WLAN-STATION BLE-Dual Role bridge Application

User guide
Version 0.1

December 2015

Redpine Signals, Inc.

2107 N. First Street, #680

San Jose, CA 95131.

Tel: (408) 748-3385

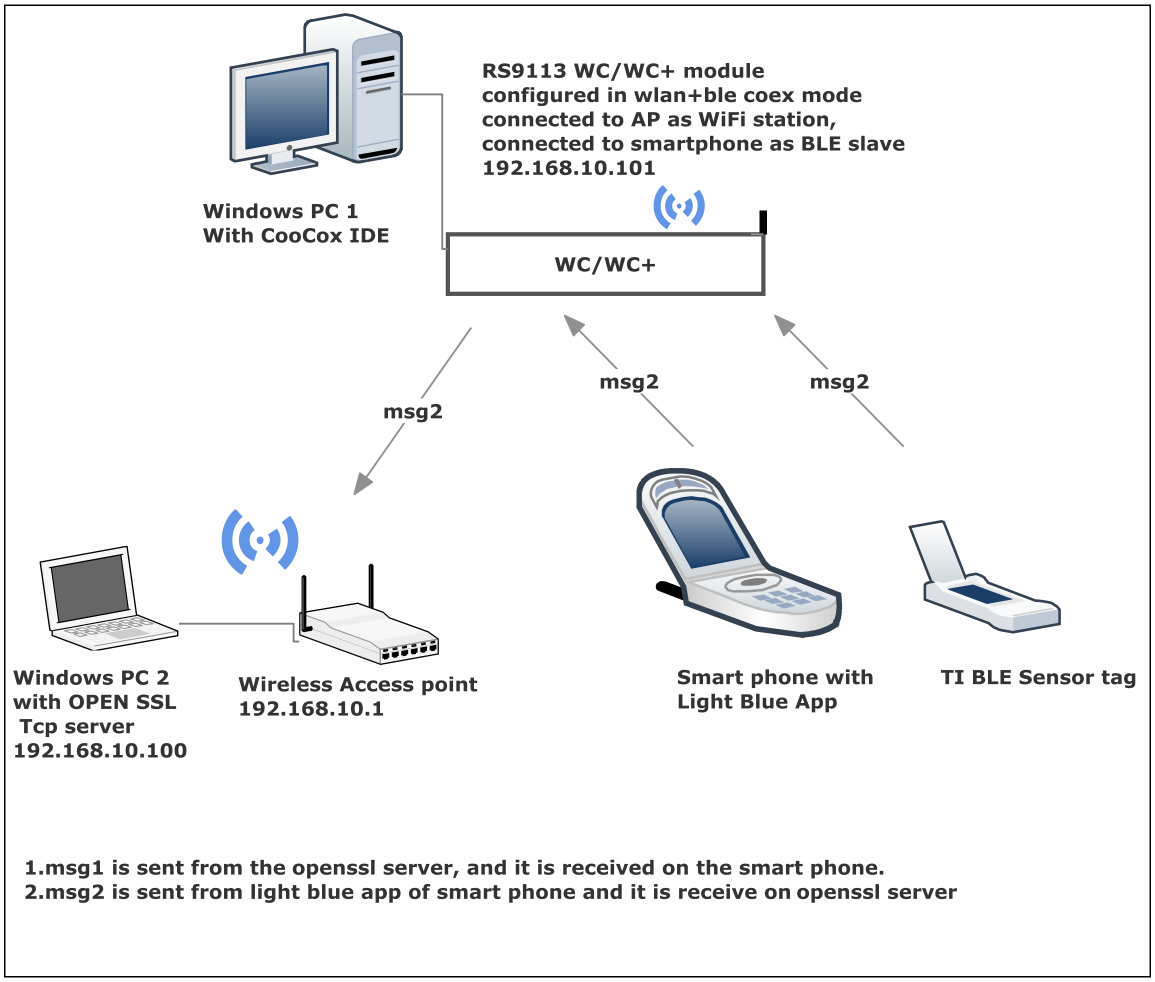
This project is applicable to all the WiSeConnect variants like WiSeConnect Plus, WiSeMCU and WYZBEE. The term WiSeConnect refers to its appropriate variant.

Application Overview:

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

Setup required:

1. Windows PC with Coocox IDE
2. WiSeConnect device
3. Smart phone/tablet with LE Application (Ex: Light blue App in iPad mini)
4. WLAN Access Point and a Windows PC
5. TI simple sensor tag or 3rd party BLE dongle as a slave.


RS9113 WC/WC+ module
configured in wlan+ble coex mode
connected to AP as WiFi station,
connected to smartphone as BLE slave
192.168.10.101
Windows PC 1
With CooCox IDE
WC/WC+
msg2
msg2
msg2
Windows PC 2
with OPEN SSL
Tcp server
192.168.10.100
Wireless Access point
192.168.10.1
Smart phone with
Light Blue App
TI BLE Sensor tag
1.msg1 is sent from the openssl server, and it is received on the smart phone.
2.msg2 is sent from light blue app of smart phone and it is receive on openssl server

Description:

The coex application has WLAN and BLE tasks and acts as an interface between Smartphone and sensor tag and PC.

Smartphone and sensor tag interacts with BLE task, while PC[1] interacts with WLAN task.

When Smartphone and sensor tag connects and sends messages/notifications to WiSeConnect, BLE task accepts and sends to WLAN task, which in turn sends to Access Point connected PC.

Thus messages can be seamlessly transferred from Smartphone and sensor tag to Windows PC.

Details of the Application:

[1] Both PC and WiSeConnect WLAN would be connected to a Wireless Access Point, thus both are connected together wirelessly

WiSeConnect WLAN acts as a Station and connects to an Access Point

WiSeConnect BLE acts as a Peripheral (Slave) device and Central (Master), while Smart phone acts as a Central (Master) device and Sensor tag acts as a Peripheral (Slave) device.

Initially, proprietary Simple chat service is created at GATT Server (WiSeConnect device) to facilitate message exchanges.

- The WLAN task (running in WiSeConnect device) mainly includes following steps.
  1. Connects to a Access Point
  2. Exchanges data over SSL socket with the peer(Windows PC)
- The BLE task (running in WiSeConnect device) mainly includes following steps.
  1. Creates chat service
  2. Configures the device to scanning.
  3. After connection of slave configures the device to Advertise

WLAN and BLE tasks forever run in the application to serve the asynchronous events.

Configuring the WLAN task:

Edit the **`rsi_wlan_app.c`** file in the following path to establish connection with the Access-point.

`sapis/examples/wlan_ble/wlan_station_ble_dual_role_bridge/`

1. From given configuration,

   SSID refers to the Access point to which user wants to connect.

   SECURITY_TYPE is the security type of the Access point.

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

```
#define SSID                "<ap_name>"
#define SECURITY_TYPE        <security-type>
#define PSK                   ""
```

2. Load the SSL CA- certificate using rsi_wlan_set_certificate API after wireless initialization.

> **Note**: rsi_wlan_set_certificate expects the certificate in the form of linear array. Python script is provided in the release package named "`certificate_script.py`" in the following path "`sapis/examples/utilities/certificates`" to convert the pem certificate into linear array
>
> Example: If the certificate is ca-cert.pem, give the command as
>
> ```
> python certificate_script.py ca-cert.pem
> ```
>
> The script will generate cacert.pem, in which one linear array named *cacert* contains the certificate

3. Enable/Disable DHCP mode

    `1` – Enables DHCP mode (gets the IP from DHCP server)

    `0` – Disables DHCP mode

```
#define DHCP_MODE 1
```

4. If DHCP mode is disabled, then change the following macros to configure static IP address

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

    Example: To configure "192.168.10.101" as IP address, update the macro **`DEVICE_IP`** as **`0x650AA8C0`**.

```
#define  DEVICE_IP          0x650AA8C0
```

    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
```

5. To establish TCP connection and transfer data to the remote socket configure the below macros. If SSL is enabled, open the socket with protocol type as 1.

    Internal socket port number.

```
#define DEVICE_PORT 5001
```

    Port number of the remote server

```
#define SERVER_PORT 5001
```

IP address of the remote server

```
#define SERVER_IP_ADDRESS 0x650AA8C0
```

6. Include rsi_wlan_app.c , rsi_ble_app.c and main.c files in the project, build and launch the application

7. Open SSL server socket on remote machine

   For example, to open SSL socket with port number 5001 on remote side, use the command as given below

   ```
   openssl s_server  -accept<SERVER_PORT> -cert
   <server_certificate_file_path> -key
   <server_key_file_path> -tls<tls_version>
   ```

   Example: `openssl s_server -accept 5001 -cert server-cert.pem -key server-key.pem -tls1_2`

   Note: All the certificates are given in the release package

   path: `sapis/examples/utilities/certificates`

8. Enable/Disable power save

   1 – Enables Power save mode

   0 – Disables Power save mode

   ```
   #define WLAN_POWER_SAVE 0
   ```

   By default Power save is disabled.

Update the Wlan configuration file:

`sapis/include/rsi_wlan_config.h`

CONCURRENT_MODE	DISABLE
RSI_FEATURE_BIT_MAP	FEAT_SECURITY_OPEN
RSI_TCP_IP_BYPASS	DISABLE
RSI_TCP_IP_FEATURE_BIT_MAP	(TCP_IP_FEAT_DHCPV4_CLIENT | TCP_IP_FEAT_SSL | TCP_IP_FEAT_SINGLE_SSL_SOCKET | TCP_IP_TOTAL_SOCKETS_2)
RSI_CUSTOM_FEATURE_BIT_MAP	0
RSI_BAND	RSI_BAND_2P4GHZ

Configuring the BLE Application:

Edit the **`rsi_ble_app.c`** file in the following path of the Application

`sapis/examples/wlan_ble/wlan_station_ble_dual_role_bridge/`

Configure the below macros in the Application file.

1. RSI_BLE_NEW_SERVICE_UUID - The attribute value of the newly created service. Ex: 0xAABB
2. RSI_BLE_ATTRIBUTE_1_UUID - The attribute type of the first attribute under this Service. Ex: 0x1AA1
3. RSI_BLE_ATTRIBUTE_2_UUID - The attribute type of the second attribute under this Service. Ex: 0x1BB1
4. RSI_BLE_MAX_DATA_LEN - Maximum length of the attribute data(limited to max of 20 bytes)
5. RSI_BLE_APP_DEVICE_NAME - Name of the WiSeConnect device to appear during Scanning by peer devices.
6. BLE_PS_ENABLE – To Enable/Disable power save.
7. RSI_BLE_DEV_ADDR – Address of the peer device to connect.
8. BLE_DUAL_ROLE_FIRST_MASTER – To create local device as a master first.

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

9. RSI_BLE_ATT_PROPERTY_READ – Used to set read property to an attribute value.
10. RSI_BLE_ATT_PROPERTY_WRITE - Used to set write property to an attribute value.
11. RSI_BLE_ATT_PROPERTY_NOTIFY - Used to set notify property to an attribute value.
12. RSI_BLE_CHAR_SERV_UUID - The attribute type of the characteristics to be added in a service. Ex: 0x2803
13. RSI_BLE_CLIENT_CHAR_UUID - The attribute type of the client characteristics descriptor to be added in a service characteristic. Ex: 0x2902
14. BT_GLOBAL_BUFF_LEN – Number of bytes required for the Application and the Driver.

Executing the coex Application:

1. Connect WiSeConnect device to the Windows PC running Cocoox IDE.
2. Configure the macros in the files located at

   `sapis/examples/wlan_ble/wlan_station_ble_dual_role_bridge/rsi_ble_app.c`

   `sapis/examples/wlan_ble/wlan_station_ble_dual_role_bridge/rsi_wlan_app.c`

3. Build and launch the application.
4. Advertise 3rd party TI sensor tag.
5. After the program gets executed, If BLE_DUAL_ROLE_FIRST_MASTER macro as 1 then WiSeConnect BLE is in scanning state and it creates a connection with TI sensor tag. If BLE_DUAL_ROLE_FIRST_MASTER macro as 0 then WiSeConnect BLE is in Advertising state and WLAN has established SSL socket with peer(PC).
6. Open a LE App in the Smartphone and do scan.
7. In the App, WiSeConnect BLE would appear with the name configured in the macro RSI_BLE_APP_DEVICE_NAME.
8. Initiate BLE connection from the App.
9. After BLE connection is established, send a message or notification from the App to WiSeConnect BLE. Observe this message in the PC connected via SSL socket with WiSeConnect WLAN.
10. rsi_ble_app_send_to_wlan() function defined in rsi_wlan_app.c to send message from BLE task to WLAN task.
11. With the help of wlan task, message is transferred to PC.