

Firmware Upgradation From Server 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 station and upgrades firmware through remote TCP server.

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.

Copyright © 2015 Redpine Signals, Inc. All rights reserved.

Table of Contents

1	Introduction	4
1.1	Application Overview	4
1.1.1	Overview.....	4
1.1.2	Sequence of Events.....	4
1.2	Application Setup	4
1.2.1	SPI based Setup Requirements	4
1.2.2	UART/USB-CDC based Setup Requirements	5
2	Configuration and Execution of the Application	6
2.1	Initializing the Application	6
2.1.1	SPI Interface.....	6
2.1.2	UART/USB-CDC Interface.....	6
2.2	Configuring the Application	6
2.3	Executing the Application	8

Table of Figures

Figure 1: Setup Diagram.....	5
------------------------------	---

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 upgrade new firmware to WiSeConnect device using remote TCP server.

In this application, WiSeConnect device connects to Access Point and establishes TCP client connection with TCP server opened on remote peer. After successful TCP connection, application sends the firmware file request to remote TCP server and server responds with Firmware file and waits for the next firmware file request. Once firmware file receives from the TCP server, Application loads the firmware file into device using firmware upgrade API and gets next firmware file from TCP server. After successful firmware upgrade, firmware upgrade API returns 0x03 response.

1.1.2 Sequence of Events

This Application explains user how to:

- Configure as station mode
- Open TCP server socket at Access Point
- Connect to Access Point and open TCP client socket
- Request Firmware file from remote server
- Send firmware file from remote server
- Upgrade the received Firmware into the device.

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
- Wireless Access point
- Linux PC with TCP server application (TCP server application providing as part of release package)

Note: TCP server application providing in release package in the following path:
[sapis/examples/wlan/firmware_upgradation/firmware_upgrade_tcp_server.c](#)

1.2.2 UART/USB-CDC based Setup Requirements

- Windows PC with Dev-C++ IDE
- WiSeConnect device
- Wireless Access point
- Linux PC with TCP server application (TCP server application providing as part of release package)

Note: TCP server application providing in release package in the following path:
sapis/examples/wlan/firmware_upgradation/firmware_upgrade_tcp_server.c

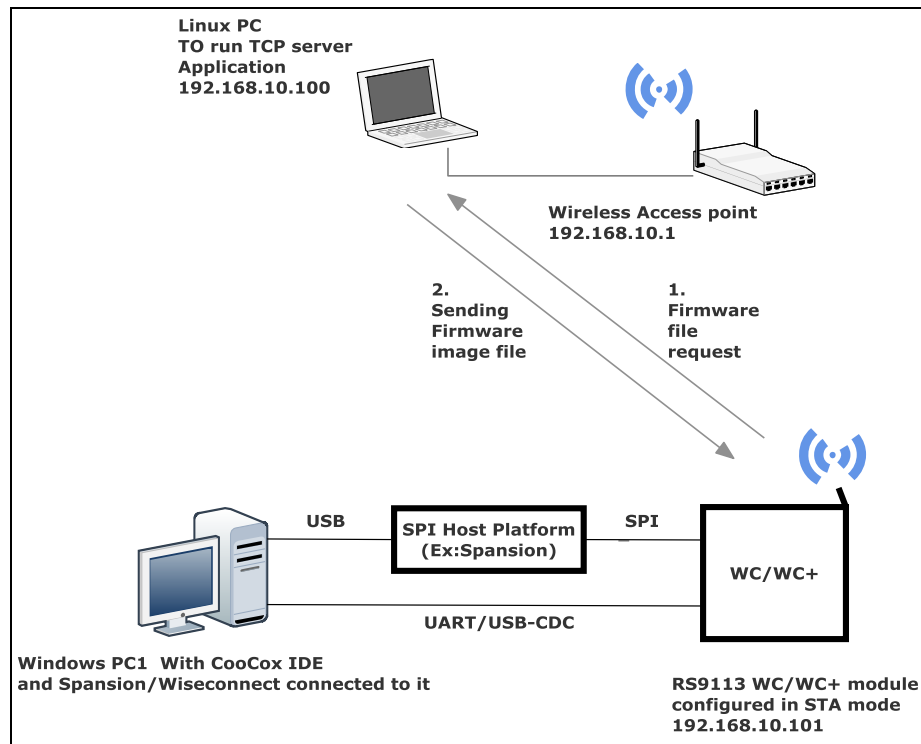


Figure 1: Setup Diagram

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 *firmware_upgrade* example in CoCoX 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 *firmware_upgrade* example in Dev-C++ IDE

Note: Do not add *firmware_upgrade_tcp_server.c* file which is present in Firmware_Upgrade folder to Spansion/Dev-C++ project.

2.2 Configuring the Application

1. Open *sapis/examples/wlan/firmware_upgrade/rsi_firmware_upgradation_app.c* file and update/modify following macros:

SSID refers to the name of the Access point.

```
#define SSID "<ap_name>"
```

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
```

SERVER_PORT port refers remote TCP server port number which is opened in Linux PC.

```
#define SERVER_PORT 5001
```

SERVER_IP_ADDRESS refers remote peer IP (Linux PC) address to connect with TCP server socket.

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

Example: To configure "192.168.0.100" as remote IP address, update the macro **SERVER_IP_ADDRESS** as **0x6400A8C0**.

```
#define SERVER_IP_ADDRESS 0x6400A8C0
```

RECV_BUFFER_SIZE refers Memory for receive data

```
#define RECV_BUFFER_SIZE 1027
```

To configure IP address

DHCP_MODE refers whether IP address configured through DHCP or STATIC

```
#define DHCP_MODE 1
```

Note: If 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 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.

The IP address needs to be configuring to the device 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
```

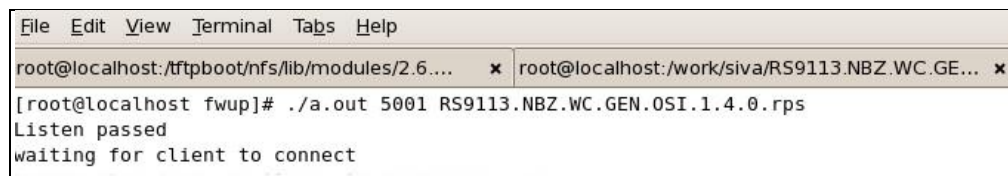
2.3 Executing the Application

1. Configure the Access point in OPEN/WPA-PSK/WPA2-PSK mode to connect WiSeConnect device in STA mode.
2. Copy TCP server application present in release package

"*sapis/examples/wlan/firmware_upgradation/firmware_upgarde_tcp_server.c*" into Linux PC which is connected to Access point through LAN. Compile and run by providing port number and Firmware file path.

gcc firmware_upgarde_tcp_server.c

./a.out 5001 RS9113.NBZ.WC.GEN.OSI.x.x.x.rps



```
File Edit View Terminal Tabs Help
root@localhost:/work/siva/RS9113.NBZ.WC.GE... x
[root@localhost fwup]# ./a.out 5001 RS9113.NBZ.WC.GEN.OSI.1.4.0.rps
Listen passed
waiting for client to connect
```

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

5. After the program gets executed, device connects to AP and open TCP client socket.
6. After TCP connection established with remote server, application sends firmware file request to the server.
7. Server receive request and sends Firmware file in chunks.
8. After receiving chunk from remote server, application again sends firmware request to server. Server will wait for the firmware request from WiSeConnect device before sending next chunk.
9. Packet is sent to the device in chunks as shown in the given below figure. After successful upgradation in TCP server terminal shows "reach end of file".


```
File Edit View Terminal Tabs Help
root@localhost:/ftpboot/nfs/lib/modules/2.6... x root@localhost:/work/siva/RS9113.NBZ.WC.GE... x
size of data1==1024
send returns 1027
Pkt sent no:1529
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1530
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1531
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1532
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1533
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1534
waiting for recv
recv length == 0x3
size of data1==1024
send returns 1027
Pkt sent no:1535
waiting for recv
recv length == 0x3
reach end of file
█
```

10. In Application *rsi_firmware_upgradation_app.c*, *rsi_fwup_load* API returns 0x03 response after successful firmware up gradation and closes TCP client socket.

Note: After Firmware upgradation, Device needs to be reboot to get effective of new firmware file. After reboot, Device will take few minutes to give CARD READY indication after first reboot.