UHF Reader Android App Manual

Jence Bangladesh

August 10, 2025

Contents

User Guide

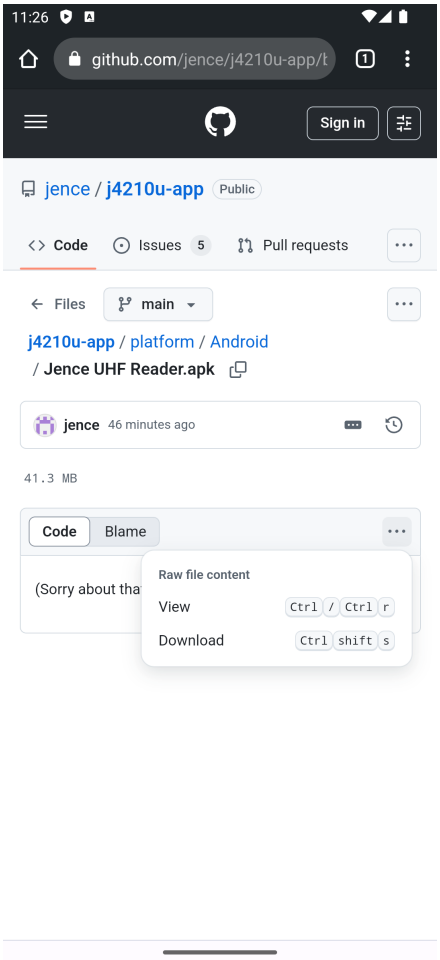
1 Introduction

This app is designed to interface with UHF RFID readers such as the J4210U, J4211UH, J4212UH , J4220U, J4224UX and J4311UH. It allows users to scan RFID tags, configure reader settings, and view reader information.

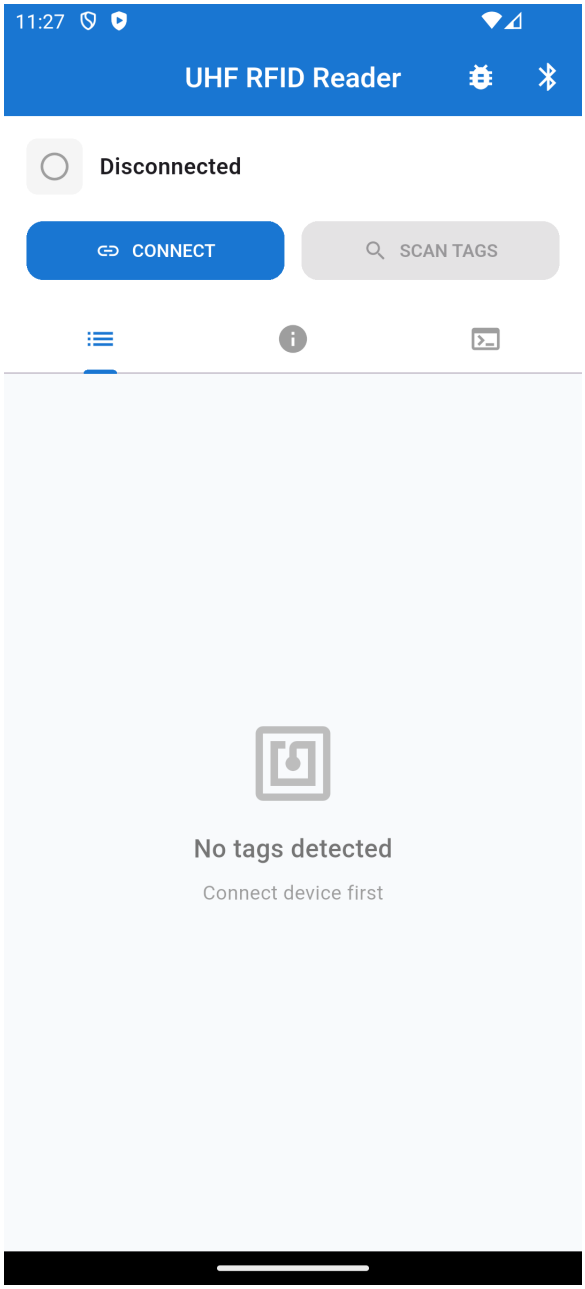
- Works with multiple devices via USB OTG
- Real-time inventory and tag logging
- Simple and intuitive UI

2 Installation

- Go to Our Github Page.
- Click on the download button.



- Install the app. After Installing you will see this interface



Your app is successfully installed

3 Device-Specific Usage

The app supports the following UHF RFID reader models:

- J4210U
- J4211UX
- J4212U
- J4220U
- J4224UX

- J4311UH

Each model may require slightly different setup or behaves differently when interacting with the app.

3.1 J4210U,J4212U,J4220U,J4224UX

Features:

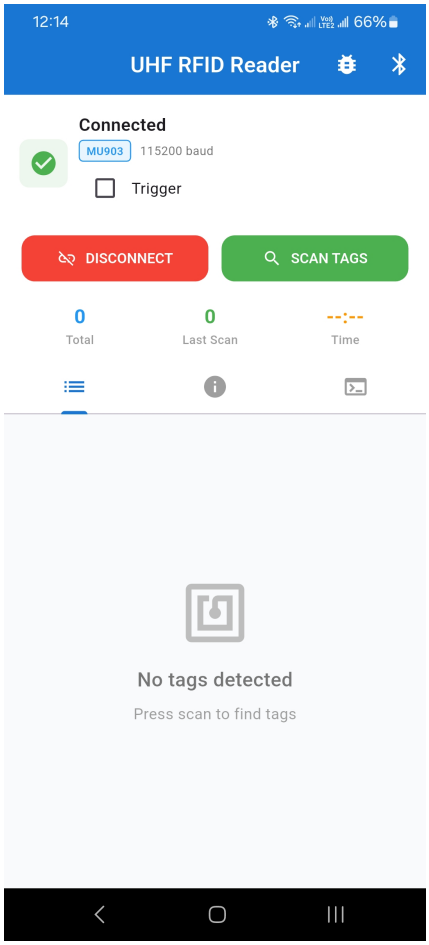
- Has debug option
- Support to modify Device Settings
- Supports fast inventory scans.

Connection:

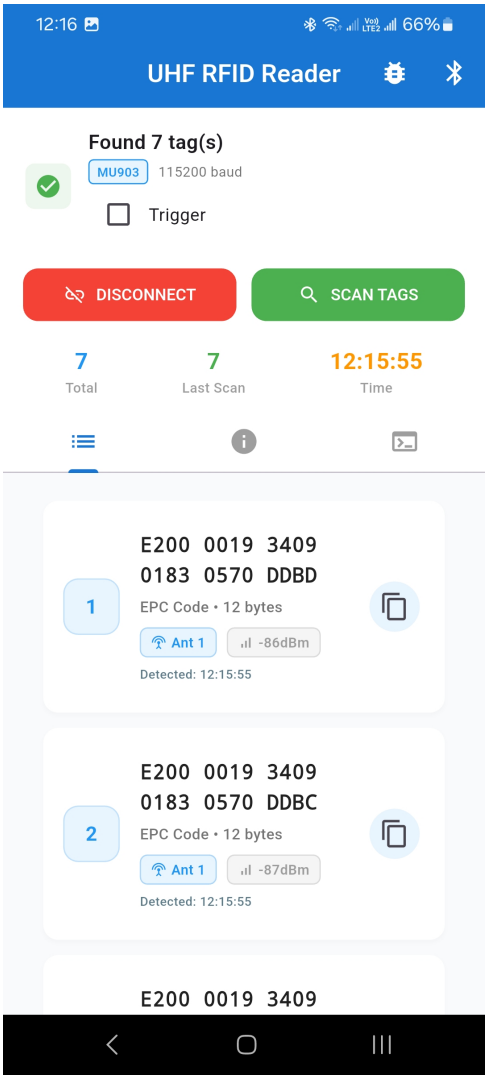
- Plug the Uhf device into your phone using USB OTG.
- Press the connect button.
- Allow USB permission when prompted by Android.

3.1.1 App interface

After Connecting the app interface will show like this:



- press the scan button to start scanning



- There are 3 tabs in the app.
  - **Tag Tab**: show tag info.
  - **Info Tab**: Show reader information and configure device setting.
  - **Debug Tab**: Show debugging logs.
- For configuring device settings go to the info tab ,Scroll down to Reader setting. Here you'll find some settings to modify.

3.2 J4211UX

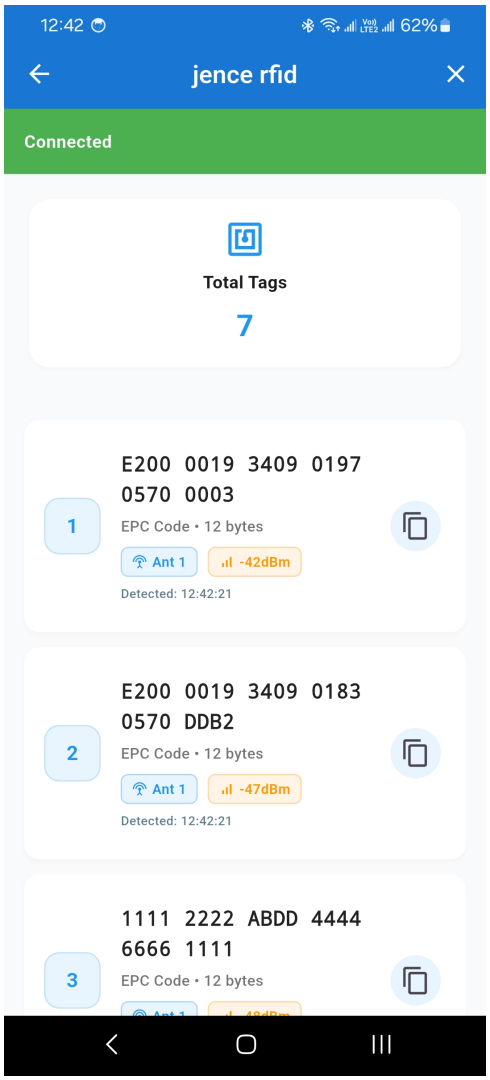
Connection process is same as shown in 3.1 .But there is a trigger button in this device. If you want to use trigger button on the Device to scan tags then just check the Trigger option. Then you can scan with trigger button.

3.3 J4311UH

In this device there not cable connection is needed.

- Turn on send data in bluetooth in the J4311UH.
- Turn on bluetooth in your phone and Scan.
- Connect with device using password **1234**.
- Go to our app and click on the bluetooth in the topbar.

- Then Press Connect to UHF Reader.
- Select the device that you paried.
- Press the button on the reader it will show data of the tag it scanned on the display as well as the mobile screen.



Developer Guide

4 Overview

This mobile application is developed using the Flutter framework and provides a modern interface to interact with UHF RFID readers, including J4210, J4211,J4212,J4220, J4224, and J4311. The app supports both wired and wireless communication: J4311 connects wirelessly, while other uhf reader connect via USB.

The main features of the application include real-time RFID tag scanning and the ability to modify reader settings directly from the mobile device. The entire application, including communication and data parsing logic, is implemented in Dart, with no reliance on native (C/C++) code. This makes the codebase fully portable and easy to maintain or extend.

This guide is intended for developers who want to understand, customize, or contribute to the project. It includes information about setting up the development environment, understanding the app architecture, and working with various parts of the codebase.

Key Features:

- Real-time UHF RFID tag scanning
- Reader configuration through the app
- Supports both USB and wireless communication
- Fully implemented in Dart with modular Flutter UI

Technologies Used:

- Flutter (Dart)
- USB Serial Communication (pure Dart)
- Wireless communication (Bluetooth over Dart)

Developers are encouraged to begin with the environment setup section, followed by exploring the communication modules and UI components to better understand and customize the app.

5 Repository Setup

Before running the app, ensure that Flutter is properly installed on your system. Follow the official Flutter installation guide for your operating system:

- **Windows:** https://docs.flutter.dev/get-started/install/windows
- **macOS:** https://docs.flutter.dev/get-started/install/macos
- **Linux:** https://docs.flutter.dev/get-started/install/linux

Once Flutter is installed and set up, follow the steps below to clone and run the UHF RFID app.

1. Visit the GitHub repository at:
   `https://github.com/jence/j4210u-app`
2. Navigate to the following directory:
   `platform/android/J4210u_dart`
3. Clone or download the repository folder to your local machine.
4. Open the `J4210u_dart` folder in **Visual Studio Code**.
5. Run the following command in the terminal to install project dependencies:

   ```
   flutter pub get
   ```

6. To launch the app for the first time:
   - Open the main Dart file: `lib/main.dart`
   - From the top menu, select: `Run > Run Without Debugging`
7. Make sure your Android device is connected via USB with developer mode enabled.
8. The app should build and run on your device. This initial run may take a few minutes as dependencies are resolved and compiled.

A step-by-step video tutorial will be provided soon to visually guide you through the entire setup and first run.

5.1 In Case of Build Failed

Remove the package attribute from AndroidManifest.xml

1. **Go to:**dependencies\direct dependencies\flutter_bluetooth_serial-0.4.0\android\src\main\Androi
2. Change this line: $\langle manifestxmlns : android = "http://schemas.android.com/apk/res/androi$
   $"io.github.edufolly.flutterbluetoothserial"\rangle$

To: $\langle manifestxmlns : android = "http://schemas.android.com/apk/res/android"\rangle$

Set the namespace in build.gradle

1. Go To: **Go to:**dependencies\direct dependencies\flutter_bluetooth_serial-0.4.0\android\build.gra
2. Add this line in top of android block

```
android {
    namespace'io.github.edufolly.flutterbluetoothserial' //

    <- Required now! compileSdkVersion 33 // existing code...

    }
```

Clen and Get Pub:Run This Commands

```
flutter clean
flutter pub get
```

6 Customization Guidelines

This section provides instructions for developers who want to customize the UHF RFID Reader app. Whether you're adding new features, modifying how tag data is handled, or adapting the app for specific use cases, this guide outlines where to find and how to adjust the relevant parts of the codebase.

1. Viewing and Using Tag Data (Wired Devices)

- The screen for scanning and displaying tag data for USB-connected readers (J4210, J4331) is implemented in:
  `lib/screens/rfid_reader_screen.dart`
- You can extract and use tag properties like EPC ID from this screen for other purposes such as logging, filtering, or triggering external actions.

2. Protocol Handling and Backend Logic (Wired Devices)

- All USB communication logic and reader command handling is located in:
  `lib/services/uhf_rfid_manager.dart`
- This file manages sending commands and parsing incoming data for wired RFID devices.

3. UHF Driver Code

- The app supports both MU903 and MU910 UHF modules. Each module has a dedicated Dart file for its protocol and parsing logic:
  - `lib/widgets/mu903_protocol.dart`
  - `lib/widgets/mu910_protocol.dart`
- Modify these files to change how raw data is parsed or commands are constructed, depending on which UHF module your device uses.

4. Wireless Communication (J4311 Device)

- No special driver setup is needed for wireless readers like J4311.
- The main UI for wireless communication is located in:
  `lib/screens/uhf_reader_screen.dart`
- All JSON-based message parsing and Bluetooth logic is implemented in:
  `lib/services/bluetooth_service.dart`

- You can modify this service to support different data formats or customize how the app handles incoming data.

5. General UI Customization

- UI components and screen layouts are located in the `lib/screens/` and `lib/widgets/` directories.
- Use Flutter's widget system to add or modify buttons, lists, dialogs, etc.
- For state changes, use the Provider-based state management already implemented in the app.

6. Logging and Debugging

- Add `print()` or use the `logger` package to log commands and responses in:
  - `uhf_rfid_manager.dart` (wired)
  - `bluetooth_service.dart` (wireless)
- These logs can help in debugging communication or verifying device responses.

With this modular architecture, developers can easily extend the app for new devices, alternate communication formats, or custom business logic with minimal code changes.