# Simple Peripheral 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 in BTLE peripheral mode and connects with remote BTLE Central device.

**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 device in simple peripheral mode and how to connect from remote Central device.

### 1.1.2 Sequence of Events

This Application explains user how to:

- Set a local name to the device
- Configure the device to advertise
- Continue advertising even after disconnection with the peer

## 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 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 GATT client

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

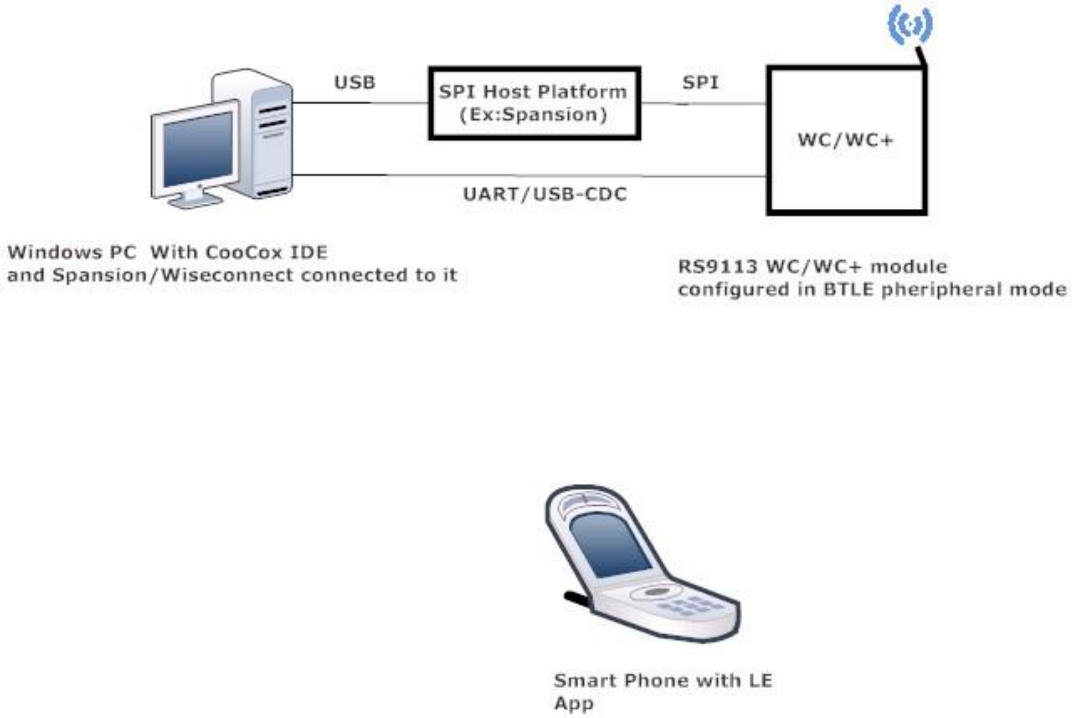


**Figure 1: Setup Diagram**

# 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 ***simple_peripheral*** 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 ***simple_peripheral*** example in Dev-C++ IDE

## 2.2 Configuring the Application

1. Open ***sapis/examples/ble/simple_peripheral/rsi_ble_simple_peripheral.c*** file and update/modify following macros :

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

```
#define RSI_BLE_LOCAL_NAME          "WLAN_BLE_SIMPLE"
```

**RSI_SEL_ANTENNA** refers to the antenna which is to be used by WiSeConnect module.

If user using internal antenna then set,

```
#define RSI_SEL_ANTENNA        RSI_SEL_INTERNAL_ANTENNA
```

If user using external antenna (U.FL connector) then set,

```
#define RSI_SEL_ANTENNA        RSI_SEL_EXTERNAL_ANTENNA
```

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

Following are the event numbers for advertising, connection and Disconnection events.

```
#define RSI_APP_EVENT_ADV_REPORT          0
#define RSI_APP_EVENT_CONNECTED           1
#define RSI_APP_EVENT_DISCONNECTED        2
```

**BT_GLOBAL_BUFF_LEN** refers Number of bytes required by the application and the driver

```
#define BT_GLOBAL_BUFF_LEN         10000
```

1. ***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_DISABLE
#defineRSI_TCP_IP_FEATURE_BIT_MAP  TCP_IP_FEAT_DHCPV4_CLIENT
#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 ***simple_chat*** 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 ***simple_chat*** example in Dev-C++ IDE

3. After the program gets executed, WiSeConnect module will be in Advertising state.
4. Open a LE App in the Smartphone and do the scan.
5. In the App, WiSeConnect module device will appear with the name cofigured in the macro **`RSI_BLE_LOCAL_NAME (Ex: "WLAN_BLE_SIMPLE")`** or sometimes observed as WiSeConenct device as internal name "**`SimpleBLEPeripheral`**".
6. Initiate connection from the mobile App.

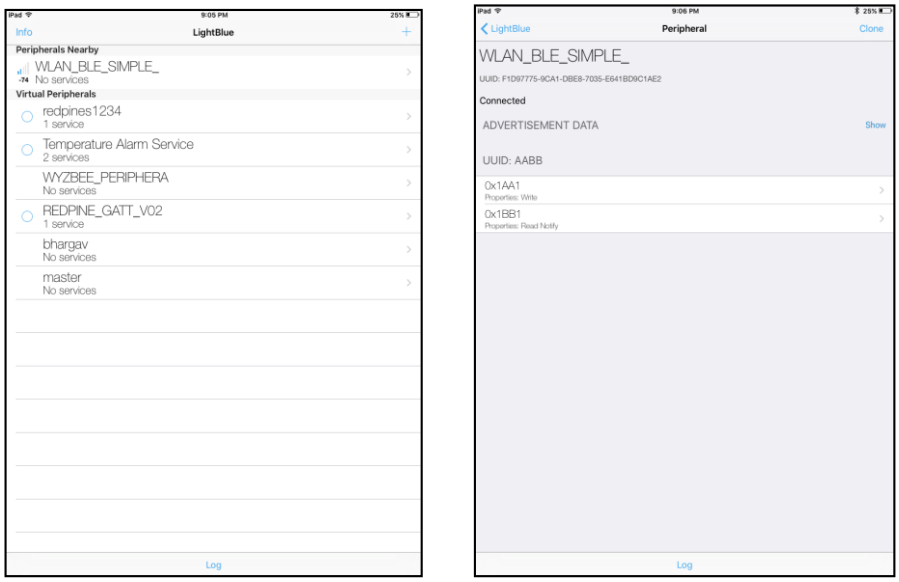


**Figure 2 Scanning for BLE devices and connecting to WLAN_BLE_SIMPLE device**

7. Observe that the connection is established between Smartphone and WiSeConnect module.