NC-200車輛檢測地磁模組是深圳市航緯電子有限公司推出的一款，使用國際先進AMR磁場技術檢測車位車輛，並能直接輸出車位車輛“有/無”判定的數位信號的地磁模組產品。該模組是一款超小體積，超低功耗，能為車位元引導系統客戶節省開發時間和成本.

產品特點：

、靈敏度高，車輛判定準確率≧98%；

、體積小，重量輕，可實現一體化 超低尺寸安裝

、可檢測卡車、拖車、客車、轎車等各類常見車型

、即時對多車道的車輛經過資訊進行採集和統計；

、檢測原理先進，屬於新型車輛檢測器，內置高精度感測器。

、不需要線圈，無需切割地面埋線圈，簡化安裝和維護的工作量

、環保、低碳，綠色：探測時不發射任何探源，對環境，對車輛無電磁干擾

技術參數和規格

尺寸：21.8\*12.8mm

功耗：28~60uA(工作)7uA(休眠)

工作電壓：2.7~3.3V

工作溫度：-30—80℃

輸出方式：TTL

車位檢測週期間隔(可設置)：1-10s

車輛檢測原理：

車輛本身含有的鐵磁物質會對車輛存在區域的地磁信號產生影響，使車輛存在區域的地球磁力線發生彎曲。當車輛經過感測器附近，感測器能夠靈敏感知到信號的變化，經信號分析就可以得到檢測目標的相關資訊。

# 矽睿車輛檢測技術建設智慧城市停車系統

隨著物聯網概念的普及，越來越多的物聯智慧方案走進我們的生活之中。上海矽睿科技有限公司(QST Corporation)作為一家以MEMS感測器及智慧傳感方案為核心競爭力，專注於物聯網業務的高科技企業，憑藉在物聯傳感領域豐富的技術積累，開發出一系列智慧物聯應用方案。

現代化的智慧城市是物聯網的重要主題之一，而智慧城市的建設離不開智慧化的交通管理。矽睿科技依靠自身掌握的磁感測器核心技術，推出了達到國際先進水準的**智慧停車檢測模組(NC-200)**；該模組功能完整，介面齊備，可靠性高，能方便的嵌入各類智慧停車管理系統，且根據實際系統需求具有良好的適應性和擴展性。目前該模組已經通過了市場客戶的認證，開始為客戶提供批量供貨服務。

當前汽車檢測技術主要分為超聲波、視頻、磁場三類。矽睿自主開發的智慧停車檢測模組(NC-200)採用了地磁檢測技術。該模組配置了公司自主研發的國內首創AMR三軸磁感測器；具有低功耗、高靈敏度和高可靠性的特點。該款磁感測器的各向異性磁阻技術(AMR)獲得美國Honeywell公司的獨家授權，公司利用該項技術研發的多款三軸磁感測器在包括智慧手機在內的各應用領域都獲得了成功。

[](http://qstcorp.w182.mc-test.com/Nova/wp-content/uploads/2016/01/1.jpg)

磁檢測模組(NC-200)

內置了地磁感測器和配套檢測演算法軟體，能直接輸出車位車輛“有/無”判定的數位信號，介面通用，方便集成。該模組超小體積，超低功耗，能為車位元引導系統客戶節省開發時間和成本。

[](http://qstcorp.w182.mc-test.com/Nova/wp-content/uploads/2016/01/2.jpg)

特點：

基於QMC5983感測器設計；

自帶溫度補償功能；

低成本(可連接RF傳輸)；

室內外停車場都可安裝；

超低功耗；

車輛判定準確率≧98%；

安裝簡便；

超低尺寸安裝8\*12.8mm

**智慧停車檢測模組應用示例**

移動用戶端收費管理方案

概述：車位檢測器安裝在停車泊位內自動探測“有車/無車”，“車來/車走”資訊，並將檢測到的資訊上傳到基於雲平臺及移動用戶端的收費系統。該系統下雲平臺可實現發佈車位元資訊，計算停車費用，記錄收費明細等功能；車主通過移動端支付停車費、查詢及引導空車位。

[](http://qstcorp.w182.mc-test.com/Nova/wp-content/uploads/2016/01/3.jpg)

特點：

無需固定式設備，也無需手持設備投入；

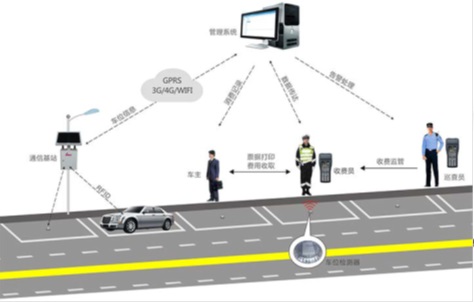
無需人工收費，運營成本低；

為停車引導資訊提供即時資料支撐；

為政府運營單位提供了空車位元情況統計分析，為決策提供了資料支撐。

人工監控與收費管理方案

概述：車位檢測器安裝在停車泊位內自動探測“有車/無車”，“車來/車走”信息。車位探測有車來，收費員做停車申請列印停車計時單給車主，並拍照取證上傳系統，車輛離開收費員結算收繳停車費並列印停車結算單給車主。

[](http://qstcorp.w182.mc-test.com/Nova/wp-content/uploads/2016/01/4.jpg)

特點：

簡便高效： 圖形化介面，操作簡單；

規範透明： 多種資訊提示，各類票據列印，支援拍照取證；

多重監控： 車位即時探測，各類告警提醒，即時資訊查詢；

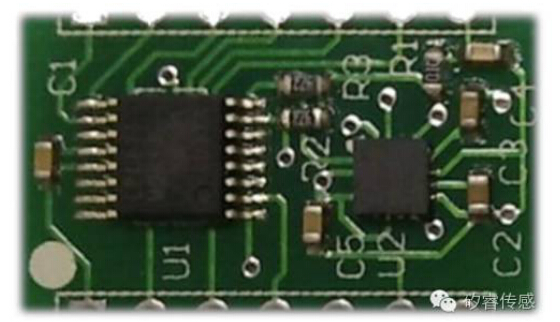
資料齊全： 記錄詳細，報表齊全，追繳有據；

推廣方便： 業務操作簡單，可迅速開展停車收費業務。

# 矽睿科技車輛檢測技術建設智慧城市停車系統

   智慧產品圈    2016-09-20

［中國，上海－2016年8月31日］隨著物聯網概念的普及，越來越多的物聯智慧方案走進我們的生活之中。現代化的智慧城市是物聯網的重要主題之一，而智慧城市的建設離不開智慧化的交通管理。矽睿科技依靠自身掌握的磁感測器核心技術，推出了達到國際先進水準的智慧停車檢測模組(NC-200)。該模組功能完整，介面齊備，可靠性高，能方便的嵌入各類智慧停車管理系統，且根據實際系統需求具有良好的適應性和擴展性。



磁檢測模組(NC-200)內置了地磁感測器和配套檢測演算法軟體，能直接輸出車位車輛“有/無”判定的數位信號，介面通用，方便集成。該模組超小體積，超低功耗，能為車位元引導系統客戶節省開發時間和成本。

矽睿科技磁檢測模組(NC-200)的特點是：

　　1.基於QMC5983感測器設計；

　　2.自帶溫度補償功能；

　　3.低成本(可連接RF傳輸)；

　　4.室內外停車場都可安裝；

　　5.超低功耗；

　　6.車輛判定準確率≧98%；

　　7.安裝簡便；

　　8.超低尺寸安裝8\*12.8mm



矽睿科技首席技術官萬虹博士表示：“當前汽車檢測技術主要分為超聲波、視頻、磁場三類。矽睿科技自主開發的智慧停車檢測模組(NC-200)採用了地磁檢測技術。該模組配置了公司自主研發的國內首創AMR三軸QMC5983磁感測器；具有低功耗、高靈敏度和高可靠性的特點。QMC5983磁感測器的各向異性磁阻技術(AMR)獲得美國Honeywell公司的獨家授權，我們利用該項技術研發的多款三軸磁感測器在包括智慧手機在內的各應用領域都獲得了成功。”

icfrom銷售總裁馮軍認為：“目前，中國的智慧停車場設備規模有50多億，並以30%的速度在快速增長。有網路資料指出，汽車後市場規模達7000億元，結合停車需求，中國智慧停車規模可達3000億元以上4000億元以下。城市化發展速度的加快對城市道路、交通、商業中心、商貿市場、物流產業園、停車場、機場等大型公共設施以及住房建設納入重要的發展環節。基於矽睿科技的磁檢測模組(NC-200)而開發的兩種應用管理方案，有這廣闊的市場應用的前景。”

矽睿科技智慧停車檢測模組應用管理方案有兩種：

第一：移動用戶端收費管理方案

移動用戶端收費管理方案是車位檢測器安裝在停車泊位內自動探測“有車/無車”，“車來/車走”資訊，並將檢測到的資訊上傳到基於雲平臺及移動用戶端的收費系統。該系統下雲平臺可實現發佈車位元資訊，計算停車費用，記錄收費明細等功能；車主通過移動端支付停車費、查詢及引導空車位。

其特點為：

　　1.無需固定式設備，也無需手持設備投入；

　　2.無需人工收費，運營成本低；

　　3.為停車引導資訊提供即時資料支撐；

　　4.為政府運營單位提供了空車位元情況統計分析，為決策提供了資料支撐。

第二：人工監控與收費管理方案

人工監控與收費管理方案是車位檢測器安裝在停車泊位內自動探測“有車/無車”，“車來/車走”信息。車位探測有車來，收費員做停車申請列印停車計時單給車主，並拍照取證上傳系統，車輛離開收費員結算收繳停車費並列印停車結算單給車主。

其特點為：

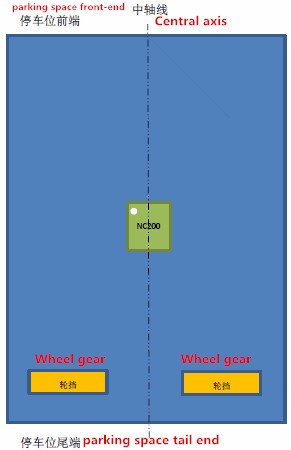
　　1.簡便高效：圖形化介面，操作簡單；

　　2.規範透明：多種資訊提示，各類票據列印，支援拍照取證；

　　3.多重監控：車位元即時探測，各類告警提醒，即時資訊查詢；

　　4.資料齊全：記錄詳細，報表齊全，追繳有據；

　　5.推廣方便：業務操作簡單，可迅速開展停車收費業務。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PIN | NAME | 管腳名稱 | 模式 | 功能說明 備註 |
| 1 | GND | 電源地 |  |  |
| 2 | RECORD | 初始化磁場資料 | IN | 低電平有效 |
| 3 | RX |  | IN | 工廠測試用 |
| 4 | ISP | ISP control port | IN, PULLUP | 工廠測試用 |
| 5 | STATE | 車位狀態 | OUT | 指示高(有)低(無) |
| 6 | GND | 電源地 |  |  |
| 7 | VCC | 電源輸入 |  |  |
| 8 | Reset | 系統重置 | IN, PULLUP | 工廠測試用 |
| 9 | SLEEP/WAKE UP |  | IN | 工廠測試用 |
| 10 | SSEL1 |  | IN | 工廠測試用 |
| 11 | TX | 串列埠輸出 | IN, PULLUP | 工廠測試用 |
| 12 | BUSY(SSEL2) | 磁場感測器的狀態指示 | IN |  |
| 13 | SEL2 | 檢查時間選擇2 | IN | 懸空為高 |
| 14 | SEL1 | 檢查時間選擇1 | IN | 懸空為高 |

# Arduino - 使用RF模組

    關於無線射頻的應用最常見的大概就是遙控玩具了所以不多做說明，要讓Arduino使用RF收發模組必須要知道Arduino的Digital pin 0 ＆ pin 1，pin 0是RXD(Receive Data)接收資料的，pin 1是TXD(Transmit Data)送出資料的，所以要發送資料的Arduino的TXD要接上RF發射模組，接收資料的Arduino的RXD要接上RF接收模組，這樣兩個電路就可以"單向"傳輸資料了，其實原本我打算使用IR的方式控制我的小六足，但是後來發現IR會有角度的問題，所以後來就決定使用RF的方式。

準備：

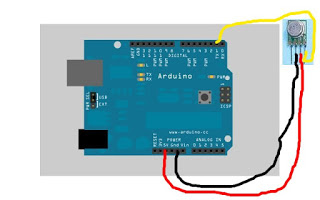
RF收發模組(比如TG-11，頻率315MHz)

[VirtualWire Library](http://dl.dropbox.com/u/16634624/VirtualWire.7z)(解壓縮放到Arduino IDE下的libraries資料夾即可)

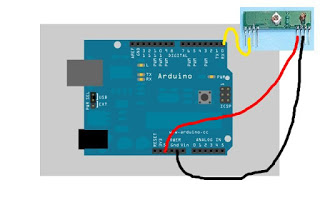
實驗：

在開始之前要先看看VirtualWire Library的一些宣告跟定義，[這裡](http://dl.dropbox.com/u/16634624/VirtualWire.pdf)或[這裡](http://www.open.com.au/mikem/arduino/VirtualWire.pdf)，vw\_set\_ptt\_inverted用來啟動RF模組，vw\_setup用來初始化傳輸速度，接著vw\_set\_tx\_pin與vw\_rx\_start分別用來設定TXD與RXD的腳位，vw\_rx\_start則讓接收模組開始接收資料，vw\_send則是要送出的資料與其長度，vw\_wait\_tx等資料送出完畢後返回，vw\_get\_message用來將接收到的資料與其長度存在buffer區塊，VW\_MAX\_MESSAGE\_LEN是預定義的資訊最大長度，兩組電路的接線也很簡單，首先兩組電路的RF模組都要接上電源以及接地(紅線與黑線)，發射資料的Arduino要將Digital pin 1(TXD)接上RF發射模組的Data腳(黃線)，接收資料的要將Digital pin 0(RXD)接上RF接收模組的Data腳(黃線)，實際腳位會因模組的不同而有所變動，但是概念都相同的，可以看下圖是TG-11收發模組的接線圖。

發射電路：

[](https://3.bp.blogspot.com/-pO6ChaZ0g5s/WMuMrCsz6EI/AAAAAAAAJ7o/AhuX91fNUhgMetUYJ6yTvz817QI885TiQCEw/s1600/at.jpg)

接收電路：

[](https://3.bp.blogspot.com/-QDWrDirfSO0/WMuMuYzF1fI/AAAAAAAAJ7s/S3jtHNOhYPEJjItPnrO9uOR2wY8Z7DlWACEw/s1600/ar.jpg)

程式：

每次送出一位元組的資料，用字元"a"與"b"作為兩個狀態，當接收電路收到"a"表示設定Digital pin 13為HIGH讓上面的LED發亮，收到"b"時則將Digital pin 13設定為LOW關閉LED。

發射端：

#include <VirtualWire.h>

const char \*a = "a";

const char \*b = "b";

void setup()

{

  vw\_set\_ptt\_inverted(true);

  vw\_setup(2400);

  vw\_set\_tx\_pin(1);

}

void loop()

{

  delay(1000);

  vw\_send((uint8\_t \*)a, strlen(a));

  vw\_wait\_tx();

  delay(1000);

  vw\_send((uint8\_t \*)b, strlen(b));

  vw\_wait\_tx();

}

接收端：

#include <VirtualWire.h>

uint8\_t buf[VW\_MAX\_MESSAGE\_LEN];

uint8\_t buflen = VW\_MAX\_MESSAGE\_LEN;

const int LED = 13;

void setup()

{

  pinMode(LED, OUTPUT);

  digitalWrite(LED,LOW);

  vw\_set\_ptt\_inverted(true);

  vw\_setup(2400);

  vw\_set\_rx\_pin(0);

  vw\_rx\_start();

}

void loop()

{

  if (vw\_get\_message(buf, &buflen)) {

    switch(buf[0])

    {

      case 'a':

      digitalWrite(LED,HIGH);

      break;

      case 'b':

      digitalWrite(LED,LOW);

      break;

    }

  }

}

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

因為我的相機被人拿去使用，所以暫時無法拍攝一些DEMO....等我買了D3100在陸續補上XDD

＝＝＝＝＝＝＝＝＝＝＝＝＝＝2011/07/08＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

如果VirtualWire這個lib跟Servo這個lib一起compile會出現像multiple definition of `\_\_vector\_11'這類error，原因似乎是兩個lib都使用同樣的timer1，可以改用SoftwareServo這個舊版的lib解決這問題，但是控制的servo似乎有數量限制，目前在弄小六足原本是打算用兩片Arduino個別控制三隻腳，這樣會各佔用9個pin，然後再用第三片Arduino去連接RF跟一些感測器，接著在讓那片Arduino跟另外兩片控制腳的去溝通XD

# Arduino 之間或與 Raspberry Pi 之間的 RF 433MHz 通訊

網頁中所使用的零件可至露天賣場訂購：

[RF433MHz(無線發射接收模組)入門學習套件 (可用於 Raspberry Pi, Arduino, 單晶片, 8051, AVR)](http://goods.ruten.com.tw/item/show?21303027865142)

[RF433MHZ入門學習套件 - 加購 RF433 MHZ 無線發射接收模組](http://goods.ruten.com.tw/item/show?21404268434165)

[RF433MHZ - ASK 無線發射接收模組套件 (可用於 Raspberry Pi, Arduino, 單晶片, 8051, AVR)](http://goods.ruten.com.tw/item/show?21404203012267)

[RF433MHZ入門學習套件 - 加購 433MHZ - ASK 無線發射接收模組套件](http://goods.ruten.com.tw/item/show?21404268475957)

|  |
| --- |
| [D:\教育事業\Arduino\References\RF433MHz\ㄆㄜ ㄊㄧㄡ ㄙˋ_ [ wireless-RF ] Arduino 之間或與 Raspberry Pi 之間的 RF 433MHz 通訊_files\DSC_0680.jpg](ㄆㄜ%20ㄊㄧㄡ%20ㄙˋ_%20%5b%20wireless-RF%20%5d%20Arduino%20之間或與%20Raspberry%20Pi%20之間的%20RF%20433MHz%20通訊_files/DSC_0680.jpg) |
| RF 模組測試 - VirtualWire 函示庫 |

許多最近購買無線發射接收模組套件的使用者，來信、留言或是線上討論了一些問題，因此我花了一些時間做了個實驗，使用兩組 Arduino ( Arduino Duemilanove 和 Arduino Pro Mini ) 並使用 VirtualWire 函示庫來做為RF 發射與接收效果的測試，而這兩塊 Arduino 板子不一定要跟我用一樣，用手邊現有的就可以了。

RF 發射或接收模組，都是使用 5V 電源輸入。Arduino Pro Mini 板作為發射控制器，RF 發射端的 data 接腳接到 Arduino Pro Mini 板子的 D11 接腳；Arduino Duemilanove 板作為接收控制器，RF 接收端的 data 接腳接到 Arduino Duemilanove 的 D12 接腳。

下面的程式需要安裝 VirtualWire 函式庫，請下載 ( [VirtualWire-1.27.zip](http://www.airspayce.com/mikem/arduino/VirtualWire/VirtualWire-1.27.zip) ) 壓縮檔之後直接解壓縮在 Arduino 的 libraries 目錄下，然後打開 Arduino IDE，此時就會看到在 File \ Examples \ VirtualWire 下看到 client、receiver、transmitter 和 server 四個範常式式，下麵是我們用到的兩個用來測試的程式，程式列表如下：

transmitter.pde, Arduino Pro Mini, @5V, digital IO pin number 12 --> [ RF data pin ]

// transmitter.pde

// Simple example of how to use VirtualWire to transmit messages

// Implements a simplex (one-way) transmitter with an TX-C1 module

// See VirtualWire.h for detailed API docs

// Author: Mike McCauley (mikem@airspayce.com)

// Copyright (C) 2008 Mike McCauley

// $Id: transmitter.pde,v 1.3 2009/03/30 00:07:24 mikem Exp $

#include <VirtualWire.h>

void setup(){

Serial.begin(9600);// Debugging only

Serial.println("setup");

// Initialise the IO and ISR

// vw\_set\_ptt\_inverted(true);// Required for DR3100

vw\_setup(2000);// Bits per sec

}

void loop(){

const char \*msg = "hello";

digitalWrite(13, true);// Flash a light to show transmitting

vw\_send((uint8\_t \*)msg, strlen(msg));

vw\_wait\_tx();// Wait until the whole message is gone

digitalWrite(13, false);

delay(200);

}

transmitter.pde 在確認發送 "hello" 資料出去之後，會再等待 200 ms 再繼續下一次傳送，資料傳送的速度每秒 2000 bits。

receiver.pde, Arduino Duemilanove, @5V, digital IO pin number 11 --> [ RF data pin ]

// receiver.pde

//

// Simple example of how to use VirtualWire to receive messages

// Implements a simplex (one-way) receiver with an Rx-B1 module

//

// See VirtualWire.h for detailed API docs

// Author: Mike McCauley (mikem@airspayce.com)

// Copyright (C) 2008 Mike McCauley

// $Id: receiver.pde,v 1.3 2009/03/30 00:07:24 mikem Exp $

#include <VirtualWire.h>

void setup()

{

Serial.begin(9600);// Debugging only

Serial.println("setup");

// Initialise the IO and ISR

//vw\_set\_ptt\_inverted(true);// Required for DR3100

vw\_setup(2000);// Bits per sec

vw\_rx\_start();// Start the receiver PLL running

}

void loop()

{

uint8\_t buf[VW\_MAX\_MESSAGE\_LEN];

uint8\_t buflen = VW\_MAX\_MESSAGE\_LEN;

if (vw\_get\_message(buf, &buflen))// Non-blocking

{

int i;

digitalWrite(13, true);// Flash a light to show received good message

// Message with a good checksum received, dump it.

Serial.print("Got: ");

for (i = 0; i < buflen; i++)

{

Serial.print(buf[i], HEX);

Serial.print(" ");

}

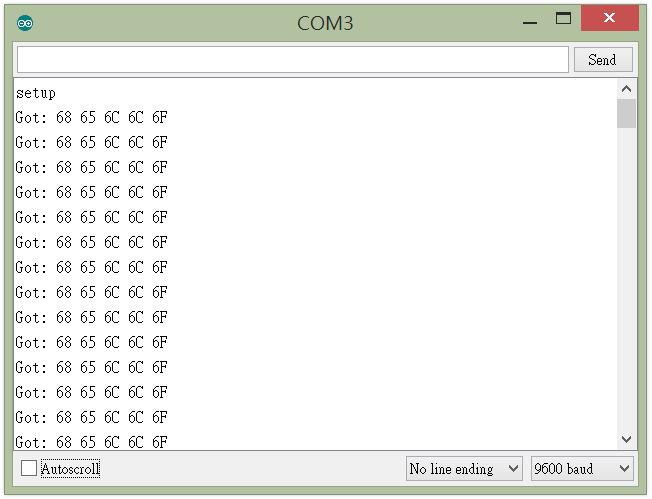
Serial.println("");

digitalWrite(13, false);

}

}

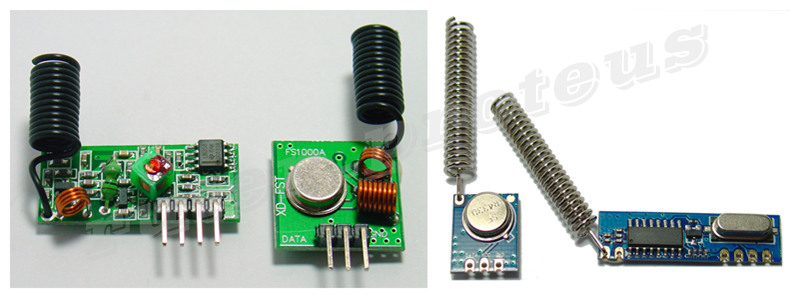
reveiver.pde 設定 Bits per sec 為 2000 與 transmitter.pde 一樣，只要正確接收傳送過來的資料，就會將每個字元以 16 進位元的方式，輸出到 Serial Monitor 上。如下圖所示，將 "hello" 以 68 65 6C 6C 6F 來做表示。

[](ㄆㄜ%20ㄊㄧㄡ%20ㄙˋ_%20%5b%20wireless-RF%20%5d%20Arduino%20之間或與%20Raspberry%20Pi%20之間的%20RF%20433MHz%20通訊_files/COM3_output.png)

receiver.pde 接收到的資料以16進位元的方式顯示在 Serial Monitor 上

測試：

記得在賣場的介紹中說過，下麵這兩組無線模組，發射與接收端可以交換著使用，不限定一定要同組才能做通訊。為了測試上的方便，左邊這一組稱為 RF433M ( -T：發射；-R：接收 )，右邊這一組稱為 ASK ( -T：發射；-R：接收 )。

[](ㄆㄜ%20ㄊㄧㄡ%20ㄙˋ_%20%5b%20wireless-RF%20%5d%20Arduino%20之間或與%20Raspberry%20Pi%20之間的%20RF%20433MHz%20通訊_files/DSC03574-crop-horz-blog.png)

|  |
| --- |
| 左邊 RF433M：( 接收、發射 )；右邊：ASK ( 發射、接收 ) |

我的測試方法很簡單，固定一組接收模組接收訊號並放置在一樓的樓梯旁邊的桌子，然後由 1 樓至 4 樓分別使用不同的發射端發射訊號，再由樓下一人監看接收端燈的亮滅速度。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ASK-R |  | 1F | 2F | 3F | 4F |
| RF433M-T | V | V | V | V |
| ASK-T | V | V | V | V |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RF433M-R |  | 1F | 2F | 3F | 4F |
| RF433M-T | V | V | X | X |
| ASK-T | V | V | V | V |

結果就如上表所示，ASK 這一組無線發射接收模組的穿牆效果很好，無論是作為接收或是發射都能達到應有的效果，這也是 ASK 這一組與眾不同之處。所以若是要用在遮蔽物比較多的場合，ASK 這一組會是一個好的選擇。

另外，說到 VirtualWire 函式庫，它的傳送方式有點像是網路的 [UDP](http://zh.wikipedia.org/wiki/%E7%94%A8%E6%88%B7%E6%95%B0%E6%8D%AE%E6%8A%A5%E5%8D%8F%E8%AE%AE) 通訊，但只傳送字元陣列不能傳輸數字，因此要傳輸數字格式，例如 12.34，就必須將其拆成 '1', '2', '.', '3', '4' 傳送，這是在傳送資料時比較麻煩的地方。但列舉一段為何使用 VirtualWire 函式庫而不使用 UART 的討論：

As discussed in the RFM documentation, ASK receivers require a burst of training pulses to synchronize the transmitter and receiver, and also requires good balance between 0s and 1s in the message stream in order to maintain the DC balance of the message. UARTs do not provide these. They work a bit with ASK wireless, but not as well as this code.

也就是使用 VirtualWire 函式庫會比使用 UART 來的好。

Arduino <-- ( RF433M or ASK ) --> Raspberry Pi 之間的通訊：

Arduino 與 Raspberry Pi 使用 RF433M 模組的通訊 ( ASK 模組同樣也適用 )，我在網路上找到兩篇網頁，裡面所用的函式庫也可以用來控制市面上的一些使用 315 MHz 和 433 MHz 的無線插座。但書到底其實就是逆向工程，先將訊號擷取出來做分析，然後找出規則再傳送編碼回去控制插座開或關，傳送的過程並不是只有一次，而是將控制插座開或關的編碼傳送多次以確保遠端收到訊號，詳細可以去看一下下面談到的程式碼或是參考部落格的這篇文章 "[[ Wireless-RF] 使用樹莓派模擬 HT12E 遙控器編碼晶片的編碼格式](http://ruten-proteus.blogspot.tw/2013/03/Wireless-RF-RPi-sim-HT12E.html)"。

在這兩個網頁中，作者已經將接線與程式碼解釋得很清楚，只要搞清楚 Arduino 與 Raspberry Pi 之間如何使用 433MHz 模組做資料傳輸，不管是賣場的 RF433M 或是 ASK 模組，要怎麼應用就看自己如何發揮了  !!! 例如傳遞現在外界環境中的溫度、大氣壓力、照度、濕度...等。

接下來，我將兩個網頁中的內容大抵說明一下，做個引導 ! 程式中所需要用到的程式可以到下面連結先下載：

RCSwitch 函式庫 ( Arduino 版本 )：作為發射與接收時使用的函式庫。

RCSwitch 函式庫 ( Raspberry Pi 版本 )：做為發射端時使用的函式庫。

[433Utils](https://github.com/ninjablocks/433Utils)：混合上面兩個函式庫的 Raspberry Pi 和 Arduino 的 RF 工具程式，包括 Arduino 的 sketch 檔案與 Raspberry Pi 的命令列程式。Raspberry Pi 使用 433Utils 之前必須先安裝 WiringPi 函式庫，然後編譯產生所需要的程式 ( [WiringPi 安裝](https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/" \t "_blank)可參考 Dragon 的網頁，或是直接參考下面網頁中的安裝說明 )。

\* [433Mhtz RF communication between Arduino and Raspberry Pi: Arduino as receiver](http://www.homautomation.org/2014/03/02/433mhtz-rf-communication-between-arduino-and-raspberry-pi-arduino-as-receiver/)

發射端：Raspberry Pi

接收端：Arduino

Raspberry Pi 作為發射端，使用 RCSwitch 函式庫撰寫發射端的程式 ( source code：433Utils / RPi\_utils / codesend.cpp )。

設定 GPIO #0 ( line 24 ) 接到 RF Module 的 DATA 接腳；接收命令列參數並轉換為數字傳遞給 code ( line 27 )；使用 RCSwitch 函式庫之前，先初始化 wiringPi 函示庫 ( line 29 )；初始化 RCSwitch 函式庫，並設定資料傳送的腳位為輸出 ( line 31 ... line 32 )；傳送 24-bit 長度的 code ( line 34 )

Utils / RPi\_utils / codesend.cpp

/\*

'codesend' hacked from 'send' by @justy

- The provided rc\_switch 'send' command uses the form systemCode, unitCode, command

which is not suitable for our purposes. Instead, we call

send(code, length);// where length is always 24 and code is simply the code

we find using the RF\_sniffer.ino Arduino sketch.

Usage: ./codesend decimalcode

(Use RF\_Sniffer.ino to check that RF signals are being produced by the RPi's transmitter)

\*/

#include "RCSwitch.h"

#include <stdlib.h>

#include <stdio.h>

int main(int argc, char \*argv[]) {

// This pin is not the first pin on the RPi GPIO header!

// Consult https://projects.drogon.net/raspberry-pi/wiringpi/pins/

// for more information.

int PIN = 0;

// Parse the firt parameter to this command as an integer

int code = atoi(argv[1]);

if (wiringPiSetup () == -1) return 1;

printf("sending code[%i]\n", code);

RCSwitch mySwitch = RCSwitch();

mySwitch.enableTransmit(PIN);

mySwitch.send(code, 24);

return 0;

}

Utils / RPi\_utils / RCSwitch.cpp, enableTransmit()

/\*\*

\* Enable transmissions

\*

\* @param nTransmitterPin Arduino Pin to which the sender is connected to

\*/

void RCSwitch::enableTransmit(int nTransmitterPin) {

this->nTransmitterPin = nTransmitterPin;

pinMode(this->nTransmitterPin, OUTPUT);

}

send() 在傳送之前會將數字資料轉換為 length 長度 bit 的二進制 ASC Code 的 '0' 或 '1'，前方沒用到的 bit 全部補 '0'，最後將資料整合再傳送。

Utils / RPi\_utils / RCSwitch.cpp, send()

void RCSwitch::send(unsigned long Code, unsigned int length) {

this->send( this->dec2binWzerofill(Code, length) );

}

Utils / RPi\_utils / RCSwitch.cpp, enableReceive()

void RCSwitch::enableReceive() {

if (this->nReceiverInterrupt != -1) {

RCSwitch::nReceivedValue = NULL;

RCSwitch::nReceivedBitlength = NULL;

wiringPiISR(this->nReceiverInterrupt, INT\_EDGE\_BOTH, &handleInterrupt);

}

}

Utils / RPi\_utils / RCSwitch.cpp, dec2binWzerofill()

/\*\*

\* Turns a decimal value to its binary representation

\*/

char\* RCSwitch::dec2binWzerofill(unsigned long Dec, unsigned int bitLength){

static char bin[64];

unsigned int i=0;

while (Dec > 0) {

bin[32+i++] = ((Dec & 1) > 0) ? '1' : '0';

Dec = Dec >> 1;

}

for (unsigned int j = 0; j< bitLength; j++) {

if (j >= bitLength - i) {

bin[j] = bin[ 31 + i - (j - (bitLength - i)) ];

}else {

bin[j] = '0';

}

}

bin[bitLength] = '\0';

return bin;

}

Arduino 作為接收端，使用 RCSwitch 函式庫撰寫接收端的程式 ( source code：433Utils / Aduino\_sketches / RF\_Sniffer / RF\_Sniffer.ino )。

程式設定 Arduino 的 Serial Port 速度為 9600 bps ( line 13 )，接收到的資料會在這邊做顯示 ( line 27 ... line 33 )。

RCSwitch 接收的方式需要用到中斷，以 Arduino Duemilanove 為例，有兩個中斷 ( INT0, INT1 ) 可以用 ( 位於板子的右下角，也就是 Digital IO pin 號碼 2 和 3 )。程式使用 INT0 負責接收傳送過來的訊號，因此 line 14 函式裡面的參數設為 0，並且在每次訊號發生變化時都會呼叫中斷處理程式。

|  |
| --- |
| [D:\教育事業\Arduino\References\RF433MHz\ㄆㄜ ㄊㄧㄡ ㄙˋ_ [ wireless-RF ] Arduino 之間或與 Raspberry Pi 之間的 RF 433MHz 通訊_files\tumblr_mr9ynw2X1H1s5t695o1_1280.png](ㄆㄜ%20ㄊㄧㄡ%20ㄙˋ_%20%5b%20wireless-RF%20%5d%20Arduino%20之間或與%20Raspberry%20Pi%20之間的%20RF%20433MHz%20通訊_files/tumblr_mr9ynw2X1H1s5t695o1_1280.png) |
| Arduino Duemilanove 接腳圖, 來源：[http://pighixxx.tumblr.com//image/57805840162](http://pighixxx.tumblr.com//image/57805840162" \t "_blank) |

一但有資料進來之後，就可以開始開始接收資料 ( line 19 )；先確認資料是否始無效的字元輸入 ( line 23 ... line 24 )，如果不是就可以接收並顯示資料的相關資訊 ( line 25 ... line 34 )，所有的資料都可以使用 Arduino 的 Serial Monitor 觀看。

Utils / Aduino\_sketches / RF\_Sniffer / RF\_Sniffer.ino

/\*

RF\_Sniffer

Hacked from http://code.google.com/p/rc-switch/

by @justy to provide a handy RF code sniffer

\*/

#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {

Serial.begin(9600);

mySwitch.enableReceive(0);// Receiver on inerrupt 0 => that is pin #2

}

void loop() {

if (mySwitch.available()) {

int value = mySwitch.getReceivedValue();

if (value == 0) {

Serial.print("Unknown encoding");

} else {

Serial.print("Received ");

Serial.print( mySwitch.getReceivedValue() );

Serial.print(" / ");

Serial.print( mySwitch.getReceivedBitlength() );

Serial.print("bit ");

Serial.print("Protocol: ");

Serial.println( mySwitch.getReceivedProtocol() );

}

mySwitch.resetAvailable();

}

}

下麵的程式位於 Arduino 函式庫 libraries 目錄下，但必須要有安裝 RCSwitch 函式庫。

{Arduino libraries} / RCSwitch / RCSwitch.cpp

/\*\*

\* Enable receiving data

\*/

void RCSwitch::enableReceive(int interrupt) {

this->nReceiverInterrupt = interrupt;

this->enableReceive();

}

void RCSwitch::enableReceive() {

if (this->nReceiverInterrupt != -1) {

RCSwitch::nReceivedValue = NULL;

RCSwitch::nReceivedBitlength = NULL;600

attachInterrupt(this->nReceiverInterrupt, handleInterrupt, CHANGE);

}

}

\* [433Mhtz RF communication between Arduino and Raspberry Pi: Raspberry Pi as receiver](http://www.homautomation.org/2013/09/21/433mhtz-rf-communication-between-arduino-and-raspberry-pi/)

發射端：Arduino

接收端：Raspberry Pi

Arduino 做為發射端，使用 RCSwitch 函式庫撰寫發射端的程式。程式修改自 { Arduino libries } / RCSwitch / examples / SendDemo / SendDemo.pde，但忽略其中幾行程式碼，只留下傳送數字的代碼；不過還是可以直接使用 SendDemo.pde 做測試，不需要使用連結網頁上的。

程式設定 Digital IO Pin 號碼 10 接到發射模組的 DATA 接腳 ( line 17 )；然後重複傳送四種開關 ON / OFF 的程式碼出去 ( line 30 ... line 57 )，連結網頁中只使用到 ( line 38 ... line 43, line 56 )。

{Arduino libraries} / RCSwitch /examples/ SendDemo / SendDemo.pde

/\* Example for different sending methods

http://code.google.com/p/rc-switch/\*/

#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

void setup() {

Serial.begin(9600);

// Transmitter is connected to Arduino Pin #10

mySwitch.enableTransmit(10);

// Optional set pulse length.

// mySwitch.setPulseLength(320);

// Optional set protocol (default is 1, will work for most outlets)

// mySwitch.setProtocol(2);

// Optional set number of transmission repetitions.

// mySwitch.setRepeatTransmit(15);

}

void loop() {

/\* See Example: TypeA\_WithDIPSwitches \*/

mySwitch.switchOn("11111", "00010");

delay(1000);

mySwitch.switchOn("11111", "00010");

delay(1000);

/\* Same switch as above, but using decimal code \*/

mySwitch.send(5393, 24);

delay(1000);

mySwitch.send(5396, 24);

delay(1000);

/\* Same switch as above, but using binary code \*/

mySwitch.send("000000000001010100010001");

delay(1000);

mySwitch.send("000000000001010100010100");

delay(1000);

/\* Same switch as above, but tri-state code \*/

mySwitch.sendTriState("00000FFF0F0F");

delay(1000);

mySwitch.sendTriState("00000FFF0FF0");

delay(1000);

delay(20000);

}

Raspberry Pi 作為接收端，使用 Pi 版本的 RCSwitch 函式庫撰寫接收端的程式 RFSniffer：RFSniffer.cpp 是使用 Raspberry Pi 版本的 RCSwitch 函式庫寫的程式，使用 WiringPi 的中斷處理程式負責處理 DATA 接腳接收到的訊號，程式碼與 Arduino 版本的差不多。

使用 Pi 版本的 RCSwitch 函式庫之前，必須初始化 wiringPI 函式庫 ( RFSniffer.cpp, line 25 ... line 26 )；接收端使用 GPIO #2 與 RF 接收模組的 DATA 接腳連接，並且設定這支接腳只要準位做變化 ( 不管是高或低 ) 都會引發中斷 ( RCSwitch.cpp, line 439 ... line 450 )

/\* RF\_Sniffer

Hacked from http://code.google.com/p/rc-switch/

by @justy to provide a handy RF code sniffer \*/

#include "RCSwitch.h"

#include <stdlib.h>

#include <stdio.h>

RCSwitch mySwitch;

int main(int argc, char \*argv[]) {

// This pin is not the first pin on the RPi GPIO header!

// Consult https://projects.drogon.net/raspberry-pi/wiringpi/pins/

// for more information.

int PIN = 2;

if(wiringPiSetup() == -1)

return 0;

mySwitch = RCSwitch();

mySwitch.enableReceive(PIN);// Receiver on inerrupt 0 => that is pin #2

while(1) {

if (mySwitch.available()) {

int value = mySwitch.getReceivedValue();

if (value == 0) {

printf("Unknown encoding");

} else {

printf("Received %i\n", mySwitch.getReceivedValue() );

}

mySwitch.resetAvailable();

}

}

exit(0);

}

Utils / RPi\_utils / RCSwitch.cpp

/\*\* Enable receiving data \*/

void RCSwitch::enableReceive(int interrupt) {

this->nReceiverInterrupt = interrupt;

this->enableReceive();

}

void RCSwitch::enableReceive() {

if (this->nReceiverInterrupt != -1) {

RCSwitch::nReceivedValue = NULL;

RCSwitch::nReceivedBitlength = NULL;

wiringPiISR(this->nReceiverInterrupt, INT\_EDGE\_BOTH, &handleInterrupt);

}

}

結論：

以上就是兩個連結網頁中程式碼的說明，照著連結網頁中的接線並載入相關的程式，就可以實現 Arduino 與 Raspberr Pi 之間的無線通訊，不管是使用 RF433M 或是 ASK 哪一種無線發射接收模組都可達到相同的功能。

在這篇網頁的開頭，也做了 RF433M 與 ASK 兩種無線模組的發射與接收的實驗，這是在我自家的環境下所做的，實際應用上必須實地做測試才會知道效果好的什麼程度，環境不同測試的效果也會不同 !!!

