SSP Slave Application

User Guide

Version 0.2

May 2016

Redpine Signals, Inc.
2107 N. First Street, #540
San Jose, CA 95131.
Tel: (408) 748-3385
Fax: (408) 705-2019
Email: info@redpinesignals.com
Website: www.redpinesignals.com

About this Document

This document describes the process of bringing up the RS9113 based module as BT slave device and used for SPP chat application between two BT devices using secure simple pairing (SSP).

Disclaimer:

The information in this document pertains to information related to Redpine Signals, Inc. products. This information is provided as a service to our customers, and may be used for information purposes only. Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating this document in the future.

Table of Contents

Table of Figures

Table of Tables

No table of figures entries found.

1 Introduction

This project is applicable to all the WiSeConnect variants like WiSeConnect Plus, WiSeMCU and WyzBee. The term WiSeConnect refers to its appropriate variant.

1.1 Application Overview

1.1.1 Overview

This application demonstrates how to configure the device in Slave mode and establish SPP profile connection with remote Master device using secure simple paring (SSP) and data exchange between two devices using SPP profile.

In this Application, WiSeConnect device configures in Slave mode and waits to accept SPP profile level connection using secure simple pairing (SSP) from remote device. After successful SPP connection, Application will wait for data to receive from connected remote device. If remote device sends data to WiSeConnect device, WiSeConnect device receives the data and send back the same data to remote device using SPP profile.

1.1.2 Sequence of Events

This Application explains user how to:

- Configure WiSeConnect module to act as Slave
- Configure device to secure simple pairing (SSP)
- Configure device in discoverable and connectable mode
- Accept SPP level connection from the Smartphone
- Loop back the received messaged

1.2 Application Setup

The WiSeConnect in its many variants supports SPI and UART interfaces. Depending on the interface used, the required set up is as below:

1.2.1 SPI based Setup Requirements

- Windows PC with CooCox IDE
- Spansion (MB9BF568NBGL) micro controller

Note: If user does not have Spansion (MB9BF568NBGL) host platform, please go through the SPI-Porting guide **\sapis\docs\RS9113-WiSeConnect-SAPI-Porting-Guide-vx.x.pdf** for SAPIs porting to that particular platform.

- WiSeConnect device
- BTLE supported Smart phone with SSP and GATT client

Note: Install Light blue App for tablet for ipad mini and BLE scanner app for android smart phone.

1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- BTLE supported Smart phone with SSP and GATT client

Note: Install Light blue App for tablet for ipad mini and BLE scanner app for android smart phone.

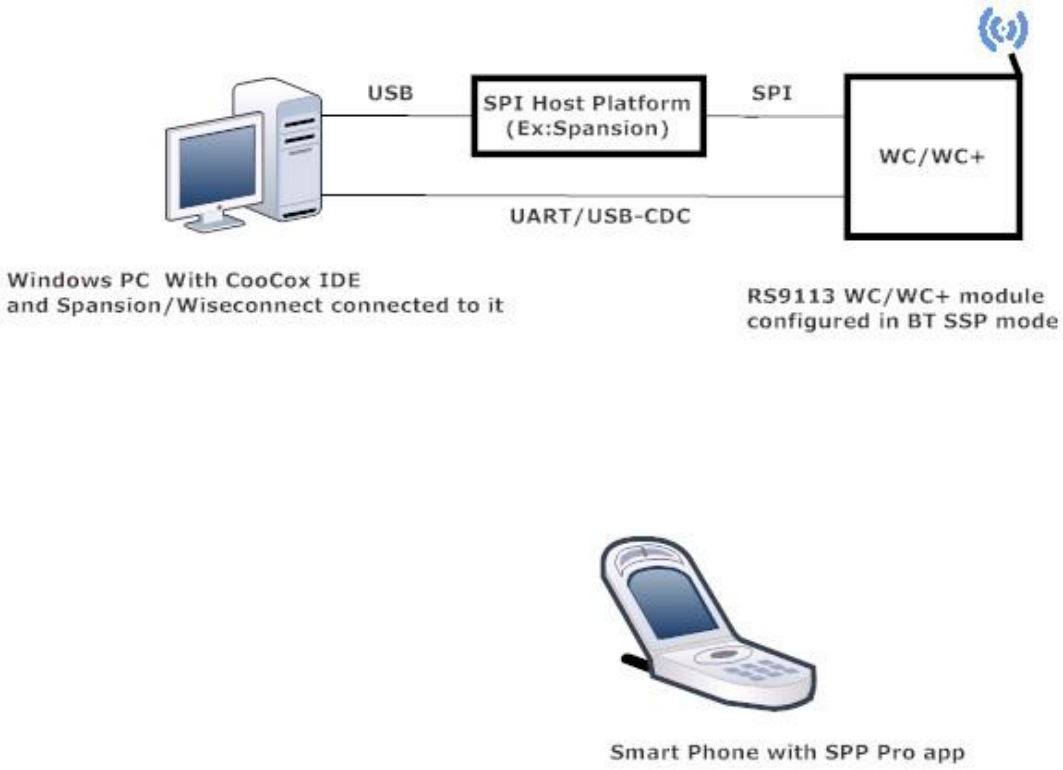


Figure 1: Setup Diagram

2 Configuration and Execution of the Application

The example application is available in the Release at **{Release $}/host/sapis/examples.** These examples will have to be initialized, configured and executed to test the application. The initialization varies based on the interface but configuration and execution are the common.

2.1 Initializing the Application

2.1.1 SPI Interface

If User using SPI interface, Please refer the document ***sapis/platforms/spansion_MB9BF568NBGL/RS9113-WiSeConnect_SAPIS_Spansion_Project_User_guide.pdf*** for opening the ***bt_ssp_test_app*** example in CooCox IDE.

2.1.2 UART/USB-CDC Interface

If User using UART interface, Please refer the document ***sapis/platforms/windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf*** for opening the ***bt_ssp_test_app*** example in Dev-C++ IDE

2.2 Configuring the Application

1. Open ***sapis/examples/bt/bt_ssp_test_app/rsi_ssp_test_app.c*** file and update/modify following macros :

RSI_BT_LOCAL_ANME refers name of the WiSeConnect device to appear during scanning by remote devices.

```
#define RSI_BT_LOCAL_NAME          "SPP_SLAVE"
```

PIN_CODE refers four bytes string required for pairing process.

```
#define PIN_CODE        "1234"
```

Following are the **non-configurable** macros in the application:

BT_GLOBAL_BUFF_LEN refers to the number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN         10000
```

2. Open ***sapis/include/rsi_wlan_config.h*** file and update/modify following macros:

```
#define CONCURRENT_MODE             RSI_DISABLE
#define RSI_FEATURE_BIT_MAP         FEAT_SECURITY_OPEN
#define RSI_TCP_IP_BYPASS           RSI_ENABLE
#define RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_BYPASS
#define  RSI_CUSTOM_FEATURE_BIT_MAP  0
#define  RSI_BAND                    RSI_BAND_2P4GHZ
```

2.3 Executing the Application

1. **SPI Interface**

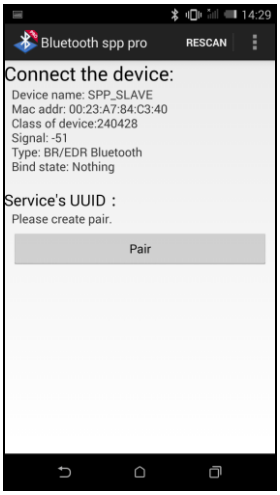
If User using SPI interface, Please refer the document ***sapis/platforms/spansion_MB9BF568NBGL/RS9113-WiSeConnect_SAPIS_Spansion_Project_User_guide.pdf*** for executing the ***bt_ssp_test_app*** example in CooCox IDE.

2. **UART/USB-CDC Interface**

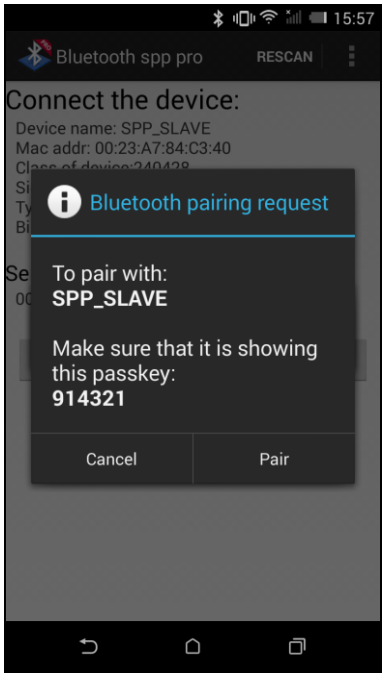
If User using UART interface, Please refer the document ***sapis/platforms/windows_uart/RS9113-WiSeConnect_SAPIS_Windows_Project_UserGuide.pdf*** for executing the ***bt_ssp_test_app*** example in Dev-C++ IDE

3. After the program gets executed, WiSeConnect module initializes the SPP profile and waits for the incoming connection.
4. Open Bluetooth SPP pro app on mobile and do the scan until WiSeConnect device (Ex: “SPP_SLAVE”) gets present in the scan list.

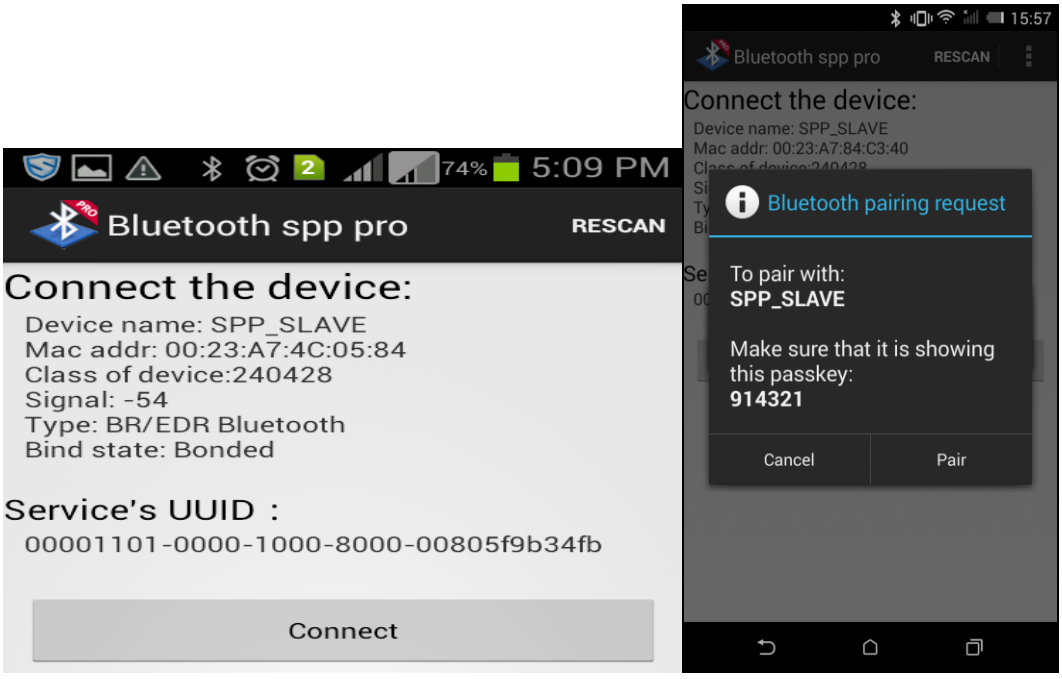
5. After successful scan, select the device and initiate pairing to WiSeConnect device.



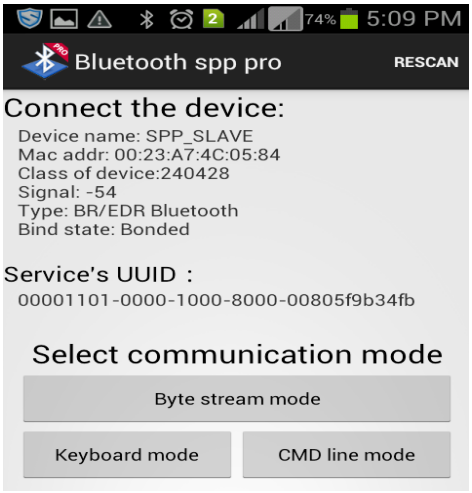
6. After initiating paring, Pairing request will pop-up at smart phone side and accept the pairing request.



7. After successful pair, initiate SPP connection to WiSeConnect module and accept the received pairing request at remote device side.



8. After successful SPP connection, select “Byte stream mode” to send and receive the data.



9. Send some data (Ex: “redpine signals”) from remote device to WiSeConenct device and same data will send back from WiSeConnect device to remote device. Please find below image for sending and receiving data from remote device.