實驗三

B1121141 葉彥辰

1解釋rf95.recv() 與 rf95.send()的用法及特性

Rf95.recv(buf, &len) 的功能為嘗試接收LoRa資料，並儲存的buf中，buf長度為len。特性為通常需要搭配 rf95.available()或 rf95.waitAvailableTimeout()來檢查是否有可讀取資料。若成功則回傳true，.否則回傳false。我們在實作中，可以將接收到的內容轉成字串 (char\*)buf 來顯示。

Rf95.send(data, len)功能為將 data陣列諄的資料透過LoRa 發送出去。特性有傳輸前會需要先透過 rf95.setFrequency(freq)來設定頻率freq，並且傳輸後常與 rf95.waitPacketSent()憶起使用來確認傳輸是否完成。

2請上網查詢LoRa普遍的封包格式，和其規格做說明。

[資料來源: <https://delorescetleh.medium.com/lora-lorawan-%E7%AC%AC%E5%8D%81%E4%B8%89%E7%AF%80-3ec09afcd5c8> ]

LoRa 的封包格式由三個部分組成: Preamble(前導碼)、Header、Payload。

| **欄位** | **說明** |
| --- | --- |
| Preamble | 預先定義的同步碼，接收端據此開始同步 |
| Header | 包含封包長度、地址（可選）與類型 |
| Payload | 實際資料內容 |
| CRC | 簡單錯誤偵測用的 checksum（可選） |

3請說明這次實驗中的LoRa封包格式為何?

本次實驗data的資料格式由使用者自己定義，格式是

[識別字元 (1 byte)][訊息資料 (N bytes)]，例如說 uint8\_t data[] = "A hellow";

則A 就是識別字元，讓主端來判斷是否要處理。

4.

實驗一註解

#include <RH\_RF95.h> // 引入 RadioHead 函式庫中的 RH\_RF95 模組 (LoRa 模組的驅動程式)

#include <SoftwareSerial.h> // 引入軟體序列通訊的函式庫，讓你能用非預設的腳位進行 Serial 通訊

SoftwareSerial SSerial(2, 3); // 建立一個 SoftwareSerial 物件，使用腳位 2 為 RX、3 為 TX

#define COMSerial SSerial // 定義一個名稱 COMSerial 來代表軟體序列埠

#define ShowSerial Serial // 定義 ShowSerial 為內建硬體序列埠（用於輸出訊息到電腦）

// 建立 LoRa 模組的驅動物件，指定使用 COMSerial 進行通訊

RH\_RF95<SoftwareSerial> rf95(COMSerial);

void setup() {

ShowSerial.begin(115200); // 初始化顯示用的序列埠（硬體序列）與電腦通訊

ShowSerial.println("RF95 client test."); // 顯示啟動訊息

if (!rf95.init()) { // 初始化 LoRa 模組，若失敗則卡住程式

ShowSerial.println("init failed");

while (1); // 無限迴圈停住系統

}

rf95.setFrequency(433.0); // 設定 LoRa 傳輸頻率為 433 MHz（視模組與法規而定）

}

void loop() {

ShowSerial.println("Sending to rf95\_server"); // 顯示即將傳送的訊息

// 準備傳送的資料

uint8\_t data[] = "hello"; // 傳送的字串訊息（uint8\_t 是 byte 陣列）

rf95.send(data, sizeof(data)); // 使用 LoRa 傳送資料

rf95.waitPacketSent(); // 等待資料傳送完成

// 準備接收回覆訊息（以下註解部分可以啟用來實作雙向通訊）

/\*

uint8\_t buf[RH\_RF95\_MAX\_MESSAGE\_LEN]; // 建立一個接收緩衝區

uint8\_t len = sizeof(buf); // 指定緩衝區長度

if (rf95.waitAvailableTimeout(3000)) { // 等待最多 3 秒，看是否有資料可接收

if (rf95.recv(buf, &len)) { // 成功接收到資料

ShowSerial.print("got reply: ");

ShowSerial.println((char\*)buf); // 顯示接收到的資料

} else {

ShowSerial.println("recv failed"); // 資料接收失敗

}

} else {

ShowSerial.println("No reply, is rf95\_server running?"); // 超時沒收到回覆

}

\*/

delay(1000); // 每秒傳送一次

}

實驗二註解

#include <RH\_RF95.h> // 引入 RadioHead 函式庫的 LoRa 模組驅動

#include <SoftwareSerial.h> // 引入軟體序列通訊函式庫

SoftwareSerial SSerial(2, 3); // 使用 GPIO 2 作為 RX、3 作為 TX 建立軟體序列埠

#define COMSerial SSerial // 定義 COMSerial 為軟體序列埠（給 LoRa 模組用）

#define ShowSerial Serial // 定義 ShowSerial 為硬體序列埠（給監控用的 Serial Monitor）

// 建立 RH\_RF95 LoRa 物件，使用 COMSerial 作為通訊通道

RH\_RF95<SoftwareSerial> rf95(COMSerial);

void setup() {

ShowSerial.begin(115200); // 啟動 Serial Monitor 通訊

ShowSerial.println("RF95 server test."); // 顯示啟動訊息

if (!rf95.init()) { // 初始化 LoRa 模組

ShowSerial.println("init failed"); // 如果失敗就印出錯誤訊息

while (1); // 停住程式

}

rf95.setFrequency(433.0); // 設定通訊頻率為 433MHz（視模組與區域法規而定）

}

void loop() {

if (rf95.available()) { // 如果有接收到 LoRa 訊息

uint8\_t buf[RH\_RF95\_MAX\_MESSAGE\_LEN]; // 建立接收緩衝區

uint8\_t len = sizeof(buf); // 設定最大長度

if (rf95.recv(buf, &len)) { // 接收訊息，如果成功

ShowSerial.print("got request: ");

ShowSerial.println((char\*)buf); // 印出接收到的訊息內容（轉為文字）

// 可選：回覆訊息給 client

/\*

uint8\_t data[] = "And hello back to you"; // 要回傳的訊息內容

rf95.send(data, sizeof(data)); // 傳送回覆訊息

rf95.waitPacketSent(); // 等待傳送完成

ShowSerial.println("Sent a reply"); // 印出回覆已送出的提示

\*/

} else {

ShowSerial.println("recv failed"); // 如果接收失敗就顯示錯誤

}

}

}

實驗三註解

////////////主端程式

#include <RH\_RF95.h>

#include <SoftwareSerial.h>

int X; // 辨識字節（用來辨識傳來資料的第一個字元）

SoftwareSerial SSerial(2, 3); // RX, TX

#define COMSerial SSerial

#define ShowSerial Serial

RH\_RF95<SoftwareSerial> rf95(COMSerial); // 建立 LoRa 物件

void setup() {

ShowSerial.begin(115200);

ShowSerial.println("RF95 server test.");

if (!rf95.init()) {

ShowSerial.println("init failed");

while (1); // 初始化失敗則停住

}

rf95.setFrequency(433.0); // 設定頻率為 433 MHz

}

void loop() {

if (rf95.available()) {

// 收到資料時執行

uint8\_t buf[RH\_RF95\_MAX\_MESSAGE\_LEN];

uint8\_t len = sizeof(buf);

if (rf95.recv(buf, &len)) {

X = buf[0]; // 取出收到資料的第一個字元 (用來辨識訊息類型)

if (X == 'A') { // 若第一個字元是 'A'

ShowSerial.print("got request: ");

ShowSerial.println((char\*)buf); // 顯示整段訊息

}

// 如果要回覆的話可以取消以下註解

/\*

uint8\_t data[] = "And hello back to you";

rf95.send(data, sizeof(data));

rf95.waitPacketSent();

ShowSerial.println("Sent a reply");

\*/

} else {

ShowSerial.println("recv failed");

}

}

}

////////////從端程式

#include <RH\_RF95.h>

#include <SoftwareSerial.h>

SoftwareSerial SSerial(2, 3); // RX, TX

#define COMSerial SSerial

#define ShowSerial Serial

RH\_RF95<SoftwareSerial> rf95(COMSerial); // 建立 LoRa 物件

void setup() {

ShowSerial.begin(115200);

ShowSerial.println("RF95 client test.");

if (!rf95.init()) {

ShowSerial.println("init failed");

while (1); // 初始化失敗則停住

}

rf95.setFrequency(433.0); // 設定頻率為 433 MHz

}

void loop() {

ShowSerial.println("Sending to rf95\_server");

// 傳送字串訊息，開頭加上 'A' 作為辨識字元

uint8\_t data[] = "A hellow"; // <--- 修正錯誤，原來的 `"A hellow""` 有語法錯

rf95.send(data, sizeof(data));

rf95.waitPacketSent(); // 等待資料傳送完成

// 可選：等待回覆（目前註解掉）

/\*

uint8\_t buf[RH\_RF95\_MAX\_MESSAGE\_LEN];

uint8\_t len = sizeof(buf);

if (rf95.waitAvailableTimeout(3000)) {

if (rf95.recv(buf, &len)) {

ShowSerial.print("got reply: ");

ShowSerial.println((char\*)buf);

} else {

ShowSerial.println("recv failed");

}

} else {

ShowSerial.println("No reply, is rf95\_server running?");

}

\*/

delay(1000); // 每秒傳送一次

}

動手做註解

#include <RH\_RF95.h>

#include <SoftwareSerial.h>

int X; // 辨識字節用

const int lightSensorPin = A0; // 亮度感測器腳位（未啟用）

int ledPin = 8; // 控制 LED 的腳位

SoftwareSerial SSerial(2, 3); // RX, TX

#define COMSerial SSerial

#define ShowSerial Serial

RH\_RF95<SoftwareSerial> rf95(COMSerial); // LoRa 模組初始化

void setup() {

pinMode(ledPin, OUTPUT); // LED 腳位設為輸出

ShowSerial.begin(115200); // 啟用序列埠

pinMode(lightSensorPin, INPUT); // 亮度感測器設為輸入

if (!rf95.init()) { // 初始化 LoRa 模組

ShowSerial.println("init failed");

while (1); // 若失敗就卡住程式

}

rf95.setFrequency(433.0); // 設定頻率為 433MHz

}

void loop() {

// --- 若要傳送感測值，可啟用以下 ---

/\*

int sensorValue = analogRead(lightSensorPin); // 讀取亮度感測器

int pwmValue = map(sensorValue, 0, 1023, 255, 0); // 越暗 -> 值越大

uint8\_t data[2] = {69, (uint8\_t)pwmValue}; // 'E'=69 作為資料代碼

rf95.send(data, sizeof(data)); // 傳送感測資料給主端

rf95.waitPacketSent();

\*/

// --- 接收主端回傳的 PWM 值 ---

uint8\_t buf[RH\_RF95\_MAX\_MESSAGE\_LEN];

uint8\_t len = sizeof(buf);

if (rf95.waitAvailableTimeout(3000)) { // 等待主端回傳資料，最多等 3 秒

if (rf95.recv(buf, &len)) {

if (buf[0] == 69) { // 若開頭是 'E' (代表是主端傳來的 PWM 資料)

ShowSerial.print("主端回傳 PWM: ");

ShowSerial.println(buf[1]);

int pwm = buf[1]; // 擷取 PWM 值

int value = 255 - pwm; // 反轉亮度（例如：主端愈亮 → LED 愈暗）

analogWrite(ledPin, value); // 寫入 PWM 控制 LED 亮度

// 回傳實際控制值給主端（做確認或回饋用）

uint8\_t dat[2] = {69, (uint8\_t)value};

rf95.send(dat, sizeof(dat));

rf95.waitPacketSent();

}

}

}

delay(100); // 每 100ms 週期處理一次

}