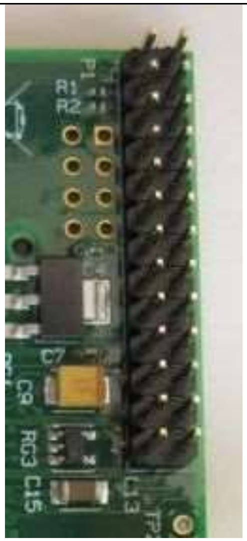
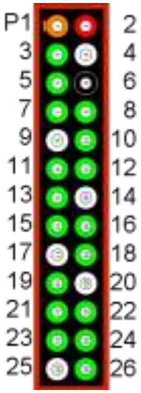
Using the Raspberry Pi GPIO connector pins

General Purpose Input Output

Dr James Dalley
dl@richardhale.co.uk

FINDING THE GPIO CONNECTOR PINS 1

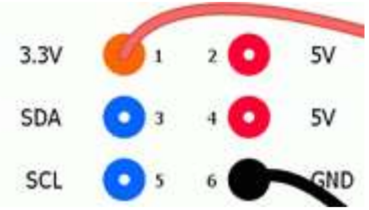




1. Find the GPIO pins.

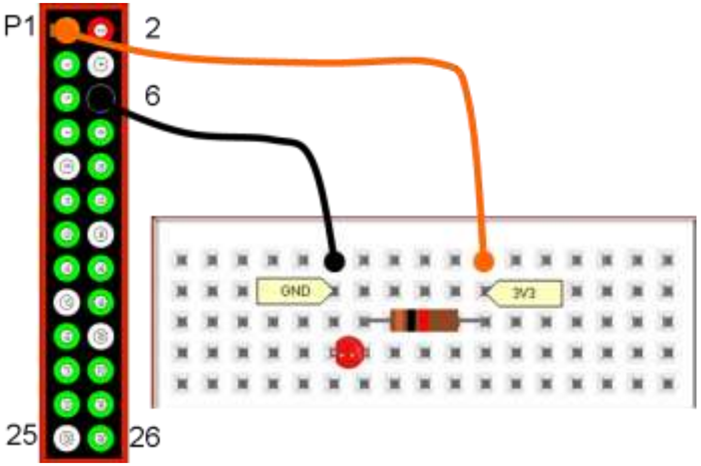
2. Look for the white P1 label for pin 1 (3.3V).

3. Find pin 6 which is ground (GND).



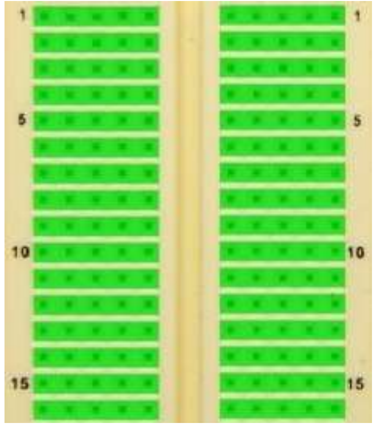
TESTING THAT THE LED WORKS! 2

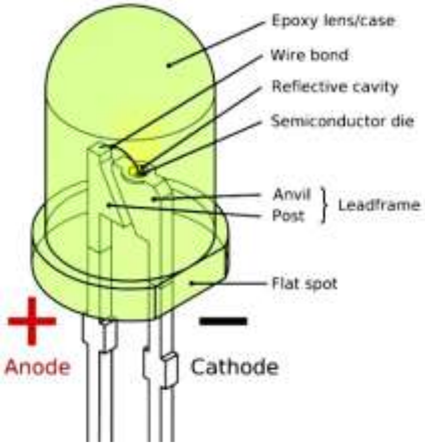
Make this circuit (ensure the LED flat spot is on the same side as the GND lead).



BREADBOARD CONNECTIONS & LED 3

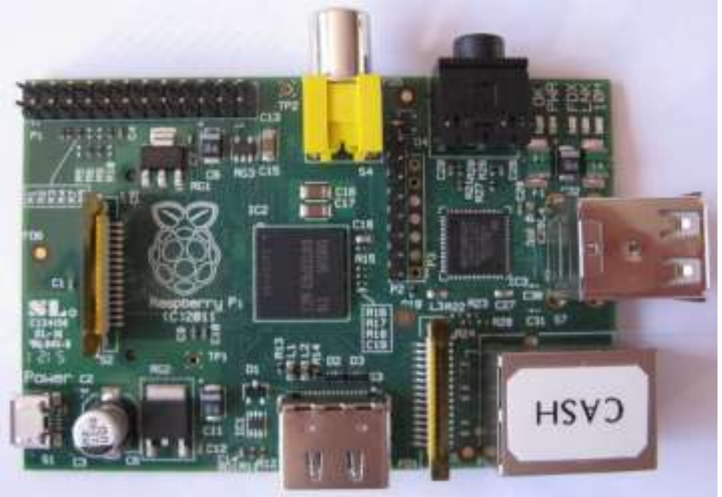
Holes in the same coloured bar are connected.





LED: + side has long lead,
- side has short lead and flat spot.

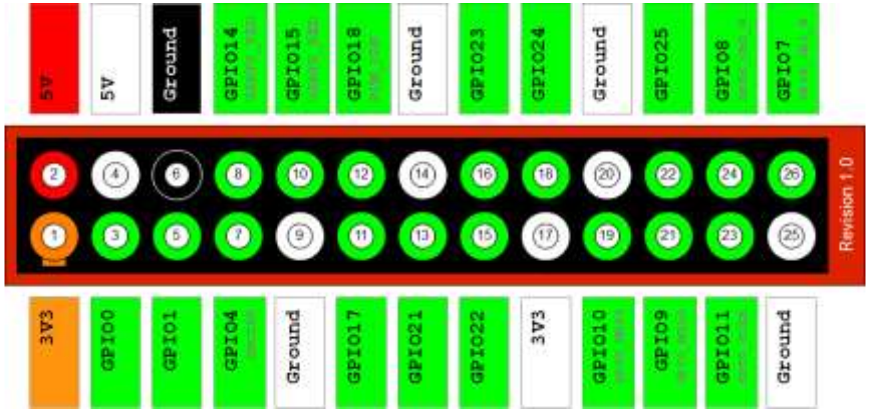
REVISION 1 RASPBERRY PI



http://www.raspberrypi-spy.co.uk/2012/09/raspberry-pi-board-revisions/

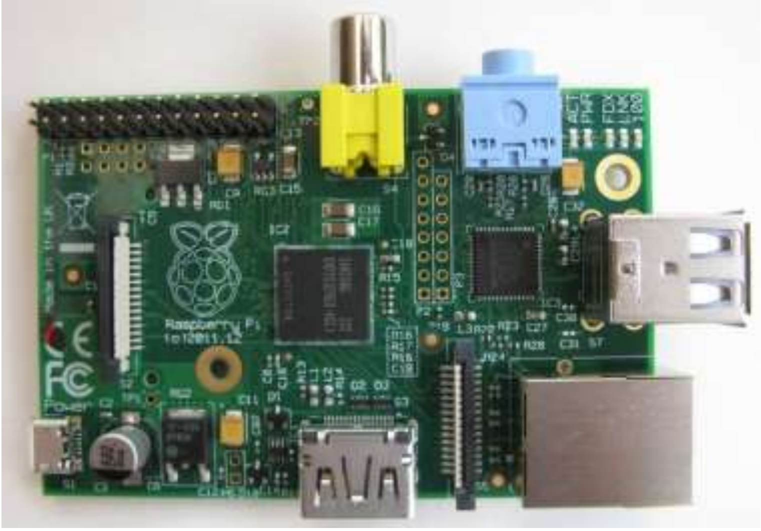
REVISION 1 GPIO CONNECTOR MAP

The GPIO connector pin numbers are shown within the black box:



http://www.raspberrypi-spy.co.uk/2012/09/raspberry-pi-board-revisions/

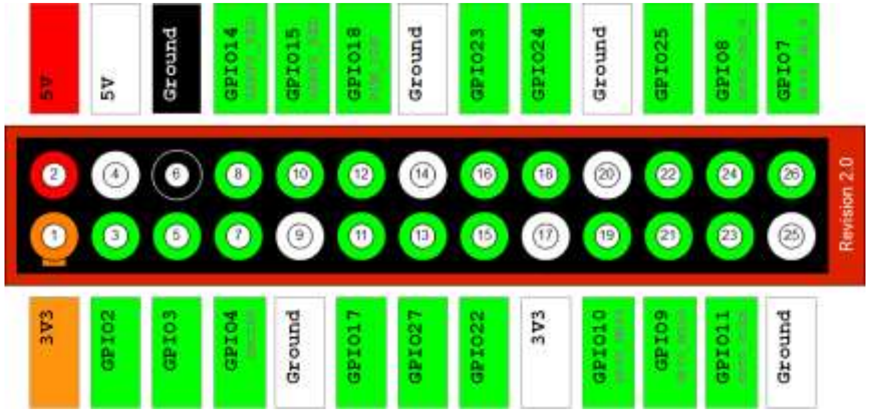
REVISION 2 RASPBERRY PI



http://www.raspberrypi-spy.co.uk/2012/09/raspberry-pi-board-revisions/

REVISION 2 GPIO CONNECTOR MAP

The GPIO connector pin numbers are shown within the black box:



http://www.raspberrypi-spy.co.uk/2012/09/raspberry-pi-board-revisions/

GOOD PRACTICE… 8

1. Try to connect to pins 1, 2 & 6 for 3.3V, 5V and Ground (0V). If you use pins 4, 9, 14, 17, 20 or 25 and have problems then go back to using 1, 2 & 6.

2. Always connect the LED flat (-ve end) to ground. It makes it much easier to spot if it is the wrong way round.

3. Raspberry Pi uses 3.3V logic, so input circuits should be connected to 3V3 (pin 1) and not the 5V pin 2.

UNDERSTANDING THE GPIO PINS 9

If we want to make the LED flash using a particular GPIO connector pin, we have to understand how that pin is controlled.

1. Using flashcards 6 or 7, find pin 22 on the GPIO connector pins within the black box.
2. Each GPIO connector pin is controlled using a ‘signal name’. Use the connector map to find the signal name for connector pin 22 (GPIO25).
3. ALWAYS keep the GPIO connector map to hand as the signal name is very unobvious.

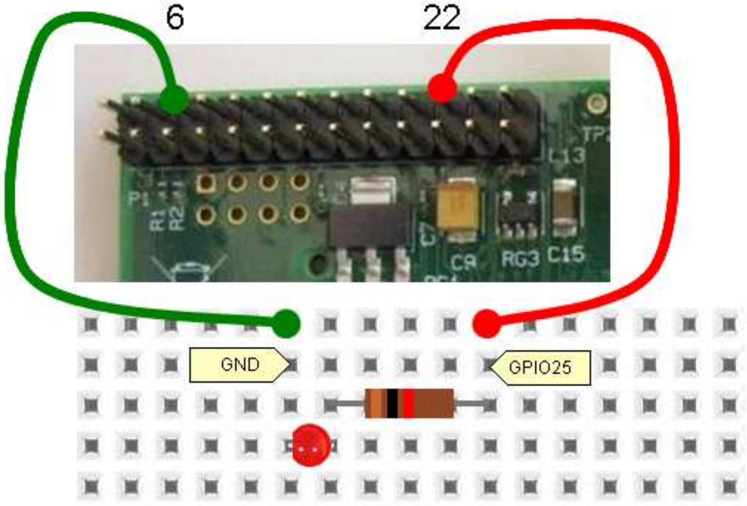
CONNECTOR PIN 22: FLASHING LED 10

```
# import libraries
import RPi.GPIO as GPIO
import time

# set up a pin/signal name as an output
GPIO.setmode(GPIO.BCM) # use BCM signalnames
GPIO.setup(25,GPIO.OUT)# -> connector pin 22

while True:               # loop forever
     GPIO.output(25,True) # conn pin 22 high
     time.sleep(1.25)     # wait 1.25s
     GPIO.output(25,False)# conn pin 22 low
     time.sleep(0.25)     # wait 0.25s
```

CONNECTOR PIN 22: FLASHING LED 11



INPUT HI, OUTPUT HI PROGRAM 12

```
# import library
import RPi.GPIO as GPIO

# set up pin/signal name inputs & outputs
GPIO.setmode(GPIO.BCM) # use BCM signalnames
GPIO.setup(7,GPIO.OUT) # conn pin26 is output
GPIO.setup(4,GPIO.IN)  # conn pin7 is input

while True:
     if GPIO.input(4)==True:  # pin7:input=hi
          GPIO.output(7,True) # pin26:high

     if GPIO.input(4)==False: # pin7:input=lo
          GPIO.output(7,False)# pin26:lo
```

INPUT LO, OUTPUT HI PROGRAM 13

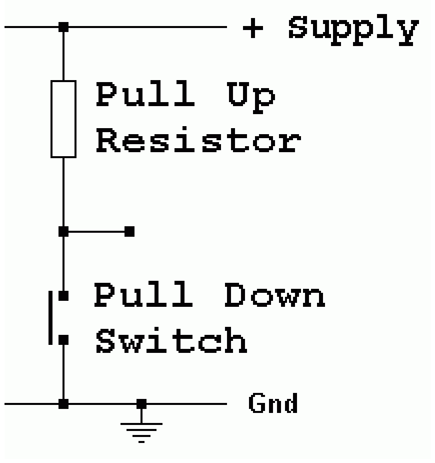
```
# import library
import RPi.GPIO as GPIO

# set up pin/signal name inputs & outputs
GPIO.setmode(GPIO.BCM) # use BCM signalnames
GPIO.setup(7,GPIO.OUT) # conn pin26 is output
GPIO.setup(4,GPIO.IN)  # conn pin7 is input

while True:
     if GPIO.input(4)==True:  # pin7:input=hi
          GPIO.output(7,False)# pin26:lo

     if GPIO.input(4)==False: # pin7:input=lo
          GPIO.output(7,True) # pin26:high
```

SWITCH INPUT CIRCUIT INFO 14

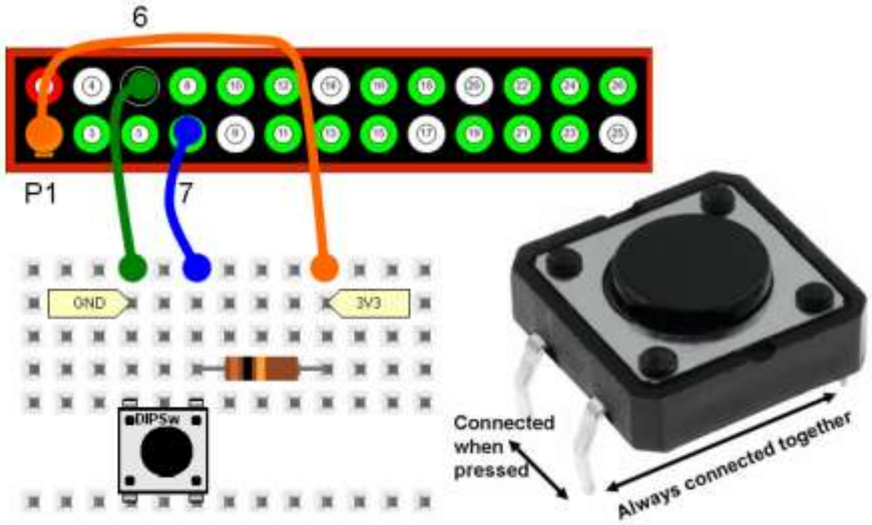


Use a 10kΩ pull-up resistor to make a 3.3V input switch. 10kΩ=BrownBlackOrange.

Attach GPIO connector pin 7 to the point between the switch and resistor.

Remember that GPIO.setup(4,GPIO.IN) sets connector pin7 as an input

SWITCH INPUT CIRCUIT (PIN 7) 15



SERVO OUTPUT CIRCUIT INFO 16

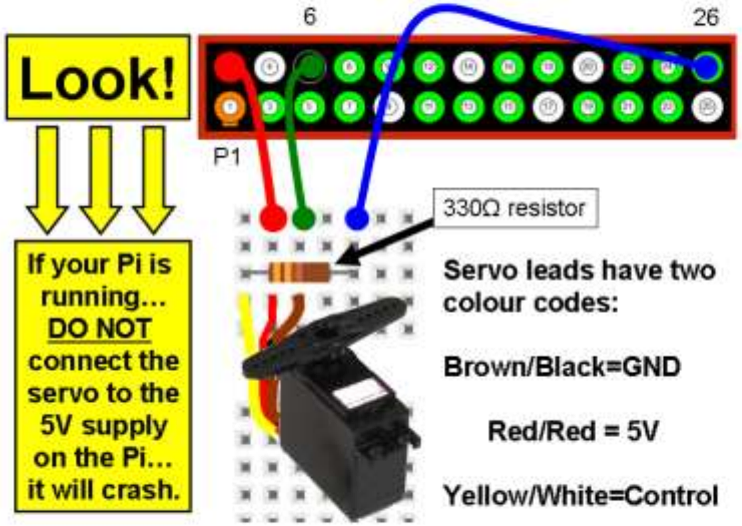
Servos often need **external power** and a 330Ω resistor in line with the control wire

Servos rotate about a 270° degree range. To control them they need a pulse every 20ms (0.02s).

The pulse width should be 0.75ms to 2.25ms. The wider the pulse, the more it rotates.

Give the servo time to get to its position before you change it.

SERVO OUTPUT CIRCUIT (PIN 26) 17



If your Pi is running... DO NOT connect the servo to the 5V supply on the Pi... it will crash.

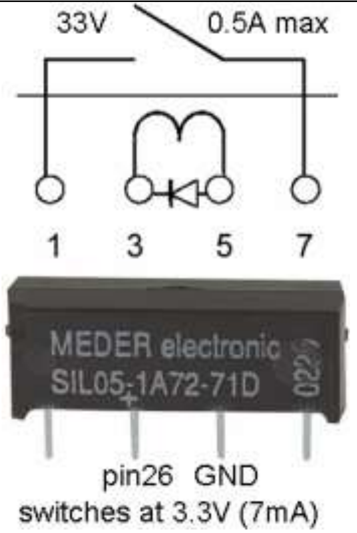
Servo leads have two colour codes:

Brown/Black=GND

Red/Red = 5V

Yellow/White=Control

RELAY OUTPUT CIRCUIT INFO 18

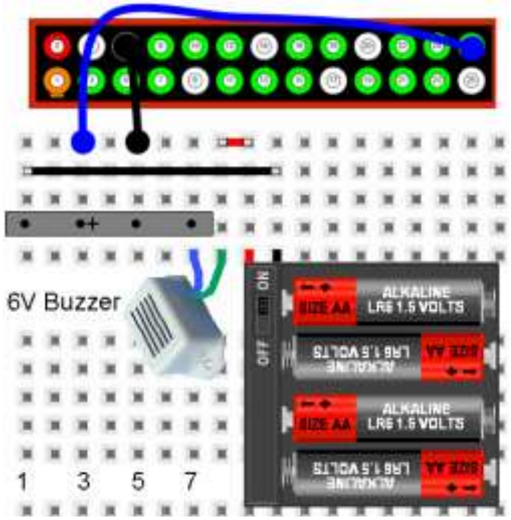


A relay allows a small current flowing through the middle 2 pins to connect a switch between the outer 2 pins.

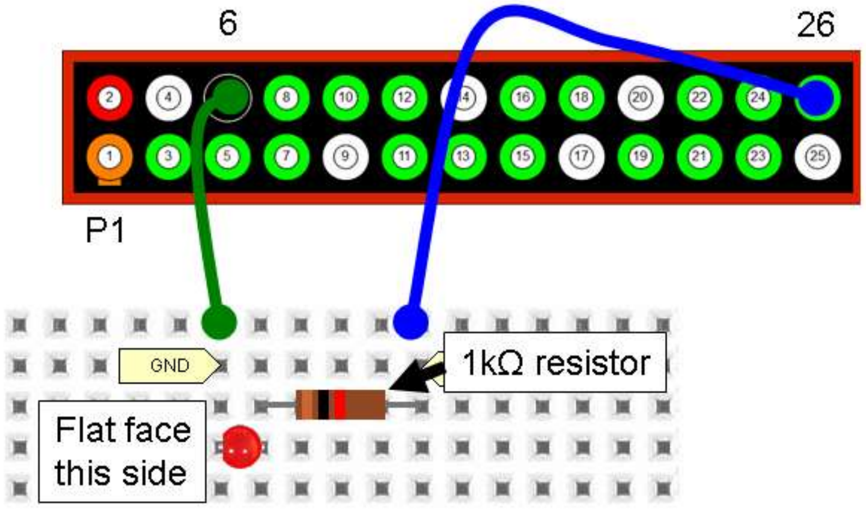
This is a great way to control circuits you have made with the Pi. The circuit you attach to the outer pins must have its own power supply to make it work.

RELAY OUTPUT CIRCUIT (PIN 26) 19

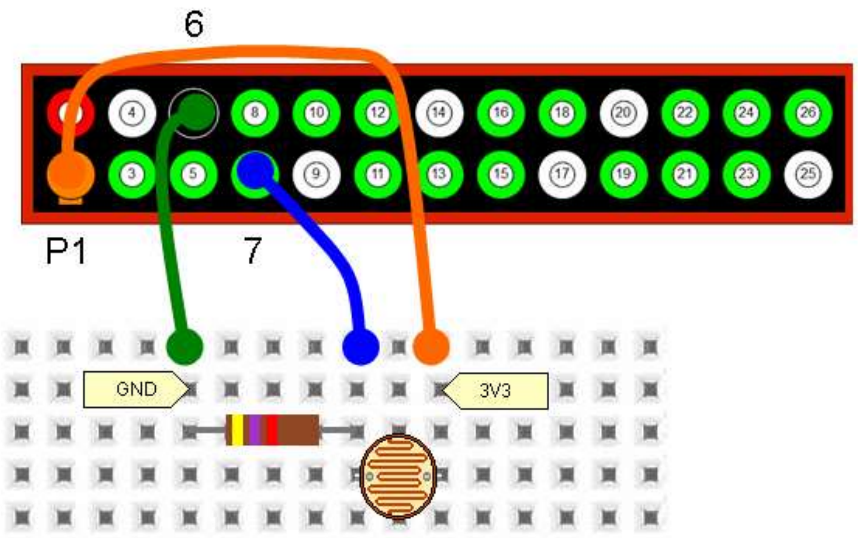
A 3.3V signal (from pin 26) is used to switch a higher voltage, higher current buzzer circuit. Top of relay→→→



LED OUTPUT CIRCUIT (PIN 26) 20

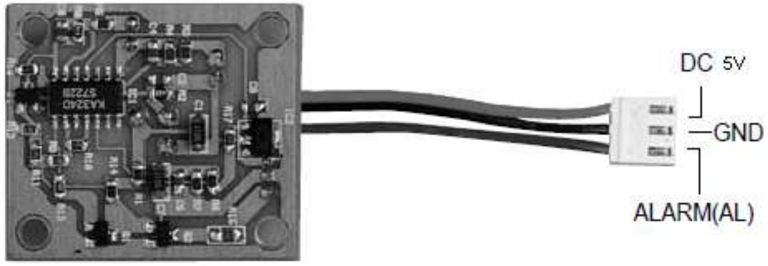


LDR INPUT CIRCUIT (PIN 7) 21



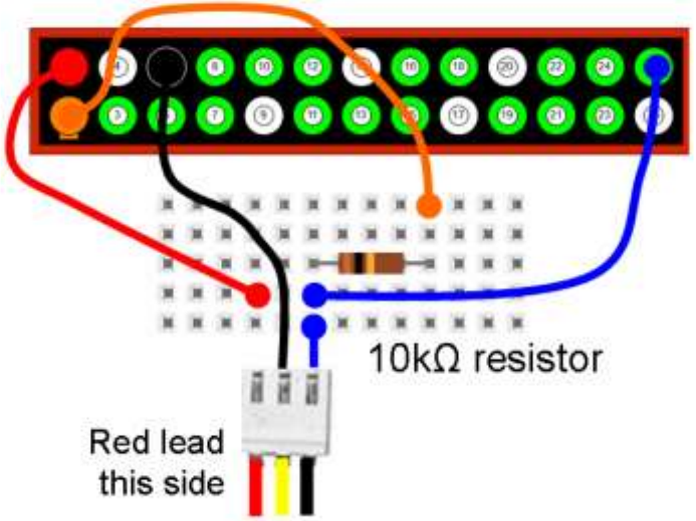
PIR INPUT CIRCUIT INFO 22

The PIR unit detects infra-red light from living objects. If an object is detected it acts like a pull-down switch and connects the ALARM wire to GND. Use a 10kΩ pull-up resistor (to 3.3V) on the alarm wire.



http://www.skpang.co.uk/catalog/pir-motion-sensor-p-796.html

PIR INPUT CIRCUIT (PIN 7) 23



```
#Version 3.2: Frank Everest, David Whale, James Dalley
# December 2013

#Basic servo demo code for GPIO connector pin 26 (signal name GPIO7)
#
# >>>>>>> Do NOT power-up the servo using the Pi's 5V  <<<<<<<<<<<
# >>>>>>>      if you are looking at this code!        <<<<<<<<<<<
#
# Shut down the Pi, take out the power lead, connect the servo
# check the wiring, recheck, then reboot the Pi, run and enjoy/adapt.

#import libraries
import RPi.GPIO as GPIO
import time

#initialise the ports
GPIO.setmode(GPIO.BCM)  #Use the BCM signal names
GPIO.setup(7,GPIO.OUT)  #GPIO connector pin 26 (signal name GPIO7)

#Servo parameters
# Short pulse: servo swings one way, long pulse: swings other way.
# The pulse width determines the servo's final position.
# Must repeat every 20ms
SHORTEST = 0.001    # 1ms = 0.001s
LONGEST = 0.00225   # 2.25ms = 0.00225s
PERIOD = 0.02       # 20ms = 0.02s

#Function to send one pulse to the servo
def PWM(ON, OFF):
    #(ON + OFF) must be = the repetiion rate: 20ms for this demo
    GPIO.output(7,True)
    time.sleep(ON)          # pulse width determines final position
    GPIO.output(7,False)
    time.sleep(OFF)         # makes up the remaining 20ms

#Function to send 20 pulses to the servo
# This gives servo time to get to get into position)
# Servo will jitter if you give it too little time
def send_command(PULSE):
    #PULSE should be in the range 0.001 to 0.00225 seconds
    for pulseNumber in range(20):   # can increase/decrease from 20
        PWM(PULSE,PERIOD-PULSE)

#Main code
# Makes servo swing one way, wait, then the other
# Press Ctrl+c on keyboard to stop the code and exit properly
try:
    while True:                 # loop forever
        send_command(LONGEST)   # swing one way (to a position)
        time.sleep(1)           # wait one sec
        send_command(SHORTEST)  # swing other way (to new position)
        time.sleep(1)           # wait one sec
except:
        GPIO.cleanup()          # tidy-up the GPIO ports before exiting
```