Wireless handle control

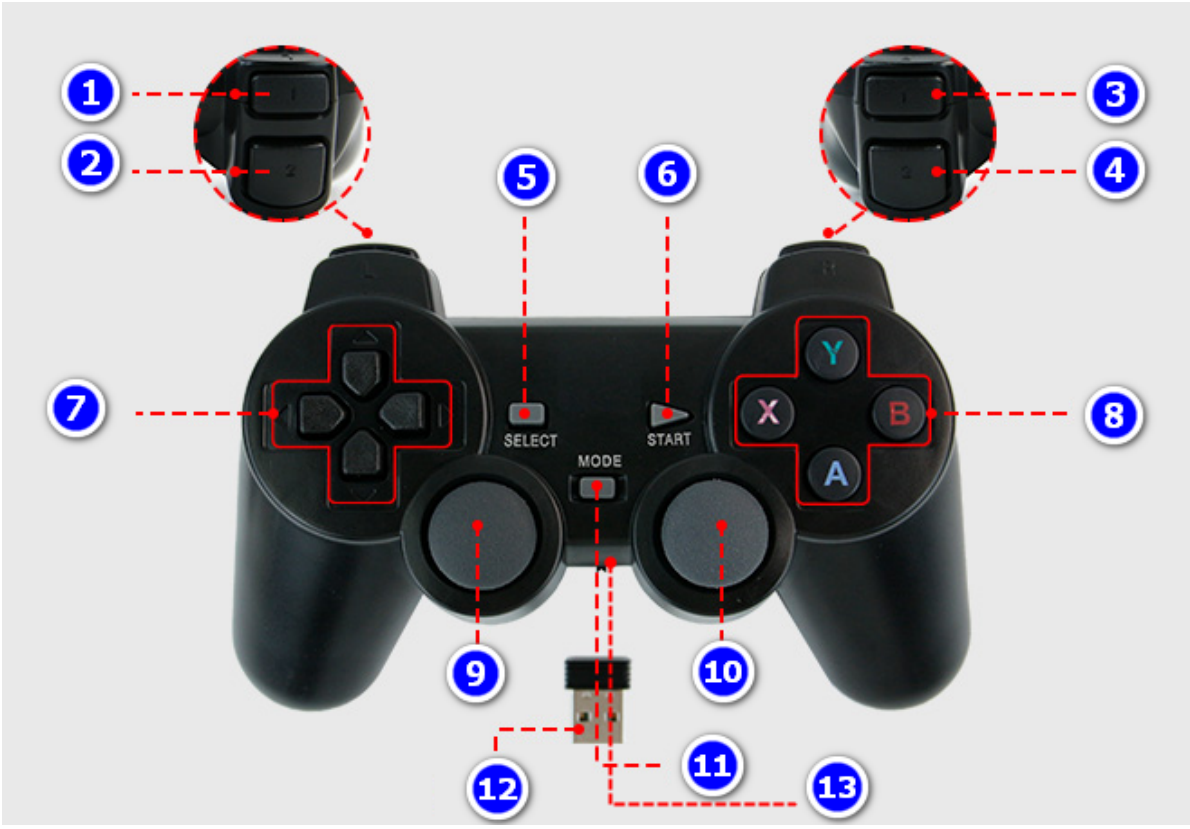
1.1 Experimental purpose

In this course, we will learn how to use the wireless handle to control dogzilla motion.

1.2 Experimental preparation

Please insert the USB receiver of the wireless handle into the USB port of the raspberry pie, then install the battery on the wireless handle and turn it on. Press start on the handle to wake up the handle.

Wireless handle key position operation is as follows:



1.3 Experimental preparation

Open the jupyterlab client and find the code path:

```
DOGZILLA/Samples/2_Control/9.dog_Ctrl.ipynb
```

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

(1) L1: perform three-axis linkage action

(2) L2: perform left-right swing action

(3) R1: perform kicking action

(4) R2: perform urination

(5) select: enable obstacle crossing mode

(6) start: wake up the USB wireless handle and restore the initial state

(7) direction key: control dogzilla forward and backward, left and right translation

(8) function keys: X controls the shoulder to the left, B controls the shoulder to the right, y controls the height to increase, and a controls the height to decrease

(9) left rocker: the rocker controls the speed of dogzilla's forward and backward translation and left and right translation. Press the rocker to control the step width

(10) right rocker: left and right control left and right rotation, up and down control head up and down, press the rocker to control the pace frequency

(11) mode: connect some computer systems to pop up the game menu, which has no practical use(12) USB wireless receiving terminal: connect the USB port of Raspberry pi.

(13)USB wireless handle power switch: turn on to turn on, and turn off to turn off

1.3 Experimental process

Open the terminal and run the following command to open the handle control code.

```
python3 ~/DOGZILLA/app_dogzilla/joystick_dogzilla.py
```

By default, the program does not open debug information. If you need to print debug information, run the following command.

```
python3 ~/DOGZILLA/app_dogzilla/joystick_dogzilla.py debug
```

The key codes are as follows.

The following content is to determine whether the handle receiver is inserted and open the handle receiver using Python's built-in open function. In general, when only one handle receiver is inserted, the device number is / dev / input / js0.

```
# Find the joystick device.
print('Joystick Available devices:')
# Shows the joystick list of the Controler, for example: /dev/input/js0
for fn in os.listdir('/dev/input'):
    if fn.startswith('js'):
        print('    /dev/input/%s' % (fn))

# Open the joystick device.
try:
    js = '/dev/input/js' + str(self.__js_id)
    self.__jsdev = open(js, 'rb')
    self.__js_isOpen = True
    print('---Opening %s Succeeded---' % js)
except:
    self.__js_isOpen = False
    print('---Failed To Open %s---' % js)
```

Next, read the data of the wireless handle receiver, read 8 data each time, and then unpack the data according to the protocol to obtain time, value, type and number.

Time: timestamp.

Value: numeric value.

Type: type. 1 is a button event and 2 is an axis event.

Number: the number of the corresponding button or axis. In combination with type, an event can be accurately located.

```
        evbuf = self.__jsdev.read(8)
        if evbuf:
            time, value, type, number = struct.unpack('IhBB', evbuf)
            func = type << 8 | number
            name = self.__function_names.get(func)
            # print("evbuf:", time, value, type, number)
            # if self.__debug:
            #     print("func:0x%04X, %s, %d" % (func, name, value))
            if name != None:
                self.__data_processing(name, value)
            else:
                if self.__ignore_count > 0:
                    self.__ignore_count = self.__ignore_count - 1
                if self.__debug and self.__ignore_count == 0:
                    print("Key Value Invalid")
        return self.STATE_OK
```

Create a new dictionary to save events of keys and axes.

```
# Defining Functional List
self.__function_names = {
    # BUTTON FUNCTION
    0x0100 : 'A',
    0x0101 : 'B',
    0x0102 : 'X',
    0x0103 : 'Y',
    0x0104 : 'L1',
    0x0105 : 'R1',
    0x0106 : 'SELECT',
    0x0107 : 'START',
    0x0108 : 'MODE',
    0x0109 : 'BTN_RK1',
    0x010A : 'BTN_RK2',

    # AXIS FUNCTION
    0x0200 : 'RK1_LEFT_RIGHT',
    0x0201 : 'RK1_UP_DOWN',
    0x0202 : 'L2',
    0x0203 : 'RK2_LEFT_RIGHT',
    0x0204 : 'RK2_UP_DOWN',
    0x0205 : 'R2',
    0x0206 : 'WSAD_LEFT_RIGHT',
    0x0207 : 'WSAD_UP_DOWN',
}
```

The event is judged and executed according to the dictionary value.

```
# Control robot
def __data_processing(self, name, value):
    if name=="RK1_LEFT_RIGHT":
        value = -value / 32767
        if self.__debug:
            print ("%s : %.3f, %d" % (name, value, self.__pace_width))
        fvalue = int(self.__pace_width * self.__WIDTH_SCALE_Y * value)
        self.__dog.move('y', fvalue)
    elif name == 'RK1_UP_DOWN':
        value = -value / 32767
        if self.__debug:
            print ("%s : %.3f, %d" % (name, value, self.__pace_width))
        fvalue = int(self.__pace_width * self.__WIDTH_SCALE_X * value)
        self.__dog.move('x', fvalue)
```

Finally, call to close the JS port of the wireless handle at the end.

```
def __del__(self):
    if self.__js_isOpen:
        self.__jsdev.close()
    if self.__debug:
        print("\n---Joystick DEL---\n")
```

1.4 Summary

In this course, we will control robot dog by handle.

It is necessary to insert the handle receiver into the USB port of the Raspberry Pi in advance, then turn on the power of the wireless handle, press the start button to wake up and connect the receiver. If the handle is not operated for a period of time, the handle will enter the sleep state, please press the start button again to wake up the handle.