Python Tutorial Raspberry Pi
基本模組函式庫

module

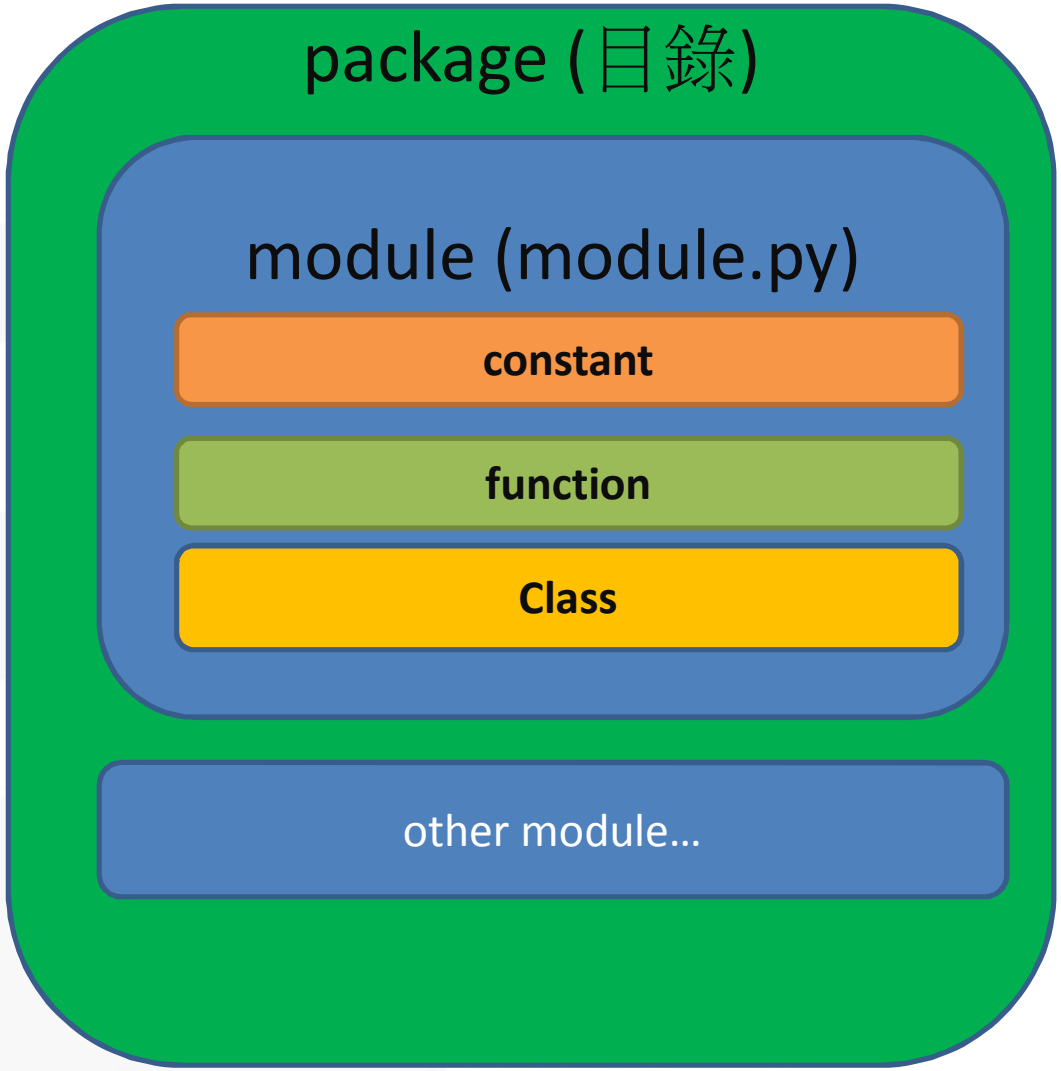
使用關鍵字 import 來匯入模組使用
配合關鍵字 as 可更模組名稱
配合關鍵字 from 可以縮短模組名稱

```
import sys
import tkinter as tk
from time import sleep
from random import *
```

function, class, module, package

```
In [8]: type(my_module)
Out[8]: module

In [9]: type(say_)
Out[9]: function
```



Package? Module?

```
In [1]: from my_package import my_module

In [ ]: my_module.
        my_module.say_hello
        my_module.say_my_name
```

```
In [2]: from my_package import my_module as mm

In [ ]: mm.
        mm.say_hello
        mm.say_my_name
```

```
In [3]: import my_package
        type(my_package)

Out[3]: module
```

```
./my_package/
├── __init__.py
├── my_module_2.py
├── my_module_3.py
└── my_module.py
```

練習

- 試著製作自己的package
- 系統預設的module之一 sys 裡面有一個 path 參數記錄著package搜尋的路徑，試著去尋找 pip 安裝 package 的地方。

```
In [15]: import sys
         sys.path

Out[15]: ['',
          '/usr/lib/python3.4',
          '/usr/lib/python3.4/plat-i386-linux-gnu',
          '/usr/lib/python3.4/lib-dynload',
          '/usr/local/lib/python3.4/dist-packages',
          '/usr/lib/python3/dist-packages',
          '/usr/lib/python3/dist-packages/IPython/extensions']
```

函式庫

- 系統相關
  - os, sys, subprocess, random, time
- 資料處理
  - json, lxml
- 多工處理
  - threading, subprocess
- 網路應用相關
  - requests, flask

sys – System specific parameters and functions

Reference: https://docs.python.org/3/library/sys.html

```
In [1]: import sys,os

In [2]: sys.path
```

模組的存放路徑

```
Out[2]: ['',
         '/usr/lib/python3.4',
         '/usr/lib/python3.4/plat-i386-linux-gnu',
         '/usr/lib/python3.4/lib-dynload',
         '/usr/local/lib/python3.4/dist-packages',
         '/usr/lib/python3/dist-packages',
         '/usr/lib/python3/dist-packages/IPython/extensions']

In [3]: sys.version
```

Python 版本資訊

```
Out[3]: '3.4.3 (default, Mar 26 2015, 22:07:01) \n[GCC 4.9.2]'

In [4]: sys.copyright

Out[4]: 'Copyright (c) 2001-2015 Python Software Foundation.\nAll Rights R
        ed.\n\nCopyright (c) 1995-2001 Corporation for National Research I
        Stichting Mathematisch Centrum, Amsterdam.\nAll Rights Reserved.'
```

sys.argv

```
#!/usr/bin/env python3

import sys

print(sys.argv)
```

```
pi@raspberrypi ~/code/test $ ./01-app.py hello world
['./01-app.py', 'hello', 'world']
pi@raspberrypi ~/code/test $
```

sys.argv[0]➔ 檔名
sys.argv[1]➔ 第一個參數
sys.argv[2]➔ 第二個參數
…

練習

- 使用 sys.argv 製作一個可傳入參數的執行檔，用來將指定 big5 編碼的文字檔轉換成 utf-8 編碼後另存新檔
  1) ex:

```
> convert.py file_big5.txt file_utf8.txt
```

os –operating system interface

```
import os
```

```
os.system("mkdir test2")
```

```
0
```

呼叫 shell 使用 command-line 指令
ex: 製作一個名為test2 的目錄
回傳一個數值代表執行狀況
(0表示正常結束)

```
os.access("./test",os.W_OK | os.R_OK)
```

```
True
```

查看"指定檔案" 狀況
回傳判斷結果 (True/False)

os.W_OK : 是否可寫入
os.R_OK : 是否可寫入
os.F_OK : 是否存在

Reference: https://docs.python.org/3/library/os.html

os –operating system interface

os 其他成員函數:
lsdir(): 顯示指定目錄下的檔案
chdir() : 移動當下工作路徑
chmod(): 改變指定檔案的權限
chown():改變指定檔案的所有者和所有團體
remove(): 刪除檔案
removedirs():刪除目錄
rename(): 改變檔名
…

random 亂數

```
import random
```

```
random.random()
```

```
0.3538438584547304 3
```

產生一個小於1.0的浮點數亂數

```
random.randint(10,20)
```

```
11
```

產生指定範圍內的整數亂數
ex: 產生一個10~20之間的亂數

```
a = [1,2,3,4,5,6,7,8]
random.shuffle(a)
print(a)
```

```
[6, 3, 7, 4, 1, 2, 8, 5]
```

把一個 List 中的元素的順序打亂

```
random.randrange(10,20,3)
```

```
16
```

產生指定範圍內的整數亂數並指定間隔
ex: 產生一個10~20之間的亂數,間隔為3

time

```
In [9]: import time

In [10]: time.time()
Out[10]: 1432862616.01428

In [11]: print ("Number of ticks since 12:00am, January 1, 1970:", time.time())
         Number of ticks since 12:00am, January 1, 1970: 1432862662.5365212
```

用浮點數表示當下系統時間(單位：秒)

https://docs.python.org/3/library/time.html#time.strftime

time.localtime()

```
print([i for i in dir(time.localtime()) if 'tm_' in i])
['tm_gmtoff', 'tm_hour', 'tm_isdst', 'tm_mday', 'tm_min', 'tm_mon
```

Attributes	Values
tm_year	2008
tm_mon	1 to 12
tm_mday	1 to 31
tm_hour	0 to 23
tm_min	0 to 59
tm_sec	0 to 61 (60 or 61 are leap-seconds)
tm_wday	0 to 6 (0 is Monday)
tm_yday	1 to 366 (Julian day)
tm_isdst	-1, 0, 1, -1 means library determines DST

Time strftime

```
In [27]: time.strftime("%Y-%m-%d, %H:%M:%S")
Out[27]: '2015-05-28, 18:29:58'
```

subprocess

```
In [1]: import subprocess

In [2]: subprocess.call(['ls','-l'])
        subprocess.call('mkdir test_dir', shell=True)

Out[2]: 0

In [3]: subprocess.check_call('mkdir test_dir', shell=True)

---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
<ipython-input-3-2203cc3ee082> in <module>()
----> 1 subprocess.check_call('mkdir test_dir', shell=True)

/usr/lib/python3.4/subprocess.py in check_call(*popenargs, **kwargs)
    559         if cmd is None:
    560             cmd = popenargs[0]
--> 561         raise CalledProcessError(retcode, cmd)
    562     return 0
    563

CalledProcessError: Command 'mkdir test_dir' returned non-zero exit status 1

In [4]: subprocess.check_call(['rmdir','test_dir'])

Out[4]: 0

In [5]: subprocess.check_output(['ls','-l'])

Out[5]: b'total 124\n-rw------- 1 onionys onionys 16488 May 28 18:57 buildin_function.i
        01:56 class.ipynb\ndrwxr-xr-x 2 onionys onionys  4096 May 28 17:42 data\n-rw---
        nary.ipynb\n-rw------- 1 onionys onionys 10112 May
        15 May 27 03:09 for loop.ipynb\n-rw------- 1 onion
```

https://docs.python.org/3/library/subprocess.html

```
import subprocess
```

```
a = subprocess.check_output(['dir'], shell=True)
```

```
a.decode('big5')
```

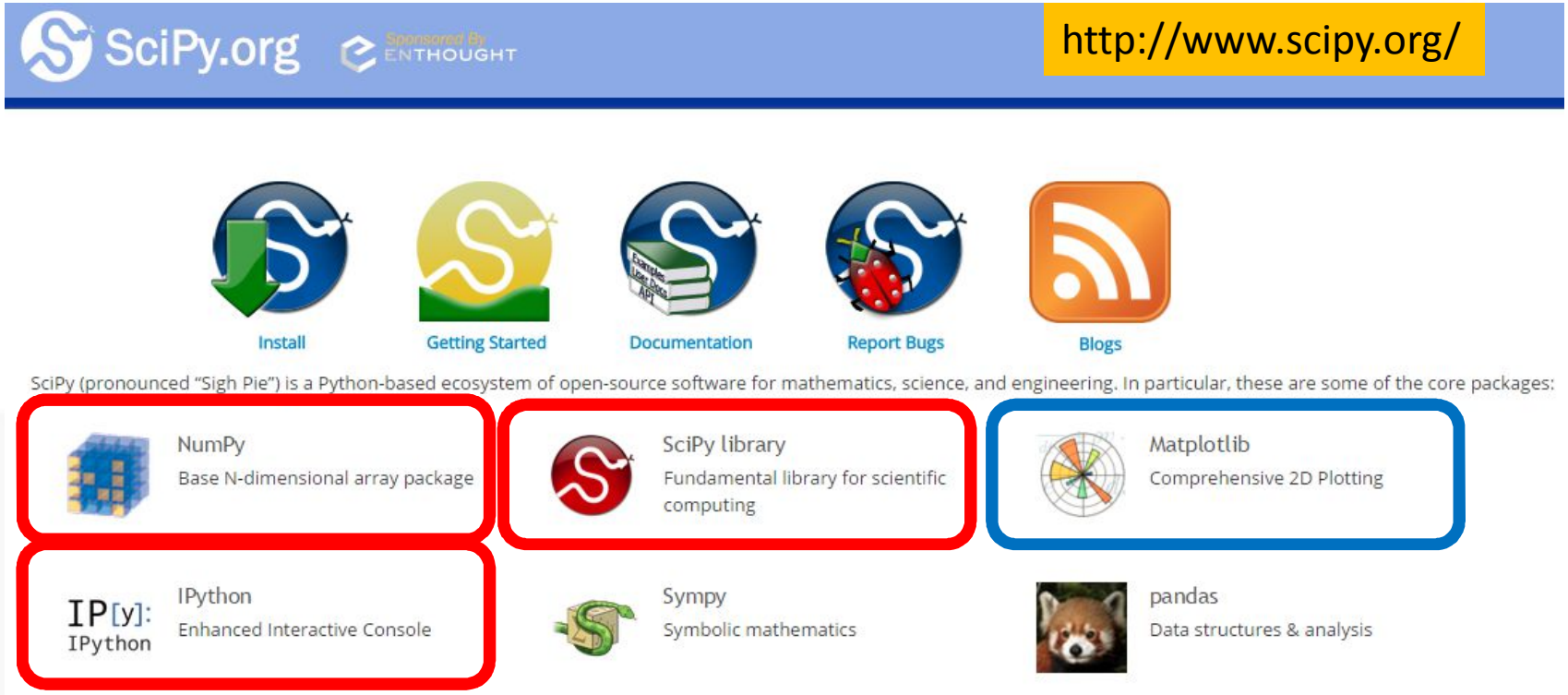
```
' 磁碟區 C 中的磁碟是 BOOTCAMP\r\n 磁碟區序號:  3AE5
錄\r\n\r\n2015/09/13  下午 04:18    <DIR>
..\r\n2015/09/13  下午 03:24    <DIR>          .ip
2,498 01-整數.ipynb\r\n2015/09/04  下午 02:14
5,348 basic IO.ipynb\r\n2015/09/10  上午 07:49
110 data.txt\r\n2015/08/26  下午 02:29
12 file_utf8.txt\r\n2015/08/26  下午 05:13
下午 04:30                22 hello.py\r\n2015/09/1
8/20  下午 04:20             2,702 Hello1.ipynb\r\
```

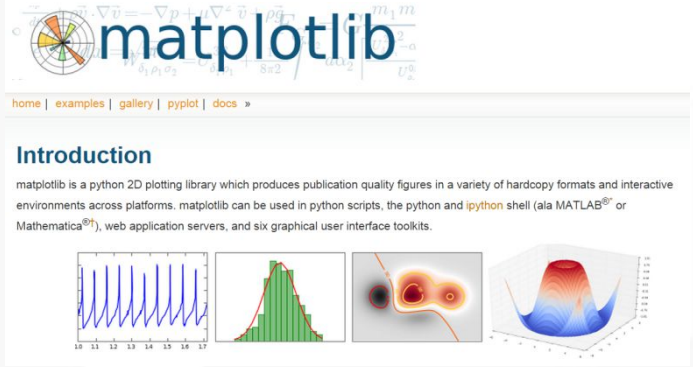
練習

寫一個 function, 使用 subprocess.check_output("…", shell=True)

配合系統指令 ifconfig 得到 目前主機的所有 IP 位址(字串形式)組成的 list 並回傳。

math, cmath, numpy, scipy, matplotlib





在 Pi 上面安裝 Python3 matplitlib 套件

```
git clone https://github.com/matplotlib/matplotlib
cd matplotlib
python3 setup.py build
sudo python3 setup.py install
```

平行處理: threading

```
#!/usr/bin/env python3


from threading import Thread
from time import sleep

system_run = 1

def do_something_func():
    while(system_run):
        sleep(1)
        print("do_something_func: running....")
    print("do_something_func: stop!!!")


if __name__ == '__main__':
    do_something_thread = Thread(target = do_something_func)
    do_something_thread.start()

    while(system_run):
        a = input()
        if a == 'a':
            system_run = 0
```

使用全域變數來控制 Thread 裡的 loop 的停止

global system_run

練習

- 建立一個thread 定時丟一個訊息給一個全域變數 List ，主程式則定時從該 List 取訊息出來顯示在螢幕上。
- 在該thread與主程式執行期間用 ctrl+C 中斷其執行觀察其結果

平行處理: multiprocess

```
#!/usr/bin/env python3

from multiprocessing import Process, Queue
from time import sleep


def some_loop_func():
    inf = open('proc_log.txt','w')
    for i in range(10):
        sleep(1)
        inf.write("some_proc:%d\n" % i)
    inf.close()


if __name__ == '__main__' :
    some_loop_proc = Process(target = some_loop_func)
    some_loop_proc.start()
```

multiprocess communication

無法使用全域變數來控制 subprocess 裡的 loop 的停止，使用內建類別Queue來做Process之間的通訊

```
#!/usr/bin/env python3

from multiprocessing import Process, Queue, current_process
from time import sleep


def some_loop_func(q):
    p = current_process()

    while True:
        sleep(1)
        if(q.empty()):
            print("%s: waiting for message ...." % p.name)
        else:
            message = q.get()
            print("%s: I got message from queue : <%s>" % (p.name, message))
            if(message == 'stop'): break

    print("some_loop_func stop!!")



if __name__ == '__main__' :
    queue_01 = Queue()

    some_loop_proc_01 = Process(name = "01", target = some_loop_func, args = (queue_01,))
    some_loop_proc_01.start()

    for i in range(2):
        sleep(2)
        print('yo man')
        queue_01.put('yo man!!')

    queue_01.put('stop')
    some_loop_proc_01.join()

    print("main loop stop!!!!")
```

練習

- 建立一個 subprocess 每兩秒丟一個訊息至主程式上(透過Queue)，主程式則每一秒定時將收到的訊息顯示在螢幕上。
  主程式會在程式開始二十秒過後，透過另一個Queue傳送”END”訊息給subprocess令其結束執行

基本內建函式

- enumerate
- zip
- map
- filter

enumerate

In [1]:
```
my_str = 'abcdefghijklmnopqrstuvwxyz'
my_list = list(my_str)
print(my_list)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', '
z']
```

In [2]:
```
for i in range(len(my_list)):
    print ("%d : %s" % (i, my_list[i]))

for i in enumerate(my_list):
    print (i)

for index, char in enumerate(my_list):
    print("== %d : %s ==" % (index,char))
```

```
0 : a
1 : b
2 : c
3 : d
4 : e
5 : f
6 : g
7 : h
8 : i
9 : j
10 : k
```

練習

- 讀取oil2.csv 並使用enumerate來附加 index 在每一行的開頭然後寫入另一個檔案。

zip

```
In [5]: a = [1,2,3,4,5]
        b = list('abcde')
        c = list('ABCDEFG')
```

```
In [6]: for i in zip(a,b,c):
            print (i)
```

```
(1, 'a', 'A')
(2, 'b', 'B')
(3, 'c', 'C')
(4, 'd', 'D')
(5, 'e', 'E')
```

```
In [10]: a = [1,2,3,4,5]
         b=['a','b','c','d','e']
         c = { x:y for x,y in zip(a,b)}
         print(c)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

練習

- 讀取 分析 oil2.csv 三個得到98, 95, 92, 和 柴油的價格list:
  oil_98 = [...]
  oil_95 = [...]
  oil_92 = [...]
  diesel = [...]
  利用zip 重新打包順序為
  [...
  (柴油, 98, 95, 92) ,
  ...]
  並寫入另一個檔案

map

```
In [ ]: def power(x):
            return x ** 2

In [ ]: a = [1,2,3,4,5,6,7]

In [ ]: for i in a:
            print (power(i))

In [ ]: for i in map(power,a):
            print (i)

In [ ]: [power(i) for i in a]

In [ ]: [i for i in map(power,a)]
```

```
In [6]: list(map( str, [1,2,3,4,5,6]))
Out[6]: ['1', '2', '3', '4', '5', '6']
```

練習

- 讀取 分析 oil2.csv 得到95汽油的價格，並利用 map 計算每個日期加滿 2.5 公升的價格，並寫入到另一個檔案
- 不使用map的作法?

filter

```
In [8]: a = [1,2,3,4,5,6,7,8,9]

        def select(x):
            return 1 if (x % 2 == 0) else 0
```

```
In [10]: [i for i in filter(select,a)]
Out[10]: [2, 4, 6, 8]
```

```
In [11]: [i for i in a if i % 2 == 0]
Out[11]: [2, 4, 6, 8]
```

```
In [3]: list(filter(select, a) )
Out[3]: [2, 4, 6, 8]
```

練習

- 讀取 分析 oil2.csv ，並利用 filter 去掉92汽油低於30元以下的資料，並寫入到另一個檔案。
- 不使用filter的作法?

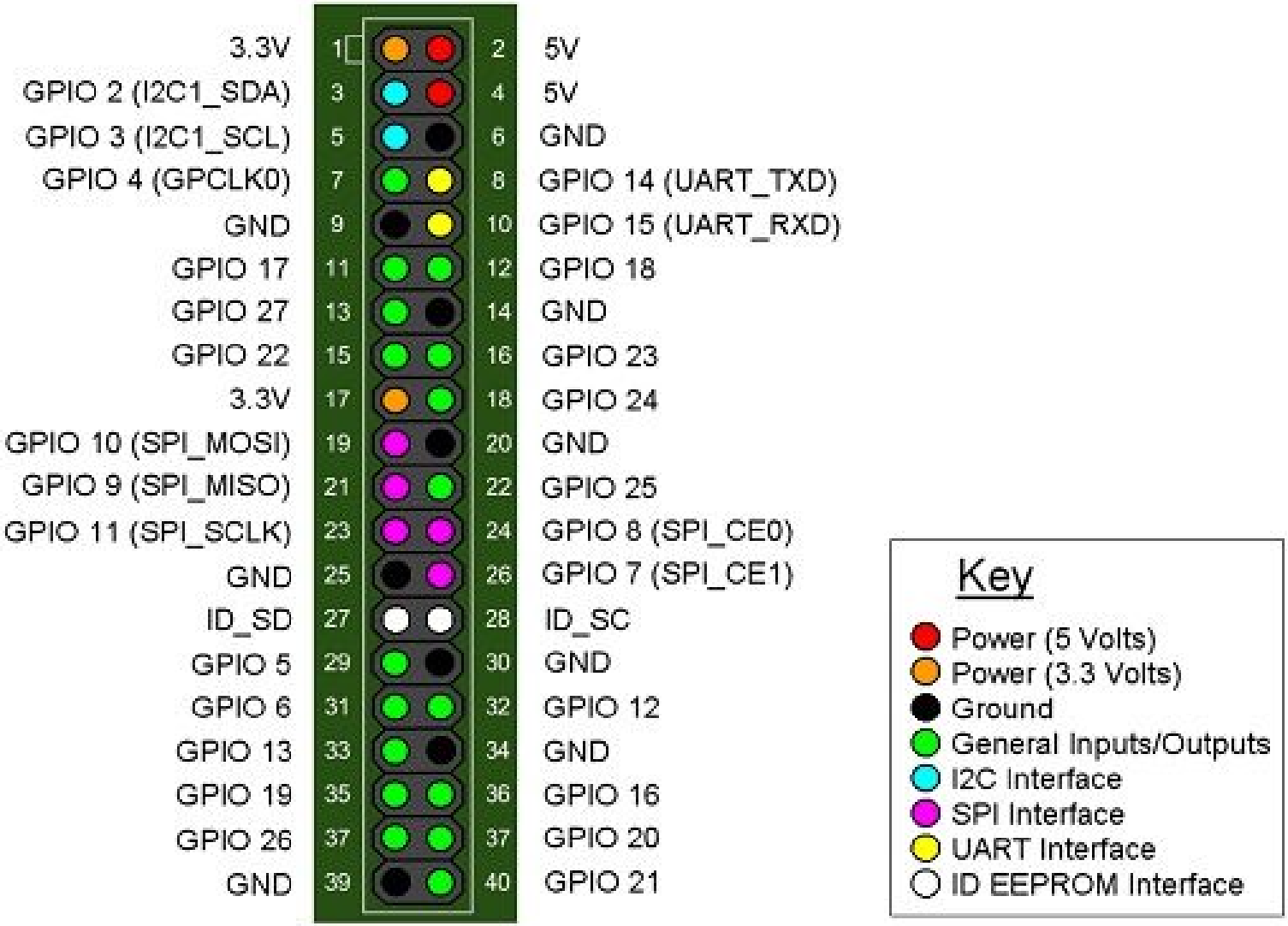
練習

- 分析 oil2.csv 得到各種油的最高價格(&出現的日期)、最低價格(&出現的日期) 、平均價格、價格標準差

- Hint: numpy 套件裡有標準差計算函式庫

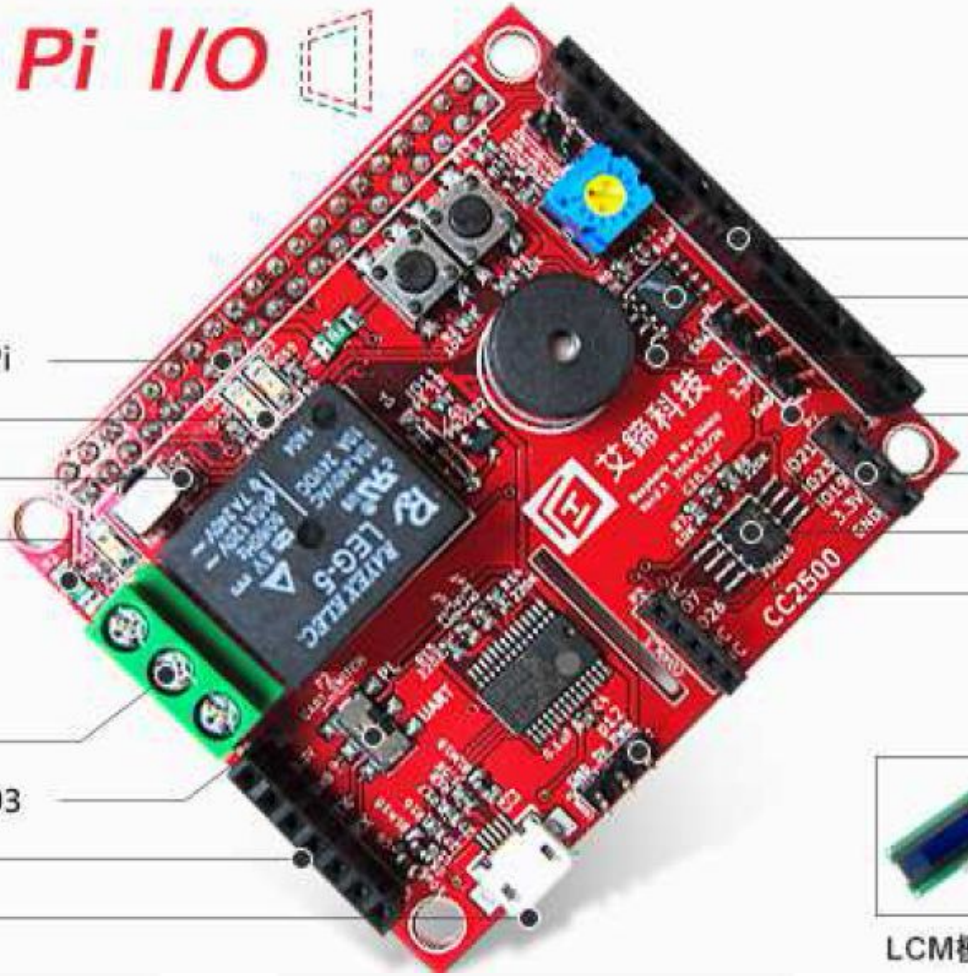
Python Tutorial Raspberry Pi
Raspberry Pi 硬體控制

Raspberry Pi 硬體控制



艾鍗Rasp Pi I/O

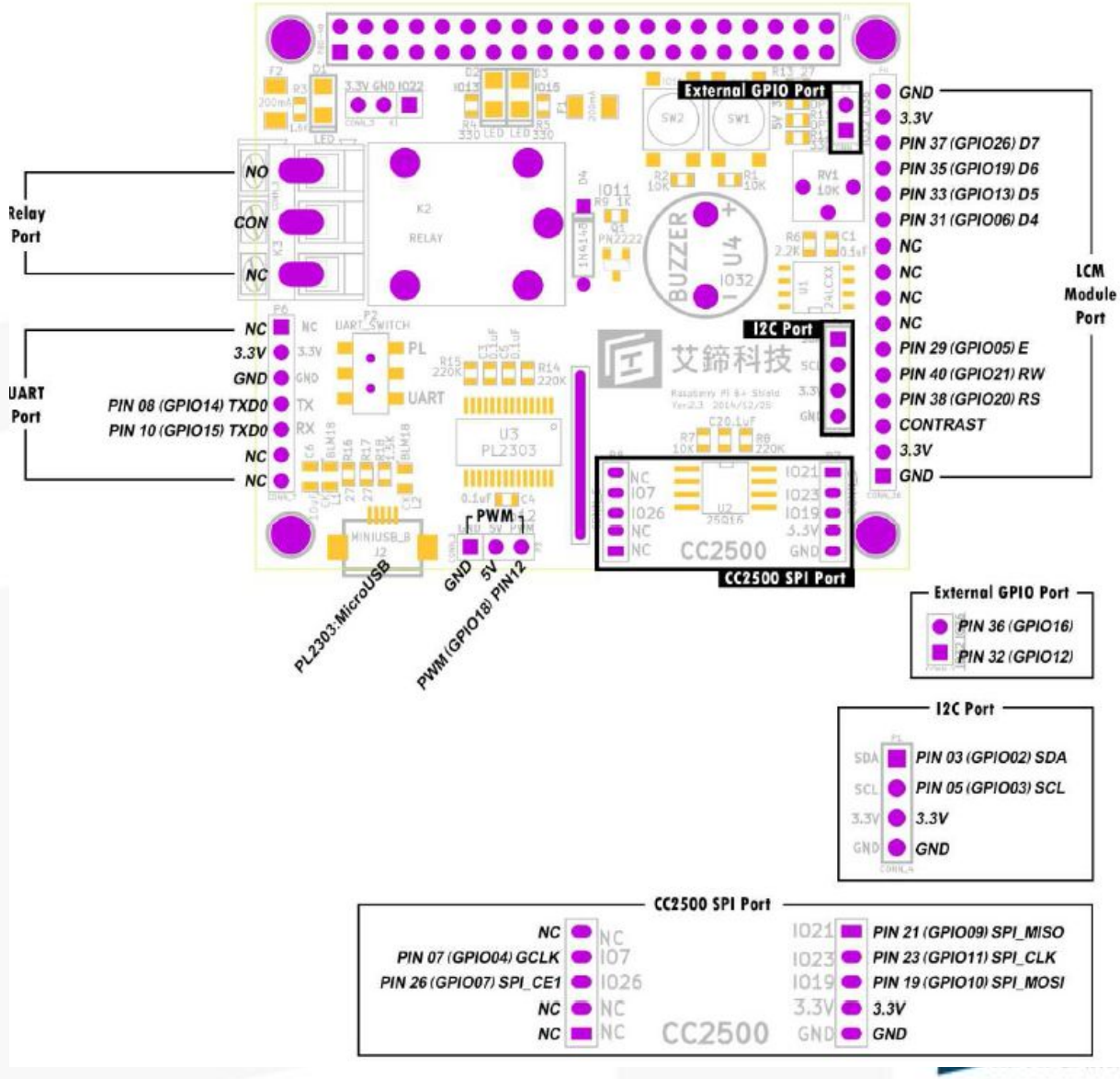
規 | 格 | 圖



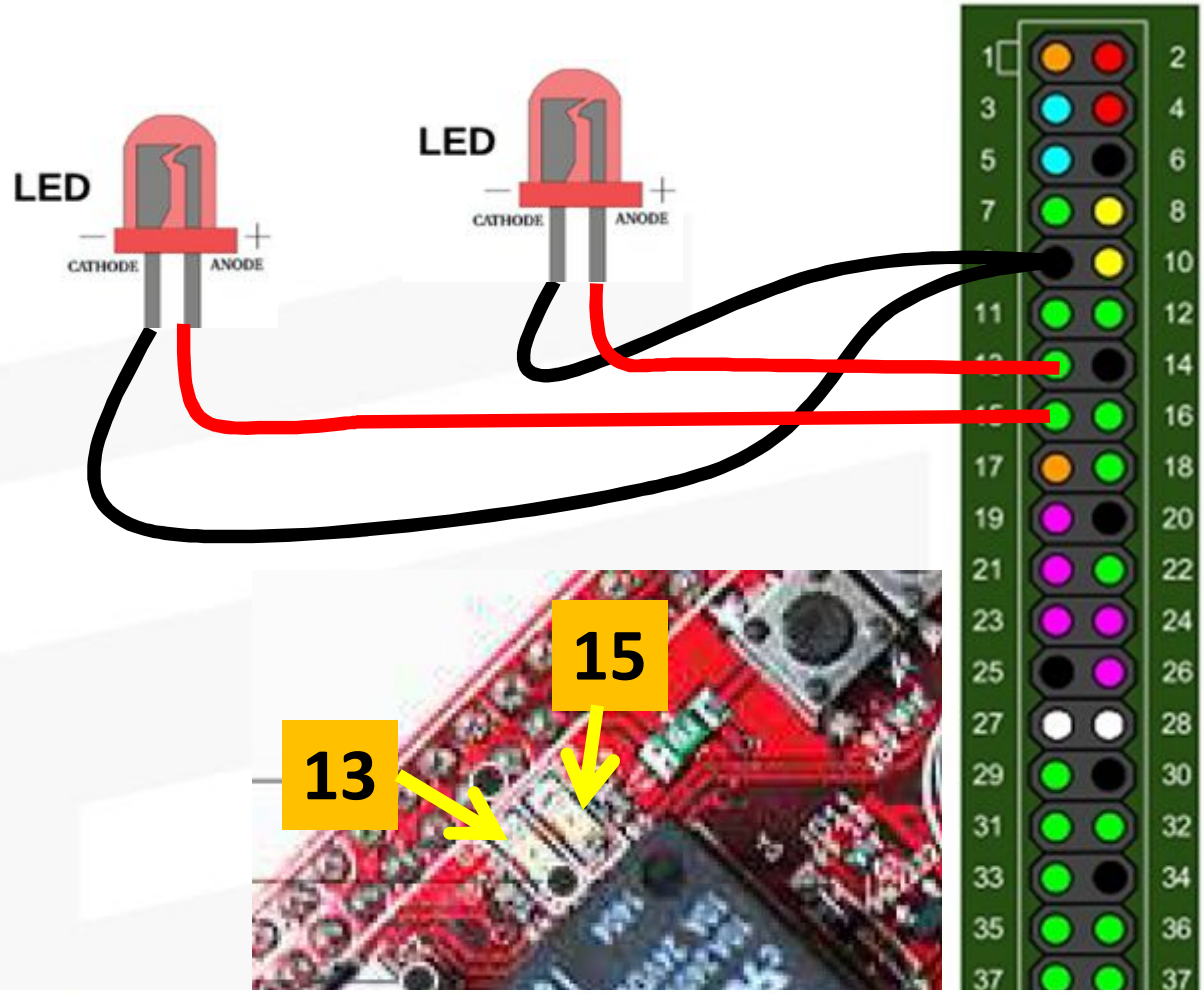
❶ Connector for Rasp Pi
❷ LED
❸ 紅外線接收器
❹ 電源指示燈
❺ 繼電器連接腳座
❻ Switch切換使用PL2303
❼ 藍芽模組腳座
❽ USB to RS232
❾ LCM模組/GPIO延伸腳座
❿ EEPROM
Ⓐ BUZZER
Ⓑ I2C延伸腳座
Ⓒ CC2500 RF模組
Ⓓ Flash
Ⓔ PWM延伸腳座

LCM模組　RF模組　藍芽模組

- GPIO
  1) LED
  2) 按鈕
  3) 繼電器
  4) 蜂鳴器
- I^2C – EEPROM 記憶體
- SPI – CC2500 無線通訊模組
- UART – BLUETOOTH
  1) MODBUS RTU Introduction


Relay Port
NO
CON
NC
UART Port
NC
3.3V
GND
PIN 08 (GPIO14) TXD0
PIN 10 (GPIO15) TXD0
NC
NC
PL2303:MicroUSB
GND
5V
PWM(GPIO18) PIN12
External GPIO Port
GND
3.3V
PIN 37 (GPIO26) D7
PIN 35 (GPIO19) D6
PIN 33 (GPIO13) D5
PIN 31 (GPIO06) D4
NC
NC
NC
NC
PIN 29 (GPIO05) E
PIN 40 (GPIO21) RW
PIN 38 (GPIO20) RS
CONTRAST
3.3V
GND
LCM Module Port
I2C Port
CC2500 SPI Port
External GPIO Port
PIN 36 (GPIO16)
PIN 32 (GPIO12)
I2C Port
PIN 03 (GPIO02) SDA
PIN 05 (GPIO03) SCL
3.3V
GND
CC2500 SPI Port
NC
PIN 07 (GPIO04) GCLK
PIN 26 (GPIO07) SPI_CE1
NC
NC
PIN 21 (GPIO09) SPI_MISO
PIN 23 (GPIO11) SPI_CLK
PIN 19 (GPIO10) SPI_MOSI
3.3V
GND
CC2500

EDU-KIT LED 線路圖



Rasp Pi I/O 子板上的兩顆發光二極體 LED01、LED02 分別對應 Raspberry Pi B+ 板上的 Pin 13 (GPIO 27)、Pin 15 (GPIO 22)。

GPIO LED Control

GPIO module import

```
from RPi import GPIO
```

```
GPIO.setmode(GPIO.BOARD)
```

LED1 Setup

```
GPIO.setup(13,GPIO.OUT)
```

Control END -- cleanup GPIO setup

```
GPIO.cleanup()
```

LED 1 output control

LED_1 pin : 13
LED_2 pin : 15

```
GPIO.output(13,1)
```

```
GPIO.output(13,0)
```

```
from time import sleep
for i in range(10):
    GPIO.output(13,1)
    sleep(0.2)
    GPIO.output(13,0)
    sleep(0.2)
```

LED Function

```
def led_on(pin):
    GPIO.output(pin,1)

def led_off(pin):
    GPIO.output(pin,0)
```

```
led_on(13)
```

```
led_off(13)
```

```
for i in range(10):
    led_on(13)
    sleep(0.2)
    led_off(13)
    sleep(0.2)
```

LED Class

```
from RPi import GPIO

GPIO.setmode(GPIO.BOARD)

class LED:
    def __init__(self,pin):
        self.pin = pin
        GPIO.setup(self.pin,GPIO.OUT)
        self.off()

    def on(self):
        GPIO.output(self.pin,1)

    def off(self):
        GPIO.output(self.pin,0)
```

```
led1 = LED(13)
led2 = LED(15)
```

```
led1.on()
```

```
led2.on()
```

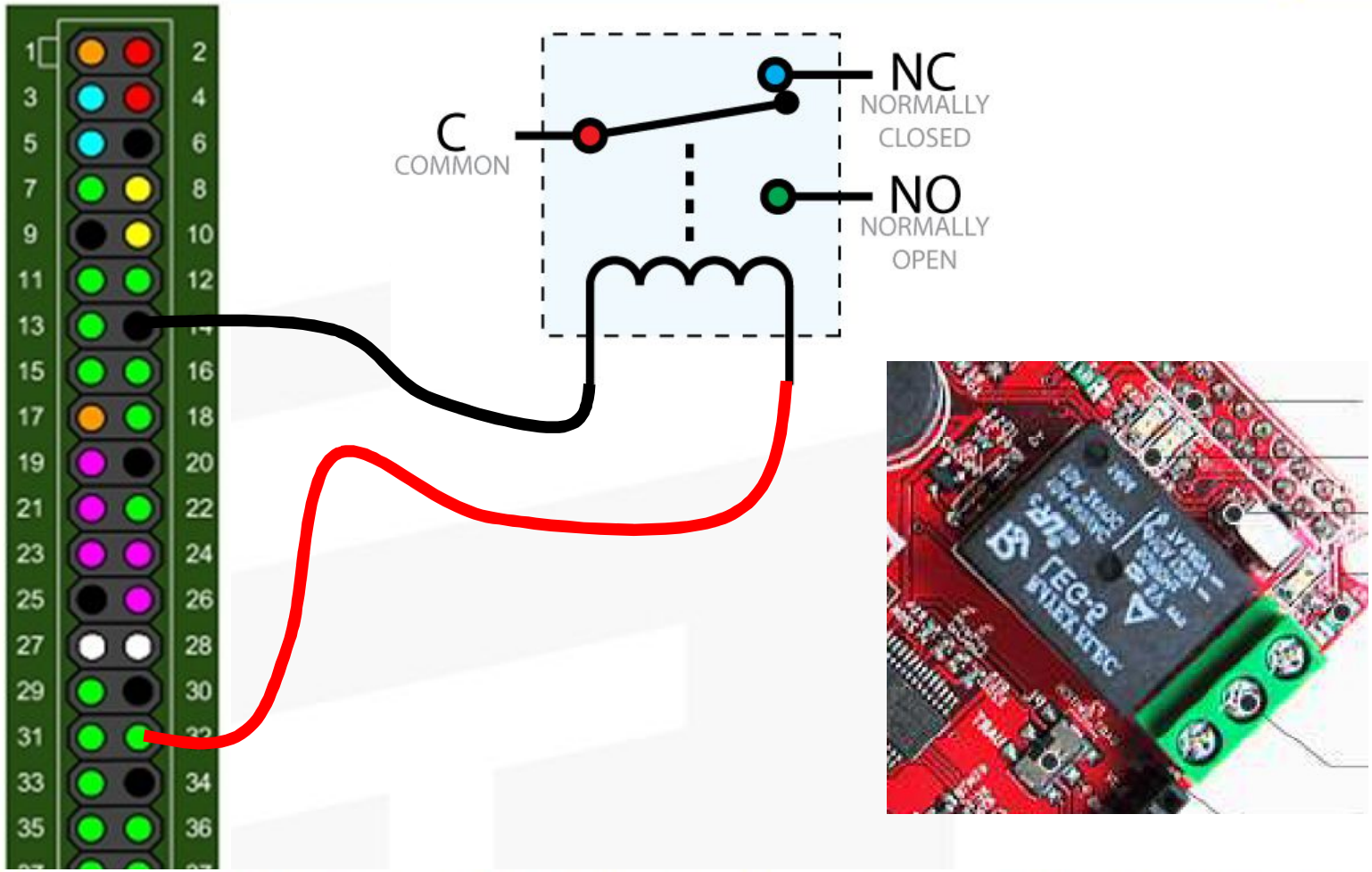
練習

- 擴充 LED 類別功能:

  bright(num):

  傳入一個數字 0 ~ 10 來控制LED 的亮度
- 增加 PWM 功能
- 平行處理 LED 亮度控制(threading,PWM)

EDU-KIT 繼電器線路圖



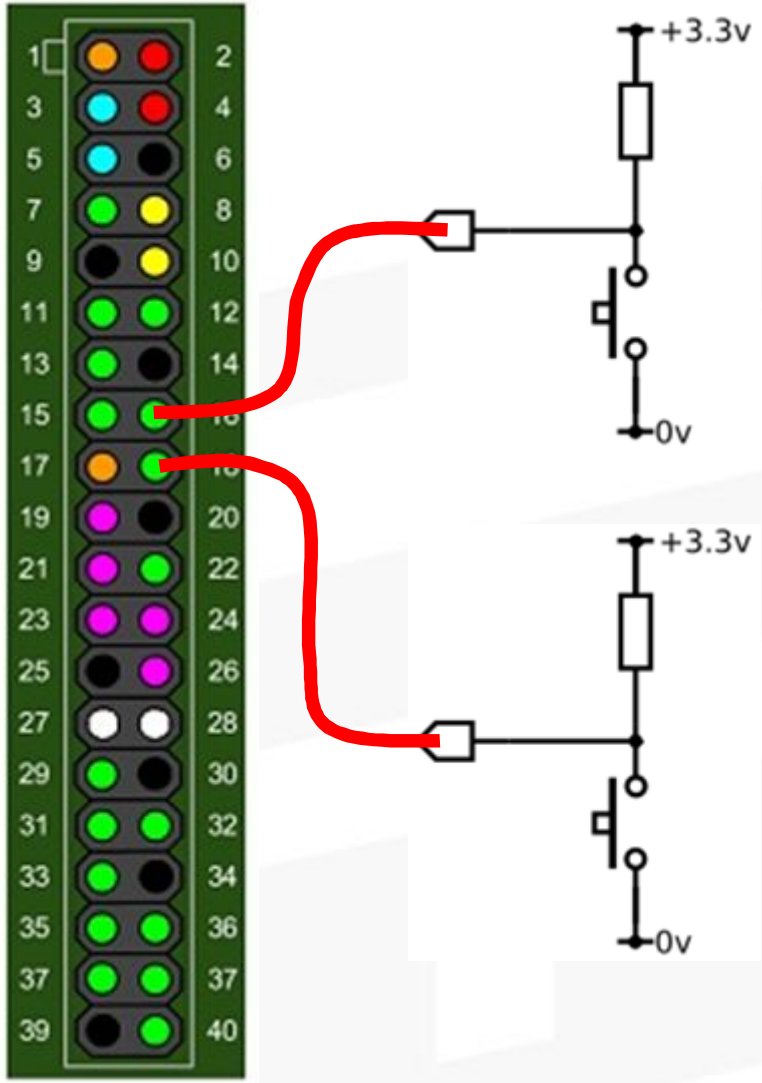
該蜂鳴器為自激式 DC 直流蜂鳴器，和 Raspberry Pi 板上的 Pin 32 連結。

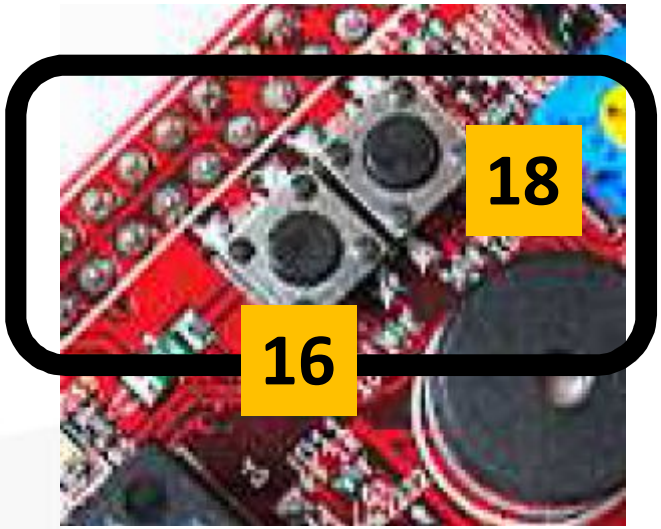
練習

- 參考 LED 類別實作 Relay(繼電器)類別:
  1) .on()
  2) .off()
  3) .state()

可以使用 RPi.GPIO 模組的input()來讀取指定pin的電位為1 (HIGH)還是0(LOW)。即使是在輸出模式。

EDU-KIT 按鈕 線路圖





GPIO Button (input mode)

button 1 pin : 16
button 2 pin : 18

```
from RPi import GPIO
GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setup(16,GPIO.IN)
```

Pull-Up and Pull-Down Resistor

```
GPIO.setup(16,GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
```

```
GPIO.setup(16,GPIO.IN, pull_up_down = GPIO.PUD_OFF)
```

```
GPIO.setup(16,GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

read input level

```
GPIO.input(16)
```

Interrupt and call back

event:

- GPIO.RISING
- GPIO.FALLING
- GPIO.BOTH

Wait for EVENT

```
GPIO.wait_for_edge(16,GPIO.RISING)
```

Detect: Is EVENT happend?

```
GPIO.add_event_detect(16,GPIO.RISING)
```

```
if(GPIO.event_detected(16)):
    print("YES")
else:
    print("NO")
```

Add the callback function to event

```
def call_back_function_one(channel):
    print("call back function one")

def call_back_function_two(channel):
    print("call back function two")
```

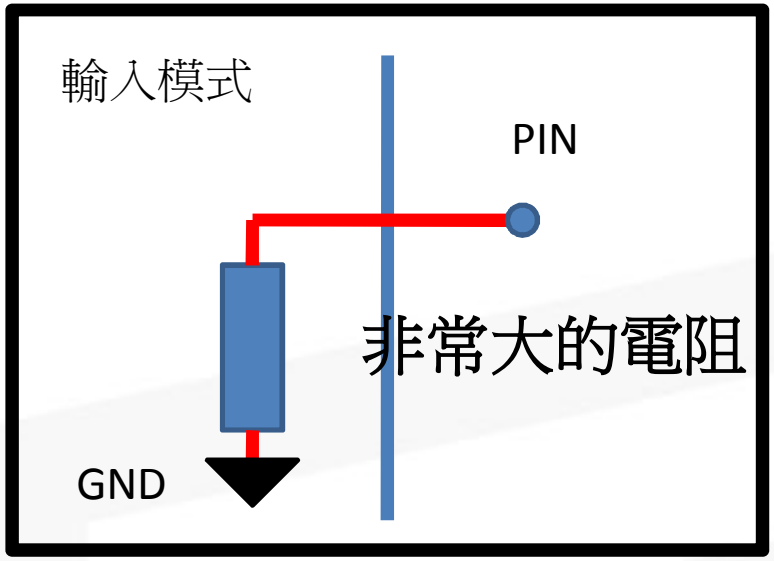
setup method one

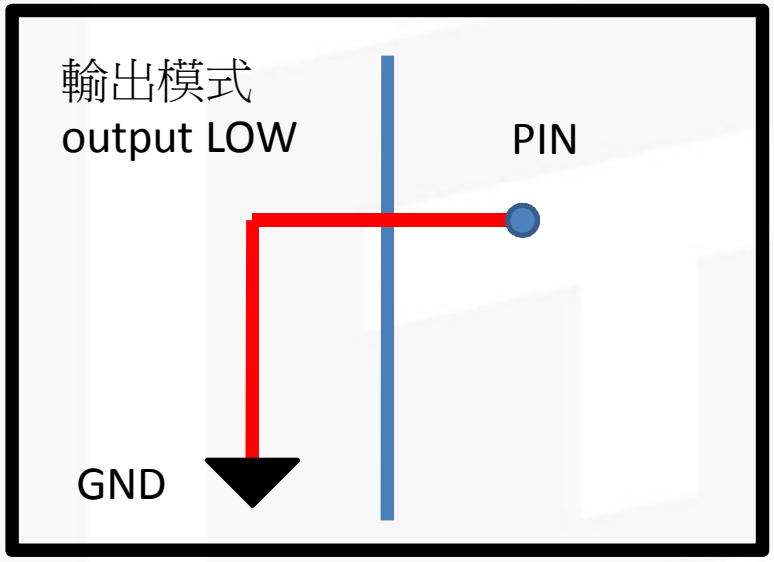
```
GPIO.add_event_detect(16,GPIO.RISING)
GPIO.add_event_callback(16, call_back_function_one)
GPIO.add_event_callback(16, call_back_function_two)
```

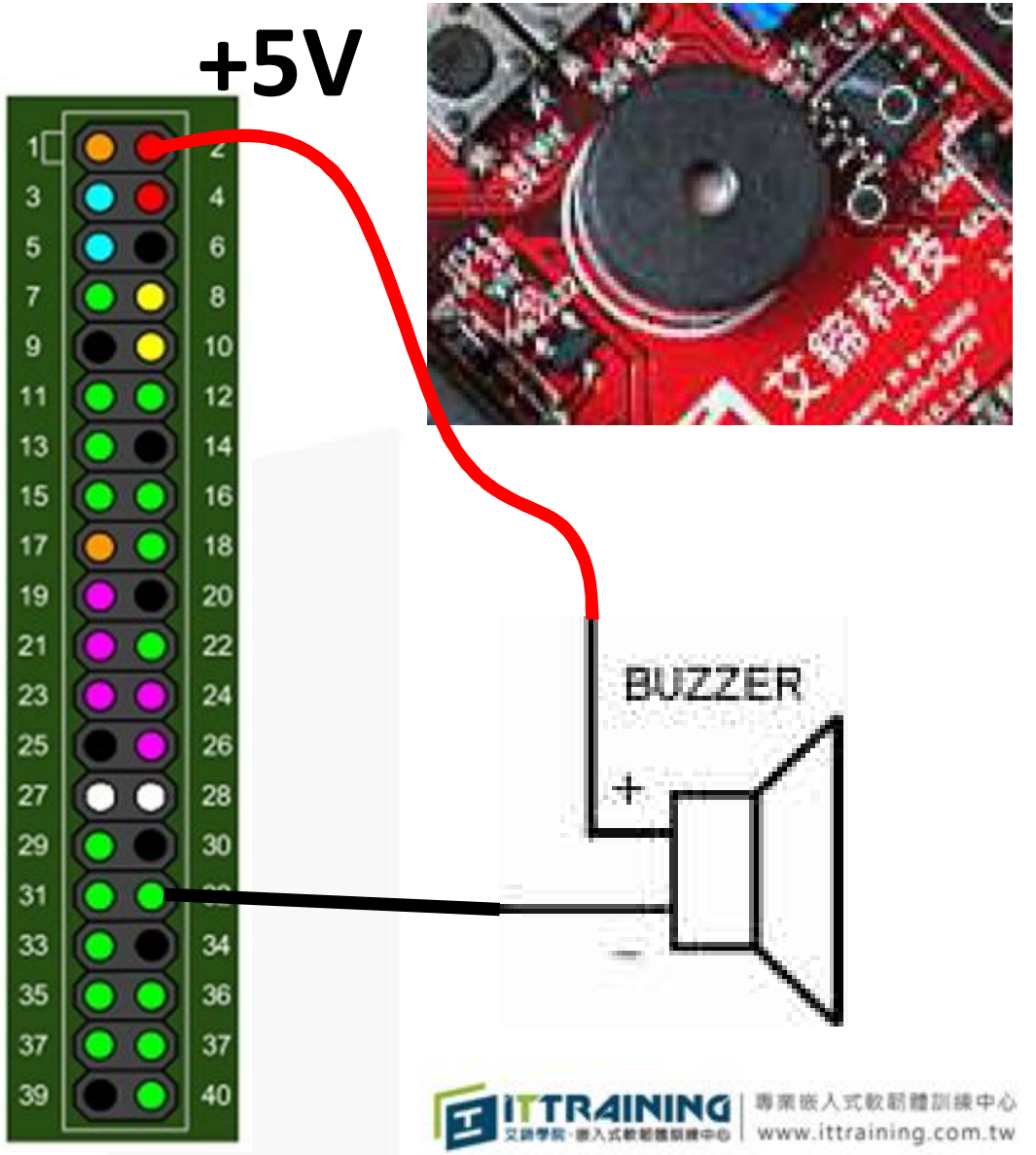
setup method two

```
GPIO.add_event_detect(16,
                      GPIO.RISING,
                      callback = call_back_function_one,
                      bouncetime=50)
# bouncetime unit : ms
```

GPIO-蜂鳴器



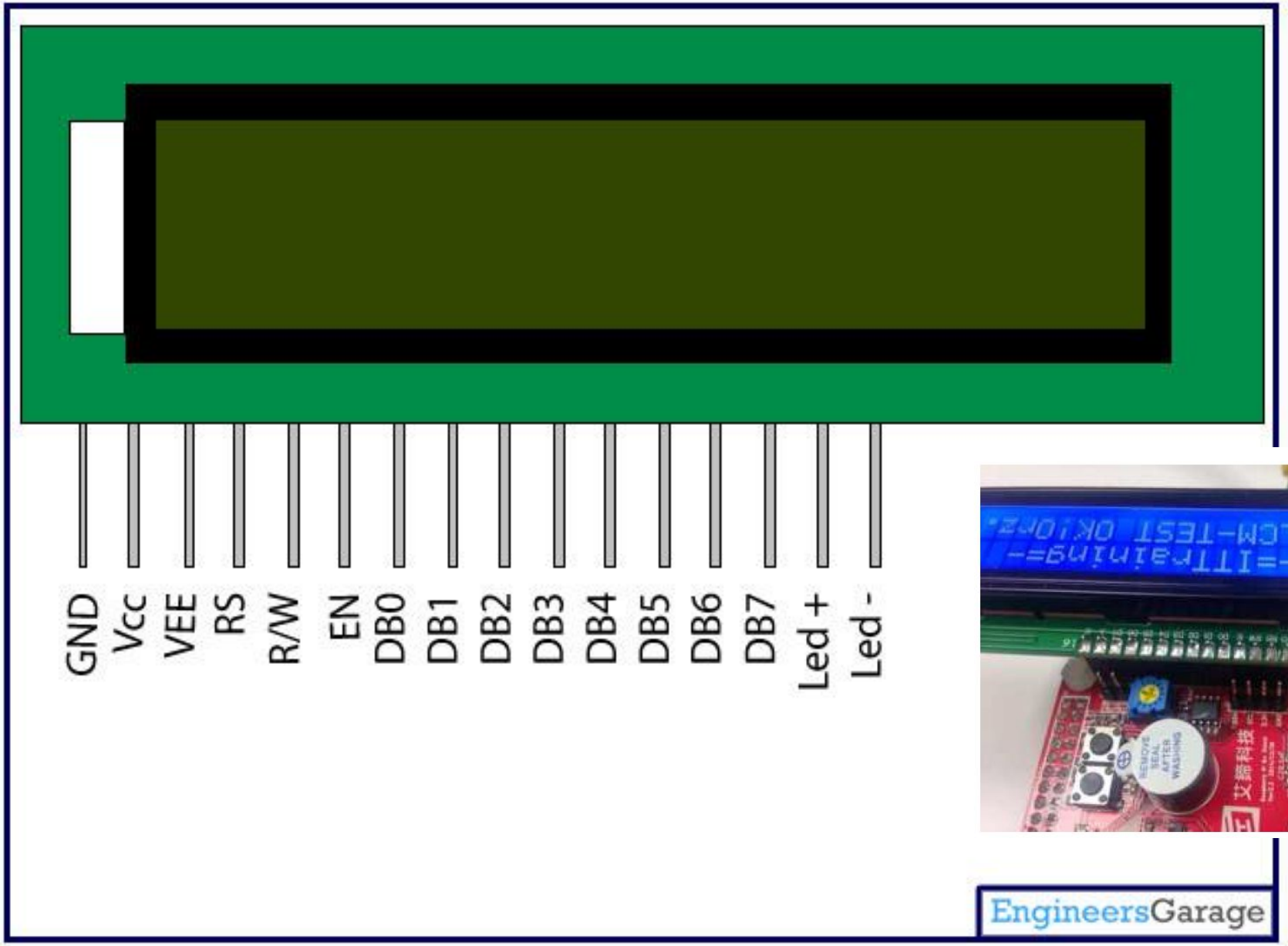




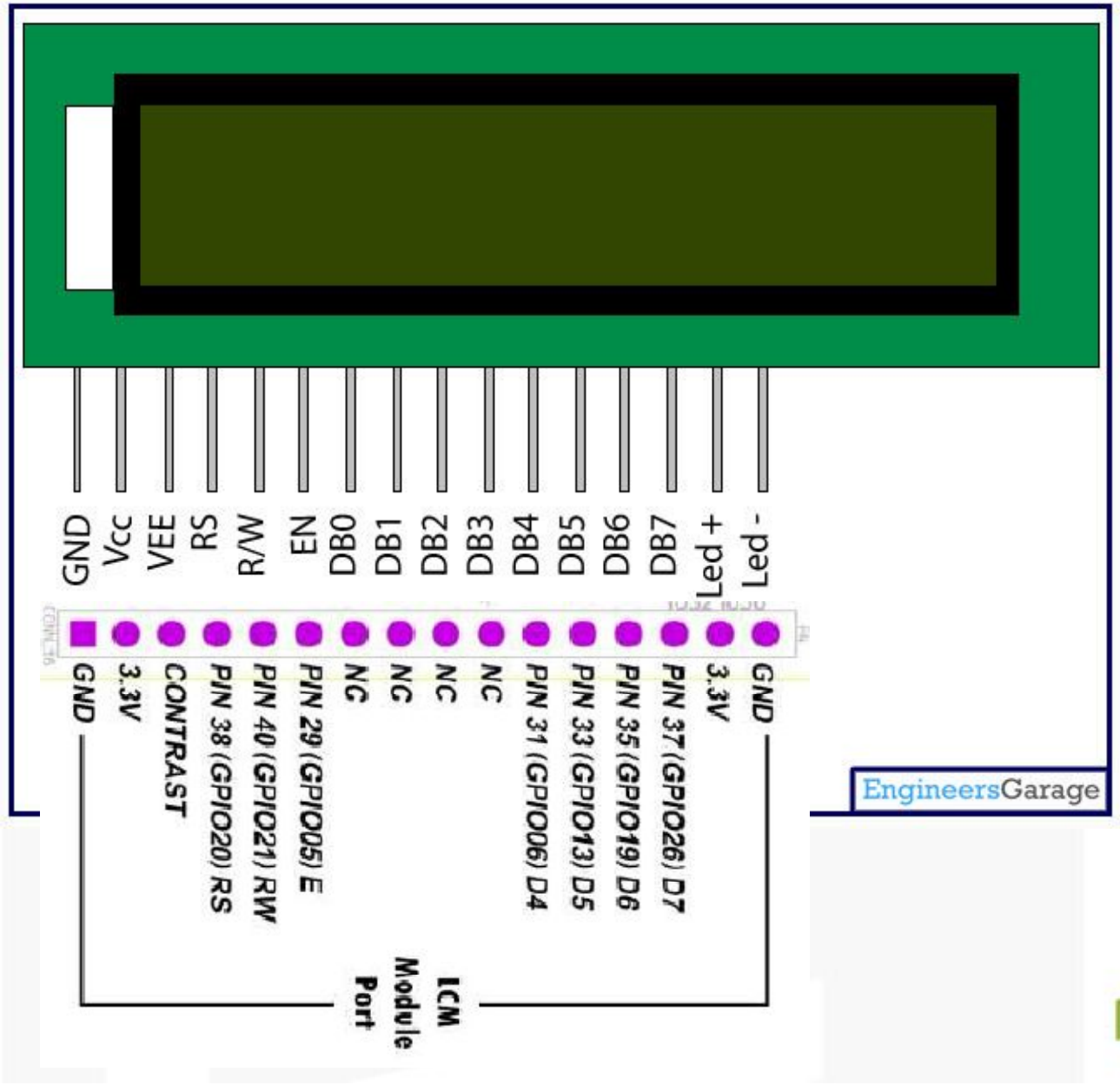
練習

- 參考 LED 類別實作 Buzzer(蜂鳴器)類別:

GPIO-OUTPUT-LCM



EDU kit : LCM針腳設定



Pinout setup

Raspberry Pi : PIN	LCM PIN
38	RS
40	RW
29	EN
31	D4
33	D5
35	D6
37	D7

(訊號功能控制接腳)

4 : RS --> 接 圈圈號碼 38 控制 "指令" 或 "資料" 讀寫模式

5 : R/W --> 接 圈圈號碼 40 控制 "讀出" 或 "寫入" 模式

6 : EN --> 接 圈圈號碼 29 通知LCD模組執行動作

LCM 操作模式

8-bit模式:

使用8支接腳傳輸**資料**和**指令**(一個動作送出/讀入完整的 8 個 bit(1 byte) 數據)

4-bit模式:

使用4支接腳來傳輸**資料**和**指令**，一個動作只能送出/讀入4個bit的數據，所以要送出/讀入完整的一個 byte 要多花一個動作(省下4個GPIO腳位)

How to Control LCM via GPIO?

LCD模組兩個基本動作:

寫入指令

寫入資料

寫入指令 Example:

移動游標到第一行第一格的位置，我們要寫入的指令是0x80(查Datasheet)，轉換成二進位: 0b10000000 。

Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	53μs	38μs	29μs

2 LINES X 16 CHARACTERS PER LINE

Char.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Line 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Line 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

https://www.adafruit.com/datasheets/TC1602A-01T.pdf

傳送前四bit

1. 第一次寫入，準備好指令前面四個bit要寫入的指令:

   EN pin 設為 low

   RW pin 設為 low ==> 跟LCD說這是一個寫入的動作

   RS pin 設為 low ==> 跟LCD說這次的寫入的 "0x80" 是"指令"

   D7~D4 分別對應"指令"(0x80 == 0b1000 0000)的 bit7 ~ bit4

   所以:

   D7 pin 設為 high (0b1000 0000)

   D6 pin 設為 low (0b1000 0000)

   D5 pin 設為 low (0b1000 0000)

   D4 pin 設為 low (0b1000 0000)

2. 用EN pin送出確認信號通知LCD說資料已經準備好了:

   EN pin 先設為high

   等一下

   EN pin 設為low

傳送前四bit -- CODE

```
GPIO.output(EN,0)
GPIO.output(RW,0)
GPIO.output(RS,0)
GPIO.output(D7, 1 if (0x80 & cmd) else 0)
GPIO.output(D6, 1 if (0x40 & cmd) else 0)
GPIO.output(D5, 1 if (0x20 & cmd) else 0)
GPIO.output(D4, 1 if (0x10 & cmd) else 0)

GPIO.output(EN,1)
sleep(0.000001)
GPIO.output(EN,0)
sleep(0.000001)
```

傳送後四bit

3. 第二次寫入，準備好指令後面四個bit要寫入的指令，RS,RW都不動

   D7~D4 分別對應"指令"(0x80 == 0b10000000)的 bit3 ~ bit0

   所以:

   D7 pin 設為 low (0b1000 0000)

   D6 pin 設為 low (0b1000 0000)

   D5 pin 設為 low (0b1000 0000)

   D4 pin 設為 low (0b1000 0000)

4. 用EN pin送出確認信號通知LCD說資料已經準備好了:

   EN pin 先設為high

   等一下

   EN pin 設為low

傳送後四bit -- CODE

```
GPIO.output(D7, 1 if (0x08 & cmd) else 0)
GPIO.output(D6, 1 if (0x04 & cmd) else 0)
GPIO.output(D5, 1 if (0x02 & cmd) else 0)
GPIO.output(D4, 1 if (0x01 & cmd) else 0)

GPIO.output(EN,1)
sleep(0.000001)
GPIO.output(EN,0)
sleep(0.00005)
```

- 指令"0x80"寫入之後，就會把游標移動到顯示螢幕上第一行第一格的位置。

- 此時要在這個位置顯示ASCII字元 "A“
- ASCII表中的16進位值 ”A” 為 0x41 二進位為 0b0100 0001

1. 第一次寫入，準備好要寫入的前面四個bit:

   EN pin 設為 low

   RW pin 設為 low ==> 跟LCD說這是一個寫入的動作

   RS pin 設為 high ==> 跟LCD說這次的寫入的 "0x41" 是"資料"

   D7~D4 分別對應"資料"(0x41 == 0b0100 0001)的 bit7 ~ bit4

   所以:

   D7 pin 設為 low (0b0100 0001)

   D6 pin 設為 high (0b0100 0001)

   D5 pin 設為 low (0b0100 0001)

   D4 pin 設為 low (0b0100 0001)

2. 用EN pin送出確認信號通知LCD說資料已經準備好了:

   EN pin 先設為high

   等一下

   EN pin 設為low

3. 第二次寫入，準備好指令後面四個bit要寫入的資料，RS,RW都不動

   D7~D4 分別對應"資料"(0x41 == 0b01000001)的 bit3 ~ bit0

   所以:

   D7 pin 設為 low (0b0100 0001)

   D6 pin 設為 low (0b0100 0001)

   D5 pin 設為 low (0b0100 0001)

   D4 pin 設為 high (0b0100 0001)

4. 用EN pin送出確認信號:

   EN pin 先設為high

   等一下

   EN pin 設為low

Write Data python code

```
GPIO.output(EN,0)
GPIO.output(RW,0)
GPIO.output(RS,1)
GPIO.output(D7, 1 if (0x80 & data) else 0)
GPIO.output(D6, 1 if (0x40 & data) else 0)
GPIO.output(D5, 1 if (0x20 & data) else 0)
GPIO.output(D4, 1 if (0x10 & data) else 0)

GPIO.output(EN,1)
sleep(0.000001)
GPIO.output(EN,0)
sleep(0.000001)
```

```
GPIO.output(D7, 1 if (0x08 & data) else 0)
GPIO.output(D6, 1 if (0x04 & data) else 0)
GPIO.output(D5, 1 if (0x02 & data) else 0)
GPIO.output(D4, 1 if (0x01 & data) else 0)

GPIO.output(EN,1)
sleep(0.000001)
GPIO.output(EN,0)
sleep(0.00005)
```

開機動作

LCD 開機動作

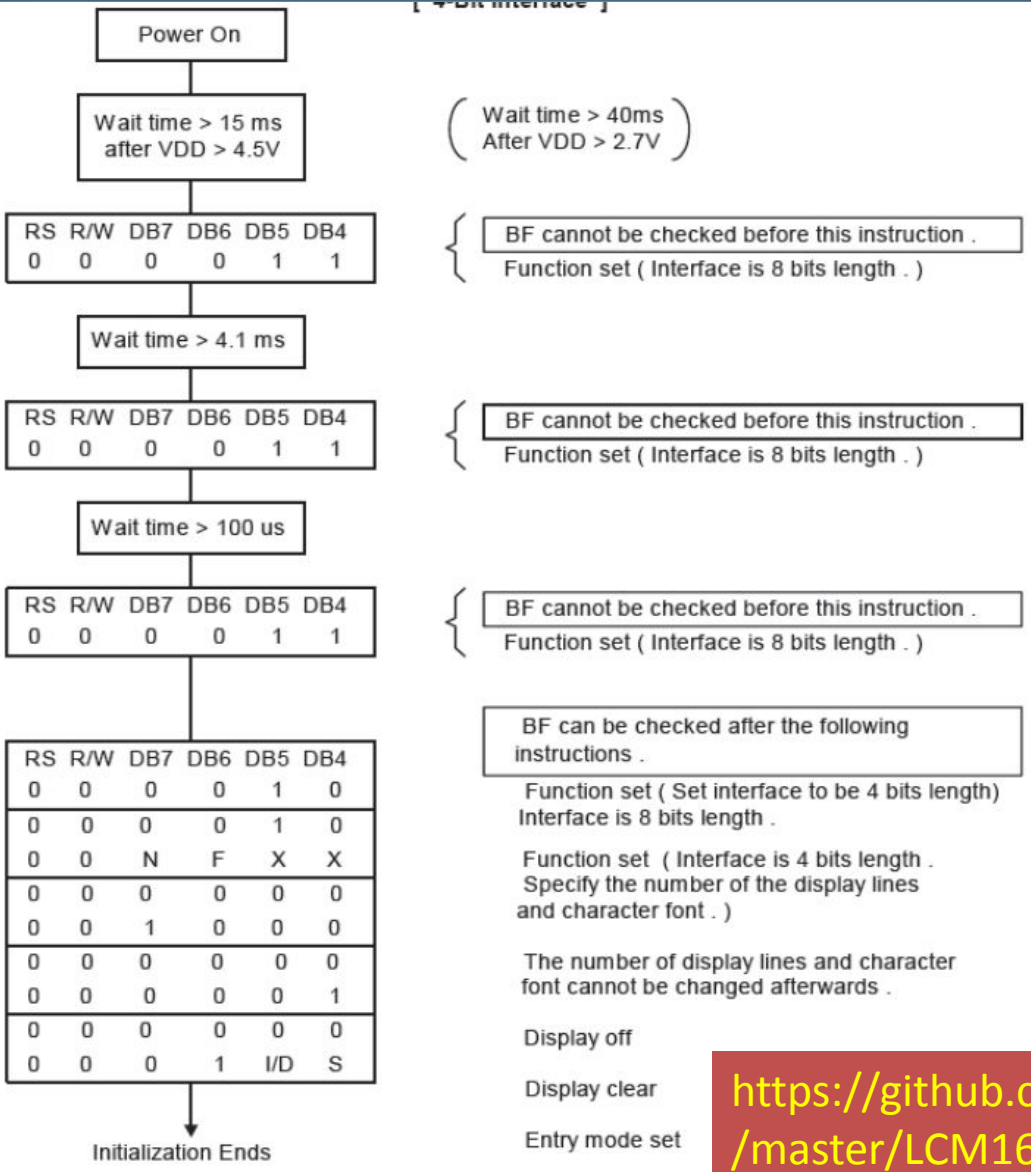
跟據Datasheet所表示，我們正在用的四條資料線+RS+RW+EN的接線接法是叫做4bit模式，而嫌GPIO太多的八條資料線+RS+RW+EN的接線接法叫做8bit模式。

而在一開始的時候要怎麼樣才能告訴LCD模組我們打算用4bit模式還是8bit模式來操作LCD? 這裡的主題是4bit，所以就不管8bit模式怎麼開機。而且4bit搞定之後，通常也不會想用8bit模式了。
還是跟據Datasheet表示，LCD接上電VDD電壓升到4.4V之後要先等待15ms然後...

---

1. EN,RS,RW,D7~D4 都設為 low。
2. EN 設 high -> 等一下 -> EN 設 low
3. D7~D4分別設為 low, low, high, low (0b0010)。
4. EN 設 high -> 等一下 -> EN 設 low
5. 等待 5ms(>4.1ms)
6. EN 設 high -> 等一下 -> EN 設 low
7. 等待 200us(>100us)
8. EN 設 high -> 等一下 -> EN 設 low
9. 寫入指令 0x28 (0b0010 1000) --> 設定為4bit操作，兩行顯示模式
10. 寫入指令 0x0c (0b0000 1100) --> 設定為每寫入一次資料，游標位置向右移一格
11. 寫入指令 0x01 (0b0000 0001) --> 清除螢幕並游標歸回原點。

開機動作



https://github.com/itrobotics/python_LCM_control/blob/master/LCM1602.py

http://blog.ittraining.com.tw/2015/02/raspberry-pi-b-python-lcd-16x2-hd44780_6.html

練習

- 在LCM模組上第一行和第二行顯示Hello World。
- 完善LCM Class:
  1) .print_line_1(msg)
  2) .print_line_2(msg)
- 配合 subprocess.check_output() 呼叫指令 ifconfig 得到 IP 位置，並將其顯示在LCM模組之上。
- 將其設定為開機即顯示(hint: /etc/)

https://github.com/itrobotics/python_LCM_control/blob/master/LCM1602.py

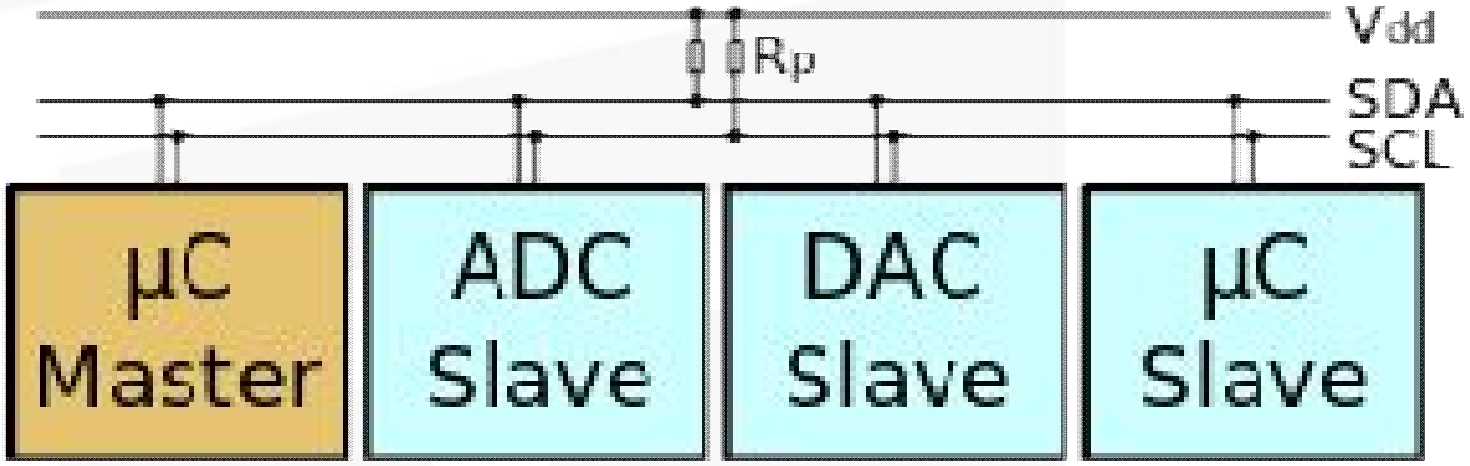
```
1048   1592   1953   24     46    592   9     crypto
1055   1596   1991   25     47    6     910   devices
1061   1599   2      26     48    60    956   device-tree
11     16     20     27     49    621   961   diskstats
110    1625   2092   28     5     626   965   driver
1101   1626   2107   29     51    659   972   execdomains
1102   1632   2109   3      52    681   973   fb
1103   1691   2110   30     523   685   975   filesystems
1116   1694   2134   31     525   693   989   fs
112    1701   2156   32     527   7     994   interrupts
pi@raspberrypi /proc $
```

> sudo raspi-config

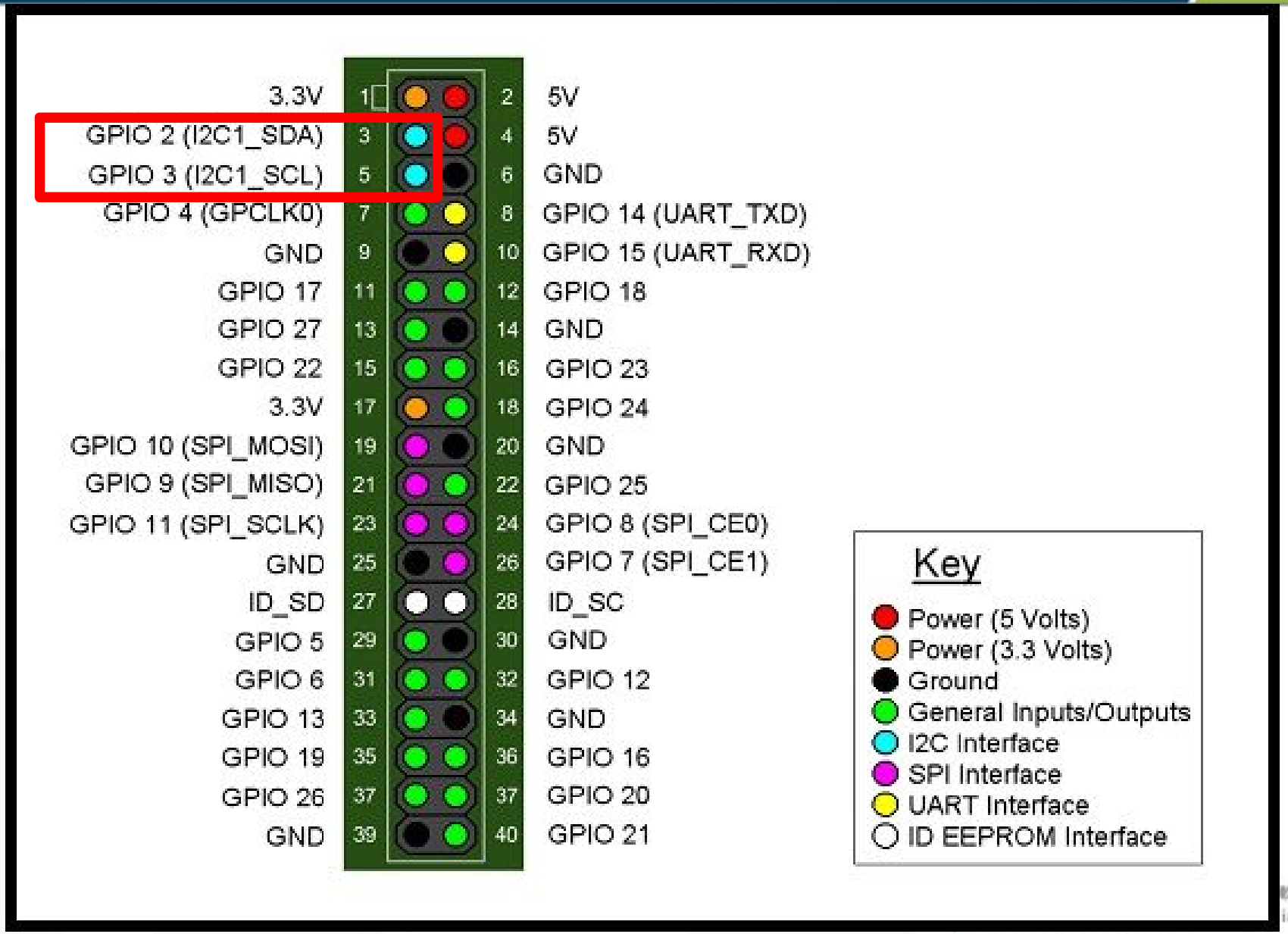
8 Advanced Options

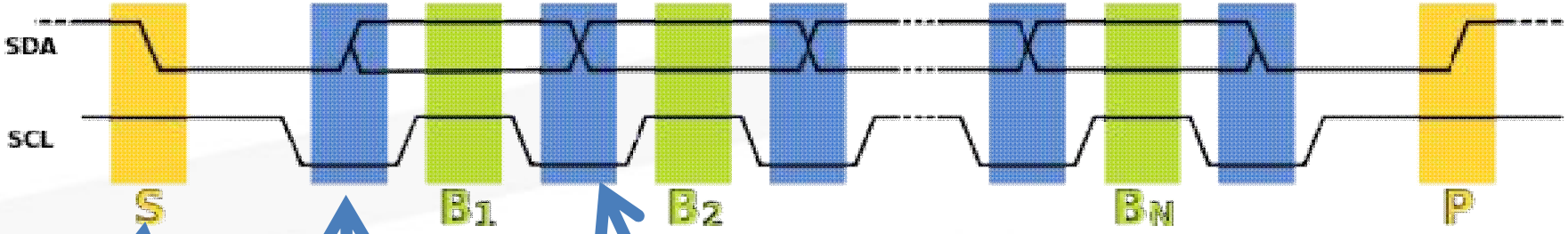
A5 Device Tree

I²C（Inter-Integrated Circuit）



Raspberry Pi 硬體控制





開始傳送

SDA 準備資料

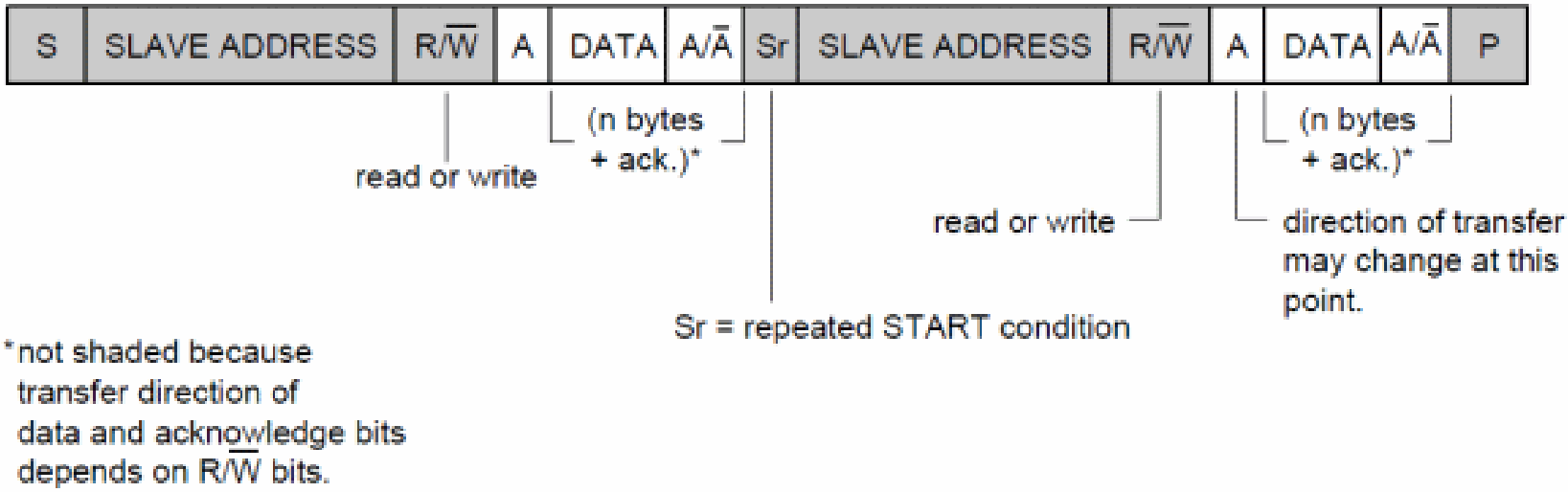
傳送bit

SDA 準備資料

傳送一個bit

結束傳送

I²C Protocol

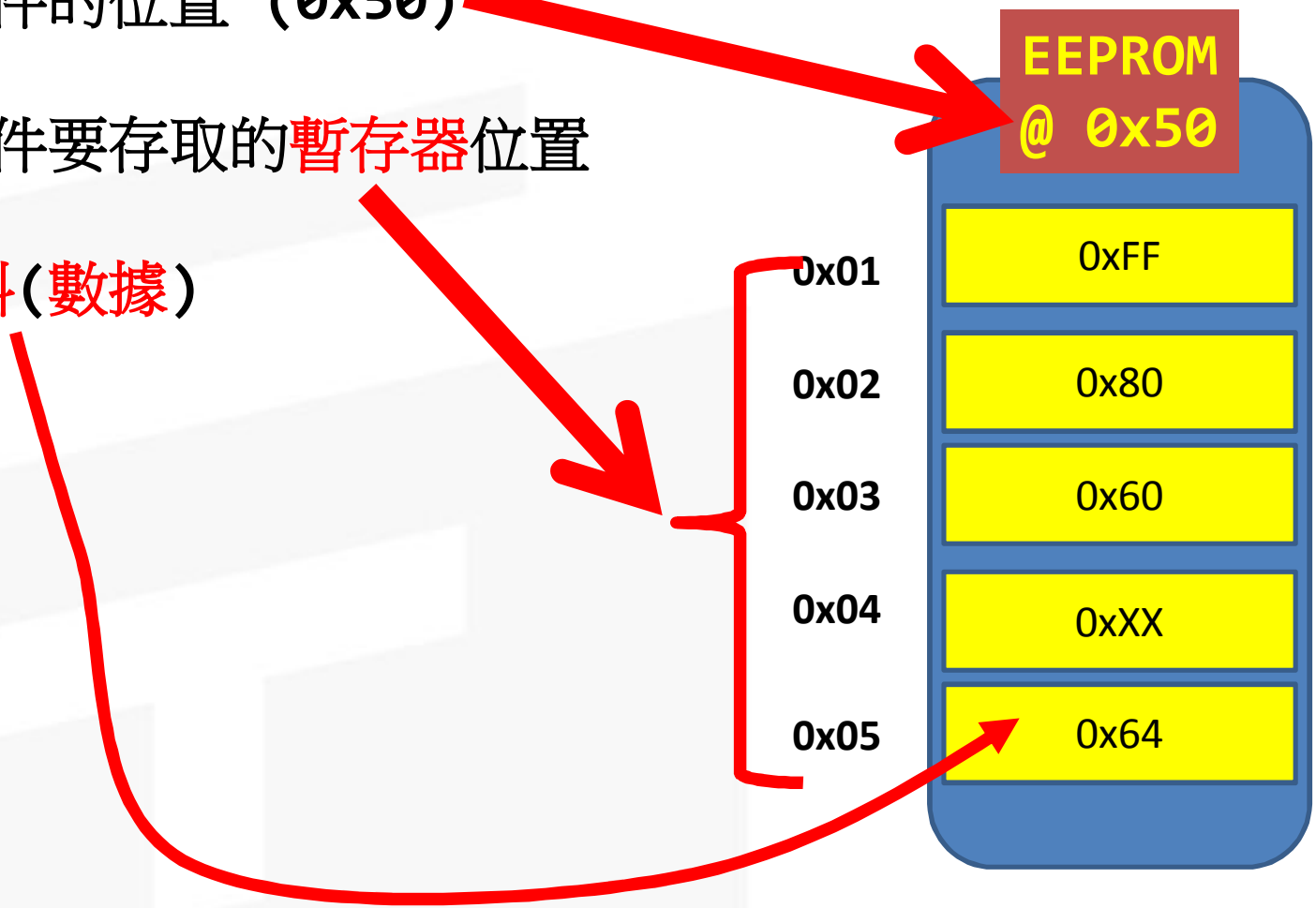


Python I²C API 使用要點

指定 I²C 元件的位置 (0x50)

指定 I²C 元件要存取的暫存器位置

要存取的資料(數據)



I²C 相關模組安裝

- apt-get install python3-smbus

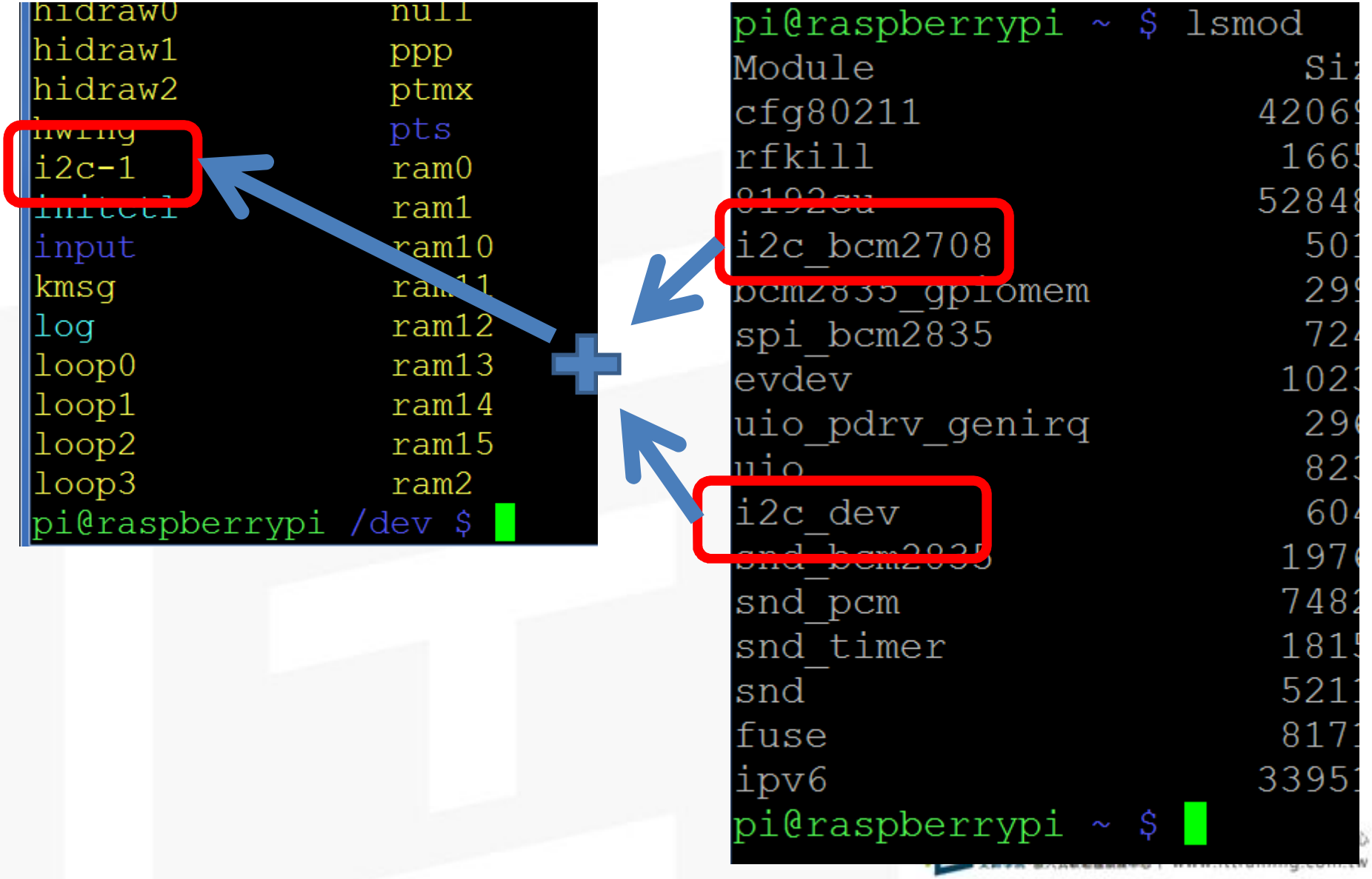
載入i2c module

- modprobe i2c-dev

使用I2C工具

- i2cdetect -y 1
- i2cdump -y 1 <指定i2c-addr>
- i2cget -y 1 <指定i2c-addr> <暫存器位置>
- i2cset -y 1 <指定i2c-addr> <暫存器位置> <byte>

Raspberry Pi – Raspbian I²C 驅動



Python 使用API(from smbus)


```
import smbus
from time import sleep
i2c_addr = 0x50
bus = smbus.SMBus(1)
```

I²C元件的位置

暫存器起始位置

要寫入的資料

i2c protocol access

```
bus.write_i2c_block_data(i2c_addr, 0x00, [0,1,2,3,4,5])
```

```
bus.read_i2c_block_data(i2c_addr, 0x00 , 10)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

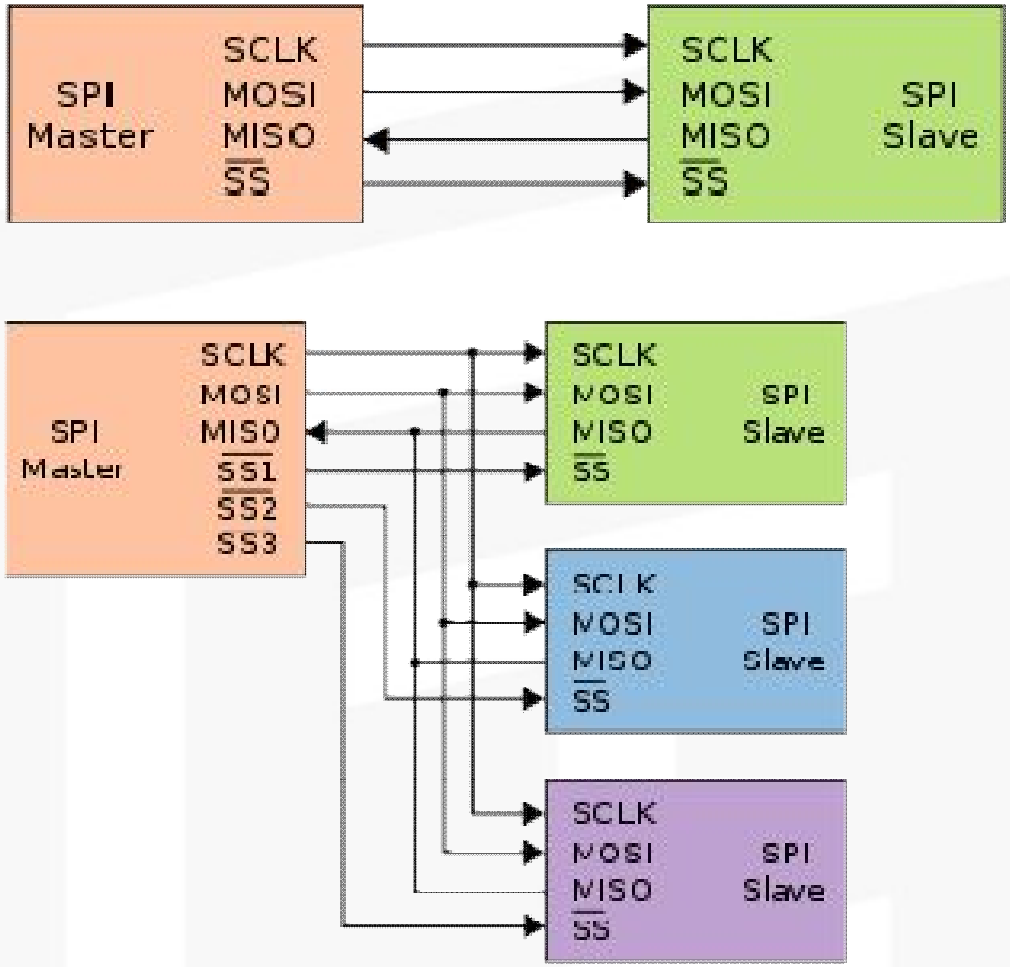
讀取到的資料

要讀取的資料數量

練習

- 將 ”Hello World” 字串寫入EEPROM 的起始暫存器位置: 0x64
- EEPROM 的大小為 256 byte ，將所有的內容讀取出來

SPI（Serial Peripheral Interface Bus，SPI）



MOSI :

Master Output Slave Input

MISO :

Master Input Slave Output

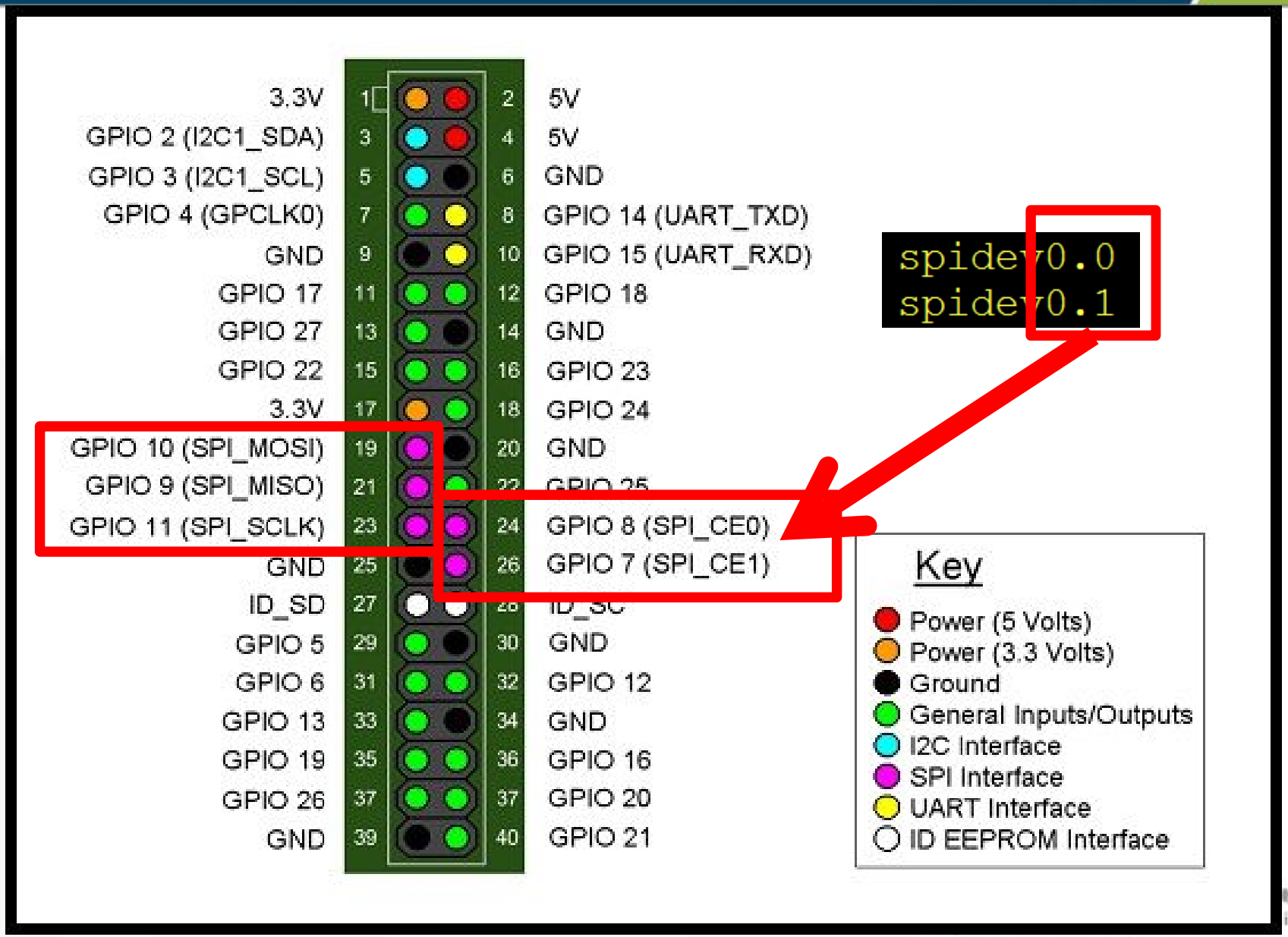
SCLK:

Serial Clock

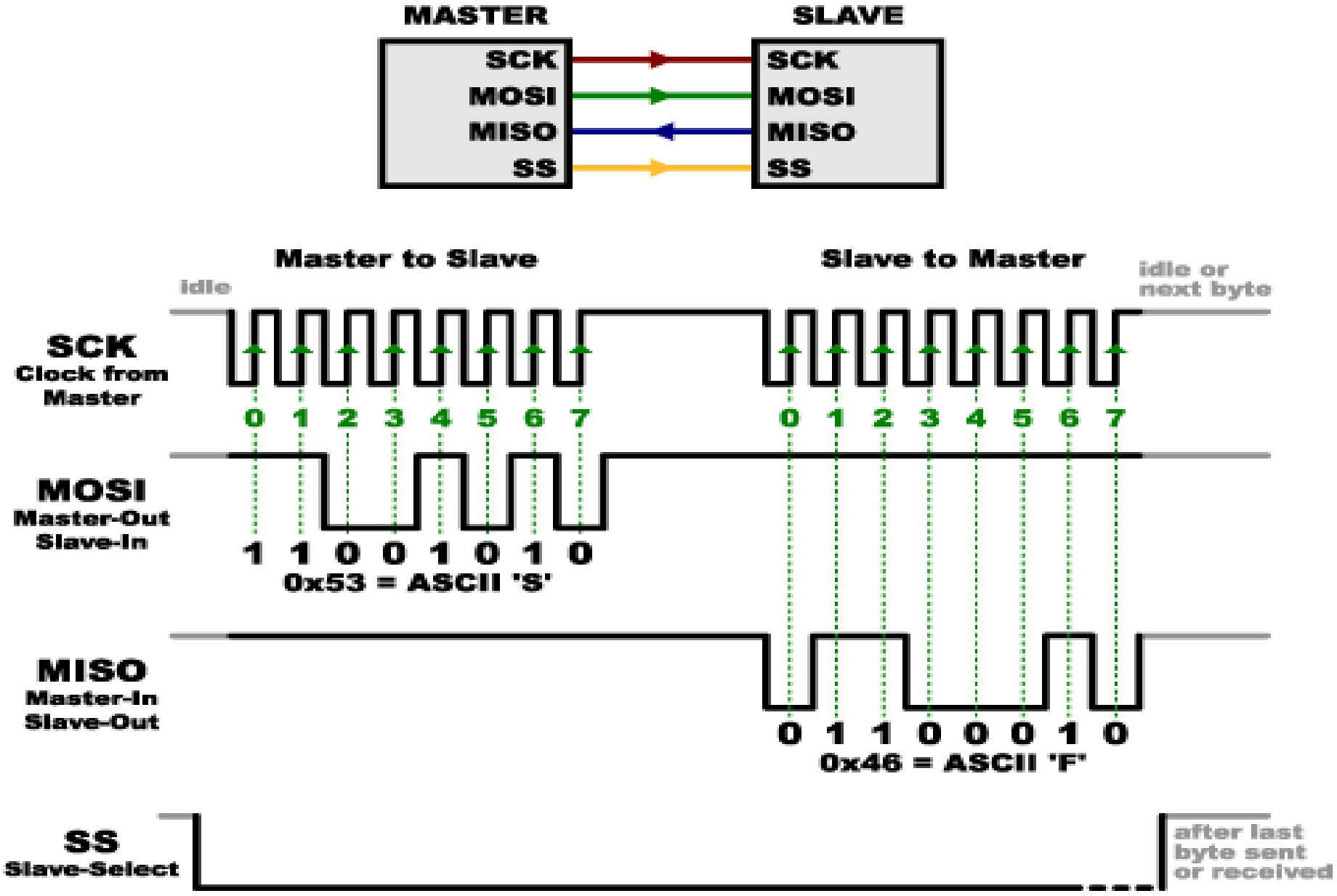
SS: (CS)

Slave Select

Raspberry Pi 硬體控制



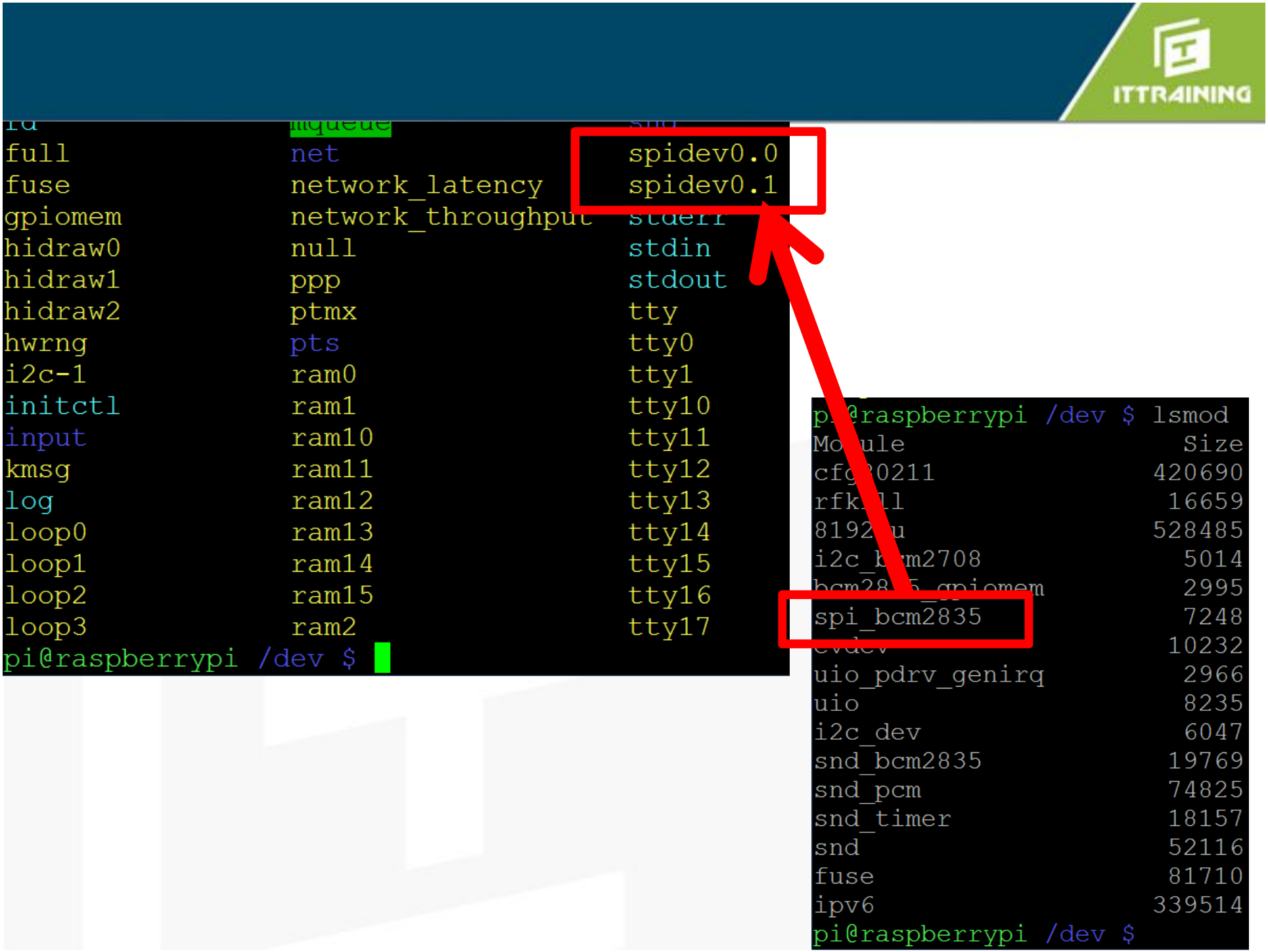
SPI transaction



SPI 模組安裝

- git clone https://github.com/doceme/py-spidev.git
- cd py-spidev
- sudo python3 setup.py install

- apt-get install python3-dev


full
fuse
gpiomem
hidraw0
hidraw1
hidraw2
hwrng
i2c-1
initctl
input
kmsg
log
loop0
loop1
loop2
loop3
net
network_latency
null
ppp
ptmx
pts
ram0
ram1
ram10
ram11
ram12
ram13
ram14
ram15
ram2
spidev0.0
spidev0.1
stdin
stdout
tty
tty0
tty1
tty10
tty11
tty12
tty13
tty14
tty15
tty16
tty17
pi@raspberrypi /dev $
pi@raspberrypi /dev $ lsmod
Size
420690
16659
528485
5014
2995
7248
10232
2966
8235
6047
19769
74825
18157
52116
81710
339514
spi_bcm2835
uio_pdrv_genirq
uio
i2c_dev
snd_bcm2835
snd_pcm
snd_timer
snd
fuse
ipv6
pi@raspberrypi /dev $

Python 使用API SPI (from spidev)

SPI access

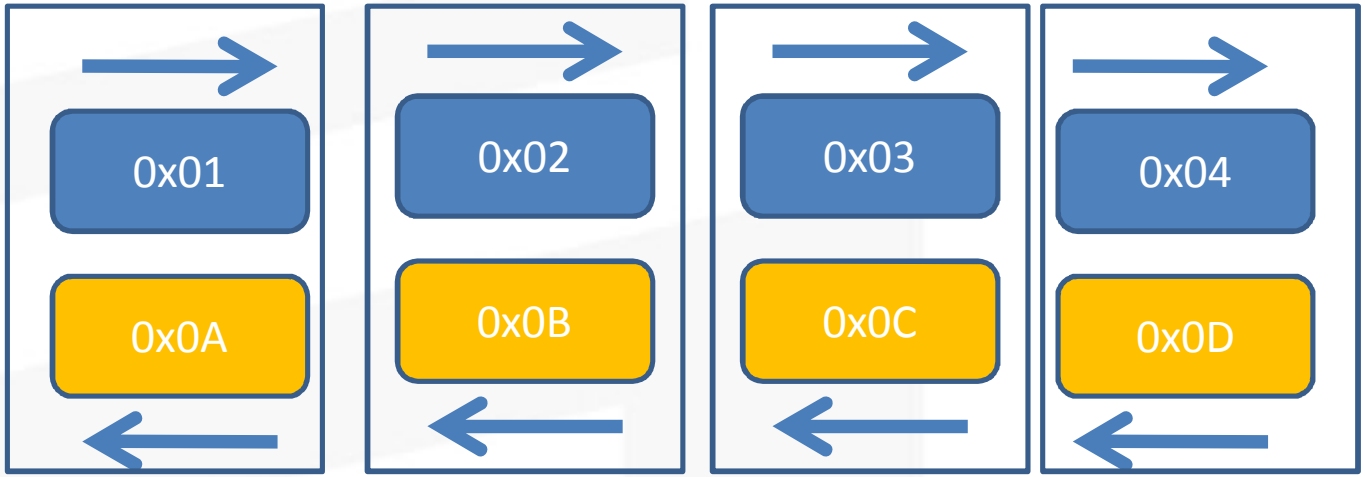
```
spidev0.0
spidev0.1
```

```
from spidev import SpiDev
from time import sleep
spi = SpiDev()
spi.open(0,1)
```

```
spi.xfer2([0xF5, 0x00])
```

```
[0, 1]
```

```
spi.xfer2([0x01,0x02,0x03,0x04])
```




```
[0x0A,0x0B,0x0C,0x0D]
```

CC2500 Operation

READ Command

7	6	5	4	3	2	1	0	
1	1							0xC0
1	0							0x80
		1	1	0	1	0	1	0x35

WRITE Command

7	6	5	4	3	2	1	0	
1	1							0xC0
1	0							0x80
		1	1	0	1	0	1	0x35

http://www.ti.com/lit/ds/symlink/cc2500.pdf

CC2500 Commands

```
from spidev import SpiDev
cc2500 = SpiDev()
cc2500.open(0,1)
```

CC2500 COMMAAND

```
# get STATE REG @ 0x35 (0b0011 0101)
# command 0xC0         (0b1100 0000)
cmd = 0xC0 | 0x35
cc2500.xfer2([cmd,0x00])
```

```
cc2500.xfer2([0x30]) # send command SRES
cc2500.xfer2([0x34]) # send command SRX
cc2500.xfer2([0x35]) # send command STX
cc2500.xfer2([0x36]) # send command SIDLE
```

	Write		Read		
	Single byte	Burst	Single byte	Burst	
	+0x00	+0x40	+0x80	+0xC0	
0x30	SRES		SRES	PARTNUM	Command strobes, status registers (read only) and multi byte registers
0x31	SFSTXON		SFSTXON	VERSION	
0x32	SXOFF		SXOFF	FREQEST	
0x33	SCAL		SCAL	LQI	
0x34	SRX		SRX	RSSI	
0x35	STX		STX	MARCSTATE	
0x36	SIDLE		SIDLE	WORTIME1	
0x37				WORTIME0	
0x38	SWOR		SWOR	PKTSTATUS	
0x39	SPWD		SPWD	VCO_VC_DAC	
0x3A	SFRX		SFRX	TXBYTES	
0x3B	SFTX		SFTX	RXBYTES	
0x3C	SWORRST		SWORRST	RCCTRL1_STATUS	
0x3D	SNOP		SNOP	RCCTRL0_STATUS	
0x3E	PATABLE	PATABLE	PATABLE	PATABLE	
0x3F	TX FIFO	TX FIFO	RX FIFO	RX FIFO	

cc2500 reset

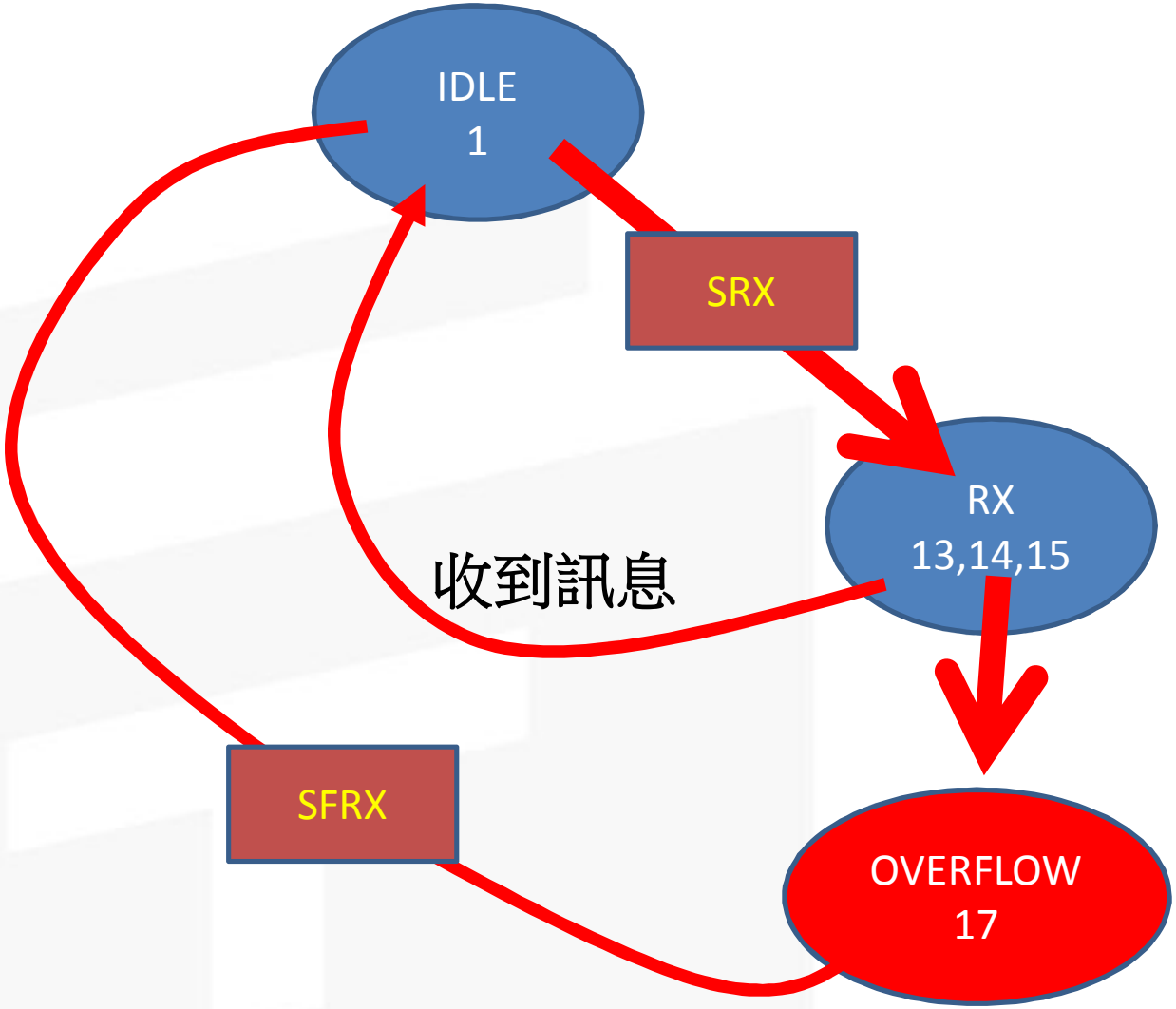
```
def reset(cc2500):
    #SRES
    cc2500.xfer2([0x30])

    reg_config = [
        0x29,0x2E,0x06,0x07,0xD3,0x91,0x61,0x04,
        0x45,0x00,0x00,0x09,0x00,0x5E,0xC4,0xEC,
        0x2C,0x22,0x73,0x22,0xF8,0x01,0x07,0x00,
        0x18,0x1D,0x1C,0xC7,0x00,0xB2,0x87,0x6B,
        0xF8,0xB6,0x10,0xEB,0x0B,0x1D,0x11,0x41,
        0x00,0x59,0x7F,0x3C,0x88,0x31,0x0B
    ]

    for reg, val in enumerate(reg_config):
        cc2500.xfer2([reg, val])

    #SIDLE
    cc2500.xfer2([0x36])
    #SFRX
    cc2500.xfer2([0x3A])
    #SFTX
    cc2500.xfer2([0x3B])
```

Receive Message



write data into FIFO

```
msg = "Hello World"
data = [ord(i) for i in msg]
data = [0b01000000 | 0x3F, len(data)] + data

cc2500.xfer2(data)  # Write into FIFO
cc2500.STX()  # send data via RF
```

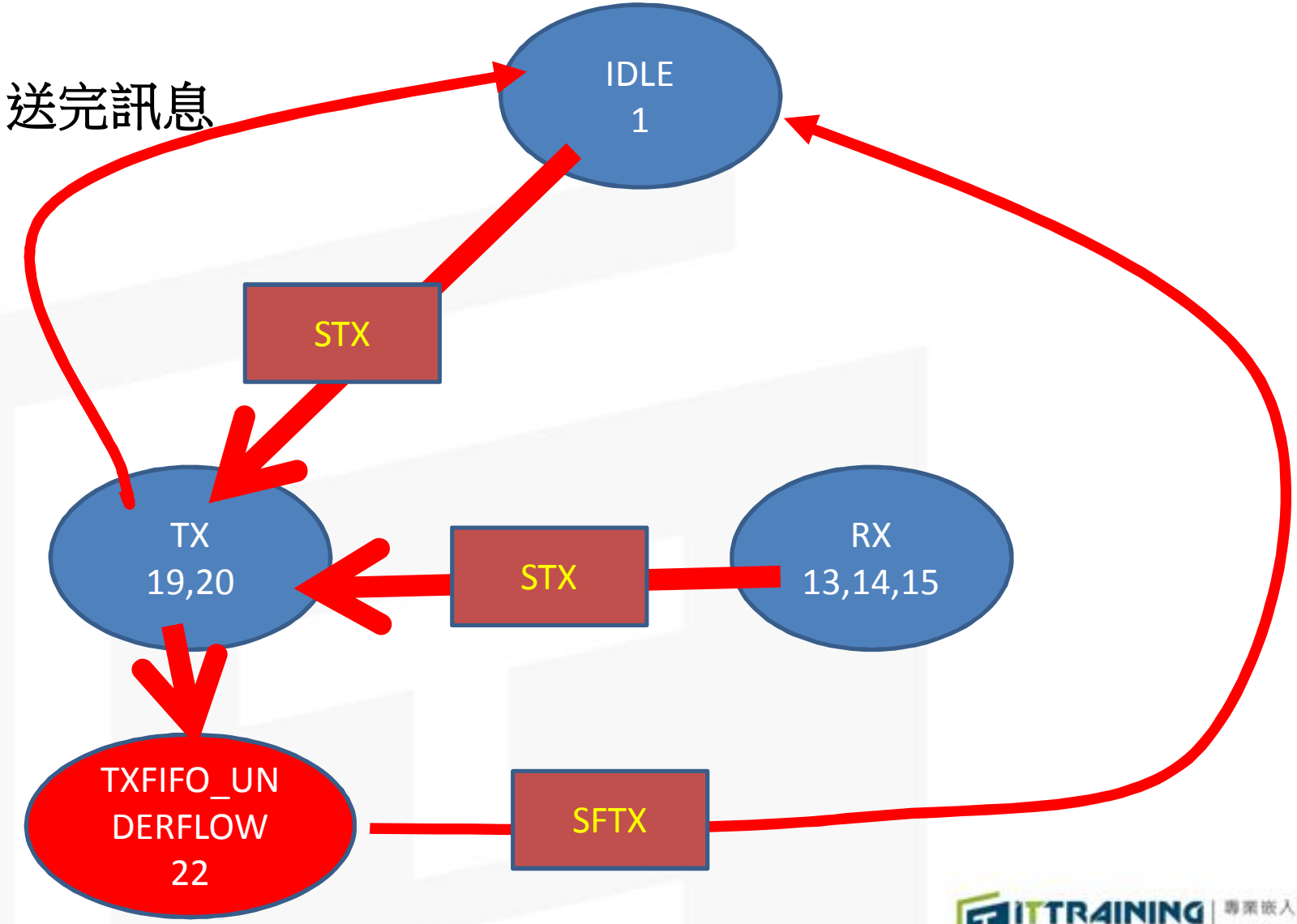
read data from FIFO

```
rx_len = xfer2([0xC0 | 0x3B, 0x00])[1]
data = cc2500.xfer2([0xC0 | 0x3F for i in range(rx_len)])
print(data)
bytes(data).decode()
```

練習

- 用 while loop 監聽訊息

Send Message



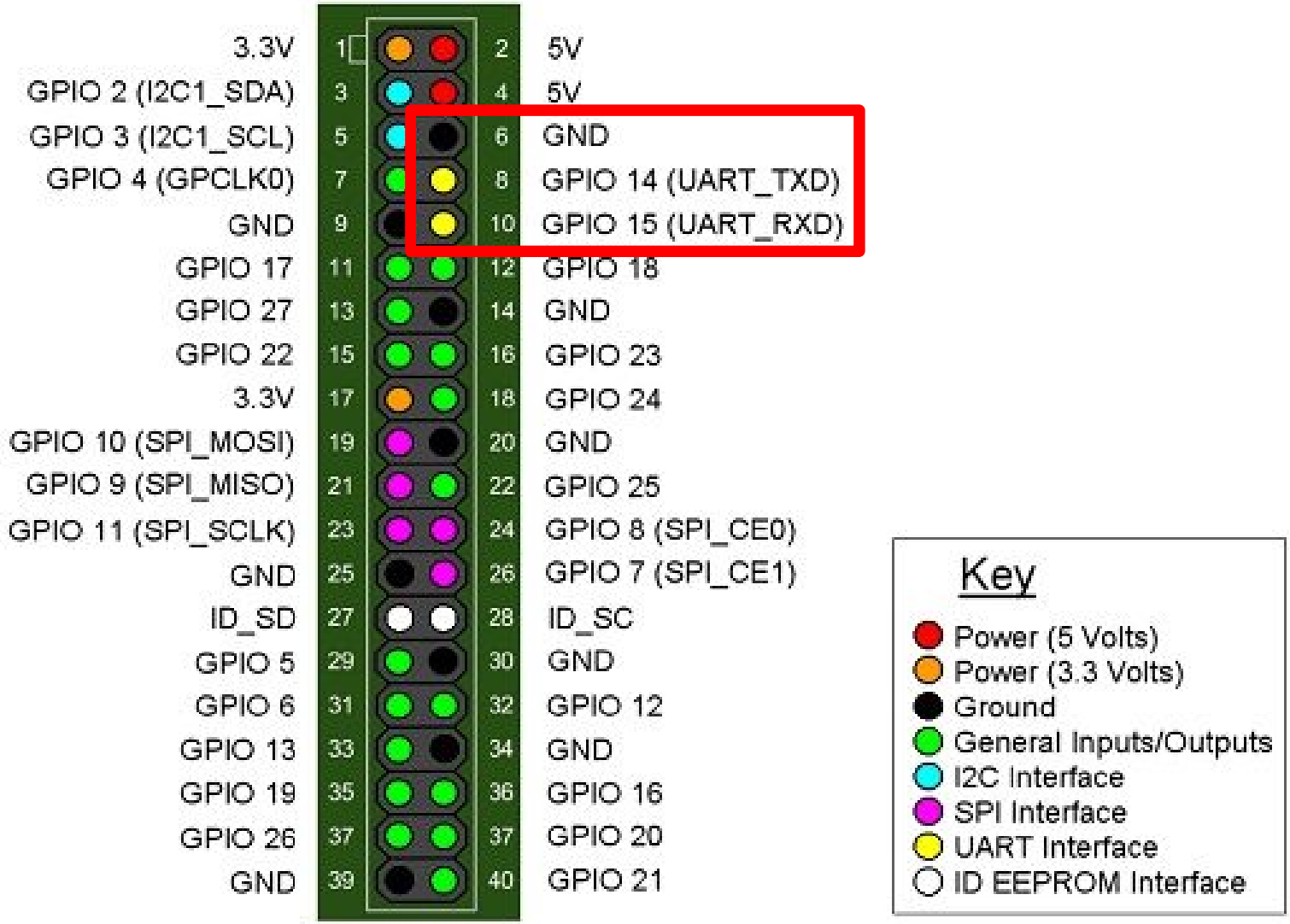
練習

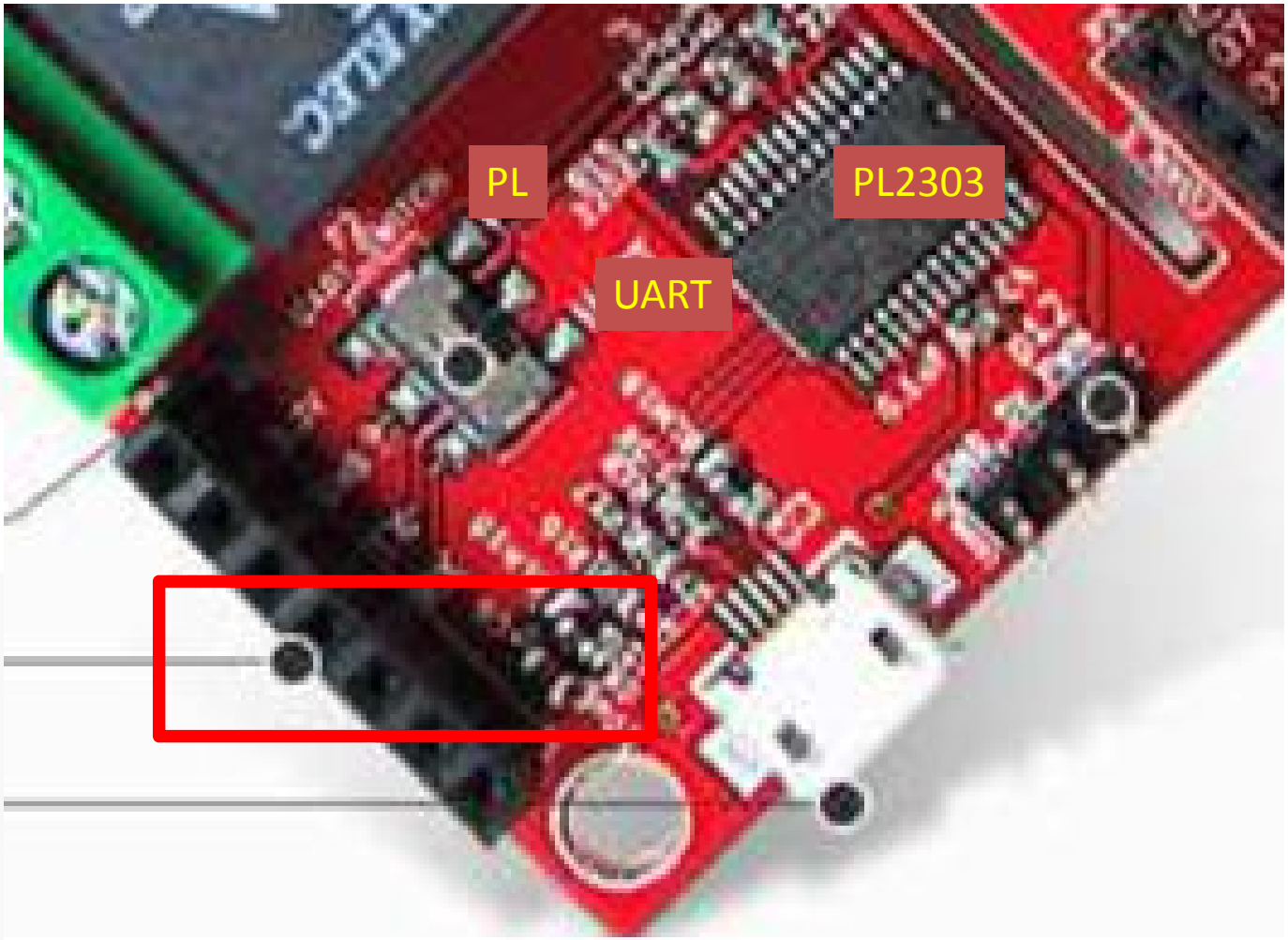
- 寫一個 function 傳送訊息

練習

- 實作一個thread 監聽訊息，主程式等待使用者輸入訊息，使用者輸入訊息之後傳送出去。

UART


3.3V 1 2 5V
GPIO 2 (I2C1_SDA) 3 4 5V
GPIO 3 (I2C1_SCL) 5 6 GND
GPIO 4 (GPCLK0) 7 8 GPIO 14 (UART_TXD)
GND 9 10 GPIO 15 (UART_RXD)
GPIO 17 11 12 GPIO 18
GPIO 27 13 14 GND
GPIO 22 15 16 GPIO 23
3.3V 17 18 GPIO 24
GPIO 10 (SPI_MOSI) 19 20 GND
GPIO 9 (SPI_MISO) 21 22 GPIO 25
GPIO 11 (SPI_SCLK) 23 24 GPIO 8 (SPI_CE0)
GND 25 26 GPIO 7 (SPI_CE1)
ID_SD 27 28 ID_SC
GPIO 5 29 30 GND
GPIO 6 31 32 GPIO 12
GPIO 13 33 34 GND
GPIO 19 35 36 GPIO 16
GPIO 26 37 37 GPIO 20
GND 39 40 GPIO 21
Key
Power (5 Volts)
Power (3.3 Volts)
Ground
General Inputs/Outputs
I2C Interface
SPI Interface
UART Interface
ID EEPROM Interface


PL
PL2303
UART

UART SETUP

- Baud Rate: 9600, 19200, 38400, 57600, 115200
- ByteSize: 7, 8
- Parity (同位檢查): Odd, Even, None
- Stop Bit: 1, 1.5,2

115200, 8, N, 1

腳位	簡寫	意義	說明
Pin1	CD	Carrier Detect	數據機通知電腦有載波被偵測到。
Pin2	RXD	Receiver	接收資料。
Pin3	TXD	Transmit	傳送資料。
Pin4	DTR	Data Terminal Ready	電腦告訴數據機可以進行傳輸。
Pin5	GND	Ground	地線。
Pin6	DSR	Data Set Ready	數據機告訴電腦一切準備就緒。
Pin7	RTS	Request To Send	電腦要求數據機將資料送出。
Pin8	CTS	Clear To Send	數據機通知電腦可以傳資料過來。
Pin9	RI	Ring Indicator	數據機通知電腦有電話進來。

Python Serial module

ITTRAINING

```
import serial
```

```
ser = serial.Serial("/dev/ttyAMA0",115200, 8, 'N' , 1,timeout=0.5)
```

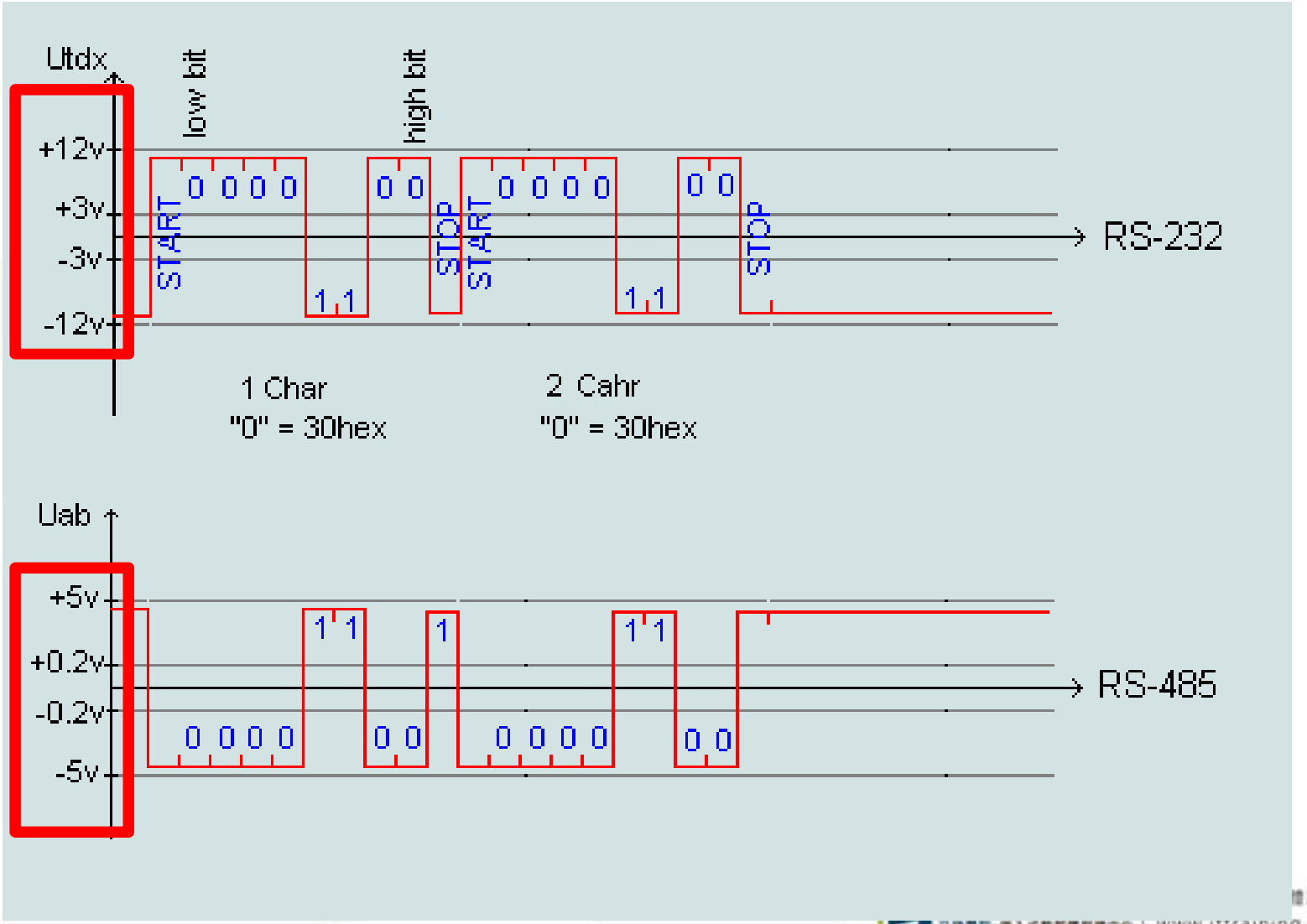
```
ser.read(size=1)
ser.write(b"hello world")
```

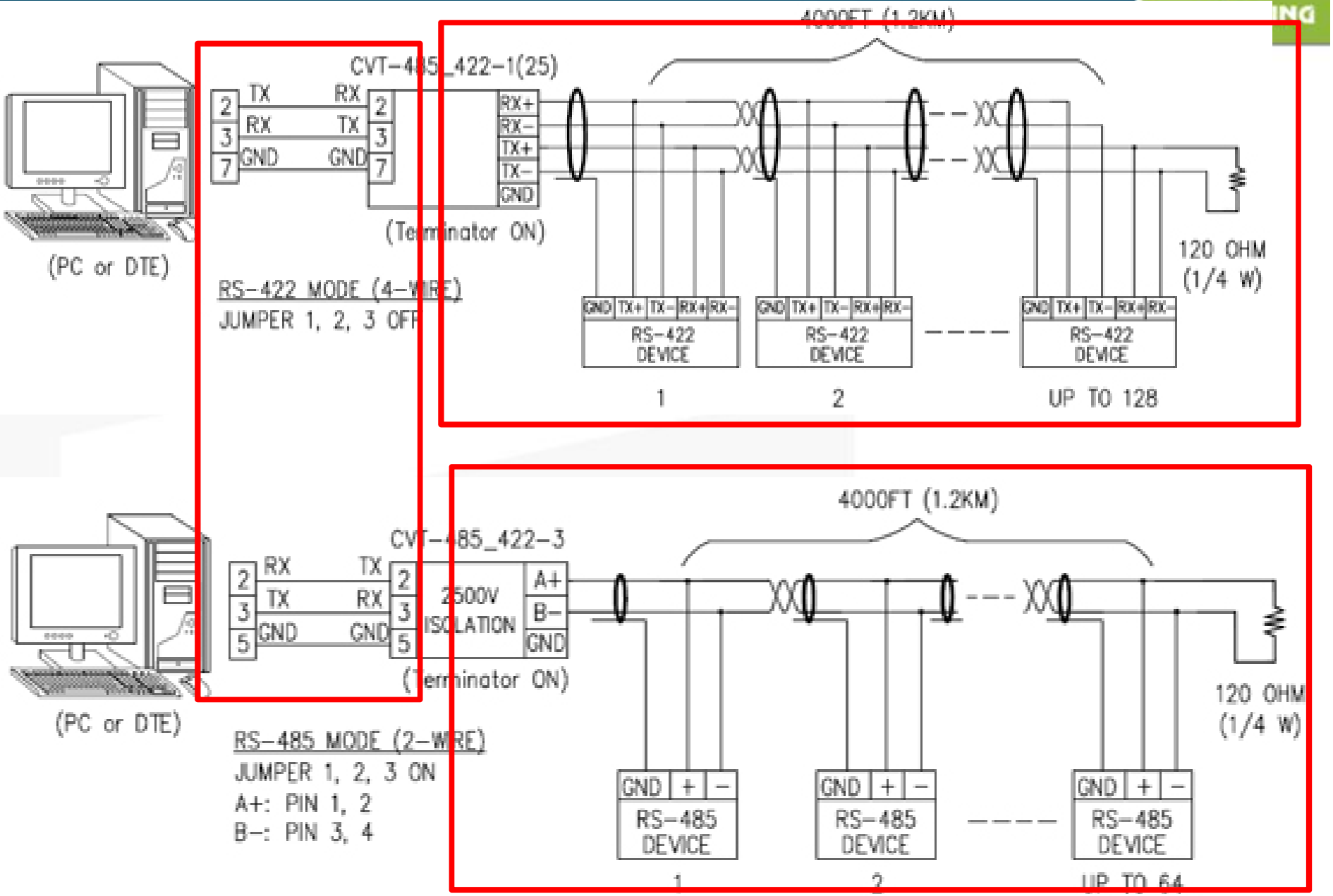
```
ser.flush()
ser.readline()
ser.readlines()
```

```
ser.setRTS(level=True)
```

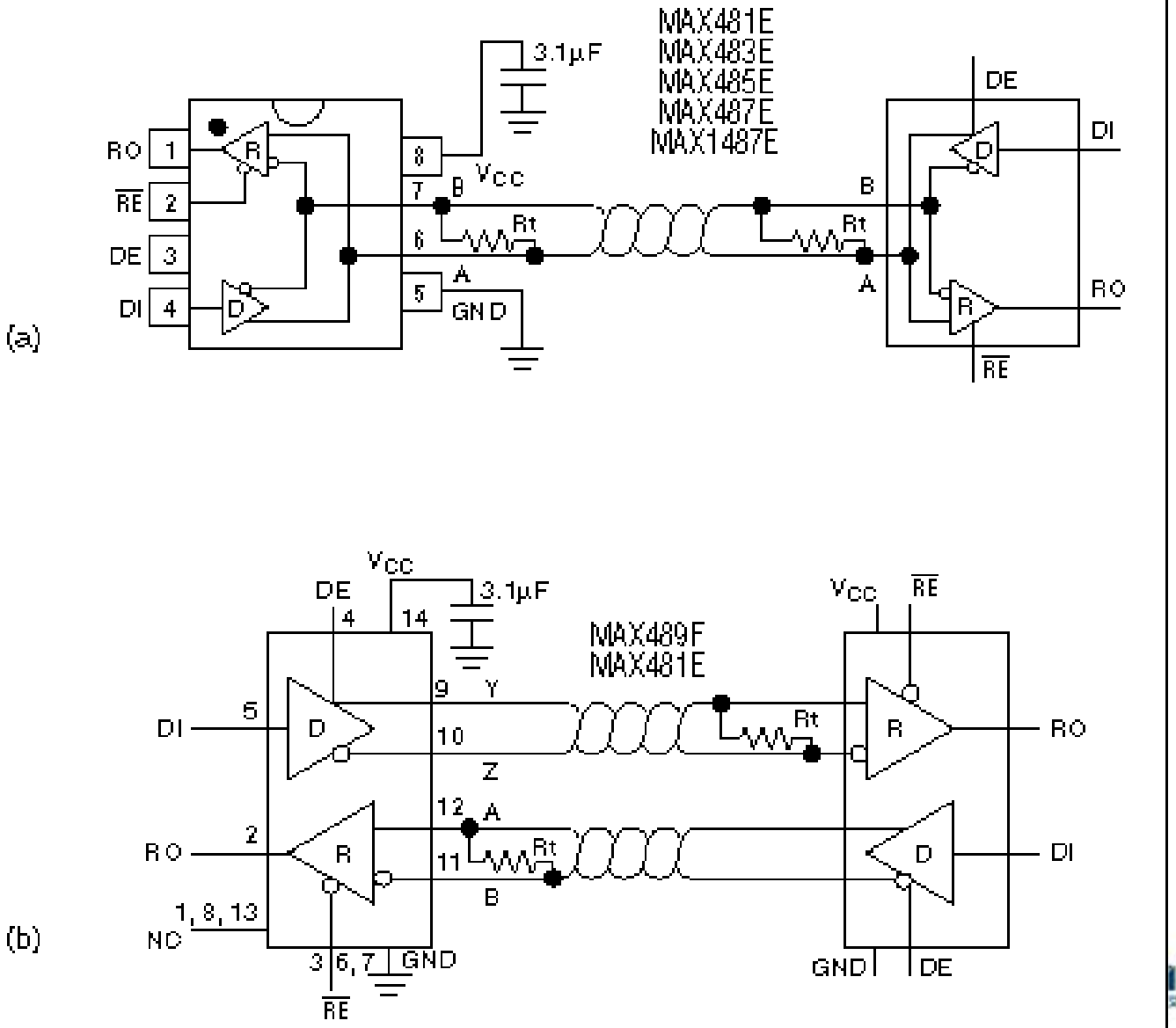
http://pythonhosted.org/pyserial/

RSS232 and RS485(RS422)




4000FT (1.2KM)
CVT-485_422-1(25)
2 TX RX 2
3 RX TX 3
7 GND GND 7
RX+
RX-
TX+
TX-
GND
(Terminator ON)
(PC or DTE)
RS-422 MODE (4-WIRE)
JUMPER 1, 2, 3 OFF
GND TX+ TX- RX+ RX-
RS-422 DEVICE
1
2
UP TO 128
120 OHM (1/4 W)
4000FT (1.2KM)
CVT-485_422-3
2 RX TX 2
3 TX RX 3
5 GND GND 5
2500V ISOLATION
A+
B-
GND
(Terminator ON)
(PC or DTE)
RS-485 MODE (2-WIRE)
JUMPER 1, 2, 3 ON
A+: PIN 1, 2
B-: PIN 3, 4
GND + -
RS-485 DEVICE
1
2
UP TO 64
120 OHM (1/4 W)

RS422, RS485



Modbus RTU Protocol

	Coil Status (1bit)	Input Status (1bit)	Holding Reg (16bits)	Input Reg (16bits)
READ	1	2	3	4
WRITE	5, 15		6, 16	

Function	Description
1	Read Coil Status
2	Read Input Status
3	Read Holding Registers
4	Read Input Registers
5	Force Single Coil
6	Preset Single Register
7	Read Exception Status
8	Diagnostics
9	Program 484
10	Poll 484
11	Fetch Comm. Event Ctr.
12	Fetch Comm. Event Log
13	Program Controller
14	Poll Controller
15	Force Multiple Coils
16	Preset Multiple Registers
17	Report Slave ID
18	Program 884/M84
19	Reset Comm. Link
20	Read General Reference
21	Write General Reference

http://modbus.org/docs/PI_MBUS_300.pdf

Modbus RTU example

QUERY

Field Name	Example (Hex)
Slave Address	11
Function	06
Register Address Hi	00
Register Address Lo	01
Preset Data Hi	00
Preset Data Lo	03
Error Check (LRC or CRC)	—

Python Modbus Module

```
import minimalmodbus
```

```
addr = 10
bus = minimalmodbus.Instrument('/dev/ttyUSB0',addr)
bus.serial.baudrate = 9600
bus.serial.bytesize = 8
bus.serial.parity = minimalmodbus.serial.PARITY_NONE
bus.serial.stopbits = 1
bus.serial.timeout = 0.05
bus.serial.mode = minimalmodbus.MODE_RTU
```

https://minimalmodbus.readthedocs.org/en/master/apiminimalmodbus.html

Modbus RTU: Reg access

example: read reg @ 0x10
default function code: 3

```
try:
    bus.read_register(0x10)
except:
    print("modbus read error")
```

example: write value 255 into reg @ 0x10

```
try:
    bus.write_register(0x10,255)
except:
    print("modbus write error")
```