LoRa APRS の製作と運用

周波数拡散、LoRa, APRS, XIAO RP2040, GYSFDMAXB, S-AU26,

1 LoRa + APRS の意義

1.1 LoRa とは

最近しばしば”LoRa”という用語を目にするでしょう。LoRa®(以下、LoRa と略します。) の名は Long Range に由来します。

LPWA(Low Power Wide Area) を意味する省電力広帯域通信の技術規格の一つである LoRa は、米国の Semtech Corp. が開発し、同社を中心とする共同体 LoRa Alliance がその普及を推進しています。オープンな規格の下に、そこには世界 500 社以上の企業が加盟し、その利用が広がっています。国内でも、日本 LoRa アライアンス普及開発推進協会が設立され、情報共有と LoRa 技術の利用促進を図っています。

LoRa 規格は、OSI モデルの物理層において、周波数拡散方式の一つであるチャープ (Chirp) 変調方式を用いています。さらに、広域ネットワークを構築するための OSI リンク層として、LoRaWAN プロトコルを含んでいます。

周波数拡散変調は、ベースバンド信号に比べてはるかに広い周波数帯に信号スペクトラムを拡散することで、いわゆる拡散利得を生じることになり、雑音レベル以下の目的信号を有効に復調することができます。併せて、LoRa では使用周波数帯をサブギガ帯とすることによって、2.4GHz 帯の LAN や Bluetooth のそれよりも遠距離に伝搬させます。この特性を生かして、LoRa は LPWA 界で確固たる地位を占めるでしょう。

LoRa を実現するコア・デバイスには、元祖 Semtech 社の SX1276〜SX1279 をはじめ MicrochipTechnology Inc. の ATSAMR34J16〜18 などが供給されています。ただし、これらの高集積度 IC は QFN28 または TFBGA パッケージなので手作業での実装が難しく、最終製品の小ロット生産にはコア・デバイスと周辺回路が組み込まれている Ai-Thinker 社の Ra-02、HOPERF 社の RFM95/96/97/98 または Microchip 社の RN2483 などのモジュールを用います。例えば、SX1278 が組み込まれている Ra-02 の概要は次のとおりです。

Ai-Thinker の Ra-02 LoRa モジュールの特性と形状

- LoRa スペクトラム拡散変調
- 20 dBm の定 RF 出力
- FSK, GFSK, MSK, GMSK, LoRa 及び OOK 変調
- 300 kbps までプログラム可能
- 256 byte までの CRC
- 感度:-148 dBm 以下
- 自動 RF 信号検出,CAD モードと高速 AFC
- RSSI ダイナミックレンジ:127 dB
- 通信距離:15 km
- 無線周波数:137〜525 MHz
- SPI シリアル通信

シールド内に SX1278、XO、LDO、LPF がある。

コア・デバイスの多くは、サブギガ帯の LoRaWAN を意識して設計されていますが、SX1278 は 137〜525 MHz の周波数範囲に特化されていますので、VHF または UHF、とりわけ 435 MHz 帯のアマチュアバンドでの使用が可能です。また、スプリング・アンテナを直結する形式の Ra-01 よりも IPEX/U.FL コネクタを出力端子とする Ra-02 の方がアンテナ選択の自由度があります。

LoRa が LoRaWAN を商業ターゲットとして開発されたとは言え、周波数拡散方式のメリットを Peer to Peer(又は Material to Material) の通信に使えない訳ではありません。これをアマチュア無線のパケット通信に利用することができます。

1.2 もう一つの APRS

低電力・遠距離通信デバイスである LoRa を APRS®(以下、APRS と略します。) に活用する試みは、すでにオーストリアなど欧州各国で活発です。そこでは、LoRa APRS の名で、433.575 MHz を参照周波数とする LoRa 変調の APRS 電波が盛んに実験されているそうです。従来方式である 1,200bps(F2D) または 9,600 bps(F1D) のものとは別の APRS ネットワークが構築されています。独立した APRS とはいえ、iGate 局を通じて世界共通の IS サーバーに接続されていますから、例えば aprs.fi などの画面上ではシームレスに位置情報やメッセージの閲覧ができます。

欧州での実例を見ると、一般に LoRa APRS のトラッカー (移動側) は 100 mW 程度の省電力で運用し、iGate(固定側) との間 20〜50 km 程度を最大通達距離としているようです。また、山上などのデジピート局は置かず, その代わりに自宅などに簡単に設置できる iGate 局が数多く必要な様子です。次の図は、2021 年 5 月 20 日時点でのベルギーにおける APRS の運用状況で、LoRa APRS は現在着々と増殖中の模様です。

移動局は”Car”または”Person”のシンボルマークで現在位置が表示されていますが、それが標準方式の APRS か LoRa APRS かは区別できません。
固定局の”I”は標準方式の iGate であり、”L”が LoRa の iGate です。

図:FaceBook のグループ「LoRa APRS」への Luc Bodson の投稿を引用

1.3 日本独特の 438MHz 帯での運用

日本において LoRa APRS を運用するとなると、欧州なみの 433.575(または 433.775)MHz と同じ周波数という訳にはなりません。「無線局運用規則第二百五十八条の二の規定に基づくアマチュア業務に使用する電波の型式及び周波数の使用区別」(平成二十一年三月二十五日)(総務省告示第百七十九号)によると、438.0〜439.0 MHz の周波数範囲が無条件で全ての電波の型式で使用できますので、この周波数帯でアマチュア無線の LoRa を使うことになります。

隣国と隔たっている日本の場合、UHF 帯の周波数使用は独自に決めても支障がありません。そこで、LoRa APRS の約束周波数は 438.575 MHz としました。この周波数は、LoRa の BW 最大値 (500 kHz) を用いた場合でも周波数使用区分を逸脱しないというだけで選んだので、他に特段の技術的理由がある訳ではなく、別の事情があれば変更する余地を持たせます。

他のアマチュア局との混信については、もともと周波数拡散方式と単一キャリア通信方式とのイミュニティ性は高く、仮に同時に異なる通信方式の電波が存在してもそれぞれの通信の疎通には問題がないとされています。従って、438〜439 MHz の周波数利用には格別な障害は発生しないはずです。

2 iGate(固定局) の作成

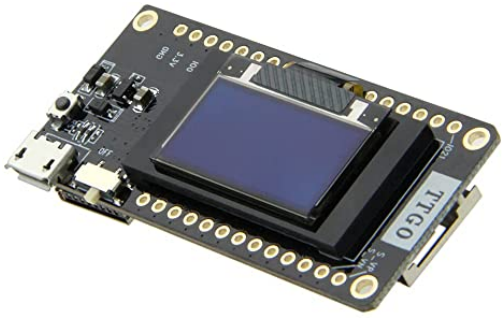
iGate とは、LoRa APRS の電波を受信・解読し、その情報をインターネットを通じて APRS データ・サーバーに提供する装置です。LoRa トラッカー (移動局) との間でメッセージの交換を予定しない局では、受信機能だけ備えており電波の送信はしません。

2.1 iGate のハードウェア例

iGate を構築するには、市場に出回っている LoRa ボードを利用するのが簡便で、その事例を Web 上でみることができ、参考になります。

ここでは、LILYGO® 製 TTGO LoRa32 OLED v2 ボードを紹介します。その主要な規格とハードウェア構成は次のとおりです。

- 入力電圧:5 V
- 許容電流:10〜14 mA
- 送信時電流:90 mA @17 dBm
- 運用周波数:433〜470 MHz
- 送信電力:20 dBm 最大
- 受信感度:-136 dBm @ LoRa 125 kHz SF=12
- FIFO 容量:64 byte
- 伝送速度:0.018 k〜37.5 kbps @LoRa
- 変調モード:FSK,GFSK,MSK,GMSK,LoRa,OOK
- インターフェース:WiFi 及び BlueTooth
- 運転温度:-40〜85 ℃
- RSSI 機能
- AFC 機能
- AGC 機能



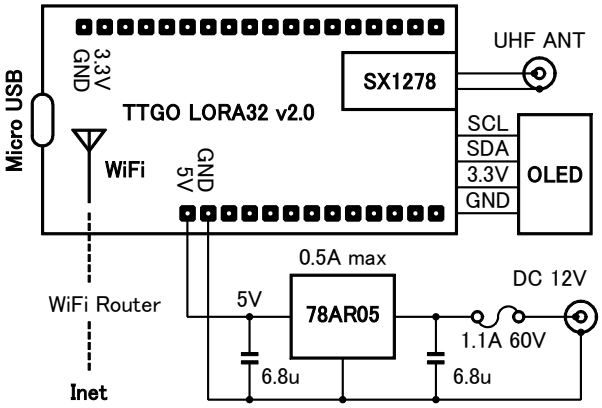
LoRa:T-LORA SX1278
MCU:ESP32
USB-UART ブリッジ:CP2104
Micro USB
Micro SD スロット
WiFi
BlueTooth
OLED

ちなみに、TTGO LoRa32 OLED v2 のデータの参照先は次です。

http://www.lilygo.cn/prod_view.aspx?TypeId=50060&Id=1319&FId=t3:50060:3

TTGO LoRa32 OLED V2 ボードを用いて、簡単に iGate 局を作ってみましょう。次の外観写真と回路図を見ると、極めてシンプルであることに気づくでしょうが、これだけで充分です。

TTGO LoRa32 V2 で作る iGate



TTGO LoRa32 OLED V2 には、OLED ディスプレーはすでに組み込まれています。また、LoRa アンテナ・コネクタは I-PEX/U.FL 仕様が備わっており、対応するケーブルで接続します。WiFi(2.4 GHz 帯) により WiFi 対応ルータとの物理的接続は済んでいます。電源は DC 5 V で電流容量 0.1 A を供給します。

本機の電源については、一点だけ留意します。USB はファームウェアのローディングを行う際に必要とするので、平常運転時に接続を外しますと、電源 5 V が供給されません。そこでボードに内蔵されているスイッチを介して、外付けの 78AR05 から電源を加えます。

2.2 TTGO LoRa32 V2 のファームウェア

iGate 装置のファームウェアは、プログラム開発コミュニティ GitHub から取得できます。ソース・コードは次の URL にありますので、ダウンロード及び解凍を行って、任意のフォルダに保存しておきます。

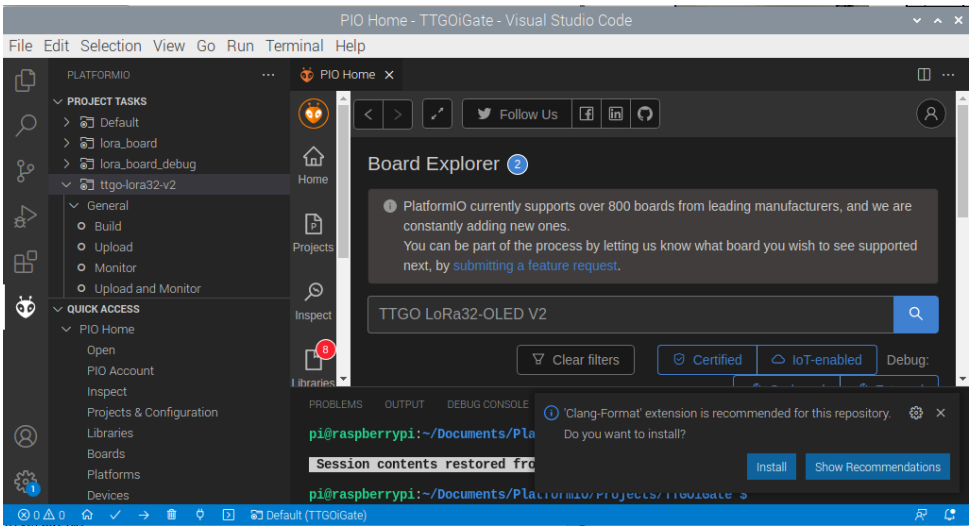
https://github.com/lora-aprs/LoRa_APRS_iGate

早速ファームウェアの作成に進みたいところですが、Microsoft 社が提供する無料のエディタ Visual Studio Code と、同じく無料のプラグインである PlatformIO IDE を使うと簡単なので、予めそれらが使える PC 環境を整えておきます。なお、開発環境は Windows、Mac、Linux いずれの PC に対応したものが用意されています。ダウンロード先は次の URL です。

https://code.visualstudio.com/download

https://platformio.org/platformio-ide

PlatformIO が正常に動作しているならば、次ページ上段の画像に似たような画面になるはずです。IDE についての詳しい説明は、Web 検索で知ることができますので、ここではそれに委ねます。



プロジェクトの作成から、ボードの FlashROM への書き込みまでの手順は次のとおりです。

1. COM ポートの選択など：platformio.ini ファイルが自動的に設定します。
2. プロジェクトの新規作成：`QUICK ACCESS >> Project & Configuration >> Create New Project` へ進みます。
3. プロジェクト名：先にソース・コードを保存したフォルダ名を記入します。
4. /data/is-cfg.json ファイルの編集： 自局の固有データに書き直します。特に、Callsign、AP、Position などは適正にします。作業は PlatformIO IDE で行えます。
5. コンパイル：`PROJECT TASKS >> Build` を使います。
6. 書き込み：同じく `PROJECT TASKS >> Upload` で完了です。

次は、4. で参照する/data/is-cfg.json ファイルの例です。適宜自局固有の文字で修正します。

```
{
    "callsign":"JA5FP",
    "wifi": {
        "AP":[{
            "SSID":"Buffalo-G-7898",
            "password":"t**********t"
            }
        }
    "beacons": {
        "message": " LoRa 438M575",
        "position":{
            "latitude":35.****,
            "longitude":140.****
            },
        "timeout":30
        },
    "lora": {
        "frequency_rx":438575000,
        "spreading_factor":12,
        "signal_bandwidth":125000,
        "coding_rate4":5
        },
    "display": {
        "always_on":true,
        "timeout":10,
        "overwrite_pin":0,
        },
```

```
    "ftp": {
        "active":false,
        "user": [ {
            "name":"ftp",
            "password":"ftp"
            }
            ]
        },
    "ntp_server":"ntp.nict.jp"
    }
}
```

全てに成功すると、FTP によって APRS サーバにデータが伝送され、aprs.fi に次のような Raw データとシンボル L の地図表示が出ます。

```
2022-03-04 10:13:22 JST: JA5FP>APLG01,TCPIP*,qAC,T2IRELAND:=3540.07NL14010.33E& LoRa 438M575
2022-03-04 10:43:22 JST: JA5FP>APLG01,TCPIP*,qAC,T2IRELAND:=3540.07NL14010.33E& LoRa 438M575
```

3 (移動局) トラッカの作成

LoRa APRS トラッカにも LILYGO®TTGO T-Beam V1.1 ESP32 などのボードが販売されていますが、比較的高価であることと、送信だけに限定するならばソフトウェアも簡単なので、自作に挑戦してみましょう。

コア・デバイスは最も一般的な SX1278 を使います。その機能と使い方を知るために、SX1276/77/78/79 Datasheet を次の URL からダウンロードし、手元に置くことをお奨めします。

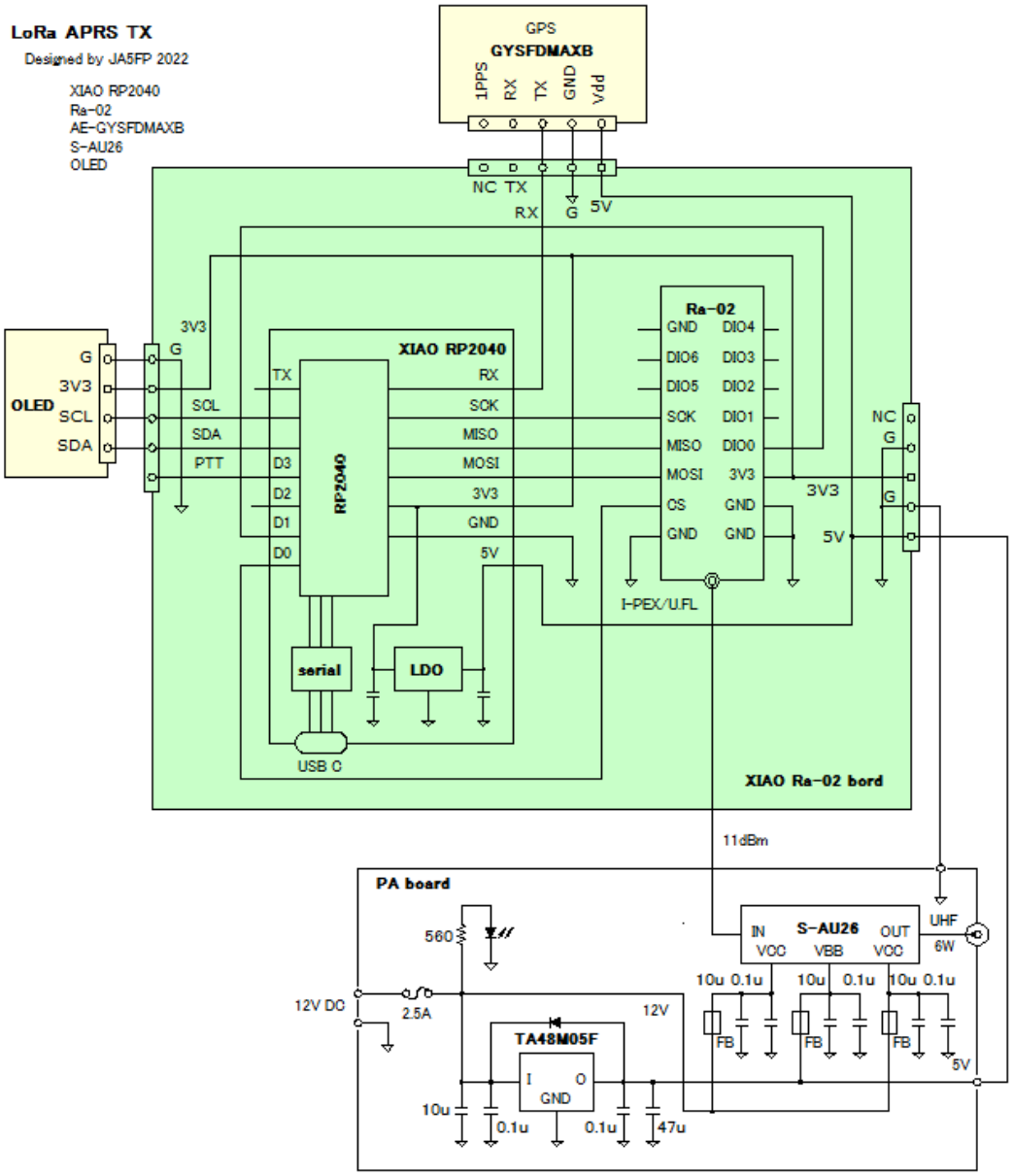
https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr
/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

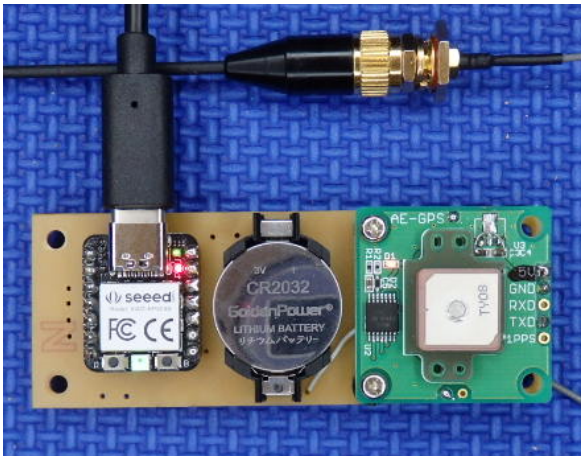
3.1 トラッカのハードウェア例

筆者の自作トラッカの回路構成を次頁の図で示します。

パーツ選定の理由を説明します。

- 制御ボードには XIAO RP2040 を使います。このボードは小型ながら本用途に十分な IO 数です。先行品種の XIAO と同じピン配置ながら MCU の RP2040 を搭載していて、低価格です。RP2040 は、SPI、I2C の他に USB 通信機能があり、UART が備わっています。
- AE-GPS GYSFDMAXB は、秋月電子から比較的安価に入手できる高感度 GPS モジュールです。室内や車中でも十分に受信します。
- Ra-02 は、SX1278 を 2 mm ピッチの SMD パッケージに納めたものです。このサイズならば、手作業のハンダ付けができます。他に、LPF、XO クロック、LDO や Ipex/MHF4 コネクタも付いています。
- 東芝製の PA パッケージ S-AU26 は、LoRa APRS の伝搬範囲を 50 km 程度まで伸ばすために増力します。これはハンディ機用の手頃なサイズです。トラッカが送信のみなので TR リレーは必要なく、PA が FM 用でありアイドリング電流が少量なので PTT スイッチは特に設けていません。





XIAOrp2040 LoRa APRS トラッカの外観

Ra-02 パッケージは裏面 (ハンダ面) に装着されている。
GPS のバック・アップ電池 CR2032 が大きい。

3.2 XIAO RP2040 と Ra-02 のファームウェア

以下に、筆者が用意したソース・コード xiao-aprs-tx.ino 及び XIAOrp2040.h を示します。コンパイル及びアップロードは Arduino IDE で行います。以下のファイルの内で、それぞれの Callsign など適正に修正をして、コンパイルします。

なお、ソース・コードの記述に関して特に留意すべき点は、次のとおりです。

- SX1278 レジスタ値の設定は SPI 通信で行いますが、その書式は「アドレス ”,” データ」です。従って、一つのコマンドは 2 バイトを送信することになります。ただし、連続したアドレスに連続してデータを送る場合には「最初のアドレス ”,” データ ”,”次のデータ」の形をとることができます。レジスタ値を取得するのは、[アドレス ”,” ダミー・バイト」を送り、MISO の 2 バイト目のデータを読みます。
- SX1278 の初期モードは FSK LF STDBY となっています。LoRa モードに変更するには、一旦 SLEEP モードにしてから行わなければなりません。
- Ra-02 の高周波回路は SX1278 の PA_BOOST に結線されていますが、SX1278 のデフォルトでは PaSelect レジスタが”0”すなわち RFO_LF になります。それでは出力されませんので、必ずソフトウェアで PA_BOOST を使うように強制します。
- LoRa のプロトコルが守られるように SX1278 が自動的にプリアンブルや CRC を挿入するので、プログラマーはそれを意識することなく、単にメッセージを入れれば伝送されます。メッセージを FiFo に送ること、及びペイロード長を指定するだけを行います。
- APRS のプロトコルに合致させるため、プリアンブルとして 3 バイト分のデータを挿入します。

```
/*******************************************************
XIAOrp2040.h  (c) 2022 Yukihisa Aida JA5FP
File name:"XIAOrp2040.h"
This file configures XIAO RP2040 module, OLED and NeoPixel LEDs.
*******************************************************/
// RP2040 pins //
#define _RST 27                     // D1
#define _CS 26                      // D0
#define _MOSI 3                     // D10
#define _MISO 4                     // D9
#define _SCK 2                      // D8
#define _SCL 7                      // D5
#define _SDA 6                      // D4
#define _PTT 29                     // D3
#define _LED 25                     // blue LED
#define _RGB_PWR 11                 // NeoPixel power
#define _RGB_PIX 12                 // NeoPixel line

/*******************************************************
station.h  (c) 2022 Yukihisa Aida JA5FP
File name:"station.h"
This file specify the individial contents of esp32-aprs-tx.
*******************************************************/

// LoRa APRS constant //
const String Callsign = "JG1JZL-9", StationSymbol = ">";
const uint8_t Power = 15;                       // in dBm
const uint16_t WaitTime = 60;                   // unit is second
const uint32_t Freq = 438575;                   // center frequency
const uint8_t PacketType = 60, Destination = 255, Source = 1; // preamble

/*******************************************************
```

```
/*************************************************************
xiao-aprs-tx.ino  (c) 2022 Yukihisa Aida JA5FP"

 This is an Arduino sketch to transmit moving informations
 for APRS iGate with LoRa Protocol on UHF amateur band.
 Reference: https://github.com/IoT4pi/LoRa-APRS-Sender
            https://github.com/lora-aprs/LoRa_APRS_Tracker/tree/master/src
 BOM: AE-GPS(MT3339) GPS module
      Ra-02(SX1278) LoRa module
      SSD1306 OLED module
      S-AU26 PA module
      XIAO(RP2040) IO contoroler
 Interface to PC: USB-C
 Development: Arduino IDE, Board: Seeed XIAO Rp2040
*************************************************************/
#include <Adafruit_TinyUSB.h>
#include <TinyGPS++.h>
#include <Adafruit_NeoPixel.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "XIAOrp2040.h"
#include "station.h"

// functions //
void writeSx1278(uint8_t, uint8_t);
uint8_t readSx1278(uint8_t);
void brightLED(uint8_t, uint8_t, uint8_t);
void darkLED(void);
void formData(void);
void sendLoRa(uint8_t);

// variables //
uint32_t nPac;                                  // times of TX
String sAPRS;                                   // APRS string

// object //
TinyGPSPlus gps;
Adafruit_SSD1306 display(128, 64, &Wire1, -1);
Adafruit_NeoPixel pixels(1, _RGB_PIX, NEO_GRB + NEO_KHZ400);

void setup() {
  uint32_t freq;

  // specify XIAO RP2040 parameters //
  Serial.begin(115200);                         // USB and monitor serial
  Serial1.begin(9600);                          // hardware serial1
  pixels.begin();
  pinMode(_RST, OUTPUT);
  pinMode(_CS, OUTPUT);
  pinMode(_LED, OUTPUT);
  pinMode(_RGB_PWR, OUTPUT);
  digitalWrite(_RST, HIGH);
  digitalWrite(_CS, HIGH);
  digitalWrite(_LED, HIGH);
  digitalWrite(_RGB_PWR, HIGH);

  // specify SPI prameters //
  SPI.begin();
  SPI.beginTransaction(SPISettings(1E6, MSBFIRST, SPI_MODE0));

  // OLED display //
  Wire1.setSDA(_SDA);
  Wire1.setSCL(_SCL);
  display.setRotation(2);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c)) {
      Serial.println("SSD1306 allocation failed");
      for(; ;);
      }
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
```

```
  display.setCursor(0, 25);
  display.print("setting --");
  display.display();

  // initiate SX1278 //
  digitalWrite(_RST, LOW);                        // reset SX1278
  delay(100);
  digitalWrite(_RST, HIGH);

  // change to LoRa mode //
  writeSx1278(0x01, 0x08);                        // FSK LF SLEEP
  writeSx1278(0x01, 0x88);                        // LoRa LF SLEEP
  writeSx1278(0x01, 0x89);                        // LoRa LF STDBY
  writeSx1278(0x09, 0x89);                        // PA BOOST Pout=11dBm(12mW)
  freq = Freq;                                    // encode Frf
  freq <<= 11;
  freq /= 125;
  writeSx1278(0x06, (char)(freq >> 16));          // MSB
  writeSx1278(0x07, (char)(freq >> 8));           // MID
  writeSx1278(0x08, (char)(freq));                // LSB
  writeSx1278(0x1d, 0x72);                        // BW=125kHz CR=4/5 explicit
  writeSx1278(0x1e, 0xc0);                        // SF=12 4096 chips/symbol
  writeSx1278(0x26, 0x08);                        // low data rate optimize
  writeSx1278(0x11, 0xf7);                        // flag mask 0b11110111
  brightLED(1, 1, 0);                             // Now ready to start
  delay(1000);
  display.clearDisplay();
  }

void loop() {
  uint8_t strLen, i;

  while(Serial1.available() > 0) {
    gps.encode(Serial1.read());
    if(gps.location.isUpdated()) {
      Serial.println(">>>>>>>>>> STATUS >>>>>>>>>>");
      Serial.print("Encoded NMEA: ");
      display.clearDisplay();
      formData();                                 // make APRS string
      brightLED(0, 1, 0);
      Serial.print("Data: ");
      Serial.print(sAPRS);
      strLen = sAPRS.length();
      Serial.print(" (Length=");
      Serial.print(strLen);
      Serial.println(")");
      Serial.print("Preamble: ");                 // view status
      Serial.print("PacketType=");
      Serial.print(PacketType);
      Serial.print(",Destination=");
      Serial.print(Destination);
      Serial.print(",Source=");
      Serial.println(Source);
      sendLoRa(strLen);                           // transmit LoRa
      Serial.print("TX done, Number=");
      Serial.println(nPac);
      display.setTextSize(1);
      display.setCursor(0, 0);
      display.print(Callsign);
      display.setCursor(60, 0);
      display.print("LoRa-APRS");
      display.setCursor(0, 15);
      display.print("Freq.");
      display.setCursor(60, 15)
      display.print(Freq);
      display.setCursor(0, 30);
```

```
      display.print("Wait(s) =");
      display.setCursor(60, 30);
      display.print(WaitTime);
      display.setTextSize(2);
      display.setCursor(60, 50);
      display.print(nPac);
      Serial.print("Wait(sec)=");
      Serial.println(WaitTime);
      Serial.println("");
      display.setCursor(0, 50);
      for(i = 0; i < 5; i ++){                    // marker on OLED
        display.print(">");
        display.display();
        delay(WaitTime * 198);
        } // >>>>>>
      } // updated
    } // while(Serial1 available)
  } // loop

void formData() {                                 // compress GPS data
  uint16_t alt, ilan1, ilan2, ilan3, ilan4;
  uint16_t ilon1, ilon2, ilon3, ilon4;
  float lati, lngi;
  uint32_t lLat, lLng, lAlt1, lAlt2, iTemp;
  double lAlt;
  String ns, ew;

  lati = gps.location.lat();
  lngi = gps.location.lng();
  alt = gps.altitude.meters();
  Serial.print("lat=");
  Serial.print(lati, 5);
  Serial.print(", lon=");
  Serial.print(lngi, 5);
  Serial.print(", alt=");
  Serial.println(alt);
  if (lati < 0) {ns = "S";} else {ns = "N";}
  if (lngi < 0) {ew = "W";} else {ew = "E";}
  lLat = 380926 * (90 - lati);
  lLng = 190463 * (180 + lngi);
  if (alt > 0){
    lAlt = log(alt) / log(1.002);
    lAlt1 = lAlt / 91;
    lAlt2 = (uint16_t)lAlt % 91;
    }
  else {
    lAlt1 = 0;
    lAlt2 = 0;
    }
  ilan1 = lLat / pow(91, 3);
  iTemp = lLat % (uint32_t)pow(91, 3);
  ilan2 = iTemp / pow(91, 2);
  iTemp = lLat % (uint32_t)pow(91, 2);
  ilan3 = iTemp / 91;
  ilan4 = lLat % (uint32_t)91;
  ilon1 = lLng / pow(91, 3);
  iTemp = lLng % (uint32_t)pow(91, 3);
  ilon2 = iTemp / pow(91, 2);
  iTemp = lLng % (uint32_t)pow(91, 2);
  ilon3 = iTemp / 91;
  ilon4 = lLng % (uint32_t)91;
  sAPRS = Callsign;                               // Source
  sAPRS += ">APLRP:!";                            // Destination
  sAPRS += "/";                                   // Station table
  sAPRS += char(ilan1 + 33);
  sAPRS += char(ilan2 + 33);
  sAPRS += char(ilan3 + 33);
```

```
  sAPRS += char(ilan4 + 33);
  sAPRS += char(ilon1 + 33);
  sAPRS += char(ilon2 + 33);
  sAPRS += char(ilon3 + 33);
  sAPRS += char(ilon4 + 33);
  sAPRS += StationSymbol;                         // Station symbol
  sAPRS += char(lAlt1 + 33);
  sAPRS += char(lAlt2 + 33);
  sAPRS += char(0x3e);
  sAPRS += "/Alt(m)=";                            // Other message
  sAPRS += alt;
  sAPRS += " LoRa ";
  sAPRS += Freq;
  sAPRS += "kHz";
  }

void sendLoRa(uint8_t txEnd) {
  uint8_t i, val;
  uint8_t lPac;                                   // length of packet
  uint16_t txTimeout;

  writeSx1278(0x0d, 0x80);                        // FIFO address
  digitalWrite(_CS, LOW);
  SPI.transfer(0x80);                             // FIFO
  SPI.transfer(PacketType);                       // 60 = 0x3c = "<"
  SPI.transfer(Destination);                      // 255 = 0xff
  SPI.transfer(Source);                           // 1
  lPac =3;                                        // preamble
  for (i = 0; i < txEnd; i ++) {                  // write to FIFO
    SPI.transfer(sAPRS.charAt(i));                // message
    }
  digitalWrite(_CS, HIGH);
  lPac += i;
  writeSx1278(0x22, lPac);                        // payload length
  Serial.print("TX start, Power=");
  Serial.print(Power);
  Serial.println("dBm");
  writeSx1278(0x01, 0x8b);                        // LoRa LF TX
  brightLED(10, 10, 10);                          // TX state
  txTimeout = 2 * lPac;
  do {
    val = readSx1278(0x12);                       // check TxDone flag
    txTimeout --;
    } while(txTimeout > 0 && val == 0x00);
  brightLED(0, 1, 0);                             // Now under routine
  if (txTimeout == 0) {
    Serial.println("Error,timeout");
    }
  else nPac ++;
  }

void writeSx1278(uint8_t addr, uint8_t val) {
  digitalWrite(_CS, LOW);
  SPI.transfer(addr | 0x80);                      // use MOSI
  SPI.transfer(val);                              // 2nd byte of MOSI
  digitalWrite(_CS, HIGH);
  }

uint8_t readSx1278(uint8_t addr) {
  uint8_t val;

  digitalWrite(_CS, LOW);
  SPI.transfer(addr);                             // use MOSI
  val = SPI.transfer(0x00);                       // 2nd byte to read MISO
  digitalWrite(_CS, HIGH);
  return(val);
  }

void brightLED(uint8_t red, uint8_t green, uint8_t blue) {
  pixels.setPixelColor(0, pixels.Color(red, green, blue));
```

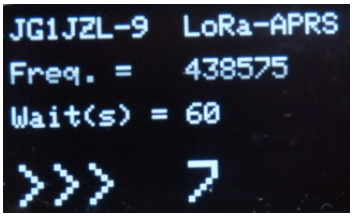
```
  pixels.show();
  }
void darkLED(void) {
  pixels.setPixelColor(0, pixels.Color(0, 0, 0));LoRa-APRS-40km.png
  pixels.show();
  }
```

3.3 モニター表示

本機のモニター情報は、次のように表示されます。先ず、Arduino IDE のシリアルモニタのテキストはこのようになります。

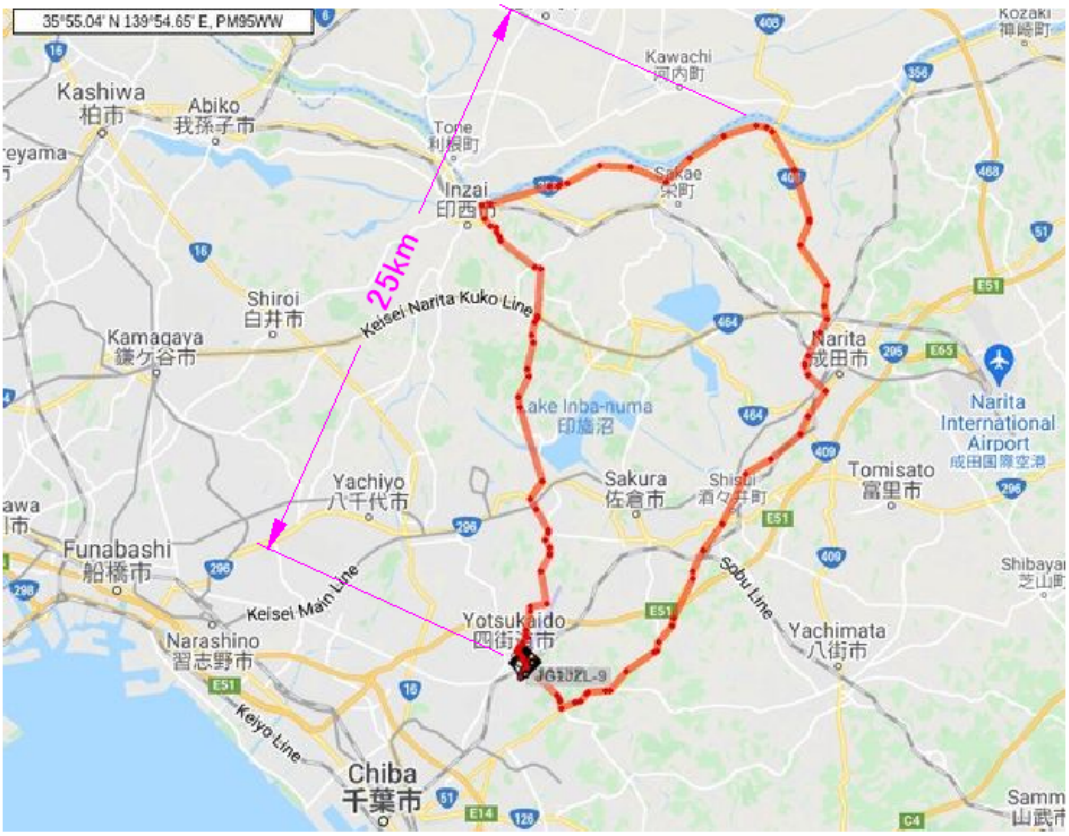
```
10:21:45.220 -> >>>>>>>>>> STATUS >>>>>>>>>>
10:21:45.220 -> Encoded NMEA: lat=35.66794, lon=140.17210, alt=37
10:21:45.220 -> Data: JG1JZL-9>APLRP:!/<K:!qtx9>4o>/Alt(m)=37 LoRa 438575kHz (Length=54)
10:21:45.220 -> Preamble: PacketType=60, Destination=255, Source=1
10:21:45.220 -> TX start, Power=37dBm
10:21:45.220 -> TX done, Number=3
10:21:45.220 -> Wait(sec)=60
10:21:45.220 ->
```

また、本機に内蔵する OLED は次のキャラクターを表示します。”>”は次回送信までの待機時間を示すマーカです。その行の数値は送信回数の累計数です。



4 LoRa APRS の出来栄え

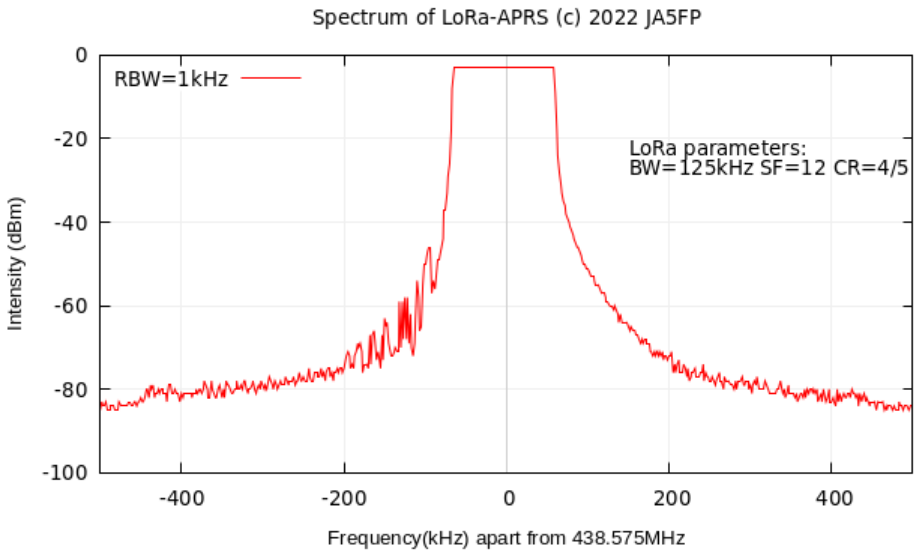
上記の第 2 項及び第 3 項で紹介した iGate とトラッカの組み合わせて、実際に aprs.fi に表れる画像を示します。iGate 局は二階建て家屋の屋根上のグランドプレーン・アンテナを使用し、トラッカ側は車載アンテナによる記録です。おおよそ 30 km は確実に届きデコードしていると認められます。



aprs.fi 上の Raw データは、圧縮モードで次のように表示されます。

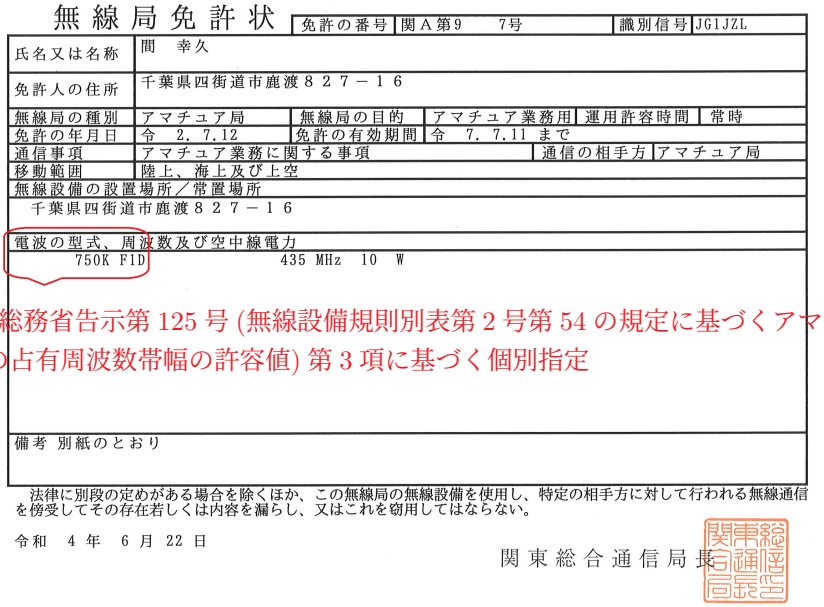
2022-03-06 12:05:14 JST: `JG1JZL-9>APLRP,qAO,JA5FP:!/<K9<qtwp>1L>/Alt(m)=20 LoRa 438575kHz`
2022-03-06 12:10:06 JST: `JG1JZL-9>APLRP,qAO,JA5FP:!/<K: ̀qtwd>5!>/Alt(m)=38 LoRa 438575kHz`

この LoRa-APRS トラッカの送信電波のスペクトラムは、次の絵のようになります。



5　LoRa APRS の免許

これが、筆者が 2022 年 6 月 22 日付けで取得した日本初の LoRa APRS 局の免許状です。



無線局免許状

免許の番号	関Ａ第9　　7号	識別信号	JG1JZL

氏名又は名称	間　幸久			
免許人の住所	千葉県四街道市鹿渡８２７－１６			
無線局の種別	アマチュア局	無線局の目的	アマチュア業務用	運用許容時間　常時
免許の年月日	令　2．7.12	免許の有効期間	令　7．7.11 まで	
通信事項	アマチュア業務に関する事項		通信の相手方	アマチュア局
移動範囲	陸上、海上及び上空			
無線設備の設置場所／常置場所				
千葉県四街道市鹿渡８２７－１６				
電波の型式、周波数及び空中線電力				
750K F1D	435 MHz　10　W			

平成 21 年総務省告示第 125 号 (無線設備規則別表第 2 号第 54 の規定に基づくアマチュア局の無線設備の占有周波数帯幅の許容値) 第 3 項に基づく個別指定

備考　別紙のとおり

法律に別段の定めがある場合を除くほか、この無線局の無線設備を使用し、特定の相手方に対して行われる無線通信を傍受してその存在若しくは内容を漏らし、又はこれを窃用してはならない。

令和　4 年　6 月 22 日

関東総合通信局長

ここで赤字の部分は、もちろん免許状に記載されているのではなく、筆者の注書きです。チャープ変調による周波数帯域 (500kHz 以下) が十分に納まる占有周波数帯幅 750kHz の許可を得ています。

実際の許可手続きは、免許権者の下部にいる審査担当者の理解能力次第で日時を要することになるでしょう。幸いに、平成 21 年告示第 125 号では、その第 3 において占有帯域幅についての特例指定ができることになっており、これを根拠にチャープ変調が認められた訳です。

直近の改正である令和 5 年 3 月 22 日総務省告示第 81 号 (アマチュア局の無線設備の占有周波数帯幅の許容値を定める件) によって、上記告示が廃止されました。その内容は、おおむね周波数等の一括表示に関するものですが、特例による占有周波数帯の個別指定の項目が消えました。新告示は令和 5 年 9 月 25 日から施行されますので、以後は特例指定の道が容易ではなくなる懸念が残ります。

免許権者は新技術や世界のアマチュア無線の動向に留意して、過去に認められてきた占有周波数帯の個別指定の方向性を保って、柔軟な対応をして欲しいところです。

以上