添付資料: LoRa 通信方式において使用するソフトウェアについて

はじめに

本資料中の URL もしくはソースコードのファイル名はクリックすることで、該当する web page や GitHub 上のソースコードを閲覧することができるようになっている。

使用するライブラリ

LoRa コントローラ (HopeRF RF98 あるいは Semtech SX1278 と呼ばれるもの) の制御には、RadioLib (`https://github.com/jgromes/RadioLib/`) を使用する。このライブラリは RF98/SX1278 に限らず、多種多様な無線モジュールを、可能な限り同一の操作方法で Arduino から制御できるように作られたものである。

申請に用いる HopeRF RF98 は、Semtech SX1278 のラベルを変更しただけのものであり、同一の製品と言われている。RadioLib には RF98 の制御に対応するクラスは実装されていないが、SX1278 クラスを使用して制御することができるためにこれを使用し、本資料中の説明も SX1278 として進める。

アプリケーション

現時点において、実用的なアプリケーションの作成計画は無い。しかし、アプリケーションを作成するための礎として、RadioLib の使用例として提供される Arduino スケッチ (`SX127x_Transmit.ino`) をベースに、暗号化しない平文を送信する実験を行いたいと考えている。

使用例のスケッチ (アプリケーション) 内で行われる処理を簡単に要約すると、

1. SX1278 と Arduino の接続に使用する I/O ピンを設定 (SX1278 radio = new Module())
2. SX1278 を初期化 (radio.begin())
3. メッセージの送信 (radio.transmit("Hello, world!"))

となる。RF98 (SX1278) と Arduino の接続は SPI を使用し、SPI の制御は Arduino 標準ライブラリである <SPI.h> を使用する。このライブラリは、RadioLib 側から呼び出されるため、スケッチ側から直接 SPI を操作することで SX1278 の動作に影響を及ぼすようなことはない。

なお、使用例のスケッチそのままではデフォルトの設定 (周波数 434.0 MHz, 周波数帯域幅 125kHz, SF=9, CR=7) で電波を発射してしまうため、これらは適切な設定を行うようコードを追加した上で実験を行う。

ライブラリの機能説明

RadioLib の SX1278 制御クラスの説明は `https://jgromes.github.io/RadioLib/class_sx1278.html` にあり、そのうち LoRa 変調を使用して通信を行うために用いる主なメソッドを記す。

begin() SX1278 を LoRa モードに設定して動作を開始

reset() SX1278 をリセット

setFrequency() 周波数設定 (137.0〜525.0 MHz)

setBandwidth() 周波数帯域幅の設定 (7.8〜500 kHz)

setSpreadingFactor() 拡散率 (SF) の設定 (6〜12)

setCodingRate() 誤り訂正符号 (CR) の設定

setOutputPower() 送信電力の設定 (2〜17 dBm)

setGain() 受信感度の設定 (0〜6)

送受信については、各種無線モジュールで共通に使用できるメソッド (PhysicalLayer Class, `https://jgromes.github.io/RadioLib/class_physical_layer.html`) から、代表的な以下の二つを記す。

transmit() 文字列の送信

receive() 文字列の受信

これらのメソッドを使用してアプリケーションを作成するが、その際は setFrequency() による周波数設定はアマチュアバンド (430〜440 MHz) の範囲内とし、setBandwidth() による周波数帯域幅の設定も 30kHz 以下となるようにする。

送信処理の流れ

RadioLib のスケッチ例 (`SX127x_Transmit.ino`) にある radio.transmit(const char* str) メソッドを使用した送信において、暗号化処理が含まれていないことを示すため、送信処理の流れを解説する。

1. radio.transmit(const char* str) は、PhysicalLayer::transmit(const char* str, uint8_t addr) に対応する[1]。

2. PhysicalLayer::transmit(const char* str, uint8_t addr) の処理は `src/protocols/PhysicalLayer/PhysicalLayer.cpp` にあり、
与えられた str の内容を変えることなく最終的に PhysicalLayer::transmit(uint8_t* str, size_t len, uint8_t addr) を呼び出す。

3. PhysicalLayer::transmit(uint8_t* str, size_t len, uint8_t addr) は `src/protocols/PhysicalLayer/PhysicalLayer.h` において純粋仮想関数として定義されているため、PhysicalLayer を継承した各コントローラの transmit() が実際の処理を担うこととなる。

4. SX1278 クラスは SX1278 Class Reference (`https://jgromes.github.io/RadioLib/class_s_x1278.html`) にあるように、PhysicalLayer → SX127x → SX1278 クラスを継承する。

5. SX1278 の送信処理は SX127x::transmit(uint8_t *data, size_t len, uint8_t addr) (`src/modules/SX127x/SX127x.cpp`) にあり、ここから SX127x::startTransmit(uint8_t* data, size_t len, uint8_t addr) を呼び出して SX1278 の送信用 FIFO にデータを書き込む。

6. ここまでの経路において、radio.transmit() で与えられたデータは無加工で SX1278 に渡るため、RadioLib 内で暗号化は行われないと言える。

【以下余白】

[1] C++ においては addr の指定を省略した場合の初期値を定義でき、ここでは省略時に addr = 0 となるようヘッダファイルに記述されている。