

---

# **E220-900T22S(JP) モジュール 利用ガイド**

Rev.1

CLEALINK TECHNOLOGY Co., Ltd.

2022-07-22

## 目次

<b>1</b>	<b>概要</b>	<b>3</b>
<b>2</b>	<b>サンプルコード実行環境の構築</b>	<b>5</b>
2.1	GPIO ヘッダーの UART ピンで UART を使用する	5
2.2	ライブラリインストール	8
<b>3</b>	<b>E220-900T22S(JP) モジュールの動作説明</b>	<b>9</b>
3.1	ノーマルモード (mode 0)	9
3.2	WOR 送信モード (mode 1)	9
3.3	WOR 受信モード (mode 2)	9
3.4	コンフィグモード (mode 3)	10
<b>4</b>	<b>サンプルコード利用方法</b>	<b>10</b>
4.1	コンフィグモード (mode 3) でのパラメータ設定	10
4.2	ノーマルモード (mode 0) でのデータ送受信	13
4.2.1	データ送信	13
4.2.2	データ受信	14

## 1 概要

E220-900T22S(JP) モジュールは UART での入出力によって、モジュールのパラメータ設定および LoRa での送受信を行うことができます。

本チュートリアルでは、入手性や認知度を鑑み、Raspberry Pi を用いて進めていきます。  
PC を使用する場合は、別途 USB-UART 変換ボード等を利用して UART 通信ができるようにする必要があります。

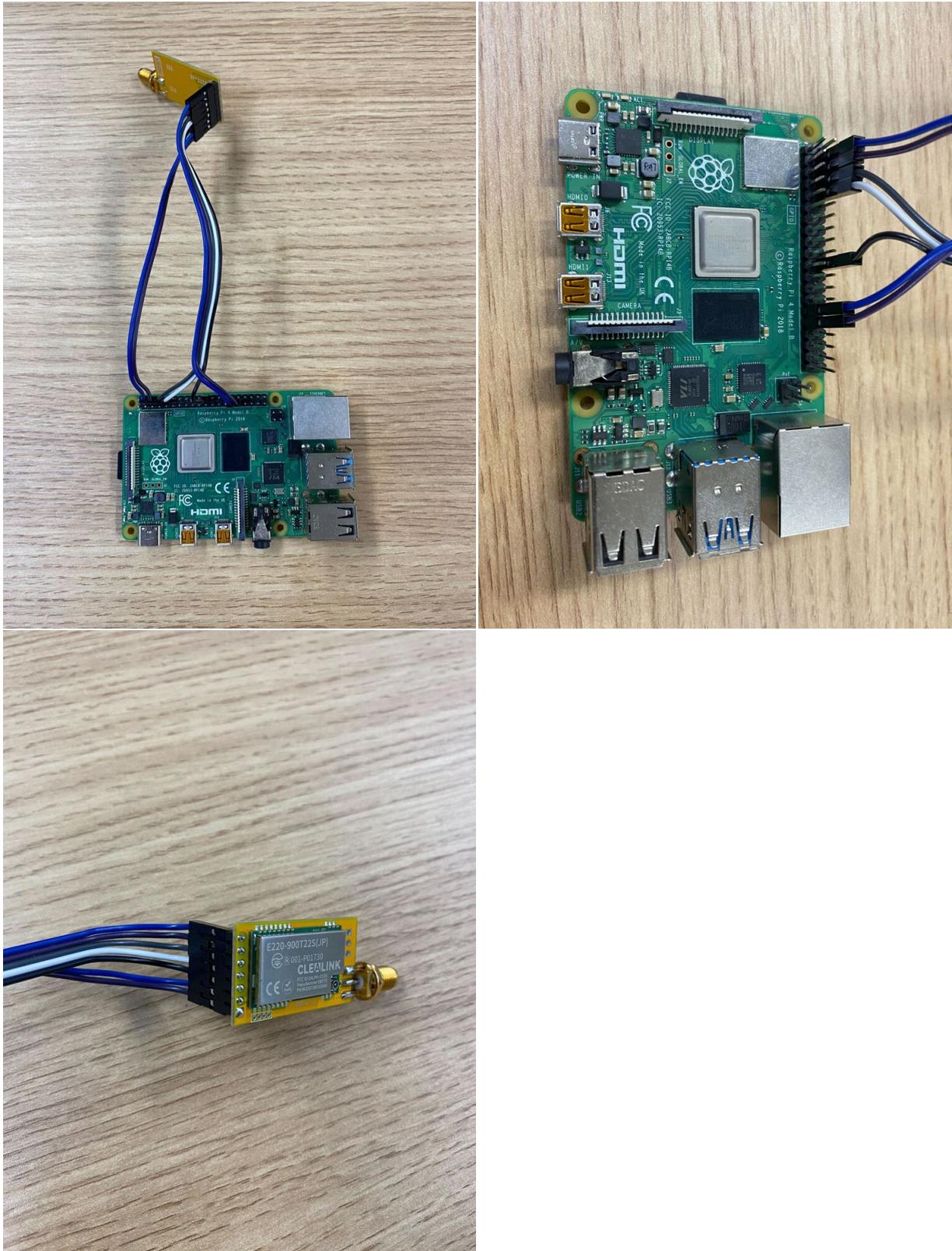
E220-900T22S(JP) モジュールと Raspberry Pi との配線は下記の接続が必要となります。

- モジュールの電源として VCC ピンと GND ピンへの接続
- UART として TXD ピンと RXD ピンへの接続
- モジュールのモード設定のため M0 ピンと M1 ピンへの接続
- モジュールからの通知信号線として AUX ピンへの接続

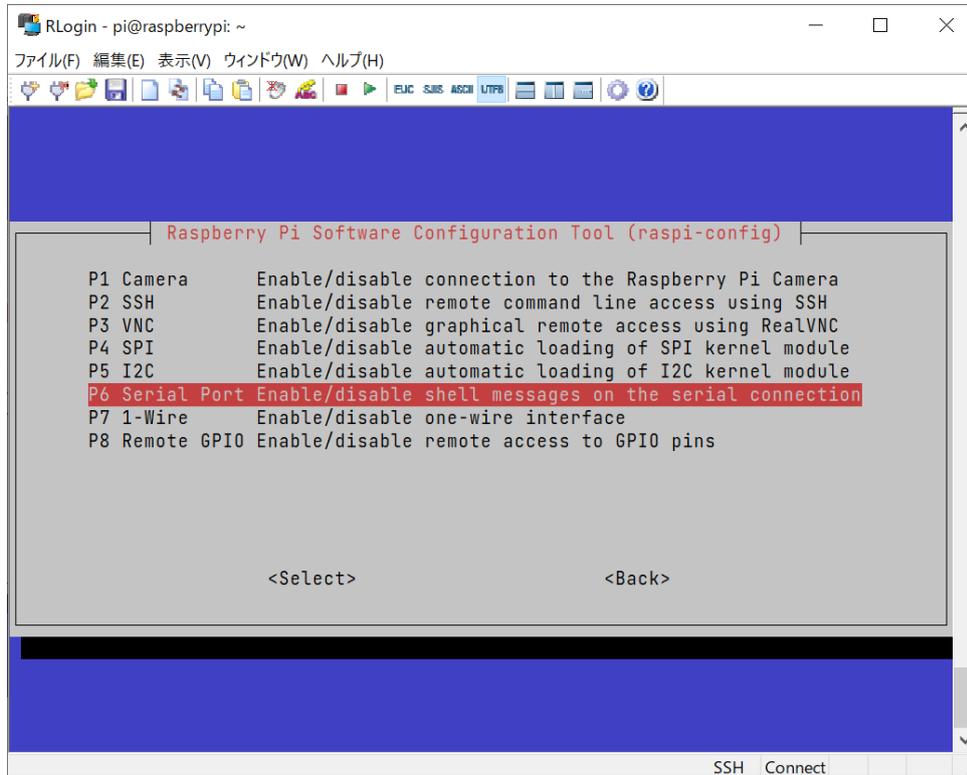
本チュートリアルを進めるにあたり、次表の通り接続してください。

E220-900T22S(JP)	Raspberry Pi
GND	GND
VCC	5V
AUX	GPIO25
TXD	GPIO15
RXD	GPIO14
M1	GPIO6
M0	GPIO5

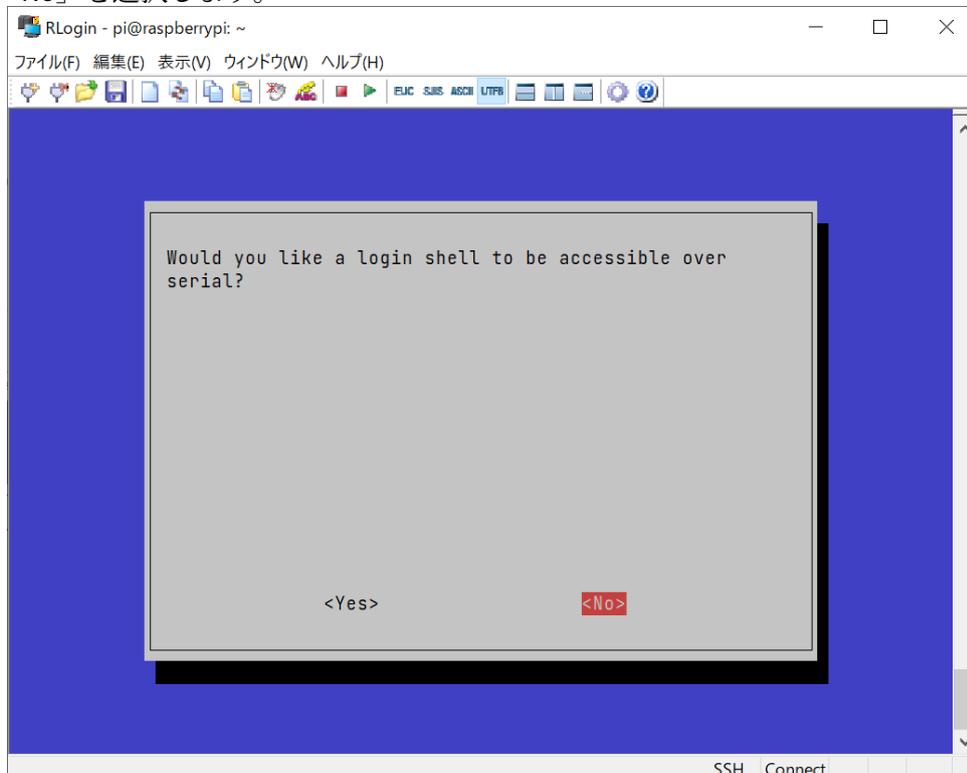
ジャンパーワイヤーを使用した接続例



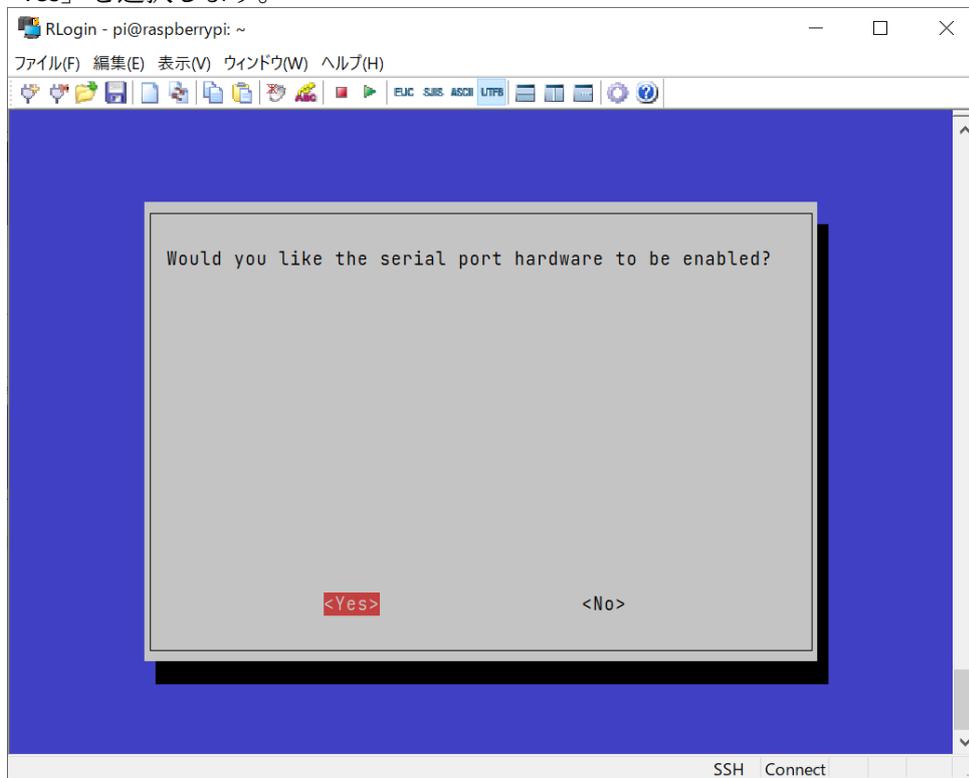




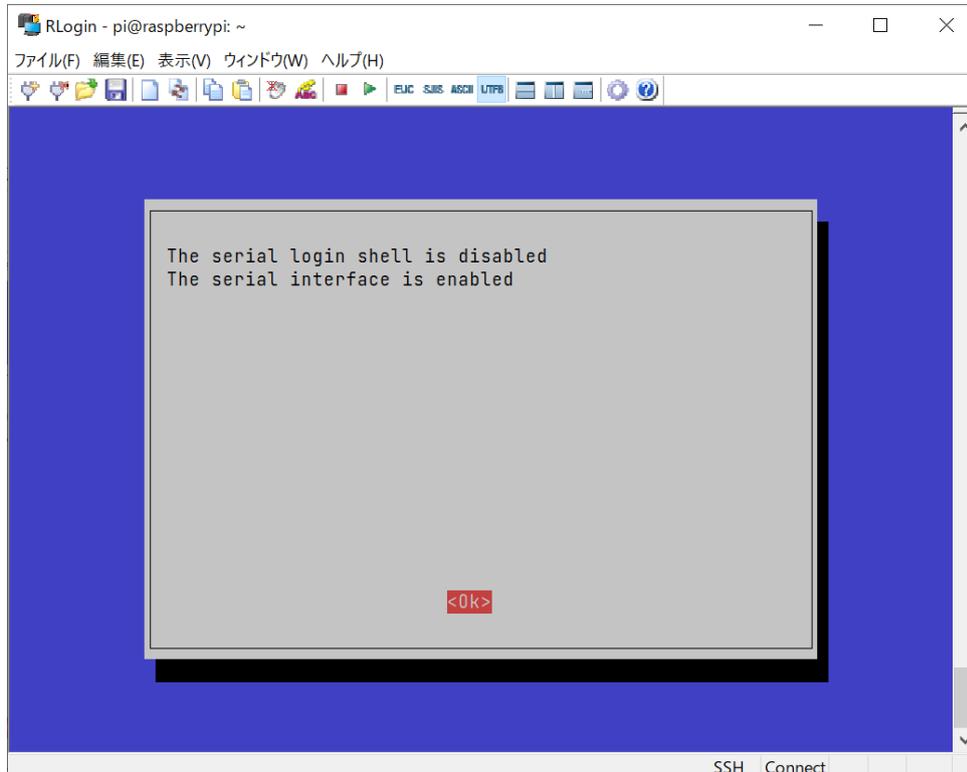
「No」を選択します。



「Yes」を選択します。



すると次の画面になり、シリアルポートが有効になります。



再起動後、次のコマンドを入力すると、/dev/配下に有効化されたシリアルポート `ttyS0` が表示されます。

```
1 pi@raspberrypi:~ $ ls -l /dev/ttyS*
2 crw-rw---- 1 root dialout 4, 64 Jul 20 15:39 /dev/ttyS0
```

## 2.2 ライブラリインストール

以下のライブラリは標準ライブラリではないためインストールする必要があります。下記のコマンドを実行してインストールしてください。

- pySerial  
Python でシリアル通信を実現するライブラリ  
`$ pip install pyserial`
- WiringPi  
Raspberry Pi の GPIO を制御するためのライブラリ  
`$ pip install wiringpi`

### 3 E220-900T22S(JP) モジュールの動作説明

E220-900T22S(JP) モジュールには M0 ピンと M1 ピンがあり、M0 と M1 の High/Low 組み合わせで 4 つの動作モードを決定します。

M0	M1	mode
Low	Low	ノーマルモード (mode 0)
High	Low	WOR 送信モード (mode 1)
Low	High	WOR 受信モード (mode 2)
High	High	コンフィグモード (mode 3)

#### 3.1 ノーマルモード (mode 0)

M0=Low, M1=Low の状態です。

UART から入力されたデータを LoRa で送信します。

LoRa で受信したデータを UART から出力します。

#### 3.2 WOR 送信モード (mode 1)

M0=High, M1=Low の状態です。

UART から入力されたデータを LoRa で送信します。

LoRa で受信したデータを UART から出力します。

LoRa 送信時は、送信前に WOR 受信モードのデバイスをウェイクアップさせるためのプリアンブル (preamble) が自動的に追加されます。

また、LoRa 受信が可能でノーマルモードと同様の動作になります。

※ WOR (Wake on Radio): ワイヤレス電波により待機状態のモジュールを活性化する機能

#### 3.3 WOR 受信モード (mode 2)

M0=Low, M1=High の状態です。

LoRa 受信したデータを UART から出力します。

LoRa 送信は出来ません。

LoRa 受信は、WOR 送信モードで送信されたデータのみ受信可能です。

このモードでは、低消費電力での受信待ち受けを行えます。

データ受信時、AUX ピンから通知信号が出力されるため、AUX ピンを割り込み端子として外部 MCU に接続することで、普段は外部 MCU をスリープさせておき、データ受信のときだけウェイクアップさせる使い方が出来ます。

そのため、外部 MCU と LoRa モジュールをあわせたトータルの消費電力も低く抑えることができますようになります。

### 3.4 コンフィグモード (mode 3)

M0=High, M1=High の状態です。

UART からコマンドを入力することでモジュールのパラメータを設定することができます。パラメータ設定時は baudrate=9600, parity=8N1 で行う必要があります。

コマンドの詳細はデータシートの 7.1 コマンドフォーマットを参照してください。

## 4 サンプルコード利用方法

/home/pi/に sample\_code フォルダを配置します。

### 4.1 コンフィグモード (mode 3) でのパラメータ設定

以下の通りスクリプトを実行し、コンフィグモード (mode 3) の状態にします。

mode3.py の処理は、M0=High, M1=High の状態にしています。

```
1 pi@raspberrypi:~ $ cd sample_code/config_code/  
2 pi@raspberrypi:~/sample_code/config_code $ python3 mode3.py
```

設定ファイル (setting.ini) にモジュールのパラメータ設定値を記述します。それぞれの項目についての詳細はデータシートの 7.2 レジスタ詳細を参照してください。また、設定ファイルはスクリプトと同じフォルダ内に配置して下さい。

setting.ini の設定例

```
1 [E220-900JP]  
2 own_address=0  
3 baud_rate=9600  
4 bw=125  
5 sf=9  
6 subpacket_size=200  
7 rssi_ambient_noise_flag=0  
8 transmitting_power=13  
9 own_channel=0
```

```
10 rssi_byte_flag=1
11 transmission_method_type=2
12 wor_cycle=3000
13 encryption_key=0
```

設定内容は次の通りとなっています。

- モジュールアドレスを 0
- UART の baud rate を 9600bps
- BW(帯域幅) を 125kHz
- SF(拡散率) を 9
- パケット長を 200byte
- RSSI 環境ノイズ機能を無効化
- 送信出力電力を 13dBm
- 周波数チャンネルを 0
- RSSI バイトを有効化
- 送信方法を固定送信モード
- WOR サイクルを 3000ms
- 暗号化キーを 0

以下の通りスクリプトを実行することで、設定ファイルの内容をモジュールへ反映できます。スクリプトの処理としては、UART でモジュールへ設定値書き込みリクエストコマンドを送り、そのレスポンスを受けています。スクリプトの出力として、リクエスト/レスポンスそれぞれのコマンド内容が表示されます。コマンドの詳細はデータシートの 7.1 コマンドフォーマットを参照してください。

```
1 pi@raspberrypi:~/sample_code/config_code $ python3 config_cui.py /dev/
  ttyS0 --apply
2 # Command Request
3 ['\0xc0', '\0x00', '\0x08', '\0x00', '\0x00', '\0x70', '\0x01', '\0x00', '\0xc5'
  , '\0x00', '\0x00']
4 # Command Response
5 ['\0xc1', '\0x00', '\0x08', '\0x00', '\0x00', '\0x70', '\0x01', '\0x00', '\0xc5'
  , '\0x00', '\0x00']
```

また、以下の通りスクリプトを実行することで、モジュールの現在のパラメータ設定状態を一覧出力できます。

スクリプトの処理としては、UART でモジュールへ設定値読み出しリクエストコマンドを送り、そのレスポンスを受けています。スクリプトの出力として、リクエスト/レスポンスそれぞれのコマンド内容が表示されます。また、そのレスポンス内容をもとに現在の設定一覧として具体的な表示をしています。

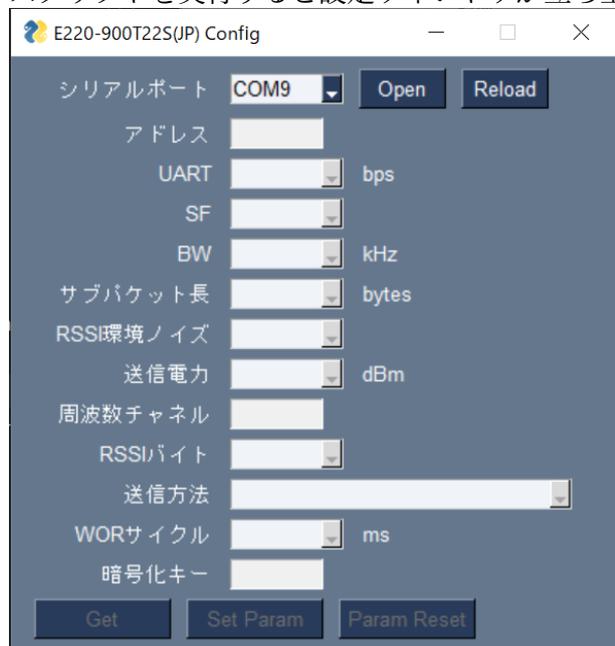
```
1 pi@raspberrypi:~/sample_code/config_code $ python3 config_cui.py /dev/
  ttyS0 --list
```

```
2 # Command Request
3 ['0xc1', '0x00', '0x08']
4 # Command Response
5 ['0xc1', '0x00', '0x08', '0x00', '0x00', '0x70', '0x01', '0x00', '0xc5',
6   , '0x00', '0x00']
7
8 現在の設定一覧
9 Address          : 0x0000
10 UART            : 9600bps
11 Air Data Rate   : 1,758bps, SF: 9, BW:125kHz
12 Sub Packet Size : 200bytes
13 RSSI Ambient noise : Disable
14 Transmitting Power : 13dBm
15 CH              : 0
16 Actual frequency : 920.6MHz
17 RSSI Byte       : Enable
18 Transmission Method : Fixed transmission mode
19 WOR Cycle       : 3000ms
```

また、Windows や Linux でのデスクトップ環境では以下の GUI スクリプトを実行することでも、CUI スクリプト同様にパラメータ設定を行うことができます。

```
1 pi@raspberrypi:~/sample_code/config_code $ python3 config_gui.pyw
```

スクリプトを実行すると設定ウィンドウが立ち上がります。起動時の画面は以下になります。



シリアルポートのコンボボックスからモジュールを接続しているポートを選択し、Open ボタンをクリックします。

Get ボタンが有効化されるのでクリックします。

モジュールのパラメータが表示されます。

The screenshot shows a configuration window titled "E220-900T22S(JP) Config". It contains several settings:

- シリアルポート: COM9 (with Close and Reload buttons)
- アドレス: 0
- UART: 9600 bps
- SF: 9
- BW: 125 kHz
- サブパケット長: 200 bytes
- RSSI環境ノイズ: 無効
- 送信電力: 13 dBm
- 周波数チャンネル: 0 (Frequency: 920.6MHz)
- RSSIバイト: 有効
- 送信方法: 固定送信モード
- WORサイクル: 3000 ms
- 暗号化キー: 0

At the bottom, there are three buttons: Get, Set Param, and Param Reset.

パラメータの値を変更し、Set Param ボタンをクリックすることでモジュールに反映されます。Param Reset ボタンをクリックすると、パラメータの値がデフォルト値になります。

## 4.2 ノーマルモード (mode 0) でのデータ送受信

以下の通りスクリプトを実行し、ノーマルモード (mode 0) の状態にします。mode0.py の処理は、M0=Low, M1=Low の状態にしています。

```
1 pi@raspberrypi:~/sample_code/config_code $ cd ../operation_code/  
2 pi@raspberrypi:~/sample_code/operation_code $ python3 mode0.py
```

### 4.2.1 データ送信

以下の通りスクリプトを実行することで、ファイルからのリダイレクトでデータ送信できます。設定ファイルで固定送信モードを設定した場合、送信スクリプトのオプションとして `-f --target_address 0 --target_channel 0` のようにして宛先アドレスと宛先チャンネルを指定する必要があります。

スクリプトの処理としては、送信データの hex dump を表示し、“SENDED” として送信完了の通知を表示します。

```
1 pi@raspberrypi:~/sample_code/operation_code $ python3 send.py /dev/  
ttyS0 -f --target_address 0 --target_channel 0 < ascii_data.txt
```

```
2 serial port:
3 /dev/ttyS0
4 send data hex dump:
5 00000000: 00 00 00 68 65 6C 6C 6F 20 77 6F 72 6C 64 21    ...hello
   world!
6 SENDED
```

#### 4.2.2 データ受信

以下の通りスクリプトを実行することで、受信データがターミナルに表示されます。  
設定ファイルで RSSI バイトを有効化した場合、モジュールはデータ受信時、シリアルポート TXD を介して受信データを出力した後、続いて RSSI バイトを出力します。  
受信スクリプトのオプションとして `--rssi` を追加することで、出力された RSSI の値を表示することが出来ます。

スクリプトの処理としては、受信データの hex dump を表示し、`--rssi` オプションが有る場合は、RSSI 値を表示します。そして、“RECEIVED” として受信完了の通知を表示します。

```
1 pi@raspberrypi:~/sample_code/operation_code $ python receive.py /dev/
   ttyS0 --rssi
2 serial port:
3 /dev/ttyS0
4 receive waiting...
5 recv data hex dump:
6 00000000: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21 92    hello world
   !.
7 RSSI: -110 dBm
8 RECEIVED
```