LoRaWAN Shield for Arduino 取扱説明書

センスウェイ株式会社

2018.04

製品概要

この度は、LoRaWAN Shield for Arduino をお買い上げいただき誠にありがとうございます。
本製品は、LoRaWAN モジュールを，Arduino シールドに変換するための基板です。この製品を使うことで，LoRaWAN モジュールを Arduino と接続することができます。

ご使用の前に、この取扱説明書をよくお読みのうえ、正しくご使用ください。また、お読みになった後は大切に保管してください。

注意事項

●必ず同梱されている指定の周辺機器をご利用ください。当社指定の周辺機器以外を使用した場合、発熱・発火・破裂など故障の原因となります。
●お客様による分解や改造、修理をしないでください。発火・感電など故障の原因となります。万が一、改造などにより本製品や周辺機器などに不具合が生じても当社では一切の責任を負いかねます。本製品の改造は電波法違反により罰せられることがあります。
● 本製品を濡れた状態で充電を行うと、感電や回路のショート、腐食が発生し、発熱・発火による火災・故障やけどの原因となります。
●本製品に水などの液体をかけないでください。万が一、液体がかかってしまった場合には、直ちに充電用ケーブルを取り外してください。水漏れや湿気による故障は当社では一切の責任を負いかねます。
● 本製品は、当社が保証する温度・湿度などのご使用条件下でご利用ください。

設置環境

故障や動作不良の原因になりますので，以下の状態や環境条件では使用しないでください。

（ア）液体の中，または液体のかかる場所
（イ）湿気が多い場所
（ウ）潮風を受ける場所
（エ）磁気のを発するものがある場所
（オ）暖房器具、こたつの中、炎天下の車内など高温になる場所
（カ）エアコンなどの風を直接受ける場所

（キ）振動の多い場所
（ク）電磁波が強い場所
（ケ）静電気が発生する場所
（コ）その他，これらに準じる条件下

本製品をご利用にあたって
・サービスエリア内でも電波の届かない場所（トンネル・地下・ビルの谷間・山間部など）では通信できません。また電波状態の悪い場所では通信できないことがあります。
・本製品は実証検証・開発を目的に利用することを前提としておりますので、商用環境でのご利用はできません。

付属品
LoRaWAN Arduino Shield 本体 1 個
アンテナ 1 個
ケーブル 1 個

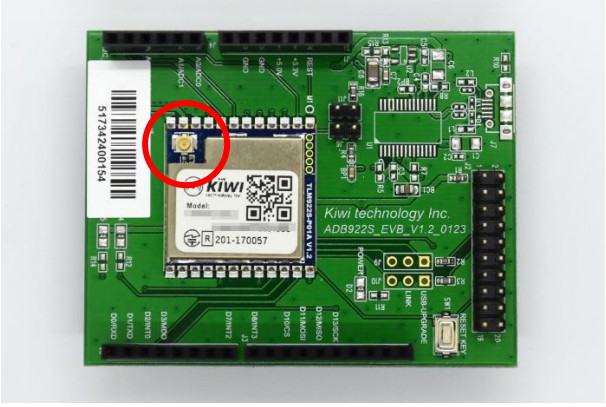
SenseWay Mission Connect への登録

LoRaWAN Shield for Arduino をご利用には、SenseWay Mission Connect への登録が必要です。
下記 URL または「SenseWay Mission Connect マニュアル」をご参照ください。

SenseWay Mission Connect
https://service.senseway.net

モジュールの使用法

ワイヤーの端子はチップの端子に差し込みます。

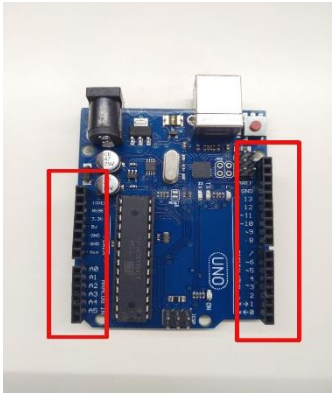


アンテナの繋ぎ方

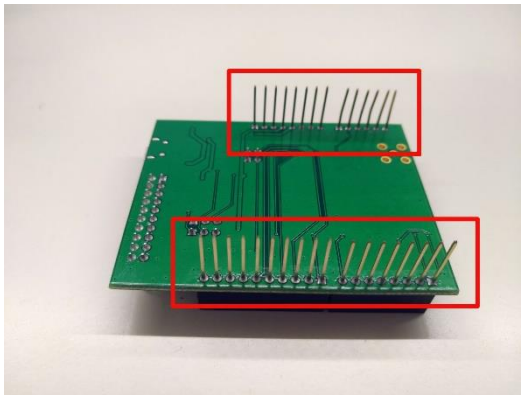
GPS アンテナとワイヤーを用意します。GPS アンテナのソケットにワイヤーのねじ部分を奥までまわして取り付けます。

Arduino とLoRaWAN シールドの接続

Arduino Uno と LoRa Shield を用意します。Arduino のピンソケット（1）にシールド裏面のピン（2）を差し込みます。シールド裏面のピンは折れやすいので注意して差し込んでください。



（1）



（2）

Arduino との内部接続

本 LoRa シールドと Arduino との接続には、D11/D12 ピンを UART シリアルで使用しています。Arduino スケッチでは、SoftwareSerial 等を使って LoRa モジュールとの通信を行い、データの送受信等を行います。

D11　UART_TX
D12　UART_RX

■Join(デバイス認証/暗号化キー交換)

LoRaWAN でのデータ送受信を行うにあたって、サーバに対して join という処理(デバイス認証/暗号化キー交換)を行う必要があります。
これは、以下の書式で行います。

```
lorawan join <Mode>
```

<Mode> join 方式として、otaa (over-the-air activation) と abp (activation by personalization) のどちらかを指定しますが、弊社システムでは、現在 otaa のみ使用可能です。

モジュール join 時コマンド応答例:

```
lorawan join otaa
> busy
> unsuccess

lorawan join otaa
> Ok
> accepted
```

join は、周囲の電波状況やその他の理由により失敗する可能性もあります。状況にもよりますが、スケッチでは、ループ等で成功する　まで繰り返すようプログラムするほうがよいでしょう。
また、一度 join に成功したステータスは、電源を切っても保持されています。

■Uplink データ送信 (モジュール→サーバ)

モジュール(デバイス) のデータ送信は、Arduino D11/12 ピンへの UART シリアル接続のコマンドラインへ、スケッチから以下の
コマンドと引数を指定することで行います。

```
lorawan tx <Type> <PortNum> <Data>
```

<Type> ACK を確認する(confirmed)かどうかを、cnf / ucnf の文字列で指定します。

<PortNum> アプリケーションを区別する等、ユーザが自由に使用可能な数値を 1～223 の間で指定します。

(弊社では、port = 11 を GPS 用途に割り当ててあり、地図へのマッピングアプリケーション等で判断に使用しています。)

弊社使用例:

0～10 = 予約
11 = GPS 座標/高度
12 = 温度/湿度/(気圧)
13 = CPU 温度(他機器温度)
14 = 数値カウント
15 = 距離
:

<Data> 送信するデータの 16 進数表記。

エンコード方法の規定はなく、デバイス側でのエンコード方法とサーバ側アプリケーションでのデコード方法が一致していれば問題ありません。

エンコード例:

弊社 GPS 用途では、GoogleMap 表記の座標を 1000000 倍したものと、高度を 16 進表記して連結し、

緯度(latitude) 35.693968 x 1000000 → 35693968 → 0x0220a590
経度(longitude) 139.766912 x 1000000 → 139766912 → 0x0854ac80

高度(altitude(m)) 31 → 0x001f→ 0220a5900854ac80001f

として 10 バイト(20 桁の 16 進数表記)を組み立てて使用しています。

モジュールデータ送信応答例:

```
lorawan tx ucnf 11 0220a5900854ac80001f
> Ok
> tx_ok

lorawan tx cnf 14 0011223344556677
> Ok
> err
```

DR 値による送信可能データサイズ

LoRa モジュールは、DR 値 (SpreadingFactor と BandWidth の組み合わせ) により、送信可能なデータサイズと到達距離が変化します。(DR 値が低いほど、到達距離は長いが、送信可能データサイズは小さくなります。)

DR 値	SF/BW	Payload(データ)サイズ(bytes)	ビットレート(bit/s)
2	SF10BW125	11	980
3	SF9BW125	53	1760
4	SF8BW125	125	3125
5	SF7BW125	242	5470

Arduino スケッチ等でのモジュール(デバイス) の DR 値の設定は、Arduino の D11/12 ピンへの UART シリアル接続
コマンドラインへコマンドと引数で指定することで行います。

モジュール DR 値設定応答例:

```
lorawan set_dr 5
> Ok
```

■Uplink データ受信 (サーバ→PC/アプリケーションサーバ等)

モジュール(デバイス) から送信したデータを、サーバ経由で、MQTT プロトコルの Subscribe により受信します。
topic 指定は、下記 MQTT アクセス情報の topic に '/rx' を追加します。

MQTT アクセス情報:

host	mqtt.senseway.net
port	1883
username	<username>
passwd	<password>
topic	lora/<username>/<devEUI>

topic 詳細

lora/<username>/<devEUI>/rx	通常 Uplink データ
lora/<username>/<devEUI>/tx	DownlinkData 投入用 Topic
lora/<username>/<devEUI>/tx_send	DownlinkData の送信がサーバーから行われた際に送信されます
lora/<username>/<devEUI>/ack	Downlink(ConfirmdDataDown 時)の ACK

mosquitto (linux 等) でのコマンドライン実行例:

```
# mosquitto_sub -h mqtt.ngc-p.com -p 1883 -u <username> -P '<password>' -t 'lora/<username>/000b78fffe0515e9/rx' -v

# mosquitto_sub -h mqtt.ngc-p.com -p 1883 -u <username> -P '<password>' -t 'lora/<username>/+/rx' -v
```

(該当アカウントに複数デバイスが関連づけられている場合、トピック指定時に '+' をワイルドカードとして使用することにより、複数デバイスのデータを取得可能。)

受信データフォーマット

以下のような JSON の改行なしフォーマットで受信可能です。

{	
"gw": [	ゲートウェイのデータ(複数ある場合は配列列挙)
{	
"date": "2017-10-09T14:21:52.035266Z",	データ受信時刻(UTC)
"rssi": -21,	受信信号強度
"snr": 11.2,	信号雑音比
"gwid": "000b78fffeb00079"	Gateway ID
},	
{	
"date": "2017-10-09T14:21:52.024932Z",	
"rssi": -29,	
"snr": 12.5,	
"gwid": "00001c497bb44c3c"	
}	
],	モジュール(デバイス)のデータ
"mod": {	DR 値 (SpreadFactor と BandWidth の組み合わせ)
"dr": "2",	サーバでのカウント値
"cnt": 3,	LoRa ポート番号(ユーザが 1～223 の間で使用可能)
"port": 11,	

"mt": "cnf", "fq": 925.8, "devEUI": "000b78fffe0515e9", "data": "0220a5900854ac80001f" } }	ACK 要求データ(Confirm)か否(UnConfirm)か 使用周波数 モジュール(デバイス)固有アドレス DevEUI データ (16 進数表記)

gw はゲートウェイ情報の [配列] となっていて、
近隣に複数ゲートウェイが存在し、それぞれで受信した場合は列挙されます。
(注:1 つしかない場合も、要素数 1 の配列となります。)

mod がモジュール情報です。
devEUI の他、モジュールから送信した時の port 番号や data が格納されています。
サーバ側のアプリケーションでは、この data を送信時のエンコード方法と逆の手順でデコードして使用します。

■Downlink データ送信 (PC/アプリケーションサーバ等→サーバ→モジュール)

MQTT の Publish により、モジュール(デバイス)へデータを送信します。
topic 指定は、MQTT アクセス情報の topic に '/tx' を追加します。

mosquitto (linux) でのコマンドライン実行例:

```
# mosquitto_pub -h mqtt.ngc-p.com -u <username> -P '<password>' -t
"lora/<username>/000b78fffe051290/tx" -m ¥
'{"cnf":true,"ref":"abcd1234","port":14,"data":"00112233"}'
```

送信データフォーマット

以下のような JSON の改行なしフォーマットで送信します。

{
"cnf": true,
"ref": "abcd1234",
"port": 14,
"data": "00112233"
}
	
	モジュール(デバイス)に、ACK を要求する場合 true
	ACK を要求した時に、対応がわかるようにユーザがつける ID
	LoRa ポート番号(ユーザが 1～223 の間で使用可能)
	データ (16 進数表記)

cnf は true の場合、デバイスがそれを受信した時に自動で ACK を返します。

ref は、ACK を要求した時に、どのメッセージへの ACK なのかの対応を確認するための ID です。ユーザが自由につけられます。

port と data については、Uplink 時と同じです。

※ MQTT 上は、上記ユーザが指定した '/tx' つき topic のデータ以外に、サーバからモジュール(デバイス)への送信が行われたことを示す '/tx_send' つき topic のデータも流れます。

モジュールでのデータ受信

モジュール(デバイス)では、上記サーバ側への MQTT Publish 以降、次のデータ送信時、サーバに受信データがある場合に、それを受信して表示します。

モジュールデータ送受信応答例:

```
lorawan tx cnf 14 01020304
> Ok
> rx 14 00112233
> tx_ok
```

Arduino のスケッチ等でこれをバンドルする場合は、上記シークエンスを処理する必要があります。

ACK 受信

サーバからのデータ送信時に、"cnf": true により、モジュール(デバイス)からの ACK を要求した場合、モジュール側は、上記データ受信の「次の」データ送信時(つまり、Publish してから次の次のデータ送信時) に、サーバに ACK が自動的に送信されます。

サーバへの ACK は、MQTT の Subscribe により、MQTT アクセス情報の topic に '/ack' を追加した topic で受信できます。

mosquitto (linux 等) でのコマンドライン実行例:

```
# mosquitto_sub -h mqtt.ngc-p.com -p 1883 -u <username> -P '<password>' -t 'lora/<username>/000b78fffe051290/ack' -v

# mosquitto_sub -h mqtt.ngc-p.com -p 1883 -u <username> -P '<password>' -t 'lora/<username>/000b78fffe051290/#' -v
```

(トピック指定時に '#' をワイルドカードとして使用することにより、以降どの文字列が階層的にあってもマッチするので、
/rx や /tx, /ack 等に流れるデータを見ることが可能です。)

ACK は、以下のような JSON の改行なしフォーマットで受信可能です。

{
"devEUI": "000b78fffe050c42",
"ref": "abcd1234",
"date": "2017-10-10T10:16:43.63027Z"
}
	
	ACK を送信したデバイスの DevEUI
	ACK を要求した時に、対応がわかるようにユーザがつけた ID
	ACK を受信した日時(UTC)