

低功率廣域網路

LoRa[®]

[感測器接收與控制]

Internet of Things

Lora簡介

LoRa技術是什麼？









LoRa技術是為了創建長距離通信連結的物理層無線調製方式。許多傳統的無線系統使用物理層頻移鍵控（FSK）調製，因為它是十分高效的低功耗方案。

LoRa技術基於線性Chirp擴頻調製，延續了移頻鍵控調製的低功耗特性，但是大大增加了通信範圍。Chirp擴頻調製已經在軍事和航天通信方面應用了幾十年，因為它具有長距離傳輸，以及很好的抗干擾性，而LoRa則是它第一次用作商業用途。

LoRa來源於Long Range這個單詞，所以它的最大優點就是長距離傳輸。單個網關或者基站可以覆蓋整個城市或者數百千米。

LPWAN的優點是什麼？

一項技術無法覆蓋所有的物聯網應用場景。WiFi和BTLE主要應用於個人設備相關的應用。蜂窩技術主要應用於需要高數據吞吐量，以及需要供電的應用場景。LPWAN的應用場景包括：長電池壽命，並且傳感器和應用在長距離下，只需要每小時只要傳遞幾次數據。

	Local Area Network Short Range Communication	Low Power Wide Area (LPWAN) Internet of Things	Cellular Network Traditional M2M
	40%	45%	15%
	Well established standards In building	Low power consumption Low cost Positioning	Existing coverage High data rate
	Battery Live Provisioning Network cost & dependencies	High data rate Emerging standards	Autonomy Total cost of ownership
	Bluetooth 4.2   WiFi	LoRa 	GSMA  3G+ / H+  4G

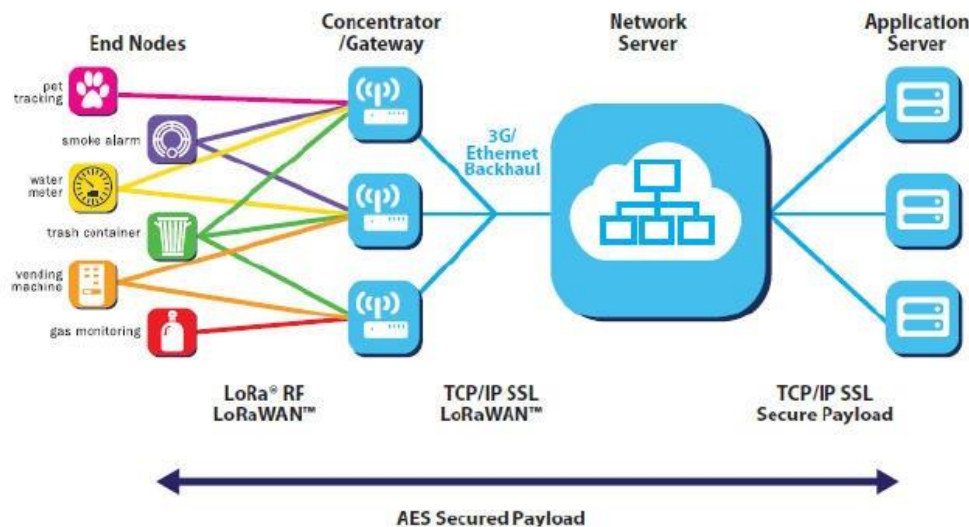
LPWAN物聯網網絡的要素有哪些？

- 網絡架構
- 通信範圍
- 電池壽命和功耗
- 抗干擾性
- 網絡容量（網絡結點的最大數目）
- 網絡安全性
- 單向或者雙向傳播
- 服務的應用

LoRaWAN是什麼呢？

• 網絡架構

許多部署好的網絡都採用網狀架構。在一個網狀網絡中，每個單獨的端結點傳遞信息到其他結點，增加通信範圍以及網絡的小區容量。當結點增加的時候，它也增加了複雜度，減少了網絡容量，也減少了電池壽命，因為結點要收發從他們不相干的結點傳遞來的信息。長距離星型架構，由於長距離連接性，從而減少了電池壽命。



LoRaWan的網絡節點不和特定網關相綁定。相反，通過結點傳輸的數據，被多個網關接受。

每個網關，通過一些信號隧道（例如蜂窩網, 乙太網, 衛星網, Wi-Fi) 將從端結點收到的數據包發向雲端。

由網絡伺服器理解和處理複雜任務，包括管理網絡，過濾冗餘數據包，進行安全性檢查，通過優化的網關進行調度確認，執行自適應數據率，等等。

如果一個結點是移動的，或者運動的時候沒有網關之間的切換，這就需要啟用資源追蹤應用，一個基於垂直物聯網目標應用程式。

• 電池壽命

LoRaWAN網絡的結點是異步通信的，並且當它們有數據準備發送的時候，會採用事件驅動，或者調度機制進行通信。這個協議採用了阿羅哈法。在一個網狀網絡或者一個異步網絡中，例如蜂窩網，結點必須頻繁的被喚醒，來同步網絡和檢查消息。這種同步，大大的消耗了能量，成為電池壽命減少的一個重要原因。在最近的GSMA對於LPWAN空間的各項技術研究和對比中，LoRaWan比其他技術要優3到5倍。

• 網絡容量

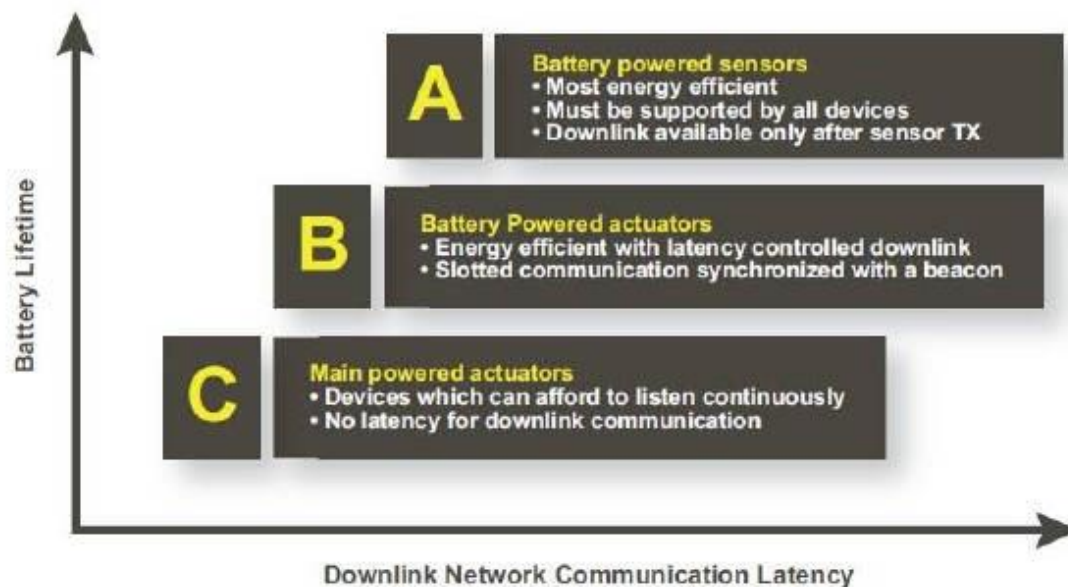
為了保證星型網絡的可行，網關必須有能力處理大量來自各個結點的信息。LoRaWAN的高網絡容量通過利用自適應的數據率，以及網關內多通道多數據機的收發器，來實現。

最重要的因素，是並發通道數，數據率，負載長度，已經結點間傳輸數據頻度。因為LoRa是擴頻基礎上的調製，所以每個信號基本上是正交的。當擴展因子變化的時候，有效的數據率也跟著改變。網關利用這種性能，能夠在同時在同一信道上接受不同的數據率。如果一個結點具有一個好的連接，並且靠近網關，網關當然可以利用最小的數據率，並且填充必須要更長的可用的頻譜。通過將數據率提高，空中的時間將被縮短，開放更多的潛在時間來傳輸。自適應的數據率也優化了結點的電池壽命。

為了應用自適應數據率，需要對稱的上下行連接，並且有足夠的下行能力。這些功能讓LoRaWAN能夠有很高的容量，並且讓網絡可擴展。一個網絡通過最少數量的設施部署，在擴容的時候，可添加更多的網關，調整數據率，減少對於其他網關的串擾，並且擴大能力6-8倍。其他LPWAN方案沒有LoRaWan的擴容新，因為技術取捨的緣故，限制了下行能力，或者讓下行範圍和上行範圍不對稱。

• 設備分類

不同應用的端設備具有不同的要求。在為了優化端應用配置，LoRaWAN使用了不同的設備類型。設備分類，權衡了網絡下行通信的延時和電池壽命。在一個執行器類型的應用中，下行通信演示是十分重要的因素。



雙向終端設備（類型A）：A類型端設備，允許雙向通信，每個端設備的上行傳輸，伴隨兩個端的下行接收窗口，根據自身的傳輸需要，以及採取ALOHA類型協議的小變化。A類型操作是最低功耗的端系統。





雙向終端設備具有固定接受時間槽（類型B）：對於A類型設備隨機接受窗口來說，B類型設備具有固定時間接受窗口。它根據來自網關的接受指示，來決定何時打開接受窗口。伺服器就可以知道端設備何時再傾聽數據。

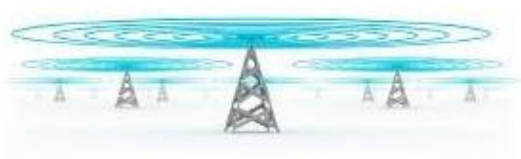
雙向終端設備具有最大接受時間槽（類型C）：C類型設備具有連續打開的接受窗口，只有在數據傳輸時才關閉。

• 安全性

安全性自然是十分重要的因素。LoRaWan具有兩層安全防護：一個是網絡，一個是應用。網絡安全保證了結點的合法性，應用的安全保證了網絡操作不能訪問用戶的應用數據。它使用AES加密來交換IEEE EUI64標誌符。

與傳統的物聯網網絡對比的情況？

<p>2000 apartments 80 buildings/staircases Sensor cost equal</p>		 <p>An average of 5 sensors/apartment (temperature, humidity and warm water consumption). Fire detection optional. Potential IoT revenue with LoRa.</p>																																														
 <p>400 repeaters 80 concentrators</p>		 																																														
<p>WM-Bus</p> <table border="1"> <tr> <td rowspan="4">CAPEX</td><td>Design, prep</td><td>4 000</td><td>€100/hour</td></tr> <tr> <td>Repeaters</td><td>32 000</td><td>€80/each</td></tr> <tr> <td>Concentrators</td><td>20 000</td><td>€250/each</td></tr> <tr> <td>Installation</td><td>24 200</td><td>€55/hour</td></tr> <tr> <td colspan="2">SUM</td><td colspan="2">€ 80 200</td></tr> </table> <table border="1"> <tr> <td rowspan="2">REVENUE</td><td>Business logic & Statistics</td><td rowspan="2">36 842</td><td rowspan="2">Yearly occurring</td></tr> <tr> <td>Billing, Invoice</td></tr> </table>		CAPEX	Design, prep	4 000	€100/hour	Repeaters	32 000	€80/each	Concentrators	20 000	€250/each	Installation	24 200	€55/hour	SUM		€ 80 200		REVENUE	Business logic & Statistics	36 842	Yearly occurring	Billing, Invoice	<p>LoRa Network Operator</p> <table border="1"> <tr> <td rowspan="4">CAPEX</td><td>Design, prep</td><td>1 600</td><td>€100/hour</td></tr> <tr> <td>GW</td><td>6 000</td><td>€1500/each (500apts/GW)</td></tr> <tr> <td>Installation</td><td>880</td><td>€55/hour</td></tr> <tr> <td>SUM</td><td colspan="2">€ 8 480</td></tr> </table> <table border="1"> <tr> <td rowspan="3">REVENUE</td><td>Business logic & Statistics</td><td colspan="2">Yearly occurring</td></tr> <tr> <td>Billing, Invoice</td><td colspan="2">Yearly occurring</td></tr> <tr> <td>Other IoT services</td><td colspan="2">Occuring</td></tr> </table>		CAPEX	Design, prep	1 600	€100/hour	GW	6 000	€1500/each (500apts/GW)	Installation	880	€55/hour	SUM	€ 8 480		REVENUE	Business logic & Statistics	Yearly occurring		Billing, Invoice	Yearly occurring		Other IoT services	Occuring	
CAPEX	Design, prep		4 000	€100/hour																																												
	Repeaters		32 000	€80/each																																												
	Concentrators		20 000	€250/each																																												
	Installation	24 200	€55/hour																																													
SUM		€ 80 200																																														
REVENUE	Business logic & Statistics	36 842	Yearly occurring																																													
	Billing, Invoice																																															
CAPEX	Design, prep	1 600	€100/hour																																													
	GW	6 000	€1500/each (500apts/GW)																																													
	Installation	880	€55/hour																																													
	SUM	€ 8 480																																														
REVENUE	Business logic & Statistics	Yearly occurring																																														
	Billing, Invoice	Yearly occurring																																														
	Other IoT services	Occuring																																														



Long Range

- Greater than cellular
- Deep indoor coverage
- Star topology



Max Lifetime

- Low power optimized
- 10-20yr lifetime
- >10x vs cellular M2M



Multi-Usage

- High capacity
- Multi-tenant
- Public network



Low Cost

- Minimal infrastructure
- Low cost end node
- Open SW

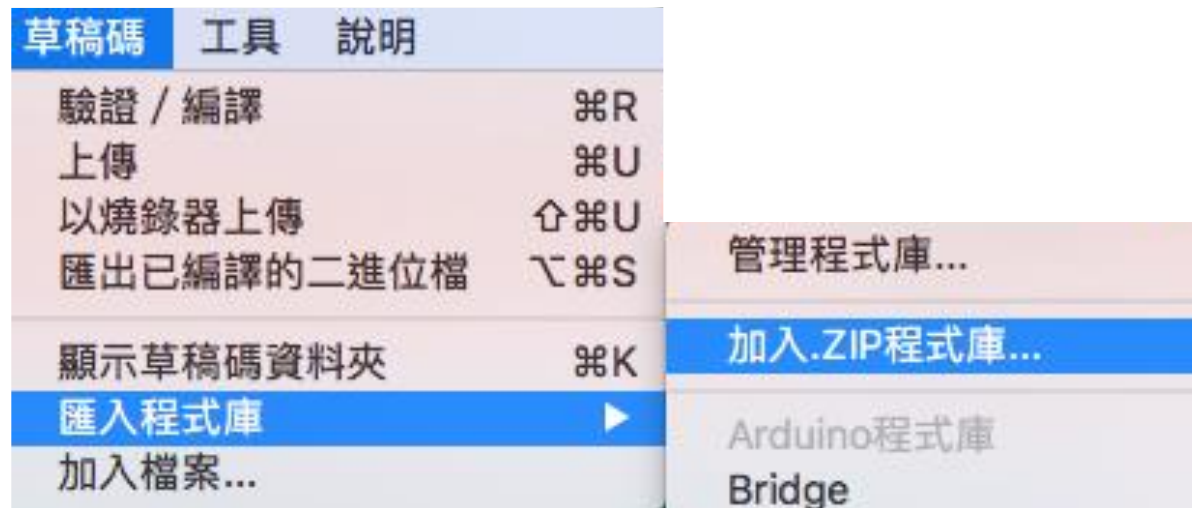
程式碼架構

首先需要先下載必要的檔案：

<https://goo.gl/b9nd78>

<https://goo.gl/YnUWK2>

- 透過 IDE 的 草稿碼->匯入程式庫->加入.ZIP程式庫，選擇下載的檔案



- 加入函式庫：

```
#include <SPI.h>
#include <RH_RF95.h>
```

- 建立Lora實體：

```
RH_RF95 rf95;
```

- 宣告群組及設備代表字元：

```
char head = '@';      // 群組代表字元
char ID = '1';         // 設備代表字元
```

- 初始化(setup)：

```
Serial.begin(9600);    // 開啟序列埠
while (!Serial);       // 等待序列埠開啟完成
//Lora初始化
if (!rf95.init())      // lora模組初始化並回傳是否成功
    Serial.println("init failed"); // 初始化失敗回傳訊息
```


- 訊息發送的函式：
在函式呼叫時，將字串透過Lora發送

```
void loraSend(String d){           // 將要發送的訊息儲存到 d 發送變數
    d = String(head) + String(ID) + d;
                                   // 在發送變數前面加上群組及設備代表字元
    int dl = d.length();           // 宣告變數存放發送長度
    uint8_t data[dl+1];           // 宣告 uint8_8 格式變數
    for(int i = 0;i<dl;i++){       // 將發送變數儲存到 uint8_t 變數
        data[i]=d[i];
    }
    data[dl] = 0;                  // 結束字元為0
    rf95.send(data, sizeof(data)); // 發送Lora訊息
    rf95.waitPacketSent();         // 等待發送
}
```

- 訊息接收的函式：
輸入設備代表字元，回傳該設備發送的訊息字串

```
String loraRead(char id){    // 將設備代表字元儲存到 id 代表變數
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];    // 宣告接收訊息變數
    uint8_t len = sizeof(buf);    // 宣告變數儲存接收訊息變數長度
    if (rf95.waitAvailableTimeout(3000)){
        // 設定等待訊息時間3秒，沒有訊息回傳false
    }
    if (rf95.recv(buf, &len)){    // 接收Lora訊息，訊息接收失敗回傳false
        Serial.print("//got reply: ");
        Serial.println((char*)buf); // 顯示接收到的訊息
        String s = String((char*)buf);    // 將訊息轉換為字串
        if(s[0]==head){    // 檢查群組代表字元是否符合格式
            if(s[1]==id){    // 檢查設備代表是否與代表變數相同
                s.remove(0,2);    // 移除群組及設備代表字元
                return s;    // 回傳
            }
        }
    }
}
```

```
}else{  
    Serial.println("//ID does not match");           // 設備代表字元不符  
    return "ID does not match";  
}  
}else{  
    Serial.println("//Format does not match");       // 格式不符  
    return "Format does not match";  
}  
}else{  
    Serial.println("//recv failed");                 // 接收訊息失敗  
    return "failed";  
}  
}else{  
    Serial.println("//No reply, is rf95_server running?"); //沒有訊息  
    return "No reply";  
}  
}
```

命令格式：

- `loraSend("發送訊息")`

將訊息透過Lora發送，發送時會在開頭附加群組及設備代表字元：

```
loraSend( "Hello world!" ); //發送 Hello wrold
```

- `loraRead('2')`

接收訊息，如果為相同群組且設備代表字元為 2 則回傳訊息字串：

```
String message = loraRead( '2' );
```

```
//宣告 message 字串變數接收訊息：
```

```
//群組及設備字元相同回傳訊息字串
```

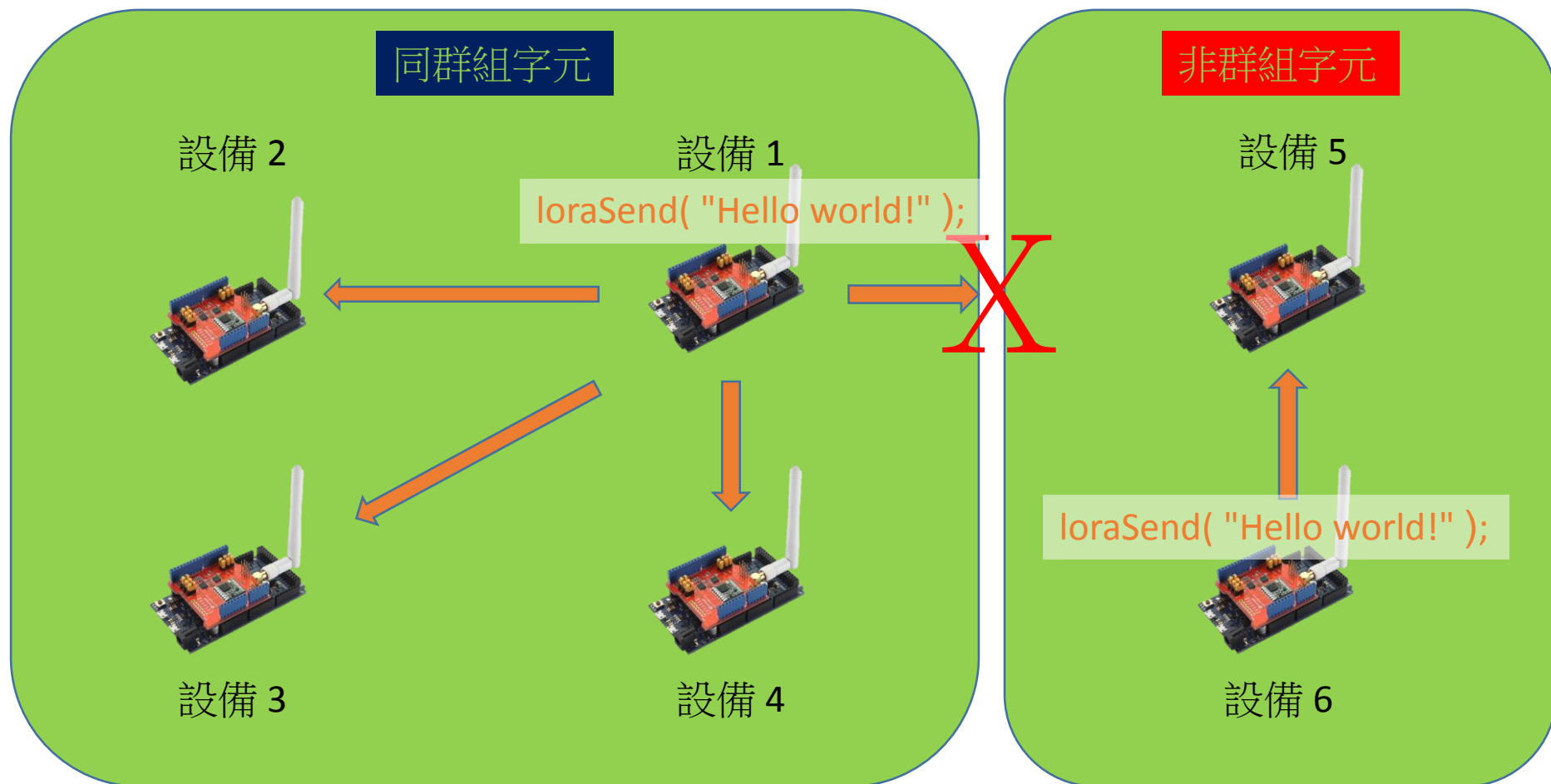
```
//沒有訊息回傳 No reply 字串
```

```
//接收訊息失敗回傳 failed 字串
```

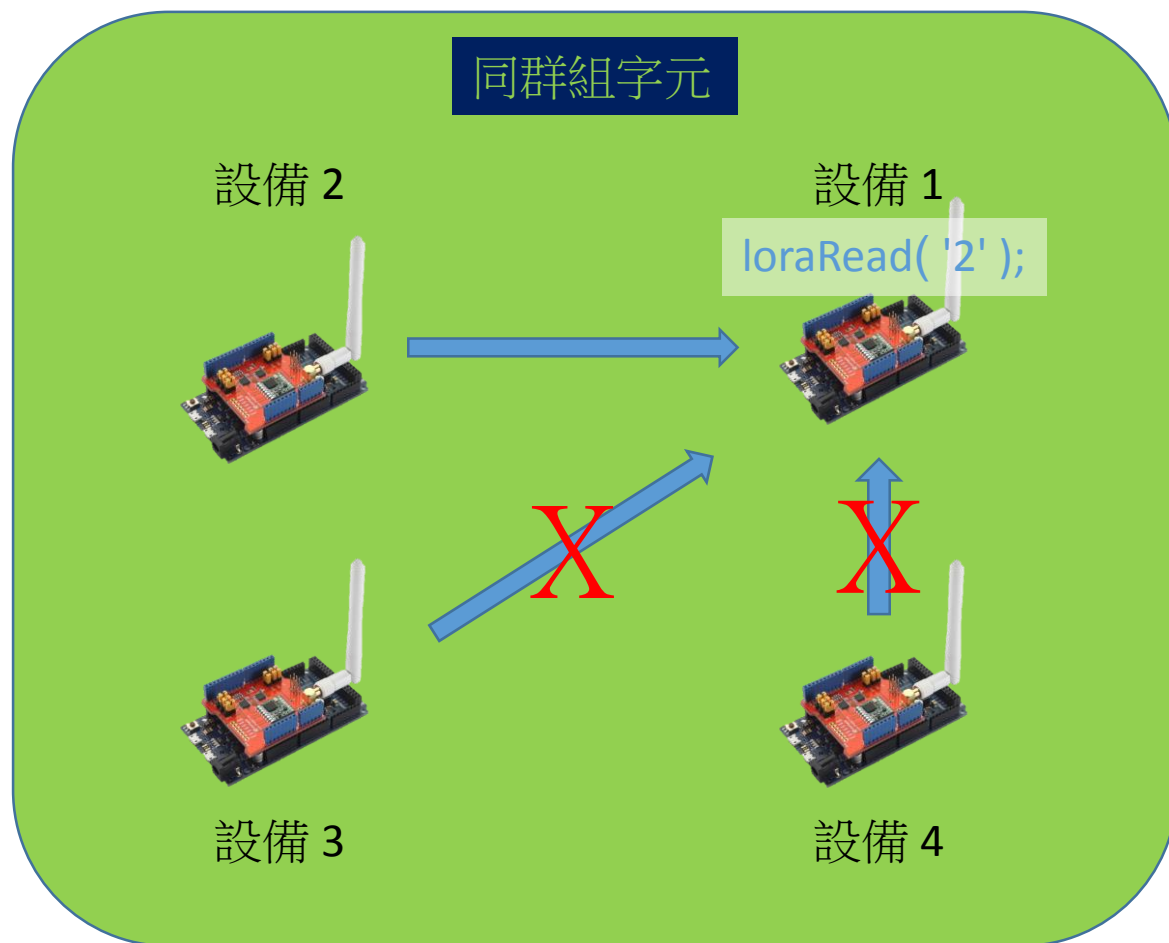
```
//群組字元不符回傳 Format does not match 字串
```

```
//設備字元不符回傳 ID does not match 字串
```

發送時同群組才有可能接收到訊息：



接收時可選擇接收哪個設備的訊息：



Lora訊息回傳

- 首先將Lora擴展板裝上天線之後，安裝到Arduino MEGA上。注意針腳對齊後再按壓，避免針腳損壞，如下圖為接收端：



接收端

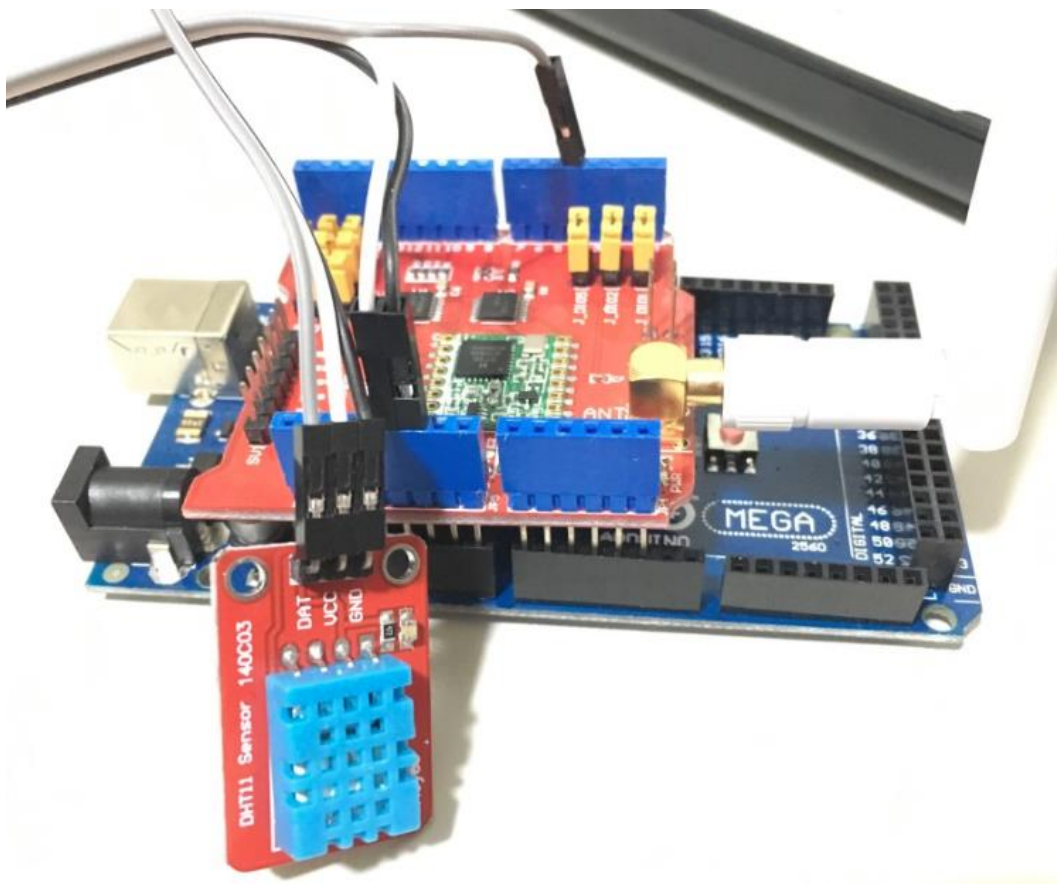
- 準備另一塊lora擴展板裝上天線，安裝到Arduino MEGA。
- 將溫濕度模組接上杜邦線連接擴展版腳位，作為感測端：

溫濕度 擴展板

DAT -> pin4

VCC -> 5V

GND -> GND



感測端

- 接收端程式碼：

```
1 //Lora 函式庫
2 #include <SPI.h>
3 #include <RH_RF95.h>
4
5 //建立Lora
6 RH_RF95 rf95;
7
8 //通訊設定
9 char head = '@'; // 群組代表字元
10 char ID = '1';    // 設備代表字元
11
12 void setup() {
13     Serial.begin(9600); // 開啟序列埠
14     while (!Serial);    // 等待序列埠開啟完成
15     //Lora初始化
16     if (!rf95.init())    // lora模組初始化並回傳是否成功
17         Serial.println("init failed"); // 初始化失敗回傳訊息
18 }
19
20 void loop() {
21     // 接收訊息
22     String s = loraRead('2'); // 宣告變數儲存代表字元為'2'的Lora設備發送的訊息
23     Serial.println(s);        // 輸出顯示變數
24 }
```

```

26 //Lora接收
27 String loraRead(char id){ // 將設備代表字元儲存到 id 代表變數
28     uint8_t buf[RH_RF95_MAX_MESSAGE_LEN]; // 宣告接收訊息變數
29     uint8_t len = sizeof(buf); // 宣告變數儲存接收訊息變數長度
30     if (rf95.waitForAvailableTimeout(3000)){ // 設定等待訊息時間3秒，沒有訊息回傳false
31         if (rf95.recv(buf, &len)){ // 接收Lora訊息，訊息接收失敗回傳false
32             Serial.print("//got reply: ");
33             Serial.println((char*)buf); // 顯示接收到的訊息
34             String s = String((char*)buf); // 將訊息轉換為字串
35             if(s[0]==head){ // 檢查群組代表字元是否符合格式
36                 if(s[1]==id){ // 檢查設備代表是否與代表變數相同
37                     s.remove(0,2); // 移除群組及設備代表字元
38                     return s; // 回傳
39                 }else{
40                     Serial.println("//ID does not match"); // 設備代表字元不符
41                     return "ID does not match";
42                 }
43             }else{
44                 Serial.println("//Format does not match"); // 格式不符
45                 return "Format does not match";
46             }

```

Lora訊息回傳

```
47  }else{  
48      Serial.println("//recv failed");           // 接收訊息失敗  
49      return "failed";  
50  }  
51  }else{  
52      Serial.println("//No reply, is rf95_server running?"); //沒有訊息  
53      return "No reply";  
54  }  
55 }
```

- 感測端程式碼：

```
1 // 函式庫
2 #include <SPI.h>
3 #include <RH_RF95.h>
4 #include "dht.h"
5
6 // 溫濕度模組設定
7 #define DHTPIN 4 // 連接4腳位
8 dht DHT;        // 建立DHT實體
9
10 //建立Lora實體
11 RH_RF95 rf95;
12
13 //通訊設定
14 char head = '@'; // 群組代表字元
15 char ID = '2';   // 設備代表字元
16
17 void setup() {
18     //序列埠
19     Serial.begin(9600); // 開啟序列埠
20     while (!Serial);    // 等待序列埠開啟完成
21     //Lora初始化
22     if (!rf95.init())    // lora模組初始化並回傳是否成功
23         Serial.println("init failed"); // 初始化失敗回傳訊息
24 }
```

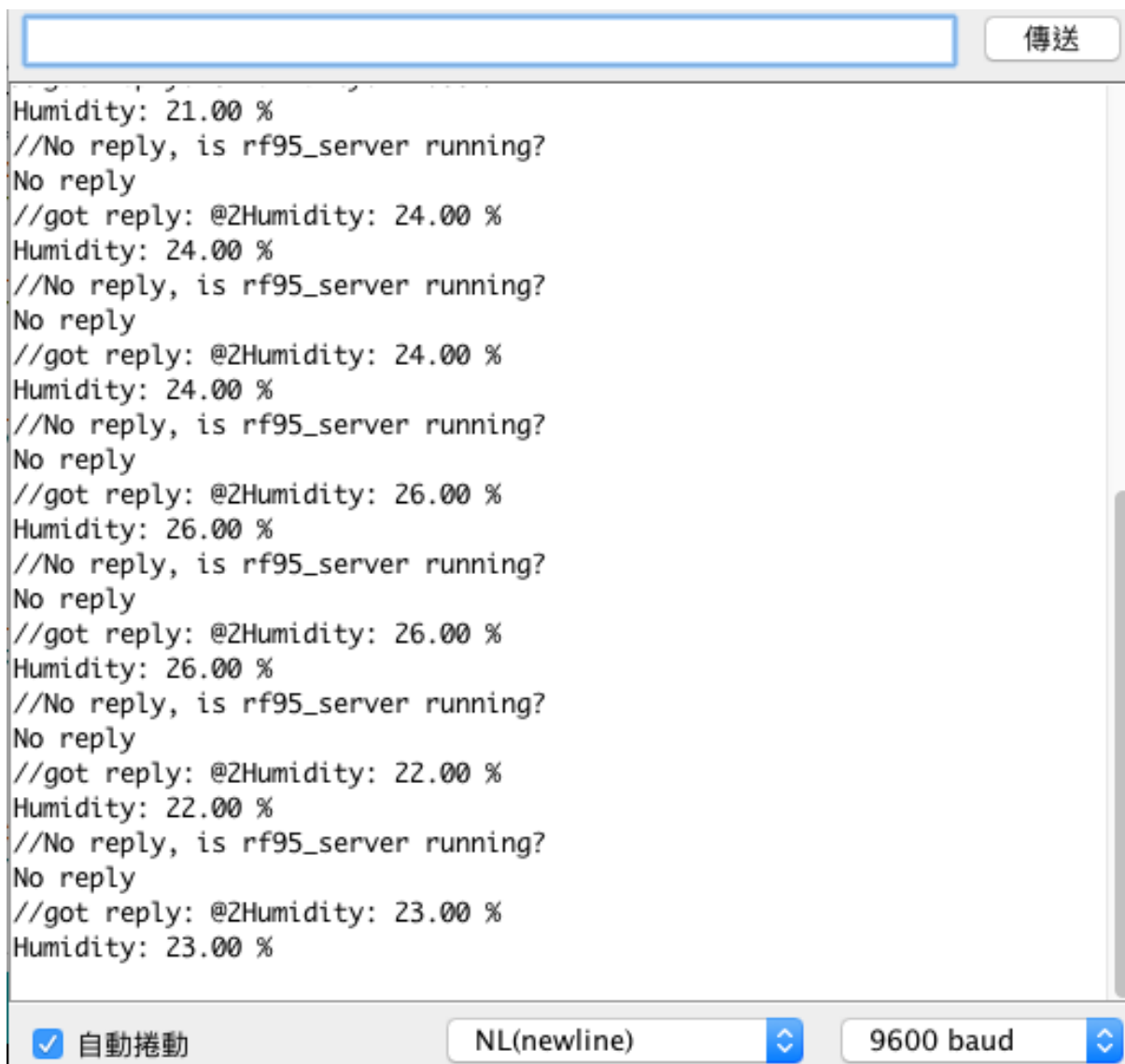


```

26 void loop(){
27     //回傳感測訊息
28     Serial.println("Send:");
29     DHT.read11(DHTPIN);           // 讀取溫濕度
30     String a = "temp: ";          // 建立字串加入溫濕度
31     a += DHT.temperature;
32     a += " C,Humi: ";
33     a += DHT.humidity;
34     a += " %";
35     loraSend(a);                  // 將訊息透過Lora發送
36     delay(400);                  // 延遲0.4秒
37 }
38
39 //Lora發送
40 void loraSend(String d){          // 將要發送的訊息儲存到 d 發送變數
41     d = String(head) + String(ID) + d; // 在發送變數前面加上群組及設備代表字元
42     int dl = d.length();          // 宣告變數存放發送長度
43     uint8_t data[dl+1];           // 宣告 uint8_8 格式變數
44     for(int i = 0; i<dl; i++){     // 將發送變數儲存到 uint8_t 變數
45         data[i]=d[i];
46     }
47     data[dl] = 0;                 // 結束字元為0
48     rf95.send(data, sizeof(data)); // 發送Lora訊息
49     rf95.waitPacketSent();        // 等待發送
50 }

```

- 回傳結果：

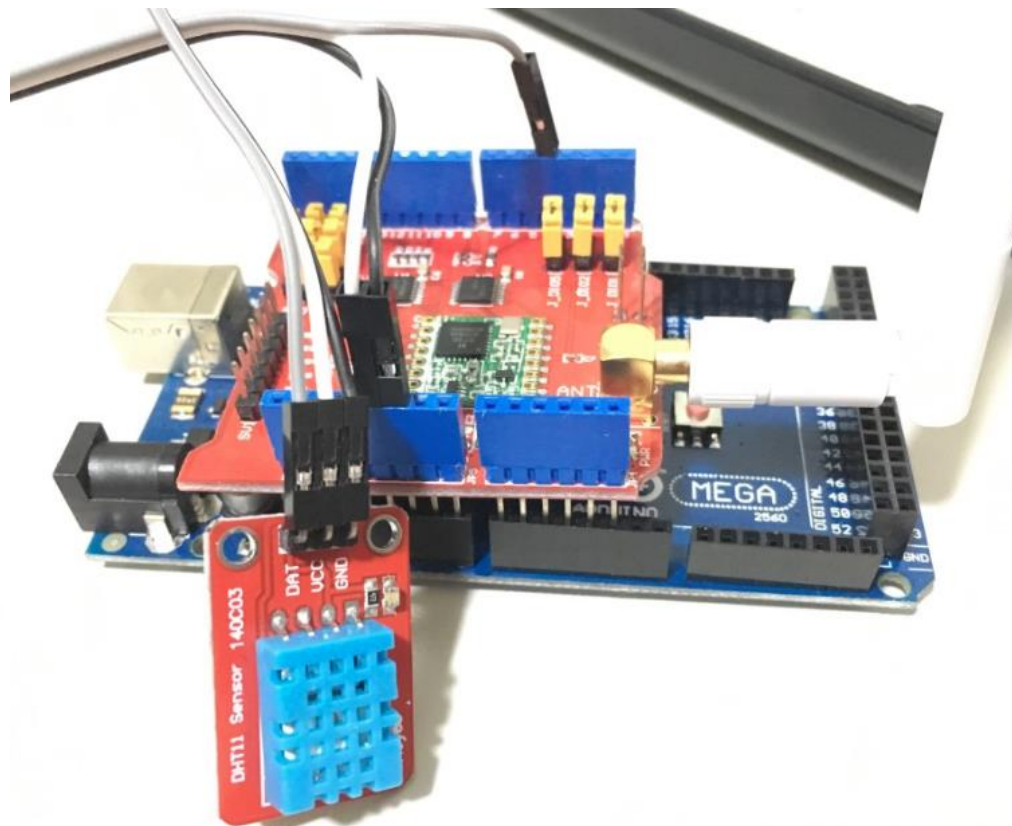


```
Humidity: 21.00 %  
//No reply, is rf95_server running?  
No reply  
//got reply: @2Humidity: 24.00 %  
Humidity: 24.00 %  
//No reply, is rf95_server running?  
No reply  
//got reply: @2Humidity: 24.00 %  
Humidity: 24.00 %  
//No reply, is rf95_server running?  
No reply  
//got reply: @2Humidity: 26.00 %  
Humidity: 26.00 %  
//No reply, is rf95_server running?  
No reply  
//got reply: @2Humidity: 26.00 %  
Humidity: 26.00 %  
//No reply, is rf95_server running?  
No reply  
//got reply: @2Humidity: 22.00 %  
Humidity: 22.00 %  
//No reply, is rf95_server running?  
No reply  
//got reply: @2Humidity: 23.00 %  
Humidity: 23.00 %
```

☒ 自動捲動 NL(newline) 9600 baud

Lora命令發送

- 硬體沿用上個範例，使用內建的LED作為顯示。
- 使用接收端作為命令發送，感測端作為LED顯示。



- 接收端增加命令發送程式碼：

```
1 //Lora 函式庫
2 #include <SPI.h>
3 #include <RH_RF95.h>
4
5 //建立Lora
6 RH_RF95 rf95;
7
8 //通訊設定
9 char head = '@'; // 群組代表字元
10 char ID = '1'; // 設備代表字元
11
12 //命令發送
13 String command = ""; // 宣告暫存變數
14
15 void setup() {
16     Serial.begin(9600); // 開啟序列埠
17     while (!Serial); // 等待序列埠開啟完成
18     //Lora初始化
19     if (!rf95.init()) // lora模組初始化並回傳是否成功
20         Serial.println("init failed"); // 初始化失敗回傳訊息
21 }
22
```

Lora命令發送

```
23 void loop() {  
24     // 接收訊息  
25     String s = loraRead('2'); // 宣告變數儲存代表字元為'2'的Lora設備發送的訊息  
26     Serial.println(s);       // 輸出顯示變數  
27     // 發送命令  
28     if(Serial.available()){ // 如果序列埠有資料  
29         char c = Serial.read(); // 宣告變數儲存字元  
30         if(c=='\n'){           // 如果為換行字元  
31             loraSend(command); // 透過Lora發送命令訊息  
32             while(s != "OK"){ // 重複執行直到接收到OK  
33                 loraSend(command); // 透過Lora發送命令訊息  
34                 s = loraRead('2'); // 接收設備'2'的回傳訊息  
35             }  
36             command = ""; // 清空命令暫存變數  
37         }else{  
38             command+=String(c); // 將字元加到命令暫存變數  
39         }  
40     }  
41 }
```

```
43 //Lora發送
44 void loraSend(String d){ // 將要發送的訊息儲存到 d 發送變數
45     d = String(head) + String(ID) + d; // 在發送變數前面加上群組及設備代表字元
46     int dl = d.length(); // 宣告變數存放發送長度
47     uint8_t data[dl+1]; // 宣告 uint8_8 格式變數
48     for(int i = 0;i<dl;i++){ // 將發送變數儲存到 uint8_t 變數
49         data[i]=d[i];
50     }
51     data[dl] = 0; // 結束字元為0
52     rf95.send(data, sizeof(data)); // 發送Lora訊息
53     rf95.waitPacketSent(); // 等待發送
54 }
55
56 //Lora接收
57 String loraRead(char id){ // 將設備代表字元儲存到 id 代表變數
58     uint8_t buf[RH_RF95_MAX_MESSAGE_LEN]; // 宣告接收訊息變數
59     uint8_t len = sizeof(buf); // 宣告變數儲存接收訊息變數長度
60
61     if (rf95.waitForAvailable(3000)){ // 設定等待訊息時間3秒，沒有訊息回傳false
62         if (rf95.recv(buf, &len)){ // 接收Lora訊息，訊息接收失敗回傳false
63             Serial.print("//got reply: ");
64             Serial.println((char*)buf); // 顯示接收到的訊息
65             String s = String((char*)buf); // 將訊息轉換為字串
```



```
66  if(s[0]==head){ // 檢查群組代表字元是否符合格式
67      if(s[1]==id){ // 檢查設備代表是否與代表變數相同
68          s.remove(0,2); // 移除群組及設備代表字元
69          return s; // 回傳
70      }else{
71          Serial.println("//ID does not match"); // 設備代表字元不符
72          return "ID does not match";
73      }
74  }else{
75      Serial.println("//Format does not match"); // 格式不符
76      return "Format does not match";
77  }
78  }else{
79      Serial.println("//recv failed"); // 接收訊息失敗
80      return "failed";
81  }
82  }else{
83      Serial.println("//No reply, is rf95_server running?"); //沒有訊息
84      return "No reply";
85  }
86 }
```

- 感測端增加LED控制程式碼：

```
1 // 函式庫
2 #include <SPI.h>
3 #include <RH_RF95.h>
4 #include "dht.h"
5
6 // 溫濕度模組設定
7 #define DHTPIN 4 // 連接4腳位
8 dht DHT;        // 建立DHT實體
9
10 //建立Lora實體
11 RH_RF95 rf95;
12
13
14 //通訊設定
15 char head = '@'; // 群組代表字元
16 char ID = '2';   // 設備代表字元
17
18 //LED
19 int led = 13;    // 使用內建LED
20
```

```
21 void setup() {  
22     //LED  
23     pinMode(led, OUTPUT);  
24     digitalWrite(led, LOW);  
25     //序列埠  
26     Serial.begin(9600);    // 開啟序列埠  
27     while (!Serial);      // 等待序列埠開啟完成  
28     //Lora初始化  
29     if (!rf95.init())      // lora模組初始化並回傳是否成功  
30         Serial.println("init failed"); // 初始化失敗回傳訊息  
31 }  
32  
33 void loop(){  
34     //回傳感測訊息  
35     Serial.println("Send:");  
36     DHT.read11(DHTPIN);    // 讀取溫濕度  
37     String a = "temp: ";   // 建立字串加入溫濕度  
38     a += DHT.temperature;  
39     a += " C,Humi: ";  
40     a += DHT.humidity;  
41     a += " %";  
42     loraSend(a);           // 將訊息透過Lora發送
```

```
43  
44 //命令辨識  
45 String r = loraRead('1'); // 宣告變數接收儲存設備代號'1'的Lora訊息  
46 if(r!="No reply"&&r!="failed"&&r!="Format does not match"  
47   &&r!="ID does not match"){ // 如果成功街接收訊息  
48   Serial.println("receive:");  
49   Serial.println(r); // 顯示接收資料  
50   loraSend("OK"); // 回傳成功訊息  
51   if(r=="on"){ // 如果訊息為"on"  
52     digitalWrite(led,HIGH); // LED亮起  
53   }  
54   if(r=="off"){ // 如果訊息為"off"  
55     digitalWrite(led,LOW); // LED熄滅  
56   }  
57 }  
58 delay(400); // 延遲0.4秒  
59 }  
60
```

```

61 //Lora發送
62 void loraSend(String d){ // 將要發送的訊息儲存到 d 發送變數
63     d = String(head) + String(ID) + d; // 在發送變數前面加上群組及設備代表字元
64     int dl = d.length(); // 宣告變數存放發送長度
65     uint8_t data[dl+1]; // 宣告 uint8_8 格式變數
66     for(int i = 0; i<dl; i++){ // 將發送變數儲存到 uint8_t 變數
67         data[i]=d[i];
68     }
69     data[dl] = 0; // 結束字元為0
70     rf95.send(data, sizeof(data)); // 發送Lora訊息
71     rf95.waitPacketSent(); // 等待發送
72 }
73
74 //Lora接收
75 String loraRead(char id){ // 將設備代表字元儲存到 id 代表變數
76     uint8_t buf[RH_RF95_MAX_MESSAGE_LEN]; // 宣告接收訊息變數
77     uint8_t len = sizeof(buf); // 宣告變數儲存接收訊息變數長度
78
79     if (rf95.waitForAvailableTimeout(3000)){ // 設定等待訊息時間3秒，沒有訊息回傳false
80         if (rf95.recv(buf, &len)){ // 接收Lora訊息，訊息接收失敗回傳false
81             Serial.print("//got reply: ");
82             Serial.println((char*)buf); // 顯示接收到的訊息
83             String s = String((char*)buf); // 將訊息轉換為字串

```

```
84   if(s[0]==head){                                // 檢查群組代表字元是否符合格式
85       if(s[1]==id){                                // 檢查設備代表是否與代表變數相同
86           s.remove(0,2);                            // 移除群組及設備代表字元
87           return s;                                // 回傳
88       }else{
89           Serial.println("//ID does not match");    // 設備代表字元不符
90           return "ID does not match";
91       }
92   }else{
93       Serial.println("//Format does not match");    // 格式不符
94       return "Format does not match";
95   }
96   }else{
97       Serial.println("//recv failed");              // 接收訊息失敗
98       return "failed";
99   }
100  }else{
101     Serial.println("//No reply, is rf95_server running?"); //沒有訊息
102     return "No reply";
103 }
104 }
```


LED控制：

1. 開啟監控視窗選擇NL做為命令的結尾
2. 再輸入on或off控制LED

沒有行結尾

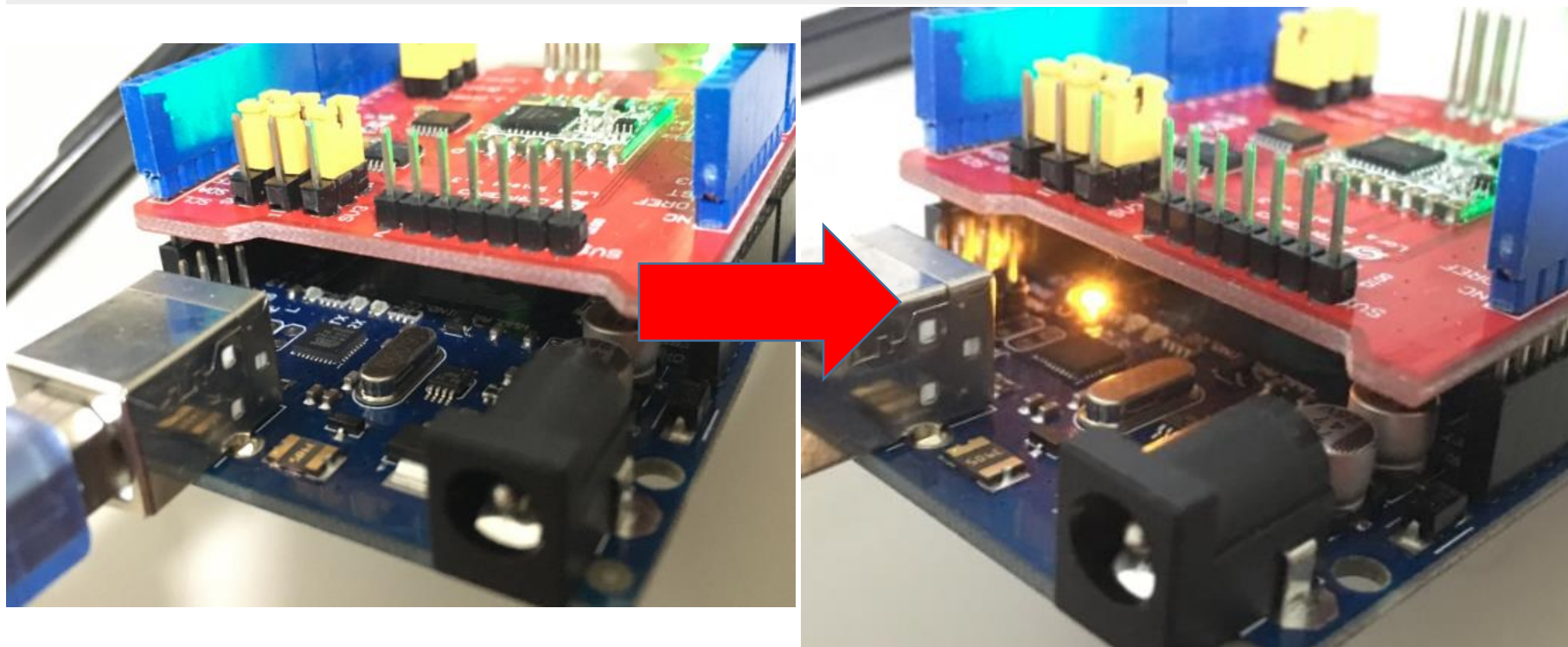
✓ NL(newline)

CR(carriage return)

NL & CR

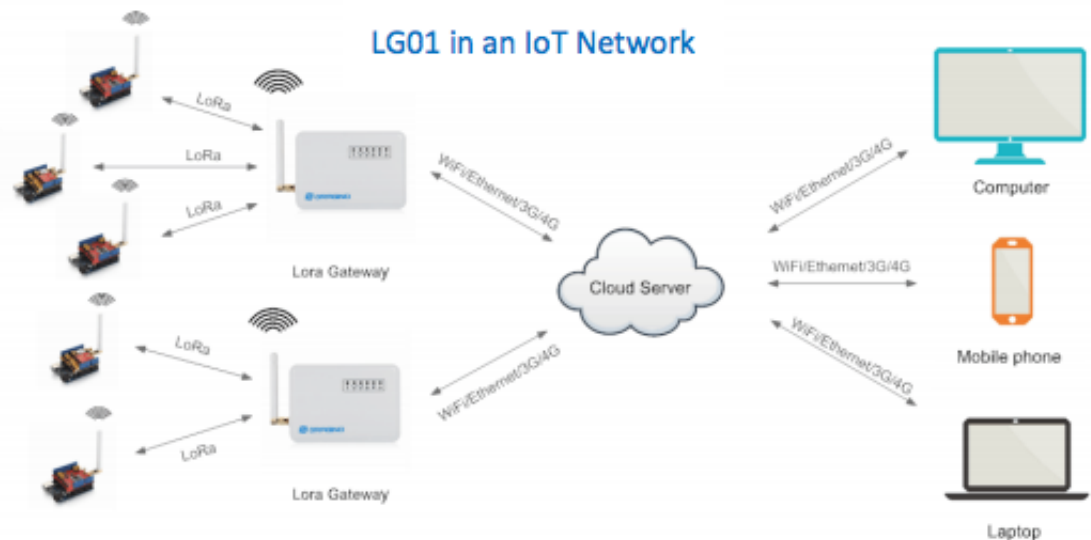
on

傳送

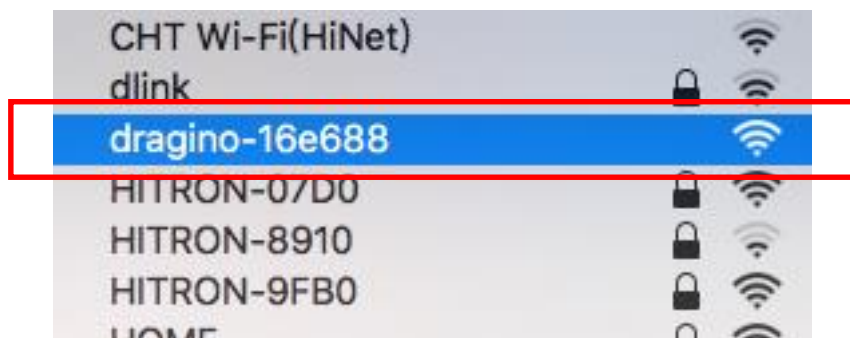


Lora gateway

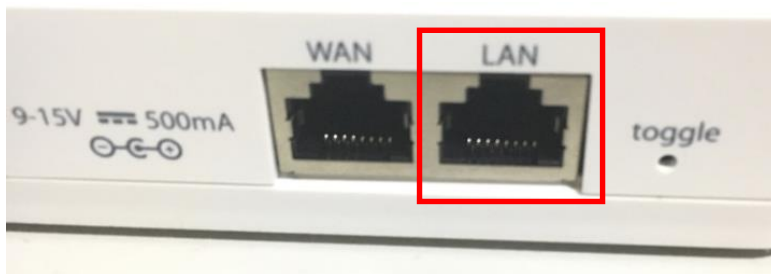
閘道器（英語：Gateway），大陸譯作「網關」、臺灣及香港譯作「閘道器」，區別於路由器（由於歷史的原因，許多有關TCP/IP的文獻曾經把網路層使用的路由器（英語：Router）稱為閘道器，在今天很多區域網路採用都是路由來接入網路，因此現在通常指的閘道器就是路由器的IP），經常在家庭中或者小型企業網路中使用，用於連線區域網路和Internet。閘道器也經常指把一種協定轉成另一種協定的裝置，比如語音閘道器，在這裡可以作為網路及Lora的通道。



Lora gateway 設定：
wifi請連接 dragino-xxxxxx



或是使用網路線連接 gateway 的 LAN



在瀏覽器輸入 10.130.1.1

Username: **root** password: **dragino**

dragino-16e688

Authorization Required

Please enter your username and password.

Username

root

Password

.....|



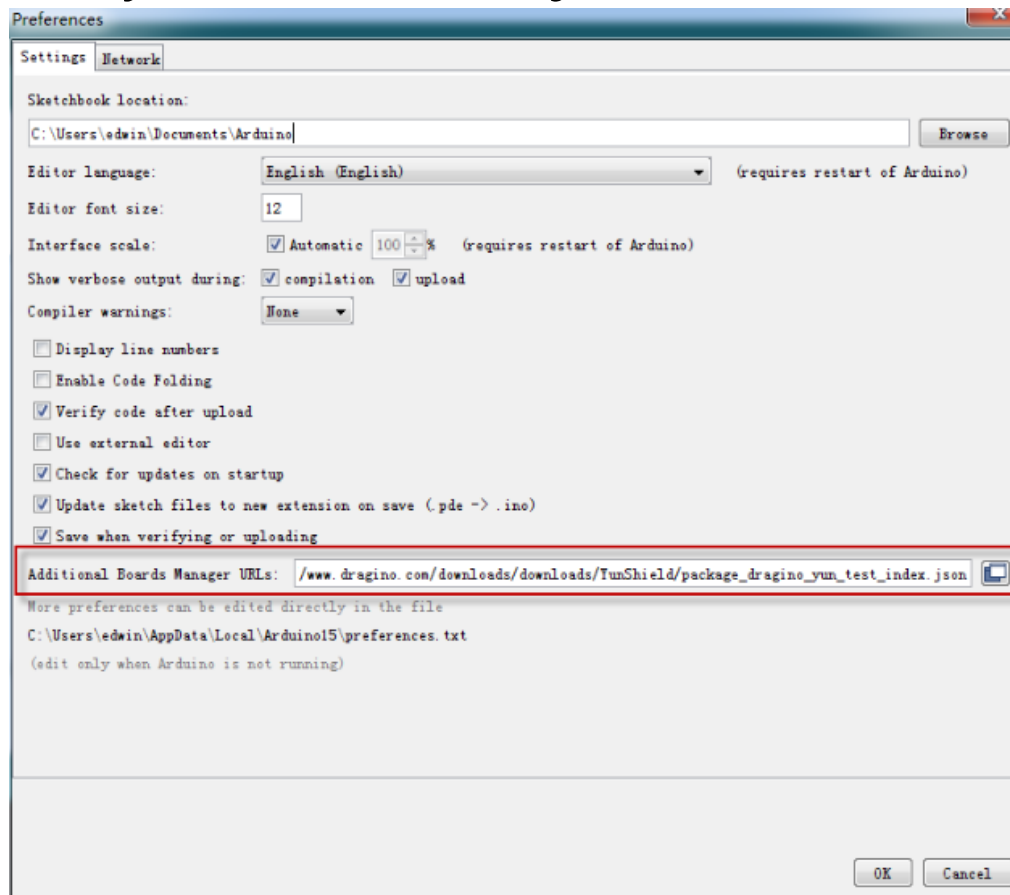
Login



Reset

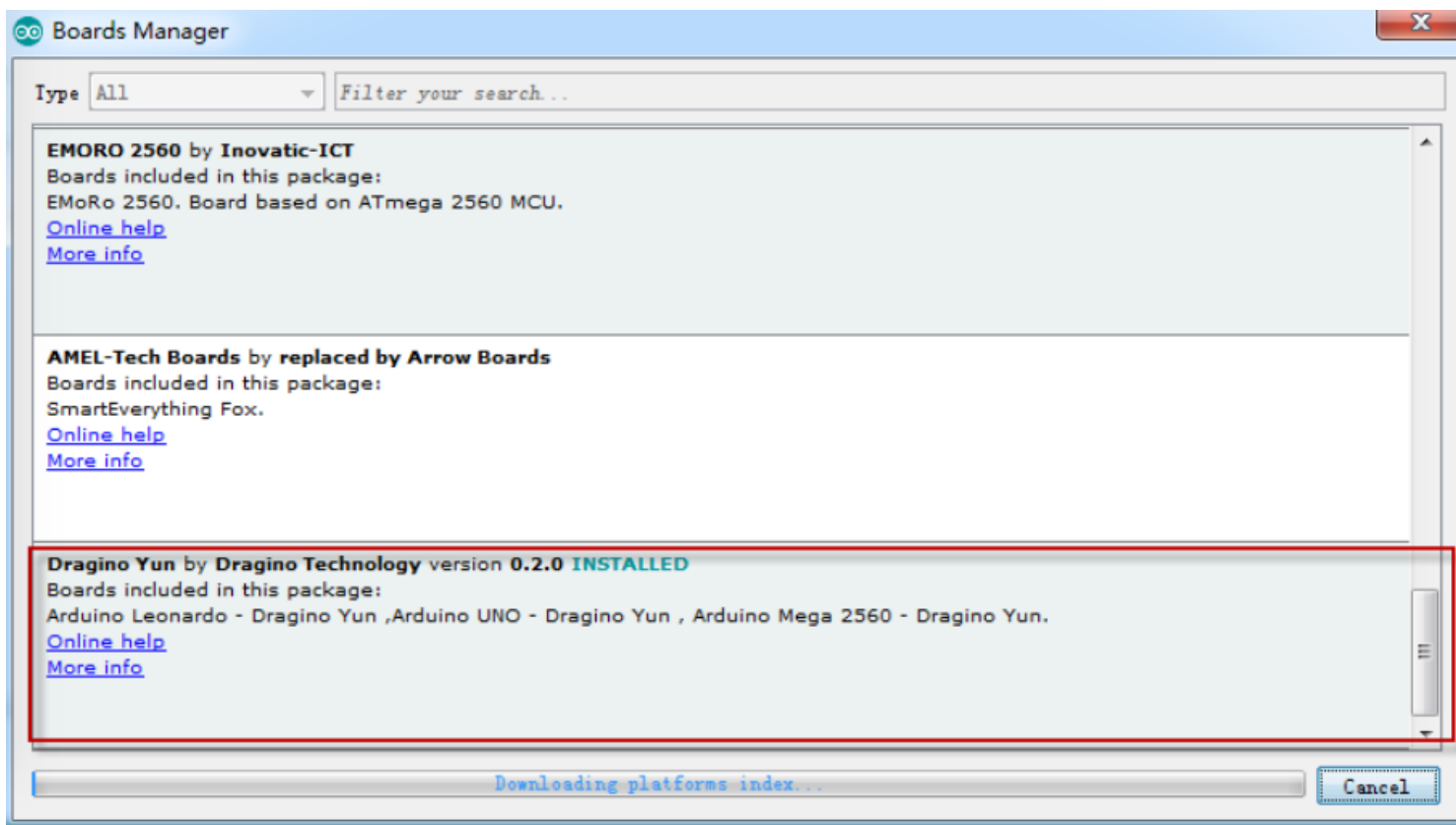
DRAGINO TECHNOLOGY CO., LIMITED

接下來要開啟 Arduino IDE，點選 檔案 -> Preference 填入網址
http://www.dragino.com/downloads/downloads/YunShield/package_dragino_yun_test_index.json

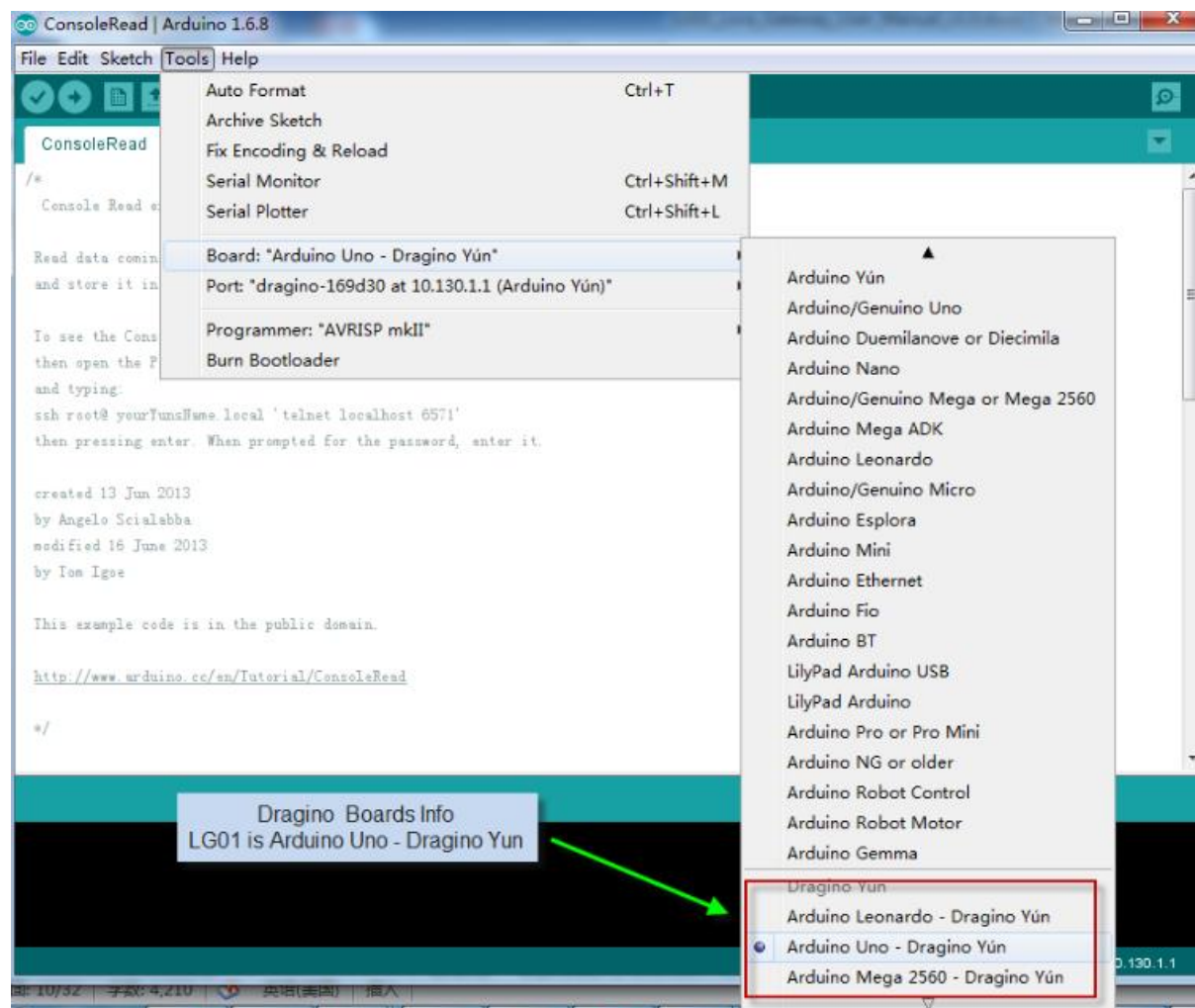


Lora gateway

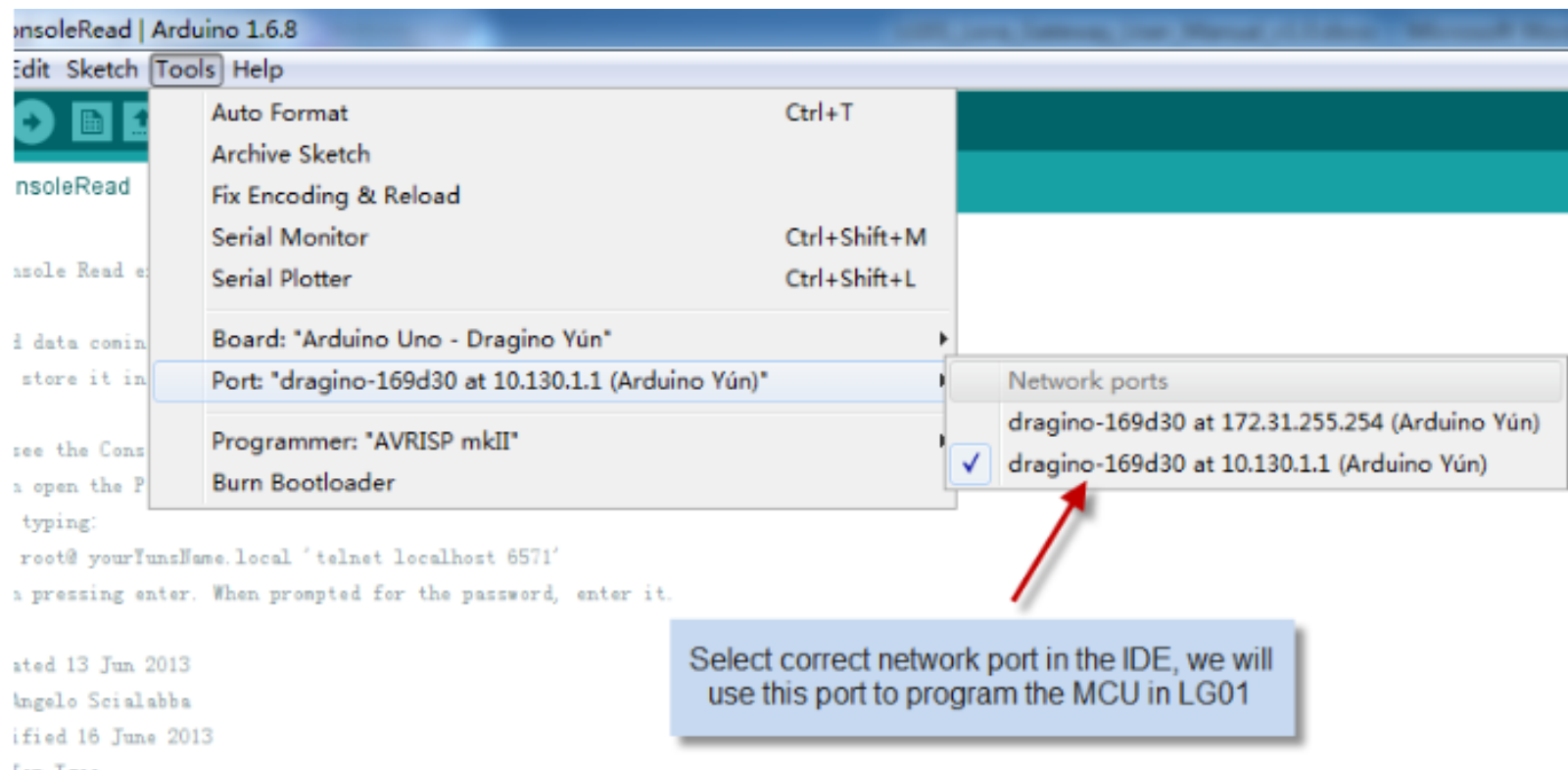
開啟 工具 -> 板子 -> 板子管理
搜尋並安裝 Dragino Yun



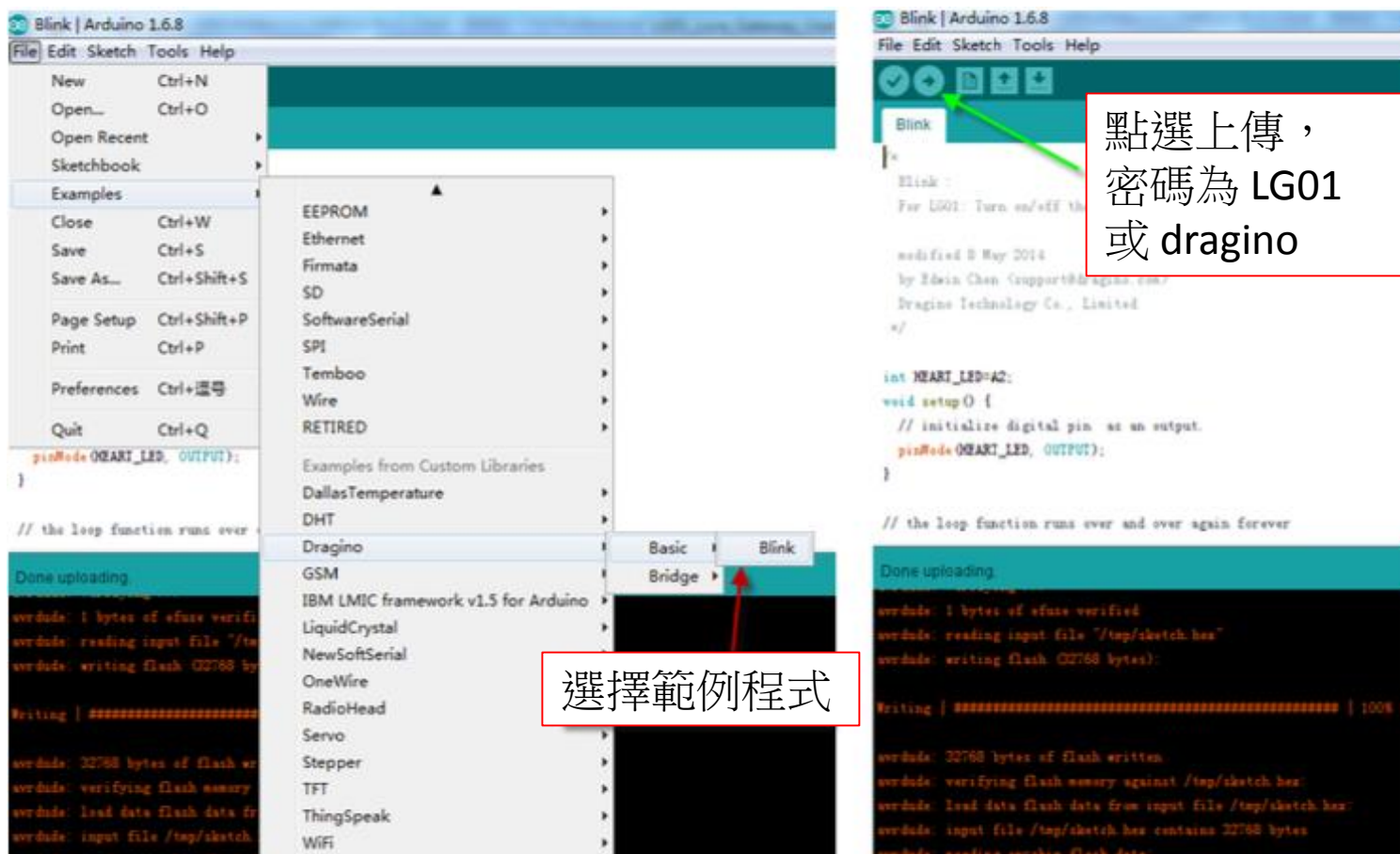
開啟 工具 -> 板子 選擇 Arduino Uno - Dragino Yun



開啟 工具 - > 序列埠 選擇 10.130.1.1(Arduino Yun)



開啟 檔案 -> 範例 -> Dragino -> Basic -> Blink ，
點選上傳燒錄，gateway上的愛心燈號會開始閃爍。



接來介紹 gateway 與擴充板的通訊，首先下載程式庫：

<http://www.airspayce.com/mikem/arduino/RadioHead/RadioHead-1.63.zip>

下載後透過 IDE 的 草稿碼->匯入程式庫->加入.ZIP程式庫，選擇下載的檔案



硬體方面可以是 Arduino+擴充板 與 gateway：

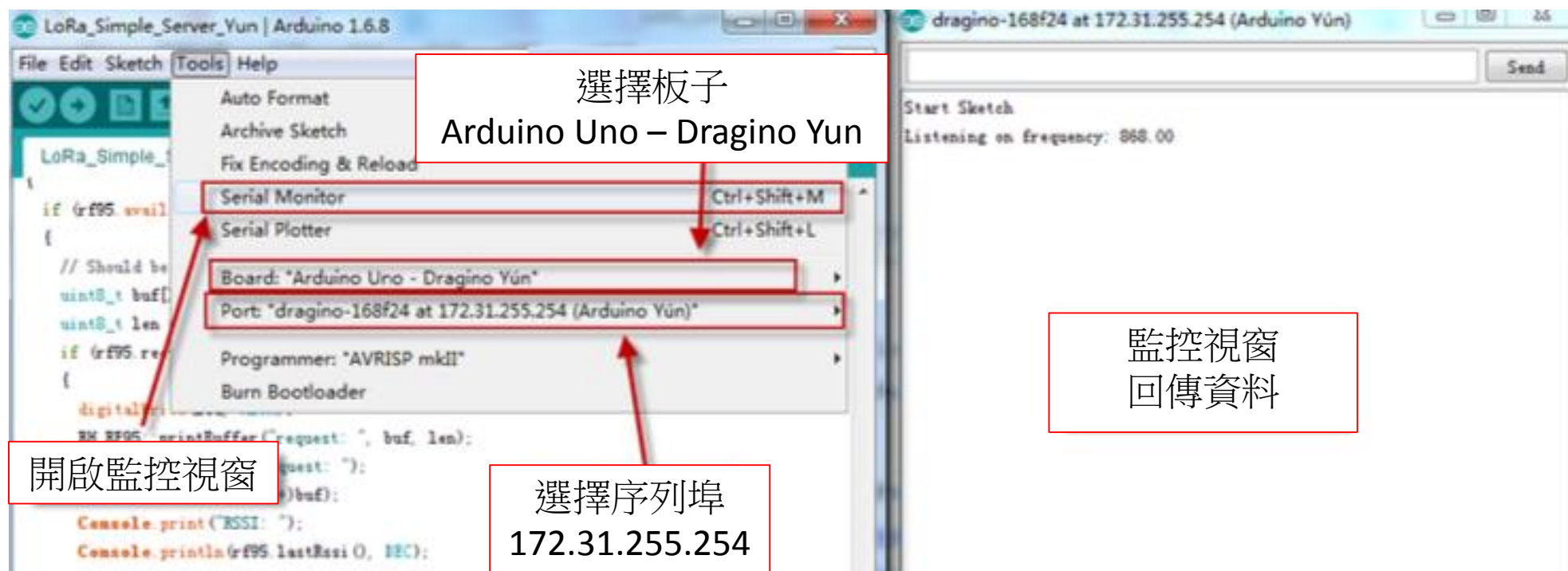


或是兩個gateway：

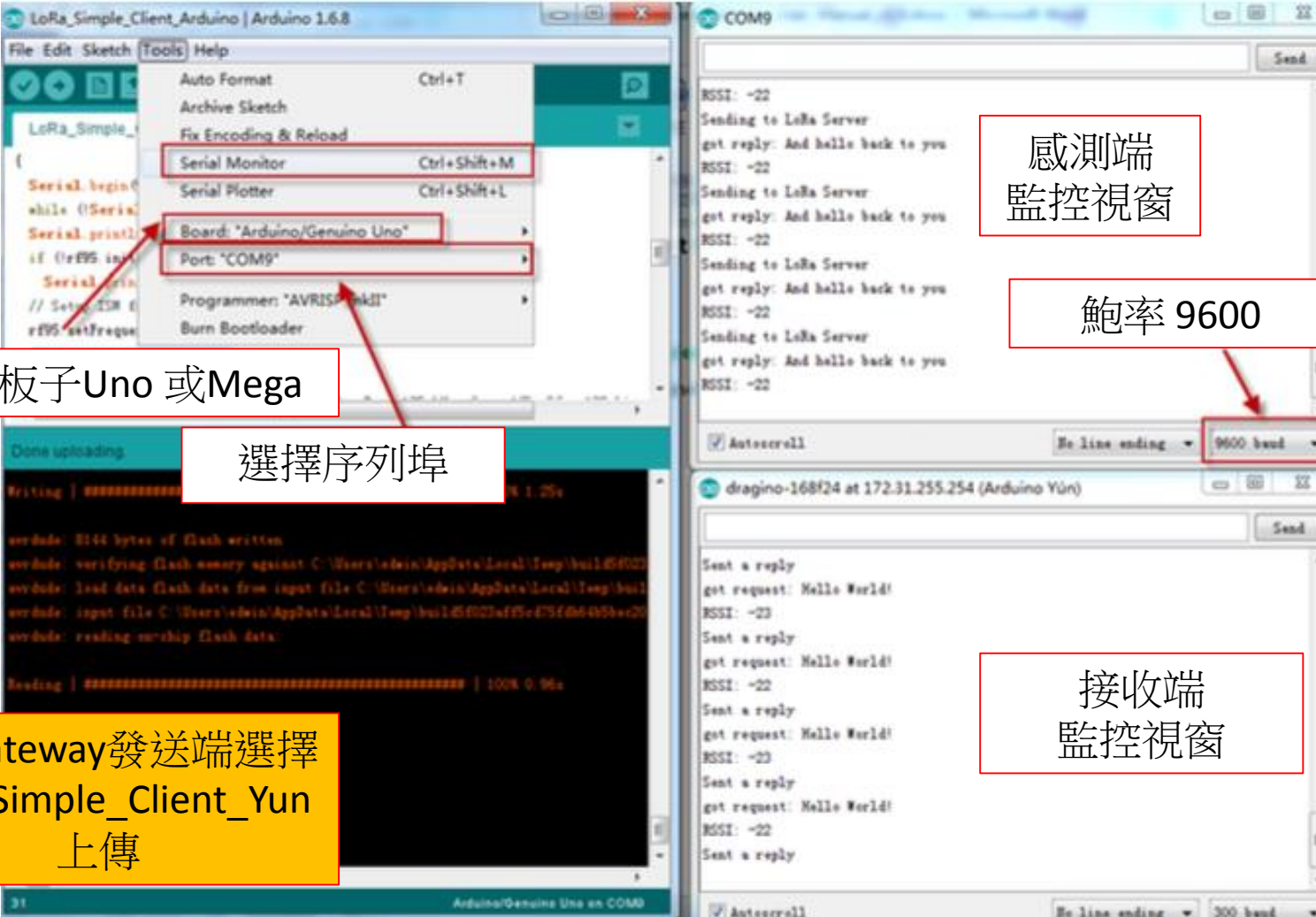


上傳 gateway 接收端程式碼：

檔案 -> 範例 -> Dragino -> LoRa -> LoRa_Simple_Server_Yun



Arduino+擴充板接收端選擇 LoRa_Simple_Client_Arduino



選擇板子Uno 或Mega

選擇序列埠

感測端
監控視窗

鮑率 9600

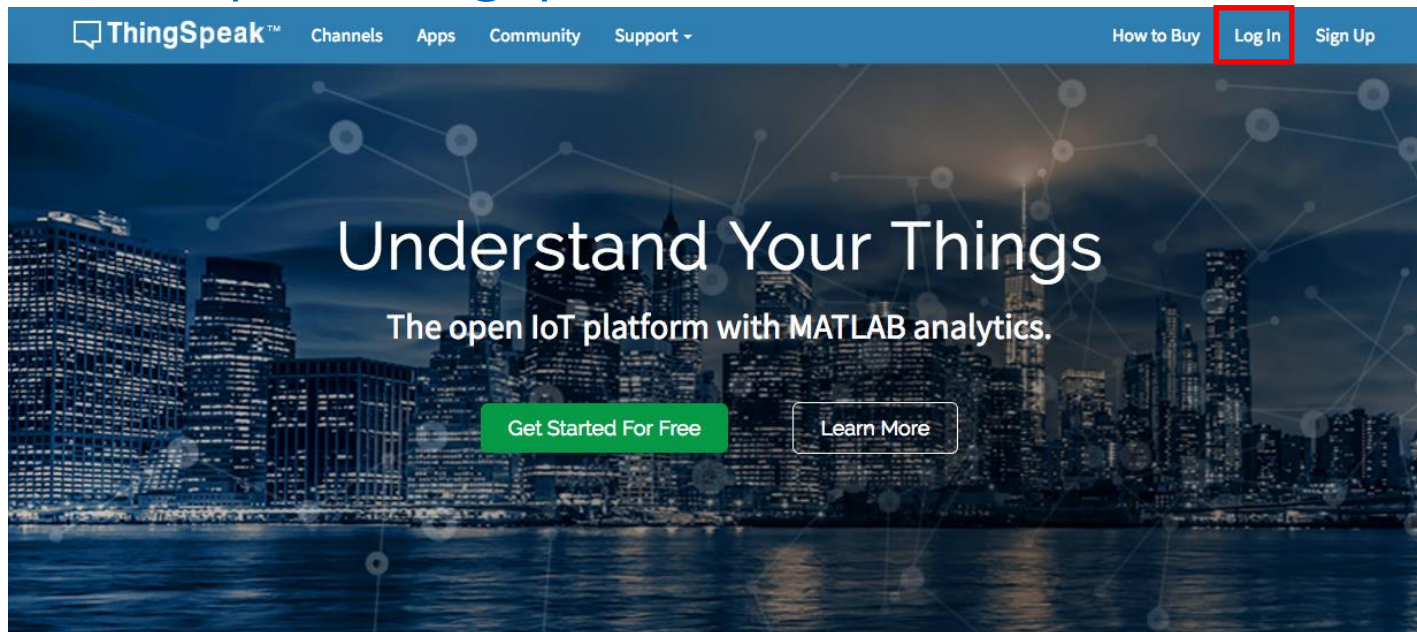
接收端
監控視窗

如為gateway發送端選擇
LoRa_Simple_Client_Yun
上傳

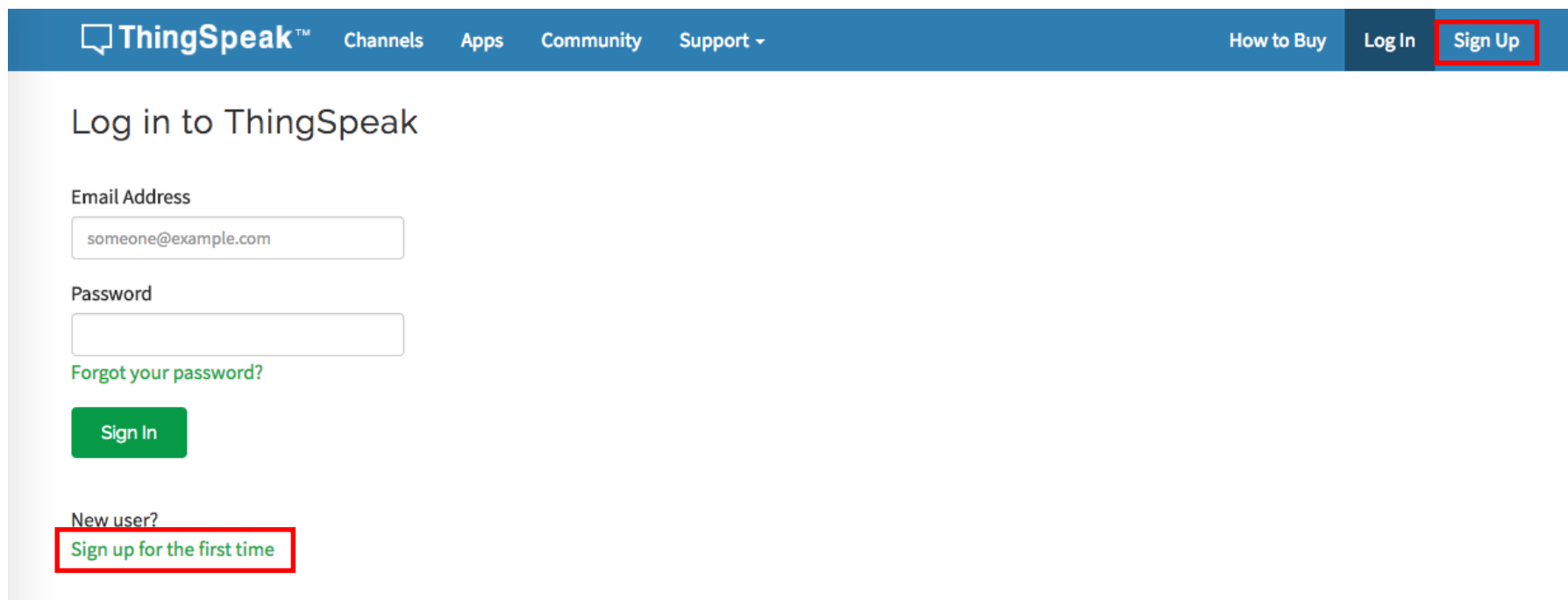
雲端資料分析

- **ThingSpeak**是“物聯網（IoT）的雲端平台，可以讓您在雲端收集和存儲感測器數據並開發IoT應用程序”，是由Mathworks（MATLAB®的創建者）研發。IoT設備可以將數據上傳到ThingSpeak（例如Arduino及Raspberry Pi等相關設備）。收集的數據可以在雲端分析。

在網頁輸入<https://thingspeak.com/>開啟網頁按下右上 **Login**



- 如果已經有ThingSpeak帳號，直接輸入帳號密碼，新使用者請點選下方 **Sign up for the first time** 或是右上的 **Sign Up**



The screenshot shows the ThingSpeak website's login and sign-up interface. At the top, a blue navigation bar contains the ThingSpeak logo and links for Channels, Apps, Community, Support, How to Buy, Log In, and Sign Up. The Sign Up link is highlighted with a red box. Below the navigation bar, the main content area is titled 'Log in to ThingSpeak'. It features two input fields: 'Email Address' (containing 'someone@example.com') and 'Password'. A green link 'Forgot your password?' is positioned below the password field. A green 'Sign In' button is located below the email field. At the bottom, under the heading 'New user?', the link 'Sign up for the first time' is highlighted with a red box.

在登入頁面輸入資料

ThingSpeak™

Channels

Apps

Community

Support ▾

How to Buy

Log In

Sign Up

Sign up for ThingSpeak

The ThingSpeak service is operated by MathWorks. In order to sign up for ThingSpeak, you must create a new MathWorks Account or log in to your MathWorks Account.

Create MathWorks Account

Email Address

電子信箱

User ID

自訂ID

?

Password

密碼

👁

Taiwan

地區

▾

First Name

名字

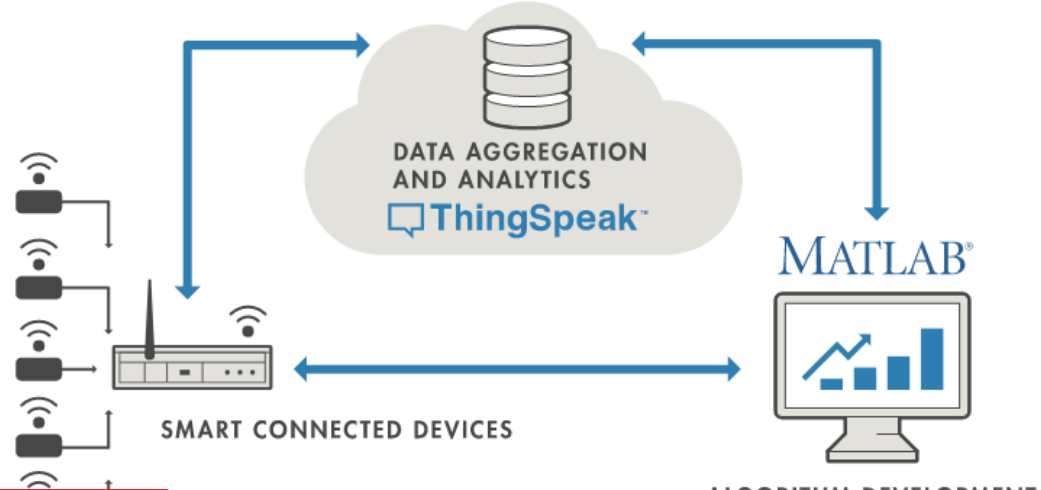
Last Name

姓氏

By clicking continue, you agree to our [privacy policy](#)

填寫完成往下拉按下

Continue



到信箱收信認證

Thank you for registering with MathWorks!

To complete the registration process, verify your email address by clicking this link:

Verify your email

Verify Your MathWorks Account

To finish creating your account, complete the following steps:

1. Go to your inbox for **ibuildertw003@gmail.com**.
2. Click the link in the email we sent you.

Once you've done this, click Continue.

Didn't get the email?

1. Check your spam folder.
2. [Send me the email again](#).
3. Contact [Customer Support](#) if you still do not have the email

Sincerely,
MathWorks Customer Service Team

[Privacy policy](#)

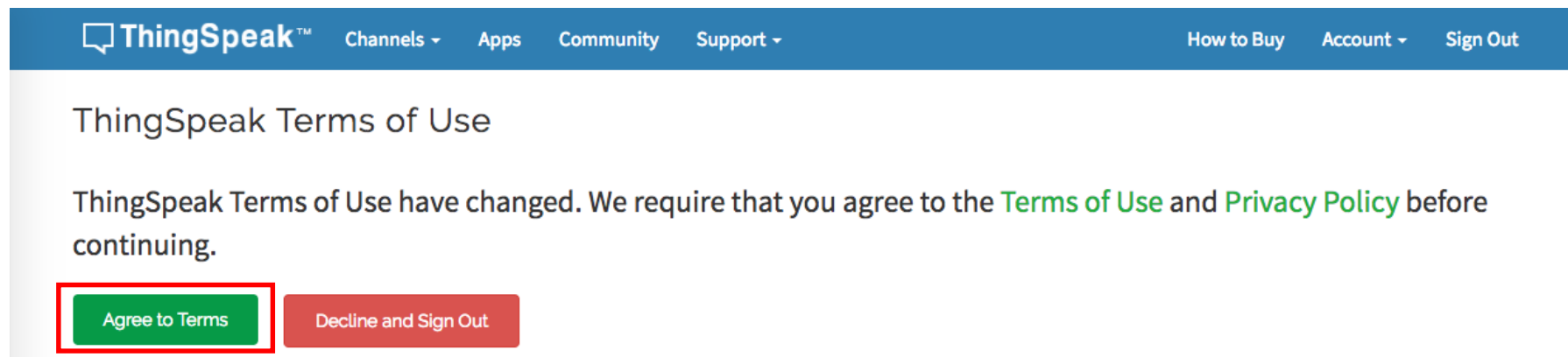
回到網頁按下

Continue

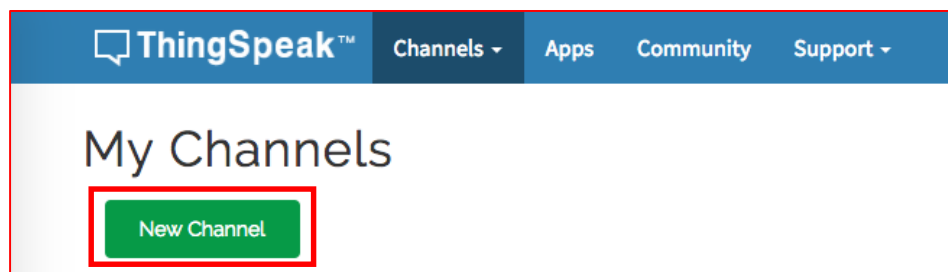
再按

OK

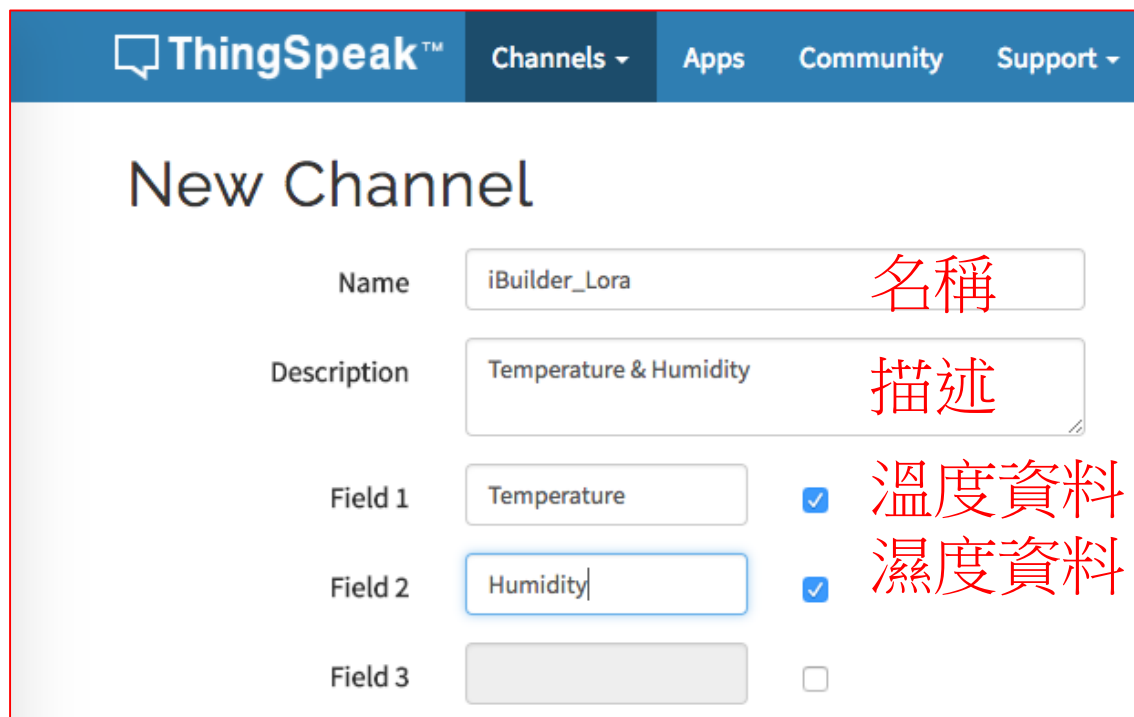
使用前要同意使用條款



會跳轉到 **My Channels** 按下 **New Channle**



輸入資料建立新通道



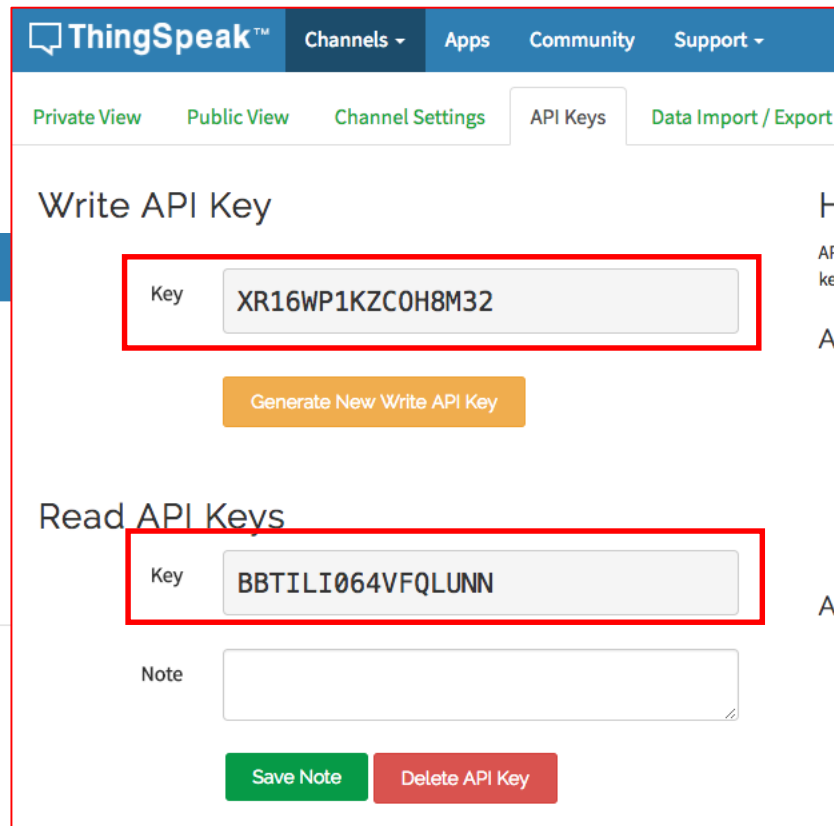
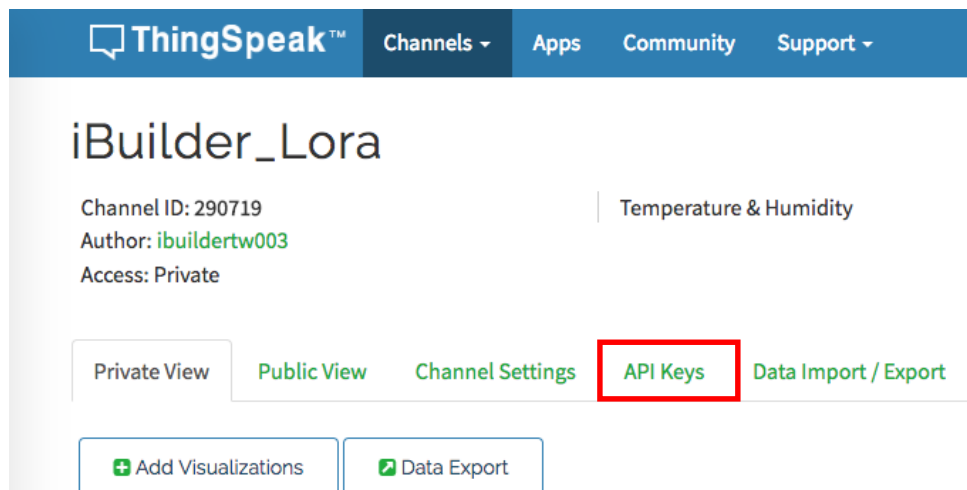
The screenshot shows the 'New Channel' form in the ThingSpeak web interface. The form includes a header with the ThingSpeak logo and navigation links (Channels, Apps, Community, Support). The main form area has the title 'New Channel' and several input fields: 'Name' (containing 'iBuilder_Lora'), 'Description' (containing 'Temperature & Humidity'), and three 'Field' inputs (Field 1: 'Temperature', Field 2: 'Humidity', Field 3: empty). To the right of each field input is a checkbox. Red Chinese text annotations are overlaid on the form: '名稱' (Name) next to the Name field, '描述' (Description) next to the Description field, and '溫度資料' (Temperature data) and '濕度資料' (Humidity data) next to the checked checkboxes for Field 1 and Field 2 respectively. A large red bracket on the right side of the form groups the checkboxes and is labeled '(需打勾)' (Need to check).

Field	Input	Checkbox	Annotation
Name	iBuilder_Lora		名稱
Description	Temperature & Humidity		描述
Field 1	Temperature	<input checked="" type="checkbox"/>	溫度資料
Field 2	Humidity	<input checked="" type="checkbox"/>	濕度資料
Field 3		<input type="checkbox"/>	

往下拉按下

Save Channel

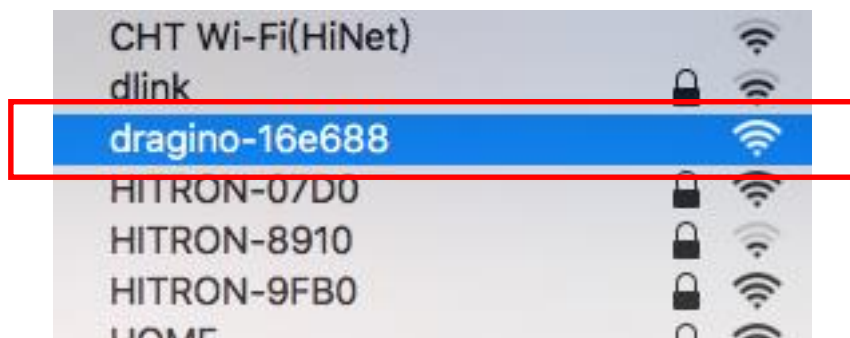
點選API Keys：
會顯示寫入及讀取的金鑰(Key)，
請紀錄下來，稍後程式碼需要金鑰



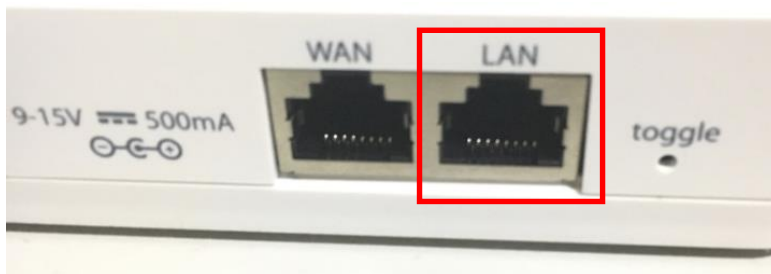
硬體設備：gateway 接收端 及 Arduino+擴充板 感測端



接下來要對 Lora gateway 進行設定：
wifi請連接 dragino-xxxxxx



或是使用網路線連接 gateway 的 LAN



在瀏覽器輸入 10.130.1.1

Username: **root** password: **dragino**

dragino-16e688

Authorization Required

Please enter your username and password.

Username

root

Password

.....|



Login



Reset

DRAGINO TECHNOLOGY CO., LIMITED

下載 ThingSpeak 的程式庫：

<https://github.com/mathworks/thingspeak-arduino>

下載後透過 IDE 的 草稿碼->匯入程式庫->加入.ZIP程式庫，選擇下載的檔案



Arduino+擴充板感測端程式碼：

檔案 --> 範例 --> Dragino --> IoTServer --> ThingSpeak --> dht11_client

上傳到Arduino板子

gateway接收端程式碼：

檔案 --> 範例 --> Dragino --> IoTServer --> ThingSpeak --> dht11_dht11_server

修改 myChannelNumber 及 myWriteAPIKey 的資訊

上傳到 gateway

開啟Thingspeak頁面顯示資料圖表：

