3.Handle control

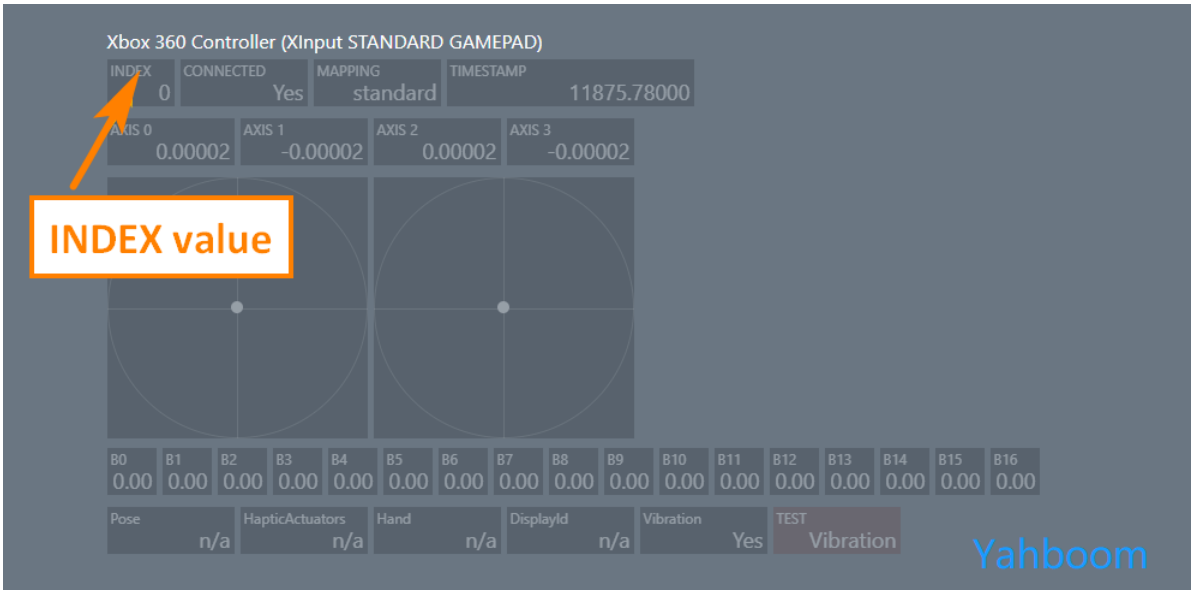
1.Handle key test

First of all, we open the http://html5gamepad.com webpage, and connect your Handle to your computer.

Because maybe your PC computer can not only connect a handle, so the default value of the index of the handle you connect is not 0, so we need to go to this page to view the handle we are currently using. The index can be used correctly.

After entering the webpage, the button on Handle must be pressed to trigger the detection and recognition.

The handle displays the corresponding handle information. The following is the detection interface of my handle. When we press the button of the handle, the corresponding button will also be pressed. We can view the mapped value of the currently pressed button and then call it into the corresponding function in our program.

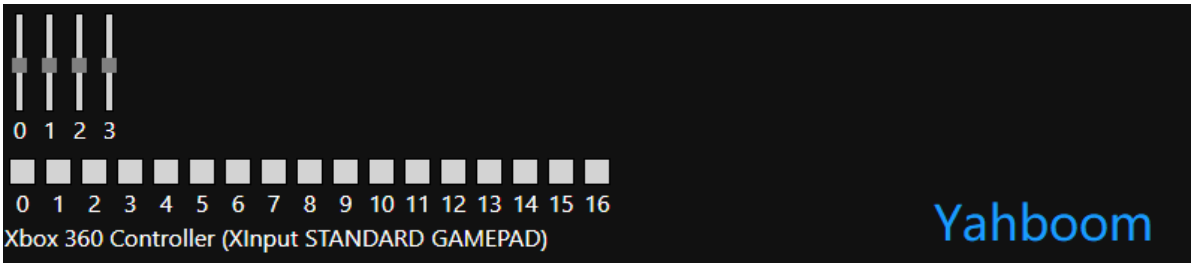


The handle controls the omnidirectional movement

This is useful if you have multiple controllers, or if some gamepads appear as multiple controllers. In order to properly use your handle to control, we need to set the index value tested by the handle test page mentioned above:

Set the index value in the following code:

After executing the above code, the handle key map shown in the figure below will be displayed:



2. Running code

Code Path: /home/pi/Dofbot/1.telecontrol/arm_handle.ipynb

The following code content needs to be executed according to the actual step, and cannot be run all at once.

```
# handle remote control
In this example we will remotely control a robotic arm using a gamepad controller
connected to a web browser machine.
```

```
import ipywidgets.widgets as widgets

controller = widgets.Controller(index=0)

display(controller)
```

```
#Function library path import
import   threading
import   time

#Thread function operation library
import   inspect
import   ctypes

#Import robotic arm objects
from   Arm_Lib import Arm_Device
Arm   = Arm_Device()
```

```
#Create a method to proactively stop a process
def   _async_raise(tid, exctype):
    """raises the exception,   performs cleanup if needed"""
```

```
    tid = ctypes.c_long(tid)
    if not inspect.isclass(exctype):
        exctype = type(exctype)
    res =   ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,
ctypes.py_object(exctype))
    if res == 0:
        raise ValueError("invalid thread   id")
    elif res != 1:
        # """if it returns a   number greater than one, you're in trouble,
        # and you should call it again with   exc=NULL to revert the effect"""
          ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)
def   stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```
# Create a method for controlling the movement of the robotic arm using a
joystick
## If the simulation mode of the handle is turned on, that is, when the red light
is on, the left direction keys cannot be used. Please use the left and right
joysticks and the keys on both sides of LR to control the robotic arm.
## Program functions:
1. The left joystick and direction keys control the No. 1 and No. 2 servo. The
left and right directions control the left and right movement of the No. 1 servo,
and the up and down directions control the forward and backward movement of the
No. 2 servo.
2. The right joystick and number keys control the No. 5 and No. 6 servos. The
left and right directions control the left and right rotation of the No. 5 servo,
and the up and down directions control the clamping and loosening of the No. 6
servo.
3. L1 and L2 control the No. 3 servo to move forward or backward.
4. R1 and R2 control the No. 4 servo to move forward or backward.
5. Press the SELECT button to set all servo angles of the robotic arm to 90
degrees.
```

```
def   Arm_Handle():
    s_time = 500
    s_step = 1
    angle_1 = angle_2 = angle_3 = angle_4   =  angle_5 = angle_6 = 90
    while 1:
        #Due to individual differences in joystick handles, all joystick reset
values are not necessarily zero, so 0.1 needs to be used as a filter to avoid
misoperation.。
        # Servo No. 2, A1 is negative up and down positive
        if controller.axes[1].value <= 0.1   and controller.axes[1].value >=
-0.1:
            time.sleep(.000001)
        else:
            if controller.axes[1].value >   0.1:
                angle_2 += s_step
            else:
                angle_2 -= s_step
            if angle_2 > 180:
                angle_2 = 180
            elif angle_2 < 0:
                angle_2 = 0
```

```
            Arm.Arm_serial_servo_write(2,   angle_2, s_time)
            time.sleep(0.01)
        # Servo No. 1, A0 left negative and right positive
        if (controller.axes[0].value <=   0.1 and controller.axes[0].value >=
-0.1):
                time.sleep(.000001)
        else:
            if controller.axes[0].value >   0.1:
                angle_1 -= s_step
            else:
                angle_1 += s_step
            if angle_1 > 180:
                angle_1 = 180
            elif angle_1 < 0:
                angle_1 = 0
            Arm.Arm_serial_servo_write(1,   angle_1, s_time)
            time.sleep(0.01)
        # Servo No. 6, NUM1=B0, NUM3=B2, A2 is up negative and down is positive
        if controller.buttons[0].value ==   True:
            angle_6 += s_step
            if angle_6 > 180:
                angle_6 = 180
            elif angle_6 < 0:
                angle_6 = 0
            Arm.Arm_serial_servo_write(6,   angle_6, s_time)
            time.sleep(0.01)
        elif controller.buttons[2].value ==   True:
            angle_6 -= s_step
            if angle_6 > 180:
                angle_6 = 180
            elif angle_6 < 0:
                angle_6 = 0
            Arm.Arm_serial_servo_write(6,   angle_6, s_time)
            time.sleep(0.01)
        elif controller.axes[2].value >   0.5:
            angle_6 -= s_step
            if angle_6 > 180:
                angle_6 = 180
            elif angle_6 < 0:
                angle_6 = 0
            Arm.Arm_serial_servo_write(6,   angle_6, s_time)
            time.sleep(0.01)
        elif controller.axes[2].value <   -0.5:
            angle_6 += s_step
            if angle_6 > 180:
                angle_6 = 180
            elif angle_6 < 0:
                angle_6 = 0
            Arm.Arm_serial_servo_write(6,   angle_6, s_time)
            time.sleep(0.01)
        # Servo No. 5, NUM2=B1, NUM4=B3, A5 is negative on the left and positive
on the right
        if controller.buttons[1].value ==   True:
            angle_5 += s_step
            if angle_5 > 180:
```

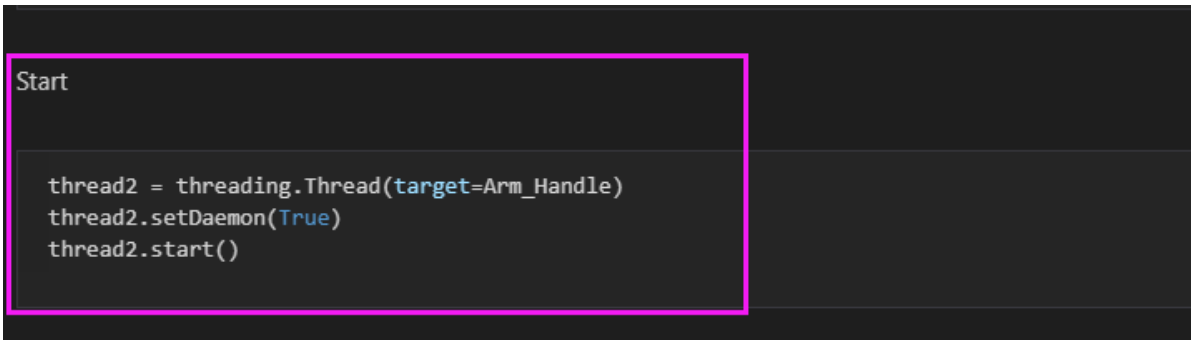
```
            angle_5 = 180
        elif angle_5 < 0:
            angle_5 = 0
        Arm.Arm_serial_servo_write(5,   angle_5, s_time)
        time.sleep(0.01)
    elif controller.buttons[3].value ==   True:
        angle_5 -= s_step
        if angle_5 > 180:
            angle_5 = 180
        elif angle_5 < 0:
            angle_5 = 0
        Arm.Arm_serial_servo_write(5,   angle_5, s_time)
        time.sleep(0.01)
    elif controller.axes[5].value >   0.5:
        angle_5 += s_step
        if angle_5 > 180:
            angle_5 = 180
        elif angle_5 < 0:
            angle_5 = 0
        Arm.Arm_serial_servo_write(5,   angle_5, s_time)
        time.sleep(0.01)
    elif controller.axes[5].value <   -0.5:
        angle_5 -= s_step
        if angle_5 > 180:
            angle_5 = 180
        elif angle_5 < 0:
            angle_5 = 0
        Arm.Arm_serial_servo_write(5,   angle_5, s_time)
        time.sleep(0.01)
    # Servo No. 4, R1=B5,R2=B7
    if controller.buttons[5].value ==   True:
        angle_4 -= s_step
        if angle_4 > 180:
            angle_4 = 180
        elif angle_4 < 0:
            angle_4 = 0
        Arm.Arm_serial_servo_write(4,   angle_4, s_time)
        time.sleep(0.01)
    elif controller.buttons[7].value ==   True:
        angle_4 += s_step
        if angle_4 > 180:
            angle_4 = 180
        elif angle_4 < 0:
            angle_4 = 0
        Arm.Arm_serial_servo_write(4,   angle_4, s_time)
        time.sleep(0.01)
    # Servo No. 3, L1=B4,L2=B6
    if controller.buttons[4].value == True:
        angle_3 -= s_step
        if angle_3 > 180:
            angle_3 = 180
        elif angle_3 < 0:
            angle_3 = 0
        Arm.Arm_serial_servo_write(3,   angle_3, s_time)
        time.sleep(0.01)
```

```
        elif controller.buttons[6].value ==   True:
            angle_3 += s_step
            if angle_3 > 180:
                angle_3 = 180
            elif angle_3 < 0:
                angle_3 = 0
            Arm.Arm_serial_servo_write(3,   angle_3, s_time)
            time.sleep(0.01)
        # Press the selection button B8 to set the servos of the robotic arm to
90 degrees
        if controller.buttons[8].value ==   True:
            angle_1 = angle_2 = angle_3 =   angle_4 = angle_5 = angle_6 = 90
            Arm.Arm_serial_servo_write6(90,   90, 90, 90, 90, 90, 1000)
            time.sleep(1)
```

```
#Enable the thread of the handle to control the robotic arm in real time by
running the following cell code
#After waiting for the handle control thread to start, you can control the
robotic arm through the handle.
thread2   = threading.Thread(target=Arm_Handle)
thread2.setDaemon(True)
thread2.start()
```

```
#End the handle thread. If this unit is run, the handle cannot control the
robotic arm.
#If the thread fails to start or end,
#Please restart the kernel and run it step by step again.
stop_thread(thread2)
```

The code only needs to run to the position shown in the figure below.



4.The remote control function of the handle is shown below.


key1 and key3
control No.6 servo
key2 and key4
control No.5 servo
Rocker up and down
control No.6 servo
Rocker left and right
control No.5 servo
Rocker up and down
control No.2 servo
Rocker left and right
control No.1 servo
Up and down key
control No.2 servo
Left and right key
control No.2 servo
R1 and R2 control
No.4 servo
L1 and L2 control
No.4 servo
SELCET key can make the
robotic arm to the centered
state