**1. 생성 및 폐기**

|  |  |
| --- | --- |
| **Beagle()** | |
| Beagle 인스턴스를 생성하고 하드웨어 비글 로봇과 통신을 연결한다.  Beagle(0)을 호출한 것과 같다. | |
| **반환 값:** | Beagle 인스턴스의 참조 |
| from roboid import \* beagle = **Beagle**() | |

|  |  |
| --- | --- |
| **Beagle(index)** | |
| Beagle 인스턴스를 생성하고 하드웨어 비글 로봇과 통신을 연결한다.  몇 번째 비글 로봇인지를 나타내는 인덱스를 index로 설정한다. 인덱스가 같으면 같은 비글 로봇이다. | |
| **파라미터:** | index: 몇 번째 비글 로봇인지를 나타내는 인덱스(0이상의 정수) |
| **반환 값:** | Beagle 인스턴스의 참조 |
| from roboid import \* beagle1 = **Beagle**(0)  beagle2 = **Beagle**(1) | |

|  |  |
| --- | --- |
| **Beagle(port\_name)** | |
| Beagle 인스턴스를 생성하고 port\_name의 시리얼 포트를 통해 하드웨어 비글 로봇과 통신을 연결한다.  Beagle(0, port\_name)을 호출한 것과 같다. | |
| **파라미터:** | port\_name: 시리얼 포트 이름(문자열) |
| **반환 값:** | Beagle 인스턴스의 참조 |
| from roboid import \* beagle = **Beagle**('COM52') | |

|  |  |
| --- | --- |
| **Beagle(index, port\_name)** | |
| Beagle 인스턴스를 생성하고 port\_name의 시리얼 포트를 통해 하드웨어 비글 로봇과 통신을 연결한다.  몇 번째 비글 로봇인지를 나타내는 인덱스를 index로 설정한다. 인덱스가 같으면 같은 비글 로봇이다. | |
| **파라미터:** | index: 몇 번째 비글 로봇인지를 나타내는 인덱스(0이상의 정수)  port\_name: 시리얼 포트 이름(문자열) |
| **반환 값:** | Beagle 인스턴스의 참조 |
| from roboid import \* beagle1 = **Beagle**(0, 'COM52')  beagle2 = **Beagle**(1, 'COM53') | |

|  |
| --- |
| **dispose()** |
| 비글 로봇의 상태를 초기화하고 통신 연결을 끊는다. |
| from roboid import \* beagle = Beagle() # 비글 로봇을 폐기한다. beagle.**dispose**() |

|  |
| --- |
| **reset()** |
| 비글 로봇의 상태를 초기화한다. |
| from roboid import \* beagle = Beagle() # 비글 로봇을 초기화한다. beagle.**reset**() |

**2. 바퀴 움직임**

|  |
| --- |
| **move\_forward()** |
| 앞으로 1초 이동한다. (기본 속도인 50%의 속도로 이동)  move\_forward(1) 또는 move\_forward(1, 50)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속도로 앞으로 1초 이동한다. beagle.**move\_forward**() |

|  |  |
| --- | --- |
| **move\_forward(sec)** | |
| sec초 앞으로 이동한다. (기본 속도인 50%의 속도로 이동)  move\_forward(sec, 50)을 호출한 것과 같다.  sec 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | sec: 이동할 시간(실수) [초] |
| from roboid import \* beagle = Beagle() # 50%의 속도로 앞으로 2초 이동한다. beagle.**move\_forward**(2) | |

|  |  |
| --- | --- |
| **move\_forward(sec, velocity)** | |
| sec초 앞으로 이동한다. (velocity 속도로 이동)  sec 값이 음수이면 반대 방향으로 이동한다.  velocity 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * cm: 이동할 시간(실수) [초]   velocity: 이동할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 앞으로 2초 이동한다. beagle.**move\_forward**(2, 50) | |

|  |  |
| --- | --- |
| **move\_forward\_pulse(pulse)** | |
| 펄스(pulse) 수만큼 앞으로 이동한다. (기본 속도인 50%의 속도로 이동)   * move\_forward\_pulse(pulse, 50)을 호출한 것과 같다. * pulse 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * pulse: 이동할 펄스 수(0 이상의 정수, 0: 정지) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 앞으로 1000 펄스 이동한다. beagle.**move\_forward\_pulse**(1000) | |

|  |  |
| --- | --- |
| **move\_forward\_pulse(pulse, velocity)** | |
| 펄스(pulse) 수만큼 앞으로 이동한다. (velocity 속도로 이동)  pulse 값이 음수이면 반대 방향으로 이동한다.  velocity 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * pulse: 이동할 펄스 수(0 이상의 정수, 0: 정지)   velocity: 이동할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 앞으로 1000 펄스 이동한다. beagle.**move\_forward\_pulse**(1000, 50) | |

|  |
| --- |
| **move\_backward()** |
| 뒤로 1초 이동한다. (기본 속도인 50%의 속도로 이동)  move\_backward(1) 또는 move\_backward(1, 50)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속도로 뒤로 1초 이동한다. beagle.**move\_backward**() |

|  |  |
| --- | --- |
| **move\_backward(sec)** | |
| sec초 뒤로 이동한다. (기본 속도인 50%의 속도로 이동)  move\_backward(sec, 50)을 호출한 것과 같다.  sec 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * sec: 이동할 시간(실수) [초] |
| from roboid import \* beagle = Beagle() # 50%의 속도로 뒤로 2초 이동한다. beagle.**move\_backward**(2) | |

|  |  |
| --- | --- |
| **move\_backward(sec, velocity)** | |
| sec초 뒤로 이동한다. (velocity 속도로 이동)  sec 값이 음수이면 반대 방향으로 이동한다.  velocity 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * sec: 이동할 시간(실수) [초] * velocity: 이동할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 뒤로 2초 이동한다. beagle.**move\_backward**(2, 50) | |

|  |  |
| --- | --- |
| **move\_backward\_pulse(pulse)** | |
| 펄스(pulse) 수만큼 뒤로 이동한다. (기본 속도인 50%의 속도로 이동)  move\_backward\_pulse(pulse, 50)을 호출한 것과 같다.  pulse 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * pulse: 이동할 펄스 수(0 이상의 정수, 0: 정지) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 뒤로 1000 펄스 이동한다. beagle.**move\_backward\_pulse**(1000) | |

|  |  |
| --- | --- |
| **move\_backward\_pulse(pulse, velocity)** | |
| 펄스(pulse) 수만큼 뒤로 이동한다. (velocity 속도로 이동)  pulse 값이 음수이면 반대 방향으로 이동한다.  velocity 값이 음수이면 반대 방향으로 이동한다. | |
| **파라미터:** | * pulse: 이동할 펄스 수(0 이상의 정수, 0: 정지) * velocity: 이동할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 뒤로 1000 펄스 이동한다. beagle.**move\_backward\_pulse**(1000, 50) | |

|  |
| --- |
| **turn\_left()** |
| 제자리에서 왼쪽으로 1초 회전한다. (기본 속도인 50%의 속도로 회전)  turn\_left(1) 또는 turn\_left(1, 50)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 왼쪽으로 1초 회전한다. beagle.**turn\_left**() |

|  |  |
| --- | --- |
| **turn\_left(sec)** | |
| sec초 제자리에서 왼쪽으로 회전한다. (기본 속도인 50%의 속도로 회전)  turn\_left(sec, 50)을 호출한 것과 같다.  sec 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 왼쪽으로 2초 회전한다. beagle.**turn\_left**(2) | |

|  |  |
| --- | --- |
| **turn\_left(sec, velocity)** | |
| sec초 제자리에서 왼쪽으로 회전한다. (velocity 속도로 회전)  sec 값이 음수이면 반대 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] * velocity: 회전할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 왼쪽으로 2초 회전한다. beagle.**turn\_left**(2, 50) | |

|  |  |
| --- | --- |
| **turn\_left\_pulse(pulse)** | |
| 펄스(pulse) 수만큼 제자리에서 왼쪽으로 회전한다. (기본 속도인 50%의 속도로 회전)  turn\_left\_pulse(pulse, 50)을 호출한 것과 같다.  pulse 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 왼쪽으로 1000 펄스 회전한다. beagle.**turn\_left\_pulse**(1000) | |

|  |  |
| --- | --- |
| **turn\_left\_pulse(pulse, velocity)** | |
| 펄스(pulse) 수만큼 제자리에서 왼쪽으로 회전한다. (velocity 속도로 회전)  pulse 값이 음수이면 반대 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) * velocity: 회전할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 왼쪽으로 1000 펄스 회전한다. beagle.**turn\_left\_pulse**(1000, 50) | |

|  |
| --- |
| **turn\_right()** |
| 제자리에서 오른쪽으로 1초 회전한다. (기본 속도인 50%의 속도로 회전)  turn\_right(1) 또는 turn\_right(1, 50)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 오른쪽으로 1초 회전한다. beagle.**turn\_right**() |

|  |  |
| --- | --- |
| **turn\_right(sec)** | |
| sec초 제자리에서 오른쪽으로 회전한다. (기본 속도인 50%의 속도로 회전)  turn\_right(sec, 50)을 호출한 것과 같다.  sec 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 오른쪽으로 2초 회전한다. beagle.**turn\_right**(2) | |

|  |  |
| --- | --- |
| **turn\_right(sec, velocity)** | |
| sec초 제자리에서 오른쪽으로 회전한다. (velocity 속도로 회전)  sec 값이 음수이면 반대 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] * velocity: 회전할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 오른쪽으로 2초 회전한다. beagle.**turn\_right**(2, 50) | |

|  |  |
| --- | --- |
| **turn\_right\_pulse(pulse)** | |
| 펄스(pulse) 수만큼 제자리에서 오른쪽으로 회전한다. (기본 속도인 50%의 속도로 회전)  turn\_right\_pulse(pulse, 50)을 호출한 것과 같다.  pulse 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 오른쪽으로 1000 펄스 회전한다. beagle.**turn\_right\_pulse**(1000) | |

|  |  |
| --- | --- |
| **turn\_right\_pulse(pulse, velocity)** | |
| 펄스(pulse) 수만큼 제자리에서 오른쪽으로 회전한다. (velocity 속도로 회전)  pulse 값이 음수이면 반대 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) * velocity: 회전할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 제자리에서 오른쪽으로 1000 펄스 회전한다. beagle.**turn\_right\_pulse**(1000, 50) | |

|  |
| --- |
| **pivot\_left()** |
| 왼쪽 바퀴를 중심으로 1초 회전한다. (기본 속도인 50%의 속도로 회전)  pivot\_left(1) 또는 pivot\_left(1, 50)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속도로 왼쪽 바퀴를 중심으로 1초 회전한다. beagle.**pivot\_left**() |

|  |  |
| --- | --- |
| **pivot\_left(sec)** | |
| 왼쪽 바퀴를 중심으로 sec초 회전한다. (기본 속도인 50%의 속도로 회전)  pivot\_left(sec, 50)을 호출한 것과 같다.  sec 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] |
| from roboid import \* beagle = Beagle() # 50%의 속도로 왼쪽 바퀴를 중심으로 2초 회전한다. beagle.**pivot\_left**(2) | |

|  |  |
| --- | --- |
| **pivot\_left(sec, velocity)** | |
| 왼쪽 바퀴를 중심으로 sec초 회전한다. (velocity 속도로 회전)  sec 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초]   velocity: 이동할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 왼쪽 바퀴를 중심으로 2초 회전한다. beagle.**pivot\_left**(2, 50) | |

|  |  |
| --- | --- |
| **pivot\_left\_pulse(pulse)** | |
| 왼쪽 바퀴를 중심으로 펄스(pulse) 수만큼 회전한다. (기본 속도인 50%의 속도로 회전)  pivot\_left\_pulse(pulse, 50)을 호출한 것과 같다.  pulse 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 왼쪽 바퀴를 중심으로 1000 펄스 회전한다. beagle.**pivot\_left\_pulse**(1000) | |

|  |  |
| --- | --- |
| **pivot\_left\_pulse(pulse, velocity)** | |
| 왼쪽 바퀴를 중심으로 펄스(pulse) 수만큼 회전한다. (velocity 속도로 회전)  pulse 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) * velocity: 회전할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 왼쪽 바퀴를 중심으로 1000 펄스 회전한다. beagle.**pivot\_left\_pulse**(1000, 50) | |

|  |
| --- |
| **pivot\_right()** |
| 오른쪽 바퀴를 중심으로 1초 회전한다. (기본 속도인 50%의 속도로 회전)  pivot\_right(1) 또는 pivot\_right(1, 50)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속도로 오른쪽 바퀴를 중심으로 1초 회전한다. beagle.**pivot\_right**() |

|  |  |
| --- | --- |
| **pivot\_right(sec)** | |
| 오른쪽 바퀴를 중심으로 sec초 회전한다. (기본 속도인 50%의 속도로 회전)  pivot\_right(sec, 50)을 호출한 것과 같다. sec 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] |
| from roboid import \* beagle = Beagle() # 50%의 속도로 오른쪽 바퀴를 중심으로 2초 회전한다. beagle.**pivot\_right**(2) | |

|  |  |
| --- | --- |
| **pivot\_right(sec, velocity)** | |
| 오른쪽 바퀴를 중심으로 sec초 회전한다. (velocity 속도로 회전)  sec 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * sec: 회전할 시간(실수) [초] * velocity: 이동할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 오른쪽 바퀴를 중심으로 2초 회전한다. beagle.**pivot\_right**(2, 50) | |

|  |  |
| --- | --- |
| **pivot\_right\_pulse(pulse)** | |
| 오른쪽 바퀴를 중심으로 펄스(pulse) 수만큼 회전한다. (기본 속도인 50%의 속도로 회전)  pivot\_right\_pulse(pulse, 50)을 호출한 것과 같다.  pulse 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 오른쪽 바퀴를 중심으로 1000 펄스 회전한다. beagle.**pivot\_right\_pulse**(1000) | |

|  |  |
| --- | --- |
| **pivot\_right\_pulse(pulse, velocity)** | |
| 오른쪽 바퀴를 중심으로 펄스(pulse) 수만큼 회전한다. (velocity 속도로 회전)  pulse 값이 양수이면 로봇의 앞쪽 방향으로 회전하고, 음수이면 뒤쪽 방향으로 회전한다.  velocity 값이 음수이면 반대 방향으로 회전한다. | |
| **파라미터:** | * pulse: 회전할 펄스 수(0 이상의 정수, 0: 정지) * velocity: 회전할 속도(실수, -100 ~ 100 [%]) |
| from roboid import \* beagle = Beagle() # 50%의 속도로 오른쪽 바퀴를 중심으로 1000 펄스 회전한다. beagle.**pivot\_right\_pulse**(1000, 50) | |

|  |  |
| --- | --- |
| **wheels(left\_velocity, right\_velocity)** | |
| 왼쪽 바퀴와 오른쪽 바퀴의 속도를 설정한다.  양수 값은 전진 방향으로의 회전을, 음수 값은 후진 방향으로의 회전을 의미한다. 부호를 제외한 절대치가 클수록 속도가 빨라진다. | |
| **파라미터:** | * left\_velocity: 왼쪽 바퀴의 속도(실수, -100 ~ 100 [%], 0: 정지) * right\_velocity: 오른쪽 바퀴의 속도(실수, -100 ~ 100 [%], 0: 정지) |
| from roboid import \* beagle = Beagle() # 양쪽 바퀴를 50%의 속력으로 앞으로 회전하게 한다. beagle.**wheels**(50, 50) # 양쪽 바퀴를 50%의 속력으로 뒤로 회전하게 한다. beagle.**wheels**(-50, -50) # 양쪽 바퀴를 정지한다. beagle.**wheels**(0, 0) | |

|  |  |
| --- | --- |
| **wheels(velocity)** | |
| 양쪽 바퀴의 속도를 설정한다.  wheels(velocity, velocity)를 호출한 것과 같다.  양수 값은 전진 방향으로의 회전을, 음수 값은 후진 방향으로의 회전을 의미한다. 부호를 제외한 절대치가 클수록 속도가 빨라진다. | |
| **파라미터:** | * velocity: 양쪽 바퀴의 속도(실수, -100 ~ 100 [%], 0: 정지) |
| from roboid import \* beagle = Beagle() # 양쪽 바퀴를 50%의 속력으로 앞으로 회전하게 한다. beagle.**wheels**(50) # 양쪽 바퀴를 50%의 속력으로 뒤로 회전하게 한다. beagle.**wheels**(-50) # 양쪽 바퀴를 정지한다. beagle.**wheels**(0) | |

|  |  |
| --- | --- |
| **left\_wheel(velocity)** | |
| 왼쪽 바퀴의 속도를 설정한다.  양수 값은 전진 방향으로의 회전을, 음수 값은 후진 방향으로의 회전을 의미한다. 부호를 제외한 절대치가 클수록 속도가 빨라진다. | |
| **파라미터:** | * velocity: 왼쪽 바퀴의 속도(실수, -100 ~ 100 [%], 0: 정지) |
| from roboid import \* beagle = Beagle() # 왼쪽 바퀴를 50%의 속력으로 앞으로 회전하게 한다. beagle.**left\_wheel**(50) # 왼쪽 바퀴를 50%의 속력으로 뒤로 회전하게 한다. beagle.**left\_wheel**(-50) # 왼쪽 바퀴를 정지한다. beagle.**left\_wheel**(0) | |

|  |  |
| --- | --- |
| **right\_wheel(velocity)** | |
| 오른쪽 바퀴의 속도를 설정한다.  양수 값은 전진 방향으로의 회전을, 음수 값은 후진 방향으로의 회전을 의미한다. 부호를 제외한 절대치가 클수록 속도가 빨라진다. | |
| **파라미터:** | * velocity: 오른쪽 바퀴의 속도(실수, -400 ~ 400 [%], 0: 정지) |
| from roboid import \* beagle = Beagle() # 오른쪽 바퀴를 50%의 속력으로 앞으로 회전하게 한다. beagle.**right\_wheel**(50) # 오른쪽 바퀴를 50%의 속력으로 뒤로 회전하게 한다. beagle.**right\_wheel**(-50) # 오른쪽 바퀴를 정지한다. beagle.**right\_wheel**(0) | |

|  |
| --- |
| **stop()** |
| 양쪽 바퀴를 정지한다. wheels(0, 0) 또는 wheels(0)을 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 50%의 속력으로 앞으로 이동한다. beagle.wheels(50, 50) # 양쪽 바퀴를 정지한다. beagle.**stop**() |

**3. 소리**

|  |  |
| --- | --- |
| **buzzer(hz)** | |
| 버저 소리의 음 높이 주파수를 hz [Hz]로 설정한다.  음 높이는 소수점 첫째 자리까지 입력할 수 있으며, 버저 소리를 끄기 위해서는 0을 입력하면 된다. | |
| **파라미터:** | * hz: 버저 소리의 음 높이(실수, 0 ~ 167772.15 [Hz], 0: off) |
| 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) | |

|  |  |
| --- | --- |
| **tempo(bpm)** | |
| 연주하거나 쉬는 속도를 bpm(분당 박자 수)으로 설정한다. | |
| **파라미터:** | * bpm: 분당 박자 수(실수 [BPM], 초기 값: 60) |
| from roboid import \* beagle = Beagle() # 연주 속도를 60 BPM으로 한다. beagle.**tempo**(60) | |

|  |  |
| --- | --- |
| **note(pitch)** | |
| 버저를 이용한 오차 0.01% 이하의 정확한 음정을 소리 낸다.  음의 높이(pitch)에서 뒤의 숫자는 옥타브를 의미하며 1부터 7까지의 값을 가질 수 있다. 예를 들어 “C4”는 4번째 옥타브의 도 음이며, “D#5”는 5번째 옥타브의 레# 음이다. 소리를 끄기 위해서는 “OFF”를 입력하면 된다.   |  |  | | --- | --- | | **음의 높이** | **설명** | | “OFF” | 소리를 끈다. | | “C4” | 4번째 옥타브의 도 음 | | “C#4” 또는 “Db4” | 4번째 옥타브의 도#(레b) 음 | | “D4” | 4번째 옥타브의 레 음 | | “D#4” 또는 “Eb4” | 4번째 옥타브의 레#(미b) 음 | | “E4” | 4번째 옥타브의 미 음 | | “F4” | 4번째 옥타브의 파 음 | | “F#4” 또는 “Gb4” | 4번째 옥타브의 파#(솔b) 음 | | “G4” | 4번째 옥타브의 솔 음 | | “G#4” 또는 “Ab4” | 4번째 옥타브의 솔#(라b) 음 | | “A4” | 4번째 옥타브의 라 음 | | “A#4” 또는 “Bb4” | 4번째 옥타브의 라#(시b) 음 | | “B4” | 4번째 옥타브의 시 음 | | |
| **파라미터:** | * pitch: 음의 높이(대소문자 구분하지 않음) |
| from roboid import \* beagle = Beagle() # 5옥타브 도# 음을 소리 낸다. beagle.**note**(“C#5”) # 소리를 끈다. beagle.**note**(“OFF”) | |

|  |  |
| --- | --- |
| **note(pitch, beats)** | |
| 버저를 이용한 오차 0.01% 이하의 정확한 음정을 beats 박자 만큼 소리 낸다.  음의 높이(pitch)에서 뒤의 숫자는 옥타브를 의미하며 1부터 7까지의 값을 가질 수 있다. 예를 들어 “C4”는 4번째 옥타브의 도 음이며, “D#5”는 5번째 옥타브의 레# 음이다. 소리를 끄기 위해서는 “OFF”를 입력하면 된다.   |  |  | | --- | --- | | **음의 높이** | **설명** | | “OFF” | 소리를 끈다. | | “C4” | 4번째 옥타브의 도 음 | | “C#4” 또는 “Db4” | 4번째 옥타브의 도#(레b) 음 | | “D4” | 4번째 옥타브의 레 음 | | “D#4” 또는 “Eb4” | 4번째 옥타브의 레#(미b) 음 | | “E4” | 4번째 옥타브의 미 음 | | “F4” | 4번째 옥타브의 파 음 | | “F#4” 또는 “Gb4” | 4번째 옥타브의 파#(솔b) 음 | | “G4” | 4번째 옥타브의 솔 음 | | “G#4” 또는 “Ab4” | 4번째 옥타브의 솔#(라b) 음 | | “A4” | 4번째 옥타브의 라 음 | | “A#4” 또는 “Bb4” | 4번째 옥타브의 라#(시b) 음 | | “B4” | 4번째 옥타브의 시 음 | | |
| **파라미터:** | * pitch: 음의 높이(대소문자 구분하지 않음) * beats: 박자(실수) |
| from roboid import \* beagle = Beagle() # 5옥타브 도# 음을 0.5 박자 소리낸다. beagle.**note**(“C#5”, 0.5) # 0.5 박자 쉰다. beagle.**note**(“OFF”, 0.5) | |

|  |  |
| --- | --- |
| **sound(sound\_id)** | |
| sound\_id에 해당하는 소리를 재생한다.  소리를 끄기 위해서는 sound\_id에 “OFF”를 입력하면 된다. sound\_until\_done 메소드와는 달리 소리를 재생하라고 명령하기만 할 뿐, 소리의 재생이 끝날 때까지 기다리지는 않는다.   |  |  | | --- | --- | | **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. | | |
| **파라미터:** | sound\_id: 재생할 소리(대소문자 구분하지 않음) |
| from roboid import \* beagle = Beagle() # 엔진 소리를 내면서 앞으로 이동한다. beagle.**sound**(“ENGINE”) beagle.move\_forward() | |

|  |  |
| --- | --- |
| **sound(sound\_id, repeat)** | |
| sound\_id에 해당하는 소리를 repeat번 재생한다.  소리를 끄기 위해서는 sound\_id에 “OFF”를 입력하면 된다.  repeat가 음수 값이면 영원히 반복하여 재생한다. | |
| **파라미터:** | * sound\_id: 재생할 소리(대소문자 구분하지 않음)   repeat: 재생할 횟수(정수) |
| from roboid import \* beagle = Beagle() # 엔진 소리를 2번 내면서 앞으로 이동한다. beagle.**sound**(“ENGINE”, 2) beagle.move\_forward() # 로봇 소리를 영원히 반복하여 재생한다. beagle.**sound**(“ROBOT”, -1)  wait(-1) | |

|  |  |
| --- | --- |
| **sound\_until\_done(sound\_id)** | |
| sound\_id에 해당하는 소리를 재생하고 재생이 끝날 때까지 기다린다.  sound\_until\_done(sound\_id, 1)을 호출한 것과 같다.  소리를 끄기 위해서는 sound\_id에 “OFF”를 입력하면 된다. sound 메소드와는 달리 소리를 재생하고 소리의 재생이 끝날 때까지 기다린다. | |
| **파라미터:** | sound\_id: 재생할 소리(대소문자 구분하지 않음) |
| from roboid import \* beagle = Beagle() # 엔진 소리를 재생한 후에 앞으로 이동한다. beagle.**sound\_until\_done**(“ENGINE”) beagle.move\_forward() | |

|  |  |
| --- | --- |
| **sound\_until\_done(sound\_id, repeat)** | |
| sound\_id에 해당하는 소리를 repeat번 재생하고 재생이 끝날 때까지 기다린다.  소리를 끄기 위해서는 sound\_id에 “OFF”를 입력하면 된다.  sound 메소드와는 달리 소리를 재생하고 소리의 재생이 끝날 때까지 기다린다.  repeat가 음수 값이면 영원히 반복하여 재생한다. | |
| **파라미터:** | * sound\_id: 재생할 소리(대소문자 구분하지 않음)   repeat: 재생할 횟수(정수) |
| from roboid import \* beagle = Beagle() # 엔진 소리를 2번 재생한 후에 앞으로 이동한다. beagle.**sound\_until\_done**(“ENGINE”, 2) beagle.move\_forward() # 로봇 소리를 영원히 반복하여 재생한다. beagle.**sound\_until\_done**(“ROBOT”, -1) | |

|  |
| --- |
| **beep()** |
| 삐 소리를 재생하고 소리의 재생이 끝날 때까지 기다린다.  sound\_until\_done(‘beep’)를 호출한 것과 같다. |
| from roboid import \* beagle = Beagle() # 삐 소리를 재생한다. beagle.**beep**() |

**4. 센서**

|  |  |
| --- | --- |
| **signal\_strength()** | |
| 신호 세기 값을 반환한다.  비글 로봇과 컴퓨터 간의 블루투스 무선 통신의 신호 세기를 나타낸다. 신호의 세기가 셀수록 값이 커진다. | |
| **반환 값:** | 신호 세기 값(정수, -128 ~ 0 [dBm], 초기 값: 0) |
| from roboid import \* beagle = Beagle() # 신호 세기 값을 얻는다. value = beagle.**signal\_strength**() | |

|  |  |
| --- | --- |
| **temperature()** | |
| 온도 센서 값을 반환한다. 로봇 내부의 온도 값이다. | |
| **반환 값:** | 온도 센서 값(정수, -41 ~ 87 [oC], 초기 값: 0) |
| from roboid import \* beagle = Beagle() # 온도 센서 값을 얻는다. value = beagle.**temperature**() | |

|  |  |
| --- | --- |
| **left\_encoder()** | |
| 왼쪽 바퀴의 엔코더 값을 반환한다. 바퀴가 앞쪽 방향으로 회전하면 값이 증가하고 뒤쪽 방향으로 회전하면 값이 감소한다. | |
| **반환 값:** | 왼쪽 바퀴의 엔코더 펄스 수(정수, -2147483648 ~ 2147483647, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # 왼쪽 바퀴의 엔코더 값을 얻는다. value = beagle.**left\_encoder**() | |

|  |  |
| --- | --- |
| **right\_encoder()** | |
| 오른쪽 바퀴의 엔코더 값을 반환한다. 바퀴가 앞쪽 방향으로 회전하면 값이 증가하고 뒤쪽 방향으로 회전하면 값이 감소한다. | |
| **반환 값:** | 오른쪽 바퀴의 엔코더 펄스 수(정수, -2147483648 ~ 2147483647, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # 오른쪽 바퀴의 엔코더 값을 얻는다. value = beagle.**right\_encoder**() | |

|  |
| --- |
| **reset\_encoder()** |
| 양쪽 바퀴의 엔코더 값을 초기화한다. |
| from roboid import \* beagle = Beagle() # 엔코더 값을 초기화한다. beagle.**reset\_encoder**() |

|  |  |
| --- | --- |
| **accelerometer\_x()** | |
| X축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | X축 가속도 센서 값(실수, -16 ~ 16, 초기 값: 0) [g] |
| from roboid import \* beagle = Beagle() # X축 가속도 값을 얻는다. value = beagle.**accelerometer\_x**() | |

|  |  |
| --- | --- |
| **accelerometer\_y()** | |
| Y축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Y축 가속도 센서 값(실수, -16 ~ 16, 초기 값: 0) [g] |
| from roboid import \* beagle = Beagle() # Y축 가속도 값을 얻는다. value = beagle.**accelerometer\_y**() | |

|  |  |
| --- | --- |
| **accelerometer\_z()** | |
| Z축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Z축 가속도 센서 값(실수, -16 ~ 16, 초기 값: 0) [g] |
| from roboid import \* beagle = Beagle() # Z축 가속도 값을 얻는다. value = beagle.**accelerometer\_z**() | |

|  |  |
| --- | --- |
| **accelerometer()** | |
| X, Y, Z축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | (인덱스, X축 가속도 센서 값, Y축 가속도 센서 값 , Z축 가속도 센서 값)으로 구성된 튜플 (센서 값: 실수, -16 ~ 16, 초기 값: 0 [g]) |
| from roboid import \* beagle = Beagle() # 가속도 값을 얻는다. index, x, y, z = beagle.**accelerometer**() | |

|  |  |
| --- | --- |
| **raw\_accelerometer\_x()** | |
| 스케일 값을 곱하지 않은 X축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | X축 가속도 센서 값(정수, -2048 ~ 2047, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # X축 가속도 값을 얻는다. value = beagle.**raw\_accelerometer\_x**() | |

|  |  |
| --- | --- |
| **raw\_accelerometer\_y()** | |
| 스케일 값을 곱하지 않은 Y축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Y축 가속도 센서 값(정수, -2048 ~ 2047, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # Y축 가속도 값을 얻는다. value = beagle.**raw\_accelerometer\_y**() | |

|  |  |
| --- | --- |
| **raw\_accelerometer\_z()** | |
| 스케일 값을 곱하지 않은 Z축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Z축 가속도 센서 값(정수, -2048 ~ 2047, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # Z축 가속도 값을 얻는다. value = beagle.**raw\_accelerometer\_z**() | |

|  |  |
| --- | --- |
| **raw\_accelerometer()** | |
| 스케일 값을 곱하지 않은 X, Y, Z축 가속도 센서 값을 반환한다.  비글 로봇의 가속도 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | (인덱스, X축 가속도 센서 값, Y축 가속도 센서 값 , Z축 가속도 센서 값)으로 구성된 튜플 (센서 값: 정수, -2048 ~ 2047, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # 가속도 값을 얻는다. index, x, y, z = beagle.**raw\_accelerometer**() | |

|  |  |
| --- | --- |
| **scale\_accelerometer()** | |
| 가속도 센서의 스케일 값을 반환한다. | |
| **반환 값:** | 가속도 센서의 스케일 값 (실수, 0.0009765625 ~ 0.0078125, 초기 값: 0.0009765625) |
| from roboid import \* beagle = Beagle() # 스케일 값을 얻는다. scale = beagle.**scale\_accelerometer**()  print(beagle.accelerometer\_x() == beagle.raw\_accelerometer\_x() \* scale) | |

|  |  |
| --- | --- |
| **listen\_accelerometer(fn, interpolation=None)** | |
| 가속도 센서 값을 얻기 위한 리스너 함수 fn을 설정한다.  리스너 함수는 fn(index, timestamp, x, y, z)의 형태를 가진다. 함수 이름이 fn일 필요는 없다.  새로운 가속도 센서 값을 얻으면 fn 함수가 호출된다. Index는 센서 값의 인덱스, timestamp는 센서 값을 측정한 시간(보간한 경우에는 보간한 때의 시간), x는 X축 가속도 센서 값, y는 Y축 가속도 센서 값, z는 Z축 가속도 센서 값이다. 함수 내에서 시간을 끌면 안 된다. | |
| **파라미터:** | fn: 새로운 가속도 센서 값을 얻었을 때 호출할 함수  interpolation:   * None: 보간법 적용하지 않음 * ‘constant’: 이전 값으로 채우기 * ‘linear’: 선으로 이어서 샘플링 |
| from roboid import \* beagle = Beagle() # 리스너 함수를 정의한다.  def listener(index, timestamp, x, y, z):  print(index, timestamp, x, y, z) beagle.**listen\_accelerometer**(listener) | |

|  |  |
| --- | --- |
| **listen\_raw\_accelerometer(fn, interpolation=None)** | |
| 스케일 값을 곱하지 않은 가속도 센서 값을 얻기 위한 리스너 함수 fn을 설정한다.  리스너 함수는 fn(index, timestamp, x, y, z)의 형태를 가진다. 함수 이름이 fn일 필요는 없다.  새로운 가속도 센서 값을 얻으면 fn 함수가 호출된다. Index는 센서 값의 인덱스, timestamp는 센서 값을 측정한 시간(보간한 경우에는 보간한 때의 시간), x는 X축 가속도 센서 값, y는 Y축 가속도 센서 값, z는 Z축 가속도 센서 값이다. 함수 내에서 시간을 끌면 안 된다. | |
| **파라미터:** | fn: 새로운 가속도 센서 값을 얻었을 때 호출할 함수  interpolation:   * None: 보간법 적용하지 않음 * ‘constant’: 이전 값으로 채우기 * ‘linear’: 선으로 이어서 샘플링 |
| from roboid import \* beagle = Beagle() # 리스너 함수를 정의한다.  def listener(index, timestamp, x, y, z):  print(index, timestamp, x, y, z) beagle.**listen\_raw\_accelerometer**(listener) | |

|  |  |
| --- | --- |
| **gyroscope\_x()** | |
| X축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | X축 자이로 센서 값(실수, -2000 ~ 2000, 초기 값: 0) [°/s] |
| from roboid import \* beagle = Beagle() # X축 자이로 값을 얻는다. value = beagle.**gyroscope\_x**() | |

|  |  |
| --- | --- |
| **gyroscope\_y()** | |
| Y축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Y축 가속도 센서 값(실수, -2000 ~ 2000, 초기 값: 0) [°/s] |
| from roboid import \* beagle = Beagle() # Y축 자이로 값을 얻는다. value = beagle.**gyroscope\_y**() | |

|  |  |
| --- | --- |
| **gyroscope\_z()** | |
| Z축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Z축 자이로 센서 값(실수, -2000 ~ 2000, 초기 값: 0) [°/s] |
| from roboid import \* beagle = Beagle() # Z축 가속도 값을 얻는다. value = beagle.**gyroscope\_z**() | |

|  |  |
| --- | --- |
| **gyroscope()** | |
| X, Y, Z축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | (인덱스, X축 자이로 센서 값, Y축 자이로 센서 값 , Z축 자이로 센서 값)으로 구성된 튜플 (센서 값: 실수, -2000 ~ 2000, 초기 값: 0 [°/s]) |
| from roboid import \* beagle = Beagle() # 자이로 값을 얻는다. index, x, y, z = beagle.**gyroscope**() | |

|  |  |
| --- | --- |
| **raw\_gyroscope\_x()** | |
| 스케일 값을 곱하지 않은 X축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | X축 자이로 센서 값(정수, -32768 ~ 32767, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # X축 자이로 값을 얻는다. value = beagle.**raw\_gyroscope\_x**() | |

|  |  |
| --- | --- |
| **raw\_gyroscope\_y()** | |
| 스케일 값을 곱하지 않은 Y축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Y축 가속도 센서 값(정수, -32768 ~ 32767, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # Y축 자이로 값을 얻는다. value = beagle.**raw\_gyroscope\_y**() | |

|  |  |
| --- | --- |
| **raw\_gyroscope\_z()** | |
| 스케일 값을 곱하지 않은 Z축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | Z축 자이로 센서 값(정수, -32768 ~ 32767, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # Z축 가속도 값을 얻는다. value = beagle.**raw\_gyroscope\_z**() | |

|  |  |
| --- | --- |
| **raw\_gyroscope()** | |
| 스케일 값을 곱하지 않은 X, Y, Z축에 대한 자이로 센서 값을 반환한다.  비글 로봇의 자이로 센서 좌표계는 로봇이 전진하는 방향이 X축, 로봇의 왼쪽 방향이 Y축, 위쪽 방향이 Z축의 양수 방향이다. | |
| **반환 값:** | (인덱스, X축 자이로 센서 값, Y축 자이로 센서 값 , Z축 자이로 센서 값)으로 구성된 튜플 (센서 값: 정수, -32768 ~ 32767, 초기 값: 0) |
| from roboid import \* beagle = Beagle() # 자이로 값을 얻는다. index, x, y, z = beagle.**raw\_gyroscope**() | |

|  |  |
| --- | --- |
| **scale\_gyroscope()** | |
| 자이로 센서의 스케일 값을 반환한다. | |
| **반환 값:** | 자이로 센서의 스케일 값 (실수, 0.003814697265625 ~ 0.06103515625, 초기 값: 0.00762939453125) |
| from roboid import \* beagle = Beagle() # 스케일 값을 얻는다. scale = beagle.**scale\_gyroscope**()  print(beagle.gyroscope\_x() == beagle.raw\_gyroscope\_x() \* scale) | |

|  |  |
| --- | --- |
| **listen\_gyroscope(fn, interpolation=None)** | |
| 자이로 센서 값을 얻기 위한 리스너 함수 fn을 설정한다.  리스너 함수는 fn(index, timestamp, x, y, z)의 형태를 가진다. 함수 이름이 fn일 필요는 없다.  새로운 자이로 센서 값을 얻으면 fn 함수가 호출된다. Index는 센서 값의 인덱스, timestamp는 센서 값을 측정한 시간(보간한 경우에는 보간한 때의 시간), x는 X축 자이로 센서 값, y는 Y축 자이로 센서 값, z는 Z축 자이로 센서 값이다. 함수 내에서 시간을 끌면 안 된다. | |
| **파라미터:** | fn: 새로운 자이로 센서 값을 얻었을 때 호출할 함수  interpolation:   * None: 보간법 적용하지 않음 * ‘constant’: 이전 값으로 채우기 * ‘linear’: 선으로 이어서 샘플링 |
| from roboid import \* beagle = Beagle() # 리스너 함수를 정의한다.  def listener(index, timestamp, x, y, z):  print(index, timestamp, x, y, z) beagle.**listen\_gyroscope**(listener) | |

|  |  |
| --- | --- |
| **listen\_raw\_gyroscope(fn, interpolation=None)** | |
| 스케일 값을 곱하지 않은 자이로 센서 값을 얻기 위한 리스너 함수 fn을 설정한다.  리스너 함수는 fn(index, timestamp, x, y, z)의 형태를 가진다. 함수 이름이 fn일 필요는 없다.  새로운 자이로 센서 값을 얻으면 fn 함수가 호출된다. Index는 센서 값의 인덱스, timestamp는 센서 값을 측정한 시간(보간한 경우에는 보간한 때의 시간), x는 X축 자이로 센서 값, y는 Y축 자이로 센서 값, z는 Z축 자이로 센서 값이다. 함수 내에서 시간을 끌면 안 된다. | |
| **파라미터:** | fn: 새로운 자이로 센서 값을 얻었을 때 호출할 함수  interpolation:   * None: 보간법 적용하지 않음 * ‘constant’: 이전 값으로 채우기 * ‘linear’: 선으로 이어서 샘플링 |
| from roboid import \* beagle = Beagle() # 리스너 함수를 정의한다.  def listener(index, timestamp, x, y, z):  print(index, timestamp, x, y, z) beagle.**listen\_raw\_gyroscope**(listener) | |

|  |  |
| --- | --- |
| **tilt()** | |
| 비글 로봇의 기울임 상태 값을 반환한다. | |
| **반환 값:** | 기울임 상태 값(정수, 앞으로 기울임: 1, 뒤로 기울임: -1, 왼쪽으로 기울임: 2, 오른쪽으로 기울임: -2, 거꾸로 뒤집음: 3, 똑바로 놓음: -3) |
| from roboid import \* beagle = Beagle() # 기울임 상태 값을 얻는다. value = beagle.**tilt**() | |

|  |  |
| --- | --- |
| **battery\_state()** | |
| 배터리 상태 값을 반환한다. | |
| **반환 값:** | 배터리 상태 값(정수, 정상: 3, 중간: 2, 부족: 1, 없음: 0) |
| from roboid import \* beagle = Beagle() # 배터리 상태 값을 얻는다. value = beagle.**battery\_state**() | |

|  |  |
| --- | --- |
| **charge\_state()** | |
| 충전 상태 값을 반환한다. | |
| **반환 값:** | 충전 상태 값(정수, 충전 중: 1, 충전 중 아님: 0) |
| from roboid import \* beagle = Beagle() # 충전 상태 값을 얻는다. value = beagle.**charge\_state**() | |

|  |  |
| --- | --- |
| **timestamp\_basic()** | |
| IMU 센서를 제외한 기본 센서 값들을 측정한 시간을 반환한다. | |
| **반환 값:** | 센서 값들을 측정한 시간(정수) |
| from roboid import \* beagle = Beagle() # 기본 센서 값들을 측정한 시간을 얻는다. value = beagle.**timestamp\_basic**() | |

|  |  |
| --- | --- |
| **timestamp\_imu()** | |
| IMU 센서(가속도, 자이로 등) 값들을 측정한 시간을 반환한다. | |
| **반환 값:** | 센서 값들을 측정한 시간(정수) |
| from roboid import \* beagle = Beagle() # IMU 센서 값들을 측정한 시간을 얻는다. value = beagle.**timestamp\_imu**() | |

**5. 서보 모터**

|  |  |
| --- | --- |
| **servo\_output\_a(degree)** | |
| 서보 모터 A를 degree 각도로 회전시킨다. servo\_outupt\_a\_until\_done 메소드와는 다르게 회전이 완료될 때까지 기다리지 않는다. | |
| **파라미터:** | * degree: 회전할 각도(정수, 0 ~ 180 [도]) |
| from roboid import \* beagle = Beagle() # 서보 모터 A를 30도로 회전시킨다. beagle.**servo\_output\_a**(30) | |

|  |  |
| --- | --- |
| **servo\_output\_b(degree)** | |
| 서보 모터 B를 degree 각도로 회전시킨다. servo\_outupt\_b\_until\_done 메소드와는 다르게 회전이 완료될 때까지 기다리지 않는다. | |
| **파라미터:** | * degree: 회전할 각도(정수, 0 ~ 180 [도]) |
| from roboid import \* beagle = Beagle() # 서보 모터 B를 30도로 회전시킨다. beagle.**servo\_output\_b**(30) | |

|  |  |
| --- | --- |
| **servo\_output\_c(degree)** | |
| 서보 모터 C를 degree 각도로 회전시킨다. servo\_outupt\_c\_until\_done 메소드와는 다르게 회전이 완료될 때까지 기다리지 않는다. | |
| **파라미터:** | * degree: 회전할 각도(정수, 0 ~ 180 [도]) |
| from roboid import \* beagle = Beagle() # 서보 모터 C를 30도로 회전시킨다. beagle.**servo\_output\_c**(30) | |

|  |  |
| --- | --- |
| **servo\_output\_a\_until\_done(degree)** | |
| 서보 모터 A를 degree 각도로 회전시키고 회전이 완료될 때까지 기다린다. | |
| **파라미터:** | * degree: 회전할 각도(정수, 0 ~ 180 [도]) |
| from roboid import \* beagle = Beagle() # 서보 모터 A를 30도로 회전시키고 회전이 완료될 때까지 기다린다. beagle.**servo\_output\_a\_until\_done**(30) | |

|  |  |
| --- | --- |
| **servo\_output\_b\_until\_done(degree)** | |
| 서보 모터 B를 degree 각도로 회전시키고 회전이 완료될 때까지 기다린다. | |
| **파라미터:** | * degree: 회전할 각도(정수, 0 ~ 180 [도]) |
| from roboid import \* beagle = Beagle() # 서보 모터 B를 30도로 회전시키고 회전이 완료될 때까지 기다린다. beagle.**servo\_output\_b\_until\_done**(30) | |

|  |  |
| --- | --- |
| **servo\_output\_c\_until\_done(degree)** | |
| 서보 모터 C를 degree 각도로 회전시키고 회전이 완료될 때까지 기다린다. | |
| **파라미터:** | * degree: 회전할 각도(정수, 0 ~ 180 [도]) |
| from roboid import \* beagle = Beagle() # 서보 모터 C를 30도로 회전시키고 회전이 완료될 때까지 기다린다. beagle.**servo\_output\_c\_until\_done**(30) | |

|  |  |
| --- | --- |
| **servo\_speed\_a(speed)** | |
| 서보 모터 A의 속도를 speed로 설정한다. speed 값을 크게 할수록 속도가 빨라진다. 서보 모터의 속도 초기 값은 5이다. | |
| **파라미터:** | * speed: 회전하는 속도(정수, 1 ~ 7) |
| from roboid import \* beagle = Beagle() # 서보 모터 A의 속도를 3으로 설정한다. beagle.**servo\_speed\_a**(3) | |

|  |  |
| --- | --- |
| **servo\_speed\_b(speed)** | |
| 서보 모터 B의 속도를 speed로 설정한다. speed 값을 크게 할수록 속도가 빨라진다. 서보 모터의 속도 초기 값은 5이다. | |
| **파라미터:** | * speed: 회전하는 속도(정수, 1 ~ 7) |
| from roboid import \* beagle = Beagle() # 서보 모터 B의 속도를 3으로 설정한다. beagle.**servo\_speed\_b**(3) | |

|  |  |
| --- | --- |
| **servo\_speed\_c(speed)** | |
| 서보 모터 C의 속도를 speed로 설정한다. speed 값을 크게 할수록 속도가 빨라진다. 서보 모터의 속도 초기 값은 5이다. | |
| **파라미터:** | * speed: 회전하는 속도(정수, 1 ~ 7) |
| from roboid import \* beagle = Beagle() # 서보 모터 C의 속도를 3으로 설정한다. beagle.**servo\_speed\_c**(3) | |

|  |
| --- |
| **release\_servo\_a()** |
| 서보 모터 A의 전원을 끈다. |
| from roboid import \* beagle = Beagle() # 서보 모터 A의 전원을 끈다. beagle.**release\_servo\_a**() |

|  |
| --- |
| **release\_servo\_b()** |
| 서보 모터 B의 전원을 끈다. |
| from roboid import \* beagle = Beagle() # 서보 모터 B의 전원을 끈다. beagle.**release\_servo\_b**() |

|  |
| --- |
| **release\_servo\_c()** |
| 서보 모터 C의 전원을 끈다. |
| from roboid import \* beagle = Beagle() # 서보 모터 C의 전원을 끈다. beagle.**release\_servo\_c**() |

|  |  |
| --- | --- |
| **servo\_input\_a()** | |
| 서보 모터 A의 현재 각도 값을 반환한다. | |
| **반환 값:** | 서보 모터 A의 현재 각도 값(정수, 0 ~ 180, 초기 값: 0) [도] |
| from roboid import \* beagle = Beagle() # 서보 모터 A의 현재 각도 값을 얻는다. value = beagle.**servo\_input\_a**() | |

|  |  |
| --- | --- |
| **servo\_input\_b()** | |
| 서보 모터 B의 현재 각도 값을 반환한다. | |
| **반환 값:** | 서보 모터 B의 현재 각도 값(정수, 0 ~ 180, 초기 값: 0) [도] |
| from roboid import \* beagle = Beagle() # 서보 모터 B의 현재 각도 값을 얻는다. value = beagle.**servo\_input\_b**() | |

|  |  |
| --- | --- |
| **servo\_input\_c()** | |
| 서보 모터 C의 현재 각도 값을 반환한다. | |
| **반환 값:** | 서보 모터 C의 현재 각도 값(정수, 0 ~ 180, 초기 값: 0) [도] |
| from roboid import \* beagle = Beagle() # 서보 모터 C의 현재 각도 값을 얻는다. value = beagle.**servo\_input\_c**() | |

**6. 라이다**

|  |
| --- |
| **start\_lidar()** |
| 라이다를 시작한다. 시작한 후 실제 값이 들어오기까지 시간이 좀 걸린다. |
| from roboid import \* beagle = Beagle() # 라이다를 시작한다. beagle.**start\_lidar**() |

|  |
| --- |
| **stop\_lidar()** |
| 라이다를 종료한다. |
| from roboid import \* beagle = Beagle() # 라이다를 종료한다. beagle.**stop\_lidar**() |

|  |  |
| --- | --- |
| **is\_lidar\_ready()** | |
| 라이다가 준비되었는지(센서 값을 얻는 상태가 되었는지) 여부를 반환한다. | |
| **반환 값:** | 준비되었으면 True, 아니면 False |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다가 준비되었는지 확인한다. value = beagle.**is\_lidar\_ready**() | |

|  |
| --- |
| **wait\_until\_lidar\_ready()** |
| 라이다가 준비될 때까지 기다린다. |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다가 준비될 때까지 기다린다. value = beagle.**wait\_until\_lidar\_ready**() |

|  |  |
| --- | --- |
| **lidar()** | |
| 라이다 센서 값을 반환한다. | |
| **반환 값:** | 라이다 센서 값(1도 간격, 360개 정수의 리스트) [mm]  각 값의 정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535 |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서 값을 얻는다. values = beagle.**lidar**() | |

|  |  |
| --- | --- |
| **left\_lidar()** | |
| 라이다 센서의 왼쪽 영역(인덱스 68부터 112까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 왼쪽 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 왼쪽 영역 평균 값을 얻는다. value = beagle.**left\_lidar**() | |

|  |  |
| --- | --- |
| **right\_lidar()** | |
| 라이다 센서의 오른쪽 영역(인덱스 248부터 292까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 오른쪽 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 오른쪽 영역 평균 값을 얻는다. value = beagle.**right\_lidar**() | |

|  |  |
| --- | --- |
| **front\_lidar()** | |
| 라이다 센서의 앞쪽 영역(인덱스 0부터 22까지와 338부터 359까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 앞쪽 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 앞쪽 영역 평균 값을 얻는다. value = beagle.**front\_lidar**() | |

|  |  |
| --- | --- |
| **rear\_lidar()** | |
| 라이다 센서의 뒤쪽 영역(인덱스 158부터 202까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 뒤쪽 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 뒤쪽 영역 평균 값을 얻는다. value = beagle.**rear\_lidar**() | |

|  |  |
| --- | --- |
| **left\_front\_lidar()** | |
| 라이다 센서의 왼쪽 앞 영역(인덱스 23부터 67까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 왼쪽 앞 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 왼쪽 앞 영역 평균 값을 얻는다. value = beagle.**left\_front\_lidar**() | |

|  |  |
| --- | --- |
| **right\_front\_lidar()** | |
| 라이다 센서의 오른쪽 앞 영역(인덱스 293부터 337까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 오른쪽 앞 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 오른쪽 앞 영역 평균 값을 얻는다. value = beagle.**right\_front\_lidar**() | |

|  |  |
| --- | --- |
| **left\_rear\_lidar()** | |
| 라이다 센서의 왼쪽 뒤 영역(인덱스 113부터 157까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 왼쪽 뒤 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 왼쪽 뒤 영역 평균 값을 얻는다. value = beagle.**left\_rear\_lidar**() | |

|  |  |
| --- | --- |
| **right\_rear\_lidar()** | |
| 라이다 센서의 오른쪽 뒤 영역(인덱스 203부터 247까지) 값들의 평균 값을 반환한다. | |
| **반환 값:** | 오른쪽 뒤 영역의 평균 값(정상 범위: 0 ~ 65534, 센서 값이 유효하지 않으면 65535) [mm] |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 오른쪽 뒤 영역 평균 값을 얻는다. value = beagle.**right\_rear\_lidar**() | |

|  |  |
| --- | --- |
| **resolution()** | |
| 라이다 센서의 해상도를 반환한다. | |
| **반환 값:** | 센서의 해상도 값 [도], 현재는 1도로 고정되어 있다. |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서의 해상도 값을 얻는다. value = beagle.**resolution**() | |

|  |
| --- |
| **lidar\_chart()** |
| 라이다 센서 값들을 차트로 보여준다. |
| from roboid import \* beagle = Beagle() beagle.start\_lidar() # 라이다 센서 값들을 차트로 보여준다. value = beagle.**lidar\_chart**() |

|  |  |
| --- | --- |
| **lidar\_mode(mode)** | |
| 라이다 센서의 모드를 설정한다. 라이다 센서의 모드를 따로 설정하지 않으면 기본 모드(raw)로 설정되어 있다.   |  |  | | --- | --- | | **모드** | **설명** | | “raw” | 센서에서 측정된 데이터 그대로 사용한다. | | “zero” | 센서 중심에서 일정 거리(80mm) 이내의 아주 가까운 거리 값을 0으로 한다. | | “trunc” | 센서 중심에서 일정 거리(80mm) 이내의 아주 가까운 거리 값을 경계 값(80mm)으로 한다. | | |
| **파라미터:** | mode: 라이다 센서의 모드(대소문자 구분하지 않음) |
| from roboid import \* beagle = Beagle()  beagle.**lidar\_mode**(‘zero’) beagle.start\_lidar() | |

**7. 전역 함수**

|  |
| --- |
| **dispose()** |
| 모든 로봇의 상태를 초기화하고 통신 연결을 끊는다. |
| from roboid import \* beagle = Beagle() # 모든 로봇을 폐기한다. **dispose**() |

|  |  |
| --- | --- |
| **parallel(function1, function2, ...)** | |
| 입력한 함수들을 동시에 실행한다.  입력하는 함수들은 fn()의 형태를 가져야 하며 함수 이름이 fn일 필요는 없다. 입력하는 함수의 개수는 제한이 없으며, 입력한 함수들을 모두 동시에(병렬 처리) 실행한다. | |
| **파라미터:** | * function1: 동시에 실행할 첫 번째 함수 * function2: 동시에 실행할 두 번째 함수 * ... |
| from roboid import \* beagle = Beagle() # 5초 앞으로 이동한다. def move():     beagle.move\_forward(5) # 양쪽 눈을 빨간색으로 깜박인다. def music():     while True:         beagle.note(“C4”, 0.5)  beagle.note(“E4”, 0.5)         beagle.note(“G4”, 0.5) # 음악을 연주하면서 앞으로 이동한다. **parallel**(move, music) # Ctrl+C 키를 누를 때까지 계속 기다린다. wait(-1) | |

|  |  |
| --- | --- |
| **parallel((function1, args1), (function2, args2), ...)** | |
| 입력한 함수들을 동시에 실행한다.  입력하는 함수들은 fn(args)의 형태를 가져야 하며 함수 이름이 fn일 필요는 없다. 입력하는 함수의 개수는 제한이 없으며, 입력한 함수들을 모두 동시에(병렬 처리) 실행한다. 입력한 함수 fn을 호출할 때 parallel 함수에 넣어준 args1, args2, …을 그대로 전달한다. 따라서 args1, args2, …는 어떠한 데이터 형이라도 가능하다. | |
| **파라미터:** | * function1: 동시에 실행할 첫 번째 함수 * function2: 동시에 실행할 두 번째 함수 * ... * args1: function1을 호출할 때 전달할 데이터 * args2: function2를 호출할 때 전달할 데이터 * … |
| from roboid import \* beagle1 = Beagle() beagle2 = Beagle() # 5초 앞으로 이동한다. def move(robot):     robot.move\_forward(5) # 양쪽 눈을 빨간색으로 깜박인다. def music(robot):     while True:         robot.note(“C4”, 0.5)  robot.note(“E4”, 0.5)         robot.note(“G4”, 0.5) # 음악을 연주하면서 앞으로 이동한다. **parallel**((move, beagle1), (music, beagle1), (move, beagle2), (music, beagle2)) # Ctrl+C 키를 누를 때까지 계속 기다린다. wait(-1) | |

|  |
| --- |
| **scan()** |
| 시리얼 포트 목록을 출력한다. |
| from roboid import \* # 시리얼 포트 목록을 출력한다. **scan**() |

|  |  |
| --- | --- |
| **set\_executable(execute)** | |
| 약 20msec마다 호출되는 콜백 함수를 등록한다.  콜백 함수는 execute()의 형태를 가지는 함수여야 한다. 함수 이름이 execute일 필요는 없다. 콜백 함수 내에서 디바이스에 쓴 데이터는 모두 같은 통신 패킷으로 전달되는 것이 보장된다. 콜백 함수가 등록되어 있으면 콜백 함수를 호출한 후에 하드웨어 로봇에게 명령을 전달한다. 콜백 함수는 약 20msec마다 호출되는데, 콜백 함수 내에서 시간을 끌면 안 되고 20msec 내에 모두 처리가 되도록 하여야 한다. | |
| **파라미터:** | * execute: 등록할 콜백 함수 |
| from roboid import \* beagle = Beagle() # 약 20msec마다 호출된다. def execute():     if beagle.front\_lidar() < 50:         beagle.wheels(-50, -50)     else:         beagle.wheels(50, 50) **set\_executable**(execute) # 콜백 함수를 등록한다. | |

|  |  |
| --- | --- |
| **wait(milliseconds)** | |
| 1000분의 1초 단위로 milliseconds 시간 동안 기다린다.  음수 값(예: -1)을 입력하면 영원히 기다린다. | |
| **파라미터:** | * milliseconds: 기다릴 시간(실수) [msec] |
| from roboid import \* **wait**(1000) # 1초 기다린다. **wait**(-1) # 영원히 기다린다. | |

|  |  |
| --- | --- |
| **wait\_until(evaluate)** | |
| 조건을 만족할 때까지 기다린다.  조건을 검사하는 함수는 evaluate()의 형태를 가지며 True 또는 False를 반환하는 함수여야 한다. 함수 이름이 evaluate일 필요는 없다. 조건을 검사하는 함수가 True를 반환하면 wait\_until 함수를 중지하고, False를 반환하면 wait\_until 함수에 계속 머물러 있는다. 조건을 검사하는 함수는 약 20msec마다 호출되는데, 함수 내에서 시간을 끌면 안 되고 20msec 내에 모두 처리가 되도록 하여야 한다. | |
| **파라미터:** | * evaluate: 조건을 검사하는 함수 |
| from roboid import \* beagle = Beagle() # 조건을 검사하여 True 또는 False를 반환한다. def evaluate():     return beagle.front\_lidar() < 50 # 앞쪽 라이다 값이 50보다 작을 때까지 기다린다. **wait\_until**(evaluate) | |

|  |  |
| --- | --- |
| **wait\_until(evaluate, args)** | |
| 조건을 만족할 때까지 기다린다.  조건을 검사하는 함수는 evaluate(args)의 형태를 가지며 True 또는 False를 반환하는 함수여야 한다. 함수 이름이 evaluate일 필요는 없다. 조건을 검사하는 함수가 True를 반환하면 wait\_until 함수를 중지하고, False를 반환하면 wait\_until 함수에 계속 머물러 있는다. 조건을 검사하는 함수는 약 20msec마다 호출되는데, 함수 내에서 시간을 끌면 안 되고 20msec 내에 모두 처리가 되도록 하여야 한다.  조건을 검사하는 함수 evaluate를 호출할 때 wait\_until 함수에 넣어 준 args를 그대로 전달한다. 따라서 args는 어떠한 데이터 형이라도 가능하다. | |
| **파라미터:** | * evaluate: 조건을 검사하는 함수 * args: evaluate를 호출할 때 전달할 데이터 |
| from roboid import \* beagle1 = Beagle() beagle2 = Beagle() # 조건을 검사하여 True 또는 False를 반환한다. def evaluate(robot):     return robot.front\_lidar() < 50 # 첫 번째 비글 로봇의 앞쪽 라이다 값이 50보다 작을 때까지 기다린다. **wait\_until**(evaluate, beagle1) # 두 번째 비글 로봇의 앞쪽 라이다 값이 50보다 작을 때까지 기다린다. **wait\_until**(evaluate, beagle2) | |

|  |
| --- |
| **wait\_until\_ready()** |
| 모든 로봇이 준비될 때까지 기다린다.  여러 대의 로봇이 동작을 시작하는 시점을 동일하게 하기 위해 사용한다. 예를 들어, 컴퓨터에 2개의 동글을 연결하고 Beagle 인스턴스를 2개 생성하였는데 비글 로봇의 전원을 하나만 켰다고 하자. wait\_until\_ready() 함수를 호출하지 않으면 첫 번째 비글 로봇이 통신으로 연결되었을 때 동작을 먼저 시작하게 된다. 두 번째 비글 로봇의 전원을 켜서 모든 로봇이 통신으로 연결된 후 동작을 같이 시작하게 하려면 Beagle 인스턴스를 생성한 후 wait\_until\_ready() 함수를 호출하면 된다. 생성된 인스턴스의 개수가 컴퓨터에 연결된 동글의 개수보다 적으면 생성된 인스턴스의 개수만큼 통신이 연결될 때까지 기다린다. 컴퓨터에 연결된 동글의 개수가 생성된 인스턴스의 개수보다 적으면 컴퓨터에 연결된 동글의 개수만큼 통신이 연결될 때까지 기다린다. |
| from roboid import \* beagle1 = Beagle() beagle2 = Beagle() **wait\_until\_ready**() beagle1.wheels(50, 50) beagle2.wheels(50, 50) |

|  |  |
| --- | --- |
| **when\_do(when, do)** | |
| when 함수의 조건을 만족하면 do 함수를 실행한다.  조건을 검사하는 함수는 when()의 형태를 가지며 True 또는 False를 반환하는 함수여야 한다. 함수 이름이 when일 필요는 없다. 조건을 만족할 때 실행할 함수는 do()의 형태를 가져야 하며 함수 이름이 do일 필요는 없다. 조건을 검사하는 함수가 True를 반환하면 do 함수를 실행하고, False를 반환하면 do 함수를 실행하지 않는다. 조건을 검사하는 함수는 약 20msec마다 호출되는데, 함수 내에서 시간을 끌면 안 되고 20msec 내에 모두 처리가 되도록 하여야 한다.  wait\_until 함수와는 달리 조건을 만족할 때까지 기다리는 것이 아니며, when-do 쌍을 등록만 하고 다음 줄의 코드로 넘어간다. 따라서 when\_do 함수를 여러 번 호출하는 경우 순서는 중요하지 않다. | |
| **파라미터:** | * when: 조건을 검사하는 함수 * do: 조건을 만족할 때 실행할 함수 |
| from roboid import \* beagle = Beagle() # 앞으로 기울이면 True를 반환한다. def when\_tilt\_forward():     return beagle.tilt() == 1 def do\_move\_forward():     beagle.move\_forward() # 비글 로봇을 뒤로 기울이면 True를 반환한다. def when\_tilt\_backward():     return beagle.tilt() == -1 def do\_move\_backward():     beagle.move\_backward() # 비글 로봇을 앞으로 기울이면 앞으로 이동하고 뒤로 기울면 뒤로 물러난다. **when\_do**(when\_tilt\_forward, do\_move\_forward) **when\_do**(when\_tilt\_backward, do\_move\_backward) # Ctrl+C 키를 누를 때까지 계속 기다린다. wait(-1) | |

|  |  |
| --- | --- |
| **when\_do(when, do, args)** | |
| when 함수의 조건을 만족하면 do 함수를 실행한다.  조건을 검사하는 함수는 when(args)의 형태를 가지며 True 또는 False를 반환하는 함수여야 한다. 함수 이름이 when일 필요는 없다. 조건을 만족할 때 실행할 함수는 do(args)의 형태를 가져야 하며 함수 이름이 do일 필요는 없다. 조건을 검사하는 함수가 True를 반환하면 do 함수를 실행하고, False를 반환하면 do 함수를 실행하지 않는다. 조건을 검사하는 함수는 약 20msec마다 호출되는데, 함수 내에서 시간을 끌면 안 되고 20msec 내에 모두 처리가 되도록 하여야 한다.  when 함수와 do 함수를 호출할 때 when\_do 함수에 넣어 준 args를 그대로 전달한다. 따라서 args는 어떠한 데이터 형이라도 가능하다.  wait\_until 함수와는 달리 조건을 만족할 때까지 기다리는 것이 아니며, when-do 쌍을 등록만 하고 다음 줄의 코드로 넘어간다. 따라서 when\_do 함수를 여러 번 호출하는 경우 순서는 중요하지 않다. | |
| **파라미터:** | * when: 조건을 검사하는 함수 * do: 조건을 만족할 때 실행할 함수 * args: when 함수와 do 함수를 호출할 때 전달할 데이터 |
| from roboid import \* beagle1 = Beagle() beagle2 = Beagle() # 앞으로 기울이면 True를 반환한다. def when\_tilt\_forward(beagle):     return beagle.tilt() == 1 def do\_move\_forward(beagle):     beagle.move\_forward() # 비글 로봇을 뒤로 기울이면 True를 반환한다. def when\_tilt\_backward(beagle):     return beagle.tilt() == -1 def do\_move\_backward(beagle):     beagle.move\_backward() # 비글 로봇을 앞으로 기울이면 앞으로 이동하고 뒤로 기울이면 뒤로 물러난다. **when\_do**(when\_tilt\_forward, do\_move\_forward, beagle1) **when\_do**(when\_tilt\_backward, do\_move\_backward, beagle1) **when\_do**(when\_tilt\_forward, do\_move\_forward, beagle2) **when\_do**(when\_tilt\_backward, do\_move\_backward, beagle2) # Ctrl+C 키를 누를 때까지 계속 기다린다. wait(-1) | |