

Lesson1 How to use MG90S Micro Servo

Table

1. Set up an environment for Raspberry Pi programs to run.....	2
2. Introduction of the servo.....	2
2.1 Servo Physical map.....	2
2.2 Pins.....	2
2.3 Parameters.....	2
2.4 Working Principle.....	3
2.5 PWM Control.....	4
3. Component List.....	5
4. Circuit connection.....	6
5. Upload the code to Raspberry Pi 4B and testing.....	6
6. Make your suggestion and get support.....	8

1. Set up an environment for Raspberry Pi programs to run

Please refer to the documentation“[CKK0015-main\Tutorial\Raspberry Pi\2 Installing and Configuring Raspberry Pi System](#)”

2. Introduction of the servo

2.1 Servo Physical map

As shown in the figure below, a single SG90 Micro Servo consists of three parts: servo, servo plate, and screws



2.2 Pins



2.3 Parameters

2.3.1 Electrical parameters

No.	(Item)	5V
2.3.1.1	No-load current	120±20mA

2.3.1.2	load current	160±20mA
2.3.1.3	No-load speed	0.11sec/60°
2.3.1.4	Quiescent Current	10mA
2.3.1.5	No-load life	50000 times
2.3.1.6	stall torque	1.5Kg.-cm
2.3.1.7	Stall current	≅ 1A
2.3.1.8	Temperature drift (ambient temperature 25°)	≅ 1°
2.3.1.9	temperature resistance	-20°C

2.3.2 Structural parameters

No.	(Item)	Specification
2.3.2.1	Physical dimension	32.6*12.1*22.5mm
2.3.2.2	Weight	12G
2.3.2.3	Mechanical limit angle	360°
2.3.2.4	Gear type	Grade 5 Metal Gear Set
2.3.2.5	Output tooth spline	20T
2.3.2.6	Wire length	250±5mm
2.3.2.7	Gear virtual position	≅ 2°
2.3.2.8	Wire	50C/0.08 OD1.0 JR Connector
2.3.2.9	Swing arm	One Word Rocker Cross rocker Flower rocker
2.3.2.10	Motor	DC motor
2.3.2.11	Shell material	PC

2.3.3 Control parameter

No.	(Item)	Specification
2.3.3.1	Control signal	PWM cycle 50HZ
2.3.3.2	Pulse width range	1000us-2000us
2.3.3.3	Amplifier type	Number
2.3.3.4	Rotation angle	90°±3°(when 1000~2000usec)
2.3.3.5	Midpoint	1500usec
2.3.3.6	Dead zone width	≅ 5usec
2.3.3.7	Direction of rotation	Clockwise(when 1000~2000usec)
2.3.3.8	Return error	≅ 1°
2.3.3.9	Over-operating angle range	180°(500-2500usec)
2.3.3.10	Angle error on both sides	≅ 5°

2.4 Working Principle

The control signal of the servo is a PWM signal with a period of 20ms, in which the pulse

width is from 0.5ms-2.5ms, and the corresponding position of the servo is 0-180 degrees, which changes linearly. Provide it with a certain pulse width, and its output shaft will remain at a corresponding angle, no matter how the external torque changes, until a new pulse signal of different width is provided to it, it will change the output angle to the new corresponding position. There is a reference voltage inside the servo, which generates a reference signal with a period of 20ms and a width of 1.5ms. There is a comparator that compares the applied signal with the reference signal to determine the direction and size, thereby generating the rotation signal of the motor. The internal control circuit board of the servo receives the control signal from the signal line, and controls the rotation of the motor. The motor drives a series of gear sets, which are driven to the output steering wheel after deceleration. The output shaft of the servo is connected to the position feedback potentiometer. When the steering wheel rotates, it drives the position feedback potentiometer. The potentiometer will output a voltage signal to the control circuit board for position feedback. The control circuit board determines the rotation direction and speed of the motor according to the position of the output shaft, so that the output shaft stops when it reaches the target.

2.5 PWM Control

2.5.1 PWM Introduction

Pulse width modulation (PWM) refers to the use of the digital output of a microprocessor to control an analog circuit, and is a method of digitally encoding the level of an analog signal.

PWM (Pulse Width Modulation): Pulse Width Modulation

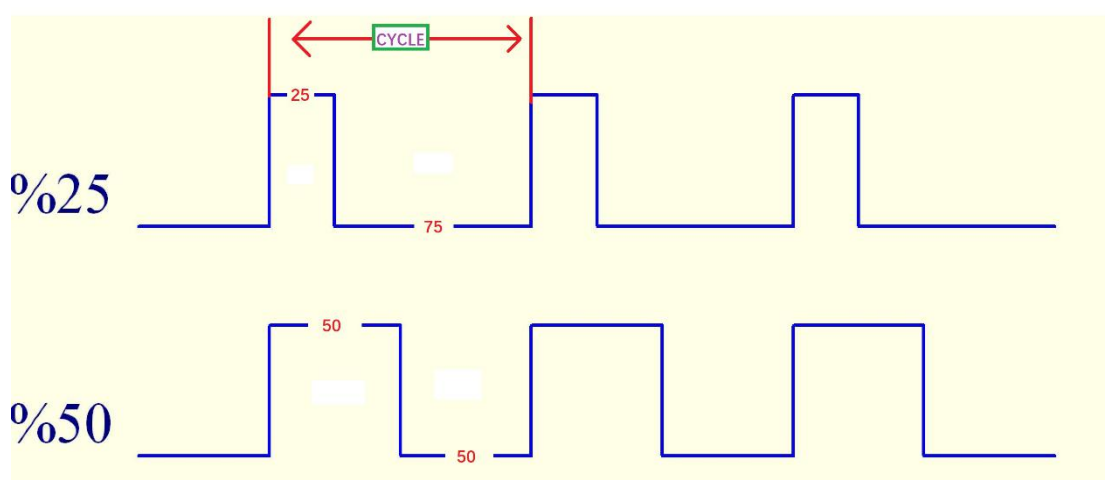
Pulse: square wave, frequency (freq)

Width: the width of the high level, the duty cycle (duty)

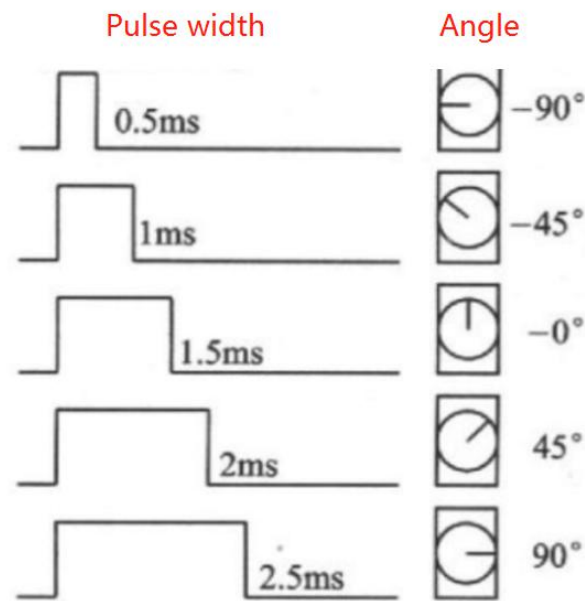
Cycle: CYCLE

Duty cycle: the proportion of high level (100&%)

Duty cycle static diagram:



The relationship between the output angle of MG90S Micro Servo and the pulse width of the input signal




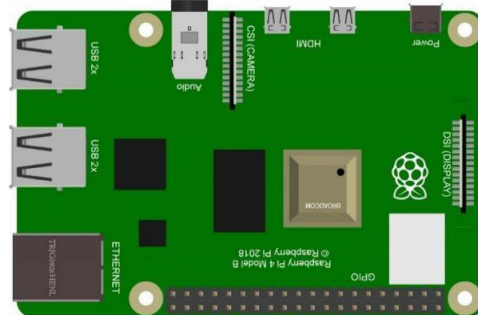
CYCLE=20ms

Duty_16=65532*Pulse Width/CYCLE

The relationship between the corresponding pulse width and duty_u16:

Pulse Width(ms)	Duty_16
0.5	1638
1	3276
1.5	4914
2	6552
2.5	8192

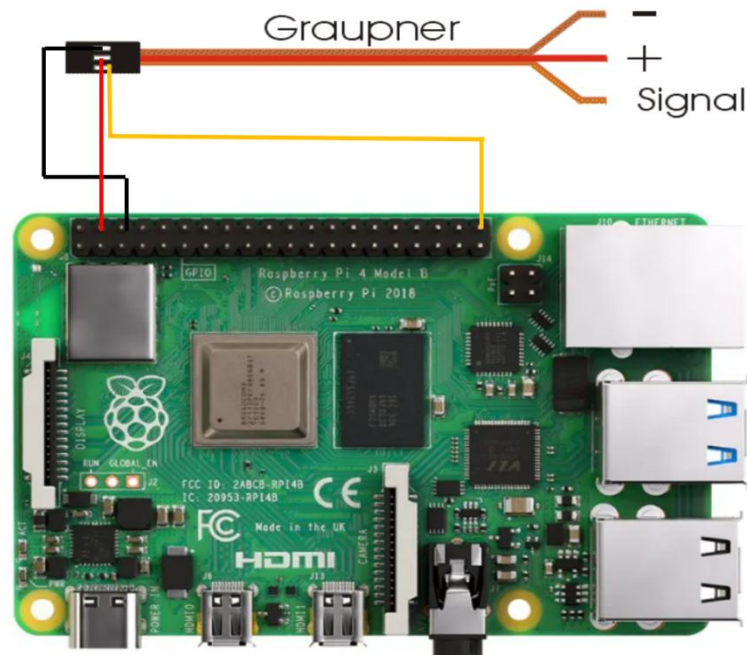
3. Component List

Component	QTY	Picture	Remark
MG90S Micro Servo	1		You need to prepare these by yourself. These just as an example, you can DIY what is you want and prepared
Raspberry Pi 4B	1		

4. Circuit connection

Connect the yellow signal pin of the MG90S Micro Servo to pin9 of the UNO board through a DuPont cable, connect the red VCC pin of the MG90S Micro Servo to the 5V of the UNO board, and connect the brown GND pin of the MG90S Micro Servo to the GND of the UNO board.

Circuit connection diagram:



The corresponding connection relationship is shown in the table below:

Raspberry Pi 4B Board Pin	MG996R Servo Pin
21 (BCM)	signal
5V	VCC
GND	GND

5. Upload the code to Raspberry Pi 4B and testing

The Servo for Raspberry Pi code is placed in the “[CKK0015-main\Tutorial\Raspberry Pi\Code](#)” folder.

Please refer to the document "[CKK0015-main\\Tutorial\\Raspberry Pi\\2 Installing and Configuring Raspberry Pi System](#)" to learn how to upload code to Raspberry Pi

When the code is upload to the Raspberry Pi 4B and run,you can see that the MG996R Servo rotates 180 degrees in a loop.

Note: The pin number used in the code is the "pin number corresponding to BCM"

Python code

Servo_test.py

code as below:

```
1.  #-*- coding: utf-8 -*-
2.  #!/usr/bin/env python
3.
4.  import RPi.GPIO as GPIO
5.  import time
6.  import signal
7.  import atexit
8.
9.  atexit.register(GPIO.cleanup)
10.
11. servopin = 21
12. GPIO.setmode(GPIO.BCM)
13. GPIO.setup(servopin, GPIO.OUT, initial=False)
14. p = GPIO.PWM(servopin,50) #50HZ
15. p.start(0)
16. time.sleep(2)
17.
18. while(True):
19.     for i in range(0,181,10):
20.         p.ChangeDutyCycle(2.5 + 10 * i / 180) #set rotation angle
21.         time.sleep(0.02)           #wait 20ms for the cycle time
22.         p.ChangeDutyCycle(0)       #Initialize
23.         time.sleep(0.2)
24.
25.     for i in range(181,0,-10):
26.         p.ChangeDutyCycle(2.5 + 10 * i / 180)
27.         time.sleep(0.02)
28.         p.ChangeDutyCycle(0)
29.         time.sleep(0.2)
```

6. Make your suggestion and get support

THANK YOU for participating in this learning experience!

If you find errors, omissions or you have suggestions and/or questions about this document, please feel free to contact us: cokoino@outlook.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our Amazon Store by search for "[LK COKOINO](#)" on Amazon. We will continue to launch fun, cost-effective, innovative and exciting products.

Thank you again for choosing Cokoino products.

LK COKOINO