BR280 机械臂 Python API 说明书

目录

1 Python 简单使用 2
2 关节控制 2
2.1 单关节控制 3
2.2 多关节控制 3
2.3 案例 5
3 坐标控制 5
3.1 单参数坐标 6
3.2 多参数坐标 6
3.3 案例 7
4 夹爪控制 7
4.1 案例 8
5 使用案例 10
5.1 LED 闪烁 10
5.2 控制机械回原点 10
5.3 单关节运动 11
5.4 多关节运动 12
5.5 控制机械臂左右摆动 13
6 API 方法详细说明 14
6.1 机械臂整体运行状态 15
6.2 机械臂运行状态和设置 15
6.3 输入程序控制模式（MDI 模式） 17
6.4 JOG 模式和操作 20
6.5 舵机控制与操作 21
6.6 夹爪控制 22

1 Python 简单使用

LED 闪烁

新建一个 Python 文件，输入以下代码可执行 LED 闪烁。

```
from pymycobot.mycobot import MyCobot

import time#以上需写在代码开头，意为导入项目包



# MyCobot 类初始化需要两个参数：串口和波特率

mc = MyCobot("/dev/arm", 115200)



i = 7

#循环 7 次 while i > 0:

    mc.set_color(0,0,255) #蓝灯亮

    time.sleep(2)    #等 2 秒

    mc.set_color(255,0,0) #红灯亮

    time.sleep(2)    #等 2 秒

    mc.set_color(0,255,0) #绿灯亮

    time.sleep(2)    #等 2 秒

    i -= 1
```

运行结果：机器人顶部灯光蓝色、红色、绿色以 2 秒的间隔持续闪烁 7 次

2 关节控制

对于串联式多关节机器人，关节空间的控制是针对机器人各个关节的变量进行的控制，目标是让机器人各个关节按照一定速度达到目标位置。

> **注意：**在设置角度时，不同的机械臂相对应的数值是不一样的，详情角度限值见下表

- 六轴

关节 ID	限值

关节 ID	限值
1	-170~170
2	-170~170
3	-170~170
4	-170~170
5	-170~170
6	无限值

2.1 单关节控制

对于关节角度的取值范围：六轴机器人约-170 ～ 170，详见上方表格图

1、`send_angle(id, degree, speed)`

- **功能：** 发送指定的单个关节运动至指定的角度
- **参数说明：**
  - `id`: 代表机械臂的关节，六轴有六个关节，有特定的表示方法。 关节一的表示法：`Angle.J1.value`。(也可以用数字 1-6 来表示）
  - `degree`: 表示关节的角度
  - `speed`：表示机械臂运动的速度，范围 0~100。
- **返回值：** 无

2、`set_encoder(joint_id, encoder)`

- **功能：** 发送指定的单个关节运动至指定的电位值。
- **参数说明：**
  - `joint_id`: 代表机械臂的关节，六轴有六个关节，有特定的表示方法。 关节一的表示法：`Angle.J1.value`。(也可以用数字 1-6 来表示）
  - `encoder`:表示机械臂的电位值，取值范围是 0 ~ 4096
- **返回值：** 无

2.2 多关节控制

1、`get_angles()`

- **功能：** 获取所有关节角度。

- **返回值：** list 一个浮点值的列表代，表所有关节的角度.

2、 `send_angles(degrees, speed)`

- **功能：** 发送所有角度给机械臂所有关节
- **参数说明：**
  - `degrees`: (List[float])包含所有关节的角度 ，六轴机器人有六个关节所以长度为 6，表示方法为：[20,20,20,20,20,20],取值范围：约-170 ～ 170，详见上方表格图
  - `speed`: 表示机械臂运动的速度，取值范围是 0-100。
- **返回值：** 无

3、 `set_encoders(encoders, sp)`

- **功能：** 发送电位值给机械臂所有关节
- **参数说明：**
  - `encoder`:表示机械臂的电位值，取值范围是 0 ~ 4096，六轴长度为 6，表示方法为：[2048,2048,2048,2048,2048,2048]
  - `sp`: 表示机械臂运动的速度，取值范围是 0-100。
- **返回值：** 无

4、 `sync_send_angles(degrees, speed, timeout=7)`

- **功能：** 同步发送角度，到达目标点返回
- **参数说明：**
  - `degrees`: 每个关节的角度值列表 `List[float]`。
  - `speed`: (`int`)机械臂运动的速度，取值范围是 0-100。
  - `timeout`: 时间默认 7s。

5、 `get_radians()`

- **功能：** 获取所有关节的弧度。
- **返回值：** `list` 包含所有关节弧度值的列表.

6、 `send_radians(radians, speed)`

- **功能：** 发送弧度值给机械臂所有关节
- **参数说明：**

- `radians`:表示机械臂的弧度值，取值范围是 -5~5。

- **返回值：** `list`包含所有关节弧度值的列表.

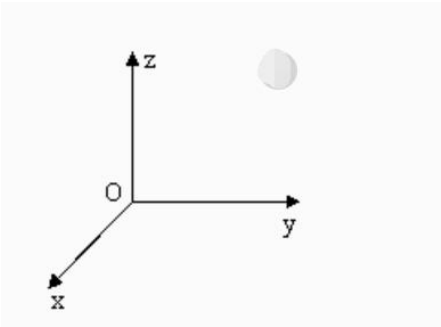
2.3 案例

更多案例参考和运行结果视频请查看 7_使用案例

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle
import time
# MyCobot 类初始化需要两个参数： 第一个是串口字符串，第二个是波特率
mc = MyCobot("/dev/arm", 115200)
# 通过传递角度参数，让机械臂每个关节移动到对应[0, 0, 0, 0, 0, 0]的位置
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2.5)
# 让关节 1 移动到 90 这个位置
mc.send_angle(Angle.J1.value, 90, 50)# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2)
# 以下代码可以让机械臂左右摇摆# 设置循环次数
while num > 0:
    # 让关节 2 移动到 50 这个位置
    mc.send_angle(Angle.J2.value, 50, 50)

    # 设置等待时间，确保机械臂已经到达指定位置
    time.sleep(1.5)

    # 让关节 2 移动到-50 这个位置
    mc.send_angle(Angle.J2.value, -50, 50)

    # 设置等待时间，确保机械臂已经到达指定位置
    time.sleep(1.5)

    num -= 1
# 让机械臂缩起来。你可以手动摆动机械臂，然后使用 get_angles()函数获得坐标数列，# 通过该函数让机械臂到达你所想的位置。
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)
# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2.5)
```

3 坐标控制

主要用于实现智能规划路线让机械臂从一个位置到另一个指定位置。分为`[x,y,z,rx,ry,rz]`，其中`[x,y,z]`表示的是机械臂头部在空间中的位置（该坐标系为直角坐标系），`[rx,ry,rz]`表示的是机械臂头部在该点的姿态（该坐标系为欧拉坐标）。算法的实现以及欧拉坐标的表示需要一定的学术知识，这里不对其过多的讲解，我们只要懂得直角坐标系就可以很好的使用这个函数了。



3.1 单参数坐标

`send_coord(id,coord,speed)`

- **功能：** 发送单个坐标值给机械臂进行移动
- **参数说明：**
  - `id`:代表机械臂的坐标，六轴有六个坐标，有特定的表示方法。 X 坐标的表示法：`Coord.X.value`，也有简易的表示方法：如 X 轴可以填写 1,Y 填写 2,以此类推
  - `coord`: 输入您想要到达的坐标值，取值范围 -300 – 300
  - `speed`: 表示机械臂运动的速度，范围是 0-100。
- **返回值：** 无

3.2 多参数坐标

1、 `get_coords()`

- **功能：** 获取当前坐标和姿态。
- **返回值：** `list` 包含坐标和姿态的列表。
  - 六轴：长度为 6，依次为 `[x, y, z, rx, ry, rz]`

2、 `send_coords(coords, speed, mode)`

- **功能：** 发送整体坐标和姿态,让机械臂头部从原来点移动到您指定点。
- **参数说明：**
  - `coords`: [x,y,z,rx,ry,rz]的坐标值，长度为 6，取值范围 -300 – 300
  - `speed`: 表示机械臂运动的速度，范围是 0-100。
  - `mode`: (`int`): 取值限定 0 和 1。
    - 0 表示机械臂头部移动的路径为非线性，即随机规划路线，只要机械臂头部以保持规定的姿态移动到指定点即可。
    - 1 表示机械臂头部移动的路径为线性的，即智能规划路线让机械臂头部以直线的方式移动到指定点。**（注意：有时机械臂不会移动，是该方法无法达到。建议改成模式 0。）**
- **返回值：** 无

3.3 案例

更多案例参考和运行结果视频请查看 7_使用案例

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Coord
import time
# MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
mc = MyCobot("/dev/arm", 115200)
# 获取当前头部的坐标以及姿态
coords = mc.get_coords()
print(coords)
# 智能规划路线，让头部以非线性的方式到达[59.9,-65.8,250.7]这个坐标，以及保持
[-50.99,83.14,-52.42]这个姿态，速度为 80mm/s
mc.send_coords([59.9, -65.8, 250.7, -50.99, 83.14, -52.42], 80, 0)
# 设置等待时间 1.5 秒
time.sleep(1.5)
# 智能规划路线，让头部以非线性的方式到达[59.9,-65.8,350.7]这个坐标，以及保持
[-50.99,83.14,-52.42]这个姿态，速度为 80mm/s
mc.send_coords([59.9, -65.8, 350.7, -50.99, 83.14, -52.42], 80, 0)
# 设置等待时间 1.5 秒
time.sleep(1.5)
# 仅改变头部的 x 坐标，设置头部的 x 坐标为-40。让其智能规划路线让头部移动到改变后的位置，，速度
为 70mm/s
mc.send_coord(Coord.X.value, -40, 70)
```

4 夹爪控制

主要目的是对机械臂进行检查、操作，需要先在机械臂上安装连接好夹爪。

自适应夹爪将夹爪插在 atom 上面的引脚上，具体看下图：

1、`is_gripper_moving( )`

- **功能：** 判断夹爪是否正在运行。
- **返回值：**
  - `0`：表示机械臂的夹爪没有运行
  - `1`：表示机械臂的夹爪正在运行
  - `-1`：表示出错

2、`set_gripper_value(value, speed)`

- **功能：** 让夹爪以指定的速度转动到指定的位置。
- **参数说明：**
    - `value`：表示夹爪所要到达的位置，取值范围 0~256。
    - `speed`：表示以多少的速度转动，取值范围 0~100。
- **返回值：** 无

3、`get_gripper_value()`

- **功能：** 获取夹爪的 `encoder` 数据信息。
- **返回值：** 夹爪的数据信息。

4、`set_gripper_int()`

- **功能：** 设置夹爪初始化位置，设置当前位置为 2048
- **返回值：** 无

5、`set_gripper_state(flag, speed)`

- **功能：** 让夹爪以指定的速度进入到指定的状态。
- **参数说明：**
    - `flag`：1 表示夹爪合拢状态，0 表示夹爪打开状态。
    - `speed`：表示以多快的速度达到指定的状态，取值范围 0~100。
- **返回值：** 无

4.1 案例

更多案例参考和运行结果视频请查看 7_使用案例

```
from pymycobot.mycobot import MyCobot

import time#输入以上代码导入工程所需要的包

def gripper_test(mc):

    print("Start check IO part of api\n")

    # 检测夹爪是否正在移动

    flag = mc.is_gripper_moving()

    print("Is gripper moving: {}".format(flag))

    time.sleep(1)
```

```
    # 以 30 的速度让夹爪到达 255 状态
    mc.set_gripper_value(255, 30)
    time.sleep(3)
    # 以 30 的速度让夹爪到达 40 状态
    mc.set_gripper_value(40, 30)
    time.sleep(3)

    # 设置夹爪的状态，让其以 70 的速度快速打开爪子
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # 设置夹爪的状态，让其以 70 的速度快速收拢爪子
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # 获取夹爪的值
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":
    # 初始化一个 MyCobot 对象
    # MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
    mc = MyCobot("/dev/arm", 115200)

    # 让其移动到零位
    mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
    time.sleep(3)
    gripper_test(mc)
```

5 使用案例

以下为各种使用的案例以及运行结果视频，您可以复制代码进行使用或者修改

注意： 各款设备的对应的波特率不尽相同，使用时请查阅资料了解其波特率，串口编号可通过计算器设备管理器或串口助手进行查看。

5.1 LED 闪烁

```
from pymycobot.mycobot import MyCobot

import time#以上需写在代码开头，意为导入项目包

# MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
mc = MyCobot("/dev/arm", 115200)

i = 7

#循环 7 次

while i > 0:

    mc.set_color(0,0,255) #蓝灯亮

    time.sleep(2)    #等 2 秒

    mc.set_color(255,0,0) #红灯亮

    time.sleep(2)    #等 2 秒

    mc.set_color(0,255,0) #绿灯亮

    time.sleep(2)    #等 2 秒

    i -= 1
```

5.2 控制机械回原点

```
from pymycobot.mycobot import MyCobot

# MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
mc = MyCobot("/dev/arm", 115200)

# 检测机械臂是否可烧入程序

if mc.is_controller_connected() != 1:

    print("请正确连接机械臂进行程序写入")
```

```
    exit(0)

# 对机械臂进行微调，确保调整后的位置所有卡口都对齐了

# 以机械臂卡口对齐为准，这里给出的仅是个案例

mc.send_angles([0, 0, 0, 0, 0, 0], 30)

# 对此时的位置进行校准，校准后的角度位置表示[0,0,0,0,0,0]，电位值表示
[2048,2048,2048,2048,2048,2048]

# 该 for 循环相当于 set_gripper_ini()这个方法

#for i in range(1, 7):

    #mc.set_servo_calibration(i)
```

5.3 单关节运动

```
from pymycobot import MyCobot

from pymycobot.genre import Angle

import time

# MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
mc = MyCobot("/dev/arm", 115200)

# 机械臂复原

mc.send_angles([0, 0, 0, 0, 0, 0], 40)

time.sleep(3)



# 控制关节 3 运动 70°

mc.send_angle(Angle.J3.value,70,40)

time.sleep(3)



# 控制关节 4 运动-70°

mc.send_angle(Angle.J4.value,-70,40)

time.sleep(3)
```

```
# 控制关节 1 运动 90°

mc.send_angle(Angle.J1.value,90,40)

time.sleep(3)



# 控制关节 5 运动-90°

mc.send_angle(Angle.J5.value,-90,40)

time.sleep(3)



# 控制关节 5 运动 90°

mc.send_angle(Angle.J5.value,90,40)

time.sleep(3)
```

5.4 多关节运动

```
import time

from pymycobot import MyCobot

# MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
mc = MyCobot("/dev/arm", 115200)



# 机械臂复原归零

mc.send_angles([0,0,0,0,0,0],50)

time.sleep(2.5)



# 控制多个关节转动的不同角度

mc.send_angles([90,45,-90,90,-90,90],50)

time.sleep(2.5)



# 机械臂复原归零

mc.send_angles([0,0,0,0,0,0],50)
```

```
time.sleep(2.5)



# 控制多个关节转动的不同角度

mc.send_angles([-90,-45,90,-90,90,-90],50)

time.sleep(2.5)
```

5.5 控制机械臂左右摆动

```
from pymycobot.mycobot import MyCobot

from pymycobot.genre import Angle

import time

# MyCobot 类初始化需要两个参数： 第一个是串口字符串， 第二个是波特率
mc = MyCobot("/dev/arm", 115200)



# 获得当前位置的坐标

angle_datas = mc.get_angles()

print(angle_datas)



# 用数列传递传递坐标参数，让机械臂移动到指定位置

mc.send_angles([0, 0, 0, 0, 0, 0], 50)

print(mc.is_paused())

# 设置等待时间，确保机械臂已经到达指定位置

# while not mc.is_paused():

time.sleep(2.5)



# 让关节 1 移动到 90 这个位置

mc.send_angle(Angle.J1.value, 90, 50)
```

```
# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2)


# 设置循环次数
num = 5


# 让机械臂左右摇摆
while num > 0:
    # 让关节 2 移动到 50 这个位置
    mc.send_angle(Angle.J2.value, 50, 50)
    # 设置等待时间，确保机械臂已经到达指定位置
    time.sleep(1.5)
    # 让关节 2 移动到-50 这个位置
    mc.send_angle(Angle.J2.value, -50, 50)
    # 设置等待时间，确保机械臂已经到达指定位置
    time.sleep(1.5)
    num -= 1


# 让机械臂缩起来。你可以手动摆动机械臂，然后使用 get_angles()函数获得坐标数列，
# 通过该函数让机械臂到达你所想的位置。
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)


# 设置等待时间，确保机械臂已经到达指定位置
time.sleep(2.5)
```

6 API 方法详细说明

在使用下列函数接口的时候请先在开头导入我们的 API 库，否则无法运行成功，即输入以下代码：

```
from pymycobot import MyCobot
```

个别函数接口有返回值，但是直接输入代码，返回的结果是没有返回值的，需要使用 `print` 函数，把结果打印出来，比如你想要获取机械臂当前设置的速度值可使用 `get_speed()`，直接输入该函数是没有结果的，正确写法是：`print(get_speed())`即可把速度值打印出来

6.1 机械臂整体运行状态

1、 `power_on()`

- **功能：** 机械臂上电。
- **返回值：** 无

2、 `power_off()`

- **功能：** 机械臂断电，所有功能将失效。注意：断电后无法让机械臂放松，即 `set_free_mode()`失效。
- **返回值：** 无

3、 `is_power_on()`

- **功能：** 判断机械臂是否上电。
- **返回值：** 1- 机械臂已通电 0 - 机械臂已断电 -1- 错误

4、 `release_all_servos`

- **功能：** 设置机械臂为自由移动模式。(能手动自由摆动机械臂)
- **返回值：** 无

5、 `is_controller_connected`

- **功能：** 判断是否与 Atom 连接。
- **返回值：** 1- 已连接 0 - 未连接 -1- 错误

6.2 机械臂运行状态和设置

1、 `pause()`

- **功能：** 让机械臂暂停当前运动。
- **返回值：** 无

2、 `stop()`

- **功能：** 让机械臂停止所有运动。
- **返回值：** 无

3、 `resume()`

- **功能：** 让机械臂恢复之前所设置的运动。
- **返回值：** 无

4、`is_paused()`

- **功能：** 判断机械臂是否为暂停状态。
- **返回值：** 1 - 已经暂停 0 - 没有暂停 -1 - 错误

5、`get_speed()`

- **功能：** 获取机器人的运动速度。
- **返回值：** 机器人当前所设定的运行速度，范围在 1-100

6、`set_speed(speed)`

- **功能：** 设置机器人的运动速度。
- **参数说明：** speed: 给入你想要设置的速度范围在 1-100，单位为 mm/s
- **返回值：** 无

7、`get_joint_min_angle(joint_id)`

- **功能：** 获取指定关节最小能运行到的角度。
- **参数说明：** `joint_id`: (`int`) 您所指定的关节，范围为 1~6
- **返回值：** `angle`: 返回来的度数

8、`get_joint_max_angle(joint_id)`

- **功能：** 获取指定关节最大能运行到的角度。
- **参数说明：** `joint_id`: (`int`)您所指定的机械臂关节关节，范围为 1~6
- **返回值：** `angle`: (`float`)返回来的度数

9、`is_servo_enable(servo id)`

- **功能：** 判断指定关节是否连通。
- **参数说明：** servo id: 你所指定的关节，范围为 1~6
- **返回值：** (int) 1 - 连通 0 - 不连通 -1 - 错误

10、`is_all_servo_enable()`

- **功能：** 判断机器人的所有关节是否连通。
- **返回值：** (int) 1 - 连通 0 - 不连通 -1 - 错误

11、`release_servo(servo_id)`

- **功能：** 放松指定的关节
- **参数说明：** servo id: (`int`)机械臂的指定关节,范围为 1~6

- **返回值：** 无

12、`get_tof_distance()`

- **功能：** 获取检测到的距离（需要外部距离检测器）。
- **返回值：** 检测到的距离值，单位为 mm。

6.3 输入程序控制模式（MDI 模式）

1、`get_angles()`

- **功能：** 获取所有关节角度。
- **返回值：** `list` 一个浮点值的列表代，表所有关节的角度.

2、`send_angle(id, degree, speed)`

- **功能：** 发送指定的单个关节运动至指定的角度
- **参数说明：**
  - `id`: 代表机械臂的关节，六轴有六个关节，有特定的表示方法。关节一的表示法：`Angle.J1.value`。(也可以用数字 1-6 来表示）
  - `degree`: 表示关节的角度，取值范围：约-170 ～ 170
  - `speed`：表示机械臂运动的速度，范围 0~100。
- **返回值：** 无

3、`send_angles(degrees, speed)`

- **功能：** 发送所有角度给机械臂所有关节
- **参数说明：**
  - `degrees`: (List[float])包含所有关节的角度 ,六轴机器人有六个关节所以长度为 6，表示方法为：[20,20,20,20,20,20],取值范围：约-170 ～ 170
  - `speed`: 表示机械臂运动的速度，取值范围是 0-100。
- **返回值：** 无

4、`get_coords()`

- **功能：** 获取当前坐标和姿态。
- **返回值：** `list` 包含坐标和姿态的列表
  - 长度为 6，依次为 `[x, y, z, rx, ry, rz]`

5、 `send_coord(id,coord,speed)`

- **功能：** 发送单个坐标值给机械臂进行移动
- **参数说明：**

  - `id`:代表机械臂的坐标，六轴有六个坐标，有特定的表示方法。 X 坐标的表示法：`Coord.X.value`，也有简易的表示方法：如 X 轴可以填写 1,Y 填写 2,以此类推
  - `coord`: 输入您想要到达的坐标值，取值范围 -300 – 300
  - `speed`: 表示机械臂运动的速度，范围是 0-100。

- **返回值：** 无

6、 `send_coords(coords, speed, mode)`

- **功能：** 发送整体坐标和姿态,让机械臂头部从原来点移动到您指定点。
- **参数说明：**

  - `coords`: [x,y,z,rx,ry,rz]的坐标值，长度为 6，取值范围 -300 – 300
  - `speed`: 表示机械臂运动的速度，范围是 0-100。
  - `mode`: (`int`): 取值限定 0 和 1。

    - 0 表示机械臂头部移动的路径为非线性，即随机规划路线，只要机械臂头部以保持规定的姿态移动到指定点即可。
    - 1 表示机械臂头部移动的路径为线性的，即智能规划路线让机械臂头部以直线的方式移动到指定点。**（注意：有时机械臂不会移动，是该方法无法达到。可以尝试改成模式 0）**

- **返回值：** 无

7、 `get_encoders()`

- **功能：** 获取机械臂所有关节的电位值。
- **返回值：** `list` 包含机械臂所有关节电位值的列表。

8、 `get_encoder(joint_id)`

- **功能：** 获取机械臂指定的单个关节的电位值
- **参数说明：**

  - `joint_id`: 代表机械臂的关节，六轴机器人有六个关节所以长度为 6，有特定的表示方法，关节一的表示法：`Angle.J1.value`。(也可以用数字 1-6 来表示）

- **返回值：** 您所指定的单个关节电位值

9、 `set_encoder(joint_id, encoder)`

- **功能：** 发送指定的单个关节运动至指定的电位值。
- **参数说明：**
    - `joint_id`: 代表机械臂的关节，六轴机器人有六个关节所以长度为6，有特定的表示方法，关节一的表示法：`Angle.J1.value`。(也可以用数字 1-6 来表示）
    - `encoder`: 表示机械臂的电位值，取值范围是 0 ~ 4096
- **返回值：** 无

10、`set_encoders(encoders, sp)`

- **功能：** 发送电位值给机械臂所有关节
- **参数说明：**
    - `encoder`: 表示机械臂的电位值，取值范围是 0 ~ 4096，六轴长度为 6，表示方法为：[2048,2048,2048,2048,2048,2048]
    - `sp`: 表示机械臂运动的速度，取值范围是 0-100。
- **返回值：** 无

11、`get_radians()`

- **功能：** 获取所有关节的弧度。
- **返回值：** `list`包含所有关节弧度值的列表.

12、`send_radians(radians, speed)`

- **功能：** 发送弧度值给机械臂所有关节
- **参数说明：**
    - `radians`:表示机械臂的弧度值，取值范围是 -5~5。
- **返回值：** 无

13、`sync_send_angles(degrees, speed, timeout=7)`

- **功能：** 同步发送角度，到达目标点返回
- **参数说明：**
    - `degrees`: 每个关节的角度值列表(`List[float]`)。
    - `speed`: (`int`)机械臂运动的速度，取值范围是 0-100。
    - `timeout`: 时间默认 7s。
- **返回值：** 无

14、`sync_send_coords(coords, speed, mode)`

- **功能：** 同步发送坐标，到达目标点返回
- **参数说明：**
  - `coords`: 坐标值列表(`List[float]`)，长度为 6，依次为 [x, y, z, rx, ry, rz]
  - `speed`: (`int`)机械臂运动的速度，取值范围是 0-100。
- **返回值：** 无

15、`is_in_position(data, flag)`

- **功能：** 判断机器人有没有到达指定的位置。
- **参数说明：**
  - `data`: 您给出的一组数据可以是角度也可以是坐标值,如：[10,20,20,10,20,10]
  - `flag`: 数据类型(取值范围 0 或 1)
    - `0`: 表示给入的数值是角度值
    - `1`: 表示给入的数值是坐标值
- **返回值：** 1 - 已到达 0 - 未到达 -1 - 错误

16、 `is_moving()`

- **功能：** 判断机械臂有没有移动
- **返回值：** 1 - 正在移动 0 - 没有移动 -1 - 错误

17、 `set_color(r, g, b)`

- **功能：** 设置机器人手臂顶部的灯光颜色。（LED 灯光控制）
- **参数说明：** r,g,b 表示机器人顶部的灯光颜色值
  - `r`: 0 ~ 255
  - `g`: 0 ~ 255
  - `b`: 0 ~ 255
- **返回值：** 无

6.4 JOG 模式和操作

1、`jog_angle(joint_id, direction, speed)`

- **功能：** 控制机器人按照指定的角度持续移动
- **参数说明：**

    - `joint_id`: 代表机械臂的关节，按照关节 id 给入 1~6 来表示
    - `direction`: 主要控制机器臂移动的方向，给入 0 为负值移动，给入 1 为正值移动
    - `speed`: 速度 0 ~ 100

- **返回值：** 无

2、 `jog_coord(coord_id, direction, speed)`

- **功能：** 控制机器人按照指定的坐标或姿态值持续移动
- **参数说明：**

    - `coord_id`: 代表机械臂的关节，按照关节 id 给入 1~6 来表示
    - `direction`: 主要控制机器臂移动的方向，给入 0 为负值移动，给入 1 为正值移动
    - `speed`: 速度 0 ~ 100

- **返回值：** 无

3、 `jog_stop()`

- **功能：** 停止 jog 控制下的持续移动
- **返回值：** 无

6.5 舵机控制与操作

1、 `set_servo_data(servo_no, data_id, value)`

- **功能：** 设置舵机指定地址的数据参数
- **参数说明：**

    - `servo_no`: 舵机的序列号，按照关节 id 给入 1 - 6。
    - `data_id`: 数据地址。
    - `value`: 取值范围 0 - 4096

- **返回值：** 无

2、 `get_servo_data(servo_no, data_id)`

- **功能：** 读取舵机指定地址的数据参数。
- **参数说明：**

    - `servo_no`: 各个舵机的序列号，按照关节 id 给入 1 - 6。
    - `data_id`: 数据地址。

- **返回值：** 数据参数值，范围 0 - 4096

3、 `set_servo_calibration(servo_no)`

- **功能：** 校准指定关节，运行至位置为角度零点，对应电位值为 2048。
- **参数说明：** servo_no: (`int`) :机械臂的指定关节,范围为 1~6
- **返回值：** 无

4、 `focus_servo(servo_id)`

- **功能：** 给指定关节上电锁轴
- **参数说明：** servo id: (`int`) :机械臂的指定关节,范围为 1~6
- **返回值：** 无

6.6 夹爪控制

1、 `is_gripper_moving( )`

- **功能：** 判断夹爪是否正在运行。
- **返回值：**
  - `0`：表示机械臂的夹爪没有运行
  - `1`：表示机械臂的夹爪正在运行
  - `-1`：表示出错

2、 `set_gripper_value(value, speed)`

- **功能：** 让夹爪以指定的速度转动到指定的位置。
- **参数说明：**
  - `value`：表示夹爪所要到达的位置，取值范围 0~256。
  - `speed`：表示以多少的速度转动，取值范围 0~100。
- **返回值：** 无

3、 `get_gripper_value()`

- **功能：** 获取夹爪的 `encoder` 数据信息。
- **返回值：** 夹爪的数据信息。

4、 `set_gripper_ini()`

- **功能：** 设置夹爪初始化位置，设置当前位置为 2048
- **返回值：** 无

5、 `set_gripper_state(flag, speed)`

- **功能：** 让夹爪以指定的速度进入到指定的状态。

- **参数说明：**
  - `flag`：1 表示夹爪合拢状态，0 表示夹爪打开状态。
  - `speed`：表示以多快的速度达到指定的状态，取值范围 0~100。
- **返回值：** 无