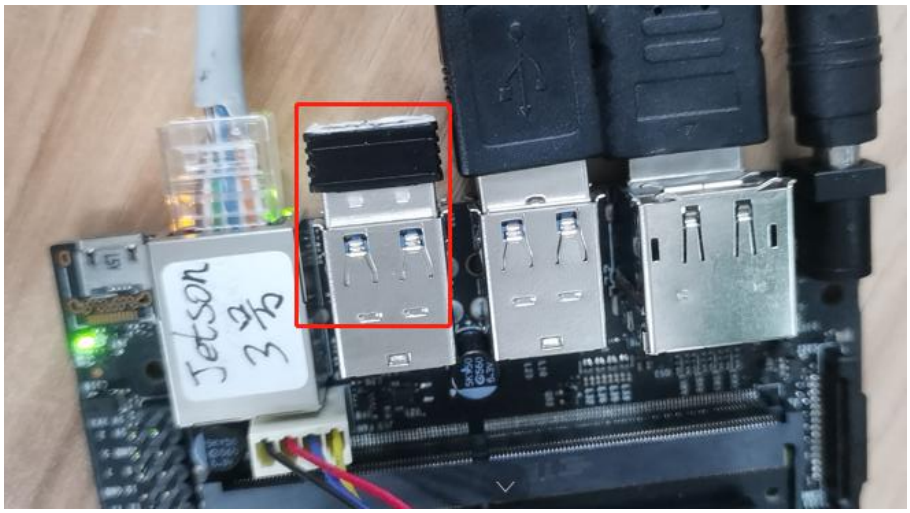
基础编程例程

一、安装手柄

将无线手柄的 USB 接口插入到主机（Jetson Nano 或树莓派）的 USB 口。（如果是使用远程 jupyter lab 登录的则需要插到远程登录的电脑的 USB 口上。）

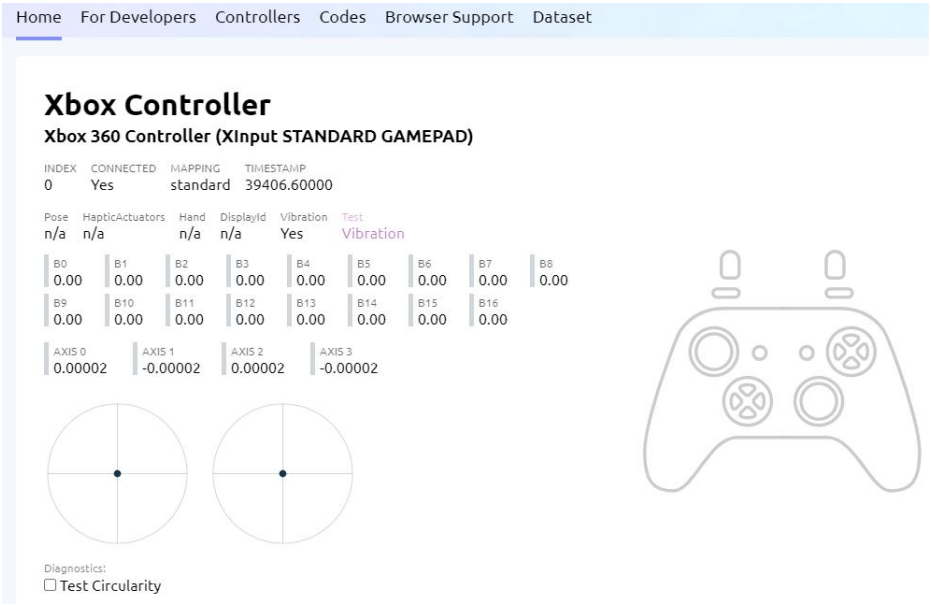


打开无线手柄的电池仓，并安装好电池，盖上盖子，然后给打开无线手柄开关（拨到 ON）。

最后按一下 START 键连接无线手柄。由于无线手柄具有省电功能，所以一段时间未操作后，自动进入休眠，需要按 START 键来激活连接。

二、查看手柄设备号

打开 http://html5gamepad.com 网址，然后操作手柄查看设备号，正常情况下只连接一个手柄设备号默认为 0。操作手柄按键可以看到有数据变化。



三、运行手柄控制程序

以下代码内容需要根据实际每一步执行，不能一次性运行全部。

手柄遥控

在本例中，我们将使用连接到 web 浏览器机器的 gamepad 控制器来打印手柄的遥控数据。

创建手柄控制器

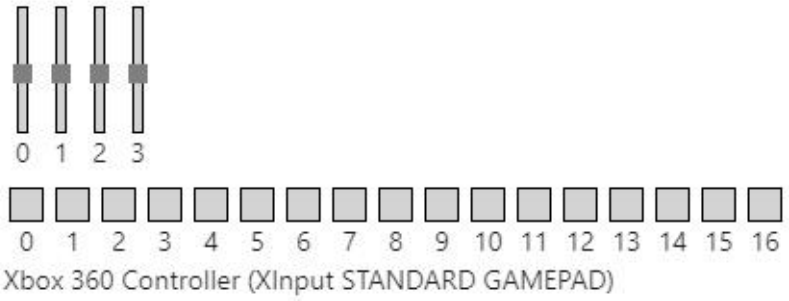
我们要做的第一件事是创建一个’Controller’widget 的实例,我们将使用它来驱动我们的 USB 手柄。

“Controller”小部件接受一个“index”参数，该参数指定控制器的数量。如果你有多个控制器，或者一些游戏手柄以多个控制器的形式出现，这是非常有用的。如果只接入一个手柄，默认 index=0，如果有多个则需要根据实际输入对应手柄的编号:

1. 打开http://html5gamepad.com此网页.
2. 按下你正在使用的手柄的按键
3. 记住当你按下按键后弹出的相应的索引号

接下来，我们将使用该索引创建并显示控制器。

```
[1]: import ipywidgets.widgets as widgets
     controller = widgets.Controller(index=0)  #用你刚测试过正在使用的控制器的索引号替代
     display(controller)
```




```
import ipywidgets.widgets as widgets
controller = widgets.Controller(index=0)  #用你刚测试过正在使用的控制器的索引号替代
display(controller)
```

```
#函数库路径导入
import threading
import time
# 线程功能操作库
import inspect
import ctypes
```

```
#创建主动停止进程的方法
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)
```

创建手柄遥杆接收和打印数据的方法
如果将手柄的模拟模式打开，（按下 ANALOG 键）即红灯亮起时，左边方向键不起作用，并且两个摇杆输出模拟值，左边摇杆是 0 号和 1 号滑动条，右边是 2 号和 3 号滑动条。

程序功能:
1. 打印每一次按下的是哪个键。

```
def USB_Handle():
    delay_time = 0.01
    while 1:
        #因为摇杆手柄个别差异,所有在遥杆复位值不一定是零,所以需要以 0.1 作为过滤,避免误操作。
        # 打印左边摇杆和方向键的数据
        # A1 上负下正
        if controller.axes[1].value > 0.1: # 左边摇杆/方向键 向下
            print('L_DOWN=%.2f'%(controller.axes[1].value))
        elif controller.axes[1].value < -0.1: # 左边摇杆/方向键 向上
            print('L_UP=%.2f'%(controller.axes[1].value))
        time.sleep(delay_time)

        # A0 左负右正
        if controller.axes[0].value > 0.1: # 左边摇杆/方向键 向右
            print('L_RIGHT=%.2f'%(controller.axes[0].value))
            time.sleep(delay_time)
        elif controller.axes[0].value < -0.1: # 左边摇杆/方向键 向左
            print('L_LEFT=%.2f'%(controller.axes[0].value))
            time.sleep(delay_time)

        # 左边摇杆按下=B10
        if controller.buttons[10].value == True:
            print('L_PRESS')
```

```
            time.sleep(delay_time)


        # 打印右边摇杆的模拟值
        # A2 上负下正, A5 左负右正
        if controller.axes[2].value > 0.1:
            print('R_UP=%.2f'%(controller.axes[2].value))
            time.sleep(delay_time)
        elif controller.axes[2].value < -0.1:
            print('R_DOWN=%.2f'%(controller.axes[2].value))
            time.sleep(delay_time)

        if controller.axes[5].value > 0.1:
            print('R_RIGHT=%.2f'%(controller.axes[5].value))
            time.sleep(delay_time)
        elif controller.axes[5].value < -0.1:
            print('R_LEFT=%.2f'%(controller.axes[5].value))
            time.sleep(delay_time)

        # 右边摇杆按下=B11
        if controller.buttons[11].value == True:
            print('R_PRESS')
            time.sleep(delay_time)


        # NUM1=B0,NUM2=B1,NUM3=B2,NUM4=B3
        if controller.buttons[0].value == True:
            print('NUM1')
            time.sleep(delay_time)
        if controller.buttons[1].value == True:
            print('NUM2')
            time.sleep(delay_time)
        if controller.buttons[2].value == True:
            print('NUM3')
            time.sleep(delay_time)
        if controller.buttons[3].value == True:
            print('NUM4')
            time.sleep(delay_time)


        # R1=B5,R2=B7
        if controller.buttons[5].value == True:
            print('R1')
            time.sleep(delay_time)
```

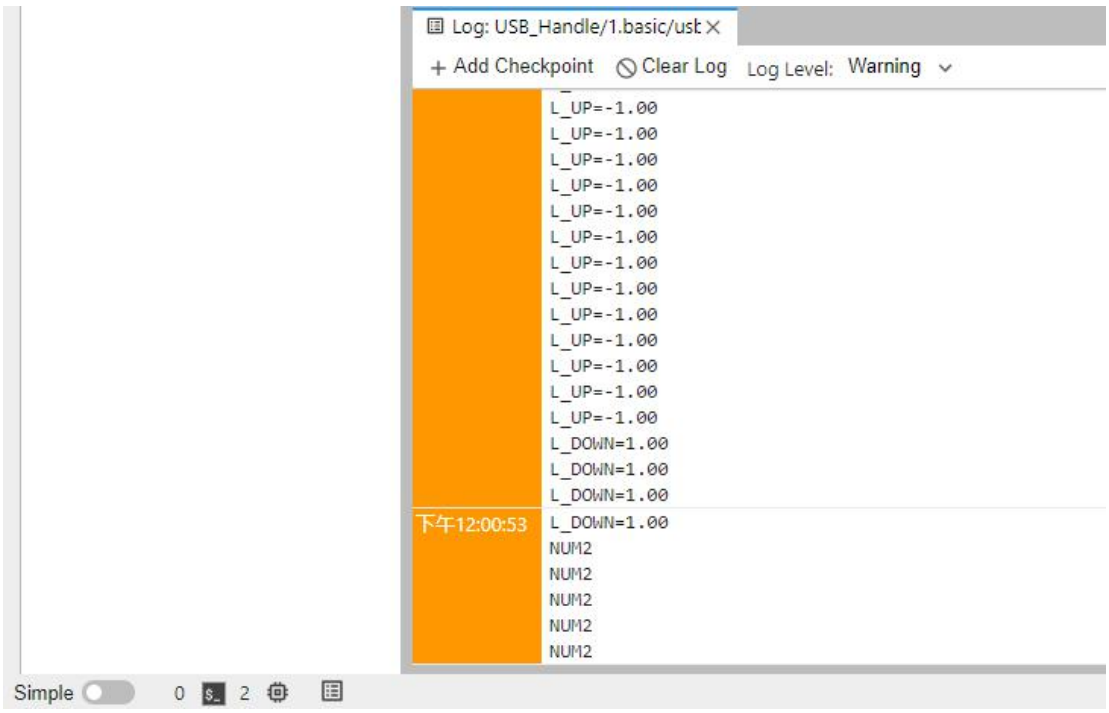
```
        if controller.buttons[7].value == True:
            print('R2')
            time.sleep(delay_time)

        # L1=B4,L2=B6
        if controller.buttons[4].value == True:
            print('L1')
            time.sleep(delay_time)
        elif controller.buttons[6].value == True:
            print('L2')
            time.sleep(delay_time)

        # SELECT=B8
        if controller.buttons[8].value == True:
            print('SELECT')
            time.sleep(delay_time)
        # START=B9
        if controller.buttons[9].value == True:
            print('START')
            time.sleep(delay_time)
```

```
#通过运行下面单元格代码开启手柄的线程
#等待手柄控制线程启动后就可以操作手柄。
thread = threading.Thread(target=Arm_Handle)
thread.setDaemon(True)
thread.start()
```

```
#结束手柄线程。
#如果出现线程启动或者结束失败的情况，
#请重新 start 一下 kernel，再重头一步步运行。
stop_thread(thread)
```

三、操作手柄

点击左下角的调试信息窗口，会弹出调试信息的窗口。按下不同的键会打印对应的数据。



同时在上面的插件会以图形变化的方式显示手柄的操作。

```
import ipywidgets.widgets as widgets
controller = widgets.Controller(index=0)  #用你刚测试过正在使用的控制器的索引号替代
display(controller)
```