UDP Client Socket 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 standard UDP client socket and sends data to remote UDP socket.

Disclaimer:

Table of Contents

Table of Figures

Table of Tables

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 UDP Protocol Overview

UDP(USER Datagram protocol) is a connectionless and non-stream oriented protocol for transferring data in either direction between a pair of users.

In UDP there is no guarantee that the messages or packets sent would reach at all.

No handshake in UDP Protocol because it is connectionless protocol

1.2 Application Overview

1.2.1 Overview

The UDP client application demonstrates how to open and use a standard UDP client socket and sends data to UDP server socket.

Once it is configured as UDP client it can establish and maintain a network conversation by means of application program for exchanging of data.

1.2.2 Sequence of Events

This Application explains user how to:

- Connect the Device to an Access point and get IP address through DHCP
- Open UDP Server socket at Access point using iperf application.
- Open UDP client socket in device
- Send data from WiSeConnect device to remote peer using opened UDP socket.

1.3 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:

1.3.1 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
- WiFi Access point
- Windows PC2
- UDP server application running in Windows PC2 (This application uses iperf application to open UDP server socket)

1.3.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- WiFi Access point

- Windows PC2
- UDP server application running in Windows PC2 (This application uses iperf application to open UDP server socket)

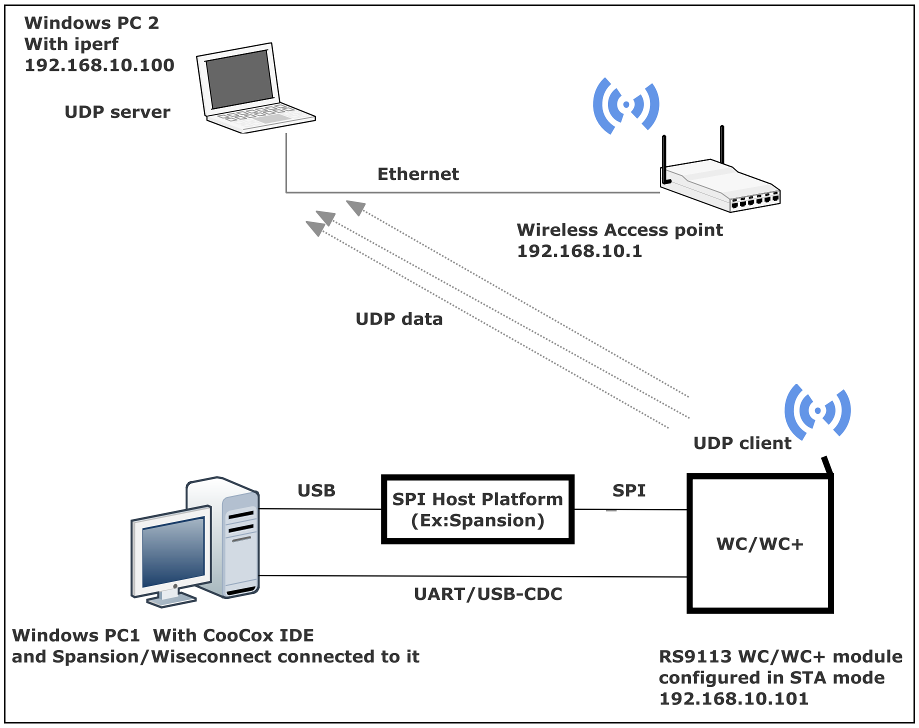


Figure 1: Setup Diagram

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

2.2 Configuring the Application

1. Open ***sapis/examples/wlan/udp_client/rsi_udp_client.c*** file and update/modify following macros,

SSID refers to the name of the Access point.

```
#define SSID                    "<ap name>"
```

CHANNEL_NO refers to the channel in which device should scan. If it is 0, device will scan all channels

```
#define CHANNEL_NO          0
```

SECURITY_TYPE refers to the type of security. In this application STA supports Open, WPA-PSK, WPA2-PSK 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_OPEN
```

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

```
#define PSK                         "<psk>"
```

DEVICE_PORT port refers UDP client port number

```
#define DEVICE_PORT             5001
```

SERVER_PORT port refers remote UDP server which is opened in windows PC2.

```
#define SERVER_PORT             5001
```

SERVER_IP_ADDRESS refers remote peer IP address to connect with TCP server socket.

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

Example: To configure "192.168.10.100" as IP address, update the macro **DEVICE_IP** as **0x640AA8C0**.

```
      #define SERVER_IP_ADDRESS      0x640AA8C0
```

NUMEBR_OF_PACKETS refers how many packets to send from device to UDP server.

```
      #define NUMBER_OF_PACKETS     1000
```

Application memory length which is required by the driver

```
      #define  GLOBAL_BUFF_LEN       8000
```

To configure IP address:

DHCP_MODE refers whether IP address configured through DHCP or STATIC

```
      #define DHCP_MODE   1
```

Note:

If the user wants to configure STA IP address through DHCP then set **DHCP_MODE** to 1 and skip configuring the following **DEVICE_IP**, **GATEWAY** and **NETMASK** macros.
(Or)
If the user wants to configure STA IP address through STATIC then set DHCP_MODE macro to "0" and configure following **DEVICE_IP**, **GATEWAY** and **NETMASK** macros.

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

Example: To configure "192.168.10.10" as IP address, update the macro **DEVICE_IP** as **0x0A0AA8C0**.

```
      #define  DEVICE_IP           0X0A0AA8C0
```

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
```

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

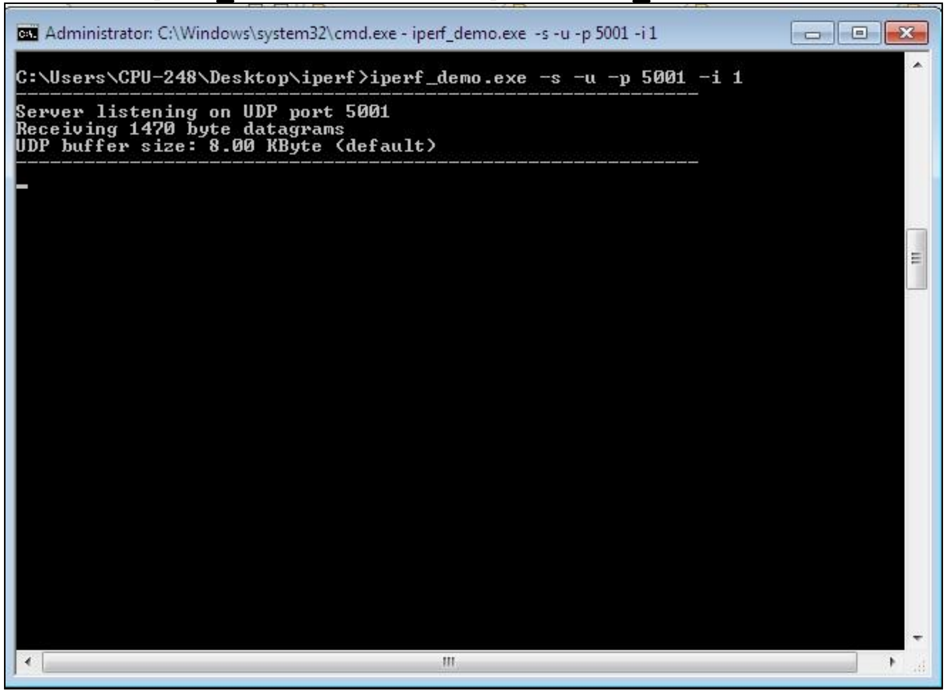
```
#define  CONCURRENT_MODE               RSI_DISABLE
#define  RSI_FEATURE_BIT_MAP           FEAT_SECURITY_OPEN
#define  RSI_TCP_IP_BYPASS             RSI_DISABLE
#define  RSI_TCP_IP_FEATURE_BIT_MAP TCP_IP_FEAT_DHCPV4_CLIENT
 #define  RSI_CUSTOM_FEATURE_BIT_MAP  0
```

```
#define  RSI_BAND                        RSI_BAND_2P4GHZ
```

2.3 Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect WiSeConnect device in STA mode.
2. Open UDP server application using iperf application in Windows PC2 which is connected to Access point through LAN.

```
iperf_demo.exe -s -u -p <SERVER_PORT> -i 1
```



Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -u -p 5001 -i 1

```
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -u -p 5001 -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
```

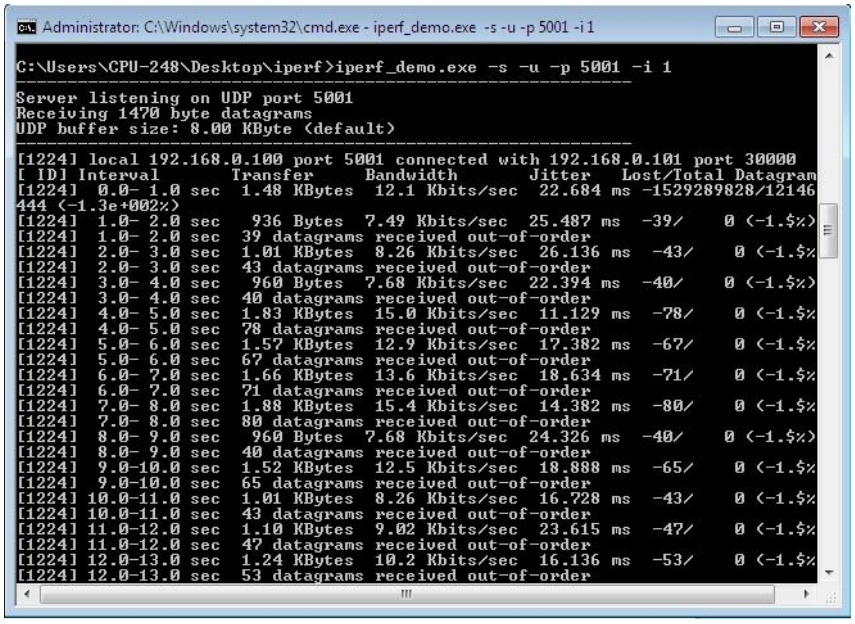
3. **SPI Interface**

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

4. **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 ***udp_client*** example in Dev-C++ IDE

5. After program gets executed, WiSeConnect Device would scan and connect to Access point and get IP.
6. After successful connection, WiSeConenct STA connects to UDP server socket opened on Windows PC2 using UDP client socket and sends configured `NUMBER_OF_PACKETS` to remote UDP server. Please refer the given below image for reception of UDP data on UDP server.



Administrator: C:\Windows\system32\cmd.exe - iperf_demo.exe -s -u -p 5001 -i 1

```
C:\Users\CPU-248\Desktop\iperf>iperf_demo.exe -s -u -p 5001 -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
------------------------------------------------------------
[1224] local 192.168.0.100 port 5001 connected with 192.168.0.101 port 30000
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagram
[1224]  0.0- 1.0 sec  1.48 KBytes  12.1 Kbits/sec  22.684 ms -1529289828/12146
444 (-1.3e+002%)
[1224]  1.0- 2.0 sec   936 Bytes  7.49 Kbits/sec  25.487 ms  -39/    0 (-1.$%)
[1224]  1.0- 2.0 sec  39 datagrams received out-of-order
[1224]  2.0- 3.0 sec  1.01 KBytes  8.26 Kbits/sec  26.136 ms  -43/    0 (-1.$%
[1224]  2.0- 3.0 sec  43 datagrams received out-of-order
[1224]  3.0- 4.0 sec   960 Bytes  7.68 Kbits/sec  22.394 ms  -40/    0 (-1.$%)
[1224]  3.0- 4.0 sec  40 datagrams received out-of-order
[1224]  4.0- 5.0 sec  1.83 KBytes  15.0 Kbits/sec  11.129 ms  -78/    0 (-1.$%
[1224]  4.0- 5.0 sec  78 datagrams received out-of-order
[1224]  5.0- 6.0 sec  1.57 KBytes  12.9 Kbits/sec  17.382 ms  -67/    0 (-1.$%
[1224]  5.0- 6.0 sec  67 datagrams received out-of-order
[1224]  6.0- 7.0 sec  1.66 KBytes  13.6 Kbits/sec  18.634 ms  -71/    0 (-1.$%
[1224]  6.0- 7.0 sec  71 datagrams received out-of-order
[1224]  7.0- 8.0 sec  1.88 KBytes  15.4 Kbits/sec  14.382 ms  -80/    0 (-1.$%
[1224]  7.0- 8.0 sec  80 datagrams received out-of-order
[1224]  8.0- 9.0 sec   960 Bytes  7.68 Kbits/sec  24.326 ms  -40/    0 (-1.$%)
[1224]  8.0- 9.0 sec  40 datagrams received out-of-order
[1224]  9.0-10.0 sec  1.52 KBytes  12.5 Kbits/sec  18.888 ms  -65/    0 (-1.$%
[1224]  9.0-10.0 sec  65 datagrams received out-of-order
[1224] 10.0-11.0 sec  1.01 KBytes  8.26 Kbits/sec  16.728 ms  -43/    0 (-1.$%
[1224] 10.0-11.0 sec  43 datagrams received out-of-order
[1224] 11.0-12.0 sec  1.10 KBytes  9.02 Kbits/sec  23.615 ms  -47/    0 (-1.$%
[1224] 11.0-12.0 sec  47 datagrams received out-of-order
[1224] 12.0-13.0 sec  1.24 KBytes  10.2 Kbits/sec  16.136 ms  -53/    0 (-1.$%
[1224] 12.0-13.0 sec  53 datagrams received out-of-order
```