UNIT 5　faya JoyStick Switch

5-1 UNIT OBJECTIVES：

When you have completed this unit, you will be able to explain the operating principle of Joy Stick Switch and use Raspberry Pi to demonstrate the control of faya JoyStick Switch.

5-2 OBJECTIVE TIME：

30 ~ 40 minutes

5-3 EQUIPMENT REQUIRED：

MTS-200 Tutor for Raspberry Pi
faya JoyStick Switch
USB Keyboard
USB Mouse (Optional)
Computer Screen (Optional)
USB Cable
Type-B Power Wires
Signal Wires
AC Power Cord

5-4 MODULE DESCRIPTION：

Figure5-1 faya JoyStick Switch

Name	faya JoyStick Switch
Main Parts	Power Jack x 2 Joystick Switch x 1 1x3-pin Stackable Header x 1
Signal Ports	VRX : analog voltage from horizontal potentiometer VRY: analog voltage from vertical potentiometer SW : Output HIGH when switch is pressed Output LOW when switch is depressed
Size	6 x 5 brick unit

A joystick is a hand switch actuated by a lever free to move in more than one axis of motion. One or more of several switch contact mechanisms are actuated depending on which way the lever is pushed, and sometimes by how far it is pushed. Joystick switches are often used to control video games, and also used for controlling machines such as wheelchairs, cranes, and robots. Joysticks usually have at least one pushbutton and potentiometers whose state can also be read by the microcontroller.

faya JoyStick Switch, as shown in Figure 5-1, uses two analog output port VRX and VRY to represents wiper voltage from horizontal potentiometer (H pot) and from vertical potentiometer (V pot). It also uses one digital port SW to output the state of the switch. When the button is pressed, it outputs HIGH, otherwise; it outputs LOW.

5-5 BACKGROUND KNOWLEDGE：

Figure 5-2 shows the schematic of faya JoyStick Switch. The voltage on the VRX pin increases as the wiper of H pot moves from middle to right, while the voltage decreases as the wiper moves from middle to left. The voltage on the VRY pin increases as the wiper of V pot moves from middle to top, while the voltage decreases as the wiper moves from middle to bottom. The tact switch (center pushbutton) is connected through a pull-up resistor to +5V. When the tact switch is opened (depressed), the voltage on the SW port is HIGH. When the tact switch is closed (pressed), the voltage on the SW port will be LOW. In normal case, the joystick's stick is in the center position due to an internal spring mechanism, so the tact switch is open (output LOW) and the wipers of V and H pots are at the middle position (output VCC/2 volt).

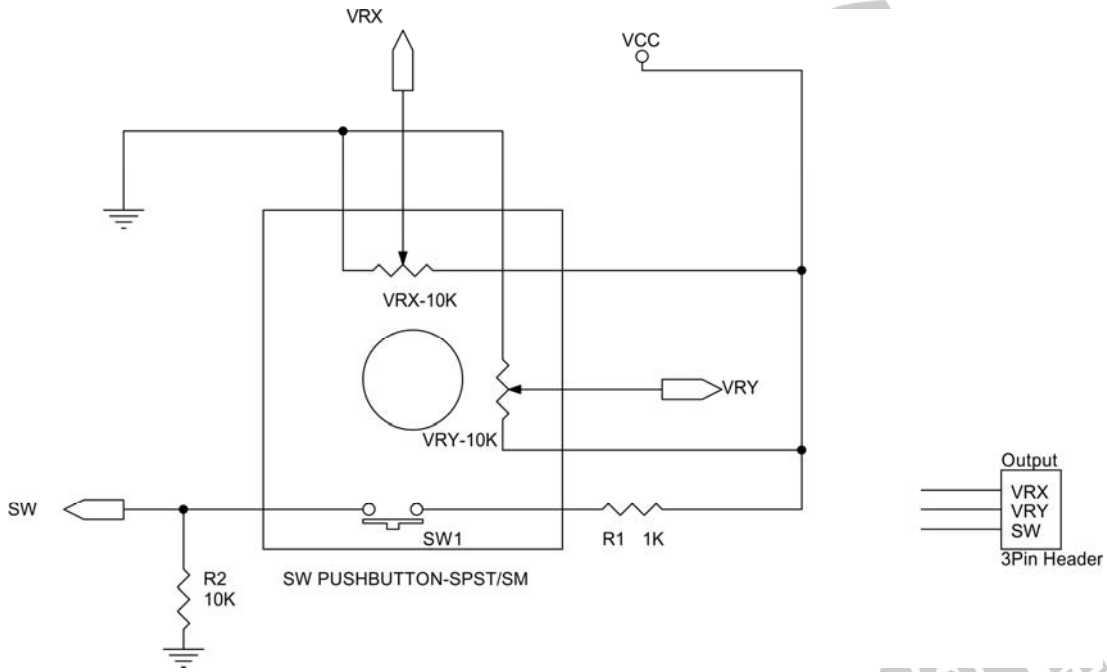

Figure5-2 schematic of faya Joystick Switch

5-6 UNIT EXERCISE：

In this exercise, we are going to demonstrate how to use Raspberry Pi to control faya JoyStick Switch.

EXERCISE OBJECTIVES

(1) Display analog value of VRX and VRY on console

PROCEDURE

(1) Connection for power wires:

Use type-B power wire to link the power sockets between MTS-200 and faya JoyStick Switch.

(2) Connection for signal wires:

	From		To	
1	MTS-200	DAC/ADC (1) AI0	faya Joystick Switch	VRX
2	MTS-200	DAC/ADC (1) AI1	faya Joystick Switch	VRY
3	Raspberry Pi	Pin 11 (Label IO17)	faya Joystick Switch	SW

(3) Click on Raspberry Pi Start Button, and select [Programming] / [Python 2 (IDLE)] to open Python Shell, as shown in Figure 5-3.

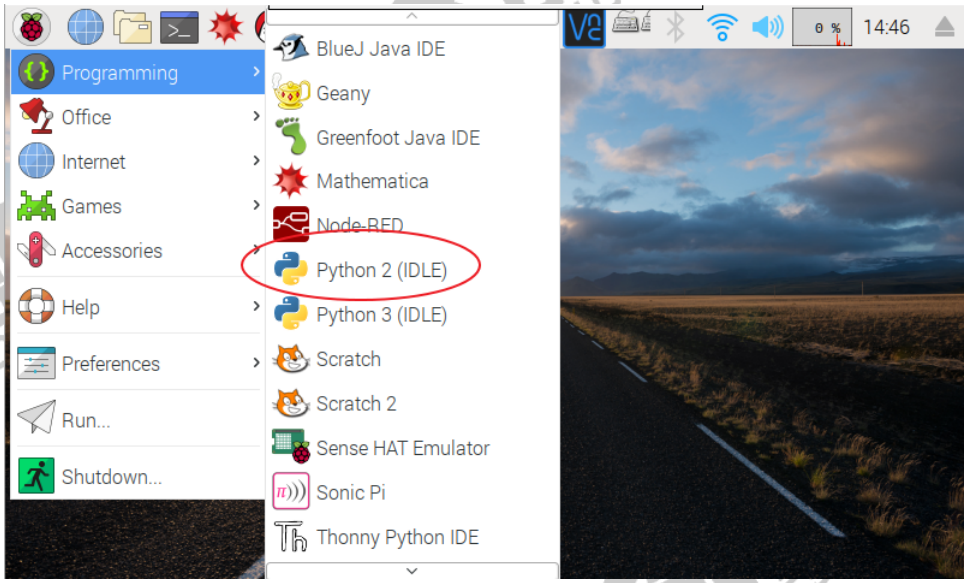

Figure5-3 Launch Python Shell

(4) After launching Python Shell, select [File] / [New File] from the menu bar, as shown in Figure 5-4.

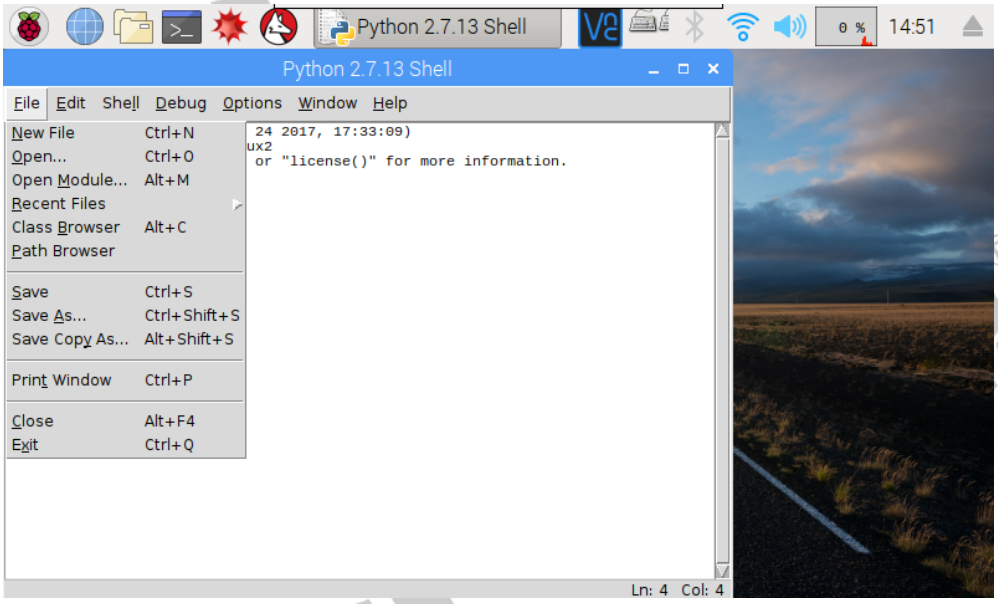

Figure 5-4 Create a new python file

(5) Carefully key-in following source code line by line in the Python Shell. Alternatively, you may load this code from the menu bar [File] / [Open…] / [home / pi / Lesson / Joystick_Switch / Joystick_Switch.py]

```
#import module
import smbus        # import smbus module to access I2C bus
import time         # import smbus module for sleep function
import RPi.GPIO as GPIO    #using GPIO pins from RPi.

#using BOARD numbering system to number the IOs
GPIO.setmode(GPIO.BOARD)

#Assign pins to the target
SW=11    # Switch connects to Pin 11 of Pi

#GPIO mode setup
GPIO.setup(SW,GPIO.IN)       # set pin connected to SW as input

# setup address for I2C and Analog In A0
address=0x48     #I2C address
AI0=0x40         #A0 address
AI1=0x41         #A1 address

# create object of SMBus class to access I2C based Python function
bus=smbus.SMBus(1)

#Main loop
while(1):                                   # infinite loop
    bus.write_byte(address,AI0)             # write AI0 value to I2C address
    bus.read_byte(address)                  # read I2C address value
    VRX=bus.read_byte(address)              # strore I2C address value to VRX
    print "VRX:",VRX,                       # print out VRX
    bus.write_byte(address,AI1)             # write AI0 value to I2C address
    bus.read_byte(address)                  # read I2C address value
    VRY=bus.read_byte(address)              # strore I2C address value to VRY
    print "VRY:",VRY,                       # print out VRX
    print "SW:",GPIO.input(SW)              # print SW state with carriage return
    time.sleep(0.1)                         # delay for 0.1 seconds
```

Code Discussion:

Similar to the exercise code in previous Unit, we use I2C interface to access the data from two input AI0 and AI1 and pint the data on the console window alternatively. We also detect the state of the SW port and print its state after two Joystick data.

(6) After completing the code, save this code as new file name [Joystick_Switch.py]. Select [Run / Run Module] from the menu bar or press [F5] key from keyboard to execute the code. Use [Ctrl] + [C] to terminate the execution.

5-7 EXERCISE RESULT：

When the position of the joystick is located at the center, the console window shows following results every 0.1 seconds.

```
===== RESTART: /home/pi/Lesson/ Joystick_Switch / Joystick_Switch.py ======
VRX: 131 VRY: 130 SW: 0
VRX: 131 VRY: 129 SW: 0
VRX: 131 VRY: 129 SW: 0
VRX: 131 VRY: 129 SW: 0
```

Move joystick from middle to right, the analog value of VRX increases until it close to maximum value 255.

```
===== RESTART: /home/pi/Lesson/ Joystick_Switch / Joystick_Switch.py ======
VRX: 147 VRY: 135 SW: 0
VRX: 154 VRY: 135 SW: 0
VRX: 199 VRY: 135 SW: 0
VRX: 255 VRY: 137 SW: 0
VRX: 255 VRY: 138 SW: 0
```

Move joystick from middle to left, the analog value of VRX decreases until it reaches minimum value 0.

```
===== RESTART: /home/pi/Lesson/ Joystick_Switch / Joystick_Switch.py ======
VRX: 83 VRY: 138 SW: 0
VRX: 68 VRY: 139 SW: 0
VRX: 41 VRY: 139 SW: 0
VRX: 11 VRY: 140 SW: 0
VRX: 11 VRY: 141 SW: 0
```

Move joystick from middle to top, the analog value of VRY increases until it close to maximum value 255.

```
===== RESTART: /home/pi/Lesson/ Joystick_Switch / Joystick_Switch.py ======
VRX: 123 VRY: 184 SW: 0
VRX: 120 VRY: 231 SW: 0
VRX: 134 VRY: 242 SW: 0
VRX: 148 VRY: 250 SW: 0
VRX: 148 VRY: 250 SW: 0
```

Move joystick from middle to bottom, the analog value of VRY decreases until it reaches minimum value 0.

```
===== RESTART: /home/pi/Lesson/ Joystick_Switch / Joystick_Switch.py ======
VRX: 125 VRY: 136 SW: 0
VRX: 125 VRY: 61 SW: 0
VRX: 124 VRY: 1 SW: 0
VRX: 120 VRY: 0 SW: 0
VRX: 118 VRY: 0 SW: 0
```

Press down the switch, the SW value will change from 0 to 1 (HIGH = Pressed). After release the switch, its value will change back to 0 (LOW = Depressed).

```
===== RESTART: /home/pi/Lesson/ Joystick_Switch / Joystick_Switch.py ======
VRX: 147 VRY: 135 SW: 0
VRX: 154 VRY: 135 SW: 0
VRX: 199 VRY: 135 SW: 1
VRX: 255 VRY: 137 SW: 1
VRX: 255 VRY: 138 SW: 0
```