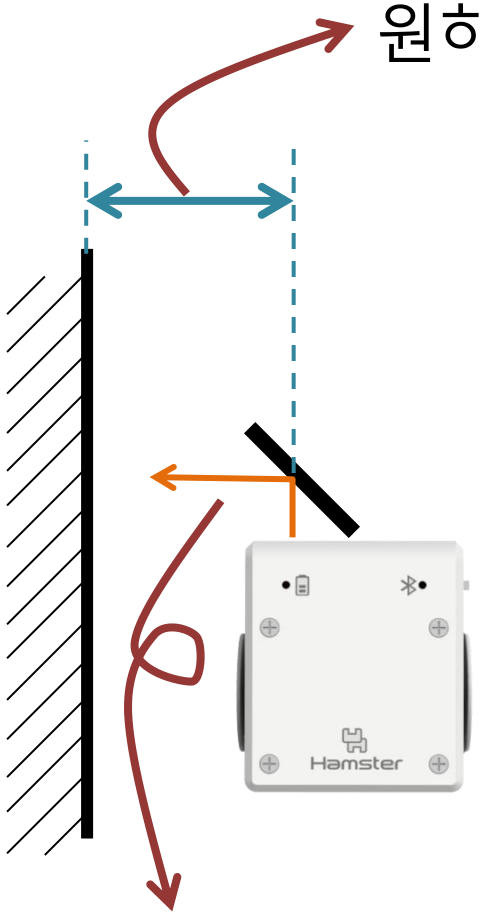
```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    int diff;
    hamster_create();

    while(1) {
        diff = hamster_left_proximity() - hamster_right_proximity()
        hamster_wheels(50 + diff * 1.3, 50 - diff * 1.3);
        wait(10);
    }
    return 0;
}
```

미로 찾기

왼쪽 반사판으로 교체해 주세요 !!



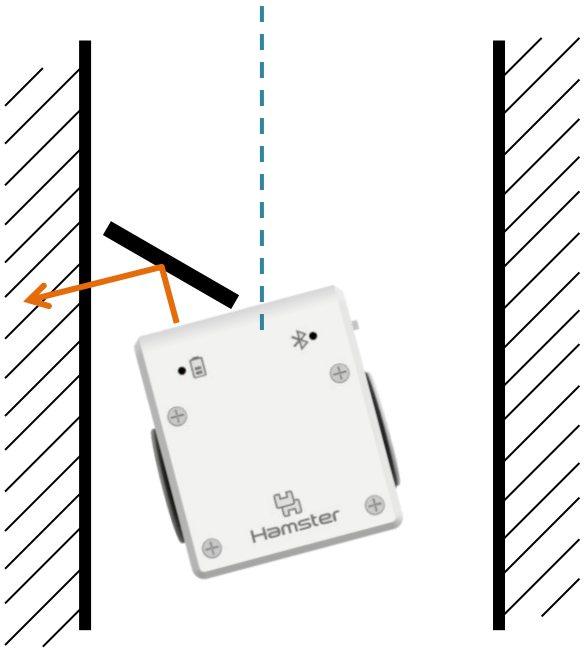
미로 찾기



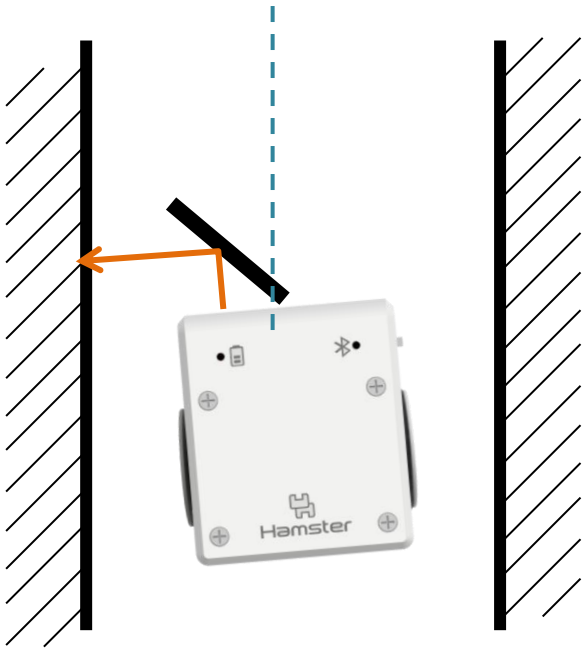
왼쪽 벽을 따라 주행

미로 찾기

오른쪽으로 **많이** 움직여야...

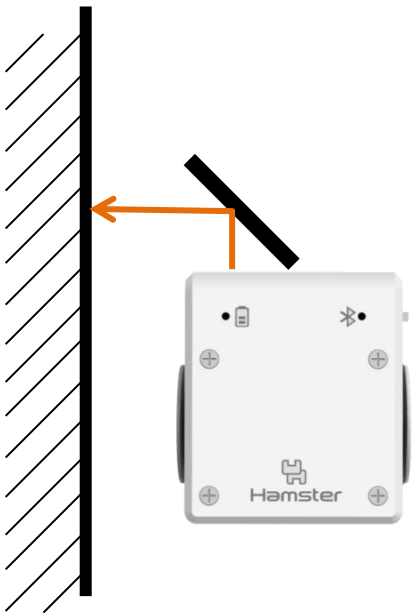


오른쪽으로 **조금** 움직이면...



미로 찾기

왼쪽 벽을 따라 주행

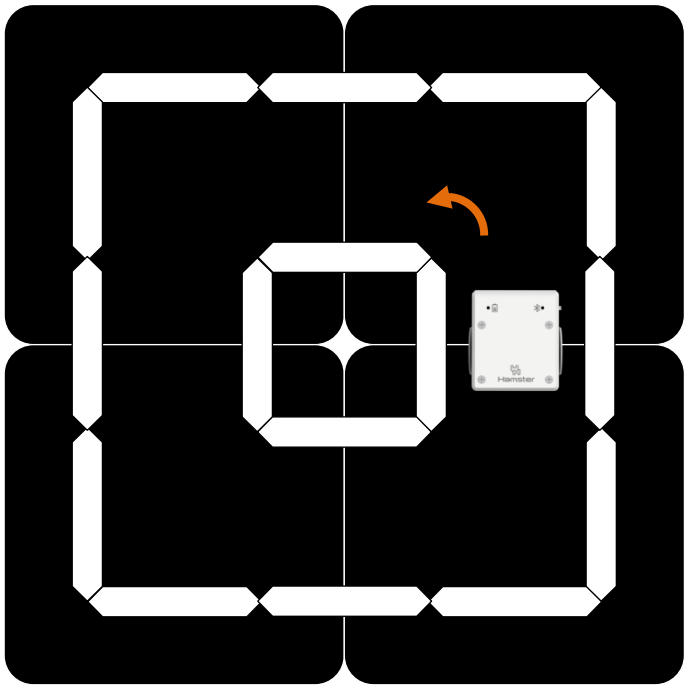



```python
from roboid import *

hamster = Hamster()
while True:
    diff = hamster.left_proximity() - 35
    hamster.wheels(50 + diff * 0.8, 50 - diff * 0.8)
    wait(10)
```

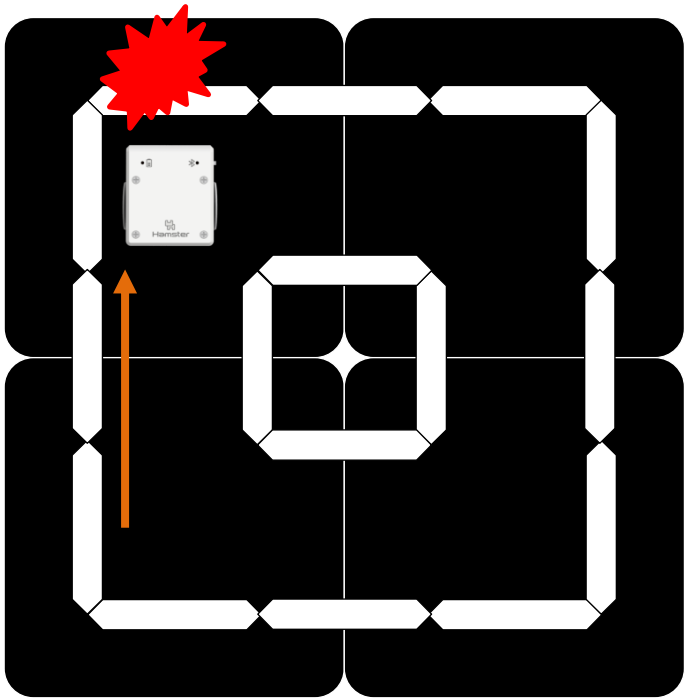
```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    int diff;
    hamster_create();

    while(1) {
        diff = hamster_left_proximity() - 35;
        hamster_wheels(50 + diff * 0.8, 50 - diff * 0.8);
        wait(10);
    }
    return 0;
}
```

모퉁이를 잘 도는가?

미로판



모퉁이를 잘 도는가?

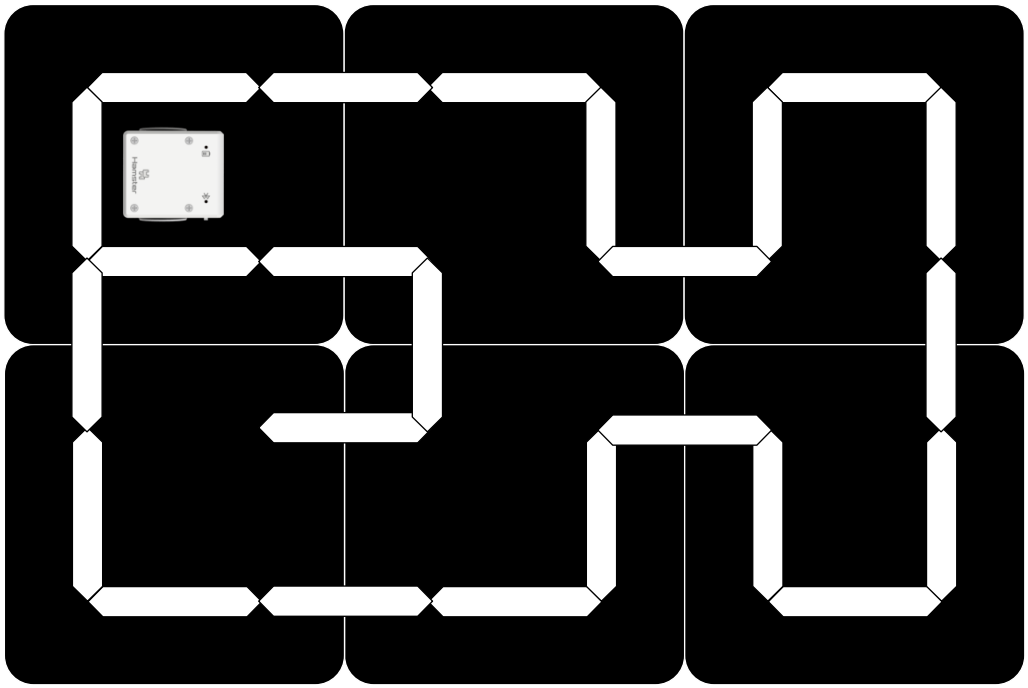


막힌 길에서 제자리 돌기

언제까지 돌아야 하나?

미로 찾기

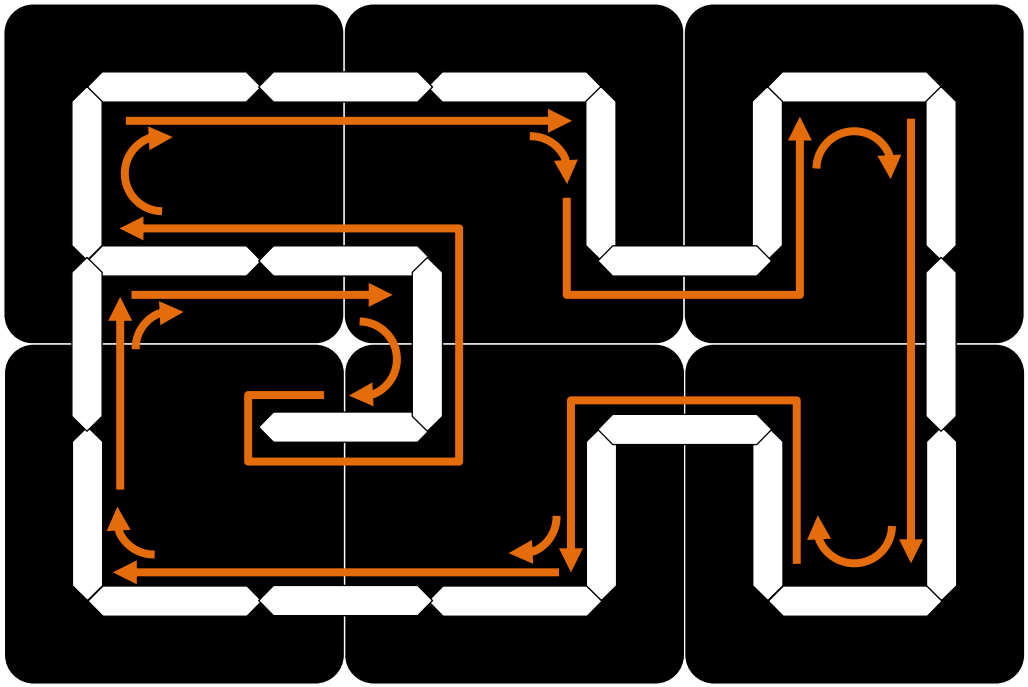
미로판



미로 찾기

- 좌수법 (좌선법): 왼쪽 → 앞쪽 → 오른쪽

- 왼쪽 벽을 따라 주행
- 앞쪽에 벽이 있으면 벽이 없을 때까지 오른쪽으로 회전



미로 찾기


클릭했을 때
무한 반복하기
만약 오른쪽 근접 센서 > 47 (이)라면
오른쪽 근접 센서 < 5 까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 -50 (으)로 정하기
아니면
왼쪽 바퀴 50 + 왼쪽 근접 센서 - 35 * 0.8 오른쪽 바퀴 50 - 왼쪽 근접 센서 - 35 * 0.8 (으)로 정하기


시작하기 버튼을 클릭했을 때
계속 반복하기
만일 오른쪽 근접 센서 〉 47 이라면
오른쪽 근접 센서 〈 5 이 될 때까지 반복하기
왼쪽 바퀴 50 오른쪽 바퀴 -50 (으)로 정하기
아니면
왼쪽 바퀴 50 + 왼쪽 근접 센서 - 35 x 0.8 오른쪽 바퀴 50 - 왼쪽 근접 센서 - 35 x 0.8 (으)로 정하기

미로 찾기

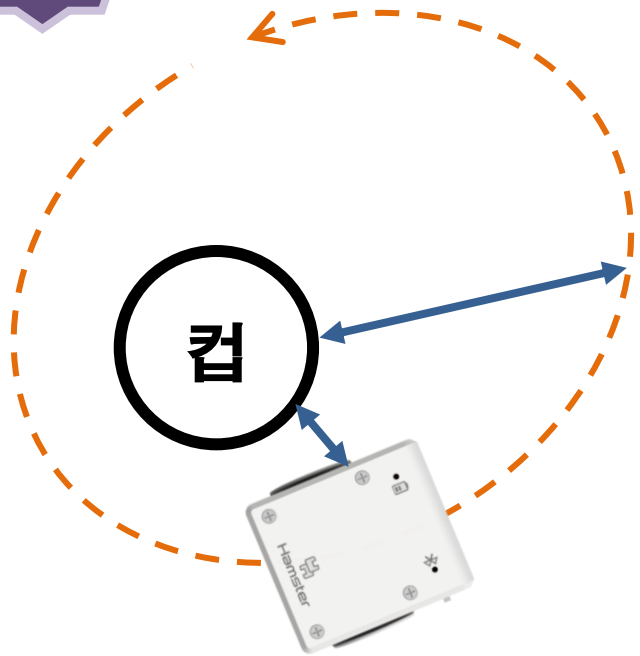

```python
from roboid import *

hamster = Hamster()
while True:
    left = hamster.left_proximity()
    right = hamster.right_proximity()
    if right > 47:
        while right > 5:
            hamster.wheels(50, -50)
            right = hamster.right_proximity()
            wait(10)
    else:
        diff = left - 35
        hamster.wheels(50 + diff * 0.8, 50 - diff * 0.8)
    wait(10)
```

```c
#include "roboid.h"

int main(int argc, char *argv[]) {
    int left, right, diff;
    hamster_create();

    while(1) {
        left = hamster_left_proximity();
        right = hamster_right_proximity();
        if(right > 47) {
            while(right > 5) {
                hamster_wheels(50, -50);
                right = hamster_right_proximity();
                wait(10);
            }
        } else {
            diff = left - 35;
            hamster_wheels(50 + diff * 0.8, 50 - diff * 0.8);
        }
        wait(10);
    }
    return 0;
}
```

생각해 봅시다

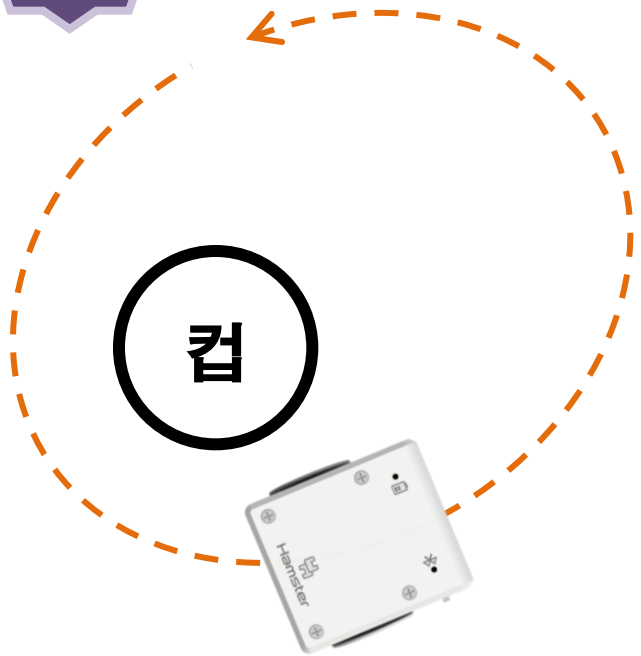
햄스터 플래닛

중급



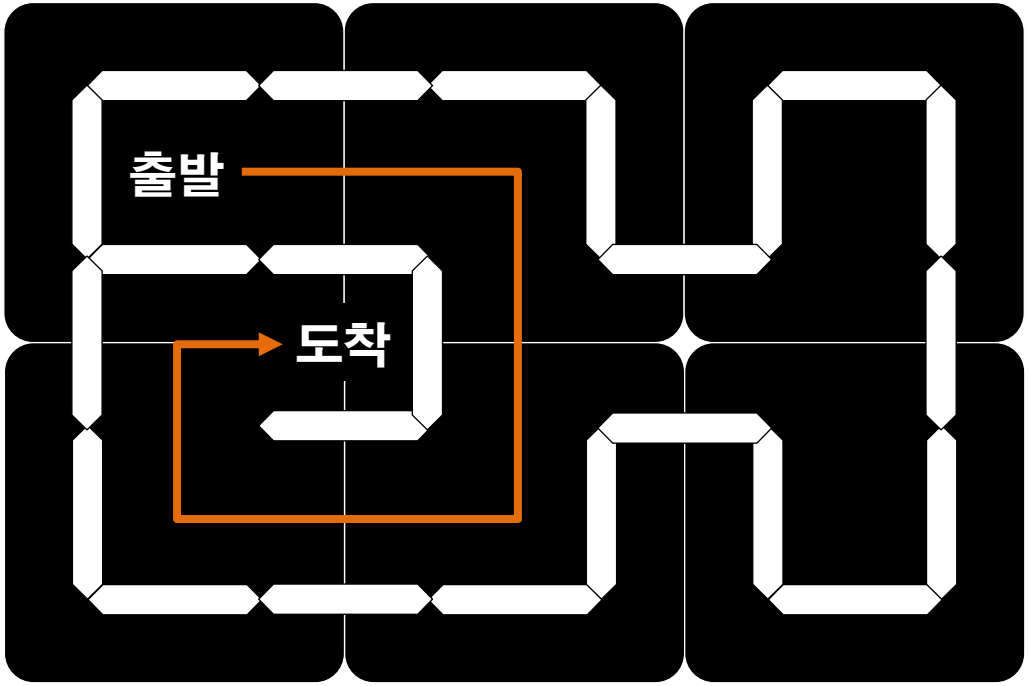
컵과의 거리가 점점 멀어지다가
점점 가까워지다가...

고급



타원 궤도를 따라 이동하기

최단 경로 찾기



알고리즘

- 미로 탐색 알고리즘
  - 좌수법(좌선법), 우수법(우선법)
  - 깊이 우선 탐색, 너비 우선 탐색
  - 다익스트라 알고리즘
  - A* 알고리즘

- 미로 제작 알고리즘
  - http://weblog.jamisbuck.org/2011/2/7/maze-generation-algorithm-recap

수고하셨습니다.

http://hamster.school

akaii@kw.ac.kr