AirChecker（C3/USB）利用ガイド

サンプルプログラム説明書

Ver.1.0.0c
CLEALINK TECHNOLOGY Co., Ltd.
2026-11-16

目次

- 1 概要
- 2 AirCheckerのハードウェア構成
- 3 Arduino開発環境のセットアップ
- 4 サンプルプログラムのファイル構成
- 5 Arduinoの起動と操作
  - 5.1 プログラムの変更など
  - 5.2 条件コンパイルの `#define` 定義について
- 6 airchecker起動後の動作
  - 6.1 3桁液晶表示について
  - 6.2 LED点灯について
- 7 ESP32C3-WROOM-02のWi-Fi、Bluetoothを使用する場合の注意事項について
- 8 ライセンス
- 9 サポートについて

1 概要

AirChecker（C3/USB）は、LoRa通信での伝送電波の強度（RSSI値）を、3桁の液晶表示により簡単に目視できる機器です。

LoRa（E220-900T22S(JP)）を使ったAir伝送での通信を行うため、送信機1個、受信機1個を1セットとして動作します。

本機はMCUとしてESP32C3-WROOM-02を搭載しており、出荷時設定の状態で、PCのUSBポートあるいはモバイルバッテリ、USB充電アダプタ等にセットするだけで、独自に動作するようになっています。

出荷時設定では、同じファームウェアが書き込まれていますので、2個をそれぞれ電源に接続すると、送信機（送信側）になるか受信機（受信側）になるかを自動判定するネゴシエーションを実行、その後自動的に送受信動作を開始し、それぞれの受信時のRSSI値を3桁液晶に表示します。

送受信動作は、以下を継続して行います。

- 送信側の送信後、受信側は受信したRSSI値を表示。一定時間受信がなければ赤色LEDを点灯。
- 受信側は受信後、受信通知を送信側に送信。
- 送信側は受信通知を受信後にRSSI値を表示し、一定時間受信がなければ赤色LEDを点灯。

サンプルプログラムについては、Arduino IDEの開発環境をセットアップし、タイマーのライブラリをセットするだけで、コンパイル＆実行することができます。

また、サンプルプログラムを自由に改編して、実験装置の制作、LoRaの学習などに利用できます。

2 AirCheckerのハードウェア構成

No.	項目	内容
1	MCU	ESP32-C3-WROOM-02
2	LoRa	E220-900T22S(JP)
3	LCD	3digit-LCD
4	LED	6-LED、4-color
5	Shift-Register	2個
6	USB	Type A

3 Arduino開発環境のセットアップ

Arduino IDEのセットアップの方法については、ネット検索で多数ヒットしますので、参考にしてください。

1. Arduino IDEをインストールし、日本語化までを実行します。
2. AirCheckerをUSBポートに差し込みます。
3. Arduinoの「ツール」メニューの「ボード」をクリックし、プルダウンメニューで「ESP32C3 DEV MODULE」を選択します。
4. 同じく「ツール」メニューでポートを確認し、ESP32-C3が表示されたポート `COMxx (mmmm ESP32-C3)` を選択します。
5. 「ファイル」メニューの「基本設定」を選択し、「追加のボードマネージャーのURL」に以下を記入（URLをカット＆ペースト）し、「OK」をクリックします。

   ```
   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json
   ```

6. 「ツール」→「ボード」→「ボードマネージャー」を開き、ボードマネージャーの検索に「ESP32」を入力します。表示された「esp32 by Espressif Systems」のバージョンの選択で、**2.0.5（推奨）**を選択し、インストールボタンをクリックします。（少し時間がかかります。）

   > ボードマネージャーは逐次更新されていますが、現状では2.0.11までのバージョンで動作確認をしています。ただし、2.0.6以降については、シリアルモニタにタイマー割り込みのエラーが表示されます。

これで、AirCheckerのためのセットアップが完了です。

プログラムを改編し、デバッグのためにシリアルモニタを使用する場合には、Arduinoの「ツール」メニューで **USB CDC On Boot: “Enabled”** を選択してください。

4 サンプルプログラムのファイル構成

ダウンロードした airchecker_verx.x.x.zip をそのまま解凍すると、airchecker フォルダが生成され、以下のファイルが展開されます。

No.	ファイル名	内容
1	airchecker.h	ハードウェアの設定他、コンパイル時に設定できる値、条件コンパイルの設定
2	airchecker.cpp	メインプログラム
3	airchecker.h	条件コンパイル、各種I/O定義などの設定用ヘッダファイル
4	esp_e220900t22s_jp_lib_3.h	E220-900T22S用ヘッダファイル
5	esp_e220900t22s_jp_lib_3.ino	E220-900T22S用初期設定、送受信など
6	LCD.ino	3桁液晶表示ソフトウェアモジュール
7	LED.ino	LED点灯用ソフトウェアモジュール
8	LoRa.ino	LoRa通信用ソフトウェアモジュール
9	timer.ino	タイマー用ソフトウェアモジュール
10	CONFIG.h	E220-900T22s(JP)のコンフィグレーション設定ファイル

解答後のファイルで、メインプログラムは、 airchecker.cpp　です。
ArduinoIDEの開発環境を使用するためには、サブフォルダ airchecker を作っていただき、その中にすべてのコードをコピーし、airchecker.cpp を airchecker.ino に変更してください。

5 Arduinoの起動と操作

エクスプローラで表示された上記のファイルで、airchecker.ino ファイルをダブルクリックすると、Arduino IDEが起動し、すべてのファイルが読み込まれます。

「マイコンボードに書き込み」ボタンをクリックすると、コンパイル～AirCheckerへの書き込みが実行され、aircheckerが動作を開始します。

5.1 プログラムの変更など

E220-900T22S(JP)が動作する主な出荷時設定は、BW：125、SF：9、固定送信モード、周波数チャンネル0などです。詳細は、データシートを参照しながら、CONFIG.h を確認してください。

CONFIG.h は、データシートのレジスタ記載順に、各種パラメータを設定できるようになっています。アドレス、チャンネル、暗号化キーなどの直接指定する数値以外は、#define の選択によって指定できます。

- #define の二重定義や設定漏れがあった場合、コンパイル時にエラーが生じます。
- WoRの設定については、WoR機能を使用するとき（Mode1、Mode2設定時）に有効になります。

5.2 条件コンパイルの #define 定義について

コンパイル時に設定できる値、条件コンパイルについて説明します。この設定は、airchecker.h に集約されています。

プログラムのバージョン記載

```
#define VERSION_NO 100  // means 1.0.0
```

起動時に3桁液晶に表示されます。

Private Air伝送の機器確認のための設定

```
#define MAGIC_KEY 0x54  // 本Air伝送確認のためのマジックキー
#define UNIQUE_ID 123   // 本Air伝送確認のための固有ID
```

自機番号

```
#define MACHINE_NO 1  // Machine No
```

上記バージョン表示のあとに表示されます。

起動時に送信側か受信側かの自動判定（Negotiation）を行うかの設定

```
#define STARTUP_NEGOTIATION 1
```

送信側動作か受信側動作を指定

```
#define SENDER_or_RECEIVER 1  // 1 : Sender / 0 : Receiver
```

サンプルプログラムでは、airchecker起動時に送信側、受信側を判定するためのNegotiationを行いますので、この定義は上記 `STARTUP_NEGOTIATION` が `0` のときに有効になります。

このとき送信側（1）か受信側（0）かを設定し、コンパイル～デバイスへの書き込みが必要です。

3桁液晶表示を上下逆転するか否かの指定

```
#define LCD_UpsideDown 0  // 液晶表示を上下逆にするか否かの設定
```

PCなどのUSBポートの差し込み位置に対応。実装されている3桁液晶は上下反転表示が可能なため、この機能を実装しています。

Arduinoのシリアルモニタを使用するか否かの指定

```
#define USE_SerialPrint 0  // Use or not Serial.print() to display Monitor
                           // or TerminalAp on PC
```

`0` の場合、`Serial.printf()` 等のコンパイル自体を行いません。

ソースプログラムを変更して、試行、デバッグなどを行う場合に `1`（有効）にします。

受信可能なデータが存在する場合に、RSSI値表示のみを行うか否か

```
#define RSSI_MEASUREMENT_ONLY false  // Air受信時のRSSI表示のみを行う場合true
```

6 airchecker起動後の動作

起動後の動作は、下記の表示の順に移行します。

1. 3桁液晶の表示テスト。
2. 全LEDの4回点滅テスト。
3. Version表示。
4. 自機番号表示～送信側、受信側自動判定のNegotiation実行。
5. 以後RSSI表示。受信時のRSSI値が表示されます。データ受信がない場合には、最後の受信時のRSSI値が表示されたままになります。

6.1 3桁液晶表示について

表示される数値は受信感度を示すRSSI値（Received Signal Strength Indication）で、3桁の数値で表示されます。

RSSI値とは電波の強さを示す値で、単位はdBm（デシベルミリワット）です。数値の範囲は、0～−140 dBm程度までの10進数で表示されますが、液晶が3桁表示になりますので、−（マイナス）は省略します。

この数値は大きい方（マイナスが小さい方）ほど電波は強いということになります。安定して通信（Air伝送）ができる目安は、0～−120 dBm程度までです。

6.2 LED点灯について

LEDはUSBコネクタ側に2個（上から黄色、緑）、アンテナ端子側に4個（上から黄色、青、赤、緑）の6個が実装されています。

LED点灯が意味する状態は、以下になります。（サンプルプログラムでの設定）

No.	位置	色	点灯が意味する状態
1	アンテナ端子側	緑	送信機として動作している場合に点灯
2	アンテナ端子側	赤	受信データがない場合に点灯
3	アンテナ端子側	青	受信機として動作している場合に点灯

No.	位置	色	点灯が意味する状態
4	アンテナ端子側	黄	受信データの先頭のMAGICNOがairchecker のものと異なる場合に点灯
5	USBコネクタ側	緑	E220-900T22S(JP)のAUXの状態（LOWの場合に点灯）
6	USBコネクタ側	黄	リザーブ（Bluetooth LE接続状態などで使用）

7 ESP32C3-WROOM-02のWi-Fi、Bluetoothを使用する場合の注意事項について

本製品に使用している無線マイコンモジュールESP32-C3-WROOM-02は、モジュール表面に刻印はありませんが、工事設計認証（技術適合認証）の取得済み品です。詳細は、以下のハイパーリンクで確認ください。

技術基準適合証明等を受けた機器の情報

また、本機器の初期状態で書き込まれているファームウェア（サンプルプログラム）は、このモジュールの無線出力を行っておりません。ユーザーによってWi-Fi、Bluetoothを実装する場合は、本認証番号を示した上で、ご活用ください。

なお、本E220-900T22S(JP) 920MHz LoRa通信モジュールは、認証番号がモジュールに刻印済みです。

本サンプルプログラムでWi-Fi、Bluetoothを利用する場合に、認証番号の表示を行う関数も用意しました。上記ファイル中の `LCD.ino` の最下部にある行で、以下のように設定します。

```
#define USING_WIFI_BLUETOOTH true
```

適切な箇所（起動時など）で、以下を実行してください。

```
Set_SEGMENT_r();
```

`r201-220555` を3桁液晶に横スクロール表示します。

8. ライセンス

本プロジェクトのソースコードおよびドキュメントは、MIT License のもとで公開しています。

個人利用・商用利用を問わず、MIT License の条件に従って、自由に使用・複製・変更・再配布することができます。

詳細については `LICENSE` ファイルを参照してください。

9. サポートについて

本プロジェクトは無償公開のサンプルおよびライブラリであり、技術サポートは提供していません。

本プロジェクトのソースコードおよびドキュメントは現状のまま（AS IS）で提供されます。
使用環境や接続機器などによる動作を保証するものではありません。

本プロジェクトの利用、変更、組み込み等は、利用者自身の責任において行ってください。

なお、本プロジェクトのソースコード、README およびその他のドキュメントは、予告なく変更される場合があります。
ソースコードの変更にあたっては、可能な限り既存バージョンとの互換性を維持する方針としています。

改訂履歴

日付	版	内容
2024-05-01	初版	初版発行
2026-11-16	第2版	タイマー割り込み処理をESP32コア標準のタイマーAPIに変更。 Arduino IDEとPlatformIOでソースコードを共通化するため、ファイル構成を `.ino` から `.cpp`／`.h` に変更。