

# RoboDog Python API

(주)주니랩

# 목 차

1. 객체 생성 및 초기화 .....	4
a. RoboDog() .....	4
b. Open() .....	4
c. Close() .....	5
2. RoboDog 동작 .....	6
a. gestrue() .....	6
b. leg_bend() .....	6
c. move() .....	7
d. leg() .....	8
e. motor() .....	9
f. rotate() .....	10
g. headLEDExp() .....	10
h. headLEDDraw() .....	11
i. ledPrint() .....	11
j. bodyLED() .....	12
k. sound() .....	12
l. extServo() .....	13
3. RoboDog 상태정보 .....	15
a. is_button_pressed() .....	15
b. get_battery .....	15
c. get_distance() .....	15
d. get_tilt() .....	16

e.	get_rotation().....	16
4.	Multi 제어.....	18
a.	RoboDogMultiOpen().....	18
b.	RoboDogMultiClose().....	18
c.	multi_leg_bend().....	19
d.	multi_move().....	19
e.	multi_rotate().....	19
f.	multi_leg().....	20
g.	multi_motor().....	20
5.	라즈베리파이.....	21
a.	rb_runfile().....	21
b.	is_rb_alive().....	21
c.	get_rb_data ().....	22
d.	get_rb_bytearray ().....	22

# 1. 객체 생성 및 초기화

## a. RoboDog()

<b>함수명</b>	robodog = RoboDog()
<b>설명</b>	RoboDog 클래스의 객체를 생성합니다. 여러 대의 로보독을 동시에 제어하려면 여러 개의 RoboDog 객체를 생성하면 됩니다. 로보독에서 전송된 패킷은 RoboDog(callback) 함수를 통해 확인할 수 있습니다.
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  def rcvCallBack(dog, packet):      # 로보독에서 패킷을 받으면 호출     print(packet[ROBODOG_TOF])    # 로보독 거리센서 값 출력  robodog = RoboDog(rcvCallBack)    # callback 함수 등록. callback 함수가 필요 없으면                                   # None 또는 공란으로 호출하면 됨  robodog.Open() sleep(2) robodog.Close() </pre>

## b. Open()

<b>함수명</b>	open(portName="None")
<b>설명</b>	로보독과 연결을 설정합니다. OS와 USB 동글은 COM 포트(VCP) 통신을 사용하며, USB 동글과 로보독은 블루투스 통신을 통해 연결됩니다. 파라미터가 None 이거나 공란일 경우, 컴퓨터에 연결된 COM 포트를 자동으로 검색해 연결을 시도합니다. 단, 여러 장치가 연결된 경우 의도하지 않은 포트로 연결될 수 있습니다. 이러한 상황에서는 포트 이름을 명시적으로 파라미터에 입력하여 원하는 포트로 연결을 설정할 수 있습니다.
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open("COM3")              # 명시적으로 COM 포트를 등록한다. sleep(2) robodog.Close() </pre>

c. Close()

<b>함수명</b>	Close()
<b>설명</b>	COM 포트를 해제합니다.
<b>예제</b>	<pre>from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open() sleep(2) robodog.Close()</pre>

## 2. RoboDog 동작

### a. gesture()

<b>함수명</b>	gesture(action)	
<b>설명</b>	로보독의 사전 정의된 동작을 실행합니다.	
<b>파라미터</b>	Action	준비 앉기자세 물구나무자세 기지개켜기 인사하기
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.gesture(물구나무자세) sleep(5) robodog.gesture(준비) sleep(3)  robodog.Close() </pre>	로보독이 물구나무 자세를 취한 뒤, 준비 자세로 복귀합니다.

### b. leg\_bend()

<b>함수명</b>	leg_bend(leftup, rightup, leftdw, rightdw)	
<b>설명</b>	로보독의 다리 높이를 설정합니다. 설정된 높이는 move() 및 rotate() 함수 실행 시 반영됩니다. 예를 들어, leg_bend 로 로보독의 다리를 낮게 설정한 뒤 move() 함수를 호출하면, 로보독이 포복하듯 움직입니다.	
<b>파라미터</b>	leftup	왼쪽 위다리
	rightup	오른쪽 위다리
	leftdw	왼쪽 아랫다리
	rightdw	오른쪽 아랫다리
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.move(50) </pre>	다리 높이를 다양하게 설정하며 앞으로 이동

	<pre> robodog.leg_bend(leftup=60, rightup=60, leftdw=60, rightdw=60) sleep(2) robodog.leg_bend(leftup=40, rightup=40, leftdw=40, rightdw=40) sleep(2) robodog.leg_bend(leftup=70, rightup=70, leftdw=40, rightdw=40) sleep(2) robodog.leg_bend(leftup=40, rightup=40, leftdw=60, rightdw=60) sleep(2) robodog.leg_bend(leftup=60, rightup=60, leftdw=60, rightdw=60) robodog.move(0) sleep(2) robodog.Close() </pre>	
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

c. move()

<b>함수명</b>	move(vel)	
<b>설명</b>	로보독을 vel 속도로 앞뒤로 이동시킵니다. rotate() 및 leg_bend()와 함께 사용하면 다양한 움직임을 구현할 수 있습니다.	
<b>파라미터</b>	vel	앞/뒤 움직임 속도 입력범위 : -100 ~100 0 - 멈춤 0 < vel - 앞으로 이동 vel < 0 - 뒤로 이동
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.move(50) sleep(3) robodog.move(0) sleep(1) robodog.move(-50) sleep(3) robodog.move(0) sleep(1)  robodog.Close() </pre>	로보독을 앞으로 50 속도로 2 초동안 이동 후, 뒤로 50 속도로 2 초동안 이동하고 멈춤

d. leg()

<b>함수명</b>	leg(what, height, step, height_speed=100, step_speed=100)	
<b>설명</b>	로보독의 다리 움직임을 제어하여 구부림 및 앞뒤 동작의 범위를 정밀하게 제어합니다.	
<b>파라미터</b>	what	LEFT_FRONT : 왼쪽 앞다리 LEFT_BACK : 왼쪽 뒷다리 RIGHT_BACK : 오른쪽 뒷다리 RIGHT_FRONT : 오른쪽 앞다리 FRONT_LEG : 앞다리 BACK_LEG : 뒷다리 LEFT_LEG : 왼쪽 다리 RIGHT_LEG : 오른쪽 다리 ALL_LEG : 네다리
	height	다리의 높이, 즉 구부림 정도를 설정 입력범위 : 20 ~90
	step	다리 앞/뒤 움직임 정도를 설정 입력 범위 : -90 ~ 90 (앞:+, 뒤:-)
	height_speed	구부림 속도를 설정 입력범위 : 10 ~100 Default : 100
	step_speed	앞/뒤 다리 움직임 속도를 설정 입력범위 : 10 ~100 Default : 100
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog()  robodog.Open()  robodog.leg(FRONT_LEG, height=70, step=0) robodog.leg(BACK_LEG, height=30, step=0) sleep(1) robodog.leg(FRONT_LEG, height=30, step=0) robodog.leg(BACK_LEG, height=70, step=0) sleep(1) robodog.leg(ALL_LEG, height=60, step=0) sleep(1)  robodog.Close() </pre>	<p>앞발을 70, 뒷발을 30 으로 설정하면 로보독 몸체가 위를 바라보게 되며, 앞발을 30, 뒷발을 70 으로 설정하면 아래쪽을 바라보게 됨</p> <p>네다리 모두 60 높이로 설정하면 로보독 몸체가 평행하게 서 있게 됨.</p>



e. motor()

<b>함수명</b>	motor(what, shoulder, knee, shld_speed=100, knee_speed=100)	
<b>설명</b>	로보독의 다리 관절에 있는 모터의 회전 각도를 정밀하게 제어합니다.	
<b>파라미터</b>	what	LEFT_FRONT : 왼쪽 앞다리 LEFT_BACK : 왼쪽 뒷다리 RIGHT_BACK : 오른쪽 뒷다리 RIGHT_FRONT : 오른쪽 앞다리 FRONT_LEG : 앞다리 BACK_LEG : 뒷다리 LEFT_LEG : 왼쪽 다리 RIGHT_LEG : 오른쪽 다리 ALL_LEG : 네다리
	shoulder	어깨 모터 각도 설정 입력범위 : -90 ~90(단위:도) 머리방향: +, 꼬리방향 :-
	knee	무릎 모터 각도 설정 입력 범위 : -90 ~ 70(단위:도) 머리방향: +, 바닥방향 :-
	shld_speed	어깨 모터 속도 설정 입력범위 : 10 ~100 Default : 100
	knee_speed	무릎 모터 속도 설정 입력범위 : 10 ~100 Default : 100
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.motor(FRONT_LEG, shoulder=-10, knee=-90, shld_speed=10, knee_speed=50) sleep(1) robodog.motor(FRONT_LEG, shoulder=-80, knee=70, shld_speed=30, knee_speed=60) sleep(1)  robodog.Close()           </pre>	<p>motor() 명령으로 앞다리를 길게 뻗었다 구부려 인사하는 동작을 표현</p> <p>앞발 어깨모터를 -10, 무릎모터를 -90 으로 설정하면 로보독 앞다리가 길게 뻗게 되 머리가 위를 바라보고, 어깨모터를 -80, 무릎모터를 70 으로 설정하면 다리를 구부리게 되어 머리가 아래를 바라보게 됨.</p> <p>어깨모터와 무릎모터의 회전해야하는 각이 서로 다를때는 각각의 회전속도를 설정하면 좀더 자연스러운 움직임을 구현할 수 있음</p>



f. rotate()

<b>함수명</b>	rotate(degree, degVel)	
<b>설명</b>	로보독을 좌우로 회전시킵니다. rotate() 및 leg_bend()와 함께 사용하면 다양한 움직임을 구현할 수 있습니다.	
<b>파라미터</b>	degree	좌/우 회전 각도 deg 값은 Open() 또는 gesture(0) 일 때 방향을 기준으로 합니다. 입력범위 : -1000 ~1000(단위:도) 0 < deg - 시계방향으로 회전 vel < 0 - 반시계방향으로 회전
	degVel	회전 속도를 설정
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  for n in range(1, 5):     robodog.rotate(degree=90*n, degVel=100)     sleep(2) robodog.rotate(degree=0, degVel=100) sleep(8) robodog.Close() </pre>	<p>로보독을 시계방향으로 90 도씩 4 번 회전하여 총 360 도를 회전. 그 후, degree =0 으로 설정하여 처음방향 (반시계방향으로 360 도)으로 회전.</p>

g. headLEDExp()

<b>함수명</b>	headLEDExp(what)		
<b>설명</b>	로보독의 사전 정의된 HeadLED 표현을 출력합니다.		
<b>파라미터</b>	what	초롱초롱	깜박
		ILOVEU	깜박
<b>파라미터</b>	what	눈감기	깜박
		감사	깜박
<b>파라미터</b>	what	고마워요	깜박
			깜박
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.headLEDExp(초롱초롱) </pre>	<p>로보독의 HeadLED 에 '초롱초롱' 표현을 출력합니다.</p>	

	sleep(5) robodog.Close()	
--	-----------------------------	--

h. headLEDDraw()

<b>함수명</b>	headLEDDraw(leftLED, rightLED)	
<b>설명</b>	HeadLED 를 제어합니다. 파라미터는 8 바이트의 bytearray 로, 각 바이트는 Dot Matrix 의 각 줄을 나타냅니다. 예) leftLED[2] =0x5F #위에서 3 번째 줄 LED 를 [OFF ON OFF ON ON ON ON ON]로 표현 leftLED[1] =0x38 #위에서 2 번째 줄 LED 를 [OFF OFF ON ON ON OFF OFF OFF]로 표현	
<b>파라미터</b>	leftLED	왼쪽 headLED 설정
	rightLED	오른쪽 headLED 설정
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  rightLED = bytearray([0x00, 0x7E, 0x18, 0x18, 0x18, 0x18, 0x7E, 0x00]) leftLED = bytearray([0x00, 0x66, 0xFF, 0xFF, 0x7E, 0x3C, 0x18, 0x00]) robodog.headLEDDraw(leftLED, rightLED) sleep(5) robodog.Close() </pre>	
	 <pre> rightLED[0]=0x00 rightLED[1]=0x7E rightLED[2]=0x18 rightLED[3]=0x18 rightLED[4]=0x18 rightLED[5]=0x18 rightLED[6]=0x7E rightLED[7]=0x00 leftLED[0]=0x00 leftLED[1]=0x66 leftLED[2]=0xFF leftLED[3]=0xFF leftLED[4]=0x7E leftLED[5]=0x3C leftLED[6]=0x18 leftLED[7]=0x00 </pre>	

i. ledPrint()

<b>함수명</b>	ledPrint(leftChar, rightChar)	
<b>설명</b>	로봇의 HeadLED 에 문자를 출력합니다.	
<b>파라미터</b>	leftChar	왼쪽 headLED 에 출력할 문자

	rightChar	오른쪽 headLED 에 출력할 문자
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open() left = ord('B') right = ord('A') robodog.ledPrint(leftChar=left, rightChar=right) sleep(5) robodog.Close() </pre>	로보독 오른쪽 headLED 에 'A', 왼쪽 headLED 에 'B'를 출력합니다.

j. bodyLED()

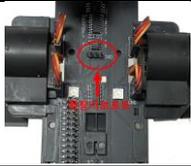
<b>함수명</b>	bodyLED(what, red, green, blue)	
<b>설명</b>	로보독의 BodyLED 의 R, G, B 색상을 제어합니다.	
<b>파라미터</b>	what	제어 할 LED 의 번호를 bit 로 설정한다. 예) ①, ③ 번 선택 => 0x05 ①, ②, ③, ④ 번 선택 => 0x0F
	red	빨간색 범위(0 ~ 255)
	green	초록색 범위(0 ~ 255)
	blue	파란색 범위(0 ~ 255)
		
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.bodyLED(what=1, red=255, green=0, blue=0) robodog.bodyLED(what=2, red=0, green=255, blue=0) robodog.bodyLED(what=4, red=0, green=0, blue=255) robodog.bodyLED(what=8, red=255, green=255, blue=255) sleep(5) robodog.Close() </pre>	로보독 body LED 를 ①번 LED 빨간색 ②번 LED 초록색 ③번 LED 파란색 ④번 LED 백색으로 설정

k. sound()

<b>함수명</b>	sound(thema, track, volume)
<b>설명</b>	로보독에 저장된 소리를 재생합니다.

파라미터	thema	개소리 명령어 효과음 피아노				
	track	개소리	명령어	효과음	피아노	
		멍멍	안녕	레이저	도	슬
으르렁		기다려	모터회전	레	라	
화난		비켜	띠리리	미	시	
신음		출발	외계신호	파	H도	
거친숨			동작			
			충돌			
volume	크게 중간 작게					
예제	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  robodog.sound(thema=개소리, track=멍멍, volume=크게) sleep(2)  robodog.Close() </pre>					

### I. extServo()

함수명	extServo(degree)		
설명	로보독 확장 서보모터를 제어한다.		
파라미터	degree	서보모터 각도 범위 -90 ~90 도	
예제	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog() robodog.Open()  while True:     robodog.extServo(-90) </pre>		확장서보모터를 -90 와 90 도 반복 회전한다.

	<pre>sleep(1) robodog.extServo(90) sleep(1)  robodog.Close()</pre>	
--	--------------------------------------------------------------------	--

### 3. RoboDog 상태정보

#### a. is\_button\_pressed()

<b>함수명</b>	is_button_pressed()	
<b>설명</b>	로보독 전원 버튼 옆에 위치한 택트스위치의 눌림 상태를 반환합니다. 1 - 눌림 0 - 안눌림	
<b>예제</b>	<pre> from pyjuni.robodog import * robodog = RoboDog() robodog.Open()  while True :     if robodog.is_button_pressed():         robodog.sound(개소리, 멍멍, 크게)         sleep(2)         sleep(0.1)  robodog.Close() </pre>	0.1 초마다 버튼상태를 확인하여 버튼이 눌렸으면 '멍멍' 소리를 낸다.

#### b. get\_battery

<b>함수명</b>	get_battery()	
<b>설명</b>	로보독의 배터리 잔량(%)을 반환합니다	
<b>예제</b>	<pre> from pyjuni.robodog import * robodog = RoboDog() robodog.Open()  sleep(0.5) print("Battery : ",robodog.get_battery(), "%") sleep(0.5) robodog.Close() </pre>	

#### c. get\_distance()

<b>함수명</b>	get_distance()	
<b>설명</b>	로보독의 HeadLED 사이에 위치한 거리 센서(TOF)에서 측정한 값을 반환합니다. (단위: cm)	
<b>예제</b>	from pyjuni.robodog import *	로보독이 앞으로 이동하면서 0.1 초

	<pre> robodog = RoboDog() robodog.Open()  sleep(0.5) robodog.move(50) while robodog.get_distance() &gt; 20 :     sleep(0.1)  robodog.move(0) sleep(0.5)  robodog.Close() </pre>	<p>간격으로 20cm 앞에 장애물이 있는지 확인 장애물이 감지되면 즉시 이동을 멈추고 프로그램을 종료</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------

d. get\_tilt()

<b>함수명</b>	get_tilt()	
<b>설명</b>	로보독에 내장된 자이로센서의 기울기 값을 반환합니다.	
<b>예제</b>	<pre> from pyjuni.robodog import * robodog = RoboDog() robodog.Open()  sleep(1) org_tilt = robodog.get_tilt() new_tilt = robodog.get_tilt()  while org_tilt == new_tilt :     sleep(0.1)     new_tilt = robodog.get_tilt()  robodog.sound(개소리, 멍멍, 크게) sleep(1)  robodog.Close() </pre>	<p>로보독이 외부 접촉을 감지하면 즉시 '멍멍' 소리를 내며 프로그램 종료 외부 접촉을 감지하는 방법은 tilt 값의 변화량을 보고 판단.</p>

e. get\_rotation()

<b>함수명</b>	get_rotation()	
<b>설명</b>	move(), leg_bend(), rotate() 등의 명령 실행 시점을 기준으로 회전 각도(degree)를 반환합니다.	

<p><b>예제</b></p>	<pre> from pyjuni.robodog import * robodog = RoboDog() robodog.Open()  sleep(0.5) robodog.move(70) robodog.rotate(1000, 70) while robodog.get_rotation() &lt; 360 :     sleep(0.05) robodog.rotate(-1000, 70) while robodog.get_rotation() &gt; 0 :     sleep(0.05)  robodog.move(0) sleep(0.5) robodog.Close() </pre>	<p>8자 돌기</p> <p>rotate() 명령으로 로보독을 회전시킨 후, 실시간으로 로보독의 실제 회전 각도를 모니터링함. 로보독이 원하는 각도에 도달할 때까지 대기.</p> <p>360도 회전을 완료하면 반대 방향으로 회전하여 8자 돌기 모양의 움직임을 수행함.</p>
------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------

## 4. Multi 제어

### a. RoboDogMultiOpen()

<b>함수명</b>	RoboDogMultiOpen( <b>nameList=None, receiveCallback=None</b> )	
<b>설명</b>	여러 대의 로보독 객체를 생성하고 open 한 뒤, 객체들을 리스트 형태로 반환합니다. nameList 가 None 인 경우, PC 에 연결된 동글의 개수만큼 객체를 생성하여 반환합니다. 단, 반환된 객체 리스트의 순서는 보장되지 않습니다. 순서를 명시적으로 지정하려면 nameList 에 COM 포트 리스트를 입력하면 됩니다.	
<b>예제</b>	<pre> 예 1) from pyjuni.robodog import *  robodogs = RoboDogMultiOpen() for n in range(len(robodogs)):     robodogs[n].leg( ALL_LEG, height=60, step=0) sleep(1) RoboDogMultiClose(robodogs)  예 2) from pyjuni.robodog import *  port_list = ["COM8", "COM31"] robodogs = RoboDogMultiOpen(port_list) for n in range(len(robodogs)):     robodogs[n].leg( ALL_LEG, height=60, step=0) sleep(1) RoboDogMultiClose(robodogs) </pre>	<pre> 예 1) PC 에 연결된 로보독을 검색하여 객체를 생성하고, 이를 리스트로 반환. 생성된 로보독 객체의 leg 메서드를 호출해 높이 60 으로 1 초 동안 서 있는 동작을 수행한 뒤, 프로그램을 종료.  예 2) COM 포트를 명시적으로 지정해 로보독 객체 생성. </pre>

### b. RoboDogMultiClose()

<b>함수명</b>	RoboDogMultiClose(dogs)
<b>설명</b>	입력된 robodogs 객체들의 COM 포트를 해제합니다

c. multi\_leg\_bend()

<b>함수명</b>	multi_leg_bend(dogs, leftup, rightup, leftdw, rightdw)
<b>설명</b>	dogs 리스트에 저장된 각 객체의 leg_bend() 메서드를 호출합니다.
<b>예제</b>	<pre> from pyjuni.robodog import *  robodogs = RoboDogMultiOpen() multi_leg_bend(robodogs, 60, 60, 60, 60)  sleep(1)  RoboDogMultiClose(robodogs) </pre>

d. multi\_move()

<b>함수명</b>	multi_move(dogs, vel)
<b>설명</b>	dogs 리스트에 저장된 각 객체의 move() 메서드를 호출합니다.
<b>예제</b>	<pre> from pyjuni.robodog import *  robodogs = RoboDogMultiOpen() multi_move (robodogs, 50)  sleep(1)  RoboDogMultiClose(robodogs) </pre>

e. multi\_rotate()

<b>함수명</b>	multi_rotate(dogs, degree, degVel)
<b>설명</b>	dogs 리스트에 저장된 각 객체의 rotate() 메서드를 호출합니다.
<b>예제</b>	<pre> from pyjuni.robodog import *  robodogs = RoboDogMultiOpen() multi_rotate (robodogs, degree=90, degVel=100)  sleep(5)  RoboDogMultiClose(robodogs) </pre>

f. multi\_leg()

<b>함수명</b>	multi_leg(dogs, what, height, step, height_speed=100, step_speed=100)
<b>설명</b>	dogs 리스트에 저장된 각 객체의 leg() 메서드를 호출합니다.
<b>예제</b>	<pre> from pyjuni.robodog import *  robodogs = RoboDogMultiOpen() multi_leg(robodogs, what=ALL_LEG, height=60, step=0) sleep(2) RoboDogMultiClose(robodogs) </pre>

g. multi\_motor()

<b>함수명</b>	multi_motor(dogs, what, shoulder, knee, shld_speed=100, knee_speed=100)
<b>설명</b>	dogs 리스트에 저장된 각 객체의 motor() 메서드를 호출합니다.
<b>예제</b>	<pre> from pyjuni.robodog import *  robodogs = RoboDogMultiOpen() multi_motor (robodogs, what=FRONT_LEG, shoulder =40, knee=40) sleep(2) RoboDogMultiClose(robodogs) </pre>

## 5. 라즈베리파이

### a. rb\_runfile()

<b>함수명</b>	rb_runfile(filename="")
<b>설명</b>	로보독에 내장된 라즈베리파이에서 실행할 파이썬 파일을 설정합니다.
<b>파라미터</b>	Filename - 라즈베리파이에서 실행될 파이썬 파일이름
<b>예제</b>	<pre>from time import sleep from pyjuni.robodog import *  robodog = RoboDog()  runfile = robodog.rb_runfile("color_detect.py") sleep(1) print(runfile) robodog.Close()</pre>

### b. is\_rb\_alive()

<b>함수명</b>	is_rb_alive ()
<b>설명</b>	로보독에 내장된 라즈베리파이가 부팅 후 정상 동작하는지 상태를 알려줍니다.
<b>예제</b>	<pre>from time import sleep from pyjuni.robodog import *  robodog = RoboDog()  while True:     if robodog.is_rb_alive():         break     sleep(0.1) print("Raspberry Pi is alive.") robodog.Close()</pre>

c. get\_rb\_data ()

<b>함수명</b>	get_rb_data(index)
<b>설명</b>	로보독에 내장된 라즈베리파이에서 전송한 데이터 리스트 중 지정된 index 위치의 값을 반환합니다. 이 값은 라즈베리파이에서 실행 중인 파이썬 파일에서 전송한 데이터로, 파이썬 코드에 따라 데이터의 의미가 달라질 수 있습니다.
<b>파라미터</b>	Index - 범위 : 0~3
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  robodog = RoboDog()  print(robodog.get_rb_data(0))  robodog.Close() </pre>

d. get\_rb\_bytearray ()

<b>함수명</b>	get_rb_bytearray ()
<b>설명</b>	로보독에 내장된 라즈베리파이에서 전송한 데이터를 bytearray 로 반환합니다. 이 값은 라즈베리파이에서 실행 중인 파이썬 파일에서 전송한 데이터로, 파이썬 코드에 따라 데이터의 의미가 달라질 수 있습니다.
<b>예제</b>	<pre> from time import sleep from pyjuni.robodog import *  go_cmd = "GO" go_bytes = go_cmd.encode('utf-8') stop_cmd = "STOP" stop_bytes = stop_cmd.encode('utf-8') robodog = RoboDog()  while True:     rb_msg = robodog.get_rb_bytearray()     if go_bytes in rb_msg:         robodog.move(50)     if stop_bytes in rb_msg:         robodog.move(0)     sleep(0.1) robodog.Close() </pre>