Servo Motor Documentation - Raspberry Pi 5 Integration

El Mehdi Adnani Kadmiri

July 21, 2025

1 Description

Servo motors (or "servo") is an electromechanical device used to precisely control the angular position, speed, or torque of a mechanism. It typically consists of a motor (often an electric motor), a position sensor, and a control circuit, all housed in one enclosure. The servomotor operates in a closed loop: it continuously compares the current position to the desired position, then adjust motor movement to eliminate any deviation. This is called a position-controlled system.

Servomotors are used in many fields, including robotics, automatic control systems, and industrial machines to perform precise and controlled movements. They are particularly valued for their accuracy, responsiveness, and ability to maintain a stable position even in the presence of external disturbances.

2 Applications

- **CNC Machines:** Computer numerical control (CNC) machines use servo motors to precisely control the cutting tools and workpieces. For complex item manufacturing, such as in metalworking and woodworking, servo motors' precise control over a wide range of speeds and locations is crucial.
- **Robotics:** Industrial robots, which perform a variety of tasks like welding, painting, and picking and packing, frequently run on servo motors. Robots can carry out intricate movement sequences thanks to the motors' accuracy and reactivity, which raises the degree of automation and increases the flexibility of production processes.
- **Material Handling Systems:** Servo motors make it easier for commodities to move through conveyors, sorters, and automated storage and retrieval systems (AS/RS) in logistics and warehousing. Their precise positional and speed control maximizes material flow, reducing handling times and boosting output.

3 Working Principle

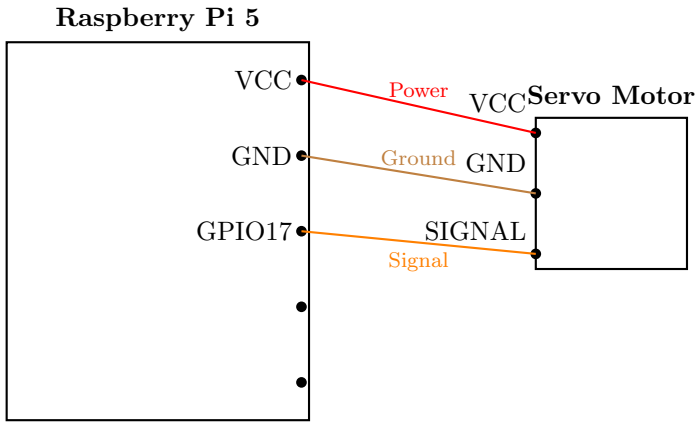
Servo motors are controlled using **Pulse Width Modulation (PWM) signals**. A PWM signal is a series of pulses where the width of the "on" pulse determines the desired position of the servo motor's shaft. The servo motor's internal circuitry interprets the pulse width and adjusts the motor's position accordingly.

- A 1 ms pulse = 0° (minimum position)
- A 1.5 ms pulse = 90° (center position)
- A 2 ms pulse = 180° (maximum position)

PWM signals are repeated approximately every 20 ms (50 Hz). The duration of the pulse within that time frame determines the servo position.

4 Wiring Diagram

Servo Motor Pin	Raspberry Pi Pin	Function
VCC	4.8 - 6 V	Power
GND	GND	Ground
OUT	GPIO17 (or any output)	signal



5 Libraries Used

Python: RPi.GPIO

These libraries allow for PWM signal generation using GPIO pins.

- **RPi.GPIO:**
  - Import: `import RPi.GPIO as GPIO`
  - Setup: `GPIO.setmode(GPIO.BCM)`
  - PWM Start: `GPIO.PWM(18, 50)`

C: pigpio Library/SoftPWM

`pigpio` offers precise PWM control in C.

- Include: `#include <pigpio.h>`
- Init: `gpioInitialise();`
- PWM: `gpioServo(18, 1500);`

`softPwm` is an extension of the WiringPi library that allows software PWM generation on any GPIO pin. It is particularly useful for controlling devices like servo motors, which require precise PWM signals for angle positioning.

- Setup: `softPwmCreate(pin, initialValue, range)`
- Change PWM value: `softPwmWrite(pin, value)`
- Example range: 0–100 (interpreted as 0% to 100% duty cycle)

6 Python Example

```
import RPi.GPIO as GPIO
import time

SERVO_PIN = 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(SERVO_PIN, GPIO.OUT)

pwm = GPIO.PWM(SERVO_PIN, 50)  # 50 Hz
pwm.start(0)

try:
while True:
# Rotate to 0
pwm.ChangeDutyCycle(2.5)
time.sleep(1)

# Rotate to 90
pwm.ChangeDutyCycle(7.5)
time.sleep(1)

# Rotate to 180
pwm.ChangeDutyCycle(12.5)
time.sleep(1)

except KeyboardInterrupt:
pwm.stop()
GPIO.cleanup()
```

7 C Example

```
#include <wiringPi.h>
#include <softPwm.h>
#include <stdio.h>

#define SERVO_PIN 1 // wiringPi pin 1 = BCM 18

int main(void) {
        wiringPiSetup();
        softPwmCreate(SERVO_PIN, 0, 200); // Range 0-200

        while (1) {
                // Rotate to 0
                softPwmWrite(SERVO_PIN, 5);
                delay(1000);

                // Rotate to 90
                softPwmWrite(SERVO_PIN, 15);
                delay(1000);

                // Rotate to 180
                softPwmWrite(SERVO_PIN, 25);
                delay(1000);
        }
```

```
                return 0;
        }
```

8 Conclusion

The servo motor enables precise angular motion control and is easily controlled by PWM signals on the Raspberry Pi. Both Python and C implementations demonstrate how simple it is to move the motor between defined positions for practical applications like robotics or camera panning.