WLAN-AP BLE bridge TCPIP bypass 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 a WiFi AP+BTLE co-ex mode in TCP/IP bypass mode.

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

The coex application demonstrates how information can be exchanged seamlessly using two wireless protocols (WLAN and BLE) running in the same device.

In this coex application, WiSeConnect BTLE device connects with remote BTLE device (Smart Phone) and WiSeConnect WiFi interface starts as an Access Point and allow stations (Windows laptop) to connect it.

The coex application has WLAN and BLE tasks and acts as an interface between remote Smartphone BTLE device and connected WiFi Station. Smartphone interacts with BLE task, while connected station interacts with WLAN task. When Smartphone connects and sends message to WiSeConnect device, BLE task accepts and sends to WLAN task, which in turn sends to connected station. Similarly, when PC sends message to WiSeConnect device, the message will be sent to Smartphone via BLE task.

Thus messages can be seamlessly transferred between PC and Smartphone.

1.1.2 Sequence of Events

1.1.2.1 WLAN Task

This Application explains user how to:

- Create device as an Access Point
- Connect stations to WiSeConnect Access Point
- Receive UDP data sent by connected station and forward to BLE task
- Send data received by BLE task to connected station using UDP protocol

1.1.2.2 BLE Task

This Application explains user how to:

- Create chat service
- Configure device in advertise mode
- Connect from Smart phone
- Receive data sent by Smart phone and forward to WLAN task
- Send data received by WLAN task and send to Smart phone

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
- Windows Laptop (STA) with UDP socket application

Note: Download UDP socket application from below link,
http://sourceforge.net/projects/sockettest/files/latest/download

- BTLE supported Smart phone with GATT client application.

Note: Install BLE scanner for GATT client application.

1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Windows Laptop (STA) with UDP socket application

Note: Download UDP socket application from below link,
http://sourceforge.net/projects/sockettest/files/latest/download

- BTLE supported Smart phone with GATT client application.

Note: Install BLE scanner for GATT client application.

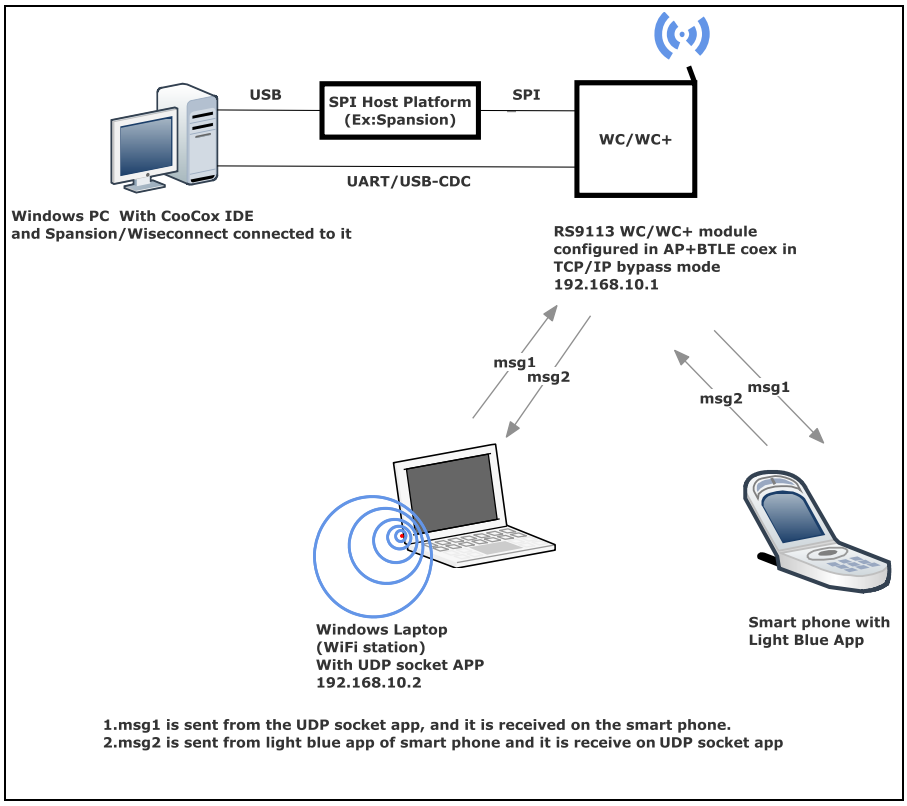


Figure 1 Setup to demonstrate WLAN AP BLE bridge tcpipbypass application

Description:

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 ***wlan_ap_ble_bridge_tcpipbypass*** 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 ***wlan_ap_ble_bridge_tcpipbypass*** example in Dev-C++ IDE

2.2 Configuring the Application

2.2.1 Configuring the WLAN task

1. Open ***sapis/examples/wlan_ble/wlan_ap_ble_bridge_tcpipbypass/rsi_wlan_ap_app.c*** file and update/modify following macros :

SSID refers to the name of the Access point.

```
#define SSID                    "REDPINE_AP"
```

CHANNEL_NO refers to the channel in which AP would be started

```
#define CHANNEL_NO              11
```

Note: Valid values for CHANNEL_NO are 1 to 11 in 2.4GHz and 36 to 48 & 149 to 165 in 2.4GHz. In this example default configured band is 2.4GHz. So, if user wants to use 5GHz band then user has to set RSI_BAND macro to 5GHz band in **sapis/include/rsi_wlan_config.h** file.

SECURITY_TYPE refers to the type of security .Access point supports Open, WPA, WPA2 securities.

Valid configuration is:

RSI_OPEN - For OPEN security mode

RSI_WPA - For WPA security mode

RSI_WPA2 - For WPA2 security mode

```
#define SECURITY_TYPE           RSI_WPA2
```

ENCRYPTION_TYPE refers to the type of Encryption method .Access point supports OPEN, TKIP, CCMP methods.

Valid configuration is:

RSI_CCMP - For CCMP encryption

RSI_TKIP - For TKIP encryption

RSI_NONE - For open encryption

```
#define ENCRYPTION_TYPE      RSI_CCMP
```

PSK refers to the secret key if the Access point to be configured in WPA/WPA2 security modes.

```
#define PSK                  "1234567890"
```

BEACON_INTERVAL refers to the time delay between two consecutive beacons in milliseconds. Allowed values are integers from 100 to 1000 which are multiples of 100.

```
#define BEACON_INTERVAL        100
```

DTIM_INTERVAL refers DTIM interval of the Access Point. Allowed values are from 1 to 255.

```
#define DTIM_INTERVAL          4
```

DEVICE_PORT refers internal UDP server port number

```
#define DEVICE_PORT            5001
```

REMOTE_PORT refers remote UDP server port number.

```
#define REMOTE_PORT            5001
```

REMOTE_IP_ADDRESS refers IP address of the connected STA.

IP address should be in long format and in little endian byte order.

Example: To configure "192.168.10.2" as **REMOTE_IP_ADDRESS** then update the macro as **0x020AA8C0**.

```
#define REMOTE_IP_ADDRESS 0x020AA8C0
```

To configure IP address:

IP address to be configured to the device should be in long format and in little endian byte order.

Example: To configure "192.168.10.1" as IP address, update the macro **DEVICE_IP** as **0x010AA8C0**.

```
#define  DEVICE_IP          0X010AA8C0
```

IP address of the gateway should also be in long format and in little endian byte order

Example: To configure "192.168.10.1" as Gateway, update the macro GATEWAY as **0x010AA8C0**

```
#define  GATEWAY            0x010AA8C0
```

IP address of the network mask should also be in long format and in little endian byte order

Example: To configure "255.255.255.0" as network mask, update the macro **NETMASK** as **0x00FFFFFF**

```
#define   NETMASK            0x00FFFFFF
```

Note: In AP mode, configure same IP address for both **DEVICE_IP** and **GATEWAY** macros

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_PSK
#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
```

2.2.2 Configuring the BLE Application

1. Open ***sapis/examples/wlan_ble/ wlan_ap_ble_bridge_tcpipbypass/rsi_ble_app.c*** file and update/modify following macros,

RSI_BLE_NEW_SERVICE_UUID refers to the attribute value of the newly created service.

```
#define RSI_BLE_NEW_SERVICE_UUID       0xAABB
```

RSI_BLE_ATTRIBUTE_1_UUID refers to the attribute type of the first attribute under this service (**RSI_BLE_NEW_SERVICE_UUID**).

```
#define RSI_BLE_ATTRIBUTE_1_UUID     0x1AA1
```

RSI_BLE_ATTRIBUTE_2_UUID refers to the attribute type of the second attribute under this service (**RSI_BLE_NEW_SERVICE_UUID**).

```
#define RSI_BLE_ATTRIBUTE_2_UUID     0x1BB1
```

RSI_BLE_MAX_DATA_LEN refers to the Maximum length of the attribute data.

```
#define RSI_BLE_MAX_DATA_LEN      20
```

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

```
#define RSI_BLE_ APP_DEVICE_NAME  "WLAN_BLE_SIMPLE_CHAT"
```

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

RSI_BLE_CHAR_SERV_UUID refers to the attribute type of the characteristics to be added in a service.

```
#define RSI_BLE_CHAR_SERV_UUID         0x2803
```

RSI_BLE_CLIENT_CHAR_UUID refers to the attribute type of the client characteristics descriptor to be added in a service.

```
#define RSI_BLE_CLIENT_CHAR _UUID      0x2902
```

RSI_BLE_ATT_PROPERTY_READ is used to set the READ property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_READ       0x02
```

RSI_BLE_ATT_PROPERTY_WRITE is used to set the WRITE property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_WRITE       0x08
```

RSI_BLE_ATT_PROPERTY_NOTIFY is used to set the NOTIFY property to an attribute value.

```
#define RSI_BLE_ATT_PROPERTY_NOTIFY      0x10
```

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

```
#define BT_GLOBAL_BUFF_LEN        10000
```

2.3 Executing the Application

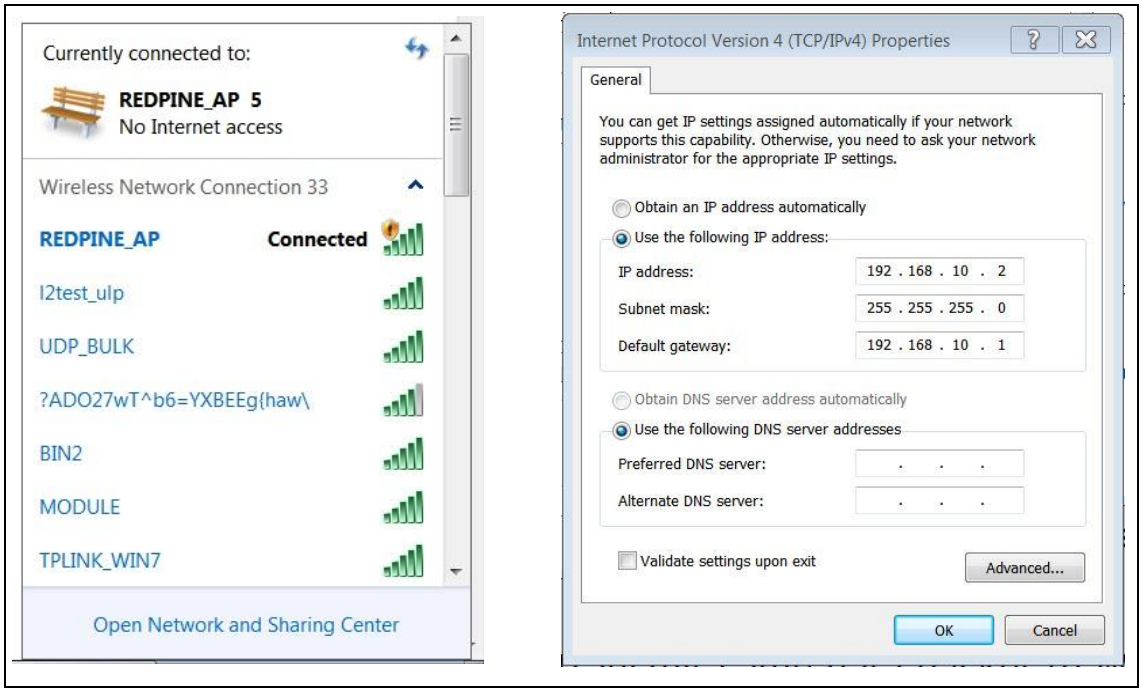
1. **SPI Interface**

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

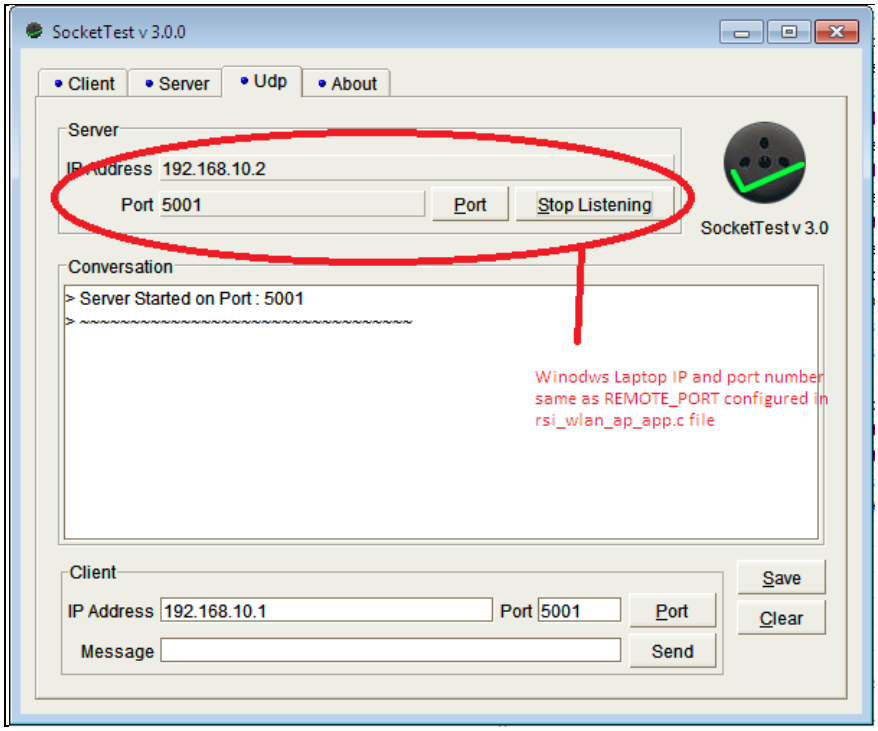
2. **UART/USB-CDC Interface**

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

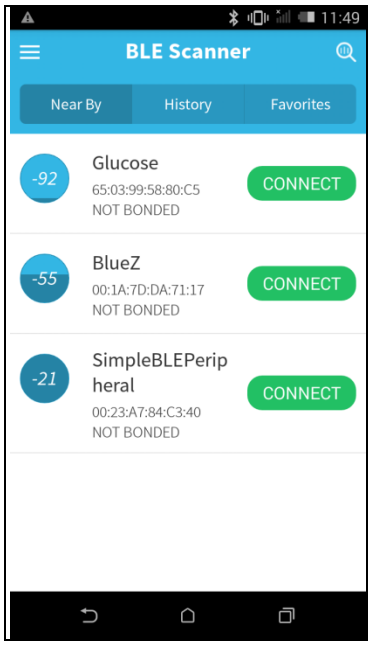
3. After the program gets executed, WiSeConnect BTLE is in Discoverable state and WLAN will create as an Access Point and opens UDP server socket on port number **DEVICE_PORT**.
4. From Windows Laptop (STA), do scan and connect to WiSeConnect Access point (Ex: “REDPINE_AP”) and assign static IP address to laptop which is same as **REMOTE_IP_ADDRESS** configured in rsi_wlan_ap_app.c file.



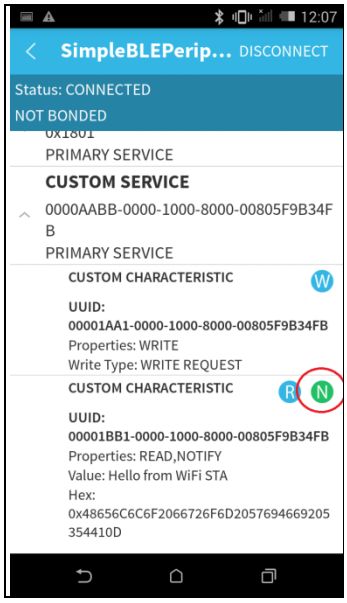
5. Install UDP socket application in Windows laptop and open UDP server socket on port number **REMOTE_PORT (Ex: 5001)** to receive the data sent by BTLE remote device.



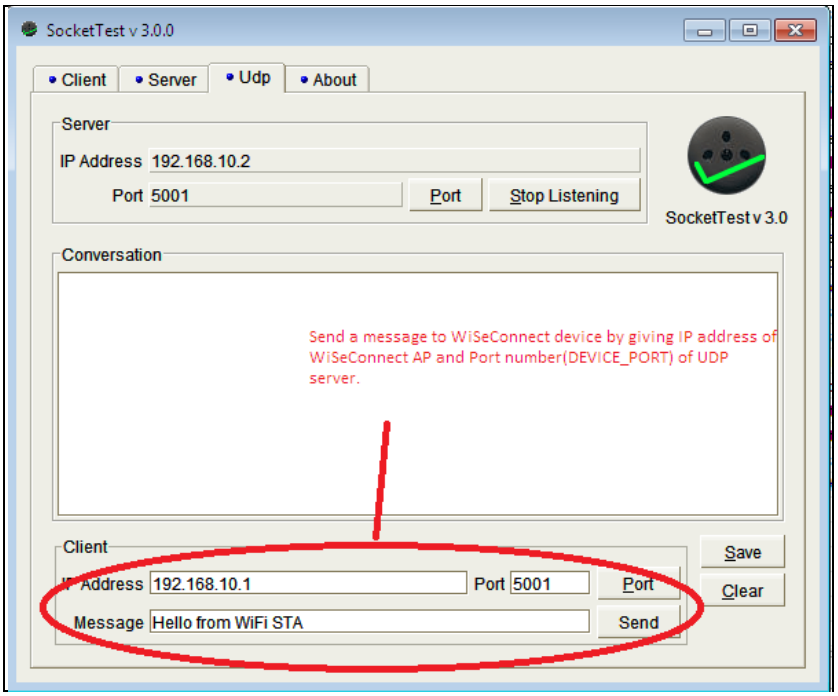
6. Open a BLE scanner App in the Smartphone and do Scan.
7. In the App, WiSeConnect module device would appear with the name configured in the macro **RSI_BLE_APP_SIMPLE_CHAT (Ex: "WLAN_BLE_SIMPLE_CHAT")** or sometimes observed as WiSeConenct device as internal name "**SimpleBLEPeripheral**".

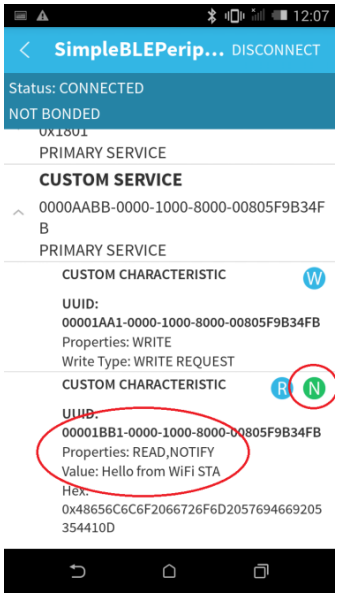


8. Initiate connection from the App.
9. After successful connection, LE scanner displays the supported services of WiSeConnect module.
10. Select the attribute service which is added **RSI_BLE_NEW_SERVICE_UUID** (Ex: 0xAABB) and enable Notification for attribute UUID **RSI_BLE_ATTRIBUTE_2_UUID (Ex: 0x1BB1)** to receive data sent by WiFi STA.



11. Successful TCP connection, send a message (Ex: “Hello from WiFi STA”) from UDP client (from laptop) to WiSeConnect device. WiSeConnect device forwards the received message from Windows laptop to remote BTLE device which is connected to WiSeConnect BTLE device over BTLE protocol. User can observe the message notification on attribute UUID **RSI_BLE_ATTRIBUTE_2_UUID (Ex: 0x1BB1)** in BTLE scanner app.

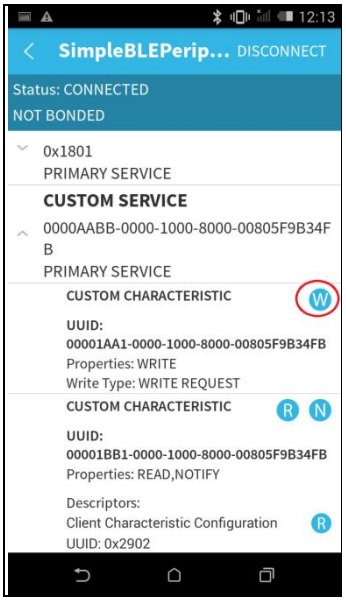


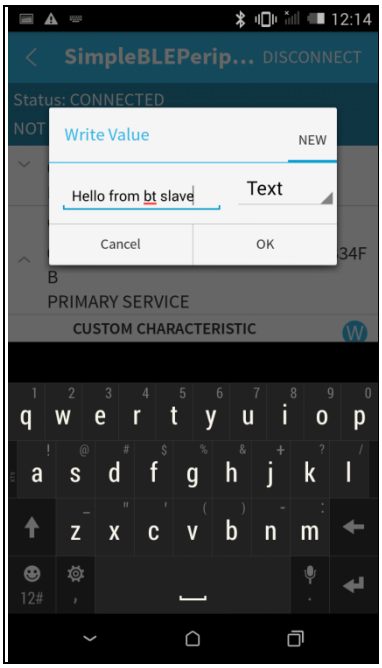


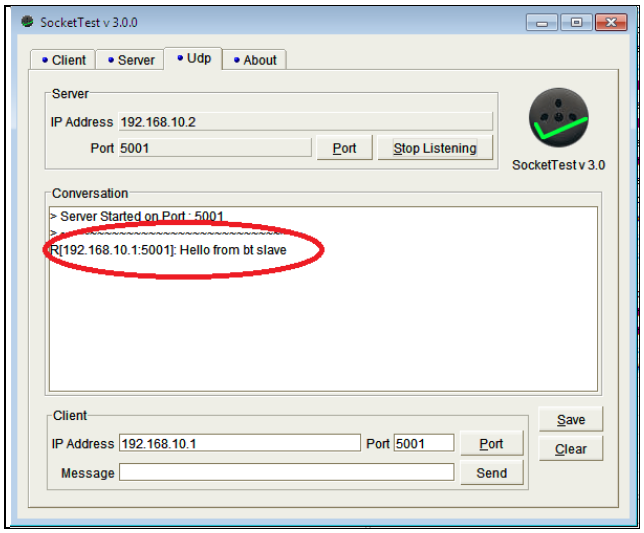
Note: rsi_wlan_app_send_to_btle() function defined in rsi_ble_app.c to send message from WLAN task to BTLE task

12. Now send a message (Ex: “Hello from bt slave”) from GATT client (from smart phone BLE scanner app) using attribute RSI_BLE_ATTRIBUTE_1_UUID (Ex: 0x1AA1) to WiSeConnect device. WiSeConnect device forwards the received message from

BTLE remote device to WiFi STA which is connected to WiSeConnect Access point over WiFi protocol. User can observe the message on UDP socket application.







Note: rsi_bt_app_send_to_wlan() function defined in rsi_wlan_ap_app.c to send message from BTLE task to WLAN task