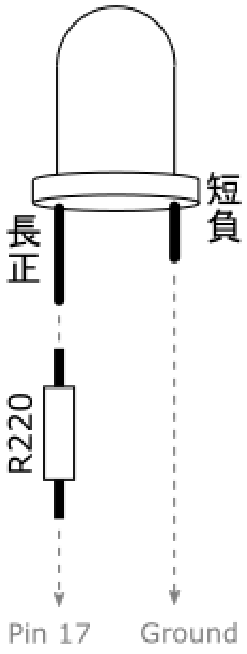
Raspberry Pi

GPIO and Camera

Author: Nick Lee

```
$ python3

>>> 1 + 3
>>> 9 - 7
>>> 6 * 3
>>> 3 / 2
>>> 3 // 2
>>> 7 % 3
>>> 4 + 2 * 3
>>> (4 + 2) * 3
>>> 2 ** 3

>>> type(3)
>>> type(3.0)
>>> type('3')

>>> 3.0 == 3
>>> '3' == 3

>>> # comment

>>> x = 1
>>> y = 2
>>> x + y
>>> x * y

>>> import time
>>> time.time()

>>> x = 25
>>> 'I am {} years old'.format(x)

>>> street = 'Nathan Road'
>>> area = 'Mongkok'
>>> 'I live on {} {}, {}'.format(x, street, area)

>>> humidity, temperature = 50.00001, 22.34567
>>> 'Today is {}% {:.2f} degree'.format(humidity, temperature)
```

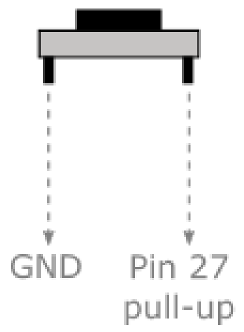
LED

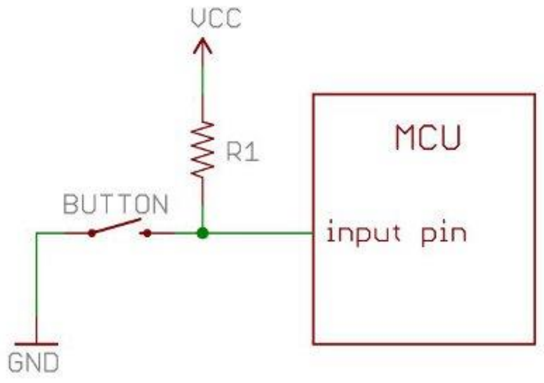


gpiozero basic recipes

- LED
- LED with variable brightness
- Traffic Lights

Button





gpiozero basic recipes

- Button

Quick reaction game

Now that you know how to handle LED and push button, let's check how quick your reaction is. Make a reaction timer as follows:

1. Turn on LED
2. User presses button to "start the clock". LED is turned off.
3. After a random number of seconds (say, 2-10 seconds), LED is turned back on!
4. User has to press button as soon as he can. Print out his reaction time.

reaction.py

```
from gpiozero import LED, Button
import time
import random

led = LED(26)
button = Button(12)

led.on()
button.wait_for_press()  ❶

led.off()
time.sleep(random.uniform(2, 10))  ❷

led.on()
on_time = time.time()  ❸

button.wait_for_press()  ❹
press_time = time.time()  ❺

print('{:.3f}'.format(press_time - on_time))
```

❶ Wait for user to get ready

❷ Delay for a random number of seconds (2-10)

❸ Remember the time when LED is turned on

❹ Wait for user to press

❺ Remember the time when user presses the button

愚蠢的交通燈

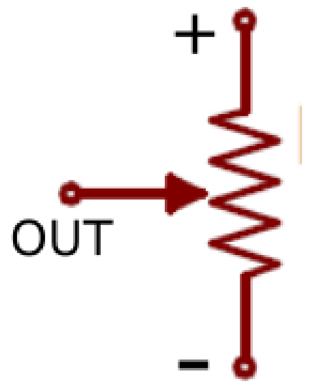
```
from gpiozero import TrafficLights, Button
from time

lights = TrafficLights(26, 19, 13)

while 1:
    lights.value = (1, 0, 0)
    sleep(1)

    lights.value = (0, 1, 0)
    sleep(1)

    lights.value = (0, 0, 1)
    sleep(1)
```

改良

1.模擬真正交通燈：紅、紅黃、綠、黃、紅

2.以按掣轉燈

Potentiometer
VR, Variable Resistor



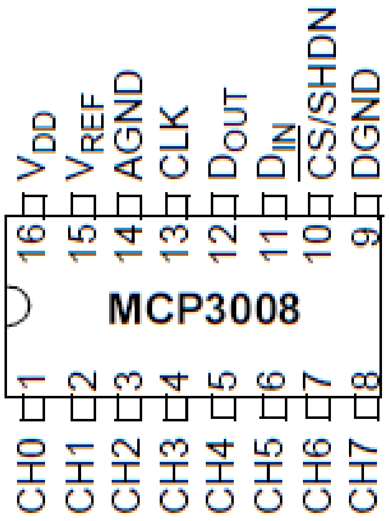
gpiozero basic recipes

- Potentiometer

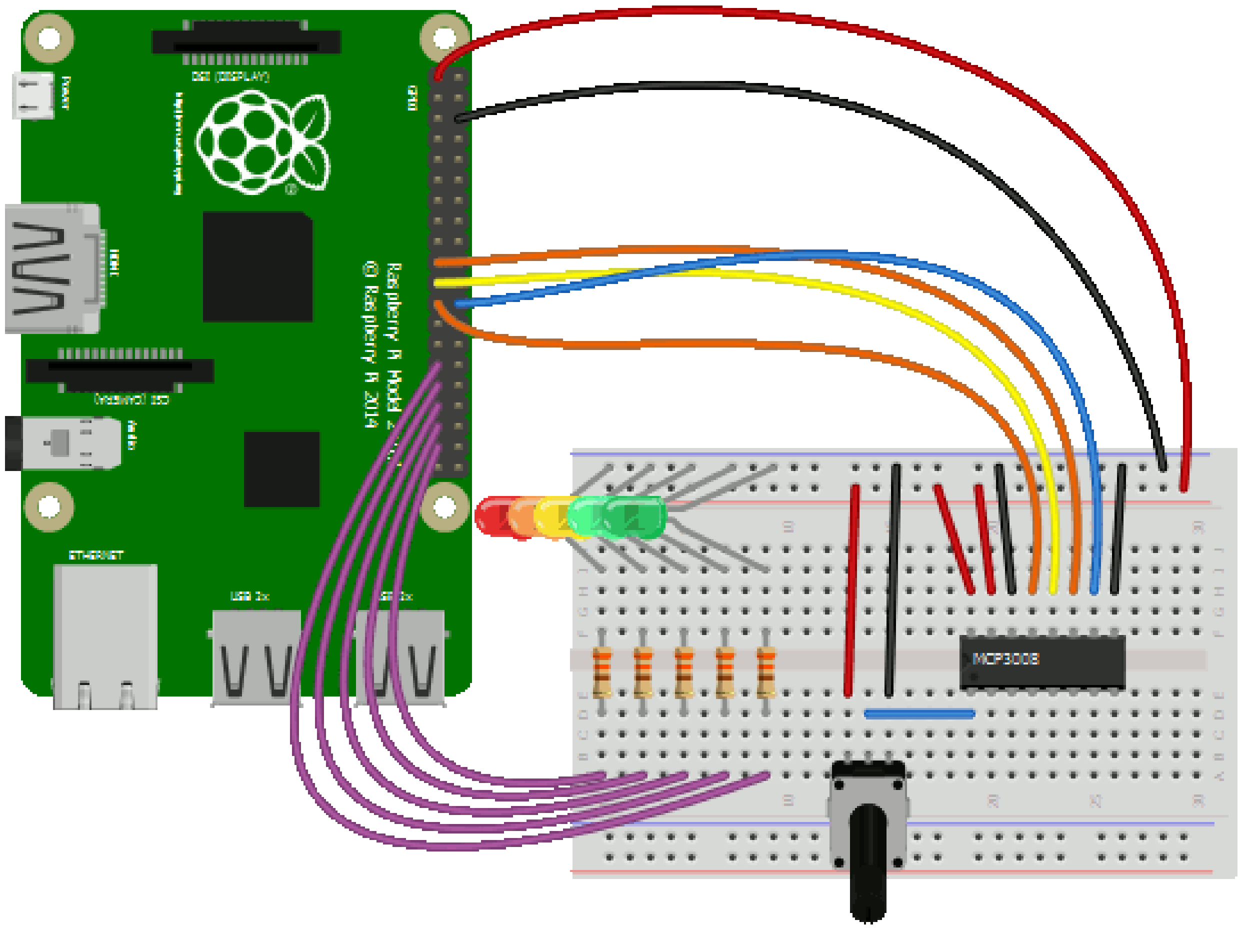
Problem

- Use a potentiometer to vary an LED's brightness

Joystick 等於兩個 VR，一個 X 軸，一個 Y 軸。



MCP3008
VDD
VREF
AGND
CLK
DOUT
DIN
CS/SHDN
DGND
16
15
14
13
12
11
10
9
1
2
3
4
5
6
7
8
CH0
CH1
CH2
CH3
CH4
CH5
CH6
CH7



MCP3008
© Raspberry Pi 2014

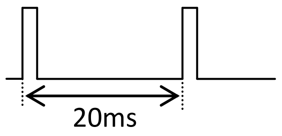
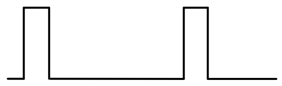
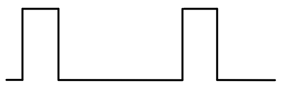
Servo

`gpiozero basic recipes:`

- `Servo`

Pulse Width Modulation controlling Servo Motors

Period 20 milliseconds, Frequency 50 Hz

angle	high time	duty cycle	
-90^0	0.5 ms	2.5 %	
-45^0	1 ms	5 %	
0^0	1.5 ms	7.5 %	
$+45^0$	2 ms	10 %	…
$+90^0$	2.5 ms	12.5 %	…

```
from gpiozero import Servo
from time import sleep

servo = Servo(17,
    min_pulse_width=0.5/1000,  ❶
    max_pulse_width=2.5/1000)

while True:
    servo.value = -1  ❷
    sleep(2)
    servo.value = 0  ❸
    sleep(2)
    servo.value = 1  ❹
    sleep(2)
```

❶ 指明最大及最小 pulse width

❷ 停在最小角

❸ 停在中位角

❹ 停在最大角

用 VR 控制 servo

```
from gpiozero import Servo, MCP3008
import time

servo = Servo(17,
    min_pulse_width=0.5/1000,
    max_pulse_width=2.5/1000)

vr = MCP3008(channel=0)

while 1:
    servo.value = vr.value
```

問題：

VR: 0 → 1
Servo: -1 → 1

怎樣由 0 → 1 轉化成 -1 → 1？

Camera

Plug in the camera module. Update the system:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo raspi-config
```

Select **Interfacing Options**, enable **Camera**.

Finish and **reboot**.

raspistill

capture photo

`raspistill -o a.jpg`	Capture photo to a file.

On LCD, **open the file manager** and double-click on the file to view the photo.

`raspistill -o a.jpg -t 1000`	Wait 1 second before capturing. Default is 5 seconds.
`raspistill -o a.jpg -hf -vf`	Flip image horizontally and vertically.
`raspistill | less`	Display options.

Interesting options:

```
-t, --timeout
-hf, --hflip
-vf, --vflip
-w, --width
-h, --height
-ss, --shutter
-ifx, --imxfx
```

raspivid

capture video

`raspivid -o b.h264 -t 5000 -fps 25`	Record 5 seconds at 25 frames per second.
`omxplayer b.h264 --fps 25`	View the video on an HDMI display. This will not work on VNC because **omxplayer** makes use of hardware acceleration in the GPU (graphics processing unit), which is only available to HDMI output.
`raspivid | less`	Display options. Similar to those of **raspistill**.

Convert h264 to mp4

```
$ sudo apt-get install gpac
```

To concatenate many files to a new file (if the destination file exists, its original content will be erased):

```
$ MP4Box -fps 25 -cat a.h264 -cat b.h264 -new c.mp4
```

To concatenate many files to an existing file (if the destination file does not exist, it will be created):

```
$ MP4Box -fps 25 -cat a.h264 -cat b.h264 c.mp4
```

Capturing to a file

```
from time import sleep
from picamera import PiCamera

camera = PiCamera()
camera.resolution = (1024, 768)
sleep(2)  # Camera warm-up time
camera.capture('foo.jpg')
```

Change Camera's Output Orientation

```
camera.hflip = True
camera.vflip = True
camera.rotation = 90
```

The **picamera** package has its own very good documentations. Explore it yourself if you want to know more:

http://picamera.readthedocs.io/

Recording video to a file

```
import picamera

camera = picamera.PiCamera()
camera.resolution = (640, 480)

camera.start_preview()
camera.start_recording('video.h264')

camera.wait_recording(5)

camera.stop_recording()
camera.stop_preview()
```

Overlaying text on the output

The camera includes a rudimentary annotation facility which permits up to 255 characters of ASCII text to be overlaid on all output.

```
import picamera

camera = picamera.PiCamera(resolution=(1280, 720), framerate=24)

camera.annotate_background = picamera.Color('black')
camera.annotate_text = 'Hehehehe'

camera.start_preview()
camera.start_recording('video.h264')
camera.wait_recording(3)
camera.annotate_text = 'Wahahahahaha'
camera.wait_recording(3)
camera.stop_recording()
camera.stop_preview()
```

Camera GUI

https://github.com/Billwilliams1952/PiCameraApp

ansi.py

```
def escape(x):
    return '\x1b[38;5;{}m'.format(x)

def reset():
    return '\x1b[0m'

def color(id, s):
    return escape(id) + s + reset()
```

1. 列出全部 256 種顏色。
   `Hint: range(0, 256), print(..., end=' ')`

2. 印 16x16 的方陣。
   `Hint: 用兩個 for loop`

3. 對齊數字。
   `Hint: rjust`

二進制	十六進制	十進制
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

2 7 4

十進制

$$\begin{array}{llcr}
4 & \times 10^0 & = & 4 \\
7 & \times 10^1 & = & 70 \\
2 & \times 10^2 & = & 200 \\
\hline
 & & & 274
\end{array}$$

1 1 0 1

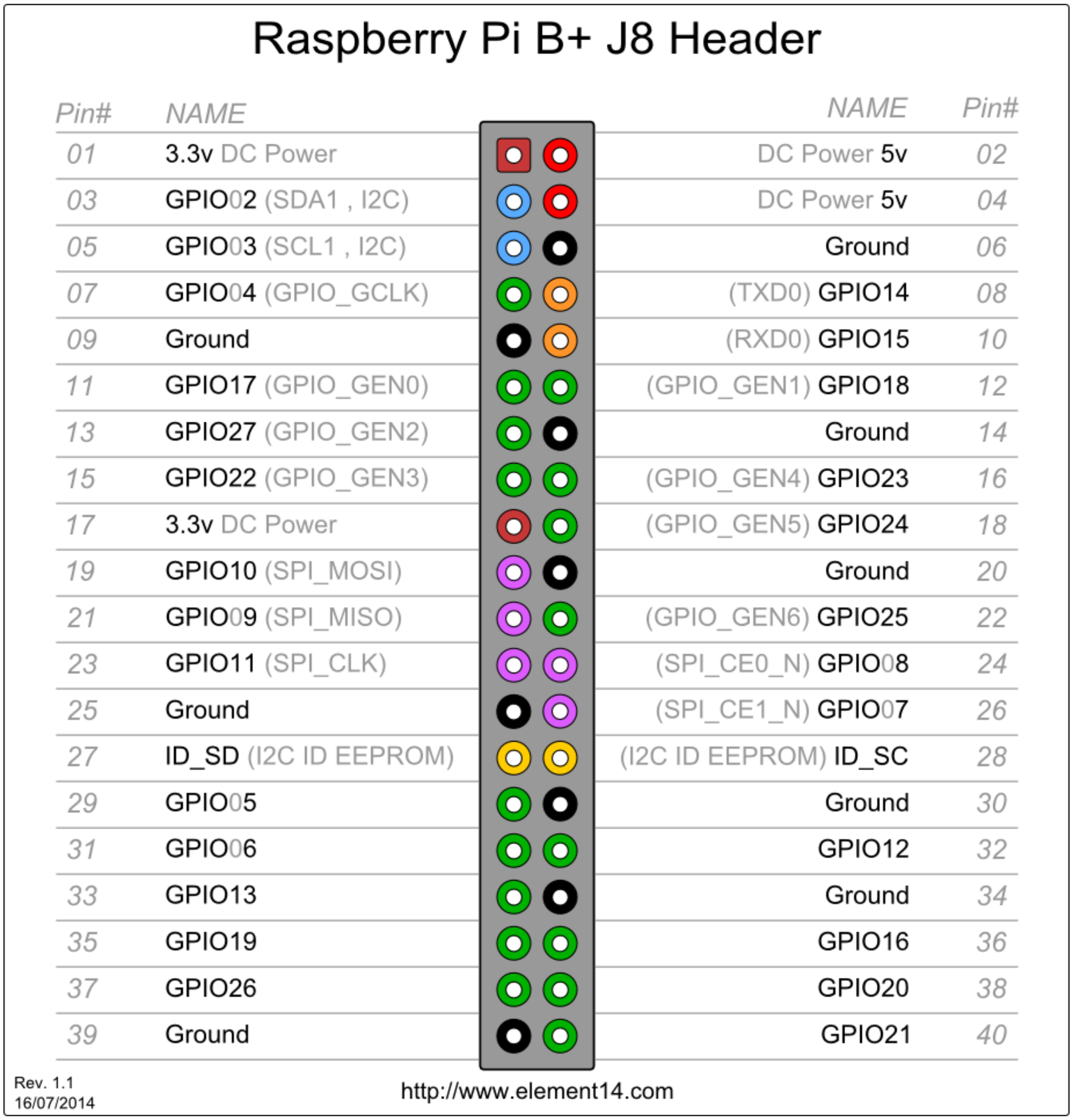
二進制

$$\begin{array}{llcr}
1 & \times 2^0 & = & 1 \\
0 & \times 2^1 & = & 0 \\
1 & \times 2^2 & = & 4 \\
1 & \times 2^3 & = & 8 \\
\hline
 & & & 13
\end{array}$$

1 B

十六進制

$$\begin{array}{llcr}
11 & \times 16^0 & = & 11 \\
1 & \times 16^1 & = & 16 \\
\hline
 & & & 27
\end{array}$$



Raspberry Pi B+ J8 Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I2C)	DC Power 5v	04
05	GPIO03 (SCL1 , I2C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 1.1
16/07/2014

http://www.element14.com