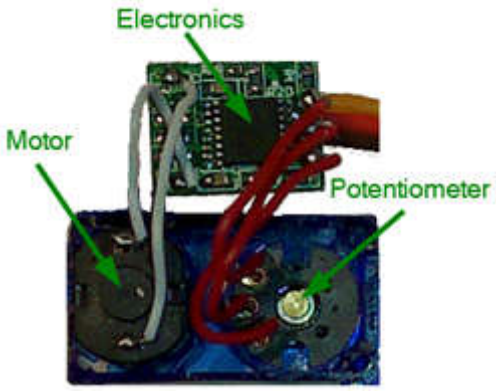
SERVOMOTORS

The source code of all examples can be downloaded from here.

How servomotors work

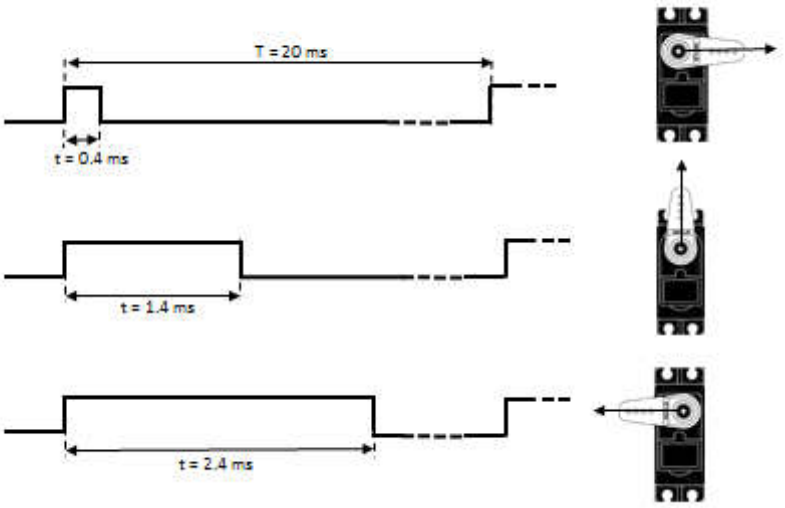
Normally a servomotor is built from a standard DC with an internal feedback control (a potentiometer attached to a reduction gear and some electronics). Its main use is to turn the gear axis to a predefined direction. The motor has 3 input terminals, GND, POWER and SIGNAL. A pulse-width modulated (PWM) signal is applied to the SIGNAL input and the direction of the axis is determined by the duration of the pulse (see the tutorial about LED dimming for more information about PWM). Keep in mind that the direction is not determined by the duty cycle, but by the length t of the on-pulse.

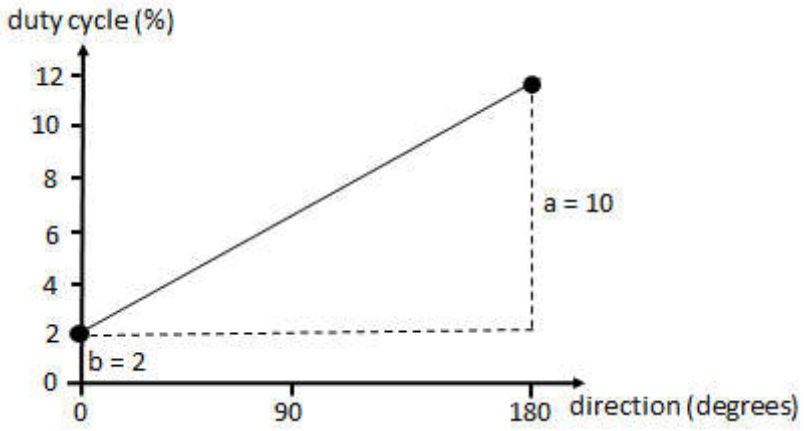


Many servomotors use a PWM frequency of fPWM = 50 Hz corresponding to a PWM period of T = 20 ms. The relationship between the pulse length t and the direction is linear and depends on the motor and the gear

Example:

t	Duty Cycle	Direction
0.4 ms	0.4/20 = 2%	0 degs
1.4 ms	1.4/20 = 7%	90 degs
2.4 ms	2.4/20 = 12%	180 degs





$$\text{duty cycle} = \frac{a}{180} * \text{direction} + b$$

Servo motors are widely used in radio-controlled toys (cars, airplanes, etc.), but also in industrial applications where a precise rotation position is needed (e.g. in robotics). An alternative (and often better solution) to achieve direction control is the **stepper motor** (see next tutorial).

Experiment 1: Motor directly connected to the GPIO

Aim:
Connect a small servomotor directly to the 5 V supply of the Raspberry Pi and control it by a GPIO digital output port using software PWM.

Caution:

Only use micro servomotors (e.g. TowerPro SG90) because of the current limitation of the 5 V supply.

Connect the brown (or black wire) to GND (pin #6), the red wire to 5 V (pin #2) and the yellow wire to any GPIO output. The servomotor is powered by 5 V, but controlled by 3.3 V.

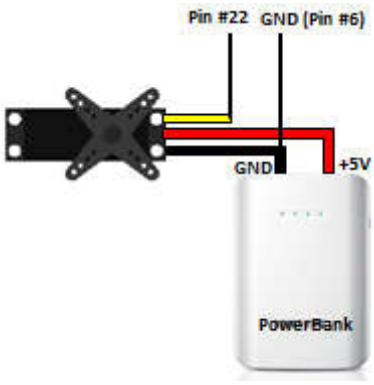
Program: [▶]

```
# Servo1.py

import RPi.GPIO as GPIO
import time

P_SERVO = 22 # adapt to your wiring
fPWM = 50  # Hz (not higher with software PWM)
a = 10
b = 2

def setup():
    global pwm
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(P_SERVO, GPIO.OUT)
    pwm = GPIO.PWM(P_SERVO, fPWM)
    pwm.start(0)

def setDirection(direction):
    duty = a / 180 * direction + b
    pwm.ChangeDutyCycle(duty)
    print "direction =", direction, "-> duty =", duty
    time.sleep(1) # allow to settle

print "starting"
setup()
for direction in range(0, 181, 10):
    setDirection(direction)
direction = 0
setDirection(0)
GPIO.cleanup()
print "done"
```

Highlight program code (Ctrl+C copy, Ctrl+V paste)

Remarks:
You must wait in your program until the servomotor is settled at the new direction before you change the direction or you finish the program.

It is a good idea to power the servomotor by an external voltage source, because servos can cause "electrical noise" that could cause the Raspberry Pi to act erratically or even break it completely. Use the following wiring:

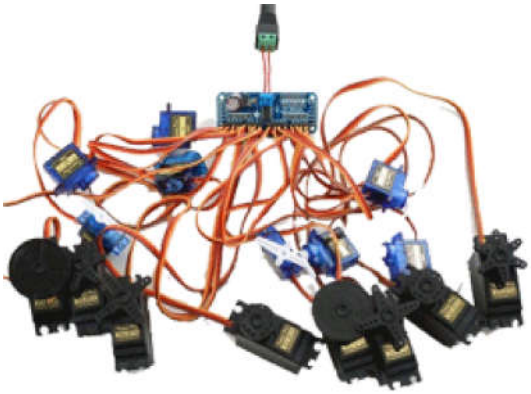


Experiment 2: Servomotors driven by the PCA9685

Generating stable PWM signals by software is a heavy burden for a microprocessor running Linux, because there are other processes running on the system that may interrupt the PWM generating code. It is a better solution to use a special purpose external chip that does the job, especially if you need several PWM signals.

The PCA9685 from NXP is designed to handle up to 16 PWM signals controlled by an I^2C interface. It is called "LED controller", but can handle servomotors as well.

The chip is only available in an 28-pin SMT package, so you either need to solder it on a SMT adapter or buy a pre-assembled package like the Adafruit 16-Channel Servo Driver Interface (see image at right).



Aim:
Connect a servomotor to channel 0 of a PCA9685 based module and perform the same action as in the previous example.

On order to simplify your programming task, use a small library *PCA9685.py* inspired by the Adafruit PWM Servo Driver (download here and copy in same directory with your program).

Program: [▶]

```
# Servo2.py
# Two servo motors driven by PCA9685 chip

from smbus import SMBus
from PCA9685 import PWM
import time

fPWM = 50
i2c_address = 0x40 # (standard) adapt to your module
channel = 0 # adapt to your wiring
a = 8.5 # adapt to your servo
b = 2  # adapt to your servo

def setup():
    global pwm
    bus = SMBus(1) # Raspberry Pi revision 2
    pwm = PWM(bus, i2c_address)
    pwm.setFreq(fPWM)

def setDirection(direction):
```

```
    duty = a / 180 * direction + b
    pwm.setDuty(channel, duty)
    print "direction =", direction, "-> duty =", duty
    time.sleep(1) # allow to settle

print "starting"
setup()
for direction in range(0, 181, 10):
    setDirection(direction)
direction = 0
setDirection(0)
print "done"
```

Highlight program code (Ctrl+C copy, Ctrl+V paste)

Remarks:
The parameters a and b must be adapted to the type of servomotor you use.

Experiment 3: A robot arm with six degrees of freedom

A robot arm has a number of degrees of freedom to move in any desired position. To control the position precisely, servomotors (or stepper motors) are widely used.

Aim:
Achieve a simple demonstration with a simple and cheap robot that has six degrees of freedom. A PCA9685 breakout module is used.

6DOF Mechanical Robotic Arm

(to be done)