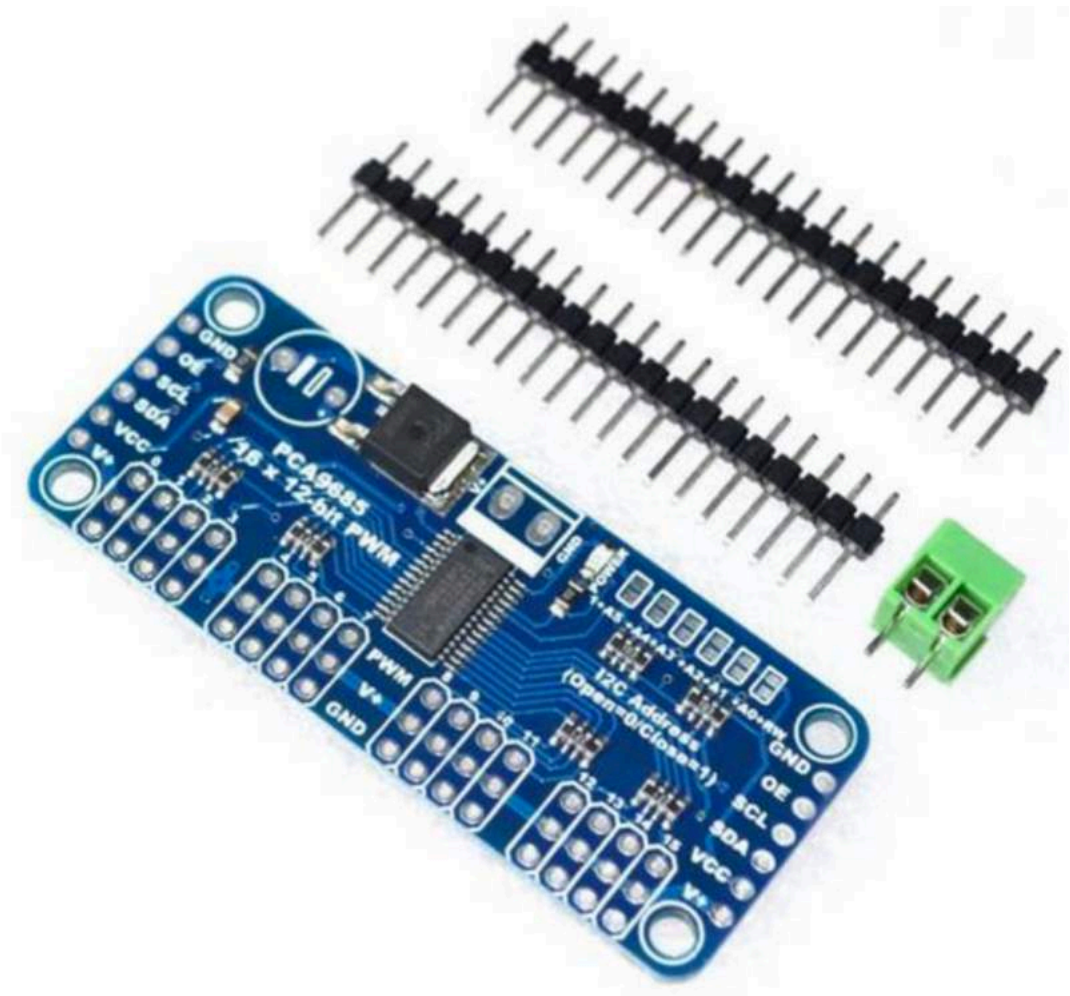
How to drive servo with PCA9685A servo driver board?

A lot of DIY projects have to deal with servos such as robotic arm, robots or steering control of aircraft model. If you want to drive multiple servos, it is necessary to use servo drive based PCA9685 chip board. The PCA9685 servo driver board in this tutorial can drive 16 servos using the I2C protocol agreement.

The chip is shown as below:



Principle:

Normally many MCU integrated servos are driven by PWM. However, there is only one PWM channel on GPIO 18 PIN which limits the number of servos. We can use PCA9685 servo drive board to drive multiple servos (16-channel 12-bit PWM / Servo Driver). It can offer extra 6V power supply. We can send the command to PCA9685 via i2c interface and

direct the servos to make different moves. In this way, a servo board is able to control 16 servos.
Tasks:

1. Master the use of wiringpi and GPIO
2. Master the use of PCA9685A Servo Driver Board
3. Learn how to drive Servo to move with underlying protocol

Electronic Components:

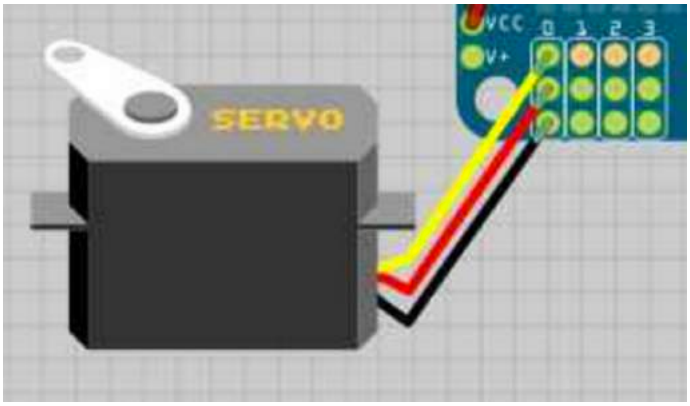
1* Raspberry Pi
1* Power Supply
1* PCA9685A Servo Drive Board
1* 9g Servo
Several Pieces of DuPont Wires

You may purchase all the components from this store:
https://www.ebay.com/sh/mystore

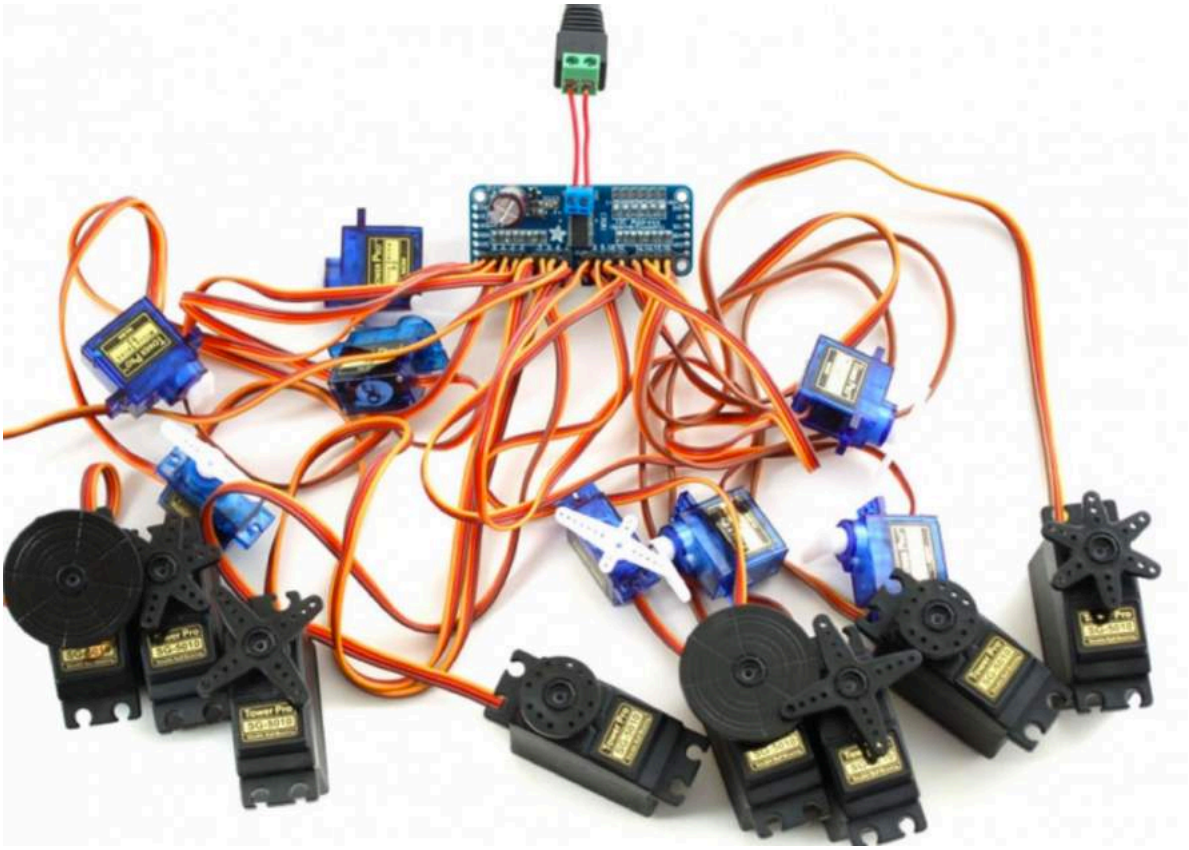
Experimental Procedures:

1. Wire Connection:

Insert the servo lead into the control board of PCA9685A servo.



If you connect multiple servos, please refer the above picture for wiring connection. Kindly notice that the V+ port requires extra power supply.



Find the I2C interface on PCA9685A board and connect them with the raspberry pi GPIO PINS.

Login and check / boot / overlay / README file:
cat /boot/overlay/README
You may see the below picture:

```
Name:   i2c-pwm-pca9685a
Info:   Adds support for an NXP PCA9685A I2C PWM controller on i2c_arm
Load:   dtoverlay=i2c-pwm-pca9685a,<param>=<val>
Params: addr                    I2C address of PCA9685A (default 0x40)
```

2. Modify the configuration

Login and edit /boot/config.txt file

sudo su –

vim.tiny /boot/config.txt

Add the following parameters:

device_tree=bcm2710-rpi-3-b.dtb

dtoverlay=i2c_arm=on

dtoverlay=i2c-pwm-pca9685a,addr=0x40

```
dtparam=i2c_arm=on
device_tree=bcm2710-rpi-3-b.dtb
dtoverlay=i2c-pwm-pca9685a,addr=0x40
dtparam=audio=on
```

Save and exit. Then reboot the raspberry pi.

4. Install software

sudo apt-get install python-smbus

sudo apt-get install i2c-tools

It will appear as following picture:

```
pi@raspberrypi:~ $ sudo apt-get -y install python-smbus
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  i2c-tools
Suggested packages:
  libi2c-dev
The following NEW packages will be installed:
  i2c-tools python-smbus
0 upgraded, 2 newly installed, 0 to remove and 77 not upgraded.
Need to get 60.8 kB of archives.
After this operation, 286 kB of additional disk space will be used.
Get:1 http://archive.raspberrypi.org/debian/ jessie/main i2c-tools armhf 3.1.1+svn-2 [51.3 kB]
Get:2 http://archive.raspberrypi.org/debian/ jessie/main python-smbus armhf 3.1.1+svn-2 [9462 B]
Fetched 60.8 kB in 9s (6260 B/s)
Can't set locale; make sure $LC_* and $LANG are correct!
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_CTYPE = "zh_CN.UTF-8",
        LANG = "en_GB.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to a fallback locale ("en_GB.UTF-8").
locale: Cannot set LC_CTYPE to default locale: No such file or directory
locale: Cannot set LC_ALL to default locale: No such file or directory
Selecting previously unselected package i2c-tools.
(Reading database ... 120672 files and directories currently installed.)
Preparing to unpack .../i2c-tools_3.1.1+svn-2_armhf.deb ...
Unpacking i2c-tools (3.1.1+svn-2) ...
Selecting previously unselected package python-smbus.
Preparing to unpack .../python-smbus_3.1.1+svn-2_armhf.deb ...
Unpacking python-smbus (3.1.1+svn-2) ...
Processing triggers for man-db (2.7.0.2-5) ...
Setting up i2c-tools (3.1.1+svn-2) ...
/run/udev or .udevdb or .udev presence implies active udev.  Aborting MAKEDEV invocation.
Setting up python-smbus (3.1.1+svn-2) ...
```

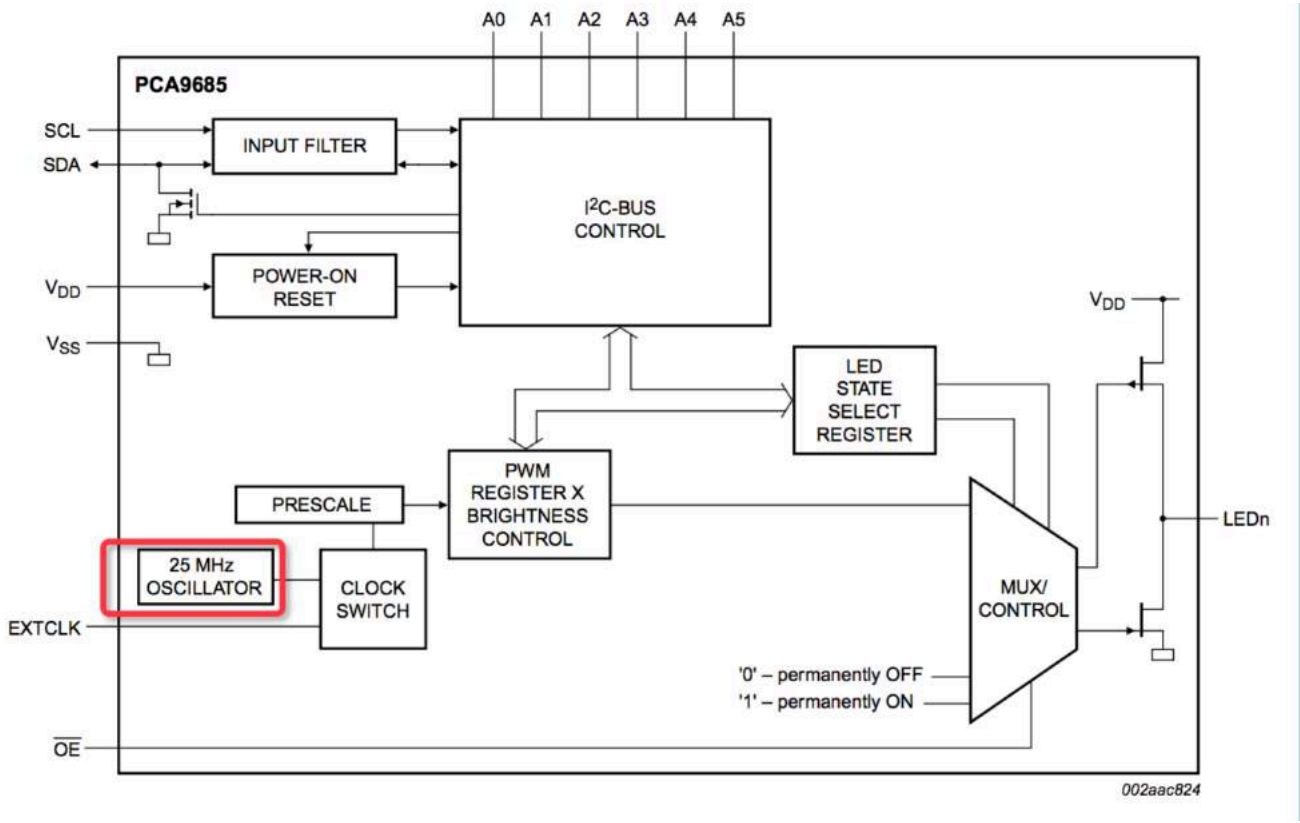
Then you can use i2cdetect command to detect i2c device that recognized by your system.

i2cdetect -y 1

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```

You will find that pca9685a driver board has been recognized as address 0x70.

5. Check out the NXP's PCA8956 datasheet and find out the clock rate of chip as following picture:

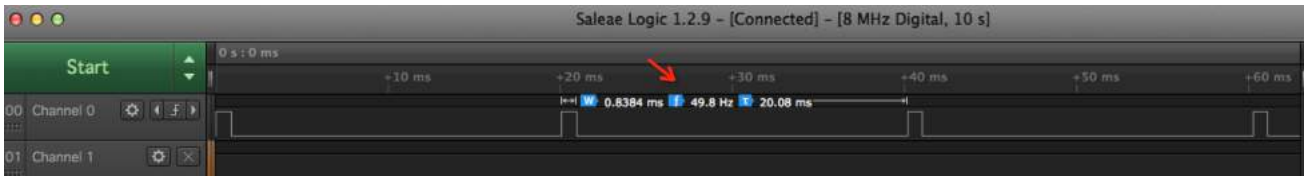


Refer: https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf

6. **Setup pwm parameters by following commands:**

```
sudo su -
cd /sys/class/pwm/pwmchip0/
echo 0 > export
echo 24000000 > /sys/class/pwm/pwmchip0/pwm0/period
echo 1 > enable
```

7. **Using logical analyzer to detect the period of PWM pulse as picture shown:**



8. **Write a shell script to control your servo:**

```
sudo vim.tiny myservo.sh
```

```
#!/bin/bash
# function: drive the servo
# Edit by yoyojacky
# Date: 2016/9/18
while true
do
    for i in 523500 1023500 1523500 2023500 2623500;
    do
        echo $i    > duty_cycle
        sleep 0.2
    done
done
```

save it.

```
sudo chmod +x myservo.sh
sudo ./myservo.sh
```

You will find that your servo is working properly, thank you!

PS: How to calculate the number that in the script file?

Here are the angles corresponding to PWM Pulses in the date sheet

0.5ms ----- 0 degree

1.0ms ----- 45 degree

1.5ms ----- 90 degree

2.0ms ----- 135 degree

2.5ms ----- 180 degree

f=50mz, T=20ms

so, you can follow this formula to calculate:

Y= X* 21300000/20ms

X	Y	Degree
0.5ms	523500	0 degree
1.0ms	1023500	45 degree
1.5ms	1523500	90 degree
2.0ms	2023500	135 degree
2.5ms	2623500	180 degree