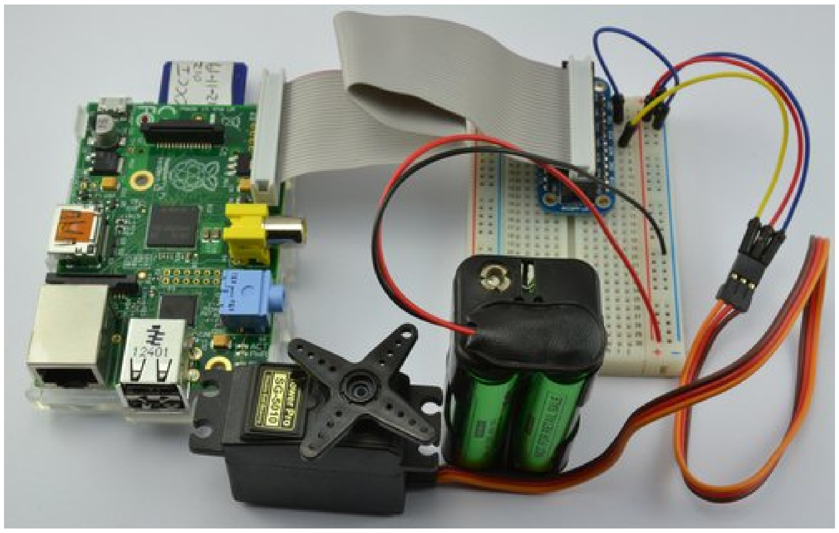
Adafruit's Raspberry Pi Lesson 8. Using a Servo Motor

Created by Simon Monk

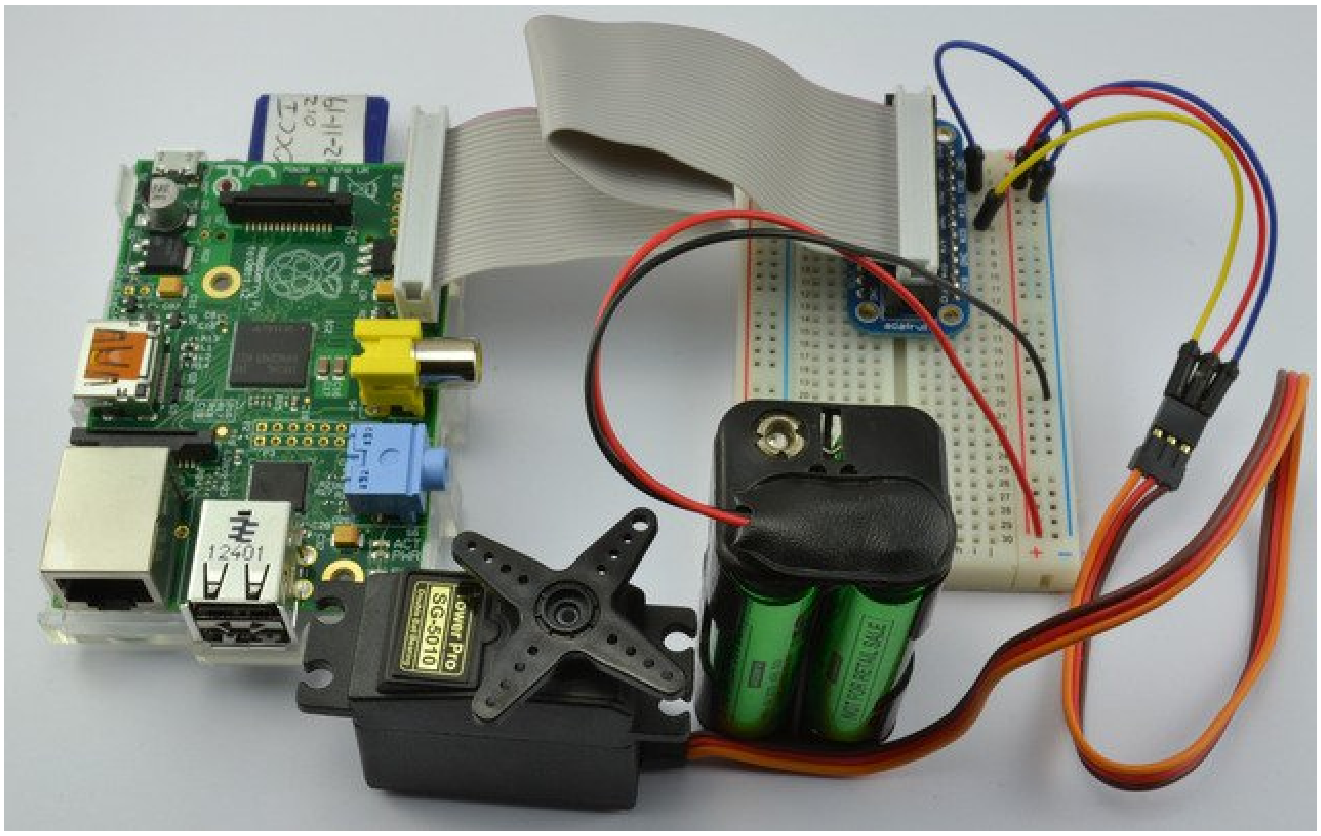


Last updated on 2016-11-03 06:17:53 AM UTC

Guide Contents

Overview

This lesson describes how to control a single servo motor using Python.



Servo motors are controlled by pulses of varying lengths. This requires fairly accurate timing. The Raspberry Pi has one pin that generates pulses in hardware, without having to rely on the operating system.

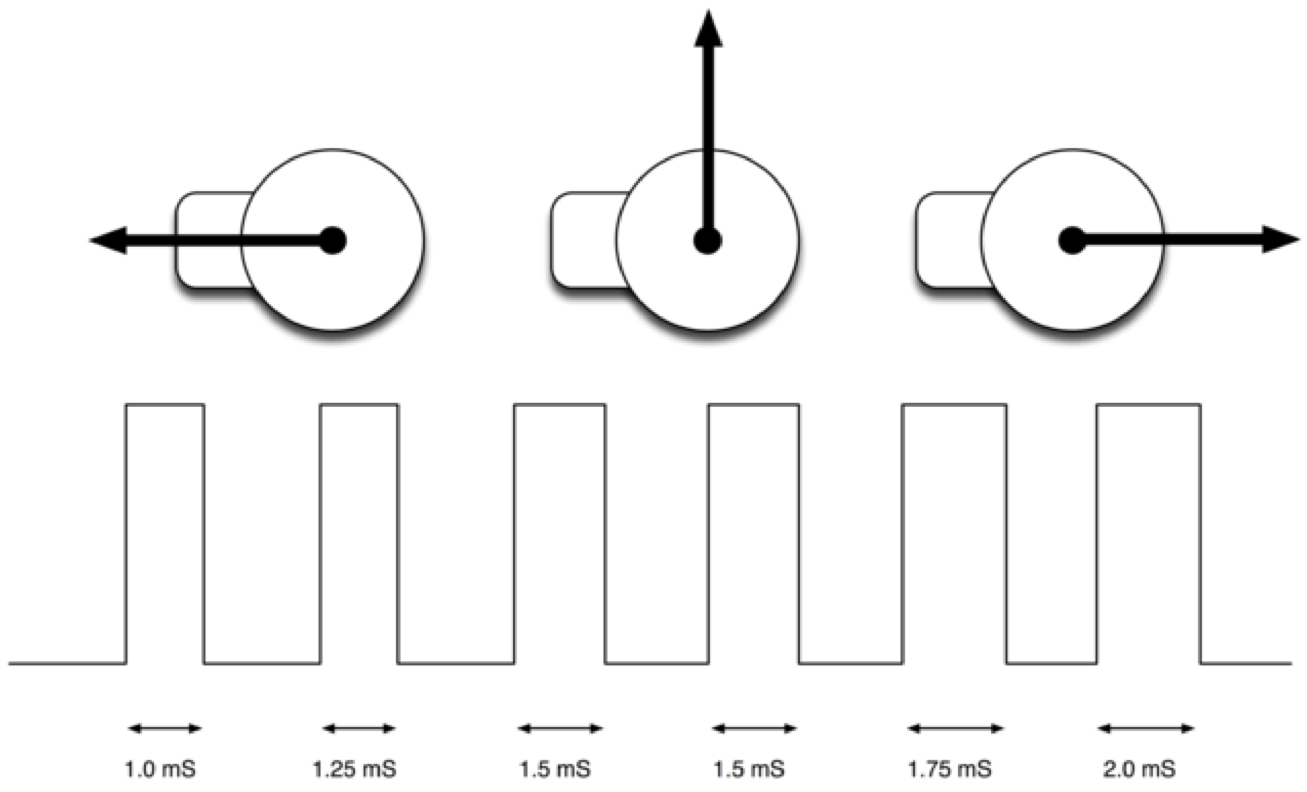
Parts

To build this project, you will need the following parts.

	Part	Qty
	Raspberry Pi	1
	Cobbler Breakout Board and IDC cable	1
	Set of male to male jumper leads	1
	Half-size breadboard	1
	Servo	1
	4 x AA or AAA battery holder and batteries.	1

Servo Motors

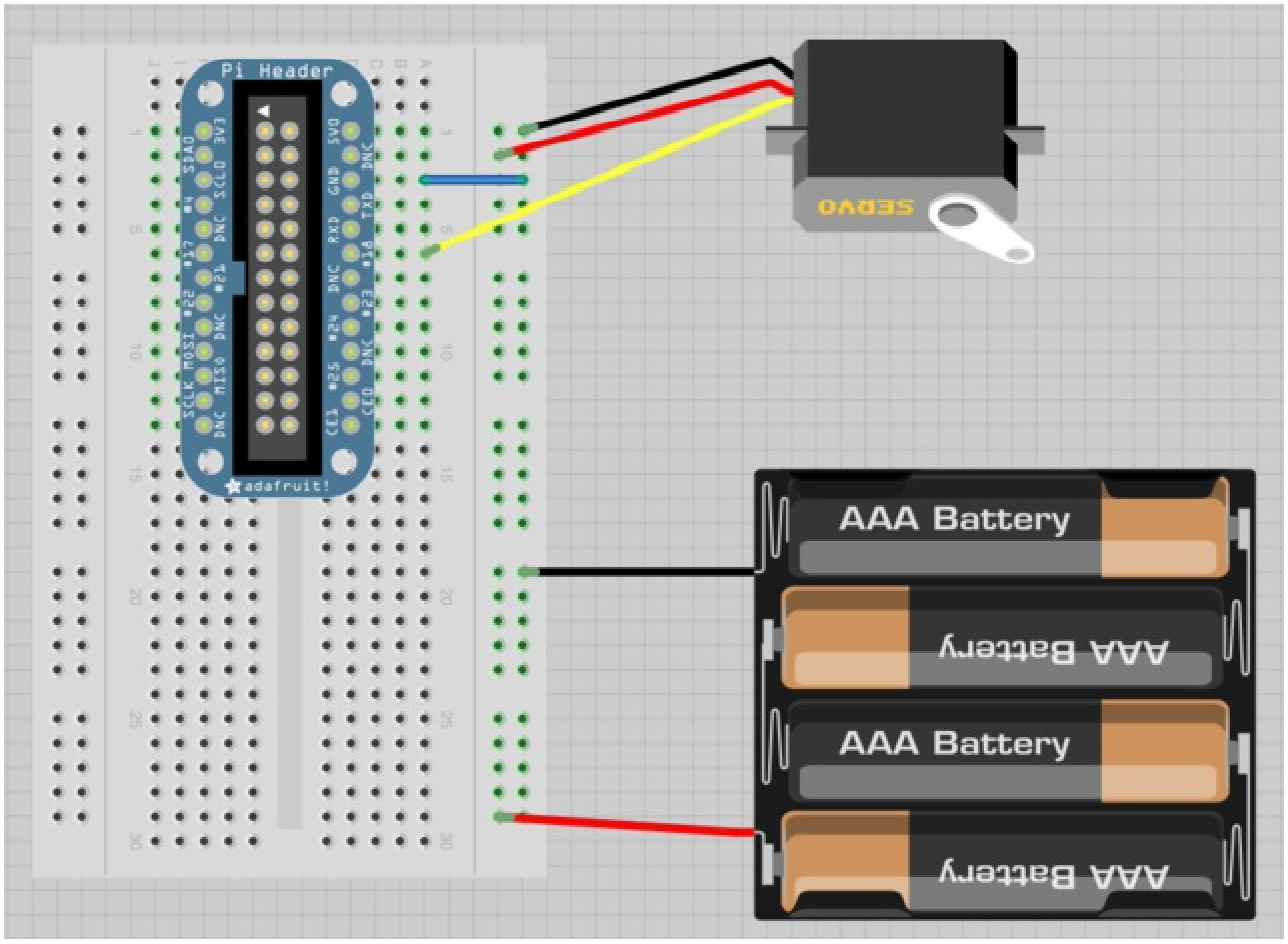
The position of the servo motor is set by the length of a pulse. The servo expects to receive a pulse roughly every 20 milliseconds. If that pulse is high for 1 millisecond, then the servo angle will be zero, if it is 1.5 milliseconds, then it will be at its centre position and if it is 2 milliseconds it will be at 180 degrees.



The end points of the servo can vary and many servos only turn through about 170 degrees. You can also buy 'continuous' servos that can rotate through the full 360 degrees.

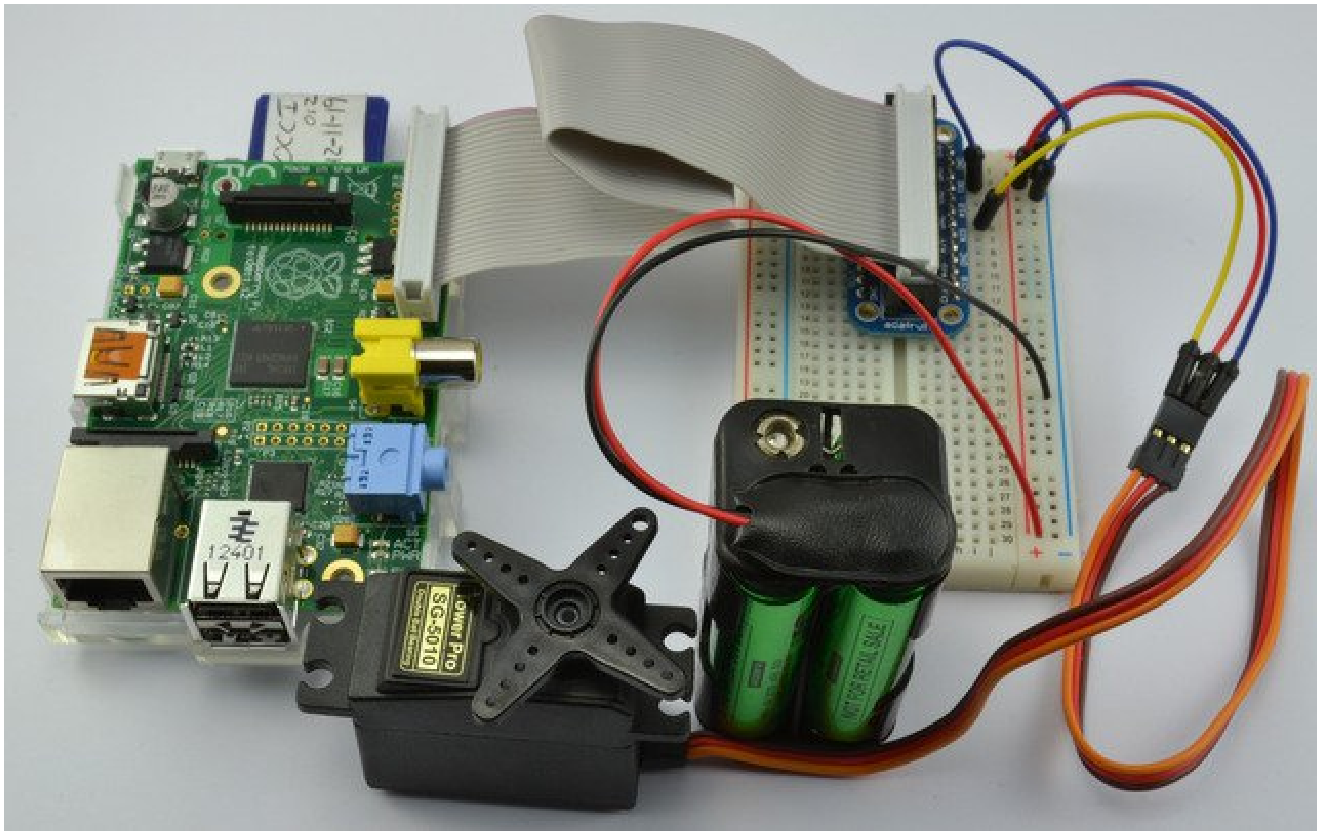
Hardware

There is only one pin on the Pi that is capable of producing pulses in this way (GPIO pin 18). This will be connected to the control pin of the servo. The power to the servo is provided by an external battery as powering the servo from the Pi itself is likely to cause it to crash as the Servo draws too much current as it starts to move. Servos require 4.8-6V DC power to the motor, but the signal level (pulse output) can be 3.3V, which is how its OK to just connect the signal line directly to the GPIO output of the Pi.



The Pi Cobbler is used to link the Raspberry Pi to the breadboard. If you have not used the Cobbler before take a look at Lesson 4 (http://adafru.it/aTH) in this series.

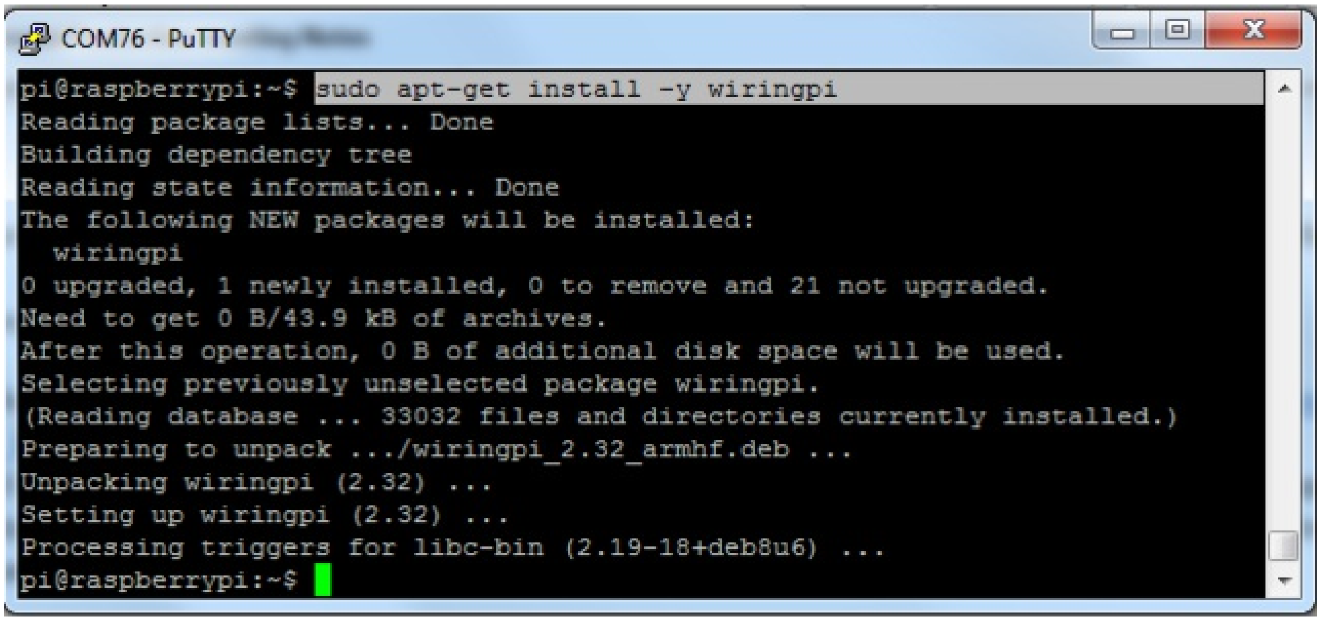
Servo motors generally come with three pin sockets attached. The red and brown sockets supply power (positive to red) and the third yellow or orange socket is for the control signal. To link the socket to the breadboard, use the male-to-male jumper wires.



Software

We'll be using the great **wiringPi** tool **gpio** to control the servo. Begin by installing **gpio** with

```
sudo apt-get install -y wiringpi
```


COM76 - PuTTY

```
pi@raspberrypi:~$ sudo apt-get install -y wiringpi
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  wiringpi
0 upgraded, 1 newly installed, 0 to remove and 21 not upgraded.
Need to get 0 B/43.9 kB of archives.
After this operation, 0 B of additional disk space will be used.
Selecting previously unselected package wiringpi.
(Reading database ... 33032 files and directories currently installed.)
Preparing to unpack .../wiringpi_2.32_armhf.deb ...
Unpacking wiringpi (2.32) ...
Setting up wiringpi (2.32) ...
Processing triggers for libc-bin (2.19-18+deb8u6) ...
pi@raspberrypi:~$
```

Set pin #18 to be a PWM output

```
gpio -g mode 18 pwm
```

Pin #18 has PWM output, but you have to set it to be the right *frequency* output. Servo's want 50 Hz frequency output

For the Raspberry Pi PWM module, the PWM Frequency in Hz = 19,200,000 Hz / pwmClock / pwmRange

If pwmClock is 192 and pwmRange is 2000 we'll get the PWM frequency = 50 Hz (thx to kev for the numbers! (http://adafru.it/sdj))

Now you can tell **gpio** to set the PWM clock to those numbers:

```
gpio pwm-ms
gpio pwmc 192
gpio pwmr 2000
```

Now you can set the servo to all the way to the left (1.0 milliseconds) with

```
gpio -g pwm 18 100
```

Set the servo to the middle (1.5 ms) with

```
gpio -g pwm 18 150
```

And all the way to the right (2.0ms) with

```
gpio -g pwm 18 200
```

Servos often 'respond' to a wider range than 1.0-2.0 milliseconds so try it with ranges of 50 (0.5ms) to 250 (2.5ms)

Of course you can try any number between 50 and 250! so you get a range of about 200 positions

Note that the Raspberry Pi PWM is not necessarily a 'stable' output, and there might be some jitter! For a steady PWM signal, you'll want to check out a dedicated Servo HAT. (http://adafru.it/2327)

Python Example!

You can also use wiringPi in python! Run

```
sudo apt-get install -y python-pip

sudo pip install wiringpi
```

The Python program to make the servo sweep back and forth is listed below:

```
# Servo Control
import time
import wiringpi

# use 'GPIO naming'
wiringpi.wiringPiSetupGpio()

# set #18 to be a PWM output
wiringpi.pinMode(18, wiringpi.GPIO.PWM_OUTPUT)

# set the PWM mode to milliseconds stype
wiringpi.pwmSetMode(wiringpi.GPIO.PWM_MODE_MS)

# divide down clock
wiringpi.pwmSetClock(192)
wiringpi.pwmSetRange(2000)

delay_period = 0.01
```

```
while True:
    for pulse in range(50, 250, 1):
        wiringpi.pwmWrite(18, pulse)
        time.sleep(delay_period)
    for pulse in range(250, 50, -1):
        wiringpi.pwmWrite(18, pulse)
        time.sleep(delay_period)
```

We're basically using the wiringpi functions but*within* python, so its easy to practice with the **gpio** utility and then use the matching commands in python

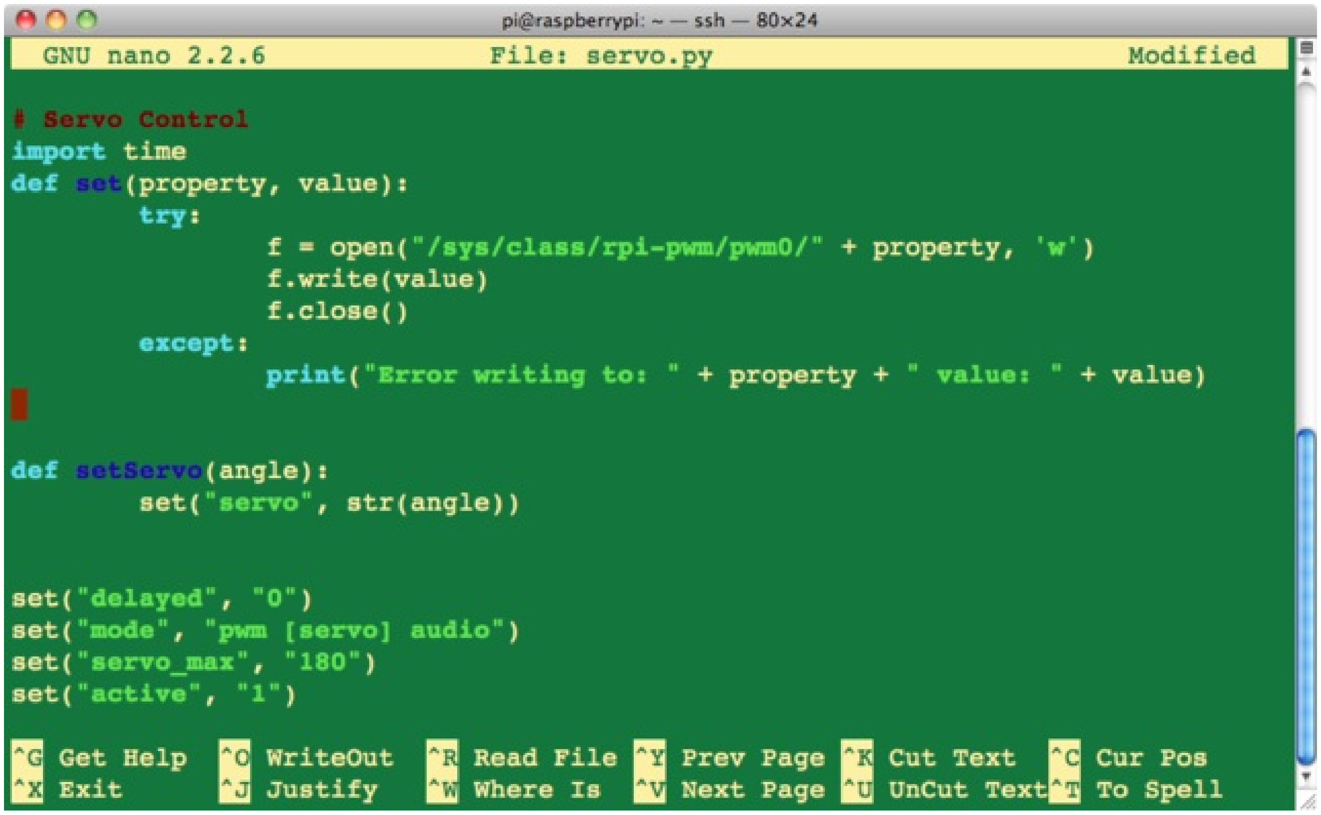
A variable (delay_period) is used to contain the time in seconds between each step of the servo.

The while loop will just continue forever or until the program is interrupted by pressing CTRL-C. Within the loop there are two near identical 'for' loops. The first counts the pulse width up from 5.0ms to 2.5ms and the second sets the pulse with starting with 2.5m and counts down to 0.5ms, moving the servo arm back and forth.

To install the software, connect to your Pi using SSH and then type the command:

```
$ nano servo.py
```

Paste the code above into the editor and then do CTRL-X and Y to save the file.



```
  GNU nano 2.2.6                File: servo.py                          Modified

# Servo Control
import time
def set(property, value):
        try:
                f = open("/sys/class/rpi-pwm/pwm0/" + property, 'w')
                f.write(value)
                f.close()
        except:
                print("Error writing to: " + property + " value: " + value)


def setServo(angle):
        set("servo", str(angle))


set("delayed", "0")
set("mode", "pwm [servo] audio")
set("servo_max", "180")
set("active", "1")

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

To run the servo program just type the following command into your SSH window:

```
$ sudo python servo.py
```

The servo should start to move straight away.

Test & Configure

If you want to make the servo move faster, try changing delay_period to a smaller value, say 0.001. Then to slow it down try increasing it to 0.1.

If you want to control more than one servo, then the easiest way to do this is to use something like the Adafruit I2C 16 channel servo / pwm controller (http://adafru.it/815). This tutorial (http://adafru.it/aPn) explains how to use it.