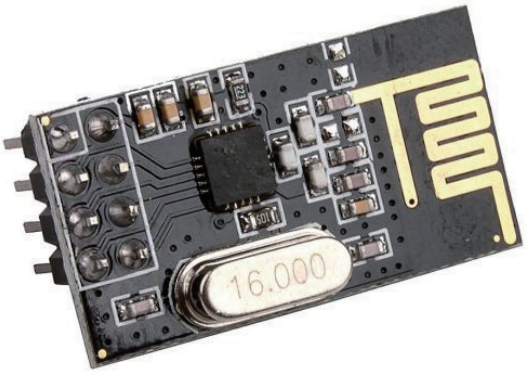
2-9 NRF24L01+



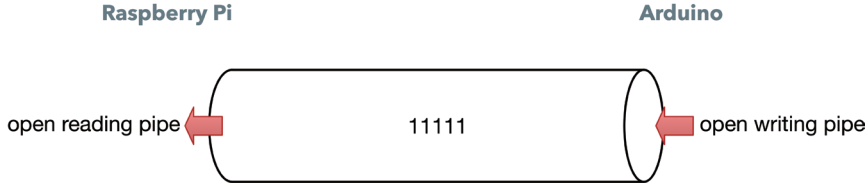
NRF24L01+ 模組是一個在 2.4G~2.5GHz 的單晶片無線通訊模組，內建天線，並且有自己的資料傳輸協定，這個協定與 Wi-Fi、藍牙、ZigBee 均不相同。這顆晶片以半雙工的方式進行資料交換，也就是當他處於監聽資料狀態時是無法送資料出去的，如果要送資料出去就必須先停止監聽，資料送出後再恢復監聽。這顆晶片不需要 NCC 許可證就可以使用了。

NRF24L01+ 發送與讀取資料的資料通道稱為 pipe（管道），讀取資料的管道稱為 reading pipe，發送資料的管道稱為 writing pipe，不論是讀還是寫，pipe 都是單向傳遞資料。

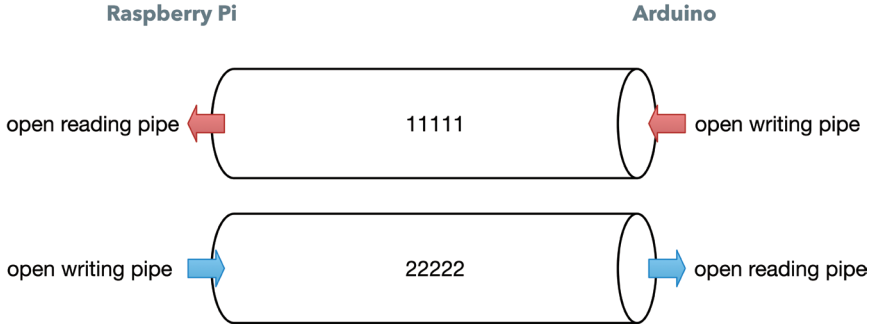
資料傳遞前，必須要先為 pipe 命名，命名方式為指定 5 個 byte 的 ASCII 碼，可以使用字元陣列來表示，例如 'abcde' 或是直接指定 ASCII 碼，例如 {0x01, 0x02, 0x03, 0x04, 0x05}。NRF24L01+ 的驅動程式中有函數可以顯示初始化的狀態，其中包含 reading pipe 與 writing pipe 的名字，這個名字以 16 進位的 ASCII 碼呈現，但需要注意的是，呈現的名字跟當初在程式中設定的名字高低位元是剛好相反的。例如程式中命名為 {0x01, 0x02, 0x03, 0x04, 0x05}，顯示時會顯示 {0x05, 0x04, 0x03, 0x02, 0x01}。

這單元會以樹莓派與 Arduino 為範例，透過 NRF24L01+ 讓兩個裝置間可以互相傳遞資料，若讀者對於 Arduino 不熟悉，使用兩個樹莓派也可以，只是樹莓派與樹莓派之間的資料交換方式就很多樣了，不一定要使用 NRF24L01+ 這顆晶片。

以 Arduino 上的感測器要將感測到的資料送到樹莓派為例，在 Arduino 上必須開啟 writing pipe，若 pipe 名稱為 11111，則在樹莓派上就必須開啟名稱為 11111 的 reading pipe，這樣 Arduino 才可以將資料寫到樹莓派上，如下圖。



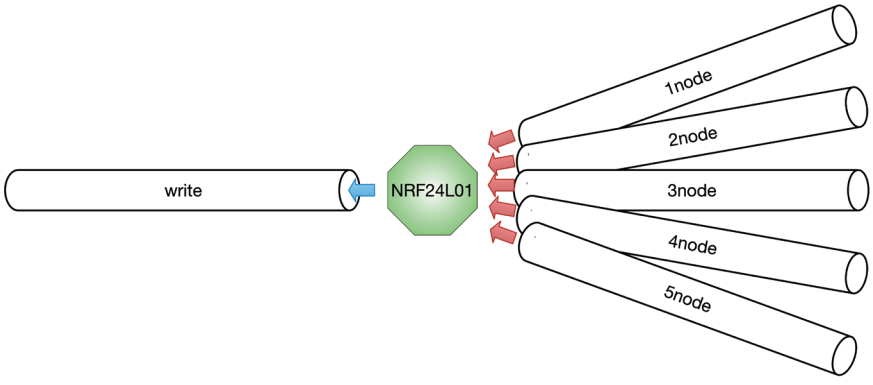
如果樹莓派收到資料後要回傳訊息給 Arduino，或是樹莓派要下指令給 Arduino，代表資料會雙向傳遞，此時 Arduino 上除了 11111 這個 writing pipe 之外，還必須再開啟另外一個 reading pipe，當然樹莓派要開啟同一個名字（例如 22222）的 writing pipe 好讓資料可以寫到 Arduino 端，兩個裝置間要開啟兩條 pipe 才能互相傳遞資料，如下圖。



一個 NRF24L01+ 最多可以有 5 個 reading pipe 與 1 個 writing pipe，換句話說，我們可以讓一個樹莓派接收 5 個 Arduino 傳過來的資料。

如果我們的裝置只要收資料而不需要傳資料出去，這時透過參數設定可以把 writing pipe 變成 reading pipe，達到每個 NRF24L01+ 最多有 6 個 reading pipe 的數量。

要讓樹莓派上的 NRF24L01+ 同時開啟 2 個以上的 reading pipe 來接收兩個來源以上的資料輸入（例如 Arduino1、Arduino2），pipe 的命名方式是有規定的。除了第一個 byte 可以不一樣之外，後 4 個 byte 必須相同，例如將 reading pipe 分別命名為 '1node'、'2node'、'3node'、'4node'、'5node'。而此時 writing pipe 的命名方式，全部 5 個 byte 跟 reading pipe 的 5 個 byte 可以完全不同，例如名稱為 'write'。

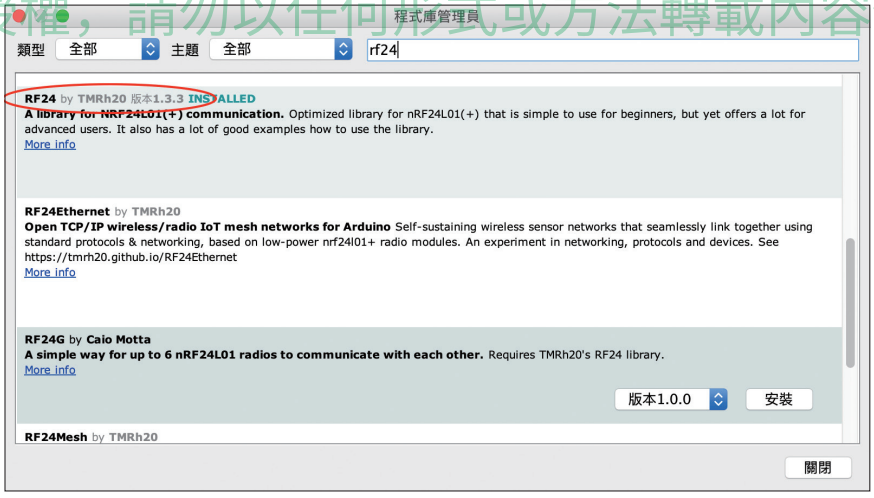


2-9-1 安裝驅動程式

不論是樹莓派還是 Arduino，都需要先安裝驅動程式，筆者挑選的驅動程式在兩個平台為同一套，並且在樹莓派上還支援 Python 語言，因此使用起來非常方便。

Arduino 端

在管理程式庫畫面搜尋「rf24」，搜尋結果會出現多筆資料，安裝名稱為 RF24 且作者為 TMRh20。



樹莓派端

在上一個畫面點選「More info」到該函數庫的官方網站（網址為 https://tmrh20.github.io/RF24/），找到 Download 連結，複製網址然後在樹莓派上用 wget 指令下載回來。

```
$ wget https://github.com/TMRh20/RF24/archive/master.zip
```

解壓縮後依序執行以下指令編譯原始碼與安裝。

```
$ sudo apt-get install libboost-python-dev
$ cd RF24-master
$ ./configure --driver=RPi
$ make
$ sudo make install
$ cd pyRF24
$ sudo python setup.py install
$ sudo python3 setup.py install
```

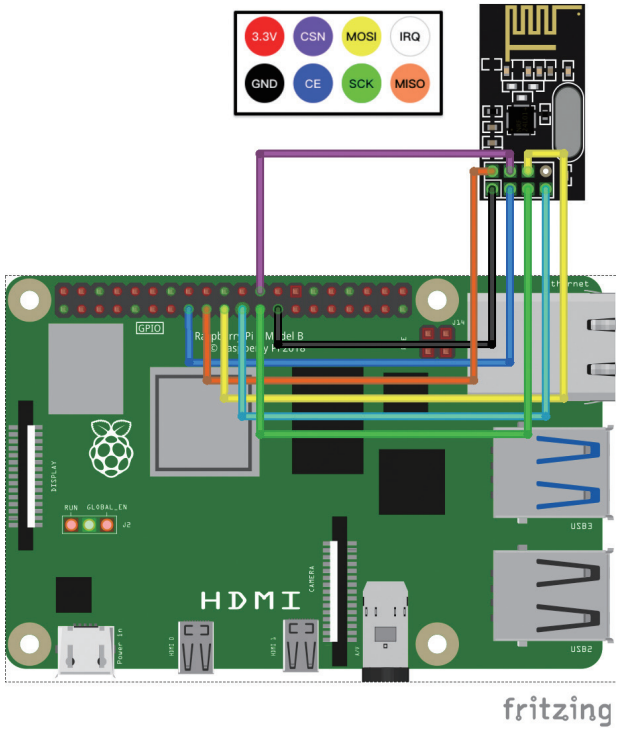
NRF24L01+ 為 SPI 協定的晶片，因此樹莓派記得要開啟 SPI 功能。接下來就可以將 NRF24L01+ 接到樹莓派與 Arduino 上，開始寫程式驅動他了。

2-9-2 硬體接線

樹莓派端的步驟與說明

1 樹莓派硬體線路接法如下圖，這顆晶片的 VCC 必須接 3.3V。

NRF24L01+	RPi
VCC	3.3V
GND	GND
CE	GPIO22（可任意選擇）
CSN	GPIO8（固定）
MOSI	GPIO10（固定）
MISO	GPIO9（固定）
SCK	GPIO11（固定）



2 硬體接好後，寫一小段程式測試一下線路連接以及相關函數庫是否安裝正確。RF24() 函數的第一個參數為 NRF24L01+ 的 CE 腳在樹莓派上的 GPIO 編號，以上圖而言為 22；第二個參數 0 代表 NRF24L01+ 的 CSN 腳接到樹莓派上 SPI 的 CE0 位置，若接到樹莓派的 CE1，第二個參數就要填 1。程式中使用 openReadingPipe() 開啟 reading pipe，第一個參數為 pipe 編號，範圍為 1~5 整數。Writing pipe 編號固定在 0 號，因此 reading pipe 編號從 1 開始。如果 openReadingPipe() 的第一個參數填 0，代表要將 writing pipe 轉成 reading pipe 使用。

```
import RF24

radio = RF24.RF24(22, 0)
radio.begin()

radio.openReadingPipe(1, b'12345')
radio.openWritingPipe(b'abcde')
radio.printDetails()
```

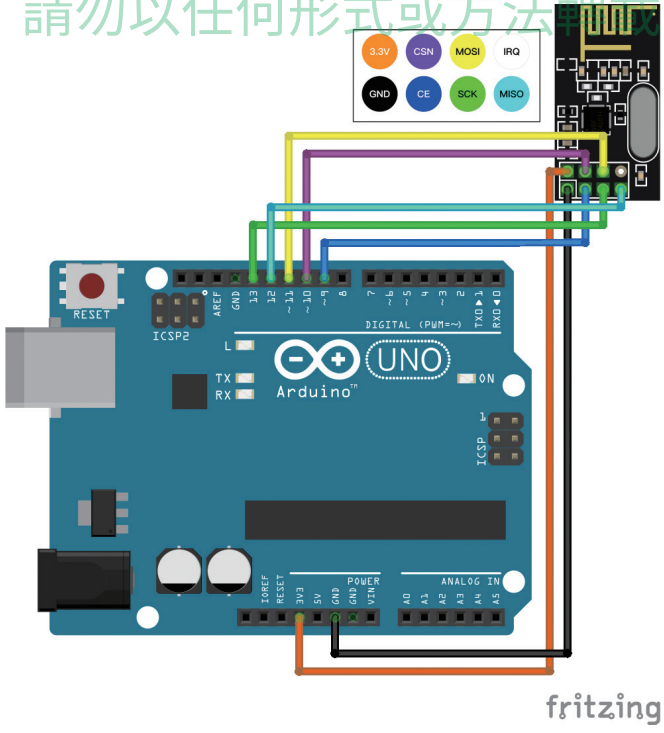
3 上述程式碼執行時必須加上 sudo，否則無法正確執行。執行後會看到以下畫面，檢查「RX_ADDR_P0-1」這個欄位。P0 代表 writing pipe 的值，我們在程式中輸入的是 abcde，換成 ASCII 碼並且高低位元對調後就是 0x6564636261，後面緊跟著的 0x3534333231，其實就是 reading pipe 也就是 P1 輸入的名字 12345。只要這個欄位的值不是 0x0 0x0，就幾乎可以確定 NRF24L01+ 會正常運作了。

```
ckk — pi@seal: ~ — ssh pi@seal.local — 70×22
pi@seal:~ $ sudo python3 nrf.py
================ SPI Configuration ================
CSN Pin           = CE0 (PI Hardware Driven)
CE Pin            = Custom GPIO22
Clock Speed       = 8 Mhz
================ NRF Configuration ================
STATUS            = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1      = 0x6564636261 0x3534333231
RX_ADDR_P2-5      = 0xc3 0xc4 0xc5 0xc6
TX_ADDR           = 0x6564636261
RX_PW_P0-6        = 0x20 0x20 0x00 0x00 0x00 0x00
EN_AA             = 0x3f
EN_RXADDR         = 0x03
RF_CH             = 0x4c
RF_SETUP          = 0x07
CONFIG            = 0x0e
DYNPD/FEATURE     = 0x00 0x00
Data Rate         = 1MBPS
Model             = nRF24L01+
CRC Length        = 16 bits
PA Power          = PA_MAX
pi@seal:~ $
```

Arduino 端的步驟與說明

1. 接下來是 Arduino 的硬體接線。

NRF24L01+	Arduino
VCC	3.3V
GND	GND
CE	9（可任意選擇）
CSN	10（可任意選擇）
MOSI	11（固定）
MISO	12（固定）
SCK	13（固定）



2 Arduino 的程式如下。開啟 Arduino IDE，匯入三個固定的標頭檔。根據硬體接線，宣告 radio 變數時指定 CE 為 pin 9，CSN 為 pin 10。最後為 writing pipe 命名為 '11111'，我們就是要透過這個 pipe 將資料寫到樹莓派去。

```
#include <SPI.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(9, 10);  // CE, CSN
byte w_pipe[] = "11111";
```

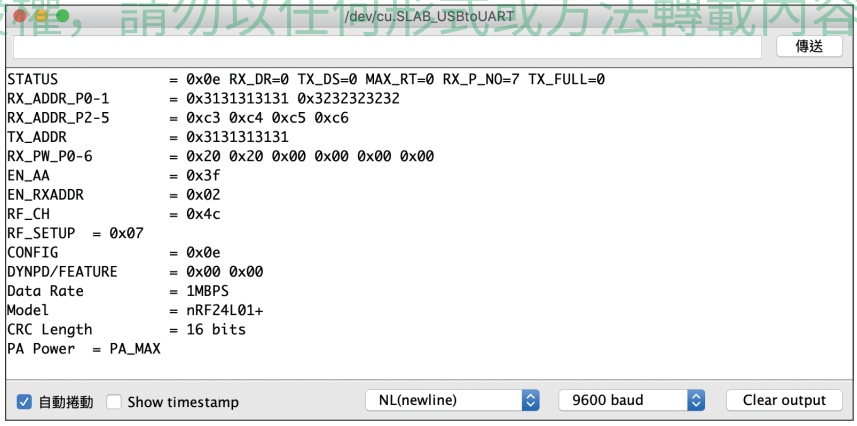
3 在 setup() 函數中，printf_begin() 的目的是為了讓最後一行 radio.printDetails() 的輸出結果能輸出到「序列埠監控視窗」，這個輸出結果可以讓我們知道 NRF24L01 是否正常初始化。setRetries() 的第一個參數為傳送失敗時每次重送的間隔時間，最大可填 15，每單位時間為 250 微秒，所以 15 代表重送的間隔時間為 4000 微秒，第二個參數決定重送幾次都不成功後就判定失敗，最大值也是 15。然後呼叫 openWritingPipe() 來開啟名稱為 '11111' 的寫入通道。

```
void setup() {
    Serial.begin(9600);
    printf_begin();

    radio.begin();
    radio.setRetries(15, 15);
    radio.openWritingPipe(w_pipe);
    radio.printDetails();
}
```

4 在 loop() 函數中，我們很簡單的每一秒鐘將變數 n + 1，然後透過 sprintf() 轉成字串後交由 RF24 函數庫的 write() 函數發送出去。

```
int n = 0;
char data[32] = {};

void loop() {
  sprintf(data, "%d", n++);
  radio.write(data, strlen(data));
  delay(1000);
}
```

5 Arduino 程式執行後，若在序列埠監控視窗中看到如樹莓派第 3 步的資料，Arduino 端的軟硬體設定也就沒什麼問題。下一節我們只要把樹莓派的程式寫完，就可以不斷的收到 Arduino 傳過來的資料了。



2-9-3 樹莓派端的接收資料程式

步驟與說明

1. 匯入 time 與 RF24 函數庫。然後根據硬體線路接線，NRF24L01+ 的 CE 接在 GPIO22，CSN 接在 GPIO8（意即 CE0），所以 RF24() 初始化函數第一個參數填 22，第二個參數填 0，最後指定 reading pipe 的名稱為 '11111'。

```
import time, RF24

radio = RF24.RF24(22, 0) #(CE, CSN)
r_pipe = b'11111'
```

2. setRetries() 的第一個參數為傳送失敗時每次重送的間隔時間，最大可填 15，每單位時間為 250 微秒，所以 15 代表重送的間隔時間為 4000 微秒，第二個參數決定重送幾次都不成功後就判定失敗，最大值也是 15。然後使用 openReadingPipe() 開啟名稱為 '11111' 的讀取通道，最後透過 printDetails() 函數來檢查 NRF24L01+ 是否都設定正確。

```
radio.begin()
radio.setRetries(15, 15)
radio.openReadingPipe(1, r_pipe)

radio.printDetails()
print("NRF24L01+ is ready.")
```

3 使用一個無窮迴圈來讀取 '11111' 通道上傳來的資料。進入迴圈前必須先透過 startListening() 函數告訴 NRF24L01+ 要去監聽通道，迴圈裡面透過 available() 函數得知是否有資料進來。讀進來的資料型態是 bytearray，所以要用 utf-8 轉成字串形式，最後把收到的資料印出來。

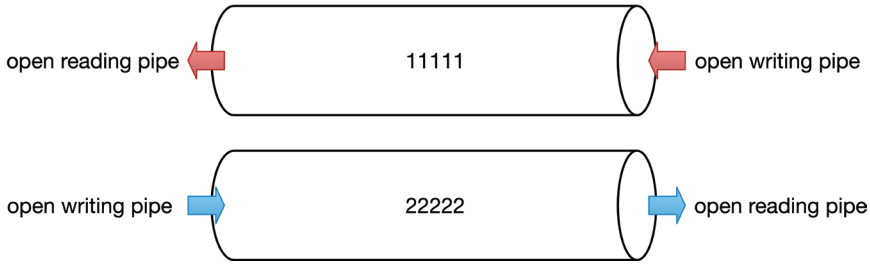
```
radio.startListening()
while True:
    if radio.available():
        len = radio.getDynamicPayloadSize()
        data = radio.read(len)
        print('[{}]'.format(data.decode('utf-8')))

    time.sleep(0.1)
```

4 用 root 身份執行這個程式，如一切順利，應該每一秒鐘可以收到 Arduino 傳過來的數字一次。

2-9-4 雙向資料傳遞

我們打算設計一個當 Arduino 送一個字串到樹莓派上時，樹莓派收到後回傳這個字串給 Arduino，相當於一個 echo 系統。這系統的 pipe 的架構為：在樹莓派上指定 '11111' 這個名稱的 reading pipe，Arduino 上的 writing pipe 要就要指定 '11111' 這個名稱。此外，在 Arduino 上指定 '22222' 為 reading pipe，樹莓派上就要設定 '22222' 為 writing pipe，如下圖：

Raspberry Pi Arduino



樹莓派端的步驟與說明

1. 與上一節不同處已用註解標示出來，主要是增加了 '22222' 為 writing pipe。

```
import time, RF24

radio = RF24.RF24(22, 0) #(CE, CSN)
r_pipe = b'11111'
w_pipe = b'22222'    # 增加

radio.begin()
radio.setRetries(15, 15)
radio.openReadingPipe(1, r_pipe)
radio.openWritingPipe(w_pipe) # 增加

radio.printDetails()
print("NRF24L01+ is ready.")
```

2. 無窮迴圈部分程式如下，要注意的是要將資料寫入 pipe 時，必須先停止 reading pipe 的監聽，寫入完成後再恢復監聽狀態。

```
radio.startListening()
while True:
    if radio.available():
        len = radio.getDynamicPayloadSize()
        data = radio.read(len)
        print('[{}]'.format(data.decode('utf-8')))
```

```
        radio.stopListening()           # 增加
        radio.write(b"echo: " + data) # 增加
        radio.startListening()          # 增加

    time.sleep(0.1)
```

3 以 root 身份執行看看。

Arduino 端的步驟與說明

1 主要增加 reading pipe 的名稱宣告。

```
#include <SPI.h>
#include <RF24.h>
#include <printf.h>

RF24 radio(9, 10);        // CE, CSN
byte w_pipe[] = "11111";
byte r_pipe[] = "22222";  // 增加
```

2 在 setup() 函數中，主要是開啟名稱為 '22222' 的 reading pipe，另外在最後透過 startListening() 函數讓 NRF24L01+ 進入監聽狀態，準備讀取樹莓派送過來的資料。

```
void setup() {
    Serial.begin(9600);
    printf_begin();

    radio.begin();
    radio.setRetries(15, 15);
    radio.openReadingPipe(1, r_pipe); // 增加
    radio.openWritingPipe(w_pipe);
    radio.printDetails();
    radio.startListening();           // 增加
}
```

3 在 loop() 函數中分成兩大部分，第一部分是從序列埠讀取資料，因此我們可以開啟「序列埠監控視窗」，然後從這個視窗中輸入想要送到樹莓派的字串。資料用 write() 函數送到寫入通道前必須先呼叫 stopListening() 函數來停止監聽狀態，資料寫完後要記得恢復監聽狀態。第二部分就是從讀取通道中取得從樹莓派送過來的資料，然後在序列埠中顯示。

```
char data[32];
void loop() {
  if (Serial.available()) {
    // 從序列埠讀資料
    delay(10);
    int n = Serial.available();
    Serial.readBytes(data, n);
    data[n] = '\0';

    // 將資料送到樹莓派
    radio.stopListening();
    radio.write(data, strlen(data));
    radio.startListening();
    printf("send '%s', len=%d\n", data, strlen(data));
  }

  if (radio.available()) {
    // 取得從樹莓派送回來的資料
    radio.read(data, 32);
    printf("\t=> %s\n", data);
  }
}
```

4 執行看看。從序列埠監控視窗輸入一個字串後應該可以看到從樹莓派返回的字串。

2-9-5 兩個 Arduino 連接一個樹莓派

這一節我們要讓兩個（以上）的 Arduino 同時向樹莓派發送資料，這樣的架構在建構一個智慧家庭時非常有用。

Arduino 端的步驟與說明

1. Arduino 端的程式碼幾乎與上一節一模一樣，只有不同的 Arduino 要設定不同名字的 reading pipe 而已。命名時要注意，只有第一個 byte 可以不一樣，之後四個 byte 必須同名稱。

   第一個 Arduino：

```
byte r_pipe[] = "1ardu";
byte w_pipe[] = "1node";
```

   第二個 Arduino：

```
byte r_pipe[] = "2ardu";
byte w_pipe[] = "2node";
```

樹莓派端的步驟與說明

1. 樹莓派要處理的事情比較多，首先要給 writing pipe 一個「尚未完成」的名字。

```
import time, RF24

radio = RF24.RF24(22, 0) #(CE, CSN)
r_pipe_1 = b'1node'
r_pipe_2 = b'2node'
w_pipe = b'ardu'          # 命名未完成
```

2. 在初始化的部分，開啟兩個 reading pipe，管道名稱跟兩個 Arduino 端設定的一樣，也就是 1node 與 2node，並且指定在編號 1 與編號 2，也就是 openReadingPipe() 的第一個參數。Writing pipe 暫時不需要開啟，等需要回送資料到 Arduino 時再開啟就可以了。

```
radio.begin()
radio.setRetries(15, 15)
radio.openReadingPipe(1, r_pipe_1)
radio.openReadingPipe(2, r_pipe_2)

radio.printDetails()
print("NRF24L01+ is ready.")
```

3. 無窮迴圈部分的程式碼如下，重要的調整已用註解標出。主要是將 radio.available() 改 成 radio.available_pipe()，available_pipe() 這個函數會傳回資料型態為 tuple 的資料 (result, pipe)，其中 result 是布林型態，表示是否有資料進來，pipe 則顯示管道編號。所以條件判斷中使用 t[0] 就可以知道是否有資料進來，再用 t[1] 就知道是哪一個編號（意即是哪一個 Arduino 的資料）。

   要回傳資料給 Arduino 時，還是一樣要記得先停止監聽狀態，然後接著開啟 writing pipe，在這個地方根據進來資料的管道編號將 writing pipe 的名字組合好（需要跟 Arduino 端 reading pipe 名字一致），然後透過 write() 函數將資料寫出去，這樣對應的 Arduino 就會收到回傳的資料了。

```
radio.startListening()
while True:
    t = radio.available_pipe()       # 調整
    if (t[0]):                       # 調整
        len = radio.getDynamicPayloadSize()
        data = radio.read(len)
        print('{}: [{}]'.format(t[1], data.decode('utf-8')))
                                     # 調整

        radio.stopListening()
        # 組合出完整的 writing pipe 名稱
        pipe_num = str(t[1]).encode('utf-8') + w_pipe
        radio.openWritingPipe(pipe_num + w_pipe)
        radio.write(b"echo: " + data)
        radio.startListening()

    time.sleep(0.1)
```

4 執行看看。可以在 Arduino 端裝一個按鈕與一個 LED，當按鈕按下去後送資料到樹莓派，然後等收到樹莓派回傳的資料後閃爍 LED。

補充說明

若一個 Arduino 要開啟好幾個 reading pipe 負責接收來自於 2 個以上的資料來源時，想要分辨資料來自於哪一個資料來源的程式碼如下。因 Arduino 的函數庫提供的函數與 Python 略有不同，因此特別補充說明。

```
int pipe_num;
if (radio.available(&pipe_num)) {
  switch (pipe_num) {
    case 1:
      // reading pipe 1
      break;

    case 2:
      // reading pipe 2
      break;
  }
}
```