Joint Control

For a serial multi-joint robot, the control of the joint space is to control the variables of each joint so as to make each joint reaches a target position at a certain speed.

> **Notice:** When setting the angle, the values corresponding to different manipulators are different. Refer to the parameter introduction section for more information.

myCobot

Single-Joint Control

send_angle(id, degree, speed)

- **Function:** to sends a specified single joint motion to a specified angle.
- Parameters:
  - `id`: to stand for the joints of a robot arm. Six axis means that the robot arm has six joints, and four-axis means it has four joints. There are specific representation methods therefor.The method to represent the joint 1:`Angle.J1.value`. (It can also be represented by numbers 1-6.)
  - `degree`: means the angle of a joint.
  - `speed`: means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** None

set_encoder(joint_id, encoder)

- **Function:** to sends a specified single joint motion to a specified potential value.
- **Parameters:**

  -
    ```
    joint_id
    ```

    : to stand for the joints of a robot arm. Six axis

    ```
    means that the robot arm has six joints, and four-axis means it
    has four joints. There are specific representation methods
    therefor. The method to represent the joint 1:` Angle.J1.value`.
    (It can also be represented by numbers 1-6.)
    ```

  - `encoder` :means the potential value of the robot arm, ranging from 0 - 4096.
- **Return value:** None

Multi-Joint Control

get_angles()

- **Function:** to get the angels of all joints.
- **Return Value:** `List`: a list of floating point values which represent the angles of all joints

send_angles(degrees, speed)

- **Function:** to send all angles to all joints.

- Parameters:
  - `degrees`: (List [float]) contains the angles of all joints. A six-axis robot has six joints, so the length is 6; and the four-axis length is 4. The representation method is [20, 20, 20, 20, 20, 20]; value range: about -170 - 170. Each joint of the four-axis robot is different. See the table above for details.
  - `speed`: means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** None

set_encoders(encoders, sp)

- **Function:** Send potential values to all joints of the robotic arm.
- Parameters:
  - `encoder`: means the potential of the robot arm, ranging from 0 - 4096. Six axis length is 6, and four axis length is 4. The way to represent:[2048, 2048, 2048, 2048, 2048, 2048].
  - `sp`: means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** None

sync_send_angles(degrees, speed, timeout=7)

- **Function:** to send an angle synchronously; return when reaching a target point.
- Parameters:
  - `degrees`: A list of angle values of each joint `List[float]`.
  - `speed`: (`int`) means the movement speed of the robot arm, ranging from 0 to 100.
  - `timeout`: The default time is 7s.
- **Return value:** None

get_radians()

- **Function:** to get the radian of all joints.
- **Return value:** `list`: a list containing radian values of all joints

send_radians(radians, speed)

- **Function:** to send radian values to all joints.
- Parameters:
  - `radians`: means the radian values of the robot arm, ranging from -5 to 5.
- **Return value:** `list`: a list containing radian values of all joints.

Simple Demo

```
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of mycobot, you can refer to these two variables to initialize MyCobot
import time

# MyCobot class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
```

```
#       M5 version is:  115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MyCobot("/dev/ttyUSB0", 115200)
#          or mc = MyCobot("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MyCobot("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot(PI_PORT, PI_BAUD)
#
# Initiate a MyCobot object
# Create object code here for windows version
mc = MyCobot("COM3", 115200)

#By passing the angle parameter, let each joint of the robotic arm move to the
position corresponding to [0, 0, 0, 0, 0, 0]
mc.send_angles([0, 0, 0, 0, 0, 0], 50)

# Set the waiting time to ensure that the robotic arm has reached the specified
position
time.sleep(2.5)

# Move joint 1 to the 90 position
mc.send_angle(Angle.J1.value, 90, 50)
# Set the waiting time to ensure that the robotic arm has reached the specified
position
time.sleep(2)

# The following code can make the robotic arm swing left and right
# set the number of loops
while num > 0:

     # Move joint 2 to the 50 position
    mc.send_angle(Angle.J2.value, 50, 50)

    # Set the waiting time to ensure that the robotic arm has reached the
specified position
    time.sleep(1.5)

    # Move joint 2 to the -50 position
    mc.send_angle(Angle.J2.value, -50, 50)

    # Set the waiting time to ensure that the robotic arm has reached the
specified position
    time.sleep(1.5)

    num -= 1

#Make the robotic arm retract. You can manually swing the robotic arm, and then
use the get_angles() function to get the coordinate sequence,
# use this function to let the robotic arm reach the position you want.
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)
```

```
# Set the waiting time to ensure that the robotic arm has reached the specified
position
time.sleep(2.5)

# Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

myBuddy

single joint control

send_angle(id, joint, angle, speed)

- **Function** Send one degree of joint to robot arm.
- **Parameters**
  - **id** – 1/2/3 (L/R/W)
  - **joint** – 1 ~ 6
  - **angle** – int
  - **speed** – 1 ~ 100
- **Returns**
  - None

get_angle(id, joint_id)

- **Function** Get the angle of a single joint
- **Parameters**
  - **id** (*int*) – 1/2/3 (L/R/W).
  - **joint_id** (*int*) – 1 - 7 (7 is gripper)

set_encoder(id, joint_id, encoder, speed)

- **Function** Set a single joint rotation to the specified potential value.
- **Parameters**
  - **id** – 1/2/3 (L/R/W).
  - **joint_id** – 1 - 6.
  - **encoder** – The value of the set encoder.
- **Returns**
  - None

multi-joint control

get_angles(id)

- **Function** Get the degree of all joints.
- **Parameters**

  id – 1/2 (L/R)
- **Returns**

  A float list of all degree.
- **Return type**

list

send_angles(id, degrees, speed)

- **Function** Send all angles to the robotic arm
- **Parameters**
  - **id** – 1/2 (L/R).
  - **degrees** – [angle_list] len 6
  - **speed** – 1 - 100

set_encoders(id, encoders, speed)

- **Function** Set the six joints of the manipulator to execute synchronously to the specified position.
- **Parameters**
  - **id** – 1/2 (L/R).
  - **encoders** – A encoder list, length 6.
  - **speed** – speed 1 ~ 100

get_radians(id)

- **Function** Get the radians of all joints
- **Parameters**

  id – 1/2 (L/R)
- **Returns**

  A list of float radians [radian1, ...]
- **Return type**

  list

send_radians(id, radians, speed)

- **Function** Send the radians of all joints to robot arm
- **Parameters**
  - **id** – 1/2 (L/R).
  - **radians** – a list of radian values(List[float]), length 6
  - **speed** – (int)1 ~ 100

Simple Demo

```
from pymycobot.mybuddy import MyBuddy
import time
mc = MyBuddy("/dev/ttyACM0", 115200)

# Send angles to the six joints of the left arm
mc.send_angles(1, [0, 0, 0, 0, 0, 0], 50)
time.sleep(3)

# Send the angle to the first joint of the left arm
mc.send_angle(1, 1, 90, 50)
time.sleep(2)
```

```
# Get the joint angle of the left arm
angles = mc.get_angles(1)
print("left angles: ",angles)

# Relax all joints of the left arm. Before running this command, please support the left arm with your hand to prevent it from falling suddenly
mc.release_all_servos(1)
```

myPalletizer

Simple Demo

```
from pymycobot.mypalletizer import MyPalletizer
from pymycobot.genre import Angle
import time
# import the project package

# Initiate a MyPalletizer object
mc = MyPalletizer("COM3",115200)
# By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, speed]
mc.send_angles([0, 0, 0, 0,], 50)
#  Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# Move joint 1 to the 50 position
mc.send_angle(1,20,50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)
# set variable "num"
num = 2
# set the number of loops
while num > 0:
    mc.send_angle(2,20,50)
    time.sleep(2)
    mc.send_angle(2,(-20),50)
    time.sleep(2)
    num -= 1

# make robot arms reach the specified position
mc.send_angles([-0.87, 41.66, -12.13, -0.17], 50)
#  Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

myArm

Simple Demo

```python
from pymycobot.myarm import MyArm
from pymycobot.genre import Angle
import time
# import the project package

# Initiate a MyArm object
mc = MyArm("/dev/ttyAMA0", 115200)
# By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, 0, 0, 0,speed]
mc.send_angles([0, 0, 0, 0, 0, 0, 0], 50)
#  Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# Move joint 1 to the 90 position
mc.send_angle(1,90,50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)
# set variable "num"
num = 2
# set the number of loops
while num > 0:
    mc.send_angle(2,20,50)
    time.sleep(2)
    mc.send_angle(2,(-20),50)
    time.sleep(2)
    num -= 1

# make robot arms reach the specified position
mc.send_angles([-5.25,-30,-20,-12,-10,-10,10],50)
#  Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```