3、Control bus servo

3.1 Experimental goals

This course mainly learns how to control the rotation of the robot's 18-degree-of-freedom servo.

3.2 Experiment preparation

The functions of the Muto hexapod robot Python library involved in this course are as follows:

motor(servo_id, angle, runtime=100)：Control the rotation of the robot's eighteen servos.

The parameter servo_id represents the ID number of the servo to be controlled. Each servo ID number is different. The value range of servo_id is [1, 18], which corresponds to the 18 bus servos on the robot.

The parameter angle represents the angle to which the servo is to be controlled. The value range of angle is [-90,90].

The parameter runtime is an optional parameter. The default value is 100, which indicates the time of the servo movement. Within the valid range, the smaller the value, the faster the rotation speed.

Servo_torque_on(servo_id=0): Turn on the servo torque

Servo_torque_off(servo_id=0): Turn off the servo torque.

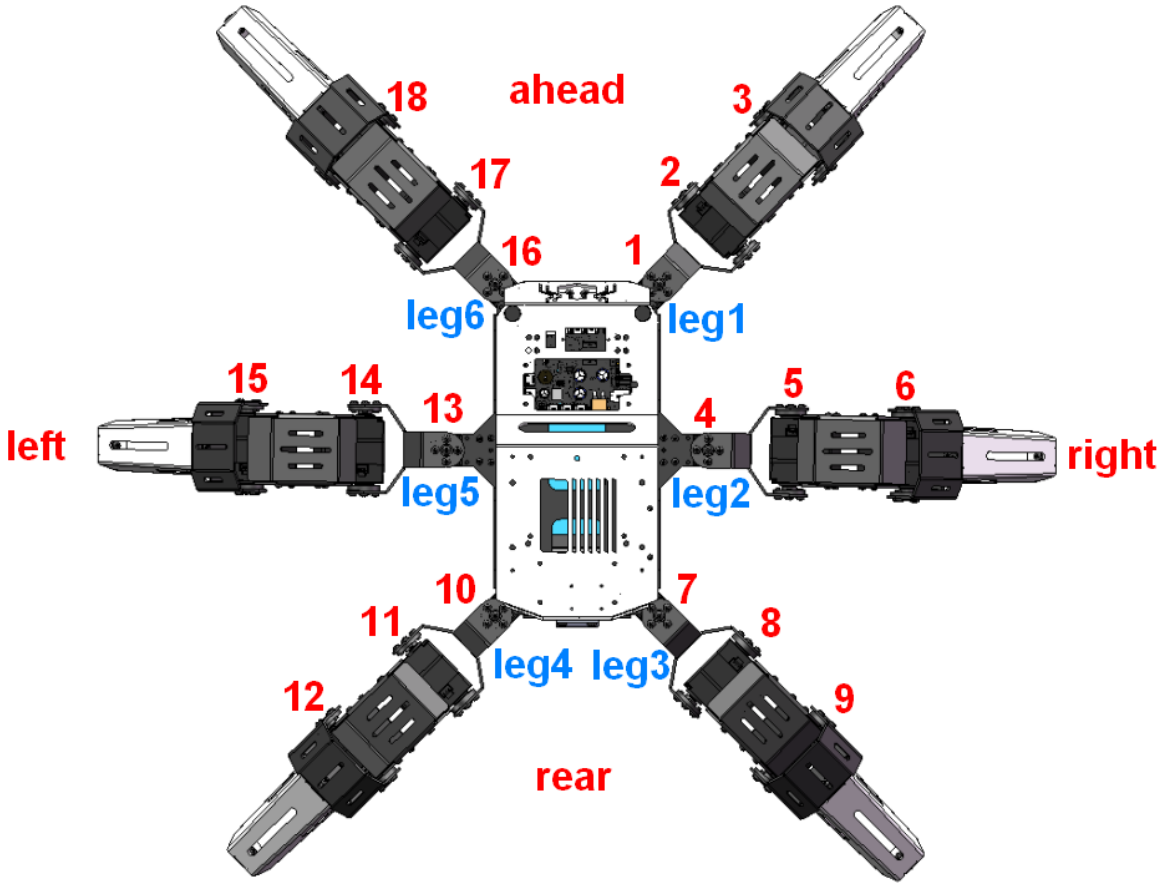
The optional parameter servo_id represents the ID number of the servo to be controlled, and the value range is [0, 18]. When servo_id=0, it means to control the torque of all servos on or off; when servo_id=[1, 18], it means Control the steering gear torque corresponding to the ID number to open or close.

load_leg(leg): Turn on the torque of the three servos on a certain leg

unload_leg(leg): Turn off the torque of three servos on a certain leg

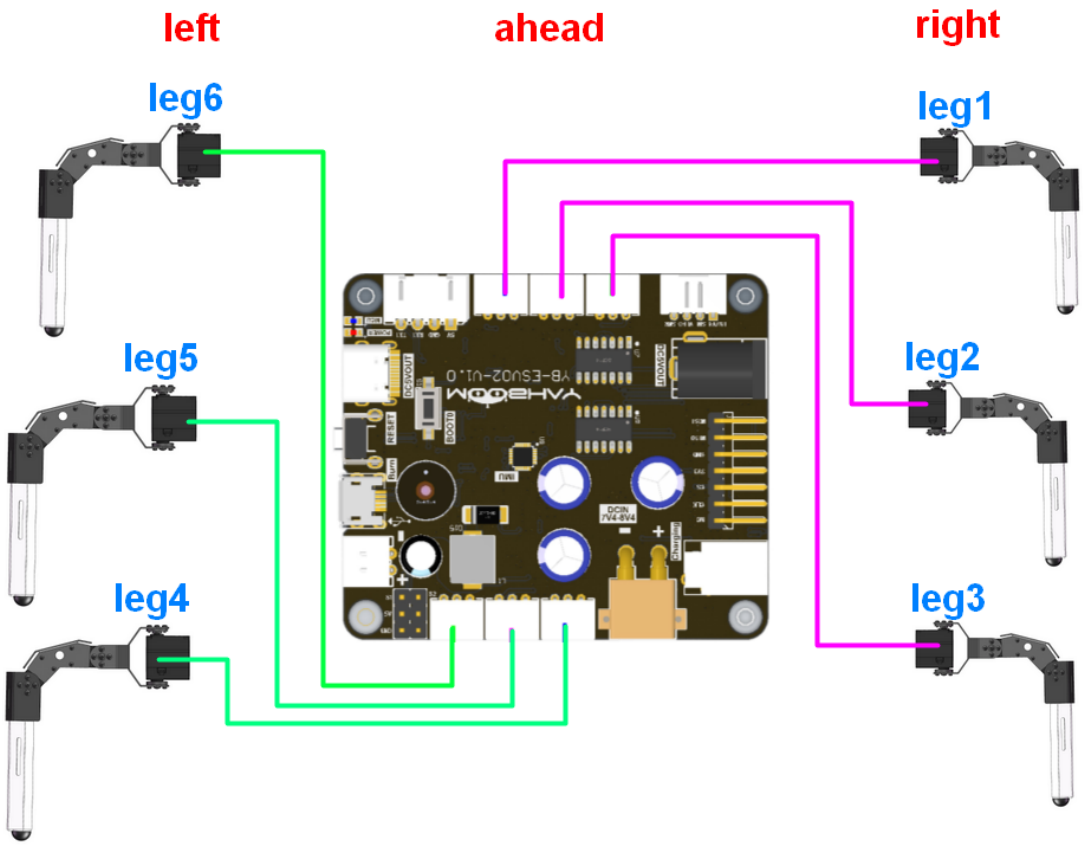
The parameter leg represents the ID number of the leg to be controlled. The value range of leg is [1, 6], which respectively represents the six legs of the robot.

The Muto robot has a total of six legs, and there are three bus servos on each leg, for a total of eighteen bus servos, which are assigned ID numbers 1-18 in sequence. The leg servo ID allocation is shown in the figure below:



The six legs of the Muto robot are divided into two groups， leg1, leg2, and leg3 belong to the same group, which is the right leg， leg4, leg5, and leg6 belong to the same group, which is the left leg. The three legs in the same group are wired in no particular order, but the left leg must be plugged into the corresponding interface of the left group and cannot be plugged into the interface of the right leg. The left leg group interfaces are the three interfaces close to the PWM servo interface, and the right leg group are the three interfaces close to the DC5V interface.

Leg connection diagram：



3.3 Experiment procedure

Open the jupyterLab client and find the code path:

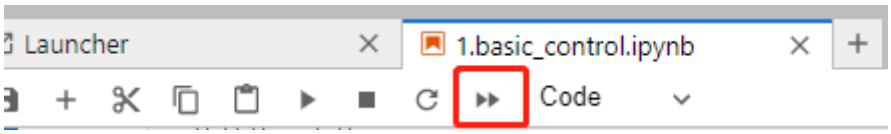
```
muto/Samples/Control/3.control_motor.ipynb
```

By default g_ENABLE_CHINESE=False, if you need to display Chinese, please set g_ENABLE_CHINESE=True.

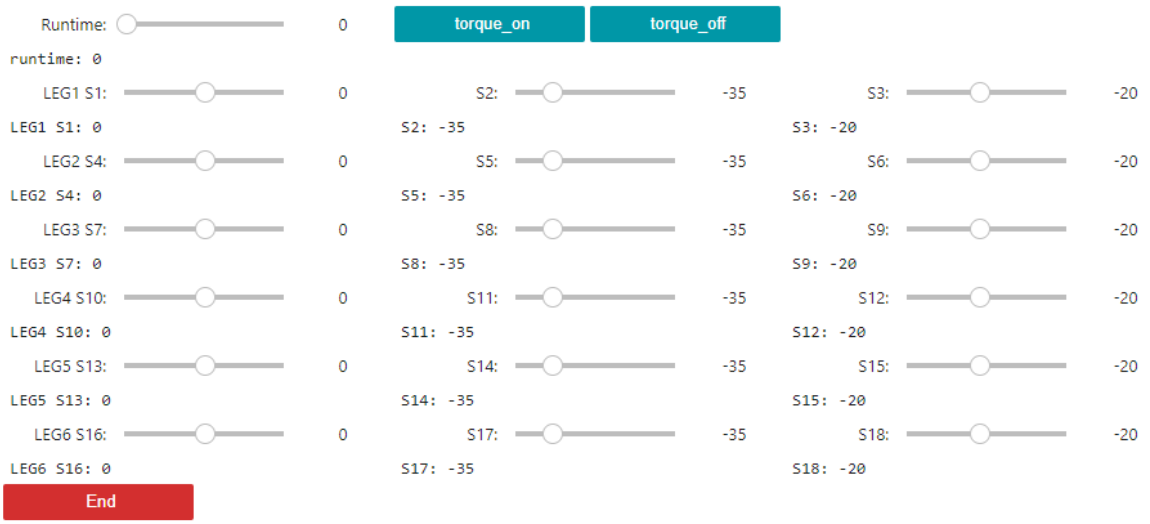
```
# 中文开关，默认为英文 Chinese switch. The default value is English
g_ENABLE_CHINESE = False

Name_widgets = {
    'End': ("End", "结束"),
    'torque_on': ("torque_on", "打开扭矩"),
    'torque_off': ("torque_off", "关闭扭矩")
}
```

Click to run all cells, and then scroll to the bottom to see the generated controls.



You can control different functions by operating different controls separately.



The Runtime slider mainly controls the movement time of the bus servo.

```
def on_slider_runtime(value):
    global g_bot, g_runtime
    g_runtime = value
    print("runtime:", value)
```

According to the ID number of the servo, the servo of each leg is controlled by a slider. Changing the value of the corresponding slider can control the servo of the corresponding ID.

```
def on_slider_S1(angle):
    print("LEG1 S1:", angle)
    g_bot.motor(1, angle, g_runtime)
def on_slider_S2(angle):
    print("S2:", angle)
    g_bot.motor(2, angle, g_runtime)
def on_slider_S3(angle):
    print("S3:", angle)
    g_bot.motor(3, angle, g_runtime)
```

The event processing of turning on and off the servo torque button is as follows.

```
# 按键按下事件处理   Key press event processing
def on_button_clicked(b):
    with output:
        print("Button clicked:", b.description)
    if b.description == Name_widgets['torque_on'][g_ENABLE_CHINESE]:
        g_bot.Servo_torque_on()
    elif b.description == Name_widgets['torque_off'][g_ENABLE_CHINESE]:
        g_bot.Servo_torque_off()
```

3.4 Experiment summary

This time, the JupyterLab control is used to control the bus servo of the hexapod robot. The bus servo has been limited in the program, and the servo movement can be controlled by sliding the slider corresponding to the ID.

When you need to turn off the torque of the servo, please click torque_off to turn off the torque. At this time, you cannot control the servo through the slide bar. You can turn the servo by hand. Click the torque_on button to restore the servo torque.

To exit the program, press the **End** button to exit the program.