Program Analysis

1. File Path

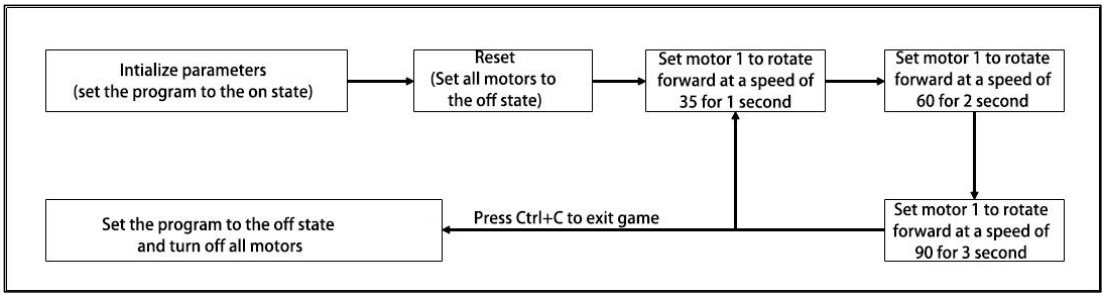
The program corresponding to this lesson is stored in **/home/pi/MasterPi/HiwonderSDK/MotorControlDemo.py**

2. Performance

After the game is started, the DC motor first rotates forwards at a speed of 35 for 1 second, then at a speed of 90 for 3 seconds. This continuous rotation pattern is repeated in a loop.

3. Program Analysis

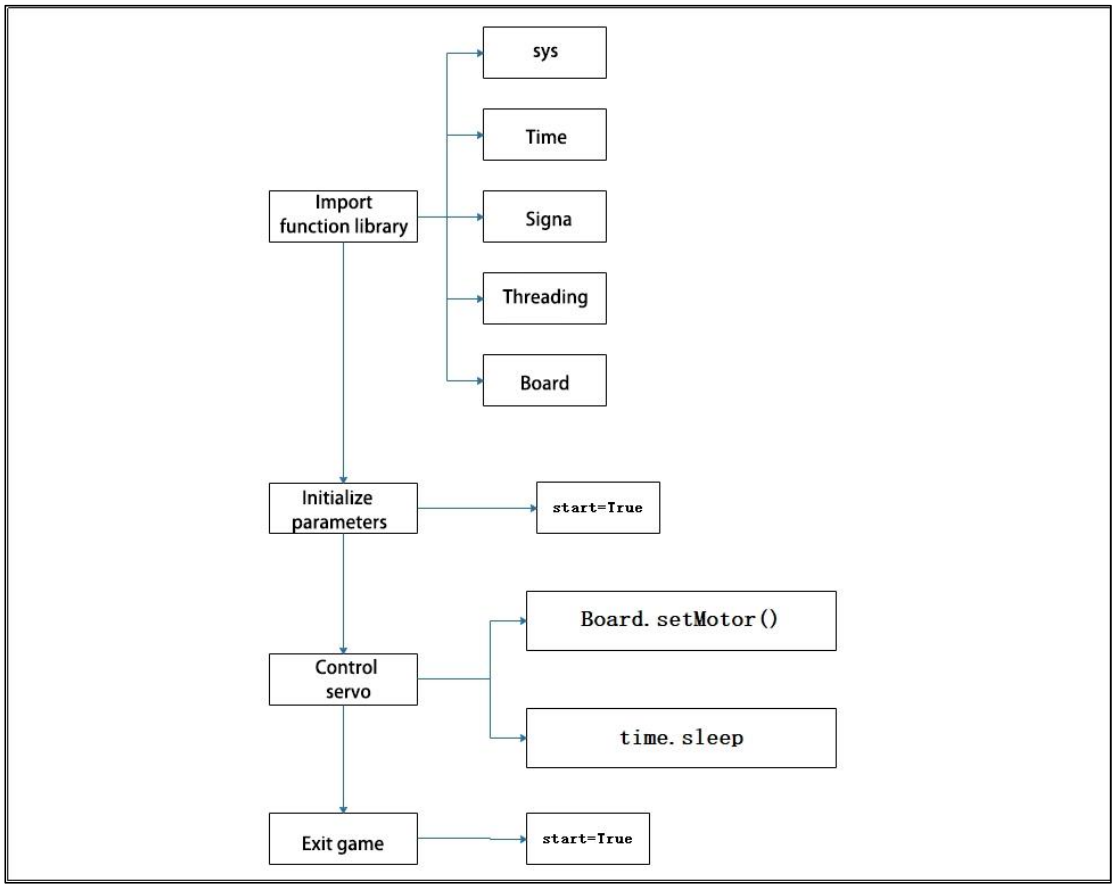
3.1 Program Logic



From the above picture, we can know that this game will first perform initialization and modify the parameters in program. Next it controls servo 1 to rotate forwards at a speed of 35 for 1 second, then at a speed of 60 for 2 seconds, and then a speed of 90 for 3 seconds. This pattern is repeated in a loop.

When pushing “Ctrl+C” to close the game, the program and the buzzer will stop working.

3.2 Program Analysis



From the above flow diagram, the program primarily divides into four parts, namely importing the function library, initializing parameters, controlling servo and closing the game. The following content will provide an explanation based on the logic of the program as depicted in the diagram.

- **Import Function Library**

Firstly, import sys, time, signal, threading and Board functions. The below table will explain the library files.

```
3  import sys
4  sys.path.append('/home/pi/MasterPi/')
5  import time
6  import signal
7  import threading
8  import HiwonderSDK.Board as Board
```

Library File	Function
sys	Provide a multitude of functions and variables
Time	The expression for handling time within a program refers to the method of obtaining the system time and formatting its output
signal	Used for receiving and processing the signal.
threading	Providing an environment for running multiple threads
Board	control sensors for executing control operations.

- **Initialization**

When executing a program, it is generally necessary to perform initialization within the program to facilitate the realization of the sequence functionalities. In this case, initialization involves setting the “Start” parameter in the program to True, indicating that the program is running.

```
27  # 关闭所有电机
28  def MotorStop():
29      Board.setMotor(1, 0)
30      Board.setMotor(2, 0)
31      Board.setMotor(3, 0)
32      Board.setMotor(4, 0)
33
34  start = True
```

- **Drive Motor**

After the initialization is completed, the next step is to control the motor's rotation. Here, the functions Board.setMotor() and time.sleep() are used.

Board.setMotor() function: The function takes two parameters inside the parentheses. For example, Board.setMotor(1, 35), where parameter 1 represents motor 1, and parameter 35 represents the speed of rotation, ranging from -100 to 100, with negative values indicating reverse rotation.

time.sleep() function: This function is used to control the duration of actions, such as the duration of the buzzer.

```
49          Board.setMotor(1, 35)   #设置1号电机速度35
50          time.sleep(1)
51          Board.setMotor(1, 60)   #设置1号电机速度60
52          time.sleep(2)
53          Board.setMotor(1, 90)   #设置1号电机速度90
54          time.sleep(3)
```

- **Quid Game**

The “start” parameter is set to “False” to quit the game and turn off all the motors.

```
35  #关闭前处理
36  def Stop(signum, frame):
37      global start
38
39      start = False
40      print('关闭中...')
41      MotorStop()   # 关闭所有电机
```