

19.5 控制 2 個馬達轉速—樹莓派小汽車實例

實驗介紹

本次實驗將透過 Python 程式語言和 L293D 控制 IC 來啟動兩顆馬達，並且分享從無到有自製的樹莓派小汽車的實際例子。

小汽車的部分要如何處理就看各位的預算，市面上可以買到很多小汽車的零件，如果想要用 DIY 的方式，可以把家裡沒有在玩的小汽車重新組裝；如果只是想要用最低的預算達到目的，也可以用壓克力或木板當車子的底座再加上兩顆馬達及輪子；如果只是練習樹莓派實驗，並不一定要購買這些小汽車的零件，執行程式練習就可以。在挑選時要注意的是直流馬達的電力需求，請注意馬達的電力需求不要超過 12V 以上，因為 L293D 最多只能到 12V DC。

實驗硬體設備

- 樹莓派的板子
- 一個 DC 直流馬達，如果能有二個就更好了
- 麵包板
- 接線
- L293D 馬達驅動 IC
- 12V 到 5V 的電池×2
- 小汽車零組件

實驗接線

這裡的電子硬體跟上一個章節很像，不一樣的地方是要控制兩個馬達，樹莓派的電源接到智慧型手機的攜帶型充電器上就可以了。

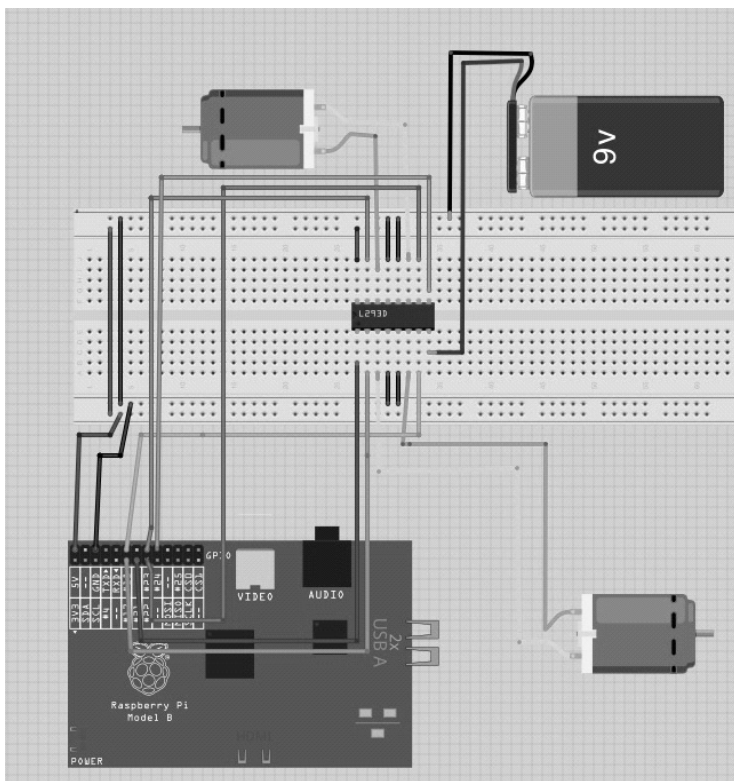


圖 19-14 硬體接線

汽車模組

現在各大拍賣網站上，只要搜尋「“arduino” 汽車」的零件都很容易買到，這樣的硬體也適用在樹莓派上。像本章範例的小汽車就是這樣取得的，如果有 3D 印表機，也可以自己下載和製造，像是「oomlout」先生的「Arduino Controlled Servo Robot」就可以達到這樣的需求，並且能免費下載 3D 模型檔 STL。

網址為：<http://www.thingiverse.com/thing:209>。

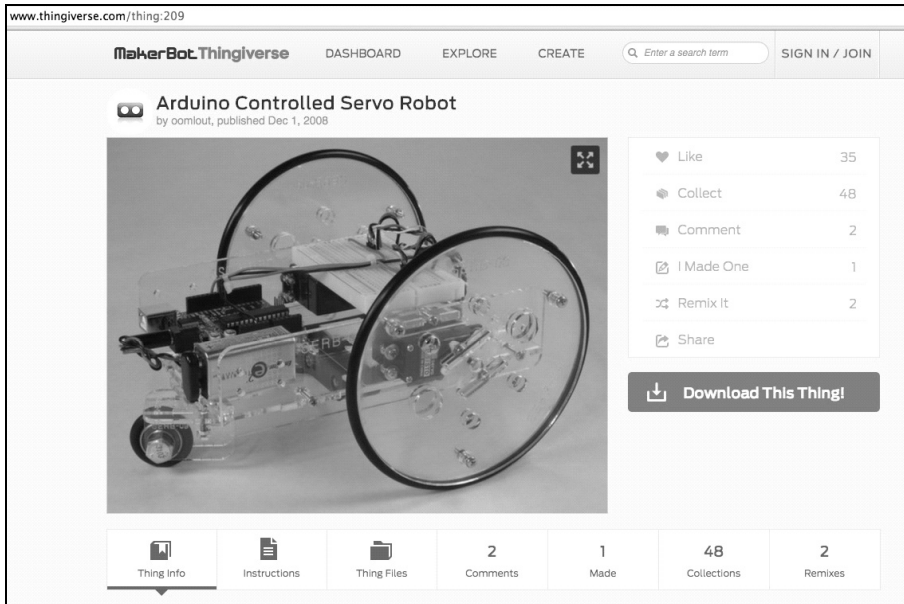


圖 19-15 3D 列印的小汽車模組

實驗步驟

Step 1 小汽車組裝

首先進行小汽車零件組裝。下面的圖片是筆者利用在網路上購買的電子小汽車的零件自行組裝而成的，過程中因為每一個套件差異太大，所以就不多說了，請依照您實際情況自行處理。如果沒有的話，也可以用二個馬達來測試。

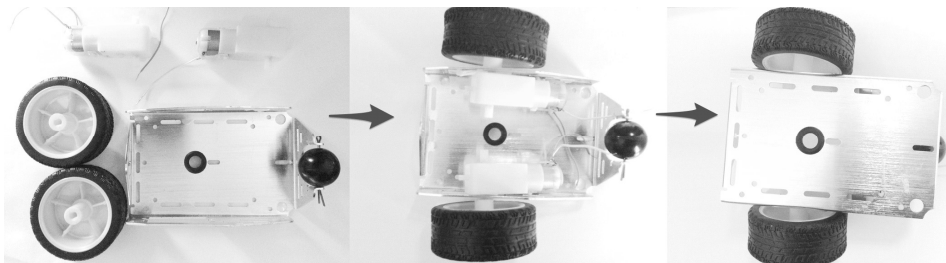


圖 19-16 小汽車組裝

Step 2 硬體接線

以筆者的經驗，在處理較複雜的專案時，都會先在實際的樹莓派上測試硬體和軟體程式後，才會跟其他的硬體作組裝的動作，所以請先用麵包板把線路跟硬體組裝好，不用急著放上小汽車，等軟體測試成功之後再放上小汽車才會是比較安全的作法，在此提醒組裝硬體線路時，請先關閉樹莓派的電源。

Step 3 硬體接線

請透過文書編輯器，把程式內容修改如下，要調整的部分是把另外一邊的馬達控制線路一起處理好。

範例 l293d_2_pwm.py

```
01 #!/usr/bin/env
02 #__author__ = "Powen Ko"
03 import RPi.GPIO as GPIO
04 from time import sleep
05
06 GPIO.setmode(GPIO.BOARD) # GPIO.BCM
07
08 MotorIN1 = 11 # 17
09 MotorIN2 = 12 # 18
10 MotorEN1 = 13 # 21
11
12 MotorIN3 = 15 # 22
13 MotorIN4 = 16 # 23
14 MotorEN2 = 18 # 24
15
16 GPIO.setup(MotorIN1,GPIO.OUT) # 設定 6 個輸出的接腳
17 GPIO.setup(MotorIN2,GPIO.OUT)
18 GPIO.setup(MotorEN1,GPIO.OUT)
19 GPIO.setup(MotorIN3,GPIO.OUT)
20 GPIO.setup(MotorIN4,GPIO.OUT)
21 GPIO.setup(MotorEN2,GPIO.OUT)
22
23 p1 = GPIO.PWM(MotorEN1,250) # PWM 的頻率=250Hz
24 p1.start(0) # 執行 PWM
25 p2 = GPIO.PWM(MotorEN2,250) # PWM 的頻率=250Hz
26 p2.start(0) # 執行 PWM
27
28 print "Turning motor on Clockwise."
29 GPIO.output(MotorIN1,GPIO.HIGH)
30 GPIO.output(MotorIN2,GPIO.LOW)
```

```

31 GPIO.output(MotorIN3,GPIO.HIGH)
32 GPIO.output(MotorIN4,GPIO.LOW)
33 for dc in range (0,101,5):      # 意思如 C 語言的 for(dc=0;dc<101;dc++)
34     p1.ChangeDutyCycle(dc)      # 調整 EN1 的 pwm 比例
35     p2.ChangeDutyCycle(dc)
36     sleep(0.2)
37
38 print "Turning motor on Counterclockwise."      # 逆時鐘的處理
39 GPIO.output(MotorIN1,GPIO.LOW)
40 GPIO.output(MotorIN2,GPIO.HIGH)
41 GPIO.output(MotorIN3,GPIO.LOW)
42 GPIO.output(MotorIN4,GPIO.HIGH)
43 for dc2 in range (0,101,5):
44     p1.ChangeDutyCycle(dc2)
45     p2.ChangeDutyCycle(dc2)
46     sleep(0.2)
47
48 print "Stopping motor"          # 關閉馬達的動作。
49 GPIO.output(MotorEN1,GPIO.LOW)
50 GPIO.output(MotorEN2,GPIO.LOW)
51 GPIO.cleanup()                 # 離開程式前，先恢復 GPIO 的接腳

```

程式解說

- 第 16-21 行：設定 6 個輸出的接腳，用來控制 IC 的動作。
- 第 23-26 行：設定 EN1 和 EN2 為 PWM 調整馬達轉速。
- 第 29-32 行：調整這 4 個接腳電位，讓 2 個馬達順時鐘旋轉。
- 第 33-36 行：以迴圈的方法改變 PWM 的佔空值，達到調整馬達轉速。
- 第 39-42 行：調整這 4 個接腳電位，讓 2 個馬達逆時鐘旋轉。
- 第 43-46 行：以迴圈的方法改變 PWM 的佔空值，達到調整馬達轉速。
- 第 49-51 行：離開時的處理動作。

執行結果

這個專案將使用 2 個直流馬達，請透過以下指令，執行 Python 的程式：

```
$ sudo python 1293d_2_pwm.py
```

執行的結果如下圖所示，二個馬達會一起朝同一個方向，每 0.2 秒由慢到快的旋轉後，再轉向另外一個方向，再做一次轉速從慢到最快的動作。

如果馬達沒有在轉，請留意是不是因為馬達吃太多電力，導致轉不動，如果發生這樣的情況，可以把外接的電池提升一些，最大到 12V2A，就會有幫助了。

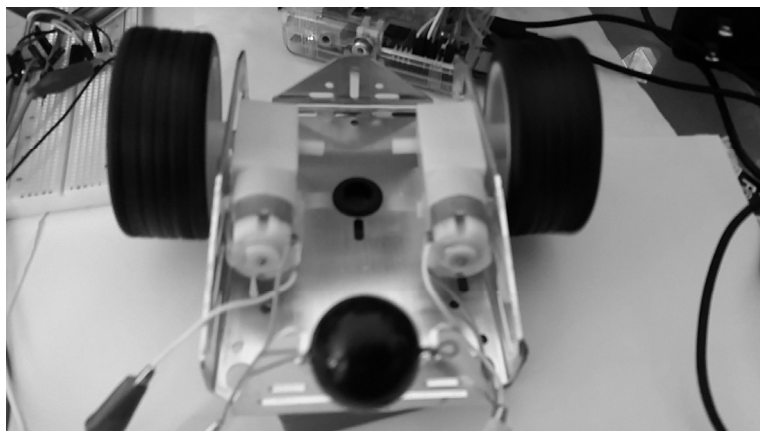


圖 19-17 執行結果

教學影片

完整的影片請參考光碟中的 *19-5_raspberryPi_L293D_3_2Motors*。

19.6 無線遙控小汽車

實驗介紹

本次實驗將會透過鍵盤來控制樹莓派的 GPIO，這樣就可以即時控制 IC L293D 來啟動兩顆馬達。本次範例的過程比較複雜，有幾個問題需要事先解決，無線遙控和把樹莓派放到小車子上，而且您一定也會發現，想要把樹莓派放在車子上面，還有幾條線需要拿掉，如：

- HDMI 的螢幕線
- 網路線
- 樹莓派的電源線

本章節將會為這些部分找到解決的方法。

實驗硬體設備

- 樹莓派的板子
- DC 直流馬達×2
- 麵包板
- 接線
- L293D 馬達驅動 IC
- 12V 到 5V 的電池×2
- 小汽車零組件
- Wi-Fi USB dongle
- USB 無線鍵盤
- 5V DC 和 2A 以上的智慧型手機的攜帶型充電器

實驗接線

同上一章節。

實驗步驟

請先完成上一章節實驗。

Step 1 程式撰寫

請透過文書編輯器，把程式內容修改如下，主要調整部分是透過鍵盤的按鍵，改變馬達的運轉方法，而這個專案中，特地替樹莓派小汽車設計了以下的動作：

- s 鍵：馬達暫停運轉。
- q 鍵：離開程式。
- 1 鍵：調整馬達轉速最快。
- 2 鍵：調整馬達轉速適中。
- 3 鍵：調整馬達轉速最慢。
- w 鍵：小汽車往前跑。

- x 鍵：小汽車往後跑。
- a 鍵：右轉。
- d 鍵：左轉。

範例 l293d_remote.py

```
01  #!/usr/bin/env
02  #__author__ = "Powen Ko"
03  import RPi.GPIO as GPIO
04  from time import sleep
05  import threading
06  import sys
07  import sys, tty, termios
08
09  def getch(): # 鍵盤處理
10      fd = sys.stdin.fileno()
11      old = termios.tcgetattr(fd)
12      try:
13          tty.setraw(fd)
14          return sys.stdin.read(1)
15      finally:
16          termios.tcsetattr(fd, termios.TCSADRAIN, old)
17
18  class KeyEventThread(threading.Thread): # 多執行緒
19      def run(self):
20          print("thread");
21          Fun()
22
23  def Fun():
24      print("Fun")
25      while True:
26          key=getch() # 按鍵處理
27          if key=='q':
28              funExit()
29              exit()
30              return
31          elif key=='1':
32              print('speed 1')
33              funSpeed(100,100)
34          elif key=='2':
35              print('speed 2')
36              funSpeed(70,70)
37          elif key=='2':
38              print('speed 3')
39              funSpeed(20,20)
40          elif key=='w':
41              print('forward') # 往前跑
```



```
42         GPIO.output(MotorIN1,GPIO.HIGH)
43         GPIO.output(MotorIN2,GPIO.LOW)
44         GPIO.output(MotorIN3,GPIO.HIGH)
45         GPIO.output(MotorIN4,GPIO.LOW)
46         funSpeed(50,50)
47     elif key=='x':                                     # 往後跑
48         print('backward')
49         GPIO.output(MotorIN1,GPIO.LOW)
50         GPIO.output(MotorIN2,GPIO.HIGH)
51         GPIO.output(MotorIN3,GPIO.LOW)
52         GPIO.output(MotorIN4,GPIO.HIGH)
53         funSpeed(50,50)
54     elif key=='a':                                     # 往左跑
55         print('left')
56         GPIO.output(MotorIN1,GPIO.HIGH)
57         GPIO.output(MotorIN2,GPIO.LOW)
58         GPIO.output(MotorIN3,GPIO.LOW)
59         GPIO.output(MotorIN4,GPIO.HIGH)
60         funSpeed(50,50)
61     elif key=='d':                                     # 往右跑
62         print('right')
63         GPIO.output(MotorIN1,GPIO.LOW)
64         GPIO.output(MotorIN2,GPIO.HIGH)
65         GPIO.output(MotorIN3,GPIO.HIGH)
66         GPIO.output(MotorIN4,GPIO.LOW)
67         funSpeed(50,50)
68     elif key=='s':
69         print('stop')
70         funSpeed(0,0)
71     else:
72         print("key="+key)
73     return
74
75
76 def funSpeed(i1,i2):
77     dc1=i1
78     dc2=i2
79     p1.ChangeDutyCycle(dc1)
80     p2.ChangeDutyCycle(dc2)
81
82
83 def funInit():
84     GPIO.setmode(GPIO.BOARD)                         # 設定 6 個輸出的接腳
85     GPIO.setup(MotorIN1,GPIO.OUT)
86     GPIO.setup(MotorIN2,GPIO.OUT)
87     GPIO.setup(MotorEN1,GPIO.OUT)
88     GPIO.setup(MotorIN3,GPIO.OUT)
89     GPIO.setup(MotorIN4,GPIO.OUT)
90     GPIO.setup(MotorEN2,GPIO.OUT)
```

```

91
92 def funExit():                                # 關閉馬達的動作
93     print "Stopping motor"
94     GPIO.output(MotorEN1,GPIO.LOW)
95     GPIO.output(MotorEN2,GPIO.LOW)
96     GPIO.cleanup()
97
98
99 MotorIN1 = 11 # 17
100 MotorIN2 = 12 # 18
101 MotorEN1 = 13 # 21
102 MotorIN3 = 15 # 22
103 MotorIN4 = 16 # 23
104 MotorEN2 = 18 # 24
105
106 print("Press 'q' to exit")
107 print("'w'=forward,'x'=backward,'a'=left,'d'=right,'s'=stop")
108 print("'1','2','3' motor speed")
109 funInit()
110 p1 = GPIO.PWM(MotorEN1,250)    # PWM 的頻率=250Hz
111 p1.start(0)
112 p2 = GPIO.PWM(MotorEN2,250)
113 p2.start(0)
114
115 kethread = KeyEventThread()
116 kethread.start()

```

程式解說

- 第 9-16 行：讀取鍵盤的資料。
- 第 40-46 行：調整這 4 個接腳電位，讓 2 個馬達順時鐘旋轉，這樣小汽車就會往前跑。
- 第 47-53 行：調整這 4 個接腳電位，讓 2 個馬達逆時鐘旋轉，這樣小汽車就會往後跑。
- 第 54-60 行：調整這 4 個接腳電位，讓 2 個馬達一正一反，這樣小汽車就會往左跑。
- 第 61-67 行：調整這 4 個接腳電位，讓 2 個馬達一反一正，這樣小汽車就會往右跑。
- 第 83-90 行：設定 6 個輸出的接腳，用來控制 IC 的動作。
- 第 110-113 行：設定 EN1 和 EN2 為 PWM 調整馬達轉速。

Step 2 測試

請先執行 Python 的程式，測試完成後，確認可以透過鍵盤的按鍵控制馬達，再把設備搬到小汽車上面。

```
$ sudo python l293d_remote.py
```

Step 3 去掉電源線

第一個是樹莓派供電問題，筆者會建議使用市面上販售的智慧型手機的攜帶型充電器，基本上它的電壓和電池的續航力都已經足夠讓樹莓派使用，不用再另外購買。請使用輸出有 5V DC 和 2A 以上的移動式手機電池，有些便宜的智慧型手機的攜帶型充電器供電就不到 2A，請留意，以免無法啟動樹莓派。



圖 19-18 智慧型手機的攜帶型充電器

Step 4 去掉網路線

樹莓派上面或許會有網路線，所以可以參考本書「設定 USB Wifi 連線」的介紹，使用 Wi-Fi USB Dongle 的方法來做網路連線。

Step 5 控制

因為要用鍵盤控制樹莓派小汽車，但又希望小汽車可以自由的行動，不會因為鍵盤的線卡在那邊，有 2 個方法可以解決：

- 使用網路和 SSH 的方法來控制樹莓派，這樣就不需要在樹莓派上連接鍵盤了。
- 也可以用無線鍵盤能達到這樣的功能。

為了方便，筆者建議控制的部分，使用 USB 無線鍵盤，您可以挑選電腦用的 USB 無線鍵盤就可以正常使用，不用再另外花錢購買特殊的鍵盤。



圖 19-19 USB 無線鍵盤

Step 6 螢幕

不需要用 HDMI 接到螢幕上，可以透過 SSH 網路連線的方法來控制樹莓派，所以可以用電腦或是手機的 SSH。

執行結果

請先把樹莓派關閉後：

- 把樹莓派放上小汽車。
- 把相關電路也放上小汽車，並用絕緣膠帶綁緊。
- 把 2 個電池（和樹莓派用的 USB 充電器）放上小汽車。
- 把電池分別接上樹莓派和 L293D。
- 等待樹莓派開機。
- 可以用電腦或是手機的 SSH 連線到樹莓派上。
- 執行程式。

這個專案中將使用 SSH 和鍵盤，來控制 2 個直流馬達的前後左右的動作，請依照以下的步驟設定：

- 請在手機或電腦上，透過 SSH Client 端軟體連線到樹莓派小汽車上。
- 執行在手機上的 SSH Client 端，用以下指令，執行樹莓派小汽車上 Python 的程式。

```
$ sudo python l293d_remote.py
```

請分別用鍵盤的 w、s、x、a、d 鍵來控制小汽車的前後左右和停止，1、2、3 鍵來控制速度。如果要離開程式，請透過 q 鍵。此時就能順利的控制樹莓派小汽車了。

注意

1. USB 無線鍵盤請接在 SSH 的電腦端，而不是接在樹莓派。
2. 根據實際測試，部分牌子的移動式手機電池，無法順利啟動樹莓派，請使用輸出有 5V DC 和 2A 以上的移動式手機電池。
3. 請多用絕緣膠帶把線路和設備固定好。



圖 19-20 執行結果

執行影片

成品展示的完整影片請參考光碟中的 `19-6_raspberryPi_L293D_4_remote_final.mp4`，就能看到樹莓派小汽車的控制情況。

教學影片

完整的影片請參考光碟中的 *19-6_raspberryPi_L293D_4_remote_T.mp4*。

19.7 iOS 和 Android 控制小汽車

實驗介紹

現在智慧型手機幾乎已是人手一機，所以本次的實驗將會實際透過使用智慧型手機，來控制上一個章節所自製的樹莓派小汽車。

實驗硬體設備

- 樹莓派的板子
- DC 直流馬達×2
- 麵包板
- 接線
- L293D 馬達驅動 IC
- 12V 到 5V 的電池×2
- 小汽車零組件
- Wi-Fi USB dongle
- USB 無線鍵盤
- 5V DC 和 2A 以上的智慧型手機的攜帶型充電器
- Android 或者是 iOS 的智慧型手機或平板

實驗接線

同上一章節。

實驗步驟

請先完成上一章節實驗，因為不同的作業系統，它所需要的設定不太一樣，所以請依照實際情況，挑選以下的章節做設定，特別提醒，iOS 和 Android 都必須在同一

個網域下，才可以順利動作，也就是說使用的智慧型手機或者是平板，必須要連結到樹莓派所使用的相同 Wi-Fi Router 路由器上，這樣彼此才能順利溝通，並且把智慧型手機的 2G、3G、4G 暫時關閉。

19.7.1 使用 Android 控制樹莓派小汽車

實驗步驟

請先完成上一章節的設定，把樹莓派小汽車啟動並且連接到網路上。

Step 1 Wi-Fi 網路連線

首先請先把 Android 手機，透過「設定\網路\Wi-Fi」的方法，連結到和樹莓派相同的 Wi-Fi 無線網路上。



圖 19-21 連結到和樹莓派相同的 Wi-Fi 無線網路上

Step 2 安裝 Android 軟體—JuiceSSH

JuiceSSH - SSH Client 這一款免費的 Android SSH APP，使用起來相當的便利好用。可以在 Android 作業系統的平板或者是智慧型手機使用，在 Android Store 上輸入 Juice SSH 就可以找到。

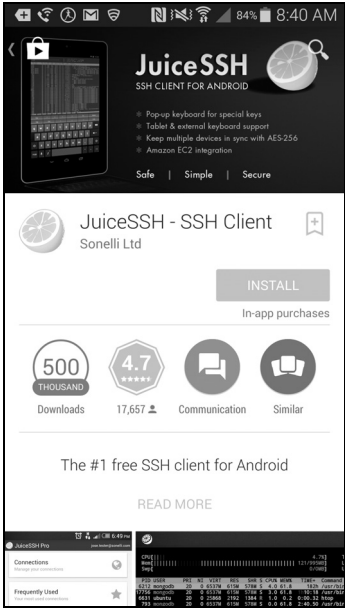


圖 19-22 透過 Google store 下載 JuiceSSH

或者也可以在網頁上，透過瀏覽器下載：

<https://play.google.com/store/apps/details?id=com.sonelli.juicessh&hl=en>

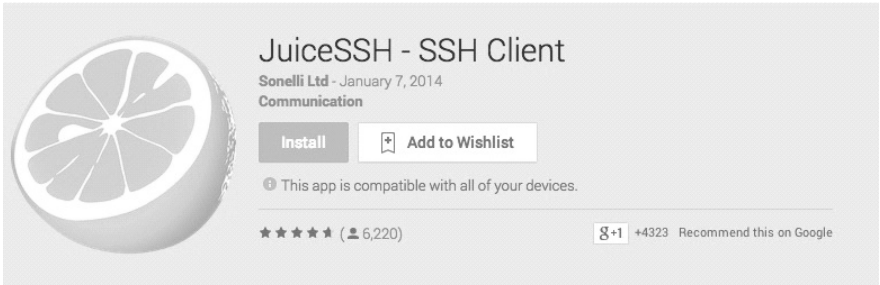


圖 19-23 好用的 Android 軟體 JuiceSSH - SSH Client

Step 3 執行 Android 軟體—JuiceSSH

接下來請在 Android 上執行 Juice SSH，在第一次的開啟時，需要作一些設定，並且新增新的連線，請點選「New Connection」鈕。

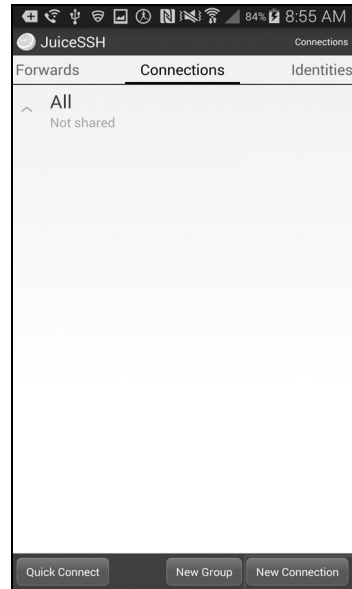


圖 19-24 在 Android 上執行 Juice SSH，並點選「New Connection」鈕

Step 4 設定網路的相關資料

接下來請設定網路的相關資料：

- 暱稱
- 連線方式
- 樹莓派的網路位址
- 帳號和密碼的設定，點選後，就會進入帳號密碼的設定頁面

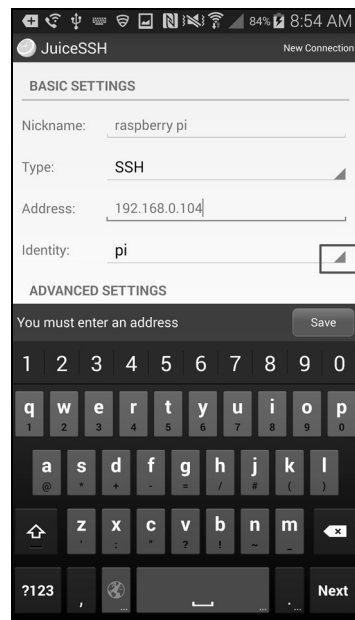


圖 19-25 設定網路的相關資料

Step 5 帳號密碼的設定頁面

在這個 Identity 的帳號密碼設定頁面，請把樹莓派的 SSH 登錄時的帳號和密碼登記在此：

- Nickname：暱稱。
- Username：SSH 登入樹莓派時的帳號，如果之前沒有修改樹莓派的帳號和密碼，這裡使用者名稱是 pi。
- Password：SSH 登入樹莓派時的密碼，如果之前沒有修改樹莓派的帳號和密碼，這裡使用者密碼是 raspberry。
- Private Key 私有關鍵鑰匙：一般來說不用修改。

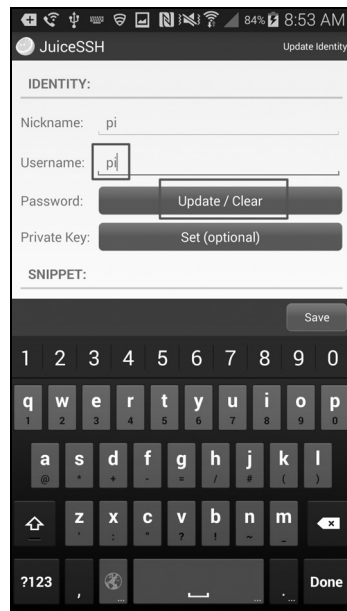


圖 19-26 帳號密碼設定頁面

Step 6 連線

請打開樹莓派小汽車的電源準備連線，並且確認樹莓派遙控小汽車已經順利連上 Wi-Fi 無線網路之後，請回到 Android 的 JuiceSSH，點選剛剛的設定選項。如果一切設定沒有錯誤，就可順利連結，然後到 SSH 文字模式中，如果發現無法連線，請確認您的設定檔的網路位置是否正確。

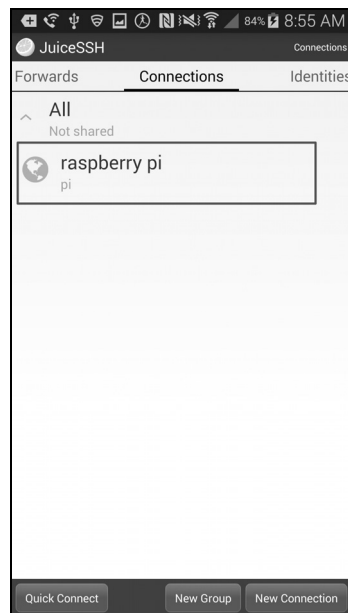


圖 19-27 網路設定的相關資料

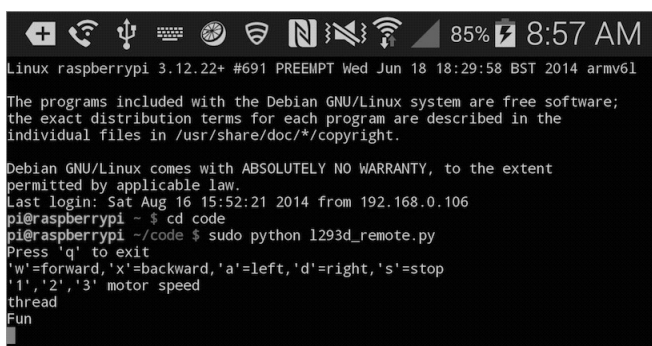
Step 7 連線到 SSH

透過 Android 手機連線到樹莓派之後，請執行同一個範例的 Python 語言，指令如下：

```
$ sudo python l293d_remote.py
```

如此就可以順利執行，操作方法也是一樣，請透過 w、x、a、d、s 按鍵來控制遙控小汽車，如果要離開，請透過 q 鍵。

執行畫面如下：



```
Linux raspberrypi 3.12.22+ #691 PREEMPT Wed Jun 18 18:29:58 BST 2014 armv6l
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 16 15:52:21 2014 from 192.168.0.106
pi@raspberrypi ~ $ cd code
pi@raspberrypi ~/code $ sudo python l293d_remote.py
Press 'q' to exit
'w'=forward, 'x'=backward, 'a'=left, 'd'=right, 's'=stop
'1', '2', '3' motor speed
thread
Fun
```

圖 19-28 透過 juices SSH 執行 l293d_remote.py

執行結果

詳細的執行結果可以參考光碟中的 19-7-1_raspberryPi_L293D_4_remote_Android.mp4，就會看到透過 Android 智慧型手機 JuiceSSH 遠端連線到 Raspberry Pi，並且執行 Python 程式，去控制樹莓派遙控小汽車。



圖 19-29 實際執行的情況

19.7.2 使用 iOS 控制樹莓派小汽車

實驗介紹

上個章節介紹的是使用 Android 系統的智慧型手機控制樹莓派遙控小汽車，本章節將介紹使用蘋果的 iOS 作業系統來控制樹莓派小汽車。其實這二種作業系統控制樹莓派的原理是一樣的，都是透過 SSH 軟體達到目的。當然有人會好奇，是不是還有其他的技術可以讓樹莓派和智慧手機相互溝通？如果各位熟悉 iOS 和 Android 系統的程式開發，就會有很多的方法，例如說透過 TCP/IP、網頁技術、藍牙等的技術都可做到。

實驗步驟

請先完成上一章節的設定，把樹莓派小汽車啟動並且連接到網路上。

Step 1 Wi-Fi 網路連線

確認 iOS 手機，已經連結到和樹莓派相同的 Wi-Fi 無線網路上，透過「設定\Wi-Fi」的方法指定到相同的無線網路。

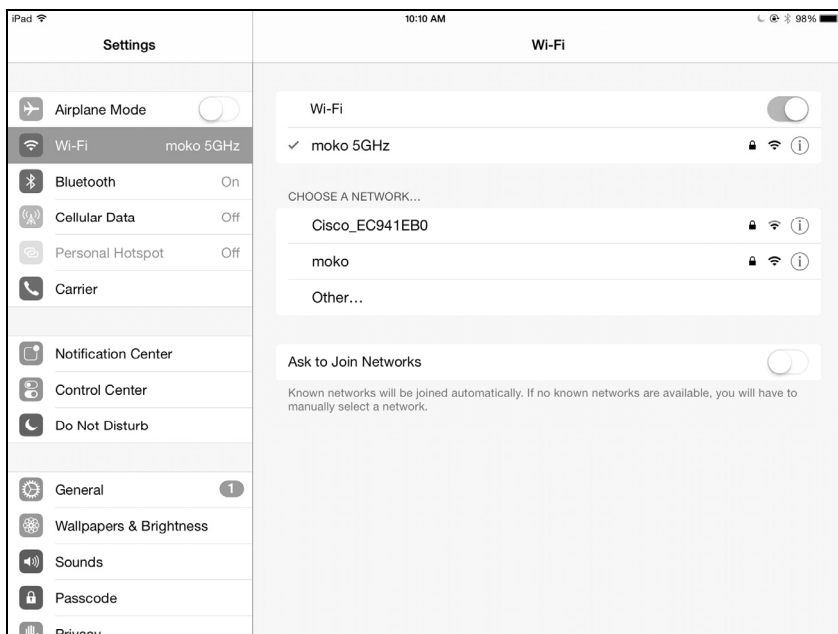


圖 19-30 連結到和樹莓派相同的 Wi-Fi 無線網路上

Step 2 安裝 Android 軟體—JuiceSSH



圖 19-31 透過 APP Store 下載 WebSSH

如果想使用 iOS 智慧型手機，從遠端 SSH 存取控制，可以使用 WebSSH 這一款免費的 APP，使用起來相當的便利和好用。可以躺在沙發上，透過手機就可以連線到 Raspberry Pi 做事情，聽起來真的是不可思議的事情，但現在透過手機版的 SSH 軟體，就可以做到。

Step 3 把樹莓派遙控小汽車電源打開

和剛剛的 Android 的步驟一樣，請打開樹莓派小汽車的電源，並且準備連線和確認樹莓派遙控小汽車已經順利連上同一個區域網路的 Wi-Fi 無線網路，並且請確認您的設定檔的網路位置是否正確。



圖 19-32 把樹莓派遙控小汽車電源打開

Step 4 啟動 WebSSH

安裝結束後，接下來請在 iOS 上執行 WebSSH，在第一次開啟時，需要作一些設定，並且新增新的連線，請點選「+」鈕，進入連線設定。

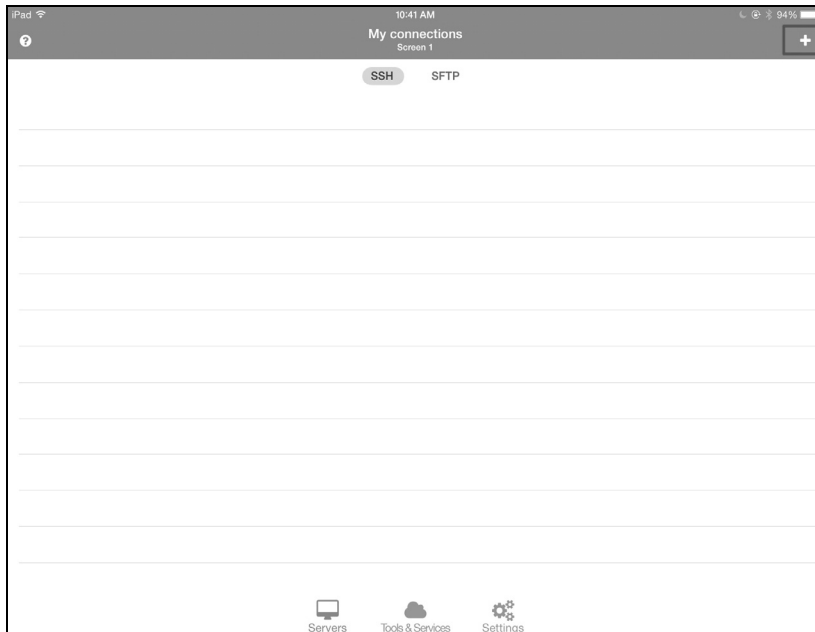


圖 19-33 在 iOS 上執行 Juice SSH，並點選「+」鈕

Step 5 設定網路的相關資料

接下來請設定網路的相關資料：

- Host 樹莓派的網路位址：請依照實際的樹莓派網路位址填寫。
- Port SSH 的連接 Port：不要修改，維持內定的 22。
- User 帳號的設定：SSH 登入樹莓派時的帳號，如果之前沒有修改樹莓派的帳號和密碼，這裡使用者名稱是 pi。
- Password 密碼的設定：SSH 登入樹莓派時的密碼，如果之前沒有修改樹莓派的帳號和密碼，這裡使用者密碼是 raspberry。

完成後，點選「Connection」的連線鈕，就可以順利地透過 SSH 連結到樹莓派上。

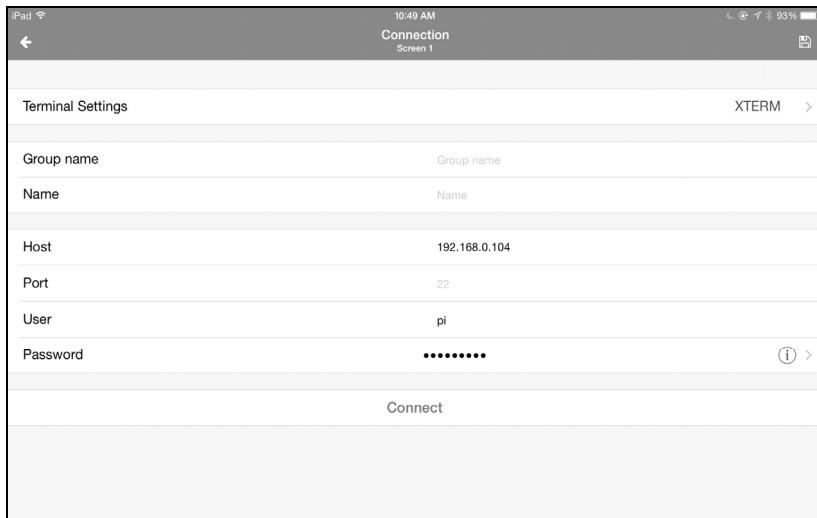


圖 19-34 設定網路的相關資料

Step 6 連線到 SSH

當順利透過 iOS 手機連線到樹莓派之後，也是執行同一個範例的 Python 語言，指令如下：

```
$ sudo python l293d_remote.py
```

在樹莓派遙控小汽車的方法也是一樣，請透過 w、x、a、d、s 按鍵來控制遙控小汽車，如果要離開，請透過 q 鍵。WebSSH 執行的畫面如下：



圖 19-35 透過 WebSSH 執行 l293d_remote.py

執行結果

完整的執行結果影片請參考光碟中的 *19-7-2_raspberryPi_L293D_5_remote_iOS.mp4*，會看到 iPad 透過 WebSSH 遠端連線到 Raspberry Pi，並且執行 Python 程式，去控制樹莓派遙控小汽車。



圖 19-36 實際執行的情況