2.1.4 PS2 Handle test

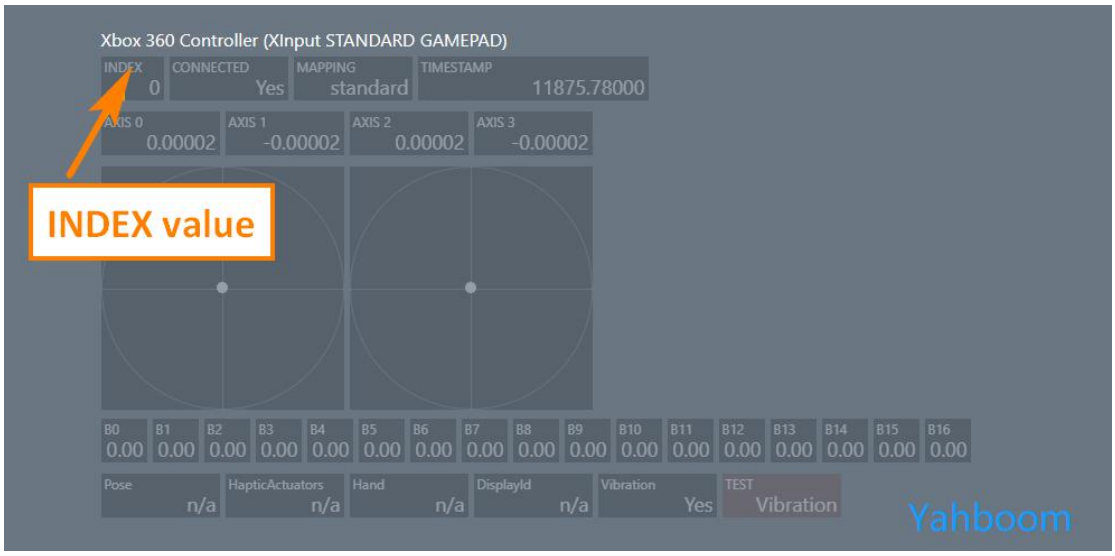
1. Handle key test

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.



2. The handle controls the omnidirectional movement

In this example, we will remotely control Jetbot using a gamepad controller connected to the web browser machine.

First, we need to create an instance of the 'Controller' widget, which we will use to drive our Raspblock. The "Controller" widget accepts an "index" parameter that specifies the number of controllers.

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 slide bars and boxes will be displayed. As shown below.

It is recommended that users use the simulation mode, we can press the ANALOG button on handle to enter the simulation, the red light will on. As shown below.

In the above figure, the slide bars 0-9 represent the analog value, the method of taking value: controller.axes [0] .value = -1.0 ~ 1.0
the boxes 0-11 represent the Bool value of the keys, the method of taking valuecontroller.buttons [0] .value = True/False

Control the Raspblock buzzer through the handle, when wepress the number 1 button of handle, buzzer will make sound, release the buzzer will off.

Code path:
/home/pi/Yahboom_Project/2Hardware_Control_cuorse/4Handle_test.ipynb

```
from Raspblock import Raspblock
import ipywidgets.widgets as widgets
import time

# Thread function operation library
import threading
import inspect
```

```
import ctypes
import matplotlib.pyplot as plt


robot = Raspblock()
controller = widgets.Controller(index=0)    # index indicates the serial number of the handle we use
display(controller)                         # Display slide bars and boxes
```

```
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)
```

```
def Remote_thread():
    while True:
        if controller.buttons[0].value == True:
            robot.Buzzer_control(1)
        else:
            robot.Buzzer_control(0)
        time.sleep(0.01)
def stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```
thread = threading.Thread(target=Remote_thread)
thread.setDaemon(True)
thread.start()
```

```
# When we need to end the entire program process, we need to run the code
stop_thread(thread)
del robot
```