

SPRESENSE™ でいろんな
LPWAを使ってみよう！

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- Spresenseからのデータを可視化してみよう！
- **SpresenseでいろんなLPWAを使ってみよう！**



LPWAとLTE (Cat-M1) について

ライセンスバンド		アンライセンスバンド				
基地局						
スター型					マルチホップ	
移動に強い	移動に弱い		移動に強い	移動に弱い		
LTE-M	NB-IoT	Sigfox	ELTRES	LoRa/ LoRaWAN	Zeta	WiSUN
1Mbps	150kbps	800kbps	数bps	250kbps	50kbps	150kbps
携帯エリア	20km	50km	100km以上	1~10km	2~10km	500m
既存基地局 利用。	既存基地局 利用。	1日の伝送量 制限あり。	移動に強く長 距離を飛ぶ。			ホームネット ワークなど。

前回のハンズオンは、**LTE cat-M1**で使ったものでした！

今回は、Spresenseにつながる様々な**LPWA**の通信の**Add-on**ボードを使ってみましょう！

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- Spresenseからのデータを可視化してみよう！
- **SpresenseでいろんなLPWAを使ってみよう！**
 - **Wi-SUN Add-onを使ってみよう！**
 - Private LoRa Add-onを使ってみよう！
 - Zeta Add-onを使ってみよう！



Wi-SUNとは？

Wi-SUNとは？

Wi-SUNは Wireless Smart Utility Network の略で、日本では特定小電力無線と呼ばれる920MHz帯で使用され、2.4GHzや5GHz帯を使用するWi-Fiと比べると、通信速度は遅いものの、通信距離は長く、障害物にも強くつながらやすく、しかも低消費電力という利点があります。

Wi-SUNの採用

Wi-SUNは日本を代表する電力会社のスマートメータ（次世代電力量計）と家庭内HEMS（Home Energy Management System）間のBluetooth通信に標準採用されています。



Wi-SUNプロファイル

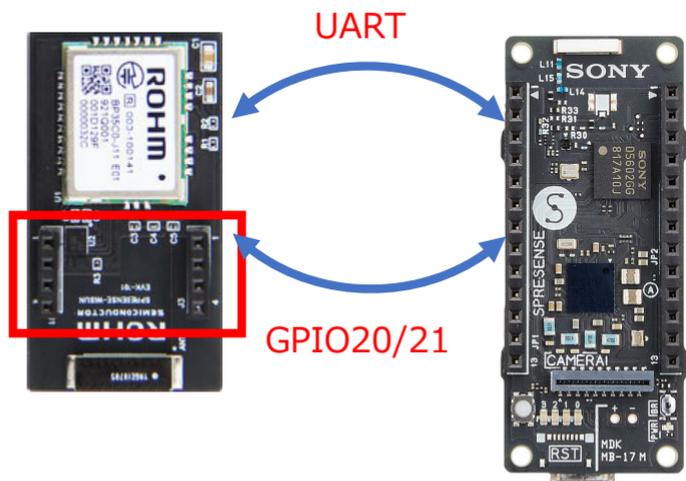
Wi-SUNはIEEE 802.15.4g規格に基づく通信で、IEEE 802.15.4gを最下層のプロトコルベースとしており、その上位層は、アプリケーションに応じてプロファイルが決められています。このWi-SUNプロファイルの作成や相互接続性を確保するための認証、およびその普及活動はWi-SUNアライアンスが行っています。

Application	HAN WG (HAN (ECHONET))	JUTA WG JUTA	FAN WG FAN	RLMM WG RLMM
Interface (Network, Transport layers, Authentication)	PANA UDP IPv6 6LowPAN 802.15.10	U-BUS Air	802.1x UDP RPL IPv6 6LowPAN	802.15.10
MAC Layer	CSMA	IEEE802.15.4/e		CSMA/RT/LE-SF
PHY Layer	IEEE802.15.4g based PHY			

【Wi-SUN プロファイル】

Wi-SUN Add-on ボードの紹介

ローム社提供のAdd-onボードです。UARTによる通信で制御されます。



詳しくは、ローム社のサイトをご確認ください。

<https://www.rohm.co.jp/support/spresense-add-on-board>

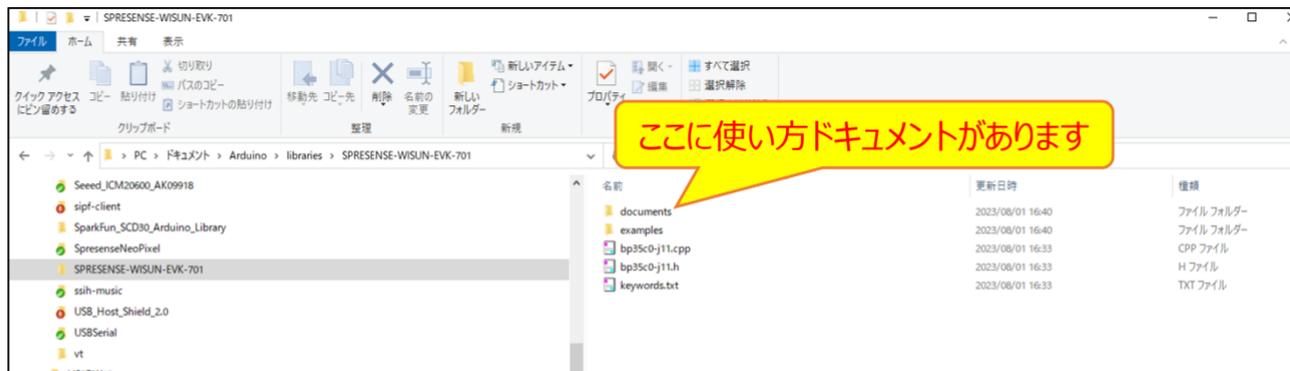
回路図などもあります。

Wi-SUNボード動作確認の流れ

Wi-SUNボードの使い方等の情報は、ロームさんのGithub上にあります。

<https://github.com/RohmSemiconductor/Arduino>

こちらを取り出した後、“SPRESENSE-WISUN-EVK-701”のディレクトリを、“Documents¥Arduino¥libraries”の下にコピーしてください。



ライブラリプロパティーがないため、ツールからのインストールができません。

Wi-SUNボード動作確認の流れ

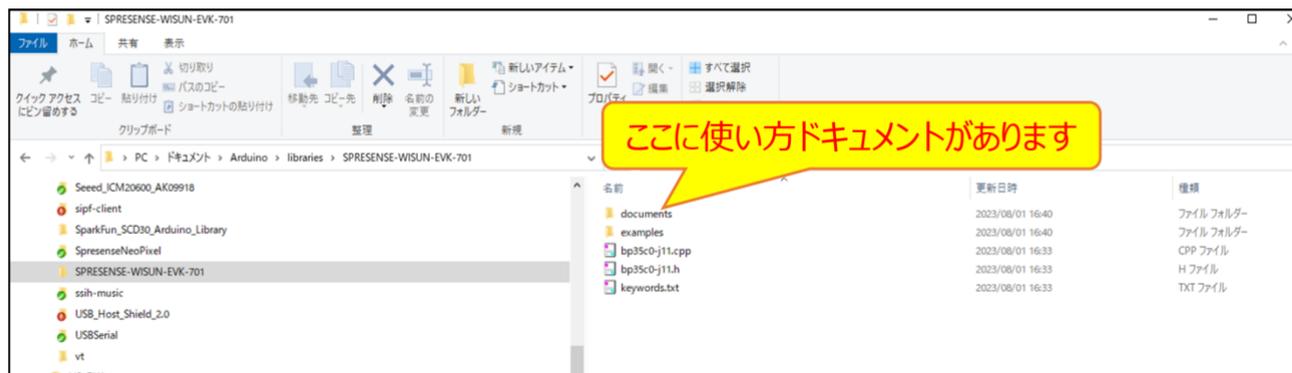
Wi-SUNボードの使い方等の情報は、ロームさんのGithub上にあります。

<https://github.com/RohmSemiconductor/Arduino>

が、こちらはあまり十分ではなかったため、自分の方で更新しました。こちらを使ってもらう方が良いかと思います。

<https://github.com/TomonobuHayakawa/Arduino-Rohm/>

こちらを取り出した後、“SPRESENSE-WISUN-EVK-701”のディレクトリを、“Documents¥Arduino¥libraries”の下にコピーか、ライブラリの **“ライブラリのインクルード->Zip形式でのライブラリのインクルード”** してください。



Wi-SUNボード動作確認の流れ

サンプルソフトウェア システム概要



SPRESENSE-WiSUN-EVK-701を子機としてUSB dongle BP35C2に対して定期的に無線データの送信を行います。

Wi-SUN通信を行うための設定は親機側はPC等で実施が必要であり、子機側はSPRESENSEで実施しています。

今回は親機側はターミナルソフト“Tera term”で動作確認をしており、子機側はArduino IDEを使用して動作確認しています。Arduino IDEでSPRESENSEを動作させるには“[Spresense Arduino Libraryの使い方](#)”を参考にしてください。

次ページより接続手順や内容について(親機側の設定例、子機側の設定内容等)説明していきます。

想定する使い方次第ですが、ここでは、Spresense をエンドデバイスとして、PCにWi-SUN dongle経由でデータをuploadするサンプルになっています。

実際には、Spresense を親機にすることもできます！

PC dongleの後継機はこちらになります。

<https://www.ratocsystems.com/products/wisun/usb-wisun/rs-wsuha/>

Wi-SUNボード動作確認の流れ

PC dongleを使うのにTeratermが必要です。

こちら (<https://ttssh2.osdn.jp/>) からダウンロード&インストールしてください。

PC dongleのドライバは、Windowsが自動的にインストールしてくれると思いますが、もし、しばらくしてもポートが見えない場合は、こちら ([Drivers - FTDI \(ftdichip.com\)](http://ftdichip.com)) からダウンロード&インストールしてください。

USB dongleの使い方は、こちらを参照してください。

[RohmSemiconductor/Arduino/SPRESENSE-WISUN-EVK-701/documents/SPRESENSE-WiSUN-EVK-701 サンプルソフトウェア説明書 ROHC.pdf](http://RohmSemiconductor.com/Arduino/SPRESENSE-WISUN-EVK-701/documents/SPRESENSE-WiSUN-EVK-701_SampleSoftwareManual_ROHC.pdf)

BP35C2の設定(Teratermで実施するコマンド)



以下のコマンドを順番に入力し、初期設定を完了させます。各コマンドの応答として“OK”が送られてきたらそのコマンドは正しく実行されています。

※半角スペースを“ ”で記載していますので、コマンドをコピーする際は注意してください。

- ①SKSREG_SF0_1
内容：動作モードを設定します。PANコーディネーターとして動作するように設定しています。
- ②SKSREG_S2_23
内容：使用するチャンネルを設定します。922.9MHzを使用しています。
- ③SKSREG_S3_5678
内容：PAN IDを設定します。“5678”を設定しています。
- ④SKSREG_SA9_1
内容：データの送受信を有効/無効を設定します。有効にしています。
- ⑤SKSTART
内容：PANA認証サーバーとして動作させます。
- ⑥SKSETHPWD_001D129F00009CB4_1111222233334444
<MACアドレス> <パスワード>
内容：指定したMACアドレスのデバイスに対し、パスワードを設定しPSKを生成します。
接続する子機のアドレスに合わせてMACアドレスおよびパスワードを設定してください。

BP35C2の設定は以上です。
尚、各コマンドの詳細については「BP35C0/BP35C2コマンドリファレンスDSE版」を参考にしてください。
資料の入手方法については[スタートガイド](#)を参考にしてください。

ドングルの起動

PCドングルを親機として起動します。TeraTerm上から以下のコマンドを順に入力します。

```
SKVER  
SKRESET  
SKSREG SF0 1  
SKSREG S2 23  
SKSREG S3 5678  
SKSREG SA9 1  
SKSTART  
SKSETHPWD 001D129F0000XXXX 1111222233334444
```

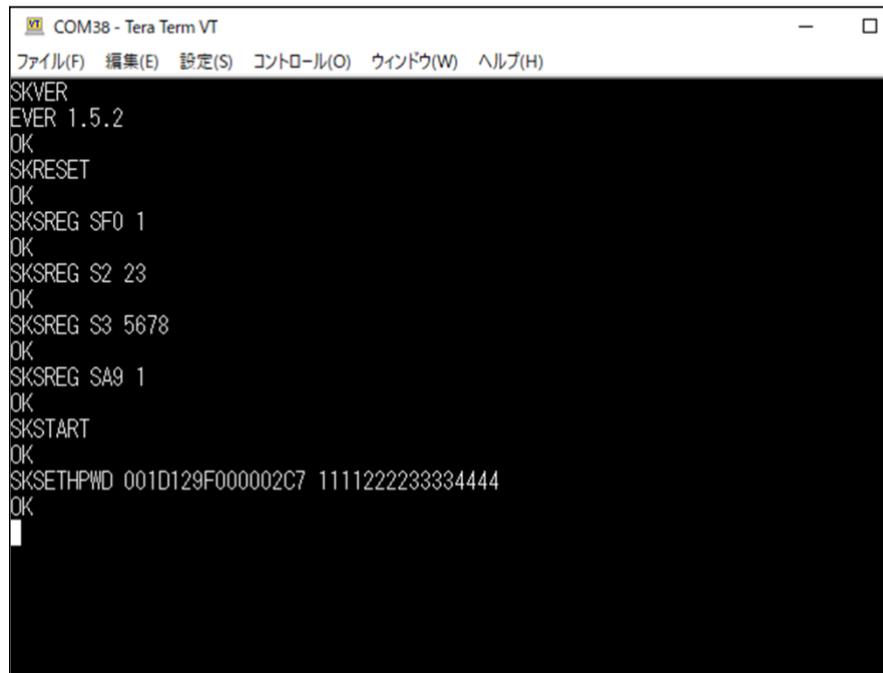
Add-onボードの上面に表記

現在持っているWi-SUN
Add-onボードのMACアドレス

これを実行すると子機からの通信待ち状態になります。

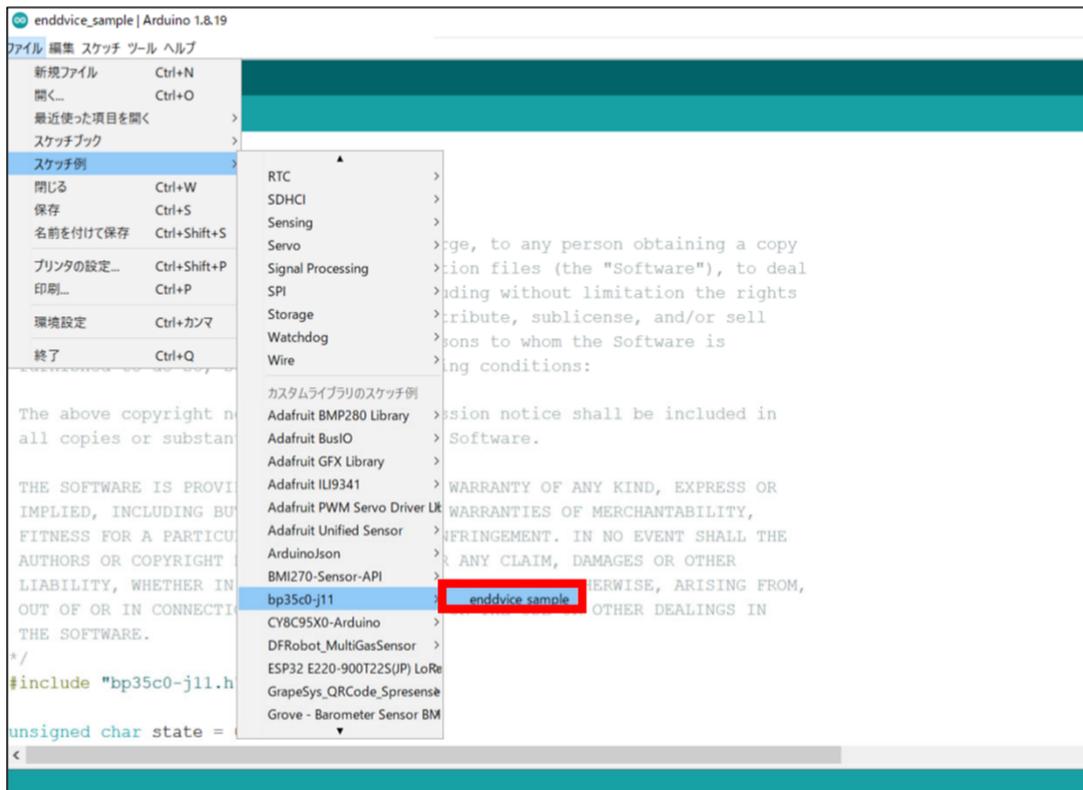
こちらのコマンドは、ここに用意しました。

https://github.com/TomonobuHayakawa/Arduino-Rohm/blob/master/SPRESENSE-WISUN-EVK-701/examples/enddvice_sample/SKSTACK-IP/PAN_Coordinator.txt



```
COM38 - Tera Term VT  
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)  
SKVER  
EVER 1.5.2  
OK  
SKRESET  
OK  
SKSREG SF0 1  
OK  
SKSREG S2 23  
OK  
SKSREG S3 5678  
OK  
SKSREG SA9 1  
OK  
SKSTART  
OK  
SKSETHPWD 001D129F000002C7 1111222233334444  
OK
```

enddevice_sample を動かしてみよう！



WiSUNライブラリは、ライブラリ名は、デバイス名 (bp35c0-j11) にしています。

このWiSUNライブラリの中に **enddevice_sample** というサンプルを作成しました！

こちらを動かしてみましよう！

このサンプルは、エンドデバイスとしての動作を行うサンプルになります。

enddvice_sample のコード解説

```

enddvice_sample | Arduino 1.8.19
ファイル 編集 スケッチ ツール ヘルプ

enddvice_sample
#include "bp35c0-j11.h"

unsigned const char password[16] = { '1', '1', '1', '1', '2', '2', '2', '2', '3', '3', '3', '3', '4', '4', '4', '4' }; // PANA認証時のパスワード
unsigned const char pair_id[8] = { 0x00, 0x1D, 0x12, 0x91, 0x00, 0x01, 0x68, 0xBD }; // 接続先MACアドレス
unsigned const char mac_adr[16] = { 0xFE, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x1D, 0x12, 0x91, 0x00, 0x01, 0x68, 0xBD }; // 接続先IPv6アドレス
unsigned const char my_port[2] = { 0x01, 0x23 }; // オープンするUDPポート
unsigned const char dist_port[2] = { 0x0E, 0x1A }; // 送信先UDPポート

BP35C0J11 bp35c0j11;
    
```

ヘッダーをインクルード

このデバイスのパスワード

相手のデバイスのMACアドレス

自分のIPV6ポート

相手のデバイスのIPV6アドレス

相手のデバイスのIPV6ポート

相手デバイス（Wi-SUNドングル）のMACアドレス、IPV6アドレス、ポートは、以下で調べられます。

```

4.2. SKINFO
現在の主要な通信設定値を表示します。

コメント例：
IPv6アドレス= FE80:0000:0000:0000:021D:1290:1234:5678
MACアドレス=001D129012345678
ch=0x21 (33ch)
PAN ID = 0x8888
アクティブ MAC 画=B 画
で設定の場合
SKINFO
Response
EINFO FE80:0000:0000:0000:021D:1290:1234:5678 001D129012345678
21 8888 0
OK
    
```

enddvice_sample のコード解説

The image shows a screenshot of the Arduino IDE interface with the file 'enddvice_sample'. The code is as follows:

```
BP35C0J11 bp35c0j11;

void setup() {
  if(!bp35c0j11.begin()){
    puts("Device error");
    exit(1);
  }

  if(!bp35c0j11.init(END_DEVICE, SLEEP_DISABLE, 0x05, 0x00)){
    puts("Init error");
    exit(1);
  }

  if(!bp35c0j11.set_auth(password){
    puts("set_auth error");
    exit(1);
  }

  if(!bp35c0j11.start_han(pair_id){
    puts("start_han error");
    exit(1);
  }

  if(!bp35c0j11.start_udp(mac_adr, my_port, dist_port)){
    puts("start_udp error");
    exit(1);
  }
}
```

Annotations in yellow callout boxes point to specific lines of code:

- インスタンス生成** (Instance Generation) points to the line: `BP35C0J11 bp35c0j11;`
- HW起動** (HW Start) points to the `begin()` call in the first `if` block.
- HW初期化** (HW Initialization) points to the `init()` call in the second `if` block.
- アカウント情報の設定** (Account Information Setting) points to the `set_auth()` call in the third `if` block.
- HANネットワークへの接続** (Connection to HAN Network) points to the `start_han()` call in the fourth `if` block.
- UDPによるデータ送信ポートの接続** (Connection of Data Transmission Port by UDP) points to the `start_udp()` call in the fifth `if` block.

enddvice_sample のコード解説

```
enddvice_sample | Arduino 1.8.19
ファイル 編集 スケッチ ツール ヘルプ
enddvice_sample
void loop() {
    static unsigned char data[MAX_WISUN_DATA_SIZE] = { 'T', 'E', 'S', 'T', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'};
    if(!bp35c0j11.send_data(data)){
        puts("start_auth error");
    }
    for(int i=0; i<sizeof(data); i++){
        data[i] = data[(i+1)%sizeof(data)];
    }
    sleep(5);
}
```

enddvce_sample の結果

このようにSpresenseから dongleヘデータの送信を行えました！
皆さんも、是非、お試しください！

```
COM38 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
SKRESET
OK
SKSREG SF0 1
OK
SKSREG S2 23
OK
SKSREG S3 5678
OK
SKSREG SA9 1
OK
SKSTART
OK
SKSETHPID 001D129F000002C7 111122223334444
OK
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 TESTABCDEFGHIJKL
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 ESTABCDEFGHIJKLE
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 STABCDEFGHIJKLES
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 TABCFGHIJKLEST
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 ABCDEFGHIJKLESTA
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 BCDEFGHIJKLESTAB
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 CDEFGHIJKLESTABC
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 DEFGHIJKLESTABCD
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 TESTABCDEFGHIJKL
ERXUDP FE80:0000:0000:0000:021D:129F:0000:02C7 FE80:0000:0000:0000:021D:1291:0001:68BD 0123 0E1A 001D129F000002C7 0 1 0010 ESTABCDEFGHIJKLE
```

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- Spresenseからのデータを可視化してみよう！
- **SpresenseでいろんなLPWAを使ってみよう！**
 - Wi-SUN Add-onを使ってみよう！
 - **Private LoRa Add-onを使ってみよう！**
 - Zeta Add-onを使ってみよう！



Private LoRaとは？

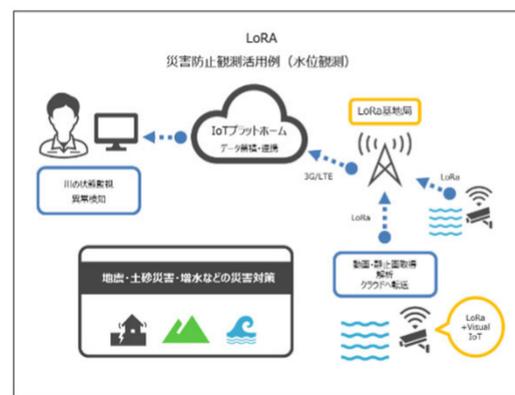
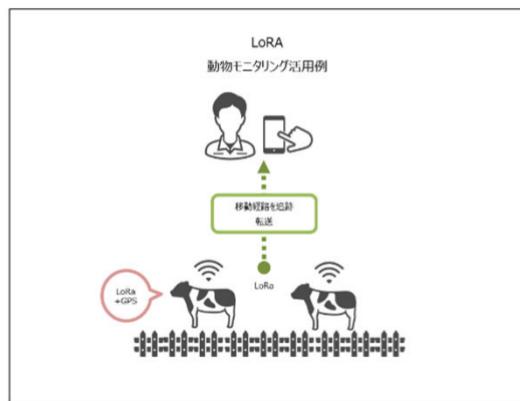
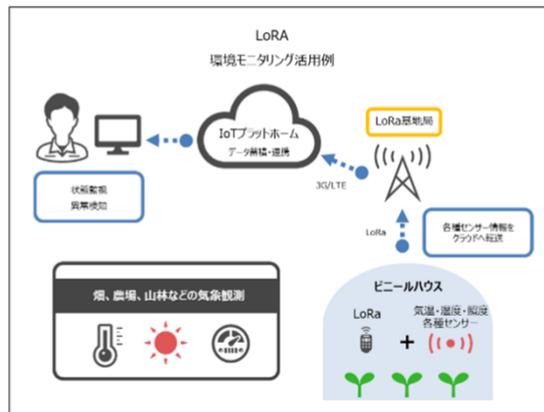
LoRa変調の特徴

LoRa通信は、920MHz帯での独自のスペクトラム拡散方式で、遠く離れた距離でも、減衰した信号から送信信号を取得でき、ノイズなどにも非常に強い耐性を持つ低消費電力通信です。

Private LoRaとは

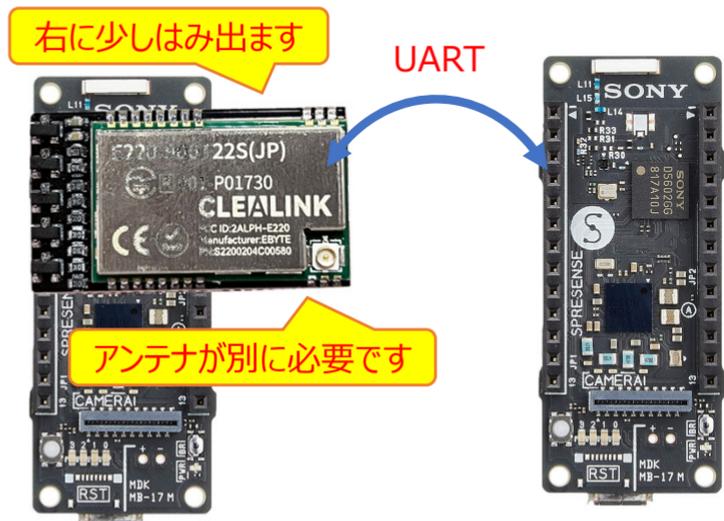
Private LoRaは、デバイス間の通信プロトコルや制御を機器ごとに独自の定めた方式で、通信料金がかからないのも魅力の1つです。

Private LoRaの活用例



LoRa Add-on ボードの紹介

クレアリンク社提供のAdd-onボードです。UARTによる通信で制御されます。



詳しくは、クレアリンク社のサイトをご確認ください。

<https://dragon-torch.tech/cat-components/extension-boards/dth-sslr/>

回路図などもあります。

この基板のモジュールでは、見通し距離で最大30km強、郊外市街地で4km前後、直径6～8kmの円域(当社実測値)を1つのアンテナ基地局でカバーできる特徴を持ちます。

LoRaボードドライバについて

LoRa Add-onクレアリンク社からこちらからドライバが提供されています。

<https://dragon-torch.tech/cat-components/extension-boards/dth-sslr/>

仕様

製品名	DTH-SSLR
RFモジュール	LoRa通信モジュール (E220-900T225 (JP)) 通信モジュールの仕様はこちら>>
対応ボード	SPRESENSE™メインボード (CXD5602PWBMAIN1)
インターフェース	UART モード制御/IO (信号レベルは3.3V TTL)
電源電圧	動作電圧 3.3V ~ 5.5V 推奨電圧範囲 3.5V ~ 5.0V
動作温度範囲	-45°C ~ +85°C
外形寸法	18mm x 33mm

インターフェース・サイズ



1. E220-900T225 (JP) LoRaモジュール 2. JPK u.FL端子

ダウンロード・サポート

■ ソニー SPRESENSE™用 ライブラリ [ZIP]

価格

ただし、一部修正したほうが良い部分などもあり、[こちらのライブラリ](#)を使ってもらえればと思います。

https://github.com/TomonobuHayakawa/Spresense-Playground/tree/master/libraries/spresense_e220900t22s_jp_lib

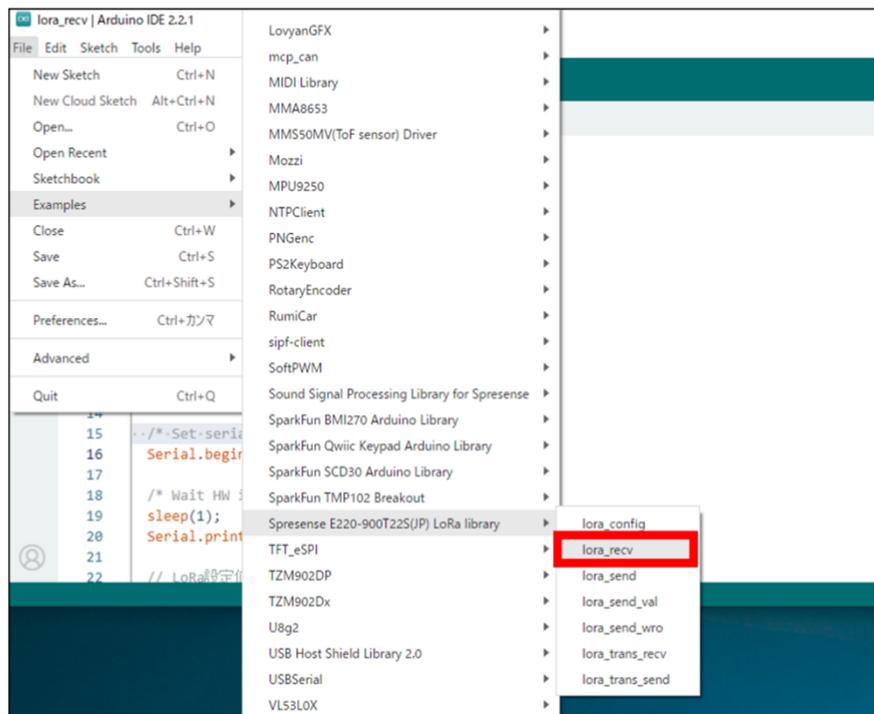
こちらのドライバをArduinoの

ライブラリのインクルード -> Zip形式のライブラリのインクルード

でインクルードしてください。

固定サイズモードでLoRa受信機として動かそう！

SpresenseをLoRa受信機（固定サイズモード）として動作させるサンプルはこちらになります。
（固定サイズの方が早く確実に送受信が可能ですが、自由度が少なくなります。）



Spresenseで、LoRaを動作させる場合、受信機として動作させるSpresenseを1台と、

送信機として動作させるSpresense 1台を1組として動作をさせます。

lora_recv のコード解説

```
lora_recv | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_recv.ino
1 #include <Arduino.h>
2 #include <vector>
3 #include "spresense_e220900t22s_jp_lib.h"
4
5 CLoRa lora;
6 struct RecvFrameE220900T22SJP_t data;
7
8 void setup() {
9     // put your setup code here, to run once:
10    pinMode(LED0, OUTPUT);
11    pinMode(LED1, OUTPUT);
12    pinMode(LED2, OUTPUT);
13    pinMode(LED3, OUTPUT);
14
15    /*.Set serial baudrate.*/
16    Serial.begin(115200);
17
18    /* Wait HW initialization done. */
19    sleep(1);
20    Serial.printf("program start\n");
21
```

インスタンス

ライブラリのインクルード

読み出し用データ構造体

lora_recv のコード解説

```
lora_recv | Arduino IDE 2.2.1
File Edit Sketch Tools Help
ψ Spresense
lora_recv.ino
21
22 // LoRa設定値
23 struct LoRaConfigItem_t config = {
24     0x0000, // own_address 0
25     0b011, // baud_rate 9600 bps
26     0b10000, // air_data_rate SF:9 BW:125
27     SUBPACKET_200_BITS,
28     0b1, // rssi_ambient_noise_flag 有効
29     0b0, // transmission_pause_flag 有効
30     0b01, // transmitting_power 13 dBm
31     0x00, // own_channel 0
32     0b1, // rssi_byte_flag 有効
33     FIX_SIZE_MODE, // transmission_method_type 固定送信モード
34     0b0, // lbt_flag 有効
35     0b011, // wor_cycle 2000 ms
36     0x0000, // encryption_key 0
37     0x0000, // target_address 0
38     0x00 // target_channel 0
39 };
40
41 // E220-900T225(JP)へのLoRa初期設定
42 if (lora.InitLoRaModule(config)) {
43     SerialMon.printf("init error\n");
44 }
45 } else {
46     Serial.printf("init ok\n");
47 }
48 }
```

LoRaの各設定

設定値をHWに設定

lora_recv のコード解説

```
lora_recv | Arduino IDE 2.2.1
File Edit Sketch Tools Help
SpreSense
lora_recv.ino
49 // ノーマルモード(M0=0,M1=0)へ移行する
50 SerialMon.printf("switch to normal mode\n");
51 lora.SwitchToNormalMode();
52
53 }
54
55 void loop() {
56   if (lora.ReceiveFrame(&data) == 0) {
57     SerialMon.printf("recv data:\n");
58     for (int i = 0; i < data.recv_data_len; i++)
59       SerialMon.printf("%c", data.recv_data[i]);
60   }
61   SerialMon.printf("\n");
62   SerialMon.printf("hex dump:\n");
63   for (int i = 0; i < data.recv_data_len; i++) {
64     SerialMon.printf("%02x ", data.recv_data[i]);
65   }
66   SerialMon.printf("\n");
67   SerialMon.printf("RSSI: %d dBm\n", data.rssi);
68   SerialMon.printf("\n");
69
70   SerialMon.flush();
71 }
72
73 delay(1);
74
75 }
76
```

設定モードから通常モードへ変更

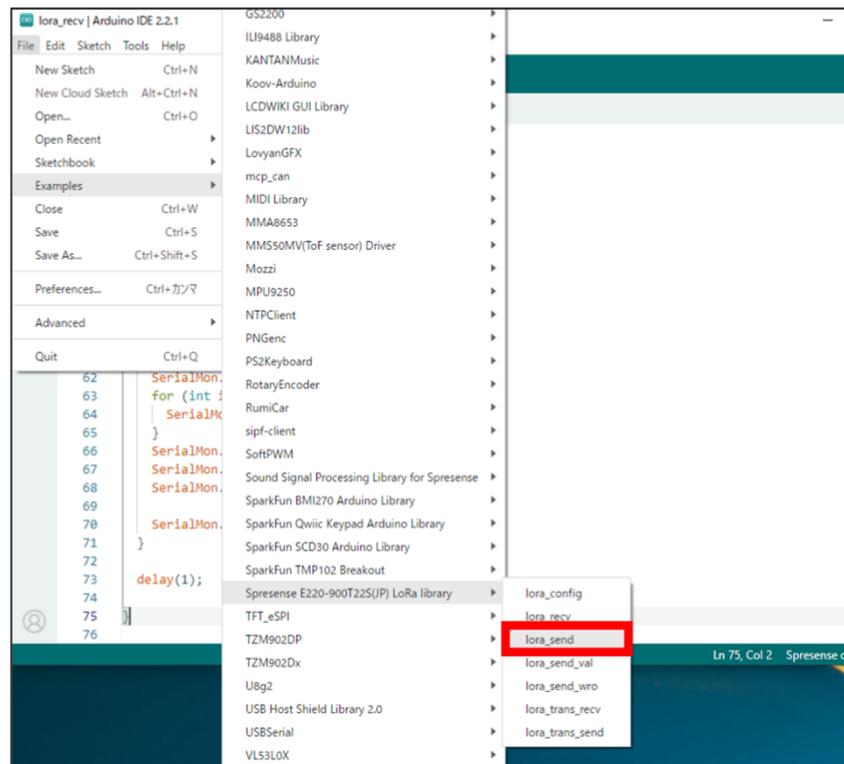
受信したデータがある
場合サイズを返す

データサイズ分表示

電波強度を表示

固定サイズモードでLoRa送信機として動かそう！

SpresenseをLoRa送信機（固定サイズモード）として動作させるサンプルはこちらになります。



Spresenseで、LoRaを動作させる場合、受信機として動作させるSpresenseを1台と、

送信機として動作させるSpresense 1台を1組として動作をさせます。

lora_send のコード解説



The image shows a screenshot of the Arduino IDE 2.2.1 interface. The main window displays the code for a sketch named 'lora_send.ino'. The code is as follows:

```
1 #include <Arduino.h>
2 #include <vector>
3 #include "spresense_e220900t22s_jp_lib.h"
4
5 /* Send data size is lora subblock size - lora header size. */
6 #define SEND_DATA_SIZE (29) // 32 - 3
7 //#define SEND_DATA_SIZE (61) // 64 - 3
8 //#define SEND_DATA_SIZE (125) // 128 - 3
9 //#define SEND_DATA_SIZE (197) // 200 - 3
10
11 CLoRa lora;
12
```

Three yellow callout boxes with red borders point to specific lines of code:

- Line 3: `#include "spresense_e220900t22s_jp_lib.h"` is annotated with "ライブラリのインクルード" (Library include).
- Lines 6-9: The `#define SEND_DATA_SIZE` lines are annotated with "固定長送信でのデータ (ペイロード) サイズ" (Data (payload) size for fixed-length transmission).
- Line 11: `CLoRa lora;` is annotated with "インスタンス" (Instance).

lora_send のコード解説

```
lora_send | Arduino IDE 2.2.1
File Edit Sketch Tools Help
SpreSense
lora_send.ino
10
11 CLoRa lora;
12
13 // LoRa設定値
14 struct LoRaConfigItem_t config = {
15     0x0000, // own_address 0
16     0b011, // baud_rate 9600 bps
17     0b10000, // air_data_rate SF:9 BW:125
18
19     #if (SEND_DATA_SIZE == 29)
20     | SUBPACKET_32_BITS,
21     #elif (SEND_DATA_SIZE == 61)
22     | SUBPACKET_64_BITS,
23     #elif (SEND_DATA_SIZE == 125)
24     | SUBPACKET_128_BITS,
25     #else
26     | SUBPACKET_200_BITS,
27     #endif
28
29     0b1, // rssi_ambient_noise_flag 有効
30     0b0, // transmission_pause_flag 有効
31     0b01, // transmitting_power 13 dBm
32     0x00, // own_channel 0
33     0b1, // rssi_byte_flag 有効
34     FIX_SIZE_MODE, // transmission_method_type 固定送信モード
35     0b0, // lbt_flag 有効
36     0b011, // wor_cycle 2000 ms
37     0x0000, // encryption_key 0
38     0x0000, // target_address 0
39     0x00 // target_channel 0
40 };
```

LoRaの各設定

lora_send のコード解説

```
lora_send.ino
39
40 void setup() {
41 // put your setup code here, to run once:
42 pinMode(LED0, OUTPUT);
43 pinMode(LED1, OUTPUT);
44 pinMode(LED2, OUTPUT);
45 pinMode(LED3, OUTPUT);
46
47 /* Set serial baudrate. */
48 Serial.begin(115200);
49
50 /* Wait HW initialization done. */
51 sleep(1);
52 Serial.printf("program start\n");
53
54
55 // E220-900T22S(JP)へのLoRa初期設定
56 if (lora.InitLoRaModule(config)) {
57     SerialMon.printf("init error\n");
58     return;
59 } else {
60     Serial.printf("init ok\n");
61 }
62
63 // ノーマルモード(M0=0,M1=0)へ移行する
64 SerialMon.printf("switch to normal mode\n");
65 lora.SwitchToNormalMode();
66 }
67
```

// E220-900T22S(JP)へのLoRa初期設定
if (lora.InitLoRaModule(config)) {
 SerialMon.printf("init error\n");
 return;
} else {
 Serial.printf("init ok\n");
}

設定値をHWに設定

// ノーマルモード(M0=0,M1=0)へ移行する
SerialMon.printf("switch to normal mode\n");
lora.SwitchToNormalMode();

設定モードから通常モードへ変更

lora_send のコード解説

```
lora_send | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_send.ino
66 }
67
68 void loop() {
69   char msg[SEND_DATA_SIZE] = { 0 };
70   static char dmy_data = '0';
71
72   for (int i = 0; i < SEND_DATA_SIZE; i++) msg[i] = dmy_data + (i % 40);
73   dmy_data > 'S' ? dmy_data = '0' : dmy_data++;
74
75   if (lora.SendFrame(config, (uint8_t *)msg, sizeof(msg)) == 0) {
76     SerialMon.printf("send succeeded.\n");
77   } else {
78     SerialMon.printf("send failed.\n");
79     SerialMon.printf("\n");
80   }
81   SerialMon.flush();
82
83   sleep(10);
84 }
85
86
```

ダミーメッセージを生成

固定長モードでのデータ送信

Configの中のパラメータからヘッダーを生成

ヘッダーにあるチャンネル、アドレスで送信

送受信結果

双方のSpresenseからの送受信結果はこのようになります。

```
COM15
program start
switch to configuration mode
# Command Request
0xc0 0x00 0x08 0x00 0x00 0x70 0xa1 0x00 0xc3 0x00 0x00
# Command Response
0xc1 0x00 0x08 0x00 0x00 0x70 0xa1 0x00 0xc3 0x00 0x00
init ok
switch to normal mode
000000123456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZVW0123456789;.<=>?@ABCDEF
send succeeded.

000000123456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZVWX123456789;.<=>?@ABCDEF
send succeeded.

00000023456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZVWXY23456789;.<=>?@ABCDEF
send succeeded.

0000003456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZVWXY3456789;.<=>?@ABCDEF
send succeeded.

000000456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZ[456789;.<=>?@ABCDEF
send succeeded.

00000056789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZ[56789;.<=>?@ABCDEF
send succeeded.
```

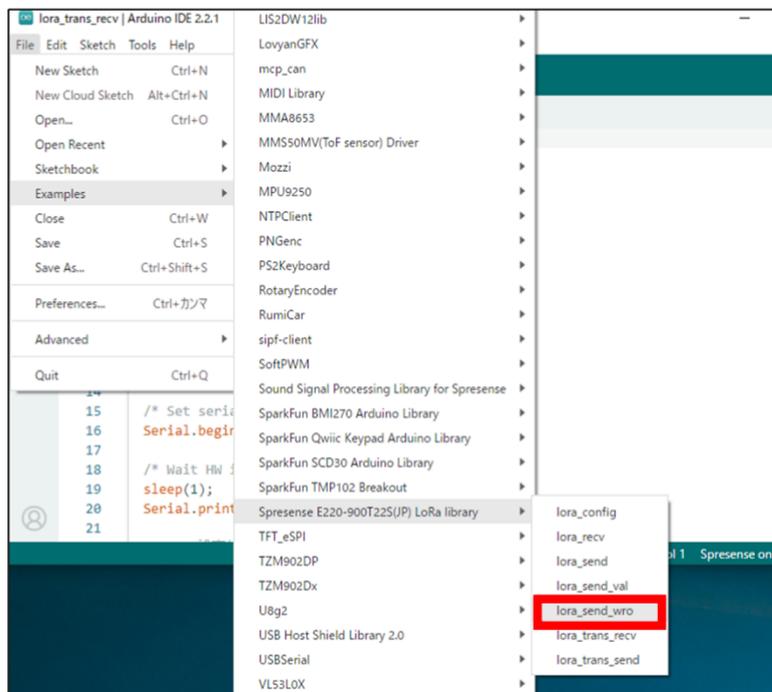
送信結果

```
COM35
09:16:36.598 -> program start
09:16:36.598 -> switch to configuration mode
09:16:36.831 -> # Command Request
09:16:36.831 -> 0xc0 0x00 0x08 0x00 0x00 0x70 0xa1 0x00 0xc3 0x00 0x00
09:16:36.924 -> # Command Response
09:16:36.924 -> 0xc1 0x00 0x08 0x00 0x00 0x70 0xa1 0x00 0xc3 0x00 0x00
09:16:36.924 -> init ok
09:16:36.924 -> switch to normal mode
09:16:43.645 -> recv data:
09:16:43.645 -> 0123456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZVW0123456789;.<=>?@ABC
09:16:43.645 -> hex dump:
09:16:43.645 -> 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 4
09:16:43.691 -> RSSI: -24 dBm
09:16:43.691 ->
09:17:04.020 -> recv data:
09:17:04.020 -> 23456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZVWXY23456789;.<=>?@ABCDEF
09:17:04.020 -> hex dump:
09:17:04.020 -> 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4
09:17:04.020 -> RSSI: -25 dBm
09:17:04.020 ->
09:17:24.356 -> recv data:
09:17:24.356 -> 456789;.<=>?@ABCDEFGHIJKLMN0PQRSTUVWXYZ[456789;.<=>?@ABCDEF
09:17:24.356 -> hex dump:
09:17:24.356 -> 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4
09:17:24.356 -> RSSI: -25 dBm
09:17:24.356 ->
```

受信結果

WROモードでの送信

送信側のSpsenseが送信しか行わない場合は、ノーマルモードではなくWROモードでの送信を行うことで、より電力を抑えることが可能です。

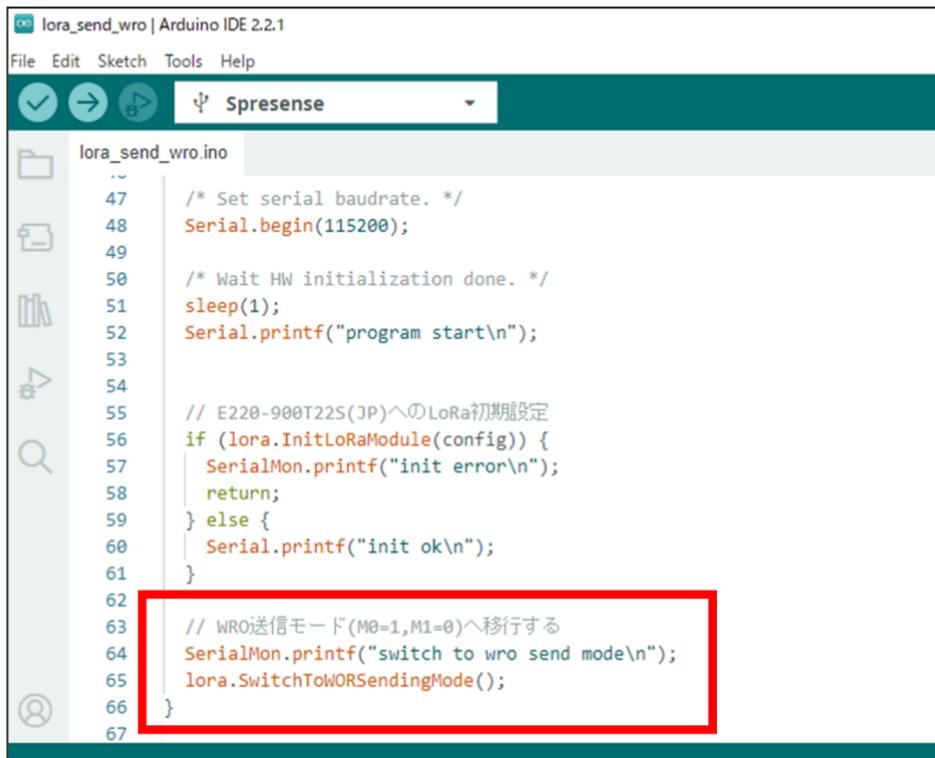


本来、受信側もWROモードでの待ち受けを行うことで、送受信で非常に省電力での動作が可能になるのですが、現時点で、cold sleepからの起動後の処理がうまくいっていないので、未対応です。

受信はWROモードができれば、サンプルを追加する予定です。

lora_send_wro のコード解説

ノーマルモードとの違いはここだけです。

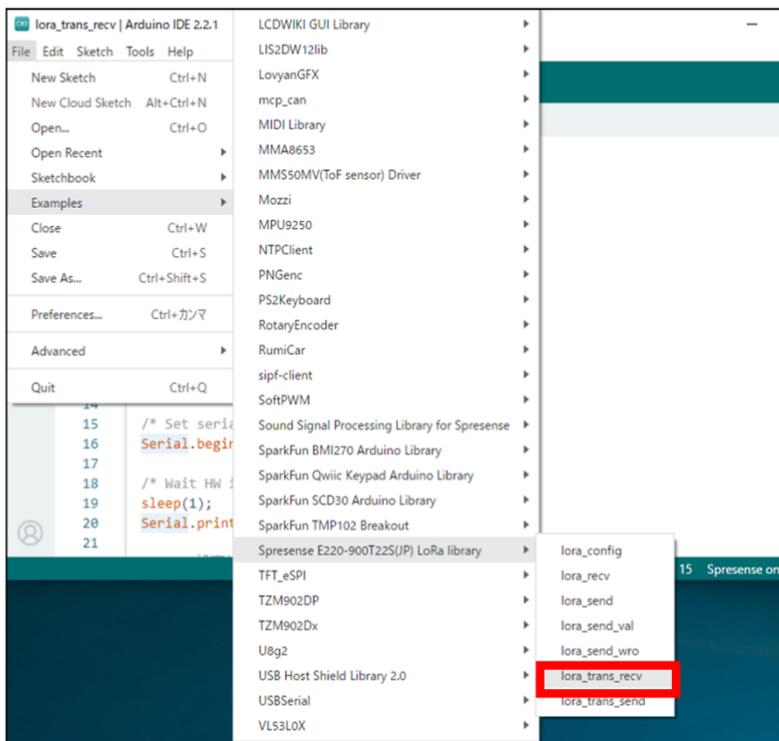


```
lora_send_wro | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_send_wro.ino
47  /* Set serial baudrate. */
48  Serial.begin(115200);
49
50  /* Wait HW initialization done. */
51  sleep(1);
52  Serial.printf("program start\n");
53
54
55  // E220-900T22S(JP)へのLoRa初期設定
56  if (lora.InitLoRaModule(config)) {
57  | SerialMon.printf("init error\n");
58  | return;
59  } else {
60  | Serial.printf("init ok\n");
61  | }
62
63  // WRO送信モード(M0=1,M1=0)へ移行する
64  SerialMon.printf("switch to wro send mode\n");
65  lora.SwitchToWORSendingMode();
66  }
67
```

送受信結果は、ノーマルモードと一緒になります。

トランスペアレントモードでLoRa受信機として動かそう！

SpresenseをLoRa受信機（トランスペアレントモード）として動作させるサンプルはこちらになります。
（固定サイズに比べ受信のタイミング等が不確定ですが、データサイズは自由です。



Spresenseで、LoRaを動作させる場合、受信機として動作させるSpresenseを1台と、

送信機として動作させるSpresense 1台を1組として動作をさせます。

lora_trans_send のコード解説

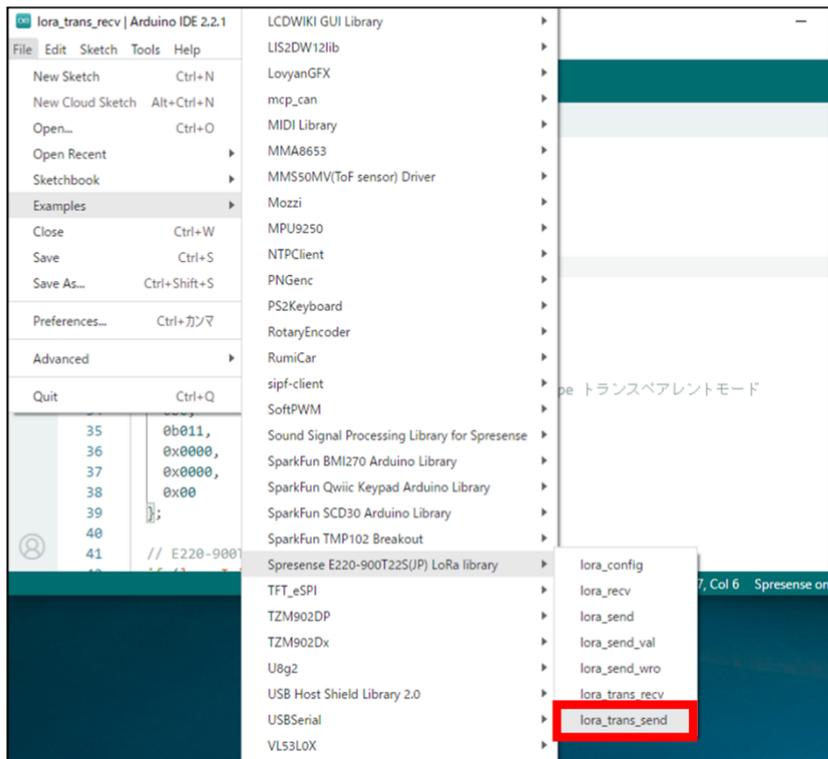
固定サイズモードとの違いはここだけです。

```
lora_trans_recv | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_trans_recv.ino
21
22 // LoRa設定値
23 struct LoRaConfigItem_t config = {
24     0x0000, // own_address 0
25     0b011, // baud_rate 9600 bps
26     0b10000, // air_data_rate SF:9 BW:125
27     0b00, // subpacket_size 200
28     0b1, // rssi_ambient_noise_flag 有効
29     0b0, // transmission_pause_flag 有効
30     0b01, // transmitting_power 13 dBm
31     0x00, // own_channel 0
32     0b1 // rssi_byte_flag 有効
33     TRANSPARENT_MODE, // transmission_method_type トランスペアレントモード
34     0b0, // lbt_flag 有効
35     0b011, // wor_cycle 2000 ms
36     0x0000, // encryption_key 0
37     0x0000, // target_address 0
38     0x00 // target_channel 0
39 };
40
```

トランスペアレントモードにしないと、
トランスペアレントモードでの送信
を受けられません

トランスペアレントモードでLoRa送信機として動かそう！

SpresenseをLoRa送信機（トランスペアレントモード）として動作させるサンプルはこちらになります。



Spresenseで、LoRaを動作させる場合、受信機として動作させるSpresenseを1台と、

送信機として動作させるSpresense 1台を1組として動作をさせます。

lora_trans_send のコード解説

固定サイズモードとの違いを示します。

```
lora_trans_send | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_trans_send.ino
9
10 // LoRa設定値
11 struct LoRaConfigItem_t config = {
12     0xFFFF, // own_address
13     0b011, // baud_rate 9600 bps
14     0b10000, // air_data_rate SF:9 BW:125
15     0b11, // subpacket_size 32
16     0b1, // rssi_ambient_noise_flag 有効
17     0b0, // transmission_pause_flag 有効
18     0b01, // transmitting_power 13 dBm
19     0x00, // own_channel 0
20     0b1, // rssi_byte_flag 有効
21     TRANSPARENT_MODE, // transmission_method_type トランスペアレントモード
22     0b0, // lbt_flag 有効
23     0b011, // wor_cycle 2000 ms
24     0x0000, // encryption_key 0
25     0x0000, // target_address 0
26     0x00 // target_channel 0
27 };
```

FFFFを設定することで同一チャネルのすべてのアドレスに送信

トランスペアレントモードに設定

lora_trans_send のコード解説

固定サイズモードとの違いを示します。

```
lora_trans_send | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_trans_send.ino
54 /* Send data size */
55 #define MAX_SEND_DATA_SIZE (20)
56
57 void loop() {
58   char msg[MAX_SEND_DATA_SIZE] = { 0 };
59   static char dmy_data = '0';
60   static int send_data_size = 10;
61
62   for (int i = 0; i < send_data_size; i++) msg[i] = dmy_data + (i % 40);
63   dmy_data > 'S' ? dmy_data = '0' : dmy_data++;
64   (send_data_size >= MAX_SEND_DATA_SIZE) ? send_data_size = 10 : send_data_size++;
65
66   if (lora.SendFrame((uint8_t *)msg, send_data_size) == 0) {
67     SerialMon.printf("send succeeded.\n");
68     SerialMon.printf("\n");
69   } else {
70     SerialMon.printf("send failed.\n");
71     SerialMon.printf("\n");
72   }
73   SerialMon.flush();
74
75   sleep(10);
76 }
77
```

サイズも変更されます

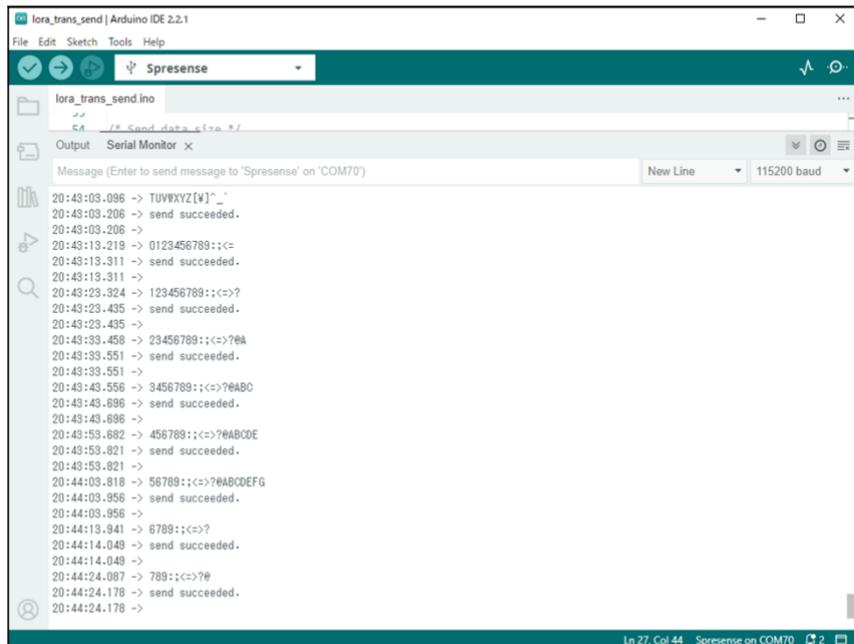
ダミーデータの生成

トランスペアレントモードでの送信

ヘッダーはなくbodyのみ

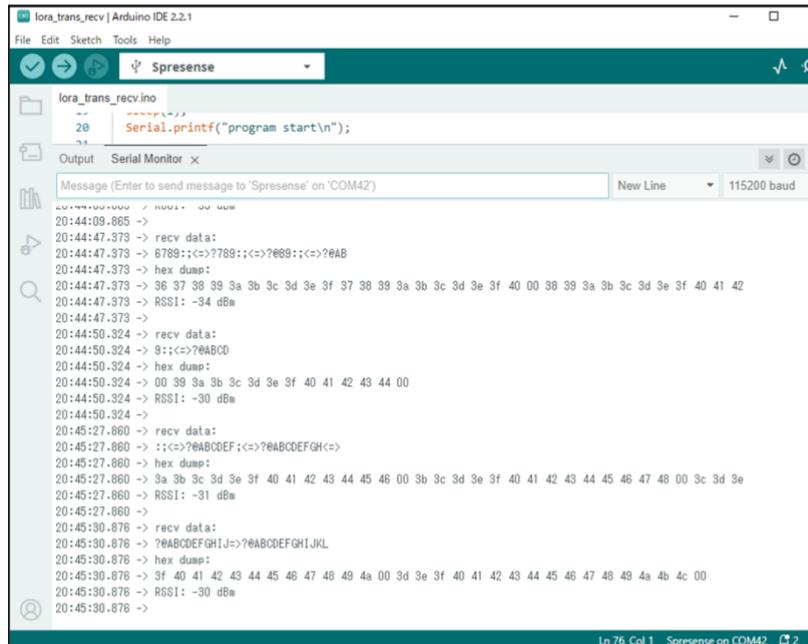
送受信結果

双方のSpresenseからの送受信結果はこのようになります。



```
lora_trans_send | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_trans_send.ino
/* Send data size */
Output Serial Monitor x
Message (Enter to send message to 'Spresense' on 'COM70') New Line 115200 baud
20:43:03.096 -> TUVWXYZ[!]~_
20:43:03.206 -> send succeeded.
20:43:03.206 ->
20:43:13.219 -> 0123456789::(<?)
20:43:13.311 -> send succeeded.
20:43:13.311 ->
20:43:23.324 -> 123456789::(<?)
20:43:23.435 -> send succeeded.
20:43:23.435 ->
20:43:33.458 -> 23456789::(<?)@A
20:43:33.551 -> send succeeded.
20:43:33.551 ->
20:43:43.556 -> 3456789::(<?)@ABC
20:43:43.696 -> send succeeded.
20:43:43.696 ->
20:43:53.682 -> 456789::(<?)@ABCDE
20:43:53.821 -> send succeeded.
20:43:53.821 ->
20:44:03.818 -> 56789::(<?)@BCDEFG
20:44:03.956 -> send succeeded.
20:44:03.956 ->
20:44:13.941 -> 6789::(<?)
20:44:14.049 -> send succeeded.
20:44:14.049 ->
20:44:24.067 -> 789::(<?)@E
20:44:24.178 -> send succeeded.
20:44:24.178 ->
Ln 27, Col 44 Spresense on COM70
```

送信結果



```
lora_trans_rcv | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense
lora_trans_rcv.ino
Serial.printf("program start\n");
Output Serial Monitor x
Message (Enter to send message to 'Spresense' on 'COM42') New Line 115200 baud
20:44:09.685 ->
20:44:47.373 -> rcv data:
20:44:47.373 -> 6789::(<?)789::(<?)@89::(<?)@AB
20:44:47.373 -> hex dump:
20:44:47.373 -> 36 37 38 39 3a 3b 3c 3d 3e 3f 37 38 39 3a 3b 3c 3d 3e 3f 40 00 38 39 3a 3b 3c 3d 3e 3f 40 41 42
20:44:47.373 -> RSSI: -34 dBm
20:44:47.373 ->
20:44:50.324 -> rcv data:
20:44:50.324 -> 9::(<?)@ABCD
20:44:50.324 -> hex dump:
20:44:50.324 -> 00 38 3c 3d 3e 3f 40 41 42 43 44 00
20:44:50.324 -> RSSI: -30 dBm
20:44:50.324 ->
20:45:27.660 -> rcv data:
20:45:27.660 -> ::(<?)@BCDEFGH<?
20:45:27.660 -> hex dump:
20:45:27.660 -> 3a 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 00 3b 3c 3d 3e 3f 40 41 42 43 44 45 46 47 48 00 3c 3d 3e
20:45:27.660 -> RSSI: -31 dBm
20:45:27.660 ->
20:45:30.676 -> rcv data:
20:45:30.676 -> ?@BCDEFGHIJ<?@BCDEFGHIJKL
20:45:30.676 -> hex dump:
20:45:30.676 -> 3f 40 41 42 43 44 45 46 47 48 49 4a 00 3d 3e 3f 40 41 42 43 44 45 46 47 48 49 4a 4b 4c 00
20:45:30.676 -> RSSI: -30 dBm
20:45:30.676 ->
Ln 76, Col 1 Spresense on COM42
```

受信結果

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- Spresenseからのデータを可視化してみよう！
- **SpresenseでいろんなLPWAを使ってみよう！**
 - Wi-SUN Add-onを使ってみよう！
 - Private LoRa Add-onを使ってみよう！
 - **Zeta Add-onを使ってみよう！**



Zeta Add-on ボードの紹介

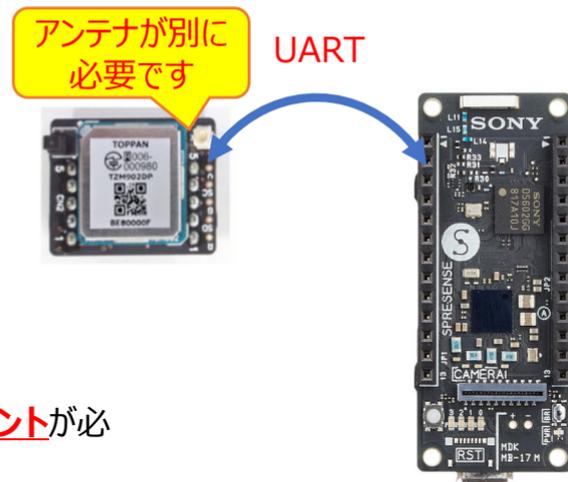
凸版社提供のAdd-onボードです。UARTによる通信で制御されます。

詳しくは、凸版印刷社のサイトをご確認ください。

<http://www.toptdc.com/product/zeta/>

実際にZetaデバイスを使用する場合、**Zetaサーバ**に接続する**Zetaアクセスポイント**が必要になります。そのため、そちらの準備も必要になります。

<http://zeta-factory.shop/shopbrand/ct86/>



Allianceメンバーであれば、貸し出し制度があり無料で評価が可能です。

<https://zeta-alliance.org/contents/startkit>

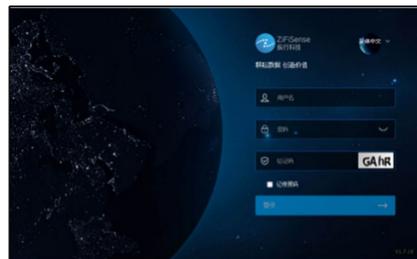


Zeta ネットワークを使用する流れ (AP設定)

まずは、アクセスポイントの設定とZetaネットワークの登録が必要です。
今回は、テクサー社のアクセスポイント (評価キット) を利用します。



アクセスポイントを説明書通りに接続し、電源を入れます。
テクサー社のデバイス管理サイトにアクセスすると、デバイスの稼働状況が見えます。



モジュール (Spresense)
が稼働していればここがON

デバイスID	デバイスSN	アラームブリス	アラーム時刻
MS	88800F04	再登録時通知多アラーム	2023-10-26 11:17:26
MS	88800F79	オフライン・アラーム	2023-07-01 17:10:00
MS	88800F76	オフライン・アラーム	2023-07-01 11:30:00

デバイス名	デバイスID	再登録日時	登録理由	最新の稼働情報
MS	88800F04	5	レポート	2023-10-26 11:17:49

※こちらはあくまで**デバイス管理のみ**で**Zetaサービスのサーバ**ではありません。

Zeta ネットワークを使用する流れ（デバイス登録）

Add-onボードについているモジュールIDを登録します。
これによりモジュールの稼働状況もわかるようになります。

このタブからモジュール登録

所属会社	デバイスID	デバイスの名前	プロトコルモード	MSタイプ	オンライン状態	アラームの状態	登録時刻	起動時刻	ファームウェアバージョン	タウントリンク時刻	バッテリー電圧	アップリンクRSSI	ダウンリンクRSSI	操作
Rental Kit 03	4F008BC5	ZETA-P	ZETA Modu	オンライン	なし	2021-03-16 10:24:45								
Rental Kit 03	4F008BD4	ZETA-P	ZETA Modu	オフライン	なし	2023-06-13 17:46:07	2023-06-13 17:46:11	2023-06-02 16:47:53	2023-06-13 17:52:01	3.35	-62	-54		
Rental Kit 03	Spresense1	ZETA-P	ZETA Modu	オンライン	なし	2023-06-30 09:22:39	2023-06-30 15:22:39	2023-06-30 17:00:09	2023-06-19 22:08:41	3.01	-36	-21		
Rental Kit 03	Spresense0	ZETA-P	ZETA Modu	オンライン	なし	2023-06-30 09:20:46	2023-06-30 09:20:51	2023-06-30 11:22:56	2023-06-19 20:02:13	3.01	-41	-23		

追加されます



モジュールIDは、ここに印刷されています

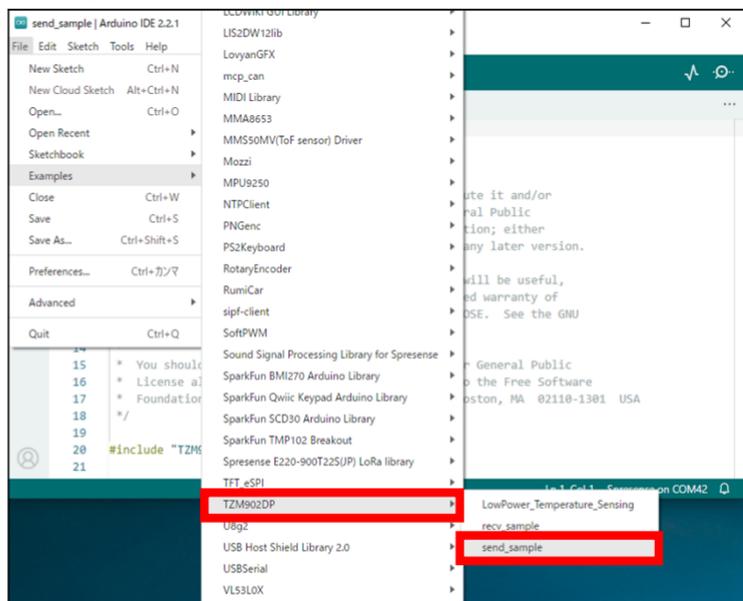
Spresenseを稼働させたとき、こちらがOnになっていないければ通信できていません。

Zetaデバイスからデータを送信しよう！

現時点で、Zetaのライブラリは公開されていません。が、凸版印刷様から事前に頂いたコードをライブラリとして以下に登録しておきました。

こちらをライブラリのインクルードでインクルードしてください。

<https://github.com/TomonobuHayakawa/TZM902DP/>



この中にダミーデータを送信するサンプル
(send_sample) を入れておきました。

こちらを利用して、動作をさせてみてください。

send_sample のコード解説

```
send_sample.ino
17
20 #include "TZM902DP.h"
21
22 TZM902DP Zeta;
23
24 /**
25  * @brief ZETAモジュールから応答またはダウンリンクを取得する
26  */
27
28 int zeta_downlink_check() {
29     char packet[16];
30     int n;
31     int retry = 10;
32
33     while (true) {
34         switch (Zeta.tick()) {
35             case WAKEUP_REASON_DOWNLINK_DATA:
36                 n = Zeta.get_downlink(packet, sizeof(packet));
37                 downlink_process(packet, n);
38                 // if (packet[4] != 0xC0) {
39                 //     puts("No control command on Result.");
40                 //     return 0;
41                 // }
42                 return WAKEUP_REASON_DOWNLINK_DATA;
43
44             case 0xFF:
45                 // Serial.println("zeta recv error.");
46                 if (--retry == 0)
47                 {
48                     return 0xFF;
49                 }
50                 delay(1000);
51                 break;
52         }
53     }
54 }
```

ライブラリのインクルード

インスタンス

戻り値をチェックする関数

send_sample のコード解説

```
send_sample | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spresense

send_sample.ino
61
62 void setup() {
63   pinMode(Zeta.ZETA_wakeup, OUTPUT); // 出力に設定
64   // pinMode(21, INPUT_PULLDOWN); // ZETA_INT
65
66   digitalWrite(Zeta.ZETA_wakeup, HIGH);
67   int resut;
68
69   Serial.begin(115200);
70
71   Zeta.begin();
72   Zeta.connect();
73
74
75   /* Turn on LED0 Connect Complete */
76   ledOn(PIN_LED0);
77   delay(1000);
78   ledOff(PIN_LED0);
79
80   Serial.println("初回起動UL");
81   //1:データ型1byte 0x0F 2:バージョン情報2byte 0x0009
82   char wakeupsensUL[3] = {0x0F, 0x02, 0x00};
83   int size = sizeof(wakeupsensUL);
84   //ZETAアップリンク Zeta.send("デバイス名(自動取得)"SEND_VARIABLE_LENGTH_DATA, "ペイロード"packet, "パケットの大きさ"size)
85   Zeta.send(SEND_VARIABLE_LENGTH_DATA, wakeupsensUL, size);
86
87   sleep(1);
88
89 }
```

各種ピンの設定

HWの初期化

ネットワークの起動
(初回アクセスを行う)

send_sample のコード解説

```
send_sample | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spsense
send_sample.ino
90
91 void loop() {
92
93   char data[128];
94   static char dmy_data = 'A';
95
96   for(int i=0;i<sizeof(data);i++){
97     data[i] = dmy_data+(i % 26);
98   }
99   data[sizeof(data)] = '\0';
100  dmy_data = dmy_data % 10;
101
102  puts(data);
103
104  if (!Zeta.send(SEND_VARIABLE_LENGTH_DATA, data, strlen(data))) {
105    sleep(1);
106    Serial.println("one minute wait");
107    Zeta.send(SEND_VARIABLE_LENGTH_DATA, data, sizeof(data));
108  }
109
110  sleep(10);
111
112 }
113
```

ダミーデータの生成

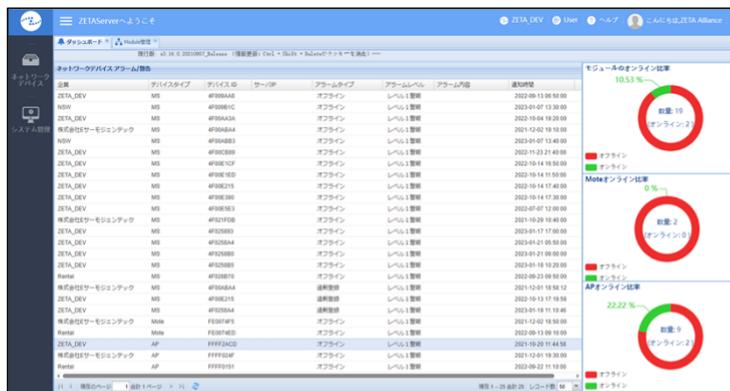
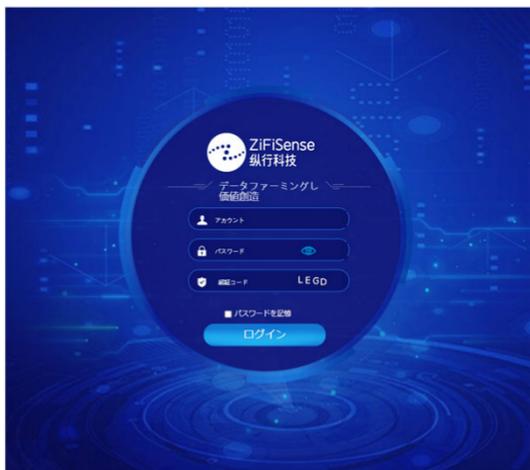
データを送信
(失敗したら1分後に再送)

Zetaサーバー側の設定

Zetaのサーバーは、こちらになります。 <http://13.115.15.0:25450/teamcms/login>

ログインするとこのようになるのですが、こちらのデバイス管理は実は関係がありません…。

単純にデバイスからの情報をフォワーダーするサーバーになります。



実はMQTT/RestAPIは、特に設定なく使えることになります。

NodeREDの設定

NodeREDの環境設定は、「Spresenseからのデータを可視化してみよう！」の中の「フリーローカを使ってローカルPCで可視化しよう！」の説明を参照してください。

フリーローカを使ってもっと手軽にローカルPCで可視化

- 外部のサービスを使う、
 - 契約が必要
 - フリー期間
 - 有償
- など、制約が発生

Sony Semiconductor Solutions Corporation © 2019 – 2023

Node-REDをローカルPCにインストール



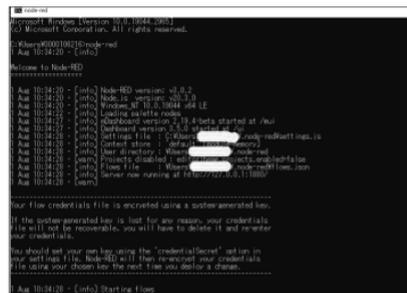
<https://nodered.jp/docs/getting-started/windows>

Node-REDの起動

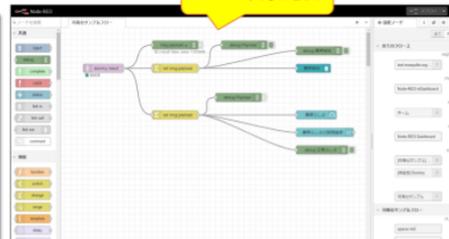
Node-REDを起動します。

※ただし、なぜかPowerShellでの起動には失敗します。

コマンドプロンプトでの実行は成功するのでコマンドプロンプトで起動します。



Node-RED
順に従って、
Node.js
npm
をインストール
その後、No



起動すると、

<http://localhost:1880>

でNode-REDが立ち上がります！

もっと手軽に
ローカルPCで
フリーのMQ
を説明します

NodeREDの設定

「フリーブローカを使ってローカルPCで可視化しよう！」で説明したMQTTSubscribeのノードの編集を行います。

- Broker
- Topic
- Client ID
(•API Key)
(•Secret Key)

を設定してください。

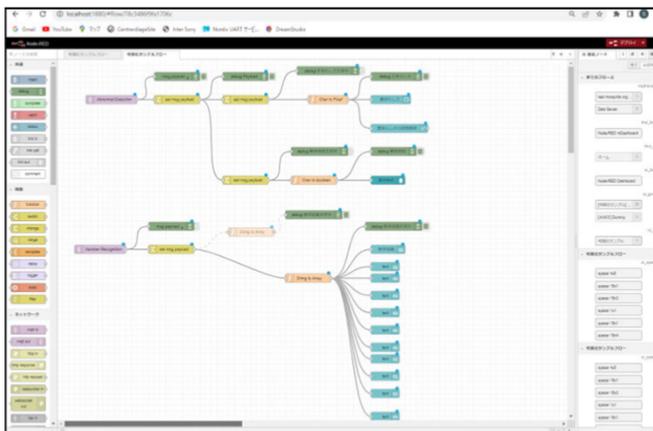
トピックに以下を設定
api_key/version/opType/uid/msgType

Zeta Server

クライアントに以下を設定
[APIKEY]:[Secret KEY]+3桁の乱

NodeREDの設定

フローを設定することで、**Zetaからのデータ**を**可視化**することができます。
今回は**AI**を使ったパイプの**異常検知**と**数字認識**をやりました。

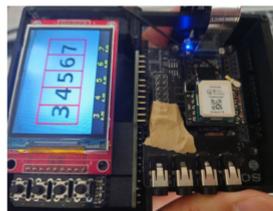


可視化サンプル

デフォルト



数字認識



Spresenseであれば、様々な**LPWA**を簡単に試すことができます！

是非、気になる**LPWA**に挑戦してみてください！