AirChecker（C3/USB）利用ガイド — PlatformIO開発環境

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 開発環境のセットアップ

3.1 Visual Studio CodeとPlatformIO IDEのインストール

1. Visual Studio Code（VSCode）をインストールします。
2. VSCodeの「拡張機能」で「PlatformIO IDE」（提供元：PlatformIO）を検索し、インストールします。
3. 初回のセットアップが完了したら、必要に応じてVSCodeを再起動します。

Arduino IDEのインストールや、追加のボードマネージャーURLの登録は不要です。Arduinoフレームワークと開発ツールは、PlatformIOがプロジェクトの設定に従って管理します。

3.2 プロジェクトを開く

配布されたPlatformIO対応プロジェクトを展開し、VSCodeの「ファイル」→「フォルダーを開く」で、`platformio.ini` があるフォルダーを開きます。`src` フォルダーだけを開かないでください。

初回ビルドでは、設定に応じて開発ツールやライブラリがダウンロードされるため、インターネット接続と時間が必要です。

3.3 ビルド環境の確認

本プロジェクトの環境名は `esp32-c3-devkitc-02` です。現在ビルドできている、プロジェクト付属の `platformio.ini` を使用してください。

主な設定項目は次のとおりです。

項目	内容
`[env:esp32-c3-devkitc-02]`	ビルド環境名
`platform`	使用するESP32開発プラットフォームとバージョン
`platform_packages`	必要な場合のツール・フレームワークの指定

項目	内容
`board`	`esp32-c3-devkitc-02`
`framework`	`arduino`
`monitor_speed`	プログラムの `Serial.begin()` と同じ通信速度
`lib_deps`	使用する外部ライブラリ（必要な場合）

現在の `platformio.ini` は本資料では未確認です。`platform` や `platform_packages` を一般的な設定で上書きすると、Arduinoコアのバージョンが変わることがあります。付属設定を維持してください。

タイマー処理をArduino-ESP32 3.xの標準タイマーAPIに置き換えた版では、旧 `ESP32_C3_TimerInterrupt` ライブラリの追加は不要です。旧版ライブラリと3.x対応コードを混在させないでください。

3.4 USB接続とCOMポートの確認

AirCheckerをPCのUSBポートに接続し、Windowsのデバイスマネージャーで COMポートを確認します。PlatformIOのターミナルでは次のコマンドでも確認できます。

```
pio device list
```

必要な場合は、既存の環境設定に以下を追加します。`COM10` は実際の番号に変更してください。

```
upload_port = COM10
monitor_port = COM10
```

ネイティブUSB経由の `Serial` を使用する場合は、USB CDCの設定も必要です。プロジェクト付属の設定を確認してください。外付けUSB-UART経由の場合とは設定が異なります。

4 サンプルプログラムのファイル構成

.cpp化した構成の例です。実際の配布ファイル名を優先してください。

ファイル／フォルダー	役割
`platformio.ini`	ボード、開発環境、ライブラリ、書き込みなどの設定
`src/airchecker.cpp`	メイン処理、`setup()`、`loop()`
`src/airchecker.h`	共通設定、I/O定義、条件コンパイルの設定
`src/CONFIG.h`	E220の通信パラメーター設定
`src/LCD.cpp`、`src/LCD.h`	3桁液晶の表示処理と公開関数の宣言
`src/LED.cpp`、`src/LED.h`	LEDの制御処理と公開関数の宣言
`src/LoRa.cpp`、`src/LoRa.h`	LoRaの送受信処理と公開関数の宣言

ファイル／フォルダー	役割
`src/timer.cpp`、必要なヘッダー	タイマー処理
E220用ソース・ヘッダー	E220の初期設定や送受信処理
`.pio/`	自動生成されるビルド結果や依存ライブラリ

`.cpp` ファイルはそれぞれ別にコンパイルされます。Arduino APIを使うファイルでは `#include <Arduino.h>` を記述します。外部に公開する関数はヘッダーで宣言し、ファイル内部だけで使う通常の関数には `static` を付けます。

複数ファイルで共有する変数は、ヘッダーで `extern` 宣言し、実体を1つの `.cpp` だけで定義します。`.cpp` ファイル自体を `#include` する必要はありません。

5 ビルド・書き込み・動作確認

5.1 ビルドと書き込み

VSCodeのPlatformIOアイコンを開き、対象環境の「PROJECT TASKS」から操作します。

操作	タスク
コンパイル	Build
コンパイルして書き込み	Upload
シリアルモニター	Monitor
ビルド生成物の削除	Clean

PlatformIOのターミナルからも実行できます。カレントフォルダーを `platformio.ini` のある場所にして実行してください。

ビルド：

```
pio run -e esp32-c3-devkitc-02
```

書き込み：

```
pio run -e esp32-c3-devkitc-02 -t upload
```

シリアルモニター：

```
pio device monitor -e esp32-c3-devkitc-02
```

モニターの終了は `Ctrl+C` です。書き込み時にCOMポートが使用中と表示される場合は、シリアルモニターやTera Termなどを閉じてください。

5.2 プログラムの変更

E220のアドレス、チャンネル、通信条件などは `CONFIG.h` を編集します。通信する機器同士で必要な設定を合わせてください。

変更後は保存し、Buildでエラーがないことを確認してからUploadを実行します。

5.3 条件コンパイルの設定

機器番号や動作条件などは、現在のソースの `airchecker.h` を編集します。元の説明書には、次の設定項目が記載されています。名称や有効値は、使用中のソースを確認してください。

設定名	内容
`VERSION_NO`	起動時に表示するバージョン番号
`MAGIC_KEY`、`UNIQUE_ID`	通信データの識別用設定
`MACHINE_NO`	自機番号
`STARTUP_NEGOTIATION`	送信側・受信側の自動判定
`SENDER_or_RECEIVER`	自動判定を使用しない場合の送信側・受信側指定
`LCD_UpsideDown`	液晶表示の上下反転
`USE_SerialPrint`	デバッグ出力の有効・無効
`RSSI_MEASUREMENT_ONLY`	RSSI表示のみを行う動作の指定

シリアル出力を確認する場合は、プログラム側の出力設定、USB CDCまたはUARTの経路、モニターのCOMポートと通信速度を合わせます。

---

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
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。

参考

- Visual Studio Code
- PlatformIO IDE for VSCode
- ESP32-C3-DevKitC-02のボード設定
- PlatformIOのArduino開発例

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。