파이썬으로
배우는
비글과 라이다

ROBOMATION
MAKE THINGS ALIVE

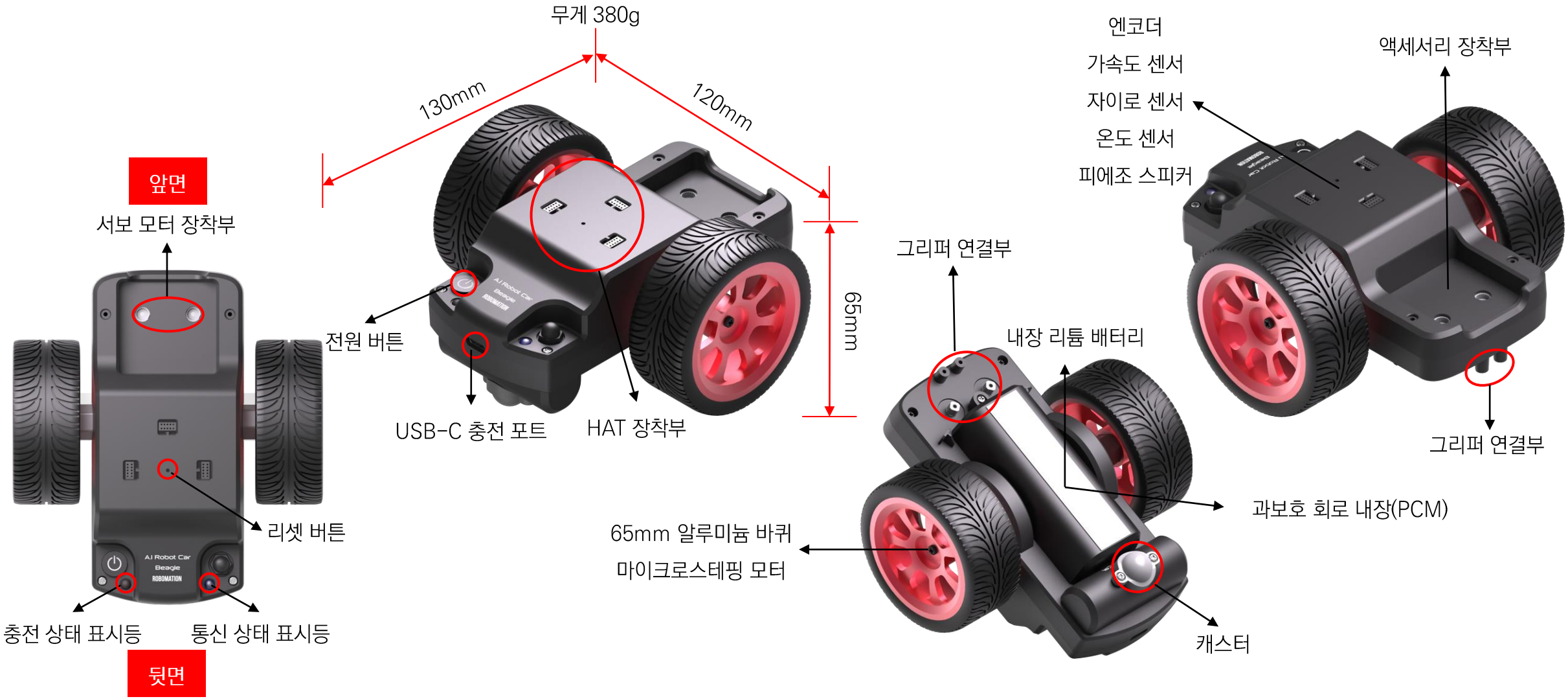
목차

1차시

수업 준비

01 하드웨어 이해

- 비글은 교육용 로봇입니다. 다음 그림과 같이 다양한 장치를 포함하고 있습니다.



01

하드웨어 이해

- 전원이 꺼진 상태에서 전원 버튼을 한 번 누르면 전원 버튼의 LED가 빨간색으로 점등되면서 전원이 켜집니다.
- 전원이 켜진 상태에서 전원 버튼을 1초 이상 누르고 있으면 부저음이 울리고 LED가 꺼지면서 전원이 꺼집니다.
- 배터리가 부족하면 전원 버튼의 LED가 깜박 거립니다.
- 배터리가 부족한 상태에서는 시스템이 제대로 동작하지 않으므로 반드시 충전해 주시기 바랍니다.

오작동 시 로봇 상단의 리셋 버튼 부분을 핀으로 딸깍
하도록 누르면 전원이 꺼지고 시스템이 초기화됩니다.

전원을 켜 LED에 불이
들어온 모습입니다.

01

하드웨어 이해

- 비글은 스마트폰용 충전기를 사용하여 충전할 수 있습니다. USB-C 단자를 비글의 충전 포트에 연결하면 됩니다.
- USB 케이블을 사용하여 충전할 수도 있습니다. USB 케이블의 USB-C 단자를 비글의 충전 포트에 연결하고, 반대쪽을 컴퓨터의 USB 포트에 연결합니다.
- 7V 이상의 과전압이 공급되면 사이렌 소리로 경고하고 전원을 차단합니다. 이 때는 코드를 뽑고 5V 케이블을 사용해 주시기 바랍니다.
- 충전 중에는 충전 상태 표시등이 빨간색으로 표시되고, 충전이 완료되면 충전 상태 표시등이 꺼집니다.
- 완전히 충전하면 약 70분 동안 사용할 수 있습니다.
- 완충까지는 약 4시간이 걸리며, 전원을 끈 상태에서 충전합니다.

01 하드웨어 이해

전원을 켜면 파란 불이
들어옵니다.

- **파란색으로 천천히 깜박임**
  연결을 기다리는 상태입니다.
  비글 로봇의 전원을 켜면 통신 상태 표시등이 파란색으로 천천히 깜박입니다.

- **파란색으로 계속 켜져 있음**
  연결된 상태입니다.
  익스프레스 리시버와 페어링되면 통신 상태 표시등이 계속 켜져 있습니다.

- **파란색으로 빠르게 깜박임**
  정상적으로 연결 후 통신을 하고 있는 상태입니다.
  데이터를 주고 받는 동안에는 통신 상태 표시등이 파란색으로 빠르게 깜박입니다.

- **통신 상태 표시등이 꺼짐**
  로봇의 전원이 꺼진 상태입니다.

※ 비글 로봇은 자사의 타 로봇과 페어링 시 사용하는 블루투스 동글과는 연결할 수 없으므로 꼭 익스프레스 리시버를 사용해 주시기 바랍니다.

01 하드웨어 이해

- 다음 그림과 같이 나노 라이다를 연결할 수 있는 확장 포트를 지니고 있습니다.

01 하드웨어 이해

- 다음 그림과 같이 햄스터 AI 카메라를 거치할 수 있습니다.

햄스터 AI 카메라 상품에 동봉된 재접착 양면 테이프를 붙여 고정할 수 있습니다.
비글용 카메라 거치대(별도 판매)를 활용하여 고정할 수도 있습니다.

01

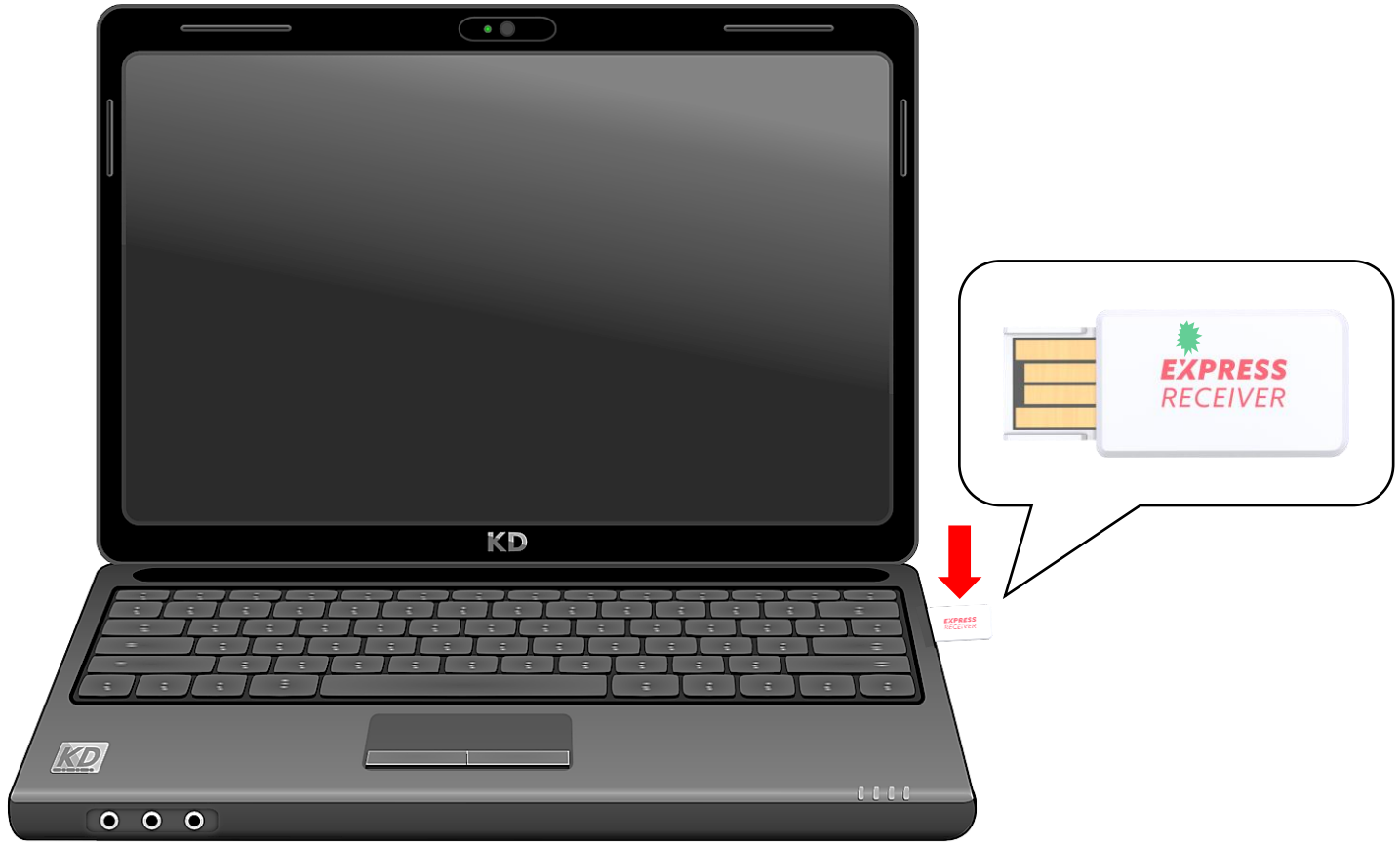
하드웨어 이해

- 나도 라이다와 햄스터 AI 카메라를 동시에 연결해 활용할 수도 있습니다.

02

로봇과 PC 연결

- 비글과의 페어링을 위해 익스프레스 리시버를 PC의 USB 단자에 꽂습니다.
- 익스프레스 리시버의 연결 상태 표시등이 초록색으로 천천히 깜빡이면 정상입니다.



02

로봇과 PC 연결

- 익스프레스 리시버에 가까이 가져가 비글의 전원 버튼을 눌러 전원을 켭니다.
- 비글에서 삑 소리가 나고 비글의 통신 상태 표시등과 익스프레스 리시버의 연결 상태 표시등이 계속 켜져 있거나 빠르게 깜빡이면 정상입니다.

03

파이썬 설치

- 파이썬 홈페이지(https://www.python.org/downloads/)에 접속해 다운로드를 클릭합니다 .
- 최신 버전의 파이썬 중 본인 pc의 사양에 맞는 버전을 다운받습니다.

Active Python Releases

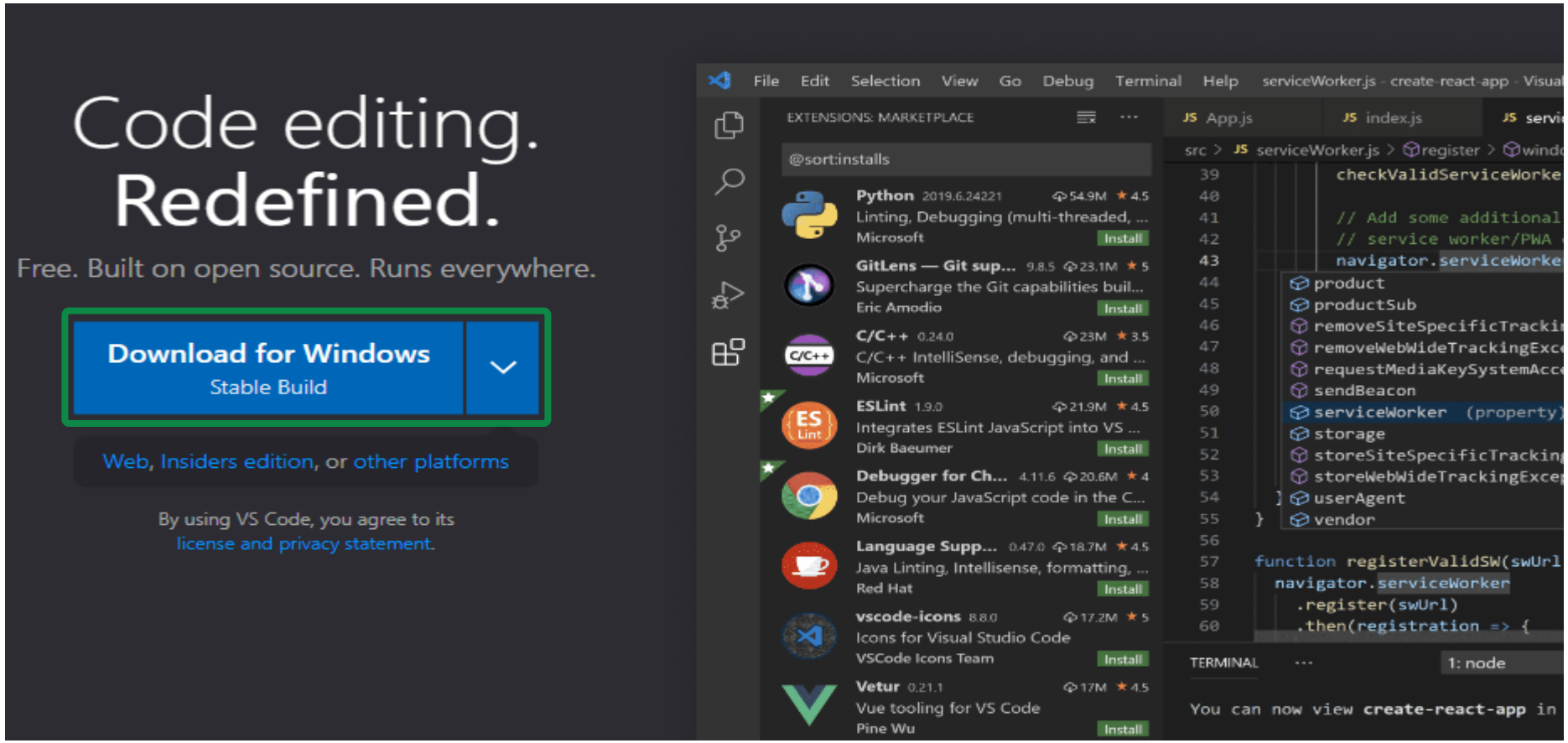
For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.12	prerelease	2023-10-02 (planned)	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

03

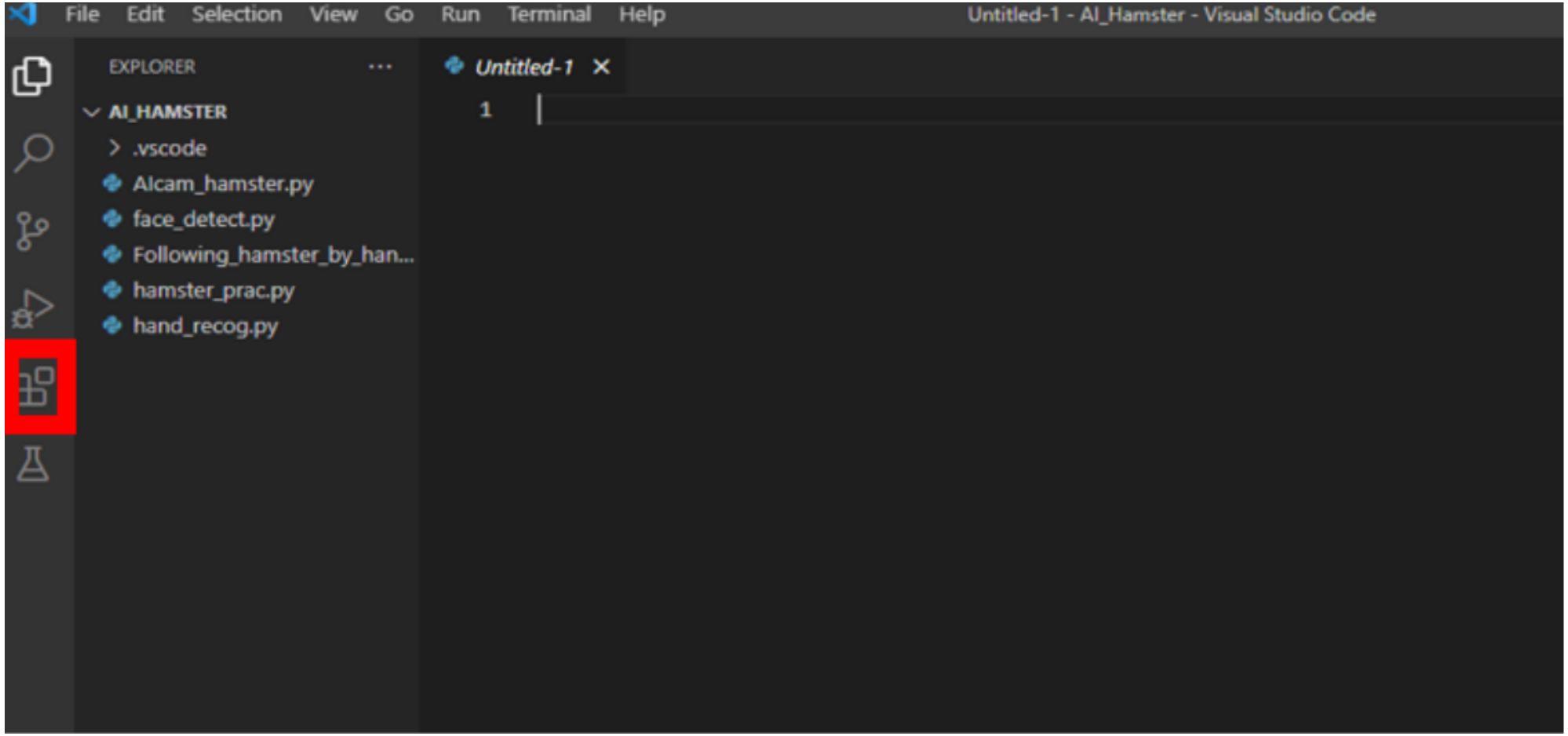
VS Code 설치

- Visual Studio Code 홈페이지(https://code.visualstudio.com/)에 접속해 Download for windows를 클릭합니다 .
- 설치 전 본인 pc의 사양과 호환되는 버전인지 확인합니다.
- 에디터는 VS Code가 아니더라도 Pycharm, Jupyter, IDLE 등 본인이 원하는 에디터를 사용하시면 됩니다.



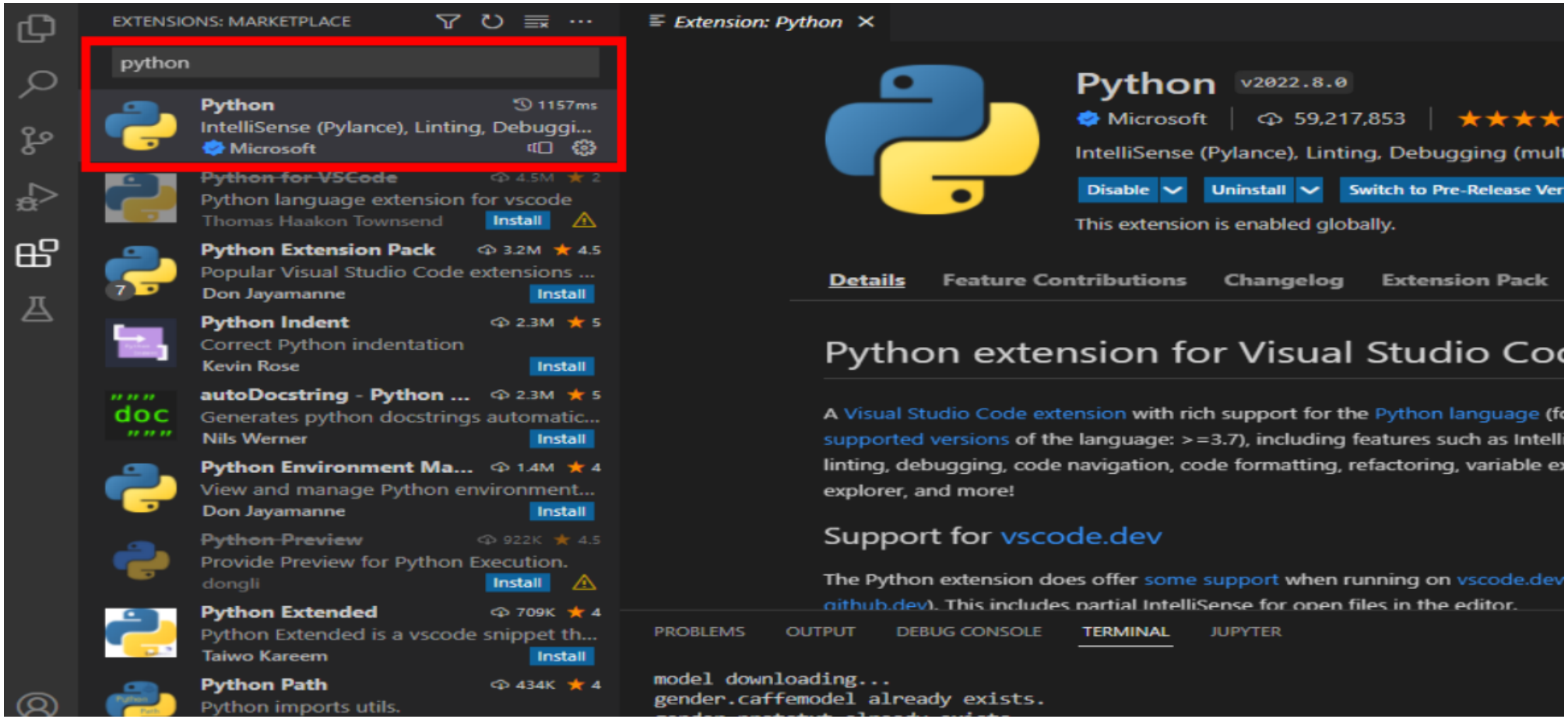
03 VS Code 설치

- VS code 실행 후 좌측 빨간 박스 부분을 클릭합니다.



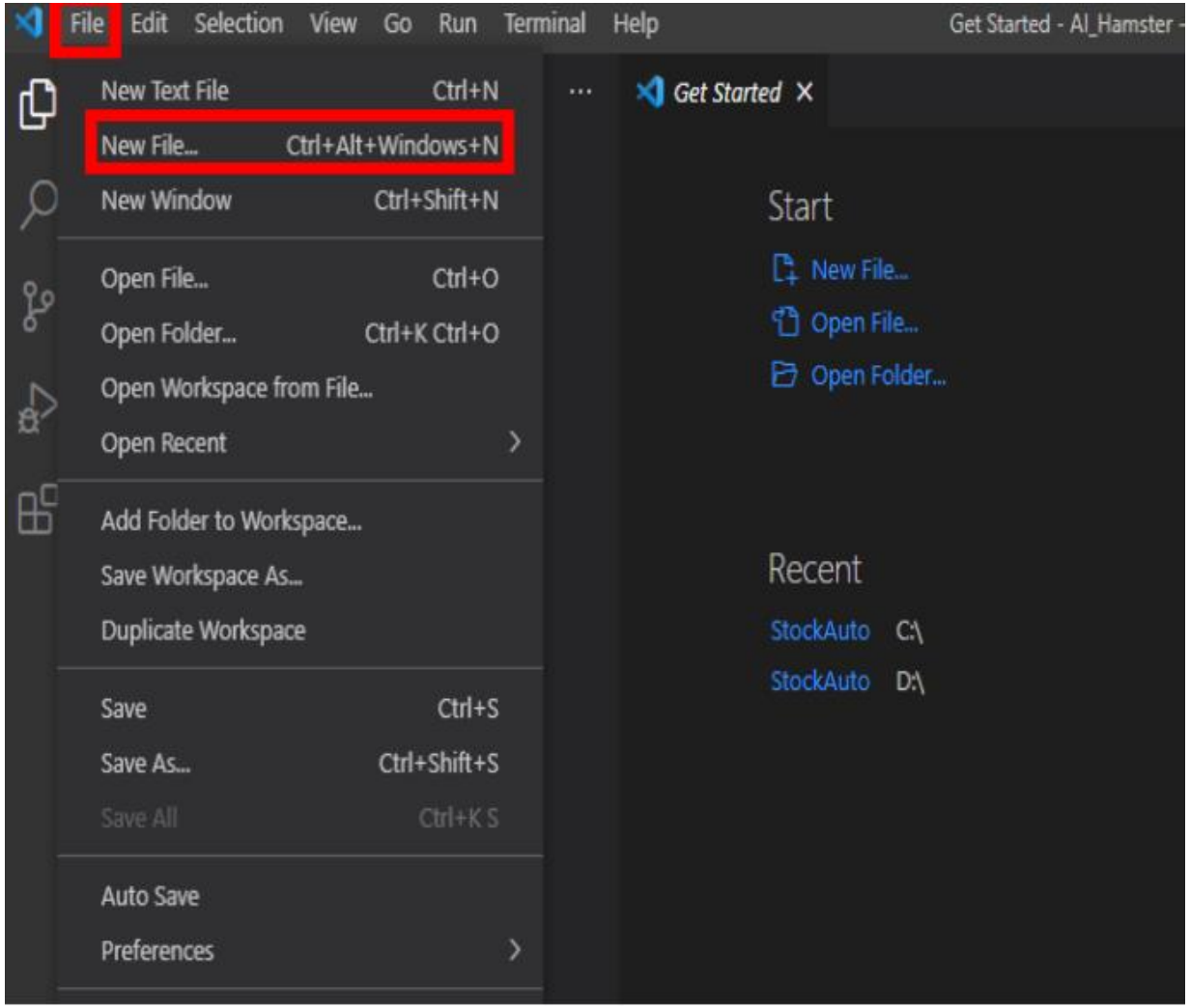
03 VS Code 설치

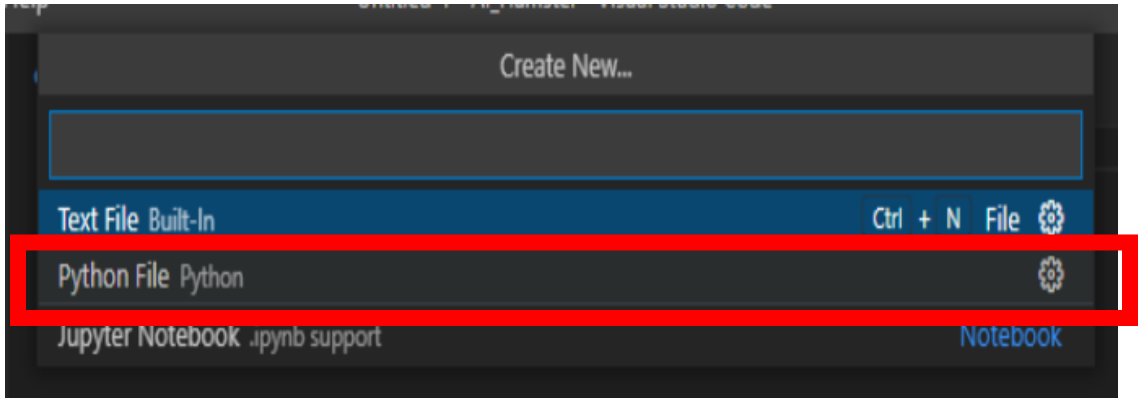
- python 입력 후 가장 상단에 있는 “Python”을 설치합니다. (Install 버튼을 클릭합니다.)



03 VS Code 설치

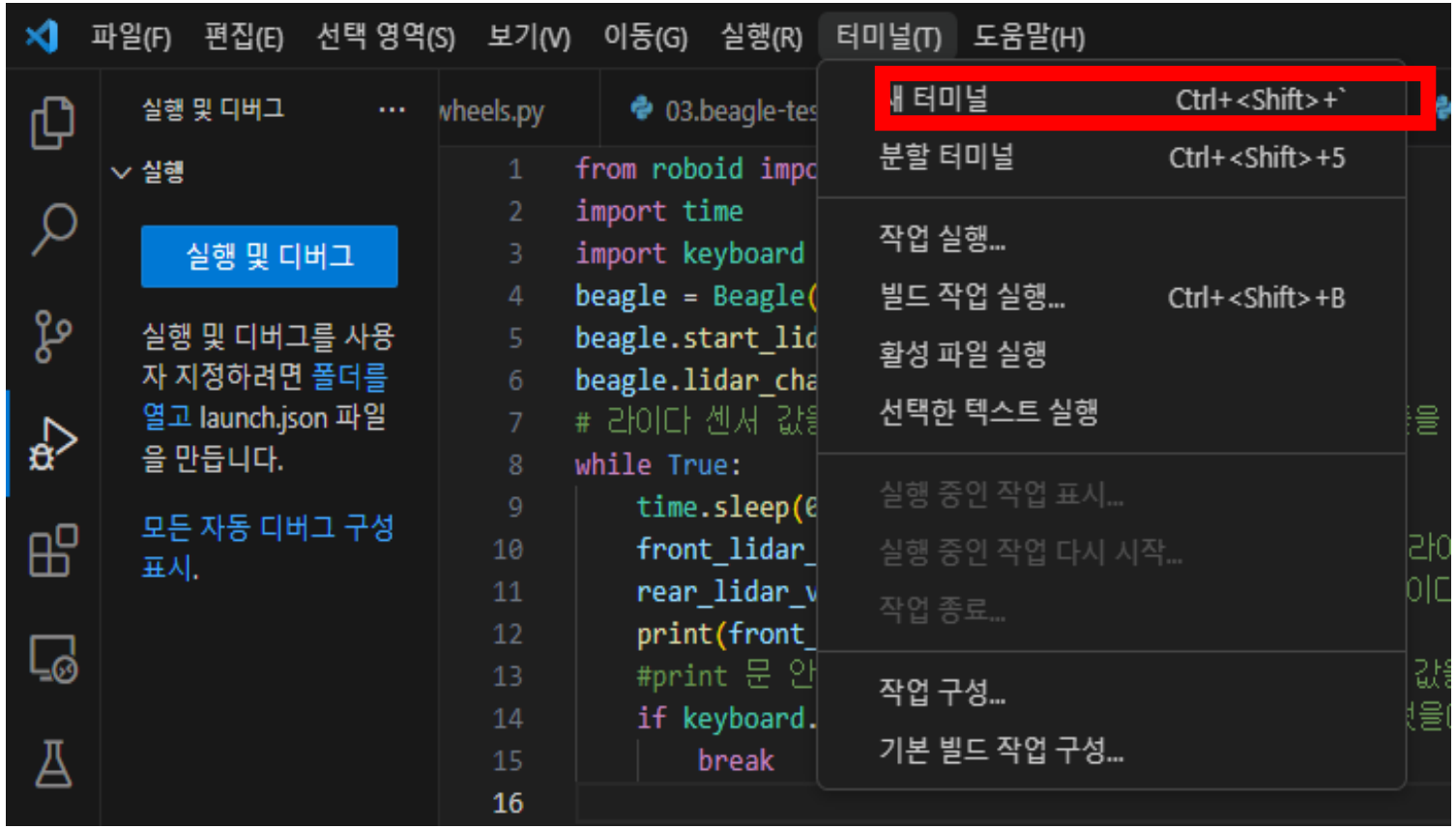
- 좌측 상단에 File 클릭 후 New File를 클릭합니다. Python File을 선택하여 python 파일을 생성합니다.





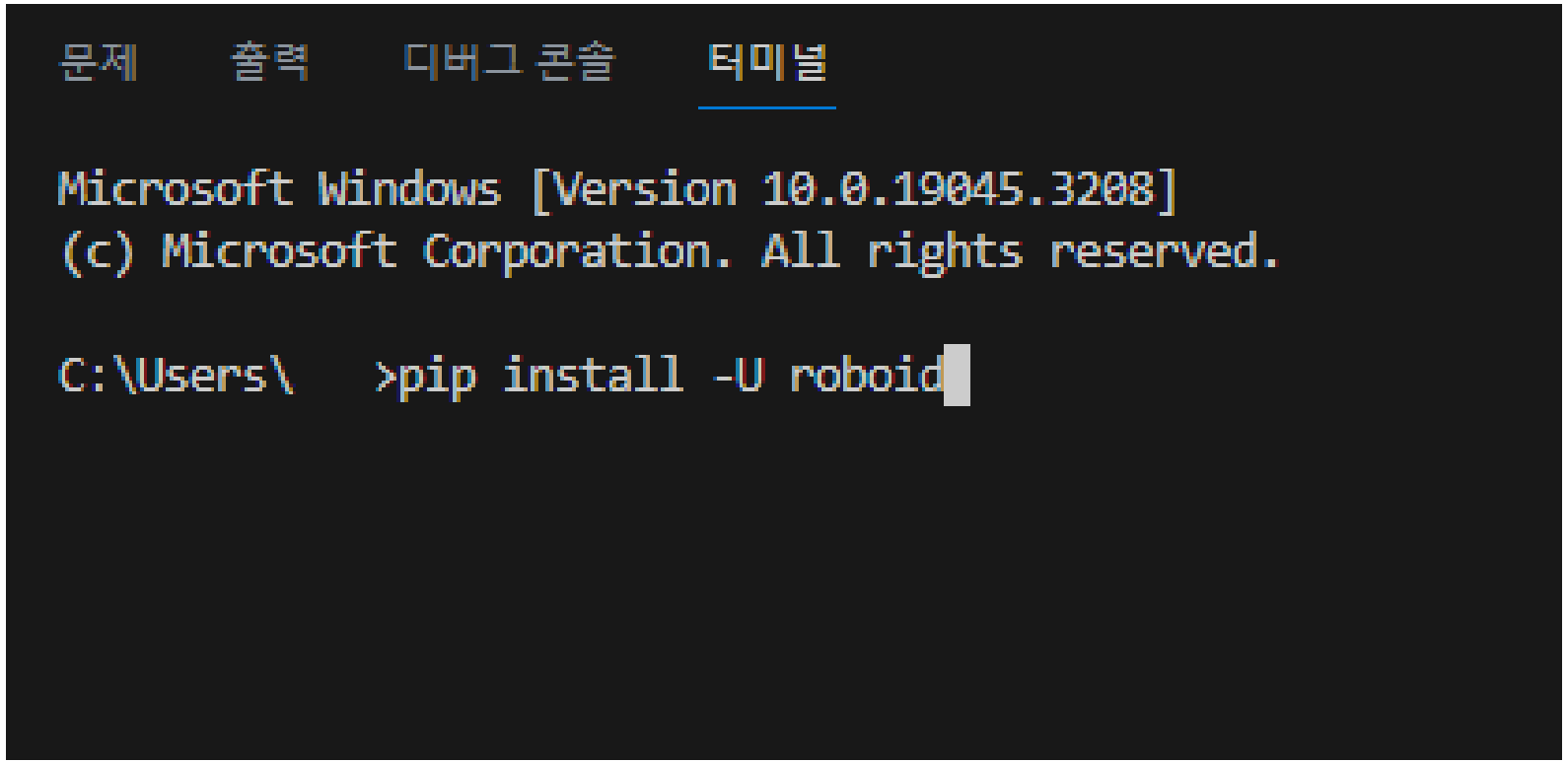
03 VS Code 설치

- 상단의 Terminal 을 클릭하고 New Terminal을 선택하면 터미널을 열 수 있습니다.



03 라이브러리 설치

- VS Code의 터미널 창에서 pip install –U roboid를 입력하여 roboid 라이브러리를 설치합니다.



2차시

코드 설명

01 Beagle 연결

- 파이썬에서 beagle 로봇과 연결 하는 방법입니다.
  (beagle 로봇의 연결을 위해 익스프레스 리시버가 컴퓨터에 연결 되어있어야 합니다.)
- pip install –U roboid로 roboid 라이브러리를 설치하는 과정이 필요합니다.

```
from roboid import *
beagle = Beagle()
```

```
1   from roboid import *   # roboid 라이브러리를 불러옵니다
2   beagle = Beagle() # 비글과 연결을 시도합니다
```

실행 시 파이썬 터미널을 확인하여 연결이 제대로 되었는지 확인할 수 있습니다.

```
Beagle[0] No available USB to BLE bridge
```

왼쪽 문구는 연결이 되지 않았을 때 나타납니다. 익스프레스 리시버의 연결 상태와 beagle 본체의 배터리 등을 확인하여 주세요

```
Beagle[0] Connected: COM8 E6:5E:87:4C:C1:DA
```

제대로 연결이 되었을 시 Connected 문구를 확인할 수 있습니다.

01 Beagle 연결

- beagle 로봇의 연결 해제는 beagle.dispose() 를 사용합니다.

```
from roboid import *
beagle = Beagle()
beagle.dispose()
```

```
1   from roboid import *    # roboid 라이브러리를 불러옵니다
2   beagle = Beagle()  # 비글과 연결을 시도합니다
3   beagle.dispose()    # 연결을 해제합니다
```

```
Beagle[0] Connected: COM8 E6:5E:87:4C:C1:DA
Beagle[0] Disposed
```

터미널에 Disposed라는 문구가 나타나면 연결이 해제된 상태입니다.

02 Beagle 이동

- beagle 로봇을 움직이는 코드를 알아봅시다.

move_forward(sec, velocity) sec초 동안 velocity%의 속도로 앞으로 이동합니다. Velocity값이 음수라면 반대 방향으로 이동합니다.

move_backward(sec, velocity) sec초 동안 velocity%의 속도로 뒤로 이동합니다. Velocity값이 음수라면 반대 방향으로 이동합니다.

turn_left(sec, velocity) sec초 동안 velocity%의 속도로 왼쪽으로 회전합니다. Velocity값이 음수라면 반대 방향으로 회전합니다.

turn_right(sec, velocity) sec초 동안 velocity%의 속도로 오른쪽으로 회전합니다. Velocity값이 음수라면 반대 방향으로 회전합니다.

stop() 양쪽 바퀴를 정지합니다.

02 Beagle 이동

- beagle 로봇을 움직이는 코드를 활용하여 움직여 봅시다.

예시:

```
from roboid import *
beagle = Beagle()
beagle.move_forward(1, 20) # 20%의 속도로 앞으로 1초 이동한다.
beagle.move_backward(1, 20) # 20%의 속도로 뒤로 1초 이동한다.
beagle.turn_left(1, 20) # 20%의 속도로 왼쪽으로 1초 회전한다.
beagle.turn_right(1, 20) # 20%의 속도로 오른쪽으로 1초 회전한다.
beagle.stop() # 양쪽 바퀴를 정지한다.
```

02

Beagle 이동

- beagle 로봇을 움직이는 코드를 추가로 알아봅시다.

pivot_left(sec, velocity)

sec초 동안 velocity%의 속도로 왼쪽 바퀴를 중심으로 앞쪽 방향으로 회전합니다. Velocity값이 음수라면 반대 방향으로 이동합니다.

pivot_right(sec, velocity)

sec초 동안 velocity%의 속도로 오른쪽 바퀴를 중심으로 앞쪽 방향으로 회전합니다. Velocity값이 음수라면 반대 방향으로 이동합니다.

예시:

```
from roboid import *
beagle = Beagle()
beagle.pivot_right(2, 20) # 20%의 속도로 오른쪽 바퀴를 중심으로 2초 회전한다.
beagle.pivot_left(2, 20) # 20%의 속도로 왼쪽 바퀴를 중심으로 2초 회전한다.
```

03

Beagle 바퀴 사용

- beagle 로봇의 바퀴를 조정해서 움직이도록 할 수 있습니다.

wheels(velocity)

양쪽 바퀴가 velocity %의 속도로 앞으로 움직입니다. Velocity 값을 음수로 하면 뒤로 움직입니다.

left_wheel(velocity), right_wheel(velocity)

왼쪽 바퀴, 오른쪽 바퀴를 velocity %의 속도로 앞으로 움직이게 합니다. Velocity 값을 음수로 하면 뒤로 움직입니다.

wheels(left_velocity, right_velocity)

left_velocity%의 속도로 왼쪽 바퀴를 회전시키고 right_velocity %의 속도로 오른쪽 바퀴를 회전시킵니다.

wait(milliseconds) 1000분의 1초 단위로 milliseconds 시간 동안 기다리게 합니다.

wheels 코드는 속도를 지정할 수는 있지만 움직이는 시간을 조정 할 수 는 없기 때문에 wait를 이용하여 이동할 시간을 지정하여 줄 수 있습니다.

위 코드를 이용하여 속도를 조절하거나 이동 방향이나 움직임을 조정할 수 있습니다.

03

Beagle 바퀴 사용

예를 들어

beagle.wheels(20, 20) 20%의 속도로 앞으로 갑니다.

beagle.wheels(−20, −20) 20%의 속도로 뒤로 갑니다.

이런식으로 바퀴의 수치를 조정하면 왼쪽, 오른쪽으로 회전과 바퀴를 중심으로 하는 pivot 기능도 구현할 수 있습니다.

beagle.wheels(−20, 20) 왼쪽 바퀴는 뒤로 오른쪽 바퀴 앞으로 움직이면 제자리 왼쪽 회전이 됩니다.

beagle.wheels(20, −20) 오른쪽 바퀴는 뒤로 왼쪽 바퀴 앞으로 움직이면 제자리 오른쪽 회전이 됩니다.

beagle.wheels(0, 20) 오른쪽 바퀴만 앞으로 움직이면 왼쪽 바퀴를 중심으로 왼쪽 회전이 됩니다.

beagle.wheels(20, 0) 왼쪽 바퀴만 앞으로 움직이면 오른쪽 바퀴를 중심으로 오른쪽 회전이 됩니다.

beagle.wheels(10, 30) 왼쪽 바퀴보다 오른쪽 바퀴가 더 빠르게 움직이게 하면 앞으로 이동하며 완만하게 왼쪽으로 가는 자연스러운 좌회전을 구현할 수 있습니다.

그 밖에도 바퀴의 수치 조정을 통해 다양한 움직임을 구현할 수 있습니다.

03

Beagle 바퀴 사용

Beagle 로봇을 wheels 코드를 이용하여 앞, 뒤로 움직이는 예시:

```
from roboid import *
beagle = Beagle()

beagle.wheels(20, 20)  # 양쪽 바퀴를 20%의 속력으로 앞으로 회전하게 한다.
wait(1000) # 1초 동안 이전 동작을 하며 기다린다. 이후 다음 동작을 수행한다.
# 지금 상황에서는 1초동안 앞으로 가는 동작 이후 뒤로 가는 동작을 실행한다.

beagle.wheels(-20, -20) # 양쪽 바퀴를 20%의 속력으로 뒤로 회전하게 한다.
wait(1000) # 1초 동안 이전 동작을 하며 기다린다. 이후 다음 동작을 수행한다.

beagle.wheels(0, 0) # 양쪽 바퀴를 정지한다.
```

03

Beagle 바퀴 사용

Beagle 로봇을 wheels 코드를 이용한 좌, 우 회전 및 pivot 동작 예시:

```
from roboid import *
beagle = Beagle()

beagle.wheels(-20, 20)  #  20%의 속력으로 왼쪽으로 제자리 회전하게 한다.
wait(1000) # 1초 동안 이전 동작을 하며 기다린다. 이후 다음 동작을 수행한다.

beagle.wheels(20, -20) # 20%의 속력으로 오른쪽으로 제자리 회전하게 한다.
wait(1000) # 1초 동안 이전 동작을 하며 기다린다. 이후 다음 동작을 수행한다.

beagle.wheels(0, 20) # 20%의 속력으로 왼쪽으로 왼쪽바퀴 중심으로 회전하게 한다.
wait(1000) # 1초 동안 이전 동작을 하며 기다린다. 이후 다음 동작을 수행한다.

beagle.wheels(20, 0) # 20%의 속력으로 오른쪽으로 오른쪽바퀴 중심으로 회전하게 한다.
wait(1000) # 1초 동안 이전 동작을 하며 기다린다. 이후 다음 동작을 수행한다.

beagle.wheels(0, 0) # 양쪽 바퀴를 정지한다.
```

04 Beagle 소리

- 코드를 통해 비글에서 소리가 나오게 할 수 있습니다.

buzzer(hz) 음 높이 주파수를 hz로 설정하여 버저 소리가 나오게 한다. 0 ~ 167772.15[Hz]까지 표현 가능하고 0으로 하면 소리를 끌 수 있다.

예시:

```
from roboid import *
beagle = Beagle()
# 버저 소리의 음 높이를 1000 Hz로 한다.
beagle.buzzer(1000)
# 버저 소리의 음 높이를 261.6 Hz로 한다.
beagle.buzzer(261.6)
# 버저 소리를 끈다.
beagle.buzzer(0)
```

04 Beagle 소리

- 비글을 통해 다양한 소리를 표현하여 봅시다.

note(pitch, beats) 버저를 이용하여 정확한 음정을 소리 낼 수 있습니다.
음의 높이(pitch)를 조절하여 옥타브를 조절 하고 음을 표현할 수 있습니다.
또한 beats박자 만큼 소리를 냅니다.
예를 들어 note("C4", 0.5)는 4번째 옥타브 도(C)의 음이며 0.5박자 소리 냅니다.

예시:

```
from roboid import *
beagle = Beagle()
beagle.note("C#5", 0.5) # 5옥타브 도#음을 0.5박자 동안 소리 낸다.
beagle.note("OFF" , 0.5) # 0.5 박자 쉰다.
```

음의 높이	설명
"OFF"	소리를 끈다.
"C4"	4번째 옥타브의 도 음
"C#4", "Db4"	4번째 옥타브의 도#(레b) 음
"D4"	4번째 옥타브의 레 음
"D#4", "Eb4"	4번째 옥타브의 레#(미b) 음
"E4"	4번째 옥타브의 미 음
"F4"	4번째 옥타브의 파 음
"F#4", "Gb4"	4번째 옥타브의 파#(솔b) 음
"G4"	4번째 옥타브의 솔 음
"G#4", "Ab4"	4번째 옥타브의 솔#(라b) 음
"A4"	4번째 옥타브의 라 음
"A#4", "Bb4"	4번째 옥타브의 라#(시b) 음
"B4"	4번째 옥타브의 시 음

04 Beagle 소리

- beagle 에 내장된 특정 소리를 낼 수 있습니다.

sound(sound_id, repeat)

sound_id의 소리를 repeat번 만큼 냅니다.

사용할 수 있는 sound_id의 목록은 옆의 표를 참고해주세요.

직접 코드를 입력하고 들어봅시다.

예시:

```
from roboid import *
beagle = Beagle()
beagle.sound("ENGINE", 2) # “ENGINE” 소리를 2번 냅니다.
```

소리	설명
“OFF”	소리를 끈다.
“BEEP”	삐 소리를 재생한다.
“RANDOM BEEP”	무작위 음 높이의 삐 소리를 재생한다.
“NOISE”	지지직 소리를 재생한다.
“SIREN”	사이렌 소리를 재생한다.
“ENGINE”	엔진 소리를 재생한다.
“CHOP”	쩝 소리를 재생한다.
“ROBOT”	로봇 소리를 재생한다.
“DIBIDIBIDIP”	디비디비딥 소리를 재생한다.
“GOOD JOB”	칭친하는 소리를 재생한다.
“HAPPY”	행복 음악을 재생한다.
“ANGRY”	화남 음악을 재생한다.
“SAD”	슬픔 음악을 재생한다.
“SLEEP”	졸림 음악을 재생한다.
“MARCH”	행진 음악을 재생한다.
“BIRTHDAY”	생일 축하 음악을 재생한다.

04 Beagle 소리

- 소리를 재생한 뒤 특정 동작을 하게 할 수 있습니다.

sound_until_done(sound_id, repeat)

내장된 sound_id의 소리를 repeat번 자리에 입력한 상수만큼 재생하고, 소리 재생이 끝날 때까지 기다립니다.

sound 메소드와는 달리 소리의 재생이 끝날 때까지 기다리는 특징이 있습니다.

예시:

```
from roboid import *
beagle = Beagle()
beagle.sound_until_done("GOOD JOB", 1)
beagle.move_forward()
# 칭찬하는 소리를 1번 재생한 후에 앞으로 이동한다.
```

소리	설명
“OFF”	소리를 끈다.
“BEEP”	삐 소리를 재생한다.
“RANDOM BEEP”	무작위 음 높이의 삐 소리를 재생한다.
“NOISE”	지지직 소리를 재생한다.
“SIREN”	사이렌 소리를 재생한다.
“ENGINE”	엔진 소리를 재생한다.
“CHOP”	쩝 소리를 재생한다.
“ROBOT”	로봇 소리를 재생한다.
“DIBIDIBIDIP”	디비디비딥 소리를 재생한다.
“GOOD JOB”	칭친하는 소리를 재생한다.
“HAPPY”	행복 음악을 재생한다.
“ANGRY”	화남 음악을 재생한다.
“SAD”	슬픔 음악을 재생한다.
“SLEEP”	졸림 음악을 재생한다.
“MARCH”	행진 음악을 재생한다.
“BIRTHDAY”	생일 축하 음악을 재생한다.

04 Beagle 소리

- 간단하게 삐 소리를 낼 수 있습니다.

beep()

삐 소리를 재생하고 소리의 재생이 끝날 때 까지 기다립니다.

sound_until_done('beep')을 호출한 것과 같습니다.

예시:

```
from roboid import *
beagle = Beagle()
# 삐 소리를 재생한다.
beagle.beep()
```

05 Beagle 키보드

- 파이썬에서는 키보드 입력을 통해 동작을 제어할 수 있습니다. 이 방법을 통해 로봇의 움직임도 제어할 수 있습니다.
- Beagle의 움직임을 키보드를 통해 구현하는 방법에 대해 알아보도록 합시다.

import keyboard 파이썬에서 키보드를 입력 값으로 사용하기 위해 설치합니다.

Import time 반복문에서 시간 지연을 하여 안정적인 동작을 하기 위해 사용합니다.

time.sleep(sec) 입력한 값의 초 동안 기다립니다.

If keyboard.is_pressed(" ") 키보드 " "를 눌렀을때 특정 동작을 하도록 하기 위해 사용합니다.

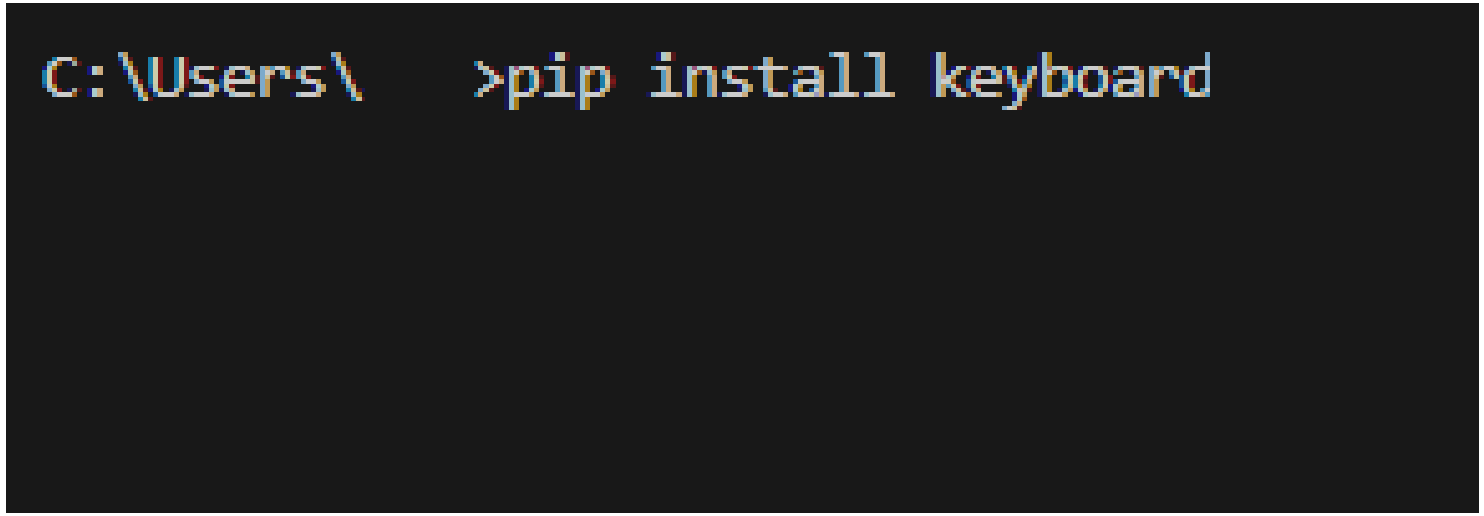
키보드를 눌렀을 때 바로 반응하게 하기 위해 while 반복문을 사용하여 지속적으로 키보드의 상태를 확인하고 반응하도록 합니다.

예를 들어 키보드 방향키 "up"키를 누르면 앞으로 가게 하거나 "q"를 누르면 소리가 나게 하는 등 의 동작을 할 수 있습니다.

05

Beagle 키보드

import keyboard 를 실행하기 위해 터미널에 pip install keyboard를 입력하여 모듈을 다운로드 받습니다.



05 Beagle 키보드

- 키보드 방향키를 눌렸을 때 비글이 움직이도록 할 수 있습니다. 반복문은 ctrl+c를 누르거나 break문으로 종료할 수 있습니다.

Beagle 키보드 사용 예시:

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 반복문인 while문을 이용하여 계속해서 변하는 센서값을 받기 위해 사용
        time.sleep(0.1)  # 0.1초단위로 반응하게 하여 안정적인 동작을 구현
        if keyboard.is_pressed("up"): # 키보드 방향키 위를 눌렀을 때
                beagle.wheels(20,20)  # beagle 로봇 앞으로 이동
        elif keyboard.is_pressed("down"):  # 키보드 방향키 아래를 눌렀을 때
                beagle.wheels(-20,-20)  # beagle 로봇 뒤로 이동
        elif keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을 때
                break   # 반복문 종료
```

06

Beagle 센서

- 비글의 다양한 센서를 파이썬을 통해 사용해 봅시다.

signal_strength() 비글 로봇과 컴퓨터 간의 블루투스 무선통신의 신호 세기를 나타냅니다. 값이 클수록 세기가 커집니다.
신호 세기 값(정수, -128 ~ 0 [dBm], 초기 값: 0)

temperature() 온도 센서 값을 반환합니다. 로봇 내부의 온도 값입니다. 온도 센서 값(정수, -41 ~ 87 [°C], 초기 값: 0)

left_encoder(), right_encoder() 왼쪽, 오른쪽 바퀴의 엔코더 값을 반환합니다. 바퀴가 앞으로 회전하면 증가하고 뒤로 회전하면 값이 감소합니다. 바퀴의 엔코더 펄스 수(정수, -2147483648 ~ 2147483647, 초기 값: 0)

reset_encoder() 양쪽 바퀴의 엔코더 값을 초기화합니다.

비글이 이동하면서 이동한 step수 만큼 펄스 값이 증가합니다.
카운트 값은 정방향은 증가하고 역방향은 감소합니다.
카운팅을 초기화하기 위해서 엔코더 값 초기화하기 코드를 실행하면 값이 0으로 초기화 됩니다.
SLAM 활동을 할 때 라이다와 함께 사용해 비글의 위치를 확인하는데 활용할 수 있습니다.

06

Beagle 센서

센서 값을 각 value에 저장하는 방법입니다.

센서 값 저장하기 예시:

```
from roboid import *
beagle = Beagle()
# 각 value에 센서 값들을 반환해서 저장 하여 줍니다.

signal_value = beagle.signal_strength() # 신호 세기를 저장합니다.
temperature_value = beagle.temperature() # 온도 센서의 값을 저장합니다.
left_encoder_value = beagle.left_encoder() # 왼쪽 바퀴 엔코더 값을 저장합니다.
right_encoder_value = beagle.right_encoder() # 오른쪽 바퀴 엔코더 값을 저장합니다.
beagle.reset_encoder() # 양쪽 바퀴 엔코더 값을 초기화합니다.
```

06

Beagle 센서

- 비글의 다양한 센서를 파이썬을 통해 사용해 봅시다.

accelerometer_x(), accelerometer_y(), accelerometer_z()

X축, Y축, Z축 가속도 센서 값을 반환합니다.

비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향입니다.
가속도 센서 값(실수, -16 ~ 16, 초기 값: 0) [g]

gyroscope_x(), gyroscope_y(), gyroscope_z()

X축, Y축, Z축 에 대한 자이로 센서 값을 반환합니다.

비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향입니다.
자이로 센서 값(실수, -2000 ~ 2000, 초기 값: 0) [°/s]

tilt() 비글 로봇의 기울임 상태 값을 반환합니다. 기울임 상태 값(정수, 앞으로 기울임: 1, 뒤로 기울임: -1, 왼쪽으로 기울임: 2, 오른쪽으로 기울임: -2, 거꾸로 뒤집음: 3, 똑바로 놓음: -3)

battery_state() 배터리 상태 값을 반환한다. 배터리 상태 값(정수, 정상: 3, 중간: 2, 부족: 1, 없음: 0)

charge_state() 충전 상태 값을 반환한다. 충전 상태 값(정수, 충전 중: 1, 충전 중 아님: 0)

06

Beagle 센서

센서 값 저장하기 예시:

```
from roboid import *
beagle = Beagle()
# 각 value에 센서 값들을 반환해서 저장 하여 줍니다.
accel_x_value = beagle.accelerometer_x() # X축 가속도 값을 얻는다.
accel_y_value = beagle.accelerometer_y() # Y축 가속도 값을 얻는다.
accel_z_value = beagle.accelerometer_z() # Z축 가속도 값을 얻는다.
gyro_x_value = beagle.gyroscope_x() # X축 자이로 값을 얻는다.
gyro_y_value = beagle.gyroscope_y() # Y축 자이로 값을 얻는다.
gyro_z_value = beagle.gyroscope_z() # Z축 자이로 값을 얻는다.
tilt_value = beagle.tilt() # 기울임 상태 값을 얻는다.
battery_value = beagle.battery_state() # 배터리 상태 값을 얻는다.
charge_value = beagle.charge_state() # 충전 상태 값을 얻는다.
```

06

Beagle 센서

센서 값을 확인하기 위해서는 지속적으로 센서의 값을 받아주어야 합니다. 이를 위해 반복문을 통해 지속적으로 변하는 센서 값을 받아주고 출력하도록 해줍니다. 센서의 값은 터미널을 통해 확인 할 수 있습니다.
확인하고 싶은 센서의 값을 정해 print문에 value 값이 출력되도록 하여 센서 값을 확인하여 봅시다

이후 센서 값을 확인하고 센서 값을 이용하여 beagle로봇의 동작을 제어 할 수도 있습니다.
예를 들어 기울기 센서를 이용하여 센서의 값을 확인 할 수 있도록 한 뒤 기울기에 따라 센서의 값이 얼마나 변하는지
확인해봅시다. 이후 확인한 수치를 이용하여 특정 수치가 되도록 로봇이 기울어지면 소리가 나오게 하는 동작도 구현할 수 있을 것입니다.

자세한 내용은 이후 beagle 활용 방법에 대해 설명할 때 다시 알려드리겠습니다.

06

Beagle 센서

센서 값 확인하기 예시:

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 반복문인 while문을 이용하여 지속적으로 변하는 센서값을 받기위해 사용
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        accel_x_value = beagle.accelerometer_x() # X축 가속도 값을 얻는다.
        accel_y_value = beagle.accelerometer_y() # Y축 가속도 값을 얻는다.
        accel_z_value = beagle.accelerometer_z() # Z축 가속도 값을 얻는다.
        print(accel_x_value) # print문으로 변경되는 센서값을 터미널에 지속적으로 출력
        #print 문 안에 원하는 value값을 바꿔보면서 터미널에서 값을 확인해보자
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

07

Beagle 라이다

Beagle 로봇에는 라이다 센서가 있습니다. 이번에는 라이다 센서를 사용하는 방법을 알아봅시다.

start_lidar() 라이다를 시작합니다. 시작한 후 실제 값이 들어오기까지 시간이 좀 걸립니다.

stop_lidar() 라이다를 종료합니다.

lidar_chart() 라이다 센서 값들을 차트로 보여줍니다. 이를 통해 주변 사물들을 인식 할 수 있습니다.

left_lidar(), right_lidar(), front_lidar(), rear_lidar(), left_front_lidar(), right_front_lidar() , left_rear_lidar(), right_rear_lidar()

각각 라이다 센서의 왼쪽, 오른쪽, 앞, 뒤, 왼쪽앞, 오른쪽앞, 왼쪽뒤, 오른쪽뒤 영역의 센서 값을 반환합니다. (정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm]

라이다 값도 센서 값 처럼 value에 값을 저장해서 출력해보면서 확인 할 수 있습니다.

07 Beagle 라이다

라이다 값 확인하기 예시:

```
from roboid import *
import time
import keyboard
beagle = Beagle()
beagle.lidar_chart() # 라이다 센서 값들을 차트로 보여준다. 이를 통해 주변 사물들을 인식 할 수 있다.
while True:   # 반복문인 while문을 이용하여 지속적으로 변하는 센서값을 받기위해 사용
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        front_lidar_value= beagle.front_lidar() # 앞 영역의 라이다 값을 얻는다.
        rear_lidar_value= beagle.rear_lidar() # 뒤 영역의 라이다 값을 얻는다.
        print(front_lidar_value) # print문으로 변경되는 라이다값을 터미널에 지속적으로 출력
        #print 문 안에 원하는 value값을 바꿔보면서 터미널에서 값을 확인해보자
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

3차시

이동 코드 활용

01 이동 거리 측정하기

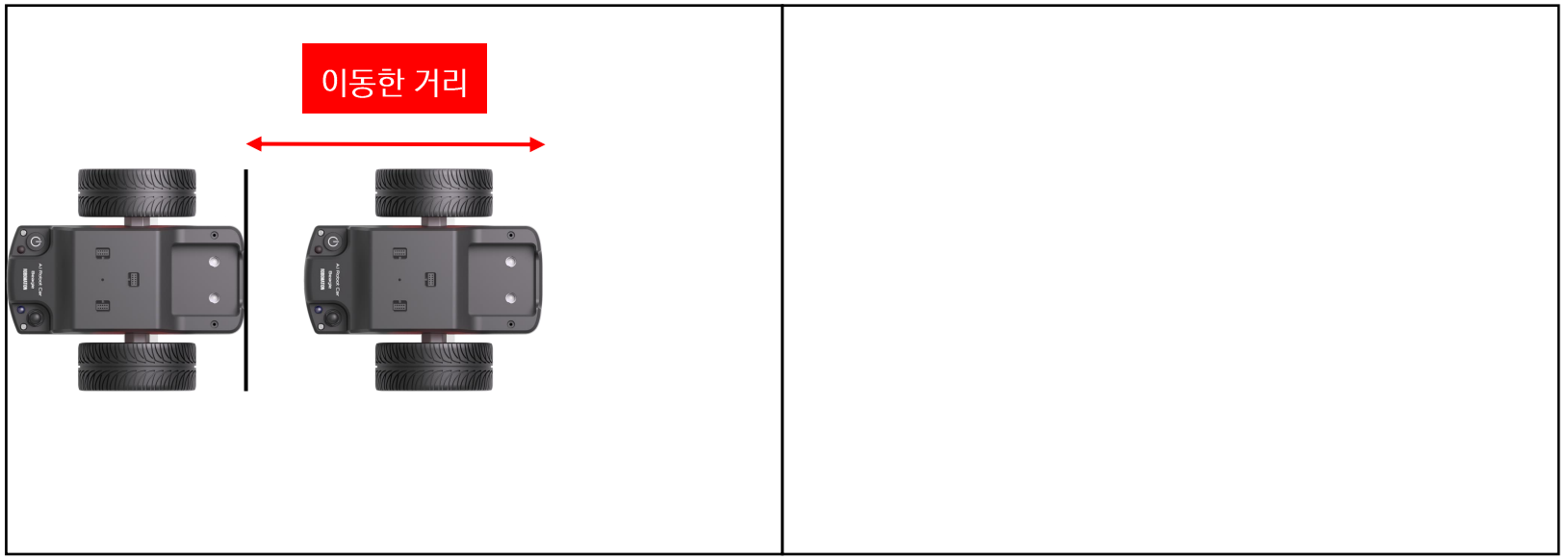
- 비글을 움직이기 위한 프로그램을 작성해 봅니다.
- 아래와 같이 코드를 작성하고 python 파일 실행을 하면 비글이 앞으로 1초 50% 속도로 이동합니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward(1, 50) # 50%의 속도로 앞으로 1초 이동한다
```

01 이동 거리 측정하기

- 비글이 이동한 거리를 측정하기 위해 A3 용지 두 개를 이어 붙입니다. 왼쪽 A3용지의 가장 끝 부분에 맞춰 비글을 내려 놓고 아래와 같이 선을 긋습니다.
- 비글이 움직이면 A3 용지가 함께 움직일 수 있으므로 A3 뒷면에 테이프 등을 붙여 움직이지 않도록 고정합니다.
- 코드 실행 후 해당 위치부터 비글이 이동한 거리를 측정합니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward(1, 50) # 50%의 속도로 앞으로 1초 이동한다
```



01

이동 거리 측정하기

- 비글이 이동한 거리를 자를 이용하여 측정한 뒤 표에 기록해 봅니다.

측정 횟수	1회	2회	3회	4회	5회
이동한 거리	cm	cm	cm	cm	cm

- 이동한 거리는 측정할 때마다 조금씩 달라질 수 있습니다.
- 앞에서 측정한 거리를 모두 더한 후, 5로 나누어 보면 비글이 1초 동안 평균적으로 얼마나 이동하는지 알 수 있습니다.

이동한 거리의 합	cm
합을 5로 나눈 값	cm

01

이동 거리 측정하기

- 이제 비글이 얼마나 빨리 이동하는지, 비글의 속력을 알아보도록 합시다.
- 속력을 좀 더 다양한 방법으로 계산해 보기 위해서 이동하는 시간을 다르게 변경하여 이동한 거리를 측정한 후, 이동한 거리를 이동한 시간으로 나누는 활동을 반복합니다.
- 속력을 구하는 식은 다음과 같습니다.

$$\text{속력(cm/초)} = \frac{\text{이동 거리(cm)}}{\text{이동 시간(초)}}$$

이동한 시간	1초	2초	3초	4초	5초
이동한 거리	cm	cm	cm	cm	cm
속력	cm/초	cm/초	cm/초	cm/초	cm/초

- 속력은 측정할 때마다 조금씩 달라질 수 있으므로 평균 속력이 얼마나 되는지 알아보기 위해 앞에서 계산한 속력을 모두 더한 후 5로 나누어 봅니다.

속력의 합	cm/초
합을 5로 나눈 값	cm/초

01

이동 거리 측정하기

- 이번에는 반대로 일정 거리를 이동하기 위해 몇 초 동안 이동해야 하는지 알아봅니다.
- 비글의 앞부분과 15cm 떨어진 위치에 선을 긋습니다.
- 앞에서 측정한 속력을 이용하여 15cm를 이동하기 위해 필요한 시간을 계산합니다.
- 앞서 배운 속력을 구하는 식을 변형하여 이동 시간을 구하는 식을 만듭니다.

$$\text{이동 시간(초)} = \frac{\text{이동 거리(cm)}}{\text{속력(cm/초)}}$$

- A4 용지의 재질이나 비글의 배터리 상태 등에 따라 이동하는 거리는 조금 달라질 수 있습니다. 앞에서 계산한 값을 입력하여 로봇이 실제 이동한 거리를 관찰한 후, 숫자를 약간씩 변경해 15cm를 이동하도록 합니다.
- 같은 방법으로 20cm, 30cm 등 다양한 거리를 이동하려면 몇 초 동안 이동해야 하는지 계산하고, 프로그램을 수정하여 실행해 봅니다.

이동 거리	15cm	20cm	30cm	45cm
이동 시간	초	초	초	초

01

이동 거리 측정하기

- 이번엔 이동하기 코드의 파라미터인 양쪽 바퀴의 속도 값을 각자 원하는 값으로 바꿔 봅니다.
- 비글의 속력, 이동한 거리, 이동 시간을 다양하게 측정 및 계산 후 친구들과 결과를 비교합니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward(1, 50) # 50%의 속도로 앞으로 1초 이동한다
```

입력할 수 있는 값의 범위는 '-100 ~ 100'이지만 음수 값을 넣으면 바퀴 이동 방향이 반대로 바뀌어 비글이 뒤로 움직이므로 양수 값을 입력합니다.

02

회전 각도 측정하기

- 비글이 제자리에서 돌기 위한 프로그램을 작성해 봅니다.
- 아래 코드를 실행하면 비글이 왼쪽으로 1초 50% 속도로 제자리에서 돕니다.

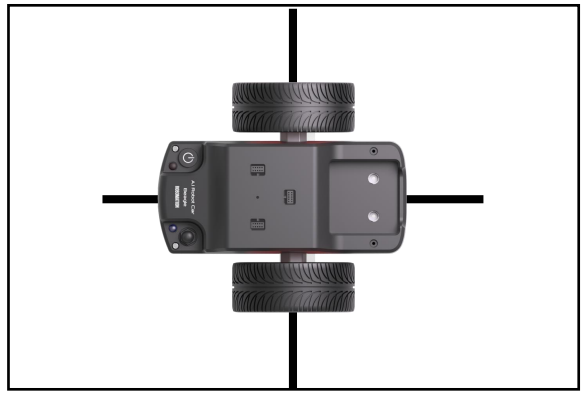
```
from roboid import *
beagle = Beagle()
beagle.turn_left(1, 50) # 50%의 속도로 왼쪽으로 1초 회전한다.
```

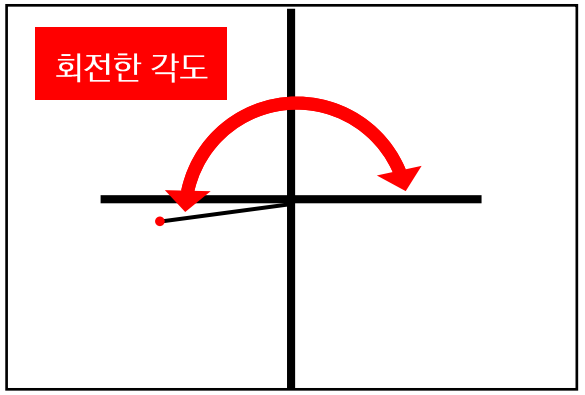
02

회전 각도 측정하기

- 비글이 얼마나 회전하였는지 측정하기 위해 A4 용지 위에 + 모양으로 선을 그리고 비글을 중앙에 맞춰 올려놓습니다. (가로선은 비글의 중심에 맞추고, 세로선은 바퀴의 중심에 맞춥니다.)
- 비글이 움직이면 A4 용지가 함께 움직일 수 있으므로 A4 뒷면에 테이프 등을 붙여 움직이지 않도록 고정합니다.
- 아래 코드 실행 후 비글이 회전을 멈추면 전면 중앙 바로 앞에 점을 찍고, 그 점과 + 표시의 중앙을 이으면 햄스터 로봇이 회전한 각도를 측정할 수 있습니다.

```
from roboid import *
beagle = Beagle()
beagle.turn_left(1, 50) # 50%의 속도로 왼쪽으로 1초 회전한다.
```





02

회전 각도 측정하기

- 비글이 회전한 각도를 측정한 뒤 표에 기록해 봅니다.

측정 횟수	1회	2회	3회	4회	5회
회전한 각도	도	도	도	도	도

- 회전한 각도는 측정할 때마다 조금씩 달라질 수 있습니다.
- 앞에서 측정한 각도를 모두 더한 후, 5로 나누어 보면 평균적으로 얼마나 회전하는지 알 수 있습니다.

회전한 각도의 합	도
합을 5로 나눈 값	도

02

회전 각도 측정하기

- 이제 비글이 얼마나 빨리 회전하는지 알아보도록 합시다.
- 회전한 각도를 회전한 시간으로 나눈 값은 각속력 또는 회전속력이라고 합니다. 각도(도)를 시간(초)으로 나누었기 때문에 회전속력의 측정 단위는 도/초입니다.
- 회전속력을 구하는 식은 다음과 같습니다.

$$\text{회전속력(도/초)} = \frac{\text{회전 각도(도)}}{\text{회전 시간(초)}}$$

- 회전속력을 좀 더 다양하게 계산해 보기 위해 회전하는 시간을 다르게 변경하여 회전한 각도를 측정한 후, 회전한 각도를 회전한 시간으로 나누어 회전속력을 계산해 봅시다.

회전한 시간	1초	2초	3초	4초	5초
회전한 각도	도	도	도	도	도
회전속력	도/초	도/초	도/초	도/초	도/초

- 회전속력은 측정할 때마다 조금씩 달라질 수 있으므로 평균적인 회전속력이 얼마나 되는지 알아보기 위해 앞에서 계산한 회전속력을 모두 더한 후 5로 나누어 봅니다.

회전속력의 합	**도/초**
합을 5로 나눈 값	도/초

02

회전 각도 측정하기

- 이번에는 반대로 일정 각도를 회전하기 위해 몇 초 동안 회전해야 하는지 알아봅니다.
- 앞에서 측정한 회전속력을 이용하여 180도를 이동하기 위해 필요한 시간을 계산해 볼 수 있습니다.
- 앞서 배운 회전 속력을 구하는 식을 변형하면 회전 시간을 구하는 식을 만들 수 있습니다.

$$\text{회전 시간(초)} = \frac{\text{회전 각도(도)}}{\text{회전속력(도/초)}}$$

- A4 용지의 재질이나 비글의 배터리 상태 등에 따라 회전하는 각도는 조금 달라질 수 있습니다. 앞에서 계산한 값을 입력하여 로봇이 실제 회전한 거리를 관찰한 후, 숫자를 약간씩 변경해 180도를 이동하도록 합니다.
- 같은 방법으로 90도, 180도 등 다양한 각도를 회전하려면 몇 초 동안 회전해야 하는지 계산하고, 프로그램을 수정하여 실행해 봅니다.

회전 각도	90도	180도	270도	360도
회전 시간	초	초	초	초

02

회전 각도 측정하기

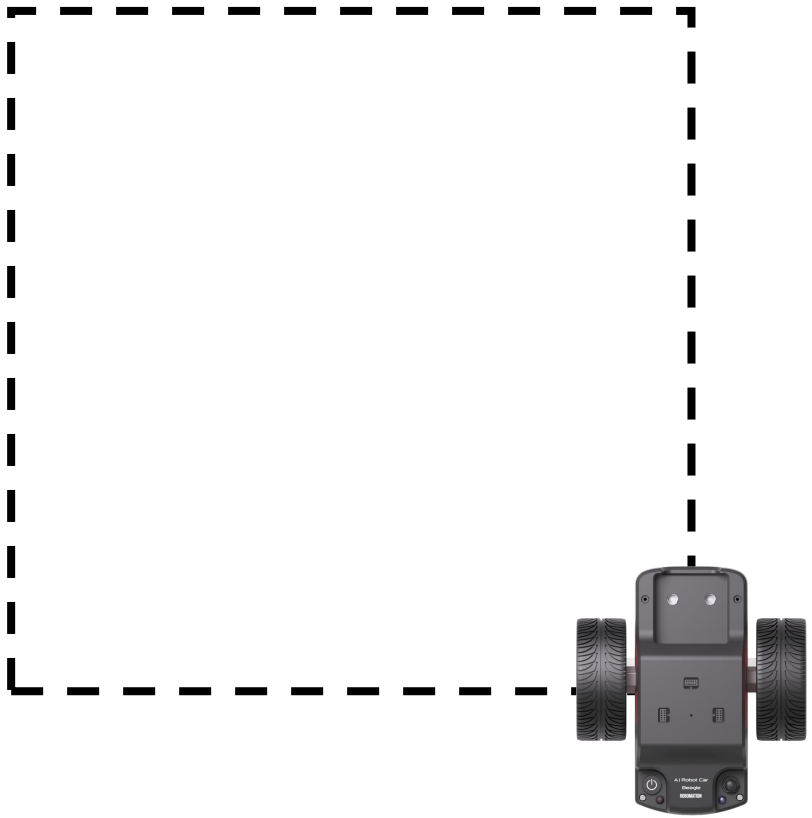
- 이번엔 제자리 돌기 블록의 네 번째 파라미터인 양쪽 바퀴의 속도 값을 각자 원하는 값으로 바꿔 봅니다.
- 비글의 속력, 회전한 각도, 회전 시간을 다양하게 측정 및 계산한 후 친구들과 결과를 비교합니다.

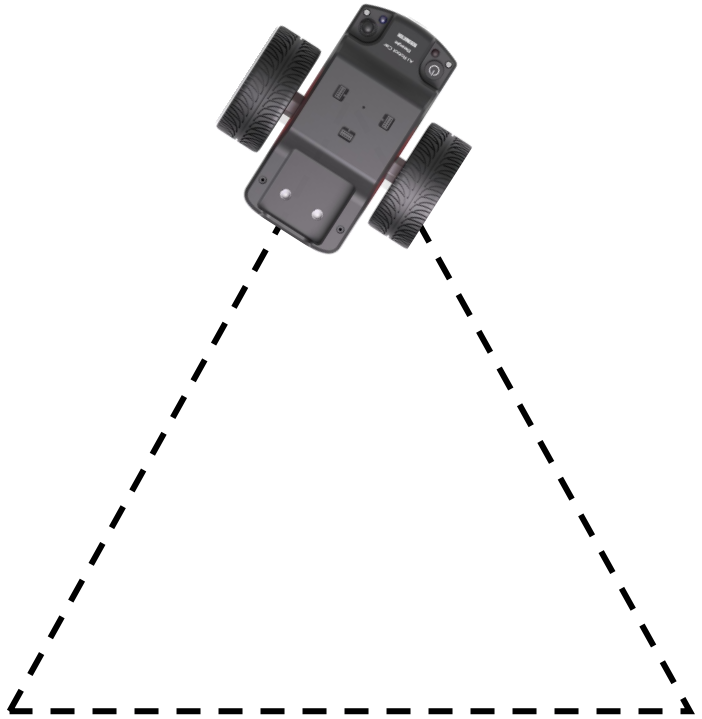
```
from roboid import *
beagle = Beagle()
beagle.turn_left( 1,  50) # 50%의 속도로 왼쪽으로 1초 회전한다.
```

입력할 수 있는 값의 범위는 '-100 ~ 100'이지만 음수 값을 넣으면
바퀴 이동 방향이 반대로 바뀌므로 양수 값을 입력합니다.

03 도형 그리기

- 앞서 배운 코드를 활용하여 프로그램을 작성해 봅니다.
- 여러가지 도형을 그리기 위해 비글에게 순서대로 명령하는 프로그램을 작성해 봅니다.
- 다음 단계 진행 전 어떤 코드를 어떻게 사용해야 할지 각자 의견을 제시합니다.



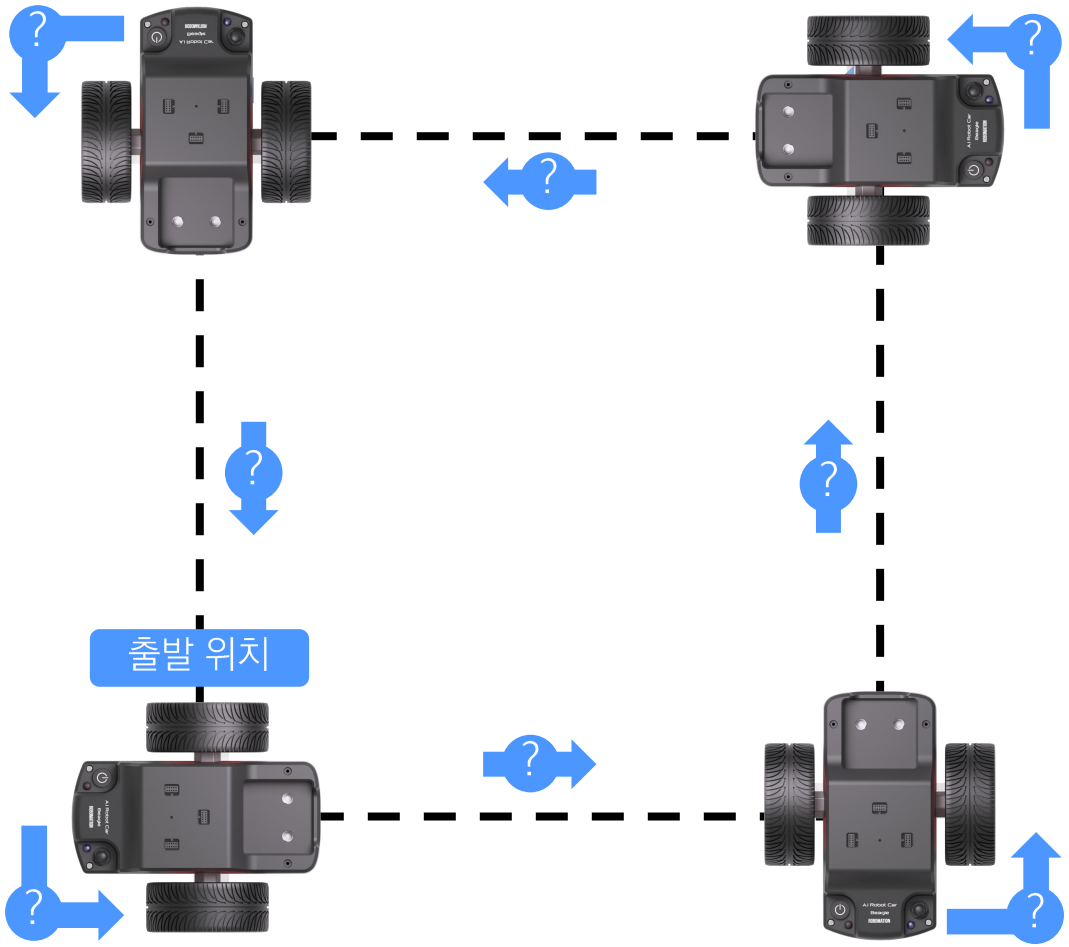


03 도형 그리기

- 비글이 정사각형으로 움직이게 하기 위해서는 어떻게 코드를 작성해야 할지 생각해 봅니다.
- 명령은 어떤 순서로 작성해야 할지, 사용할 수 있는 코드는 무엇일지 생각해 봅니다.
- 정사각형을 그리기 위해서는 몇 도로 회전해야 할지 생각해 봅니다.

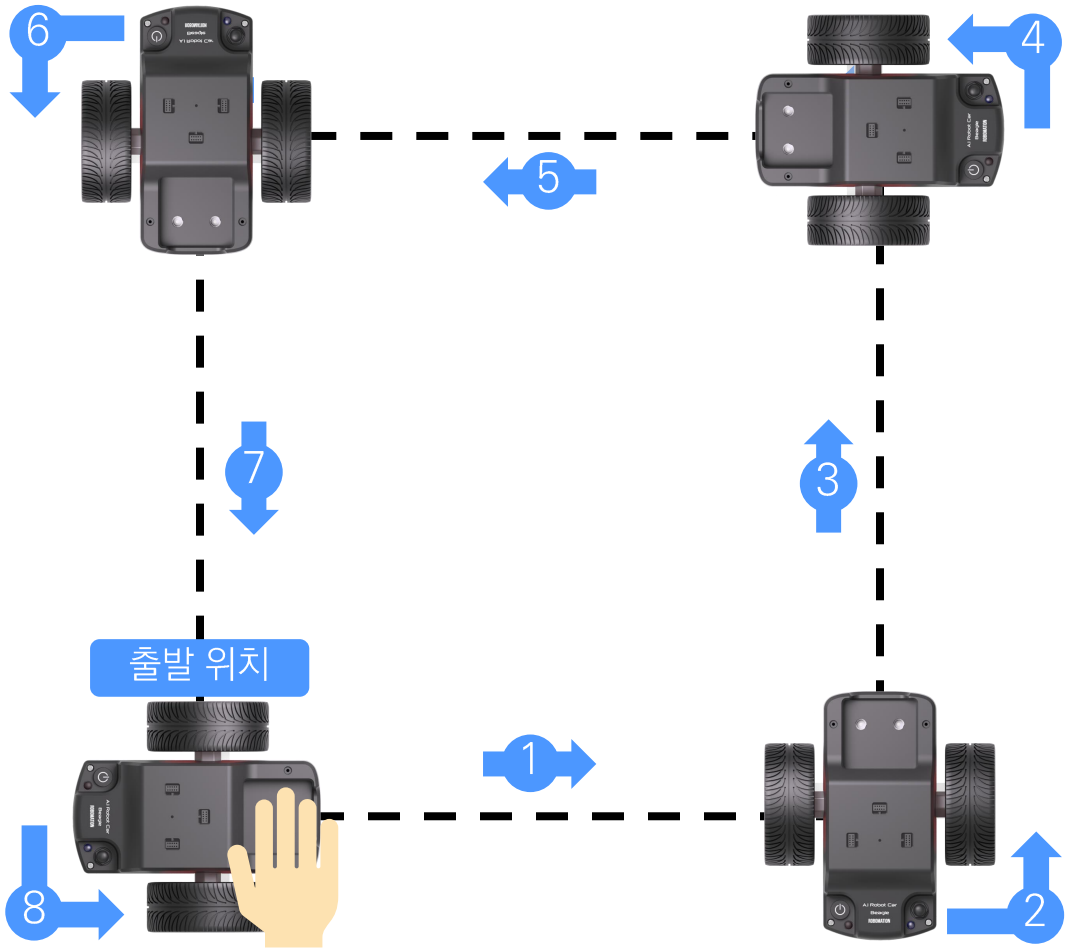
앞으로 가는 코드는 무엇일까요?

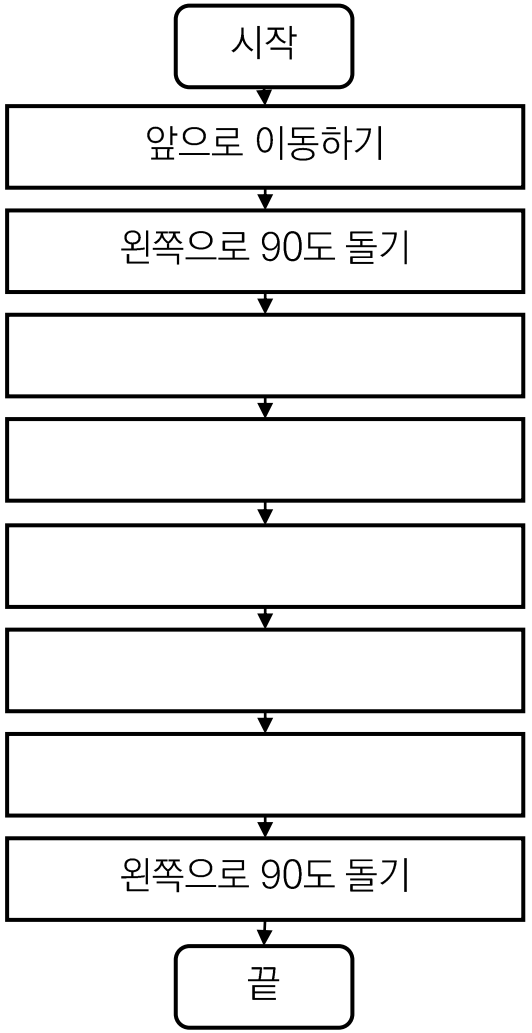
왼쪽으로 도는 코드는 무엇일까요?



03 도형 그리기

- 먼저 코드를 사용할 순서에 대해서 생각해 보고 빈 칸에 순서대로 명령을 적어 봅시다. 이러한 그림을 순서도라고 합니다.
- 적은 순서에 따라 비글을 손으로 들고 명령대로 움직여 봅니다. 비글이 정사각형 모양으로 잘 이동하면 성공입니다.





03

도형 그리기

- 이동 거리, 회전 각도 측정 시간에 사용했던 코드와 측정했던 값을 참고하여 비글이 앞으로 가고 90도로 돌게 합니다.
- 50% 속도를 기준으로 사각형 한 변의 길이와 비글이 90도를 돌게하는 값을 초 자리에 각각 입력합니다.
- 시간 만으로 움직임을 제어하기 힘들 때에는 속도를 함께 조절하여도 됩니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
```

```
from roboid import *
beagle = Beagle()
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
```

03 도형 그리기

- 코드가 완성되면 앞에서 정리한 순서대로 프로그램을 작성하고 실행하여 동작을 확인해 봅니다.
- A4 용지의 재질이나 비글의 배터리 상태 등에 따라 이동, 회전하는 거리가 변할 수 있기 때문에 입력 값은 상황에 따라 달라질 수 있습니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
```

03 도형 그리기

- 같은 동작을 반복하는 것이므로 반복문을 활용해 오른쪽과 같이 표현할 수도 있습니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.5 , 50) # 50%의 속도로 왼쪽으로 0.5초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.5 , 50) # 50%의 속도로 왼쪽으로 0.5초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.5 , 50) # 50%의 속도로 왼쪽으로 0.5초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.5 , 50) # 50%의 속도로 왼쪽으로 0.5초 회전한다.
```

→

```
from roboid import *
beagle = Beagle()

for i in range(4):

        beagle.move_forward( 1, 50)
        # 50%의 속도로 앞으로 1초 이동한다
        beagle.turn_left( 0.5 , 50)
        # 50%의 속도로 왼쪽으로 0.5초 회전한다.
```

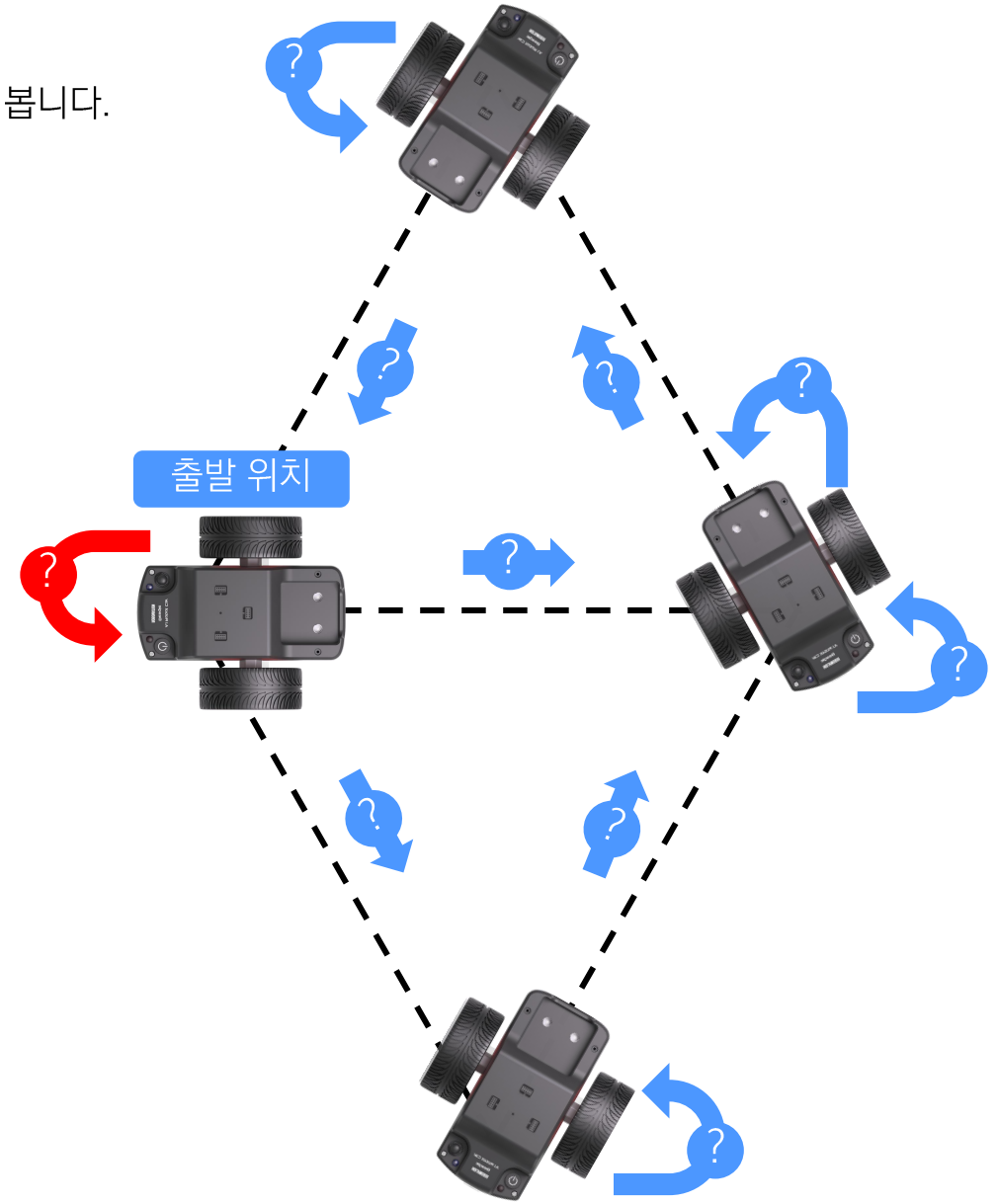
03

도형 그리기

- 비글이 마름모 모양으로 움직이게 하기 위해서는 어떻게 코드를 작성해야 할지 생각해 봅니다.
- 명령은 어떤 순서로 작성해야 할지, 사용할 수 있는 코드는 무엇일지 생각해 봅니다.
- 마름모를 그리기 위해서는 몇 도로 회전해야 할지 생각해 봅니다.
- 이번에는 회전해야 하는 각도가 2가지 이므로 회전하는 명령이 하나 더 필요합니다.
- 한 바퀴 마지막에는 출발할 때와 반대 방향을 바라보게 합니다.

앞으로 가는 블록은 무엇일까요?

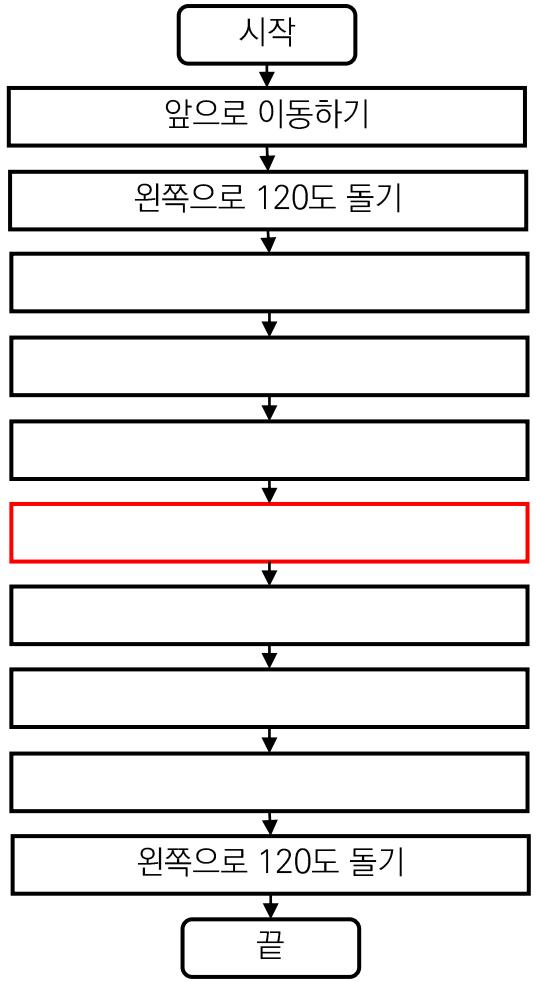
왼쪽으로 도는 블록은 무엇일까요?

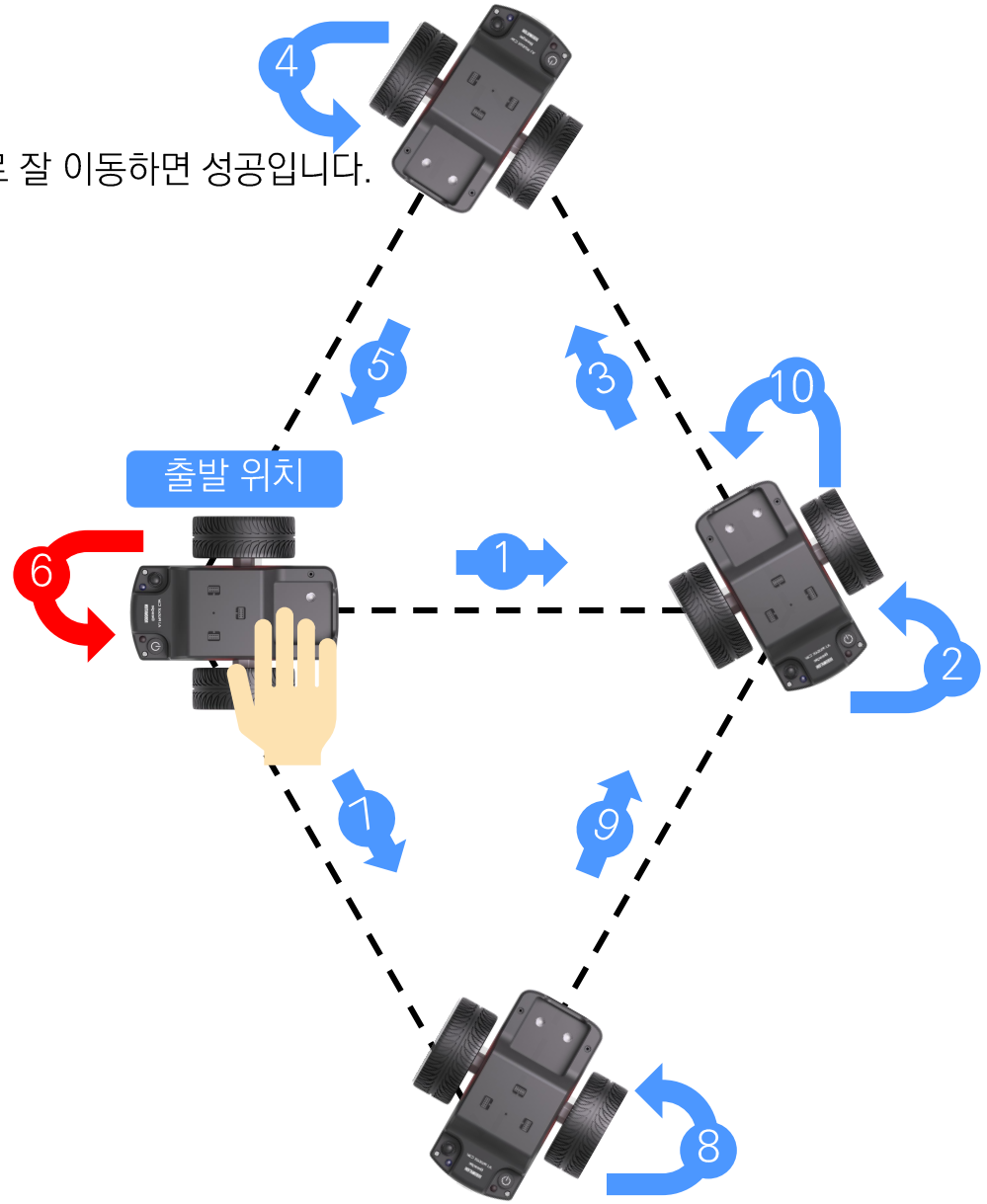


03

도형 그리기

- 먼저 블록을 배치할 순서에 대해서 생각해 보고 빈 칸에 순서대로 명령을 적어 봅시다.
- 적은 순서에 따라 비글을 손으로 들고 명령대로 움직여 봅니다. 비글이 마름모 모양으로 잘 이동하면 성공입니다.





03

도형 그리기

- 이동 거리, 회전 각도 측정 시간에 사용했던 블록과 측정했던 값을 참고하여 비글이 앞으로 가고 90도로 돌게 합니다.
- 50% 속도를 기준으로 사각형 한 변의 길이와 비글이 120도, 60도 만큼 돌게하는 값을 초 자리에 각각 입력합니다.
- 시간 만으로 움직임을 제어하기 힘들 때에는 속도를 함께 조절하여도 됩니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
```

```
from roboid import *
beagle = Beagle()
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
```

03 도형 그리기

- 코드가 완성되면 앞에서 정리한 순서대로 프로그램을 작성하고 실행하여 동작을 확인해 봅니다.
- A4 용지의 재질이나 비글의 배터리 상태 등에 따라 이동, 회전하는 거리가 변할 수 있기 때문에 입력 값은 상황에 따라 달라질 수 있습니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
beagle.move_forward( ? , 50) # 50%의 속도로 앞으로 ?초 이동한다
beagle.turn_left( ? , 50) # 50%의 속도로 왼쪽으로 ?초 회전한다.
```

03 도형 그리기

- 같은 동작을 반복하는 것이므로 반복문을 활용해 오른쪽과 같이 표현할 수도 있습니다.
- 마름모는 정삼각형 두 개를 위아래로 이어 붙인 모양이므로 반복문 안의 코드를 3번 반복하면 정삼각형을 그릴 수 있습니다.

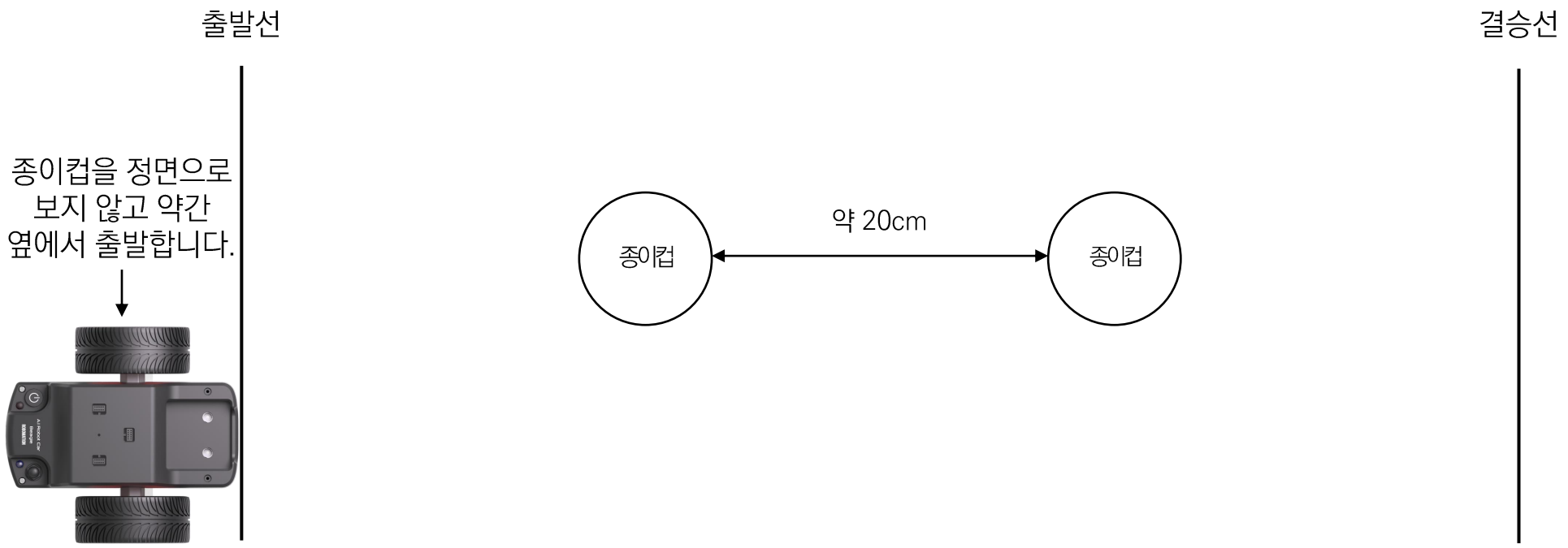
```
from roboid import *
beagle = Beagle()
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.6 , 50) # 50%의 속도로 왼쪽으로 0.6초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.6 , 50) # 50%의 속도로 왼쪽으로 0.6초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.3 , 50) # 50%의 속도로 왼쪽으로 0.3초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.6 , 50) # 50%의 속도로 왼쪽으로 0.6초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.6 , 50) # 50%의 속도로 왼쪽으로 0.6초 회전한다.
```

```
from roboid import *
beagle = Beagle()

for i in range(2):
    beagle.move_forward( 1, 50)
    # 50%의 속도로 앞으로 1초 이동한다
    beagle.turn_left( 0.6 , 50)
    # 50%의 속도로 왼쪽으로 0.6초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
beagle.turn_left( 0.3 , 50) # 50%의 속도로 왼쪽으로 0.3초 회전한다.
for i in range(2):
    beagle.move_forward( 1, 50)
    # 50%의 속도로 앞으로 1초 이동한다
    beagle.turn_left( 0.6 , 50)
    # 50%의 속도로 왼쪽으로 0.6초 회전한다.
```

04

이동하기/돌기 코드 응용하기

- 이동하기 코드와 회전 코드를 활용해 슬라럼 경기를 해봅니다.
- 슬라럼은 장애물을 세워 놓고 그 사이를 지그재그로 통과하는 것입니다.
- 대략 20cm 간격으로 놓은 종이컵 사이를 지나 결승선에 도착하도록 합니다. 결승선에 도착 하는 시간이 가장 짧은 사람이 승리합니다.

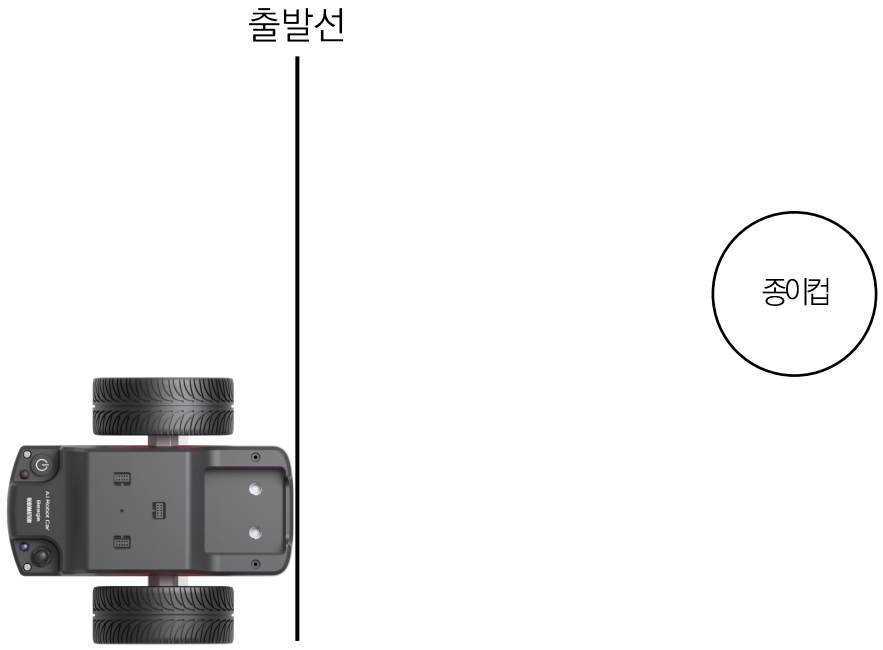


04

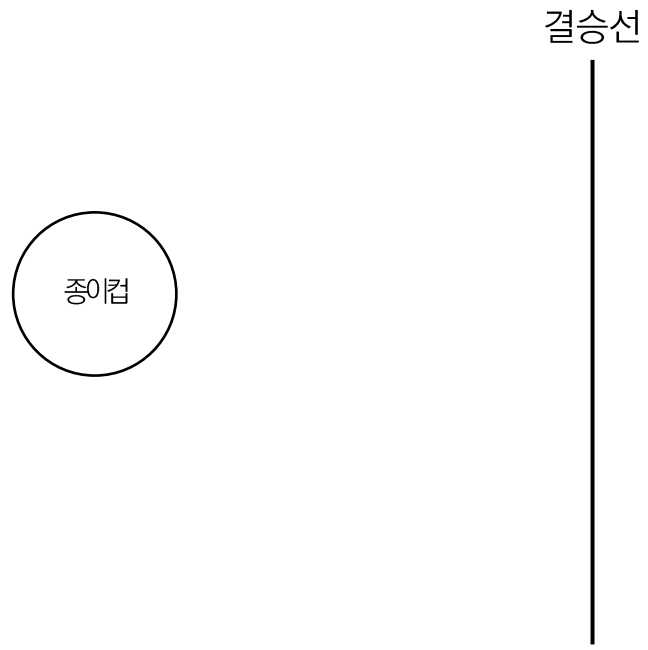
이동하기/돌기 코드 응용하기

- 사용할 수 있는 코드가 아래에 나온 두 코드일 때 비글을 어떤 경로로 이동시킬지 예상하고 종이에 그려 봅니다.
- 원하는 경로로 이동하기 위한 코드를 작성해 봅니다.

```
from roboid import *
beagle = Beagle()
beagle.move_forward(1, 50) # 50%의 속도로 앞으로 1초 이동한다
```



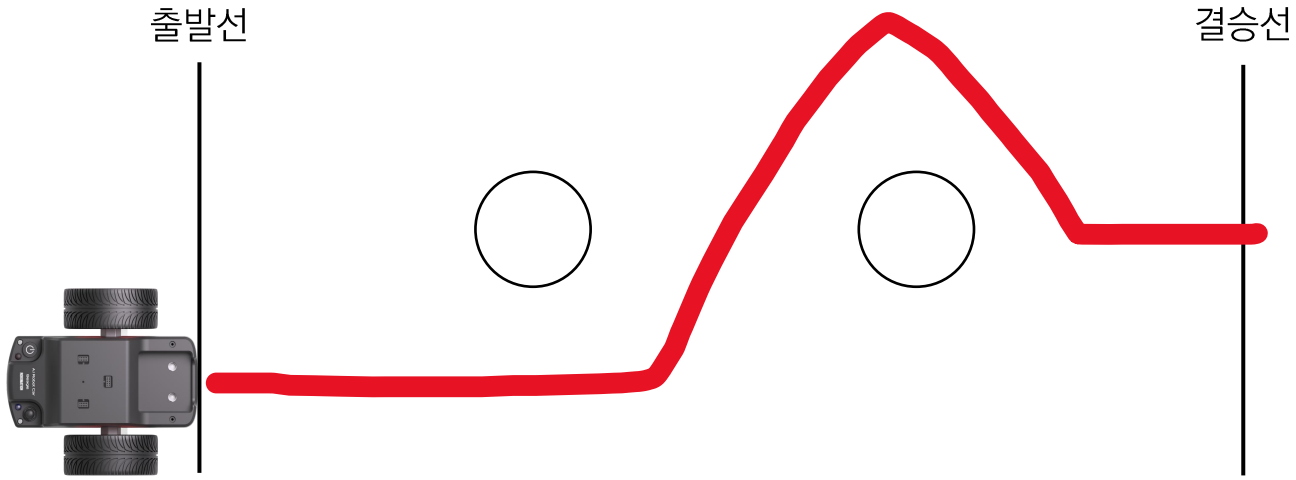

```
from roboid import *
beagle = Beagle()
beagle.turn_left(1, 50) # 50%의 속도로 왼쪽으로 1초 회전한다.
```



04 이동하기/돌기 코드 응용하기

- A4 용지의 재질이나 비글의 배터리 상태, 컵 사이의 간격 등에 따라 이동, 회전하는 거리가 변할 수 있으므로 입력 값은 상황에 따라 달라질 수 있습니다.
- 누가 가장 빠른 시간에 결승선에 도달하는 경로와 코드를 작성하였는지 시간을 기록해 친구와 비교해 봅니다.

예시




```
from roboid import *
beagle = Beagle()
beagle.move_forward(2.5, 50) # 50%의 속도로 앞으로 2.5초 이동한다
beagle.turn_left( 0.3 , 50) # 50%의 속도로 왼쪽으로 0.3초 회전한다.
beagle.move_forward(2, 50) # 50%의 속도로 앞으로 2초 이동한다
beagle.turn_right( 0.6 , 50) # 50%의 속도로 오른쪽으로 0.6초 회전한다.
beagle.move_forward(1.5, 50) # 50%의 속도로 앞으로 1.5초 이동한다
beagle.turn_left( 0.3 , 50) # 50%의 속도로 왼쪽으로 0.3초 회전한다.
beagle.move_forward( 1, 50) # 50%의 속도로 앞으로 1초 이동한다
```

05

바퀴 값으로 움직이기

- 비글 바퀴 값에 입력하는 숫자를 다르게 하여 비글이 이동하는 방향을 조종해 봅니다.
- 아래 코드의 첫 번째 파라미터와 두 번째 파라미터의 값을 바꿔가며 비글이 어떻게 움직이는지 살펴봅니다. 파라미터의 값은 -100~100 사이의 값을 입력할 수 있습니다.

```
from roboid import *
beagle = Beagle()
beagle.wheels(50, 50)
# 왼쪽 바퀴를 50% 오른쪽 바퀴를 50% 로 정하기
```

```
from roboid import *
beagle = Beagle()
beagle.wheels(-50, -50)
# 왼쪽 바퀴를 -50% 오른쪽 바퀴를 -50% 로 정하기
```

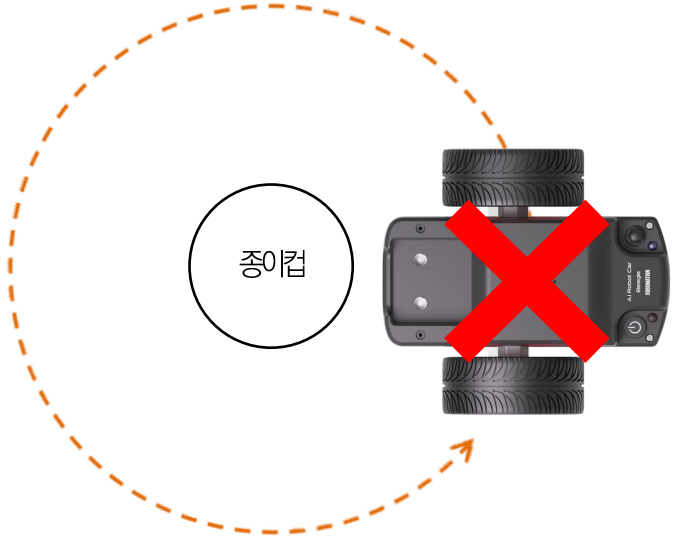
```
from roboid import *
beagle = Beagle()
beagle.wheels(60, 40)
# 왼쪽 바퀴를 60% 오른쪽 바퀴를 40% 로 정하기
```

```
from roboid import *
beagle = Beagle()
beagle.wheels(50, -50)
# 왼쪽 바퀴를 50% 오른쪽 바퀴를 -50% 로 정하기
```

05

바퀴 값으로 움직이기

- 바퀴 값에 따라 달라지는 비글의 움직임을 확인했다면 이번엔 종이컵을 놓고 비글이 그 주변을 돌도록 코드를 작성해 봅니다.
- 비글을 종이컵 옆에 왼쪽과 같이 내려놓습니다. 오른쪽 그림처럼 비글의 앞부분이 종이컵을 바라보도록 내려 놓으면 종이컵 주위를 돌기 어렵습니다.

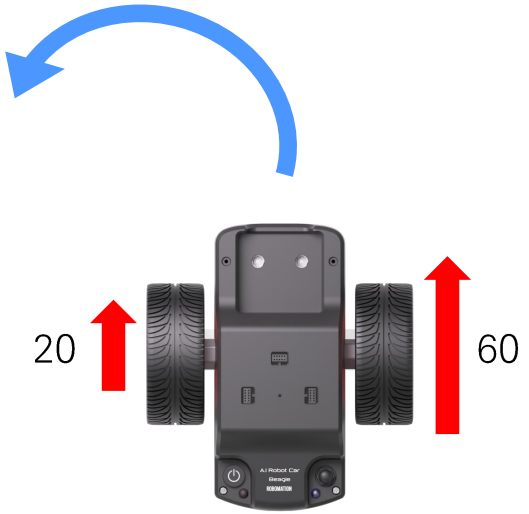


05

바퀴 값으로 움직이기

- 종이컵이 비글의 왼쪽에 놓여 있으므로 비글이 왼쪽 방향으로 돌도록 코드를 작성해야 합니다.
- 아래 블록과 같이 오른쪽 바퀴 속도를 왼쪽 바퀴 속도 보다 크게 하면 비글이 왼쪽으로 원을 그리며 돕니다.
- 두 바퀴 값 크기의 차이가 달라지면 비글이 그리는 원의 크기도 달라집니다.
- 다양한 값을 넣어 종이컵을 따라 도는데 가장 적합한 값을 찾습니다.


```
from roboid import *
beagle = Beagle()
beagle.wheels(20, 60)
# 왼쪽 바퀴를 20% 오른쪽 바퀴를 60%로 정하기
```



05

바퀴 값으로 움직이기

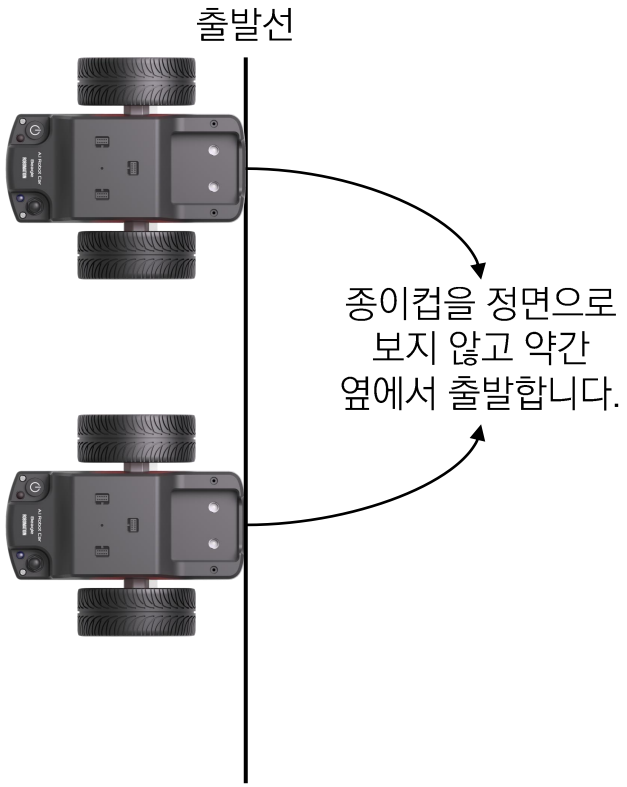
- 이번엔 연속해서 2개의 종이컵을 통과할 수 있는 프로그램을 만들어 봅니다.
- 두 종이컵은 비글의 가로 넓이 간격으로 세워둡니다.
- 파라미터 값으로 다양한 숫자를 입력하여 바퀴의 움직임을 조작합니다. 오른쪽 코드는 예시로 다양한 방식으로 코드를 작성할 수 있습니다.


```
from roboid import *
beagle = Beagle()
beagle.wheels( ? , ? )
# 왼쪽 바퀴를 ? % 오른쪽 바퀴를 ? % 로 정하기
wait(1000) # 1초 기다리기(1000ms)
beagle.wheels( ? , ? )
# 왼쪽 바퀴를 ? % 오른쪽 바퀴를 ? % 로 정하기
wait(1000) # 1초 기다리기(1000ms)
beagle.stop() # 멈추기
# beagle.wheels(0, 0)으로도 멈출수 있습니다.
```

05

바퀴 값으로 움직이기

- 출발선에서 30cm 정도 떨어진 적당한 거리에 종이컵을 둡니다.
- 출발선에서 동시에 출발하여 종이컵을 돌아 반환점으로 먼저 돌아오는 사람이 승리하는 경기입니다.
- 앞에서 공부했던 회전 방법에 대해 생각해 보고 예시 코드를 참고하여 코드를 작성합니다.



종이컵

종이컵

예시

```
from roboid import *
beagle = Beagle()
beagle.wheels( 50 , 50 ) # 앞으로 50%속도 이동
wait(?000) # ?초 기다리기(?000ms)
beagle.wheels( ? , ? )
# 왼쪽 바퀴를 ? % 오른쪽 바퀴를 ? % 로 정하기
wait(?000) # ?초 기다리기(?000ms)
beagle.wheels( 50 , 50 ) # 앞으로 50%속도 이동
wait(?000) # ?초 기다리기(?000ms)
beagle.stop() # 멈추기
# beagle.wheels(0, 0)으로도 멈출수 있습니다.
```

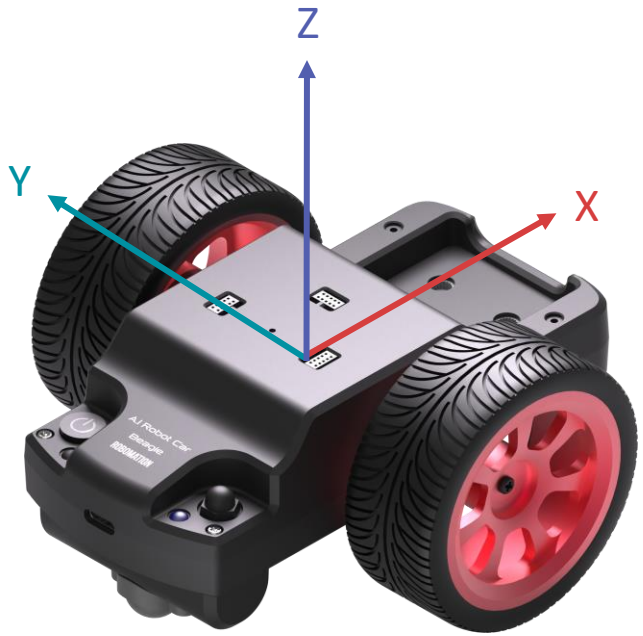
4차시

센서 코드 활용

01

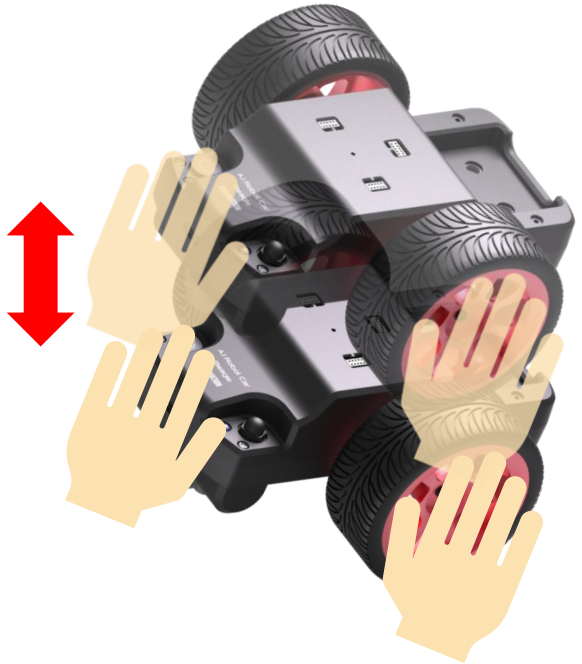
가속도 센서 알아보기

- 비글에는 3축 가속도 센서가 내장되어 있습니다.
- 가속도 센서의 X축은 로봇의 정면 방향이 양수값이고 뒷면 방향이 음수 값입니다. Y축은 왼쪽 방향이 양수 값, 오른쪽 방향이 음수 값이며, Z축은 위쪽 방향이 양수 값, 아래쪽 방향이 음수 값입니다.



01 가속도 센서 알아보기

- 각 축에 대한 가속도 값은 가속도 센서 x (g), 가속도 센서 y (g), 가속도 센서 z (g) 코드를 사용하여 알 수 있습니다.
- 터미널에 센서의 값이 한번에 많이 뜨면 보기 어려울 수 있으니 보고 싶은 센서 값을 하나씩 확인하여 봅시다.
- #print(accel_y_value), #print(accel_z_value) 처럼 print( )안에 들어갈 value 값을 바꾸면서 센서값을 확인 해봅시다..
- 프로그램을 구현하기 전 우선 비글을 들고 움직이며 어떤 방식으로 움직일 때 각 축의 값이 변화하는지 확인합니다.



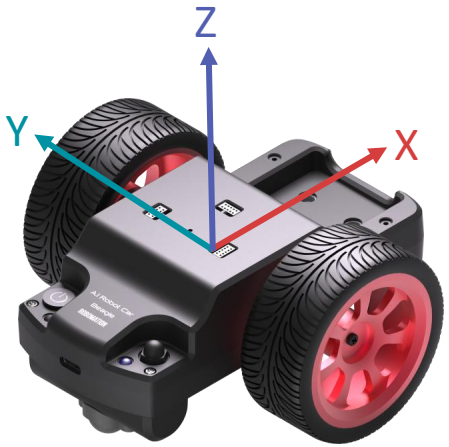
```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:
        time.sleep(0.1) #안정적인 동작을 위해 0.1초 지연시켜주기
        accel_x_value = beagle.accelerometer_x() # X축 가속도 값을 얻는다.
        accel_y_value = beagle.accelerometer_y() # Y축 가속도 값을 얻는다.
        accel_z_value = beagle.accelerometer_z() # Z축 가속도 값을 얻는다.
        print(accel_x_value) # print문으로 변경되는 센서값을 터미널에 지속적으로 출력
        #print(accel_y_value)
        #print(accel_z_value)
        if keyboard.is_pressed("esc"):
                break
```

Value 값을 바꾸면서 센서값 체크

01

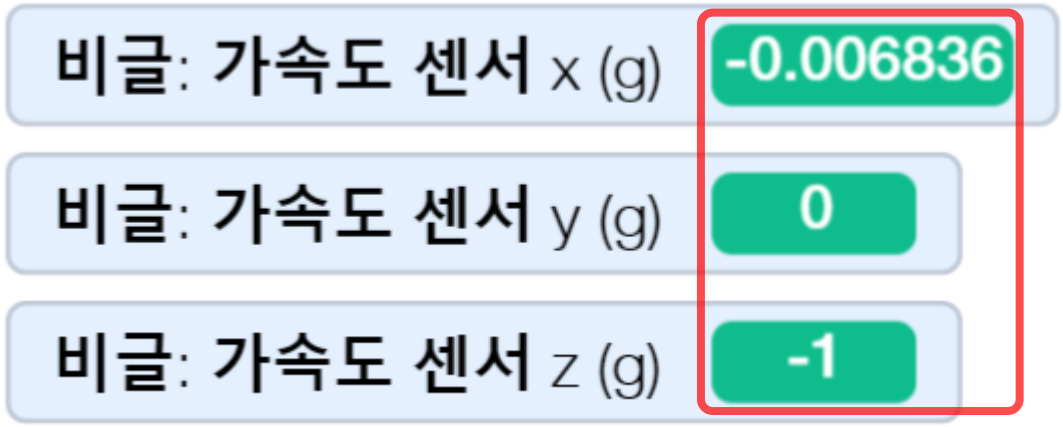
가속도 센서 알아보기

- 비글을 가만히 두면 앞뒤나 좌우로 가해지는 힘이 없으므로 X축 가속도 값과 Y축 가속도 값은 0에 가까운 값을 가집니다.
- Z축의 가속도 방향은 위쪽을 향하지만 비글에 항상 작용하고 있는 중력의 방향은 땅을 향합니다. 그러므로 움직임이 없을 때에도 비글의 Z축 가속도 값은 음수 값을 가지게 되고, 값의 크기는 중력의 값을 1로 하여 -1이 됩니다.



중력 방향

1g

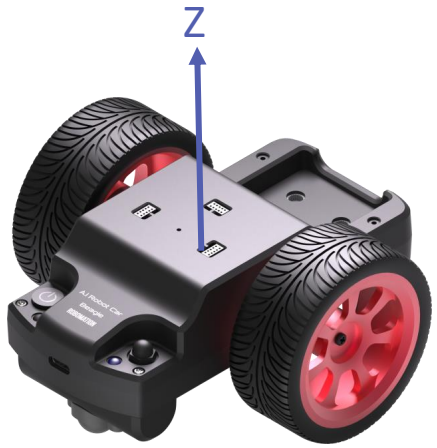


02

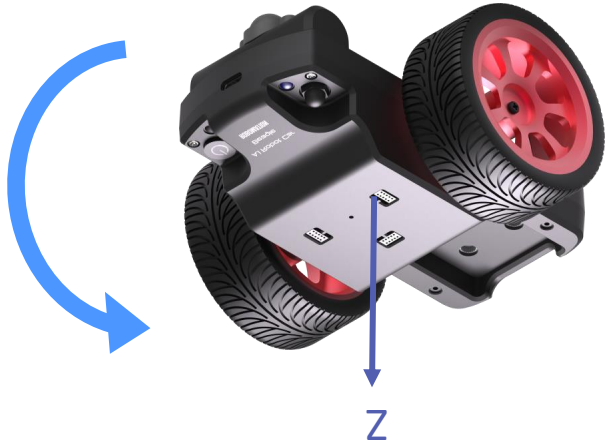
가속도 센서z 값 활용하기

- 이제 가속도 센서의 Z축 값을 활용하여 비글이 노래하는 코드를 작성해 봅니다.
- 비글을 뒤집으면 Z축 가속도 값이 양수 값을 가지며 값의 크기가 1과 같거나 커지는 것을 확인할 수 있습니다.

비글: 가속도 센서 z (g) -1



비글: 가속도 센서 z (g) 1.008789



02

가속도 센서z 값 활용하기

- 앞서 살펴 본 값을 조건문에 활용하면 비글이 뒤집어진 것을 감지할 수 있습니다.
- 가속도 센서 z (g)와 변수 값의 크기를 비교하는데 사용할 수 있는 연산 , 조건의 참과 거짓을 확인할 수 있는 코드를 조합해 Z축 값의 변화를 감지하는 코드를 작성합니다.

가속도 센서 z 사용 (가속도 센서 z의 값은 똑바로 서있는 상태가 -1 이고 뒤집어지면 1 이 됩니다.)

```
accel_z_value = beagle.accelerometer_z() # Z축 가속도 값을 얻는다.
```

만약 (z 가속도 센서 값)이

(0.1) 보다 (작다, 크다, 같다) 면

```
if accel_z_value 〈 0.1:
if accel_z_value 〉 0.1:
if accel_z_value = 0.1:
```

02 가속도 센서z 값 활용하기

- 아래와 같이 코드를 작성하면 비글이 뒤집어져 있을 때를 감지할 수 있습니다. 비글의 상태를 계속해서 감지할 수 있도록 무한 반복하기 코드를 사용했습니다.
- tilt() (기울임) 코드로도 뒤집어 진 상태를 확인 할 수 있습니다. tilt() 값이 3이면 거꾸로 뒤집은 상태입니다.
- 이제 조건문 안이나 거꾸로 뒤집었을 때 원하는 동작 코드를 삽입하면 비글이 뒤집어졌을 때 해당 동작을 수행합니다.
- 연주하기 코드를 활용하여 비글이 뒤집어 졌을 때 노래를 연주하는 코드를 작성해 봅니다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        accel_z_value = beagle.accelerometer_z() # Z축 가속도 값을 얻는다.
        if accel_z_value  >= 0.1:   # 만약 가속도 센서 z의 값이 0.1보다 크거나 같다면(똑바로 서면 -1, 뒤집어 지면 1이 된다.)
                beagle.note("C4", 1) # 4옥타브 도(C)의 소리를 재생한다
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

코드 작성

02

가속도 센서z 값 활용하기

- 이번에는 소리 코드를 이용하여 노래를 만들어 봅시다

예시

```
from roboid import *
beagle = Beagle()
beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
beagle.note("A4", 0.5) # 4옥타브 시(A)의 소리를 재생한다
beagle.note("A4", 0.5) # 4옥타브 시(A)의 소리를 재생한다
beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
beagle.note("F4", 0.5) # 4옥타브 파(F)의 소리를 재생한다
beagle.note("F4", 0.5) # 4옥타브 파(F)의 소리를 재생한다
beagle.note("E4", 0.5) # 4옥타브 미(E)의 소리를 재생한다
beagle.note("E4", 0.5) # 4옥타브 미(E)의 소리를 재생한다
beagle.note("D4", 0.5) # 4옥타브 레(D)의 소리를 재생한다
beagle.note("D4", 0.5) # 4옥타브 레(D)의 소리를 재생한다
beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
```

02 가속도 센서z 값 활용하기

- 만든 노래 코드를 가속도 센서 z를 이용하여 뒤집어졌을 때 노래하는 비글 로봇 코드를 작성하여 보세요.

예시

```
from roboid import *
beagle = Beagle()
beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
beagle.note("A4", 0.5) # 4옥타브 시(A)의 소리를 재생한다
beagle.note("A4", 0.5) # 4옥타브 시(A)의 소리를 재생한다
beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
beagle.note("F4", 0.5) # 4옥타브 파(F)의 소리를 재생한다
beagle.note("F4", 0.5) # 4옥타브 파(F)의 소리를 재생한다
beagle.note("E4", 0.5) # 4옥타브 미(E)의 소리를 재생한다
beagle.note("E4", 0.5) # 4옥타브 미(E)의 소리를 재생한다
beagle.note("D4", 0.5) # 4옥타브 레(D)의 소리를 재생한다
beagle.note("D4", 0.5) # 4옥타브 레(D)의 소리를 재생한다
beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
```

코드 작성

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:    # 무한반복
        time.sleep(0.1)
        accel_z_value = beagle.accelerometer_z()
        if accel_z_value  >= 0.1:

                #노래 코드 삽입


        if keyboard.is_pressed("esc"):
                break    # 반복문 종료
```

03

가속도 센서x, y 값 활용하기

- 이번엔 가속도 센서x, y의 값을 활용한 코드를 만들어 봅니다.
- 비글의 앞이나 뒤로 힘을 가해서 로봇이 움직이면 X축 방향으로 가속도가 작용해 값이 변합니다.

비글: 가속도 센서 x (g) -0.005859

비글: 가속도 센서 x (g) 0.317383

03

가속도 센서x, y 값 활용하기

- 비글의 왼쪽이나 오른쪽으로 힘이 가해져 로봇이 움직이면 Y축 방향으로 가속도가 작용해 Y축 가속도 값이 변화합니다.

비글: 가속도 센서 y (g) 0.000977

비글: 가속도 센서 y (g) 0.180664

03

가속도 센서x, y 값 활용하기

- 먼저 가속도 센서 x 값의 변화를 감지하여 삐 소리를 내도록 합니다.
- 가속도 센서 x (g)블록과 변수 값의 크기를 비교하는데 사용할 수 있는 연산 코드, 조건의 참과 거짓을 확인할 수 있는 코드를 조합해 x축 값의 크기 변화를 감지하는 코드를 작성합니다.

가속도 센서 x 사용.

```
accel_x_value = beagle.accelerometer_x() # x축 가속도 값을 얻는다.
```

만약 (x 가속도 센서 값)이

(0) 보다 (작다, 크다, 같다) 면

```
if accel_x_value 〈 0:
if accel_x_value 〉 0:
if accel_x_value = 0:
```

03 가속도 센서x, y 값 활용하기

- 비글의 후면을 손가락으로 툭 치면 이를 감지하여 삐 소리를 내는 코드를 작성해 봅시다.
- x 가속도 센서의 값을 한번 확인해보고 손가락으로 로봇을 톡 치면 값이 어떤 식으로 변하는지 확인해봅니다.
- 확인 한 센서의 값에 맞게 코드를 수정하면서 원하는 동작을 하도록 해봅시다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        accel_x_value = beagle.accelerometer_x() # x축 가속도 값을 얻는다.
        if accel_x_value <= 0:   # 만약 가속도 센서 x의 값이 0보다 작거나 같다면
                beagle.sound("BEEP", 1) # 삐 소리를 재생한다
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

03 가속도 센서x, y 값 활용하기

- 이번엔 어느 방향으로든 로봇을 툭 치면 소리를 내도록 코드를 작성해 봅니다.
- 힘이 작용하는 방향에 관계없이 값이 바뀌기만 해도 삐 소리가 날 수 있도록 해야 하므로 숫자의 절댓값에 따라 판단할 수 있는 코드를 활용합니다.

가속도 센서 x, y 사용.

```
accel_x_value = beagle.accelerometer_x() # x축 가속도 값을 얻는다.
accel_y_value = beagle.accelerometer_y() # y축 가속도 값을 얻는다.
```

만약 (x 가속도 센서 값)이
(0) 보다 (작다, 크다, 같다) 면

```
if accel_x_value < 0:
if accel_x_value > 0:
if accel_x_value = 0:
```

만약 (y 가속도 센서 값)이
(0) 보다 (작다, 크다, 같다) 면

```
if accel_y_value < 0:
if accel_y_value > 0:
if accel_y_value = 0:
```

03 가속도 센서x, y 값 활용하기

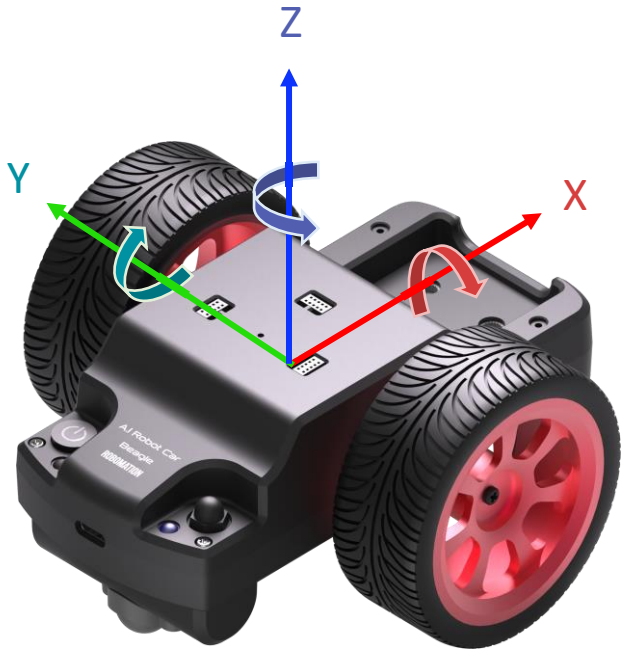
- 아래와 같이 코드를 작성하고 비글을 손가락으로 툭 치면 어느 방향에서 비글에 힘을 가하더라도 이를 감지하여 삐 소리를 냅니다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        accel_x_value = beagle.accelerometer_x() # x축 가속도 값을 얻는다.
        accel_y_value = beagle.accelerometer_y() # y축 가속도 값을 얻는다.
        if accel_x_value <= 0 or accel_y_value >= 0 :
        # 만약 가속도 센서 x 의 값이 0보다 작거나 같다면 또는 가속도 센서 y의 값이 0보다 크거나 같다면
                beagle.sound("BEEP", 1) # 삐 소리를 재생한다
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

04

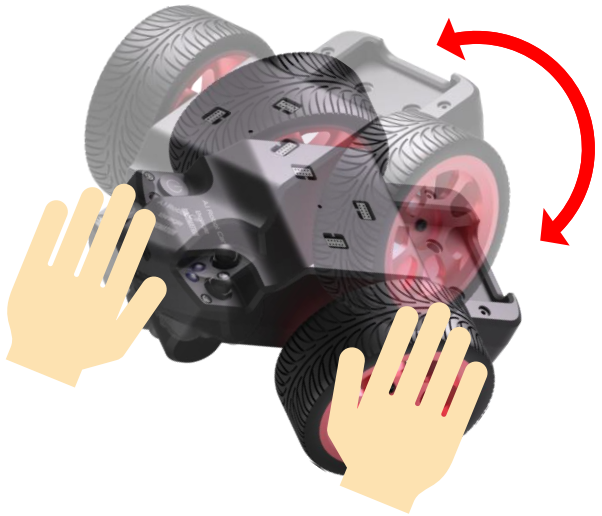
자이로 센서 알아보기

- 비글에는 각 축에 대한 각속도의 값을 감지하는 자이로 센서가 내장되어 있습니다.
- X축은 비글을 오른쪽으로 기울일 때 양수 값, 왼쪽으로 기울일 때 음수값입니다. Y축은 앞으로 기울일 때 양수 값, 뒤로 기울일 때 음수 값입니다. Z축은 비글을 평행하게 놓고 반시계 방향으로 돌리면 양수 값, 시계 방향으로 돌리면 음수 값입니다.



04 자이로 센서 알아보기

- 각 축에 대한 각속도 값은 자이로 센서 x (°/s), 자이로 센서 y (°/s), 자이로 센서 z (°/s) 블록을 사용하여 알 수 있습니다. 터미널에서 각 축의 각속도 값을 실시간으로 확인할 수 있습니다.
- 프로그램을 구현하기 전에 우선 비글을 들고 움직이며 어떤 방식으로 움직일 때 각 축의 값이 변화하는지 확인합니다.



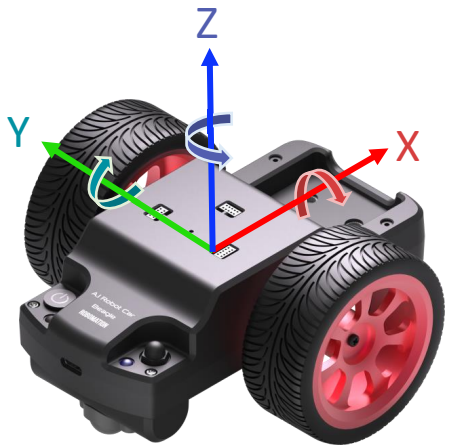
```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:
        time.sleep(0.1) #안정적인 동작을 위해 0.1초 지연시켜주기
        gyro_x_value = beagle.gyroscope_x()# X축 자이로 센서 값을 얻는다.
        gyro_y_value = beagle.gyroscope_y()# Y축 자이로 센서 값을 얻는다.
        gyro_z_value = beagle.gyroscope_z() # Z축 자이로 센서 값을 얻는다.
        print(gyro_x_value) # print문으로 변경되는 센서값을 터미널에 지속적으로 출력
        #print(gyro_y_value)
        #print(gyro_z_value)
        if keyboard.is_pressed("esc"):
                break
```

Value 값을 바꾸면서 센서값 체크

04

자이로 센서 알아보기

- 비글을 가만히 두면 움직임으로 인한 각속도의 변화가 없으므로 X축, Y축, Z축의 각속도 값은 0에 가까운 값을 가집니다.



비글: 자이로 센서 x (°/s) 0.022888

비글: 자이로 센서 y (°/s) -0.190735

비글: 자이로 센서 z (°/s) 0.183105

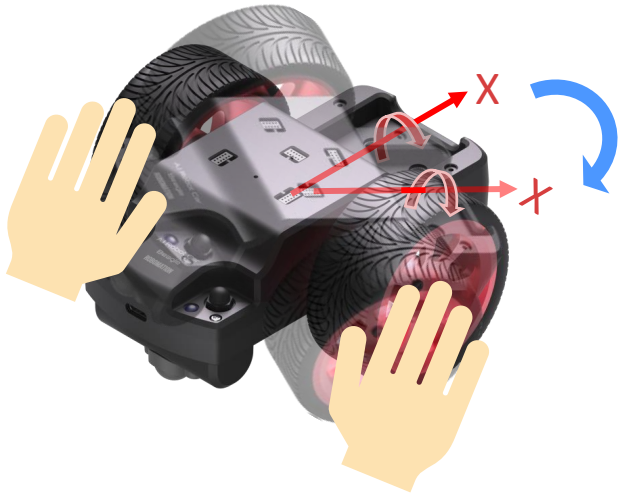
04

자이로 센서 알아보기

- 비글을 왼쪽이나 오른쪽으로 기울이는 순간 X축의 각속도 값이 변화합니다.

비글: 자이로 센서 x (°/s) -0.068665

비글: 자이로 센서 x (°/s) 249.992371



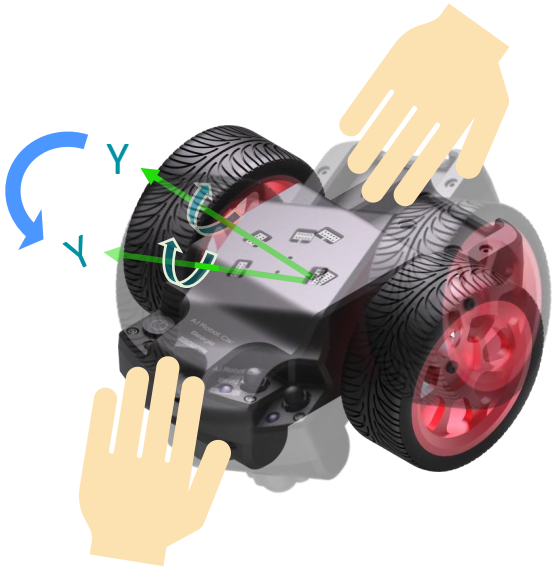
04

자이로 센서 알아보기

- 비글을 앞쪽이나 뒤쪽으로 기울이는 순간 Y축의 각속도 값이 변화합니다.

비글: 자이로 센서 y (°/s) 0.106812

비글: 자이로 센서 y (°/s) -156.12793

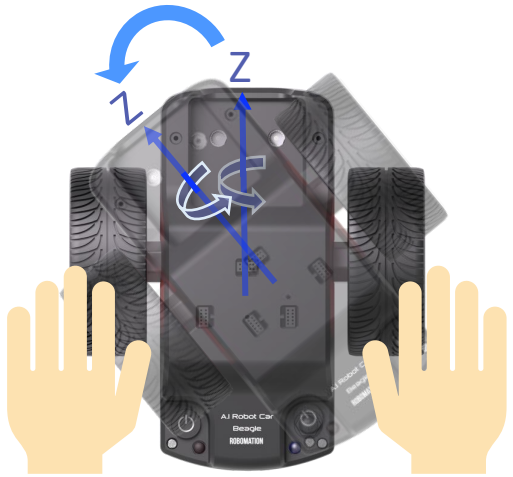


04 자이로 센서 알아보기

- 비글을 평평한 곳에 올려놓고 시계 방향이나 반시계 방향으로 돌리는 순간 Z축의 각속도 값이 변화합니다.

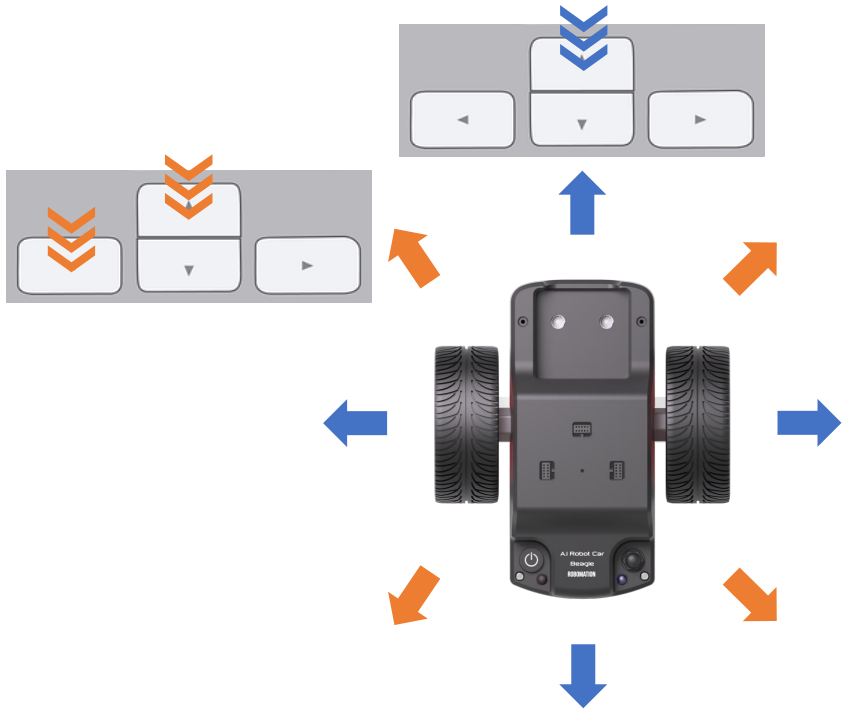
비글: 자이로 센서 z (°/s) 0.228882

비글: 자이로 센서 z (°/s) 249.504089



04 자이로 센서 알아보기

- 이제 자이로 센서로 비글의 난폭 운전을 감지하는 코드를 작성해 봅니다.
- 먼저 방향키를 눌러 이동하는 코드를 활용하여 봅시다
- 방향키를 누르고 있으면 해당 방향으로 비글이 움직입니다.
  주황색 화살표처럼 대각선으로 이동하기 위해서는
  두 방향의 키를 동시에 누르면 됩니다.



```
while True:
    time.sleep(0.1) # 안정적인 동작을 위한 0.1초 간격으로 움직이도록 설정
    if keyboard.is_pressed("up"): #위 키를 눌렀을때
        if keyboard.is_pressed("right"): # 위 + 오른쪽
            beagle.wheels(40, 15)
        elif keyboard.is_pressed("left"): # 위 + 왼쪽
            beagle.wheels(15, 40)
        else:
            beagle.wheels(20, 20) # 위 키만 눌렀을때 앞으로

    elif keyboard.is_pressed("down"): # 아래 키 눌렀을때
        if keyboard.is_pressed("right"): # 아래 + 오른쪽
            beagle.wheels(-40, -15)
        elif keyboard.is_pressed("left"): # 아래 + 왼쪽
            beagle.wheels(-15, -40)
        else:
            beagle.wheels(-20, -20) # 아래 키만 눌렀을때 뒤로

    elif keyboard.is_pressed("right"): # 오른쪽 키 눌렀을때
        if keyboard.is_pressed("up"): # 오른쪽 + 위
            beagle.wheels(40, 15)
        elif keyboard.is_pressed("down"): # 오른쪽 + 아래
            beagle.wheels(-40, -15)
        else:
            beagle.wheels(20, -20) # 오른쪽 키만 눌렀을때 오른쪽 회전

    elif keyboard.is_pressed("left"): # 왼쪽 키를 눌렀을때
        if keyboard.is_pressed("up"): # 왼쪽 + 위
            beagle.wheels(15, 40)
        elif keyboard.is_pressed("down"): # 왼쪽 + 아래
            beagle.wheels(-15, -40)
        else:
            beagle.wheels(-20, 20) #왼쪽 키만 눌렀을때 왼쪽 회전

    elif keyboard.is_pressed("esc"): # 키보드 esc키를 누르면 종료
        break
    else:
        beagle.stop() # 아무 키도 누르고 있지 않으면 멈추기
```

04 자이로 센서 알아보기

- 코드의 바퀴 값을 바꾸면 속도를 더욱 빠르거나 느리게 할 수 있습니다.

예시 코드:

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 반복문인 while문을 이용하여 지속적으로 변하는 센서값을 받기위해 사용
        time.sleep(0.1)  # 0.1초단위로 반응하게 하여 안정적인 동작을 구현
        if keyboard.is_pressed("up"): # 키보드 방향키 위를 눌렀을때
                beagle.wheels(20,20)  # beagle 로봇 앞으로 이동
        elif keyboard.is_pressed("down"):  # 키보드 방향키 아래를 눌렀을때
                beagle.wheels(-20,-20)  # beagle 로봇 뒤로 이동
        elif keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

04 자이로 센서 알아보기

- 자이로 센서 y (°/s)값의 변화를 감지하여 삐 소리를 2번 내도록 합니다.
- 자이로 센서 y (°/s)코드와 변수 값의 크기를 비교하는데 사용할 수 있는 연산 코드, 조건의 참과 거짓을 확인할 수 있는 코드를 조합해 y축 각속도 변화를 감지하는 코드를 작성합니다.
- 센서 값을 미리 확인하고 그 값에 따라 설정해봅시다.

```
gyro_y_value = beagle.gyroscope_y()# Y축 자이로 센서 값을 얻는다.

beagle.sound("BEEP", 2) # 삐 소리를 2번 재생한다
```

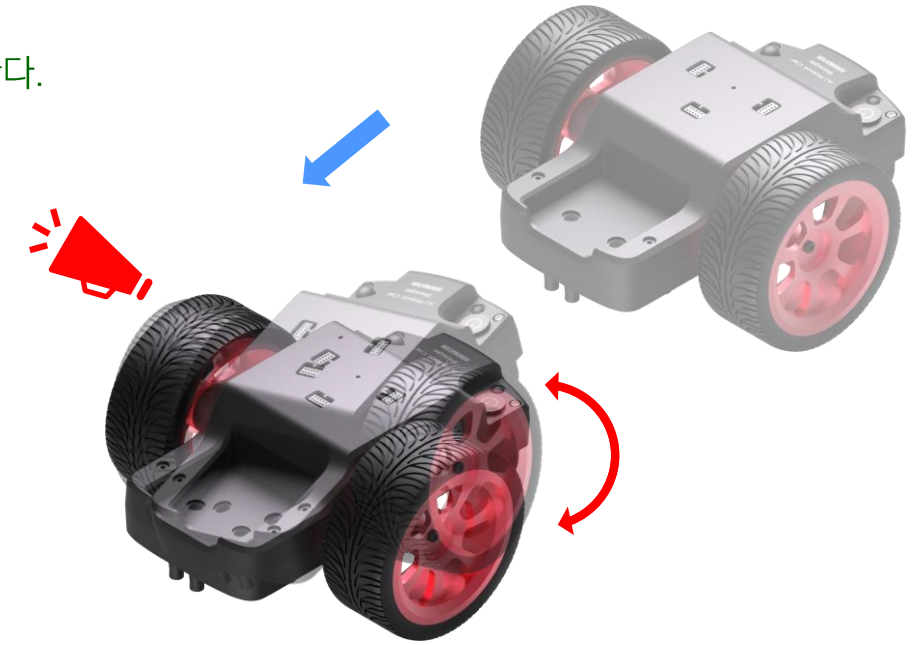
만약 (　　)이
(0) 보다 (작다, 크다, 같다) 면

```
If (　　) 〈 0:
if (　　) 〉 0:
if (　　) = 0:
```

04 자이로 센서 알아보기

- 아래와 같이 코드를 작성하고 비글을 앞쪽이나 뒤쪽으로 기울이면 이를 감지하여 삐 소리를 냅니다.
- 이 코드로 비글이 빠르게 이동하다 급정거할 때 뒷부분이 살짝 들리는 것을 감지합니다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        gyro_y_value = beagle.gyroscope_y() # y축 자이로 센서 값을 얻는다.
        if gyro_y_value >= 50 or gyro_y_value <= -50 :
        # 자이로 센서 y의 값이 앞으로 기울어지거나( 50 이상) 뒤로 기울어지면(-50 이하)
                beagle.sound("BEEP", 2) # 삐 소리를 재생한다
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```



04

자이로 센서 알아보기

- 자이로 센서 z (°/s)값의 변화를 감지하여 사이렌 소리를 내도록 합니다.
- 자이로 센서 z (°/s)코드와 변수 값의 크기를 비교하는데 사용할 수 있는 연산 코드, 조건의 참과 거짓을 확인할 수 있는 코드를 조합해 z축 각속도 변화를 감지하는 코드를 작성합니다.
- 힘이 작용하는 방향에 관계없이 값이 바뀌기만 해도 소리가 나야 하므로 숫자의 절댓값에 따라 판단할 수 있는 코드를 활용합니다.

```
gyro_z_value = beagle.gyroscope_z()# z축 자이로 센서 값을 얻는다.

beagle.sound("SIREN", 1) # 사이렌 소리를 1번 재생한다
```

만약 ( )이
(0) 보다 (작다, 크다, 같다) 면

```
If (  ) 〈 0:
if (  ) 〉 0:
if (  ) = 0:
```

04 자이로 센서 알아보기

- 아래와 같이 코드를 작성하고 비글을 빠른 속도로 왼쪽이나 오른쪽 돌리면 이를 감지하여 사이렌 소리를 냅니다.
- 이 코드로 비글이 빠른 속도로 좌우로 회전하는 것을 감지합니다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        gyro_z_value = beagle.gyroscope_z() # z축 자이로 센서 값을 얻는다.
        if gyro_z_value >= 50 or gyro_z_value <= -50 :
        # 자이로 센서 z의 값이 앞으로 기울어지거나( 50 이상) 뒤로 기울어지면(-50 이하)
                beagle.sound("SIREN", 1) # 삐 소리를 재생한다
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

05

자이로 센서 알아보기

- 키보드로 조종하는 코드에 아래 코드를 더한 후 비글을 빠른 속도로 움직이도록 조종하면 비글이 난폭하게 운전하는지를 감지할 수 있습니다.
- 비글을 조종할 때 변화하는 자이로 센서 값을 관찰하고 각자가 정한 규칙과 상황에 맞게 입력 값을 바꿔 봅니다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        gyro_y_value = beagle.gyroscope_y() # y축 자이로 센서 값을 얻는다.
        if gyro_y_value >= 50 or gyro_y_value <= -50 :
        # 자이로 센서 y의 값이 앞으로 기울어지거나( 50 이상) 뒤로 기울어지면(-50 이하)
                beagle.sound("BEEP", 2) # 삐 소리를 재생한다
        gyro_z_value = beagle.gyroscope_z() # z축 자이로 센서 값을 얻는다.
        if gyro_z_value >= 50 or gyro_z_value <= -50 :
        # 자이로 센서 z의 값이 앞으로 기울어지거나( 50 이상) 뒤로 기울어지면(-50 이하)
                beagle.sound("SIREN", 1) # 삐 소리를 재생한다

        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```

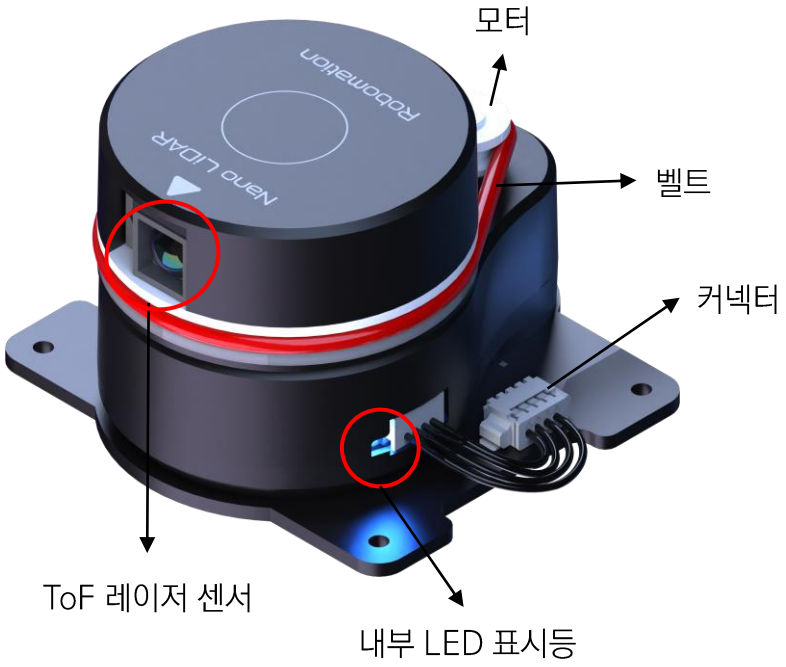
5차시

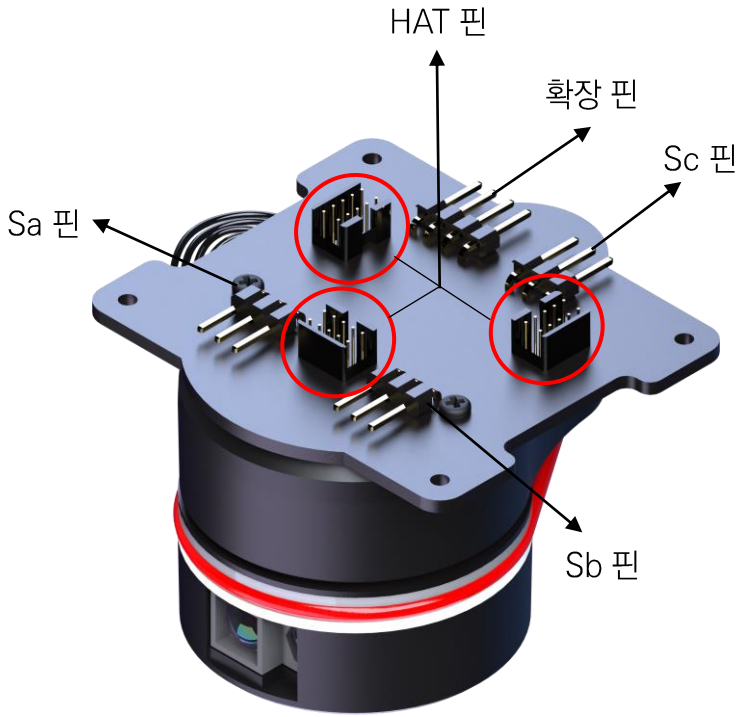
라이다 코드 활용

01

라이다 센서 알아보기

- 나노라이다는 다음 그림과 같이 다양한 장치를 포함하고 있습니다.

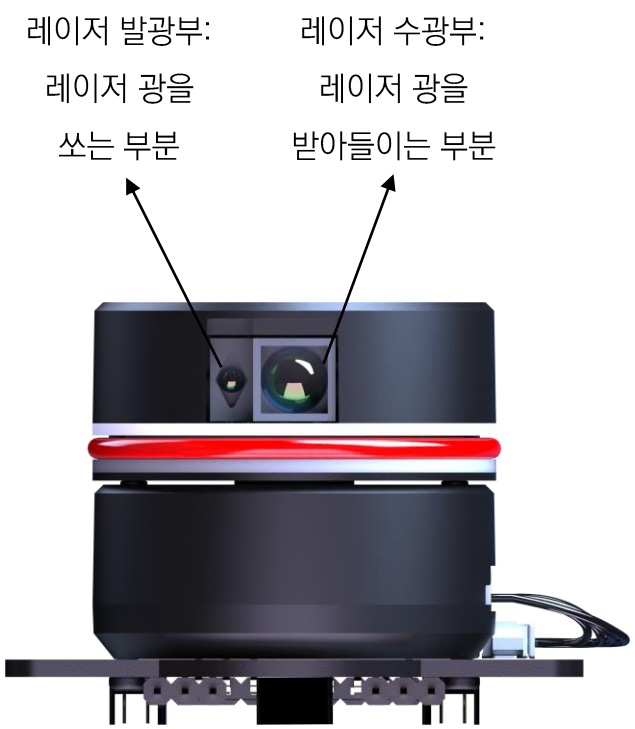


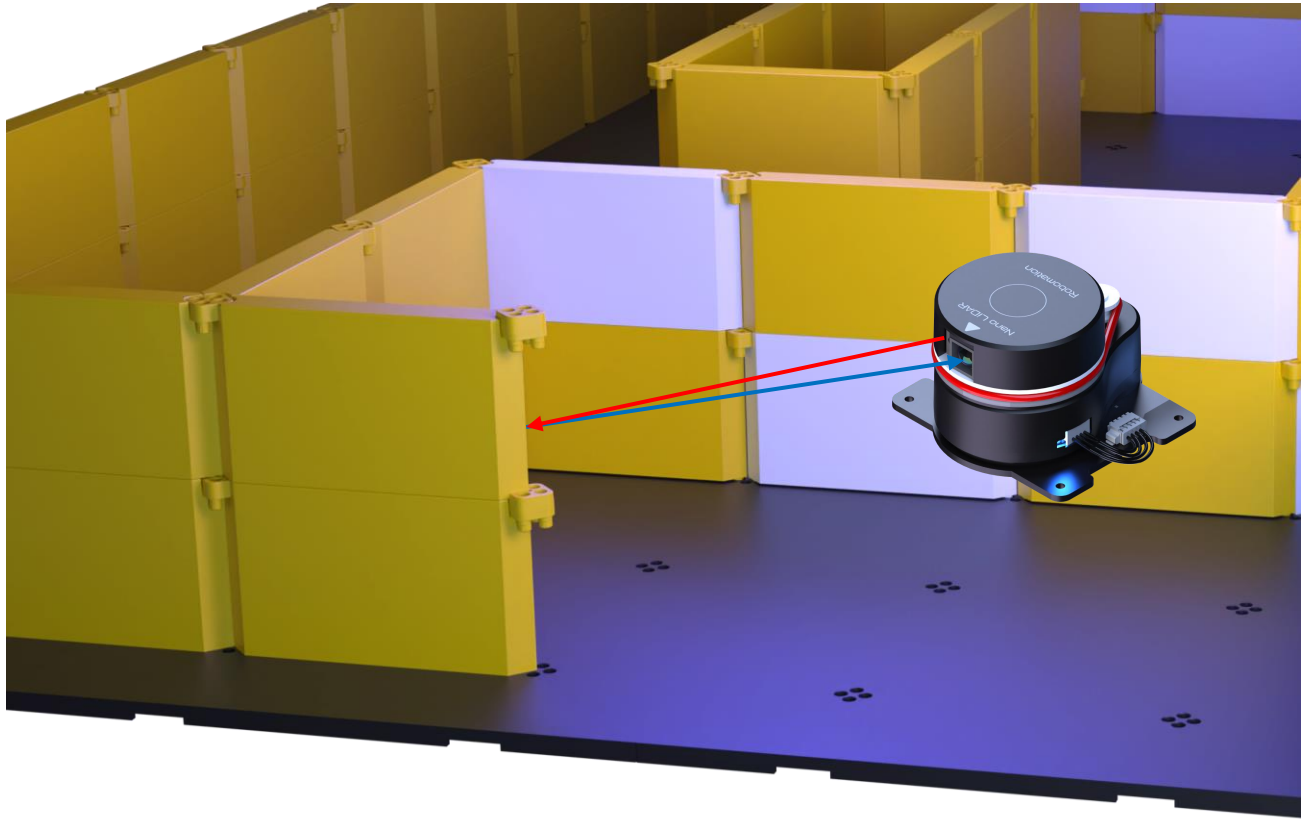


01

라이다 센서 알아보기

- 라이다는 레이저 광을 비추어 대상물에 닿아 되돌아 올 때까지의 시간차를 계측하여, 거리나 위치, 형상을 측정합니다.

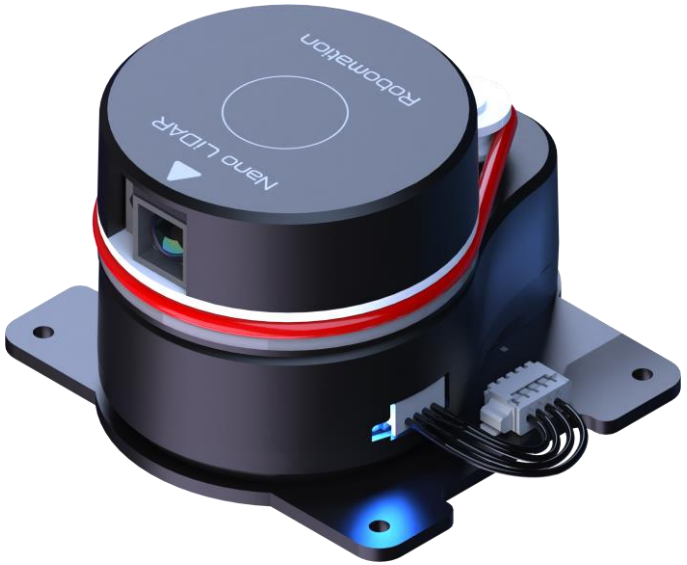




01

라이다 센서 알아보기

- 로보메이션의 나노라이다 사양은 다음과 같습니다.



항 목	사양
크기	가로37 x 세로52 x 높이 33 mm
무게	50g
측정 거리	50mm~5000mm(White objct) 50mm~3000mm(Black objct)
측정 범위	360˚
각도 해상도	1
측정 주기	5Hz
레이저 파장	850mm
레이저 안전	Class 1
거리 상대 오류	5%
상태 알림	LED x 1
동작 온도	10˚~50˚

01 라이다 센서 알아보기

- 비글의 HAT 장착부 포트에 맞춰 나노라이다의 HAT 핀을 꽂아줍니다.

01

라이다 센서 알아보기

- 비글의 전원을 켜 나노라이다에 전원이 공급되면 초기 세팅을 위해 라이다가 잠시 동안 회전하며 초기화를 합니다.
- 라이다는 반시계 방향으로 회전합니다.
- 초기화가 완료되면 라이다가 움직임을 멈추고 대기합니다.

전원을 켜 LED에 불이 들어온 모습입니다.

01

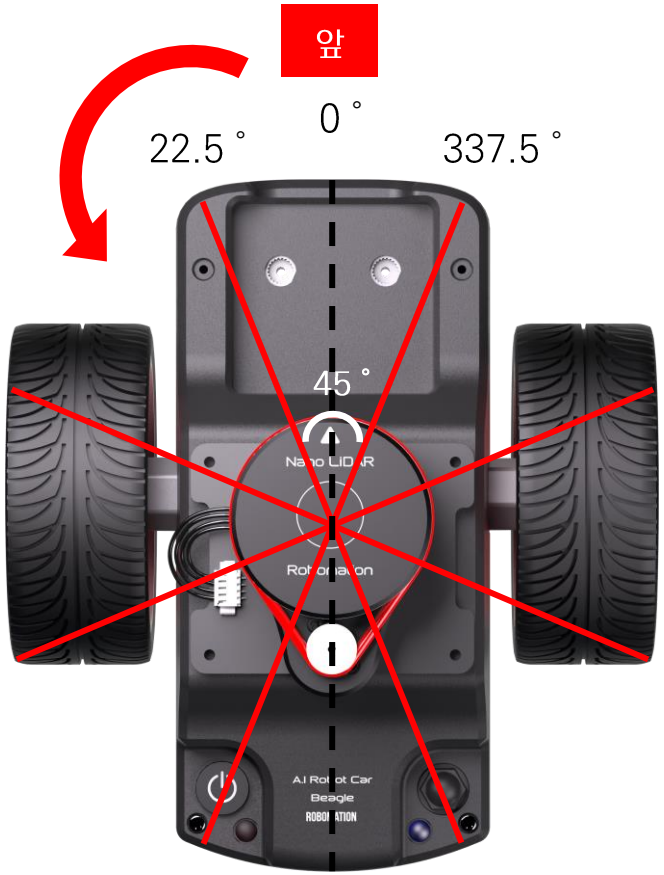
라이다 센서 알아보기

- 라이다 측면 LED 동작으로 라이다의 상태를 알 수 있습니다.

LED 동작	라이다 상태
켜짐	초기화 성공/대기 상태
느리게 깜빡임	초기화 실패
빠르게 깜빡임	데이터 측정 중

01

라이다 센서 알아보기



- 나노라이다는 반시계 방향으로 회전합니다.
- 360도를 45도씩 나누어 8등분(앞, 뒤, 왼쪽, 오른쪽, 왼쪽 앞, 오른쪽 앞, 왼쪽 뒤, 오른쪽 뒤)한 구간의 라이다 값을 활용합니다.
- 예를 들어 앞 방향의 라이다 값은 22.5도~337.5도 사이에서 측정된 각 각도 라이다 값의 평균값을 나타냅니다. 다른 구간도 각 방향의 45도 범위 내 값의 평균 값을 해당 구역의 값으로 합니다.

01

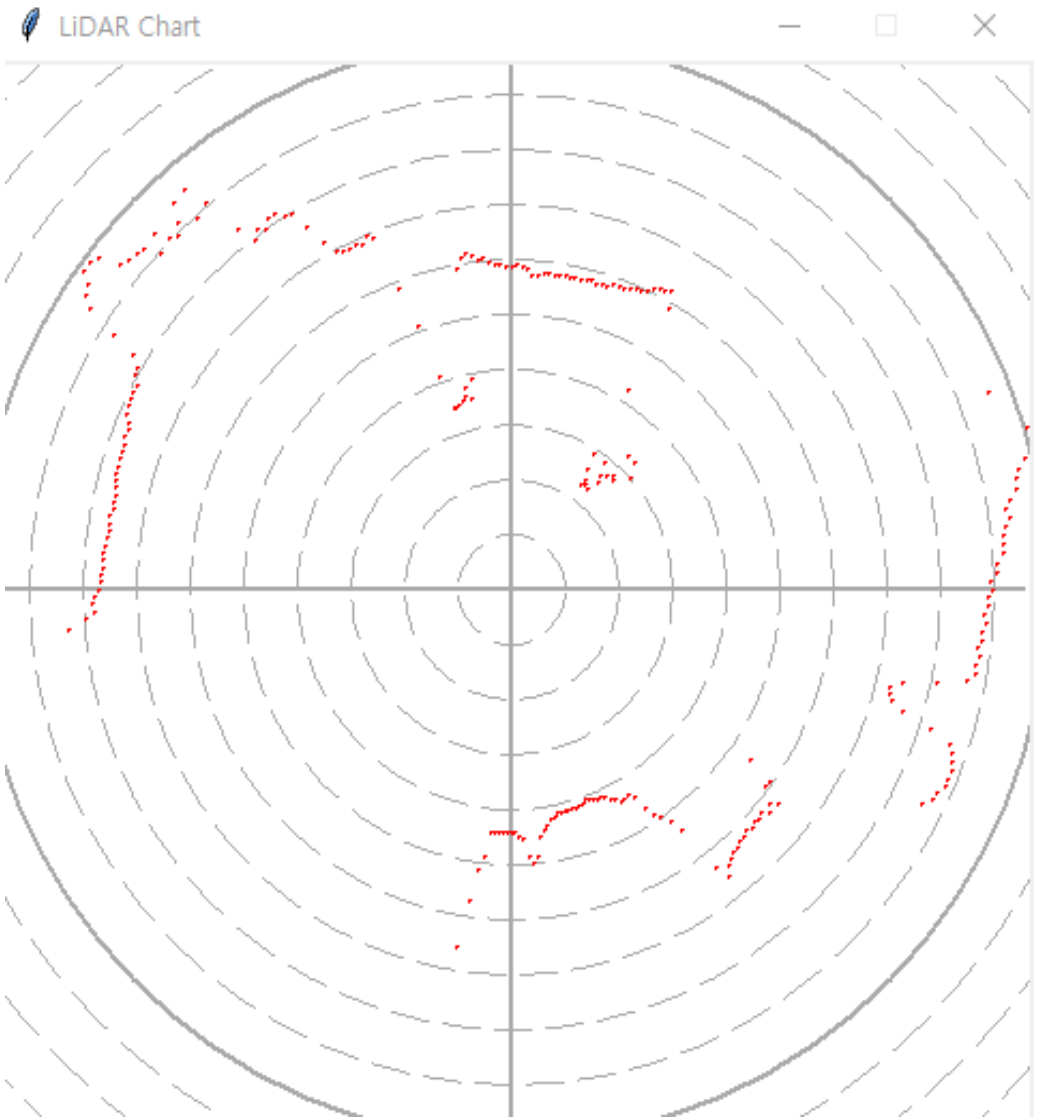
라이다 센서 알아보기

- 거리측정에서의 최소 출력 거리는 50mm로 라이다 가장자리와 대상 사이의 거리가 50mm 이상일 수 있도록 합니다.
- 50mm 이상이더라도 가까운 거리의 측정 값은 정확도가 떨어질 수 있습니다.
- 측정 대상물의 재질과 색에 따라 같은 거리에서 측정한 라이다 값도 크기가 다르게 나타날 수 있습니다.

01 라이다 센서 알아보기

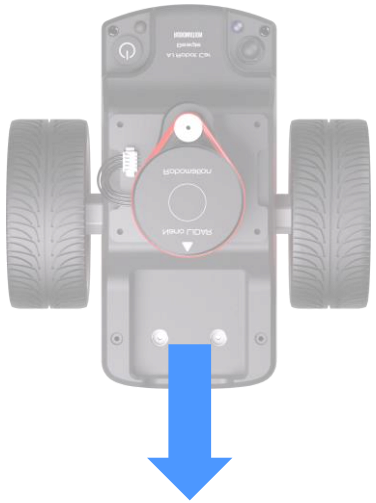
- 우선 라이다 센서가 잘 작동하는지 라이다 센서 확인 코드를 이용하여 확인해 봅시다.
  (라이다가 돌아가기 시작하고 센서 값을 받게 되기까지 잠시 시간이 걸립니다.)
- 라이다가 돌아가면 주위에 벽이나 장애물이 있으면 점들로 표시되어 주위 상태를 확인합니다.
- 라이다 차트를 통해 주변 사물을 확인해보고 라이다 센서 값을 비교하면서 기록하여 봅시다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
beagle.start_lidar() # 비글 로봇의 라이다를 시작한다.
beagle.lidar_chart()
# 라이다 센서 값들을 차트로 보여준다. 이를 통해 주변 사물들을 인식 할 수 있다.
while True:
        time.sleep(0.1)
        front_lidar_value= beagle.front_lidar()# 앞 영역의 라이다 값을 얻는다.
        rear_lidar_value= beagle.rear_lidar()# 뒤 영역의 라이다 값을 얻는다.
        print(front_lidar_value)
        #print 문 안에 원하는 value값을 바꿔보면서 터미널에서 값을 확인해보자
        if keyboard.is_pressed("esc"):  # 키보드 esc 를 눌렀을때
                break   # 반복문 종료
```



02 라이다 센서 활용하기

- 라이다 값을 이용하면 사물을 감지 할 수 있습니다. 앞서 살펴본 라이다 값을 활용하여 다양한 동작을 구현하여 봅시다.
- 라이다 센서를 활용해서 사물을 인식하면 따라가고 너무 가까워 지면 경고와 함께 멈추는 동작을 구현하여 봅시다.

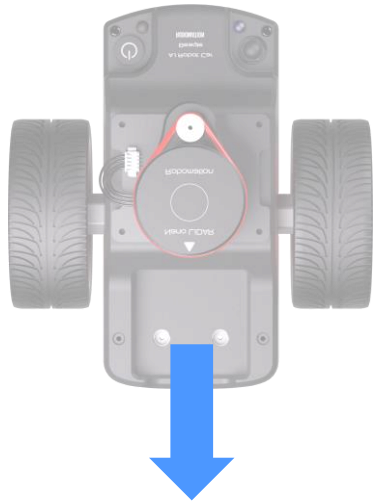


```
beagle.start_lidar() # 비글 로봇의 라이다를 시작한다.

lidar_front_value = beagle.front_lidar() # 비글로봇의 앞쪽 라이다 센서 값을 받는다.

 if lidar_front_value <= ? :    # 만약 전방 라이다 값이 ? 보다 작아지거나 같아지는 경우


  beagle.sound("BEEP", 1)

  beagle.wheels(20,20)

  beagle.stop()
```

02 라이다 센서 활용하기

- 전방에 사물을 감지하고 앞으로 이동하고 너무 가까워 지면 멈추면서 소리를 내려면 라이다 센서를 어떻게 사용해야 하는지 생각해봅시다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
beagle.start_lidar() # 비글 로봇의 라이다를 시작한다.
beagle.lidar_chart()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        lidar_front_value = beagle.front_lidar() # 비글로봇의 앞쪽 라이다 센서 값을 받는다.
        print(lidar_front_value)
        if lidar_front_value < 500 and lidar_front_value >= 80 :
        # 라이다 센서 값이 500미만(전방에 사물 감지)이고 80보다 크거나 같으면(너무 가깝지 않은 상황)
                beagle.wheels(20,20) # 앞으로 이동
        elif lidar_front_value < 80 : # 전방 라이다 값이 80 이하이면(너무 가까우면)
                beagle.sound("BEEP", 1)
                beagle.stop() # 삐 소리를 내면서 멈춘다.
        else: # 비글로봇 전방 센서 값이 500 보다 이상이면(전방에 사물을 인식 못하면)
                beagle.stop() # 비글 로봇은 안움직인다.
```



02 라이다 센서 활용하기

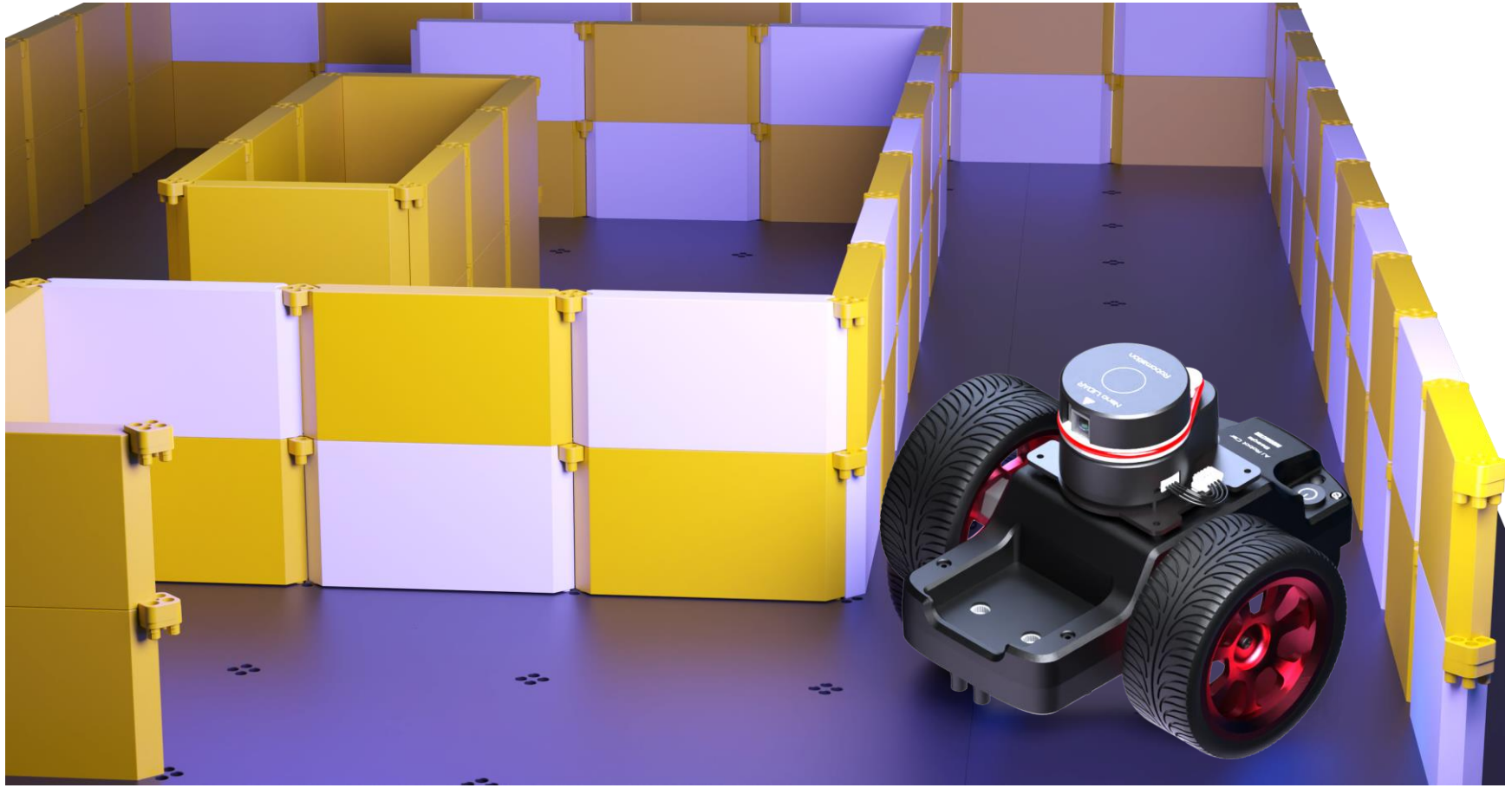
- 이번에는 라이다 센서를 이용하여 비글 로봇 뒤에 사물이 감지되면 노래가 나오도록 구상해 봅시다.

```
from roboid import *
import time
import keyboard
beagle = Beagle()
beagle.start_lidar() # 비글 로봇의 라이다를 시작한다.
beagle.lidar_chart()
while True:   # 무한반복
        time.sleep(0.1)  # 0.1초의 딜레이를 주어 안정적으로 센서 값을 0.1초 간격으로 받도록 한다.
        lidar_rear_value = beagle.rear_lidar() # 비글로봇의 뒤쪽 라이다 센서 값을 받는다.
        print(lidar_rear_value)
         if lidar_rear_value <= 80 :   # 만약 비글로봇 뒤쪽 라이다 센서 값이 80보다 작거나 같다면(가까우면)
                beagle.note("C4", 0.5) # 4옥타브 도(C)의 소리를 재생한다
                beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
                beagle.note("G4", 0.5) # 4옥타브 솔(G)의 소리를 재생한다
                beagle.note("A4", 0.5) # 4옥타브 시(A)의 소리를 재생한다
                #이런 식으로 자신이 원하는 노래를 만들어서 추가하여봅시다.
```

02

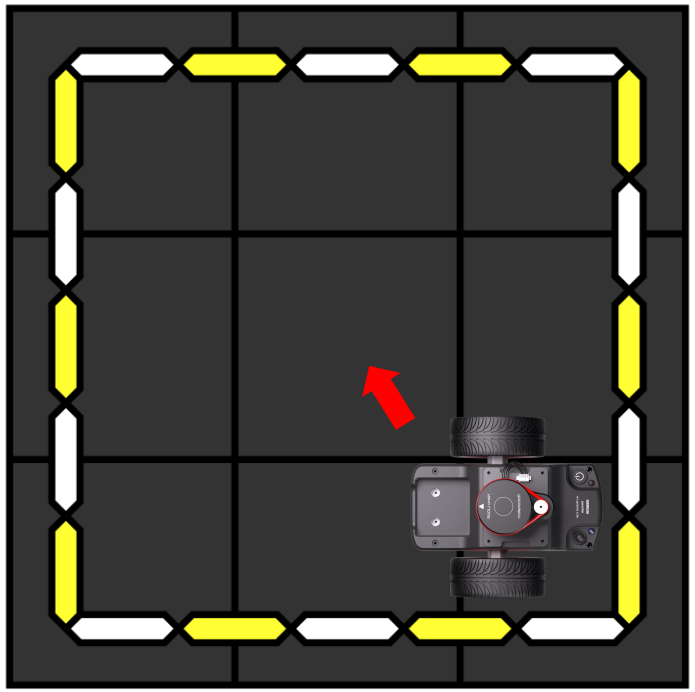
라이다 센서 활용하기

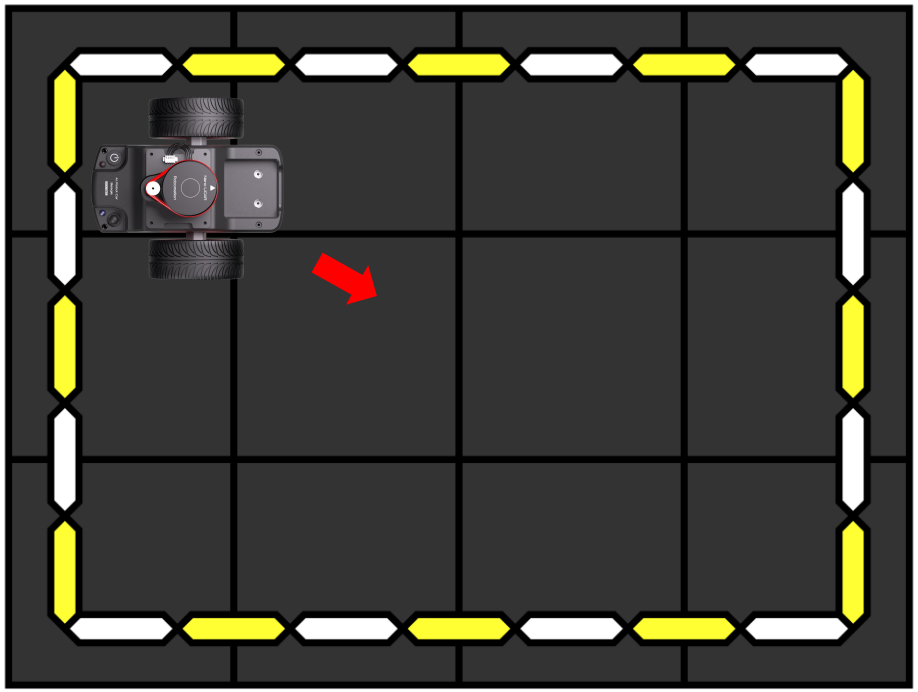
- 마지막으로 지금까지 사용했던 비글 로봇 움직이기, 소리 내기, 키보드 움직이기, 센서 활용 방법들을 이용하여 자신만의 비글 로봇 활용을 구현하여 봅시다. AI DRIVING TRACK(비글용 미로판)도 활용해 볼 수 있습니다.



02 라이다 센서 활용하기

- 추가로 주어진 beagle_test 파일을 활용해 비글을 동작해 볼 수 있습니다.
- 1~7번 파이썬 파일로 앞서 설명한 기능을 실행해 볼 수 있습니다.
- beagle_test_lidar_sensor 폴더의 main.py 를 실행하여 다양한 센서와 lidar의 상태를 체크 할 수 있습니다.
- ex.beagle_test_middle_find.py 파일을 이용하면 비글 로봇이 주위를 파악하고 막힌 사각형 공간 안에서 중심이 어딘지 파악하고, 스스로 중심으로 이동 하는 기능 또한 사용할 수 있습니다.





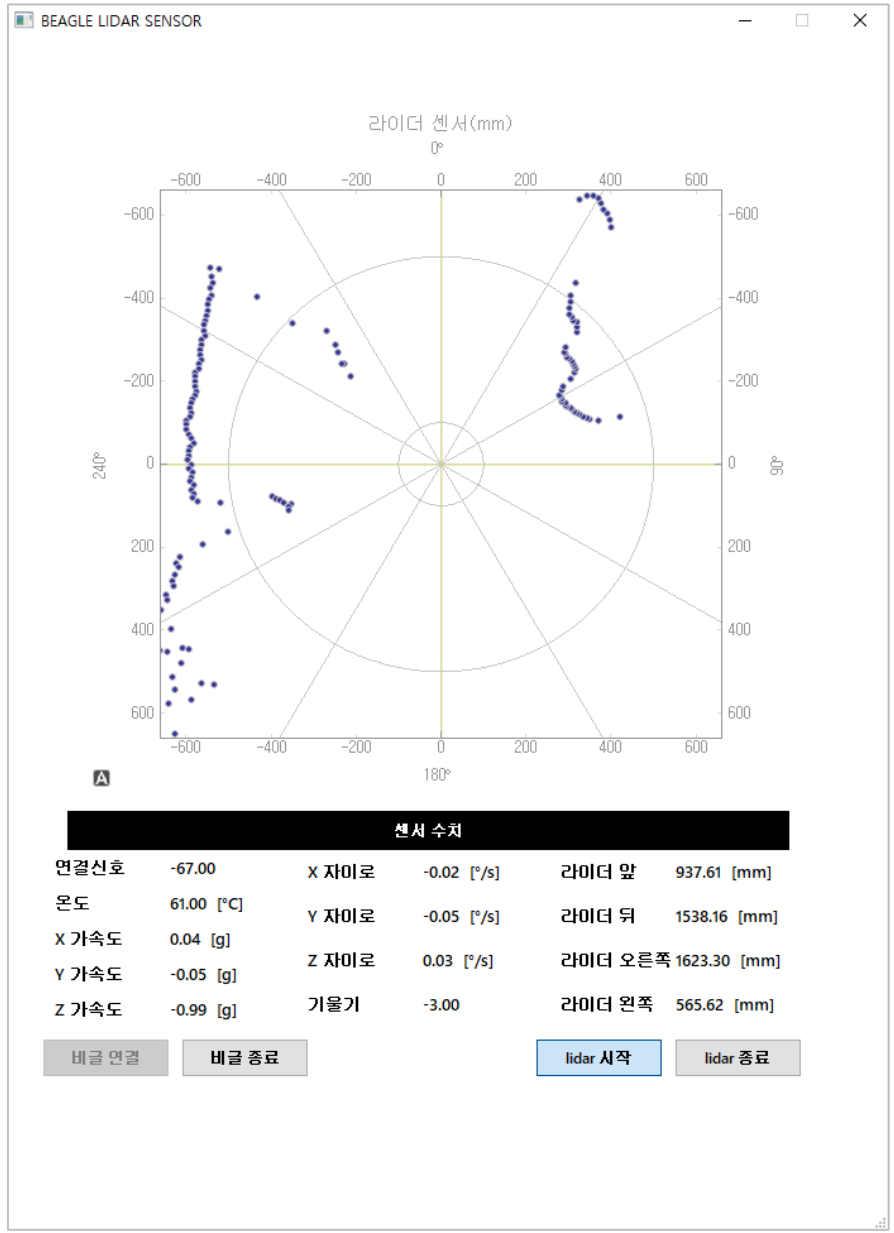
02

라이다 센서 활용하기

- beagle_test_lidar_sensor 폴더의 main.py 를 실행하면 우측 프로그램이 실행됩니다.

아래 버튼을 통해 로봇을 연결 하고 종료 할 수 있으며 lidar 시작을 누르면 라이다가 돌아가기 시작하면서 주위의 모습을 확인할 수 있습니다.

또한 이 상태에서 키보드 방향키로 이동 또한 가능하니 이것을 통해 다양한 센서 값과 라이다 값 그리고 주변 모습을 확인하여 봅시다.



사용시 주의사항으로 사용할 때 필요한 모듈의 추가적인 다운로드가 필요할 수 있습니다.
이때에는
pip install (필요한 모듈) 을 터미널에 입력하여 다운로드 하여 사용하실 수 있습니다.

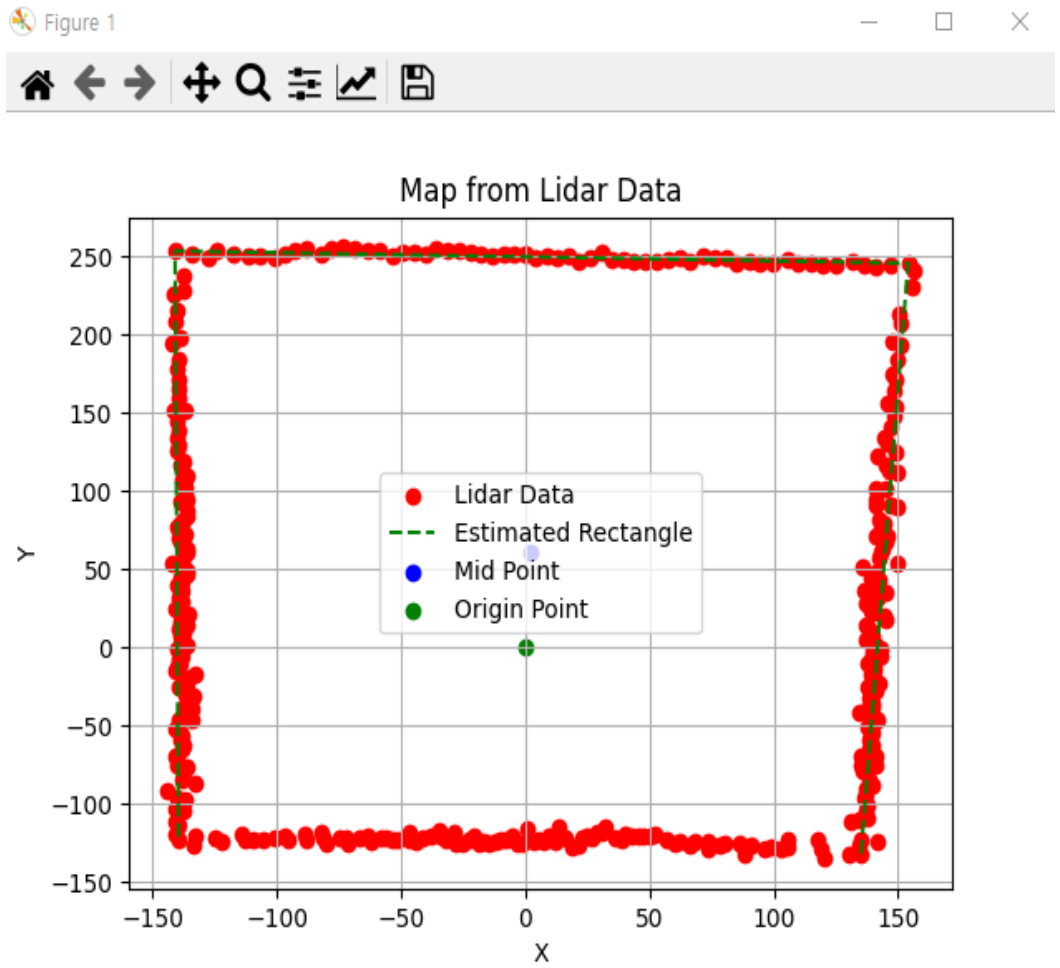
예시) `import keyboard`
위와 같이 코드 아래 밑줄이 쳐지고 실행에 오류가 날 때는 터미널에 pip install keyboard를 입력하여 모듈을 다운받습니다.

02

라이다 센서 활용하기

- ex.beagle_test_middle_find.py 를 이용하면 라이다 센서를 통해 사각형으로 막힌 공간의 중심을 찾고 이동하는 로봇을 테스트 해볼 수 있습니다.
- 우선 비글을 사각형의 사방이 막힌 공간에 나두고 ex.beagle_test_middle_find.py 를 실행시켜봅시다.
- 실행 후 잠시 기다리면 우측과 같이 주변의 공간을 라이다를 통해 파악하고,
  로봇이 어느 지점에 있는지 파악해서 MAP을 띄웁니다.
- 이후 맵을 종료하면 비글 로봇이 사각형 공간의 중심을 찾아서
  그 중심을 향해 이동하기 시작합니다.
- ex.beagle_test_middle_find.py를 통해 비글 로봇의 라이다 센서가
  어떻게 동작하는지 확인해 봅시다.

※ 코드 실행 시 필요한 모듈의 추가적인 다운로드가 필요할 수 있습니다.
이 때는 “pip install 필요한 모듈명”을 터미널에 입력하여 해당 모듈을 다운로드하시면 됩니다.



감사합니다.

ROBOMATION
MAKE THINGS ALIVE