UDP Server 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 server socket and receives data from remote UDP socket.

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   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.

1.2   Application Overview

1.2.1   Overview

The UDP server application demonstrates how to open and use a standard UDP server socket and receives data on socket sent by remote peer.

Once it is configured as UDP server 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 in device
- Send UDP data from remote peer to WiSeConnect device on opened port using UDP client socket
- Receive data from UDP client 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 client application running in Windows PC2 (This application uses iperf application to open UDP client socket)

1.3.2   UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- WiFi Access point
- Windows PC2

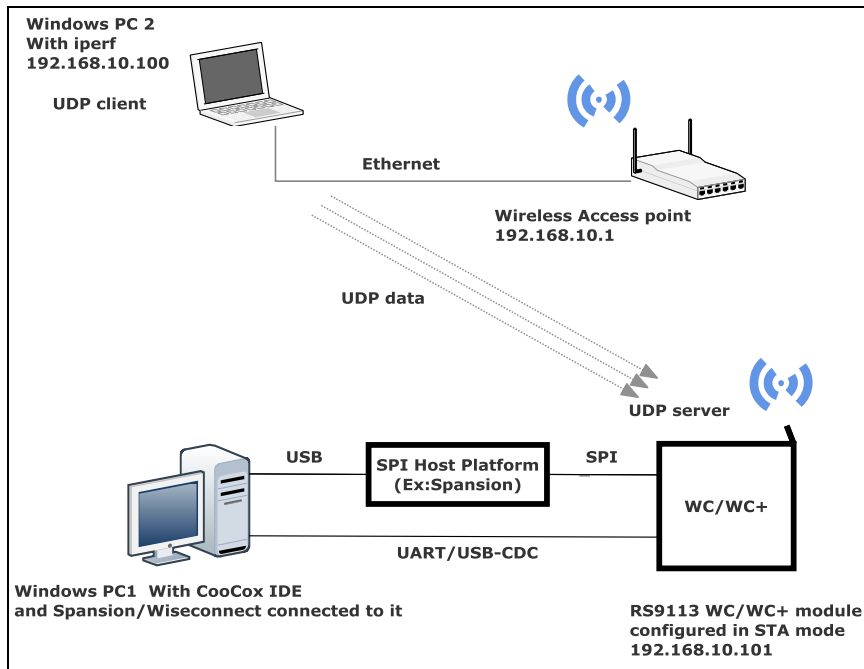- UDP client application running in Windows PC2 (This application uses iperf application to open UDP client socket)

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

2.2 Configuring the Application

1. Open *sapis/examples/wlan/udp_server/rsi_udp_server.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 TCP client port number

```
     #define DEVICE_PORT        5001
```

Receive data length

```
     #define RECV_BUFFER_SIZE        <recv_buf_size>
```

NUMEBR_OF_PACKETS refers how many packets to receive from UDP client

```
        #define  NUMBER_OF_PACKETS      <no of packets>
```

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    0
```

> **Note:** Configure STATIC IP to WiSeConnect device. So that user knows the IP address of WiSeConnect device to establish UDP connection from remote peer. In case of DHCP, User has to know the assigned IP by parsing IPCONF response.

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

Example: To configure "192.168.0.101" as IP address, update the macro **DEVICE_IP** as **0x0A0AA8C0**.

```
        #define  DEVICE_IP         0X6500A8C0
```

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          0x0100A8C0
```

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. **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_server* example in CooCox IDE.
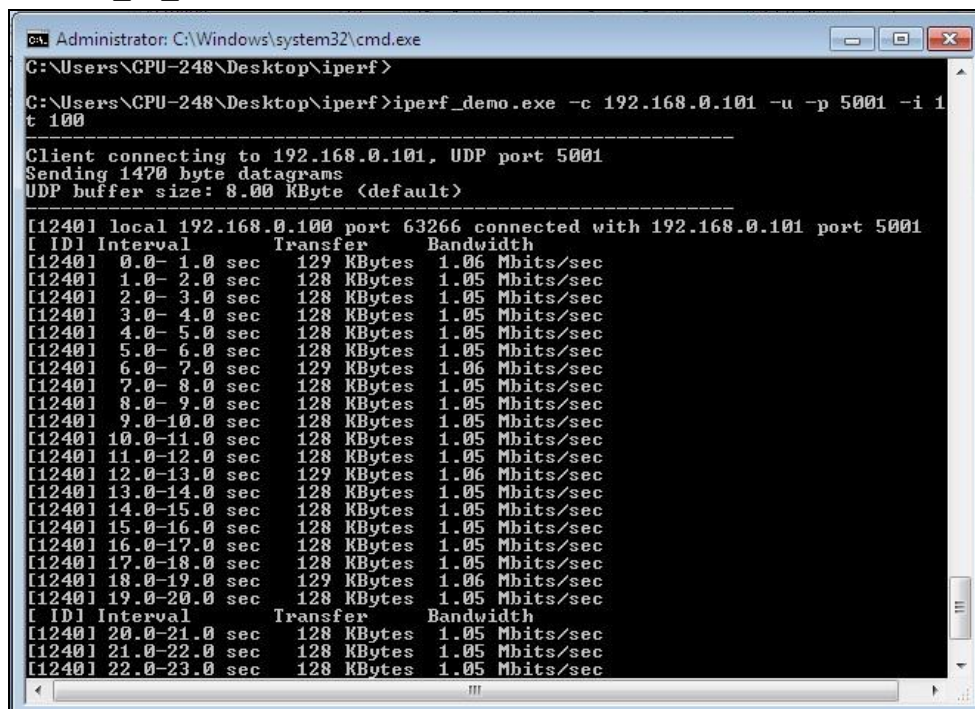
3. **UART/USB-CDC Interface**

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

4. After the program gets executed, WiSeConnect Device will be connected to Access point having the configuration as same as that of in the application and get IP.

5. Open UDP client using iperf from windows PC2 and connect to UDP server opened on WiSeConnect device on port number **DEVICE_PORT** using the following command,

   `Iperf_demo.exe –c <DEVICE_IP>  -u -p <DEVICE_PORT>  -i 1 –t 100`

6. WiSeConnect Device will receive the number of packets configured in **NUMBER_OF_PACKETS** from iperf UDP client.