Lesson 6 Falling and Standing Up

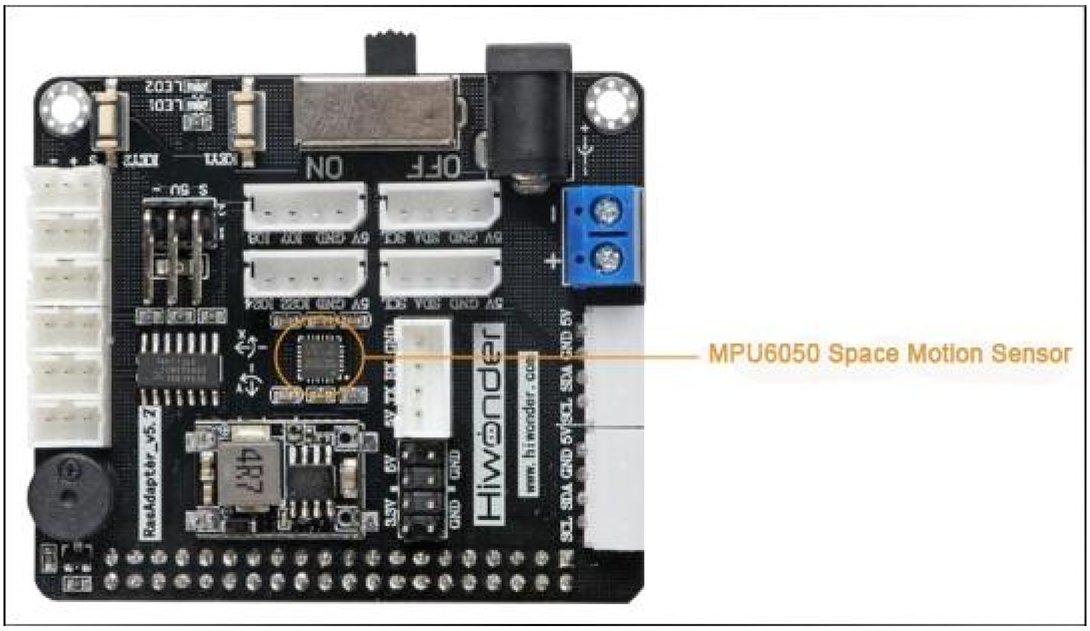
In this lesson, we will call action group in the specific program. Based on falling and standing up project, learn how to call action group in program, which deepen the proficiency of programming.

1. MPU6050 Space Motion Sensor

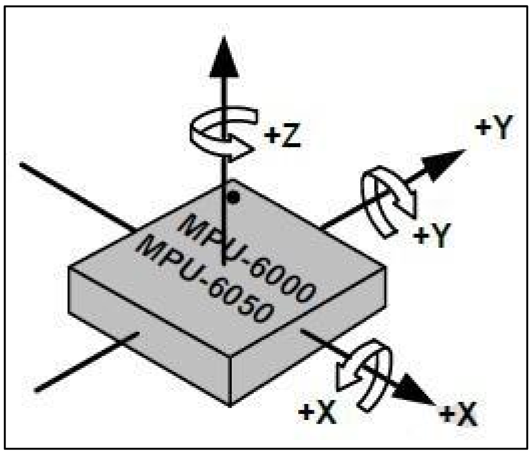
MPU6050 is a three-dimension sensor and the worlds first motion processor combining six-motion tracking devices .

It combines a 3-axis gyroscope and a 3-axis MEMS accelerometer, together with an onboard Digital Motion Processor (DMP), and the device can access external magnetometers or other sensors through an I^2C interface. After expansion, a 9-axis signal can be output through I^2C interface, and a non-inertial digital sensor, such as pressure sensor, can also be connected through I2C interface.

The position of MPU6050 on expansion board is as the figure shown below:



The definition of MPU6050 coordinate system: the front side of the chip faces forward, that is, the black dot on the chip locates in the upper left corner. At this time, with the geometric center of the chip surface as the origin.



This project is mainly to measure the angular velocity by the gyroscope on the sensor and the acceleration by the accelerometer. We will briefly understand these two sensors in the following content.

Gyroscope

The principle of gyroscope:
The orientation of the axis of a rotating object is unaffected by external forces. Based on this principle, it can be used to maintain the orientation. Then use various methods to read the orientation of axis and transmit the data signal to the control system.

Accelerometer

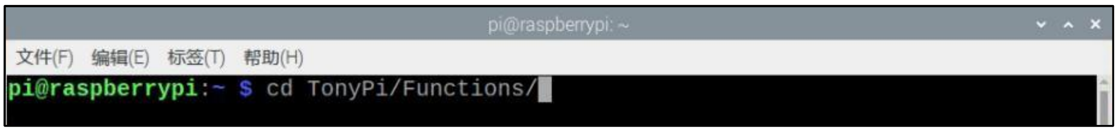
The principle of accelerometer:
Accelerometer can measure acceleration. It is usually composed of a mass block, damper, elastic element, sensitive element and an adaptive circuit. During the acceleration process, the sensor uses Newton's second law to obtain the acceleration value by measuring the inertial force applied to the mass

2. Operation Steps

1) Turn on TonyPi Pro and connect to VNC.

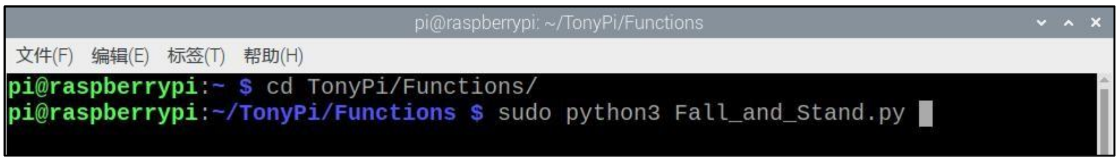
2) Press “Ctrl+Alt+T” or click icon in the upper left corner to open LX terminal.

3) Enter “cd TonyPi/Functions/” command, and then press “Enter” to come to the category of games programmings.



```
pi@raspberrypi:~ $ cd TonyPi/Functions/
```

4) Enter “sudo python3 Fall_and_Stand.py” command, and then press “Enter” to start game.


pi@raspberrypi: ~/TonyPi/Functions
文件(F) 编辑(E) 标签(T) 帮助(H)

```
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo python3 Fall_and_Stand.py
```

5) If you want to exit the game programming, press “Ctrl+C” in the LX terminal interface. If the exit fails, please try it few more times.

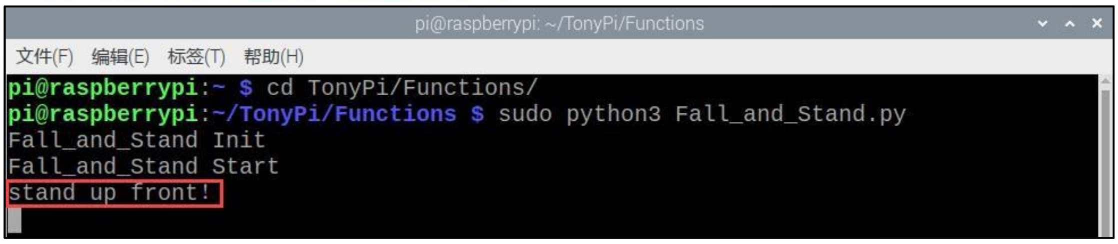
3. Project Outcome

Note:

① Place TonyPi Pro on a flat and smooth surface, otherwise it may affect the performance.

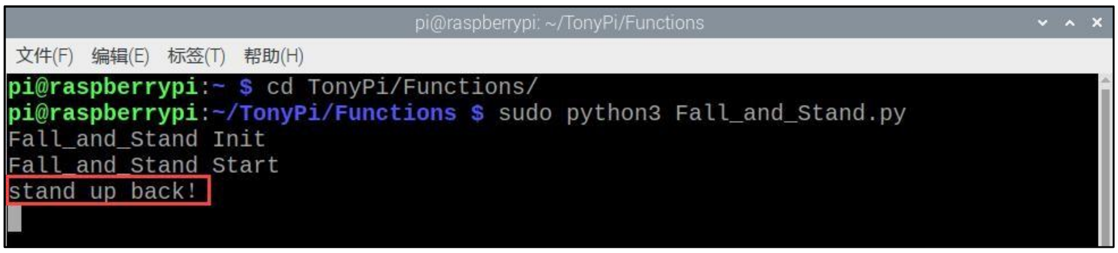
② In addition, please place the robot gently towards the front or back, not on its side.

After starting the game, the robot is initially at the stance. We gently place the robot towards the front firstly. After falling, TonyPi Pro will call action group to stand up again. At same time, you can view the direction of robot when it fell on the terminal interface, as the figure shown below：



```
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo python3 Fall_and_Stand.py
Fall_and_Stand Init
Fall_and_Stand Start
stand up front!
```

Similarly, place robot toward the back, and then the terminal interface will print the direction of robot when it fell, as the figure shown below:



```
pi@raspberrypi:~ $ cd TonyPi/Functions/
pi@raspberrypi:~/TonyPi/Functions $ sudo python3 Fall_and_Stand.py
Fall_and_Stand Init
Fall_and_Stand Start
stand up back!
```

4. Project Analysis

◆ Read the Data of MPU6050 Sensor

1) Firstly, start MPU6050 sensor and set the range of sensor. The "set_gyro_range" is to set the working range of gyroscope and the "set_accel_range" is to set the range of accelerator.

```
mpu = Mpu6050.mpu6050(0x68)

mpu.set_gyro_range(mpu.GYRO_RANGE_2000DEG)

mpu.set_accel_range(mpu.ACCEL_RANGE_2G)
```

2) Then the sensor value can be acquired, and the acquired value is converted into an angle value. Since this project only includes falling forward and backward, we just need to pay attention to the angle on y-axis and use "degrees" function to implement.

```
accel_date = mpu.get_accel_data(g=True)

angle_y = int(math.degrees(math.atan2(accel_date['y'], accel_date['z'])))
```

◆ Call Action Group

1) The related programs of action group are stored in “ActionGroupControl” module. In order to simple and neat appearance, import this module in advance and use “AGC” to present in the following program.

```
import HiwonderSDK.ActionGroupControl as AGC
```

2) Next, the robot can perform corresponding action by setting the angle threshold. Take the program of robot falling backward as an example:
When the absolute value of “angle_y” is set to be greater than 160, it is determined that the robot fell backwards. At this time, “stand up back!” is printed in the terminal, and the action group “stand_up_back” is called.

```
if abs(angle_y) > 160:
```

```
        print("stand up back！ ")

        AGC.runActionGroup('stand_up_back')
```

3) Use the “runActionGroup” function to call action group, and the parameter is the corresponding action group name, so it must be consistent with the action group name when saving.
4) To avoid the accidental injury caused by robot and increase the detection accuracy, please wait for a period of time before performing the action.

Therefore, it can be implemented by adding the flag “count1”. When the program detects that “angle_y” is greater than 160 each time, it counts once. When the count reaches a certain number of times, the action can be performed. When “count1” is greater than or equal to 5, the “stand_up_back” action should be performed and the flag is cleared to 0, and then wait for the next counting.

```
if abs(angle_y) > 160:
        count1 += 1
else:
        count1 = 0

if count1 >= 5:
        count1 = 0
        print("stand up back！ ")
        AGC.runActionGroup('stand_up_back')
```

5) The program for the robot to fall forwards is similar to the processing method above. The program code after combining the two situations is as follows:

```
count1 = 0
count2 = 0
def standup():
        global count1, count2
                accel_date = mpu.get_accel_data(g=True) #get the sensor data
                angle_y = int(math.degrees(math.atan2(accel_date['y'], accel_date['z']))) #convert
the obtained data into an angle value

                if abs(angle_y) > 160: #If the y-axis angle is greater than 160, count1 is
                incremented by 1, otherwise it is cleared
                     count1 += 1
                else:
                       count1 = 0
                if abs(angle_y) < 10: #If the y-axis angle is less than 10, count2 is incremented
                by 1, otherwise it is cleared
                     count2 += 1
                else:
```

```
            count2 = 0

        time.sleep(0.1)

        if count1 >= 5: #Stand up after the robot falls forward for a certain period of
        time.

            count1 = 0

            print("stand up back！ ")# Print the performed action name

            AGC.runActionGroup('stand_up_back')# Perform action

        elif count2 >= 5: #Stand up after the robot falls backward for a certain period of
        time.
             count2 = 0

            print("stand up front！ ")# Print the performed action name

            AGC.runActionGroup('stand_up_front')# Perform action
```

5. Exception Handling Method

Errors and Exceptions are two types of situations included in Python. Errors are mainly the common Synatax Error while exceptions refer to situations where the syntax and expression are correct, but errors will also be found during operation. In Python, syntax errors are directly displayed in the relevant terminal window, and exceptions can be prompted for errors or caught.

Because in the program, the abnormal error prompt may affect the display of the output result. At this time, you can use “try...except...” to handle the exception, and print the error directly instead of displaying it in the form of an error message. The code example is as follows:

```
try：

except BaseException as e:
```

```
        print(e)
```

Apart from “try...except...”, the exception handling statements also have “try...finally”, “raise” and “assert”. Their functions are as follows:

Exception handling statement	Function
try...except...	Receive exception notifications and catch exceptions.
try...finally	Execute the statement that must be executed.
raise	Send exception notifications and enter the exception status.
assert	Optionally send exception notifications of AssertionError type based on conditions.