NEWRACOM
A New semiconductor Company for New Era

# NRC7292 Evaluation Kit User Guide

## (Host Mode)

**Ultra-low power & Long-range Wi-Fi**

**Ver1.9**
**Aug 26, 2020**

## NEWRACOM, Inc.

**NRC7292 Evaluation Kit User Guide (Host Mode)**
**Ultra-low power & Long-range Wi-Fi**



## Office


NEWRACOM, Inc.
25361 Commercentre Drive, Lake Forest, CA 92630 USA
http://www.NEWRACOM.com

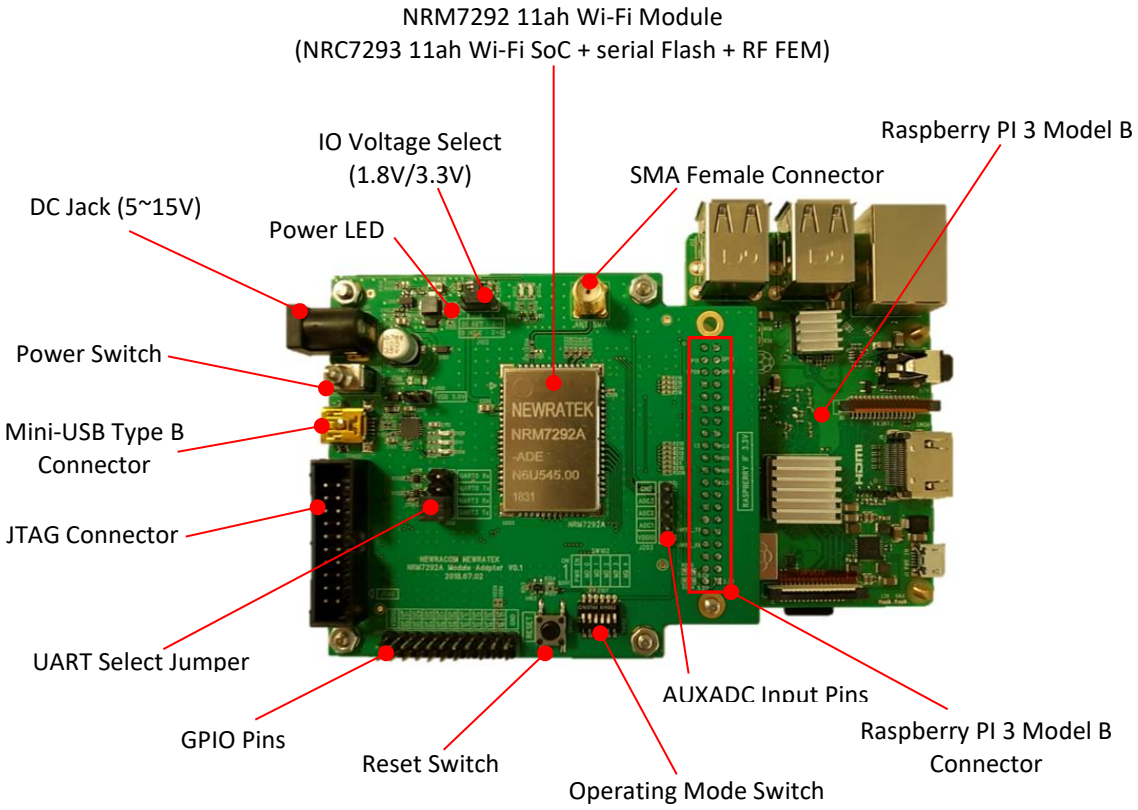
# Contents

# List of Tables

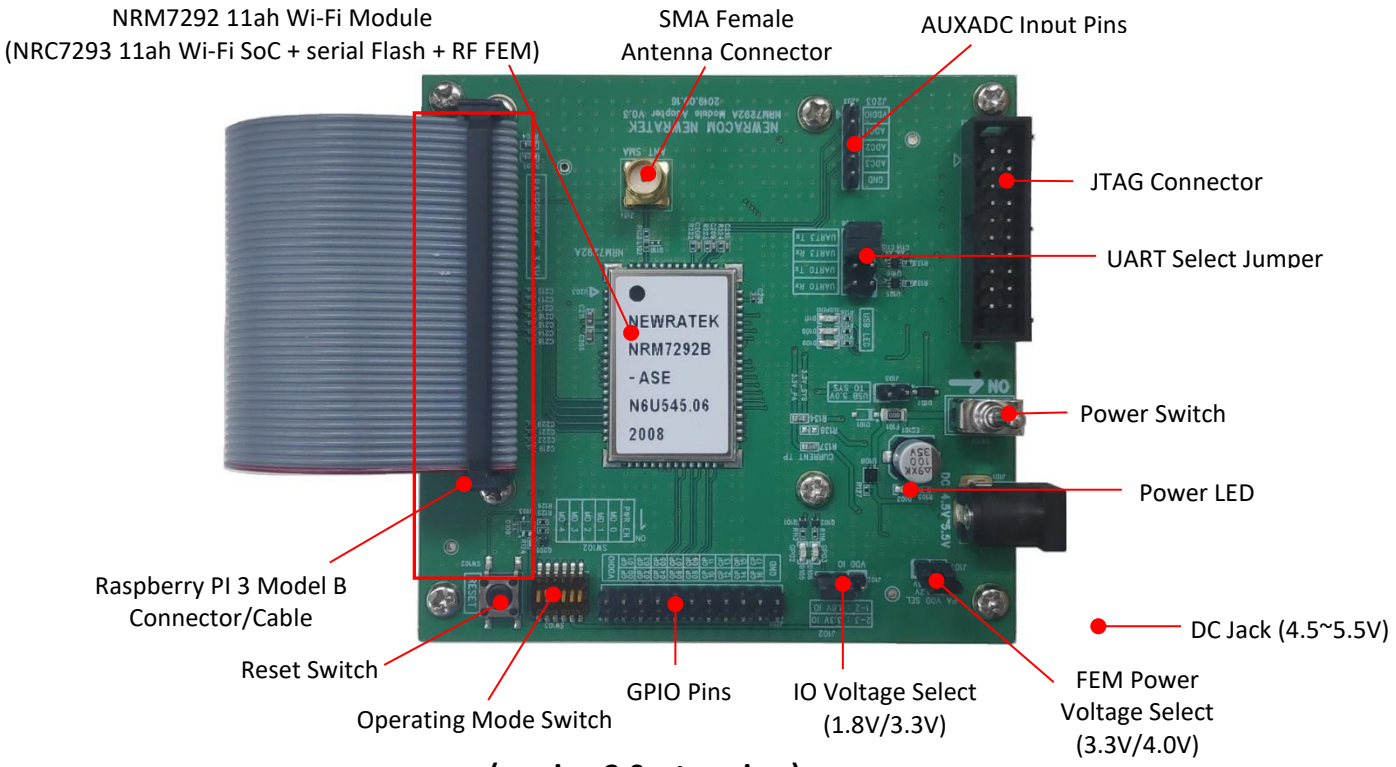
# List of Figures

# 1 Overview

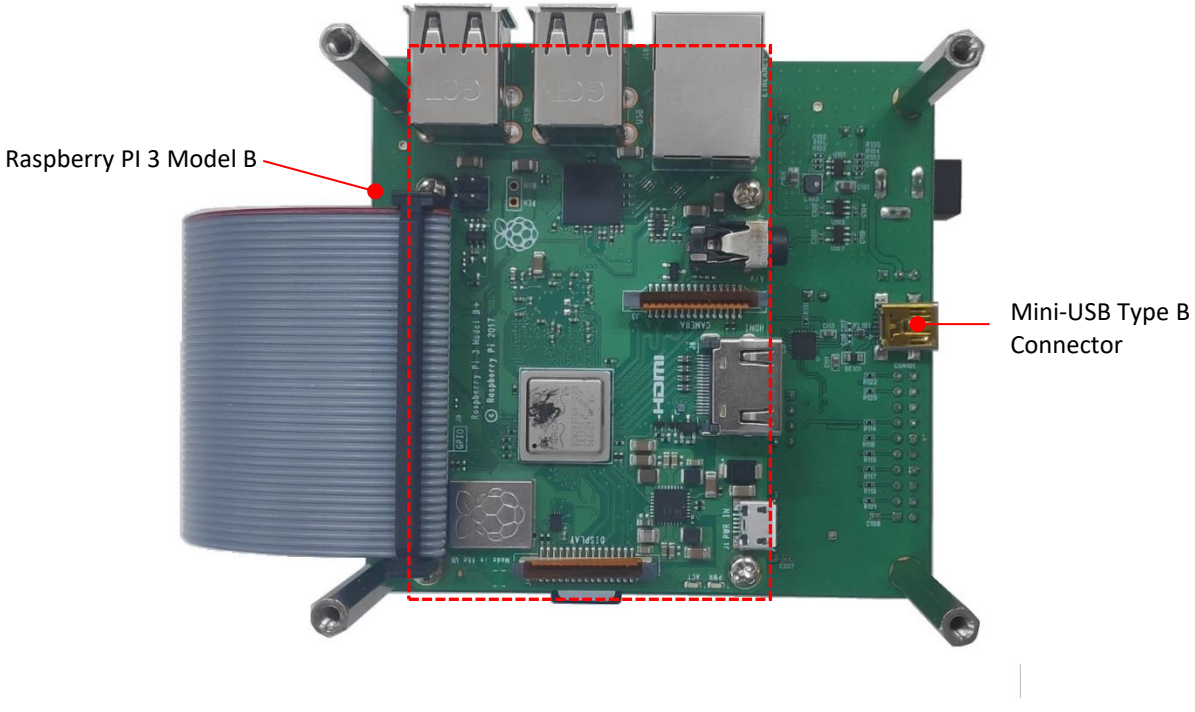
This document introduces NEWRACOM's NRC7292 Evaluation kit (EVK). The EVK kit is used to evaluate the performance of the NRC7292 Wi-Fi module containing NEWRACOM's IEEE 802.11ah Wi-Fi System on Chip (SoC) solution.




**(version 1.0)**




**(version 2.0 – top view)**




**(version 2.0 – bottom view)**

**Figure 1.1 NRC7292 evaluation board**


**Figure 1.2 Host mode configuration**

## 1.1 HW list

As shown in Figure 1.3, NRC7292 EVK consists of three boards.

### 1.1.1 NRC7292 module board

NRC7292 module contains IEEE 802.11ah Wi-Fi SoC solution. It also includes a RF front end module (FEM) to increase transmission power up to +23 dBm. Onboard serial flash memory can be used for over-the-air (OTA) software development and with 32KB cache in the NRC7292 supports the execution in place (XIP) feature.

### 1.1.2 NRC7292 adapter board

NRC7292 adapter board mainly offers communication interfaces to sensors or an external host. It also supplies the main power of NRC7292 Wi-Fi module.

### 1.1.3 Host board

NRC7292 module can be used either in standalone or slave to host processor via serial peripheral interface (SPI) or universal asynchronous receive transmitter (UART). Raspberry PI 3 can be one of the hosts used for normal operation, evaluation, and testing. When used in standalone, Raspberry PI3 board is not needed because NRC7292 operates without an additional host processor.

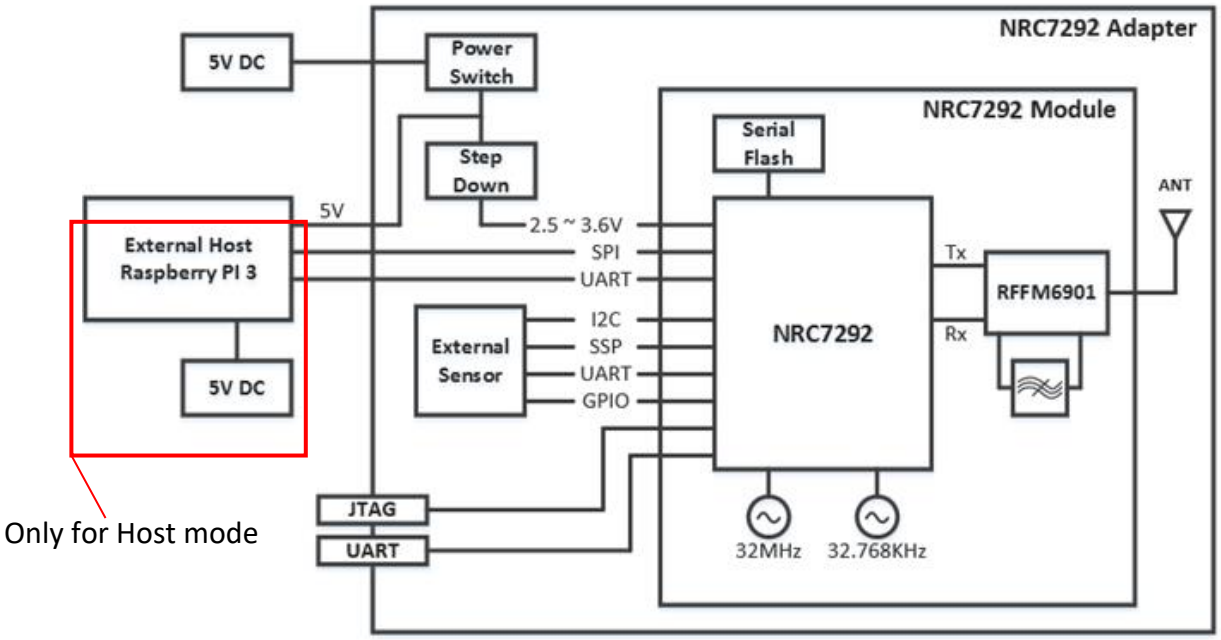
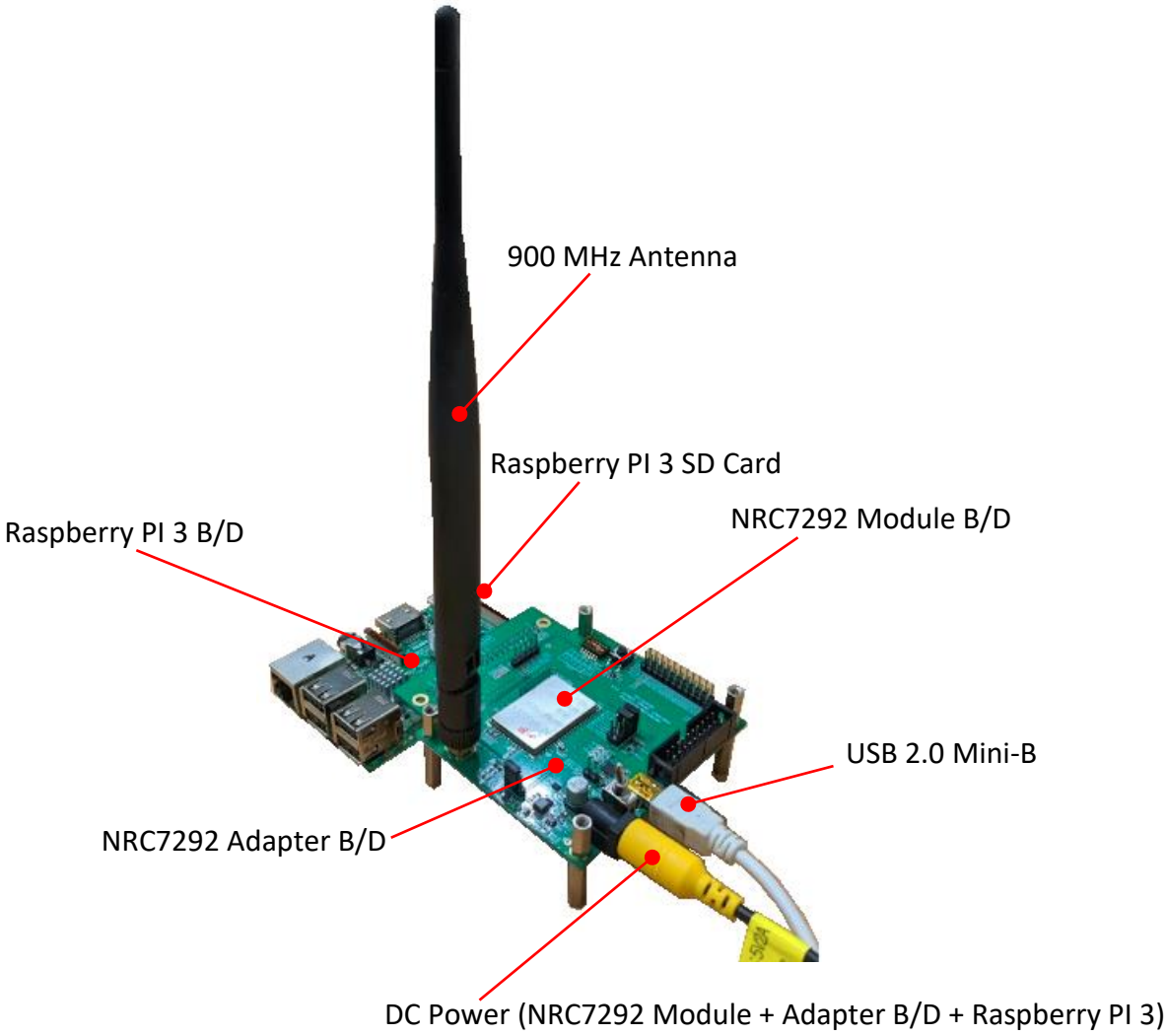



**Figure 1.3 Block diagram of NRC7292 evaluation board**

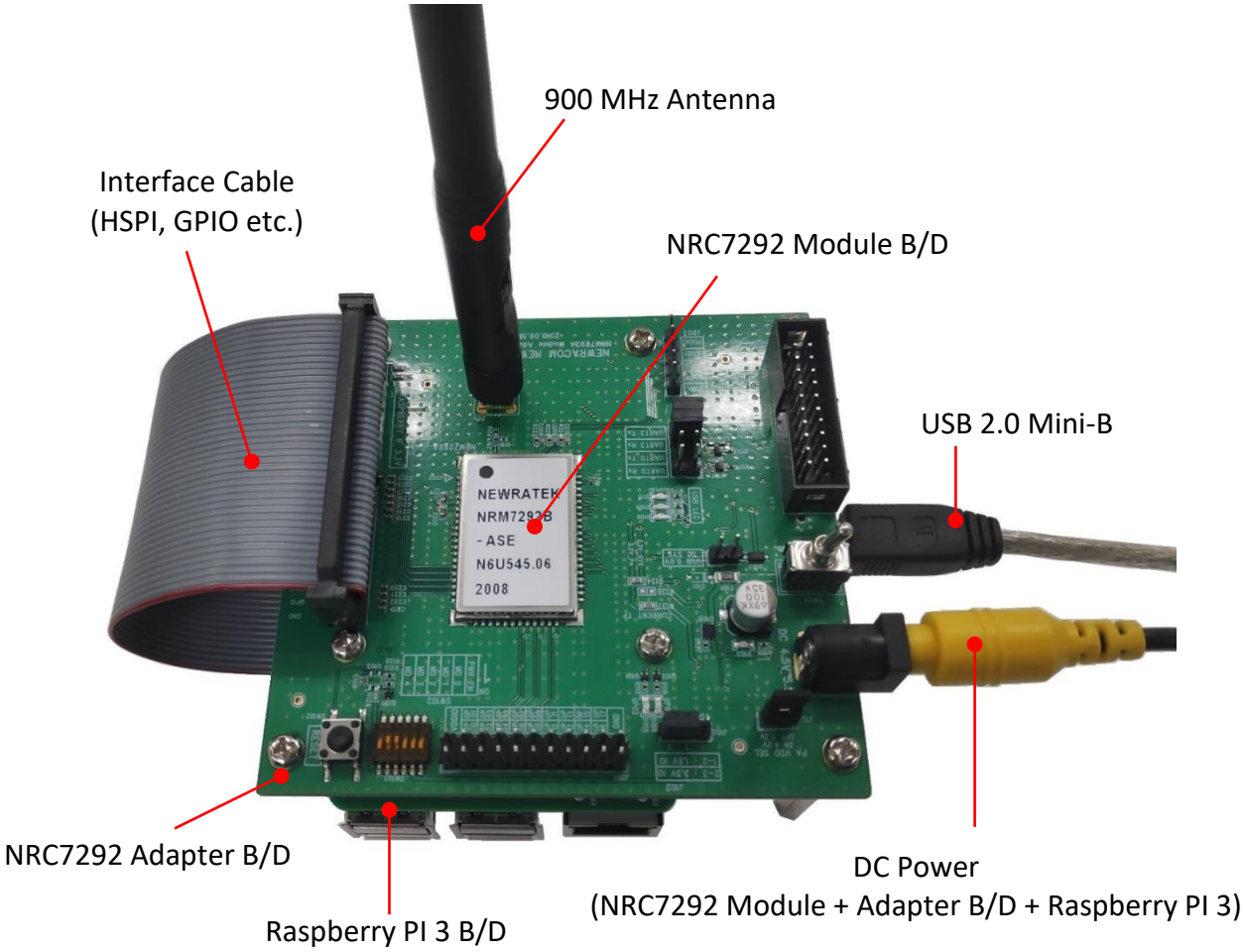
## 1.2 Kit list

NRC7292 EVK includes:

- NRC7292 Wi-Fi module board
- NRC7292 Adapter board
- Raspberry PI 3 board
- SD card with Linux OS, NRC7292 firmware, Wi-Fi driver, and scripts
- DC 5V (2A) power for EVK (Raspberry PI 3 + Wi-Fi module + Adapter board)
- 900 MHz Antenna




**(version 1.0)**




**(version 2.0)**

**Figure 1.4 NRC7292 hardware set**

# 2 NRC7292 EVK manipulation

This chapter describes how to manipulate NRC7292 EVK directly or remotely. To directly control NRC7292 EVK, I/O devices such as: keyboard, mouse, monitor, and HDMI cable are needed. Otherwise, users can use terminal programs such as: Tera Term, Mobaxterm, VNC, and etc. to control NRC7292 EVK remotely.

## 2.1 Direct manipulation

Using I/O devices is the simplest way to control NRC7292 EVK. It includes Raspberry PI3 which supports various I/O, especially useful is the HDMI connection. User can simply display Raspberry Pi (RPi) to a monitor by HDMI cable. Other USB-type keyboard and mouse can be utilized as input devices. However, if these I/O devices are not available, users have to additional options to manipulate NRC7292 EVK.

## 2.2 Remote manipulation

Raspberry PI3 has an Ethernet port, so users can remotely connect to RPi with terminal programs. There are two ways to obtain an IP address on RPi Ethernet, one way is static IP and the other is dynamic IP (DHCP). The initial (default) setting is set to obtain an IP address from the external DHCP server. Furthermore, the RPi can be operated as a DHCP server by changing the configuration. When enabled, the DHCP server will assign a static IP address (192.168.100.1) to RPi Ethernet and run the DHCP server. The PC will then obtain an IP address from RPi after connecting by Ethernet cable to the RPi. After the PC receives the IP address from RPi, then users can remotely access RPi (192.168.100.1) with terminal programs. Figure 2.1 and Figure 2.2 show Mobaxterm and VNC displays of PC receiving the IP address (192.168.100.10) after connecting to RPi.

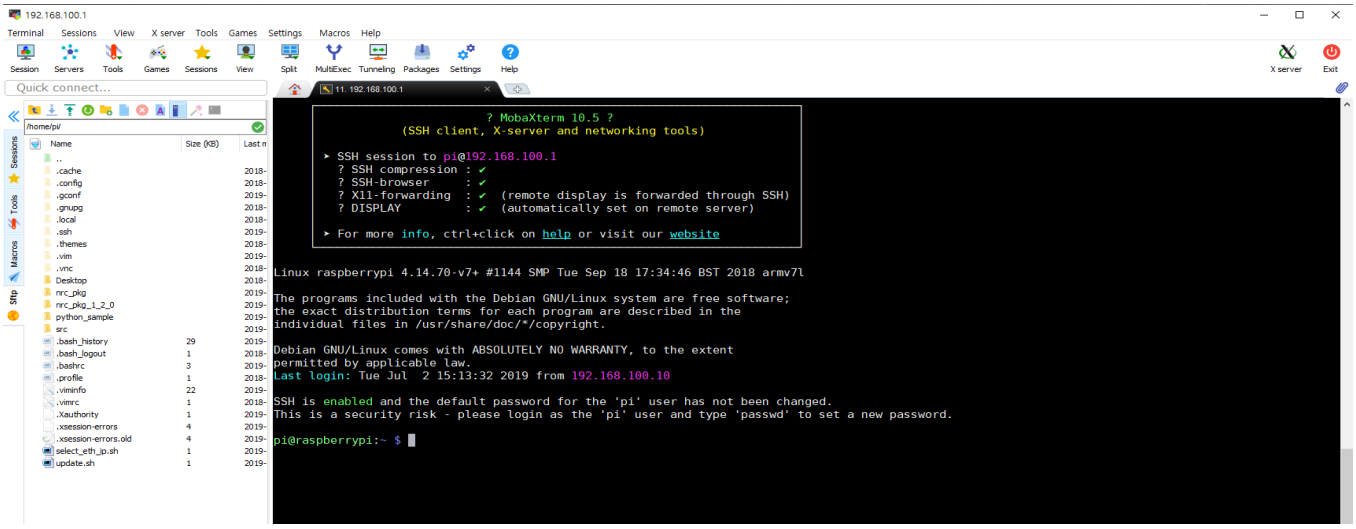



**Figure 2.1 Mobaxterm display**

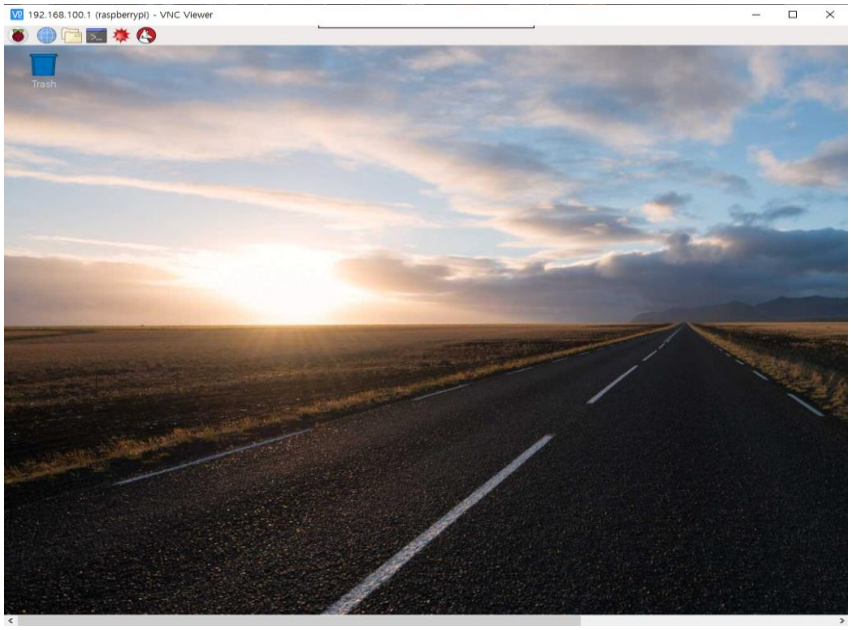



**Figure 2.2 VNC viewer display**

## 2.3 IP setting for Ethernet

By default, the RPi's Ethernet port is assigned an IP address through a DHCP server. To set up as an intra network, the Ethernet port can be used as a DHCP client by connecting the Ethernet cable from a switch, hub, or router to the RPi. The DHCP server on an intra network is assigned an Ethernet IP address and remotely accesses the RPi with the same IP address.

Script file (~/nrc_pkg/script/conf/etc/CONFIG_IP)

```
# Config for Ethernet with DHCP server
USE_ETH_DHCP_SERVER=N   # Use DHCP Sever : Y(use DHCP Server) or N(use DHCP Client)
ETH_DHCPS_IP=192.168.100.1    # only valid when using DHCP Server
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.10,255.255.255.0,24h # only valid when usi

# Config for static IP on Ethernet without DHCP server
USE_ETH_STATIC_IP=N           # Use ETH STATIC IP : Y(use ETH_IP for static ip) or N(
ETH_STATIC_IP=192.168.100.11   # only valid when using static IP without DHCP Server
ETH_STATIC_NETMASK=24          # only valid when using static IP without DHCP Server
```

**Figure 2.3 Configuration of IP setting for Ethernet DHCP client**

If the RPi Ethernet port works as a DHCP server, change it to “USE_ETH_DHCP_SERVER=Y” and run the script and reboot the RPi. After the reboot, the Ethernet IP is set to 192.168.100.1. One device connected with an Ethernet cable will be assigned an IP of 192.168.100.10. If more than one DHCP client devices are required, change the 'End IP address' parameter of 'ETH_DHCPS_CONFIG' from '192.168.100.10' to the required range such as ”ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.20,255.255.255.0,24h”

```
# Config for Ethernet with DHCP server
USE_ETH_DHCP_SERVER=Y   # Use DHCP Sever : Y(use DHCP Server) or N(use DHCP Client)
ETH_DHCPS_IP=192.168.100.1    # only valid when using DHCP Server
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.10,255.255.255.0,24h # only valid when us
                        Start IP address      End IP address
# Config for static IP on Ethernet without DHCP server
USE_ETH_STATIC_IP=N           # Use ETH STATIC IP : Y(use ETH_IP for static ip) or N
ETH_STATIC_IP=192.168.100.11  # only valid when using static IP without DHCP Server
ETH_STATIC_NETMASK=24         # only valid when using static IP without DHCP Server
```

**Figure 2.4 Configuration of IP setting for Ethernet DHCP server**

If the RPi's Ethernet Port is to work with Static IP instead of a DHCP Server, change it to “USE_ETH_DHCP_SERVER=N” and “USE_ETH_STATIC_IP=Y” and run the script and reboot the RPi. After reboot, the RPi's Ethernet IP is set to 192.168.100.11 and the device connected to the Ethernet cable is set to Static IP.

```
# Config for Ethernet with DHCP server
USE_ETH_DHCP_SERVER=N   # Use DHCP Sever : Y(use DHCP Server) or N(use DHCP Client)
ETH_DHCPS_IP=192.168.100.1    # only valid when using DHCP Server
ETH_DHCPS_CONFIG=192.168.100.10,192.168.100.10,255.255.255.0,24h # only valid when us

# Config for static IP on Ethernet without DHCP server
USE_ETH_STATIC_IP=Y           # Use ETH STATIC IP : Y(use ETH_IP for static ip) or N
ETH_STATIC_IP=192.168.100.11  # only valid when using static IP without DHCP Server
ETH_STATIC_NETMASK=24         # only valid when using static IP without DHCP Server
```

**Figure 2.5 Configuration of IP setting for Ethernet Static IP**

# 3 NRC7292 EVK AP/STA operation

This chapter explains how to start the IEEE 802.11ah AP operation and enable STA to connect to AP.

## 3.1 Start Script

The “start.py” in ~/nrc_pkg/script folder is the unified script used to initiate AP, STA, Sniffer, and STA with Ucode. For each operation, the different arguments are needed.

Three arguments which are *sta_type*, *security_mode*, *country* is necessary for an AP or STA operation.

However, for Sniffer operation, two additional arguments, *channel*, and *sniffer mode*, are needed.

```
pi@raspberrypi:~/nrc_pkg/script $ ./start.py
Usage:
        start.py [sta_type] [security_mode] [country] [channel] [sniffer_mode]
Argument:
        sta_type      [0:STA   |  1:AP  |  2:SNIFFER  | 3:RELAY | 4:STA+Ucode]
        security_mode [0:Open  |  1:WPA2-PSK  |  2:WPA3-OWE  |  3:WPA3-SAE]
        country       [US:USA  |  JP:Japan  |  TW:Taiwan  | KR:Korea | EU:EURO | CN:China]
        ------------------------------------------------------------
        channel       [S1G Channel Number]   * Only for Sniffer
        sniffer_mode  [0:Local | 1:Remote]   * Only for Sniffer
Example:
        OPEN mode STA for Korea                : ./start.py 0 0 KR
        Security mode AP for US                : ./start.py 1 1 US
        Local Sniffer mode on CH 40 for Japan  : ./start.py 2 0 JP 40 0
Note:
        sniffer_mode should be set as '1' when running sniffer on remote terminal
```

**Figure 3.1 Usage of start.py script**

***NOTE***

***Executing “start.py” script overwrites some lane in the dhcpcd.conf (~/nrc_pkg/etc/dhcpcd/) and the dnsmasq.conf (~/nrc_pkg/etc/dnsmasq) files as shown below. To keep the user’s configuration, users should leave that lane blanked.***

| **dhcpcd.conf file** | **dnsmasq.conf** |
|---|---|
| Line 59, 62 //IP address for wlan0<br>Line 65, 68 //IP address for wlan1<br>Line 71, 72 //IP address for eth0 | Line 2 //DHCPS configuration for eth0<br>Line 3, 4 //DHCPS configuration for wlan0 or wlan1 |

## 3.2 AP mode operation

**1) Open terminal with SSH**

After boot-up of AP board, connect via SSH to the AP by using the terminal emulator like MobaXterm. The ID and PW are as follows:

- ID : pi
- PW : raspberry

**2) Run script**

To run AP, a user should give “1” as *sta_type* and select one of the security mode and country code.

**parameter setting**

- *sta_type* : **1 (AP)**
- *security_mode* : available mode are 0(open), 1(WPA2-PSK), 2(WPA3-OWE) and 3(WPA3-SAE)
- *country* : available country codes are US, JP, TW, KR, EU, CN

  Ex) open mode AP to be used in Korea : ./start.py 1 0 KR

**3) Check results**

Once AP runs by start script, there are several procedures that are executed: 1) clear apps 2) copy firmware 3) load module 4) set configurations 5) start the hostapd 6) set NAT 7) start DNSMASQ, and performed in sequential order. If all procedures are successful, the user can find the wlan0 interface with the IP address created as shown in Figure 3.3.

```
pi@raspberrypi:~/nrc_pkg/script $ ./start.py 1 0 KR
------------------------------
Model            : 7292
STA Type         : AP
Security Mode    : OPEN
Country Selected : KR
Download FW      : uni_s1g.bin
TX Power         : 17
------------------------------
NRC AP setting for HaLow...
[*] Set Max CPU Clock on RPi
1200000
1200000
1200000
1200000
Done
[0] Clear
hostapd: no process found
wireshark-gtk: no process found
rmmod: ERROR: Module nrc is not currently loaded
[1] Copy
total 460
drwxr-xr-x 2 pi pi   4096 Jul  1 13:40 .
drwxr-xr-x 4 pi pi   4096 Jul  1 13:40 ..
-rwxr-xr-x 1 pi pi    218 Jul  1 13:40 copy
-rwxr-xr-x 1 pi pi 228664 Jul  1 13:40 nrc7292_cspi.bin
-rwxr-xr-x 1 pi pi 228664 Jul  2 16:01 uni_s1g.bin
-rw-r--r-- 1 root root 228664 Jul  2 16:01 /lib/firmware/uni_s1g.bin
Config for AP is done!
done
[2] Loading module
sudo insmod ~/nrc_pkg/sw/driver/nrc.ko fw_name=uni_s1g.bin disable_cqm=0 hifspeed=16000000
[3] Set tx power
nrf txpwr 17
success
[4] Set aggregation number
Input TID is (0)
set maxagg 1 8
success
[5] Set guard interval
set gi long
success
[6] Start hostapd
Configuration file: /home/pi/nrc_pkg/script/conf/KR/ap_halow_open.conf
wlan0: interface state UNINITIALIZED->COUNTRY_UPDATE
Using interface wlan0 with hwaddr 00:10:40:39:78:62 and ssid "halow_demo"
wlan0: interface state COUNTRY_UPDATE->ENABLED
wlan0: AP-ENABLED
[7] Start NAT
[8] Start DNSMASQ
```

**Figure 3.2 Results of running AP (1/2)**

```
[9] ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.5  netmask 255.255.255.0  broadcast 192.168.3.255
        inet6 fe80::b331:2b4a:6f0e:3bb2  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:0e:07:04  txqueuelen 1000  (Ethernet)
        RX packets 268  bytes 21862 (21.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 249  bytes 38445 (37.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 38  bytes 5292 (5.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 5292 (5.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.200.1  netmask 255.255.255.0  broadcast 192.168.200.255
        inet6 fe80::6154:4ff9:1632:e2a0  prefixlen 64  scopeid 0x20<link>
        ether 00:10:40:39:78:62  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26  bytes 4035 (3.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

HaLow AP ready
Done.
```

**Figure 3.3 Results of running AP (2/2)**

## 3.3 STA mode operation

**1) Open terminal with SSH**

After boot-up of STA board, connect via SSH to the STA by using the terminal emulator like MobaXterm. The ID and PW are as follows:

- ID : pi
- PW : raspberry

**2) Run script**

Running STA is similar to the procedure as AP's except for giving "0" as *sta_type*.

**parameter setting**

- *sta_type* : **0 (STA)**
- *security_mode* : available modes are 0(open), 1(WPA2-PSK), 2(WPA3-OWE) and 3(WPA3-SAE)
- *country* : available country codes are US, JP, TW, KR, EU, CN

Ex) security mode STA to be used in Japan : ./start.py 0 1 JP

**3) Check results**

Once STA runs by start script, there are several procedures that is performed, 1) clear apps 2) copy firmware 3) load module 4) Set configurations 5) start wpa supplicant 6) start scan AP 7) connect to AP 8) start DHCP client are performed sequentially. If all the procedures are successful, STA finally gets an IP address.

```
pi@raspberrypi:~/nrc_pkg/script $ ./start.py 0 0 KR
------------------------------
Model            : 7292
STA Type         : STA
Security Mode    : OPEN
Country Selected : KR
Download FW      : uni_s1g.bin
TX Power         : 17
------------------------------
NRC STA setting for HaLow...
[*] Set Max CPU Clock on RPi
1200000
1200000
1200000
1200000
Done
[0] Clear
hostapd: no process found
wireshark-gtk: no process found
[1] Copy
total 460
drwxr-xr-x 2 pi pi   4096 Jul  1 13:41 .
drwxr-xr-x 4 pi pi   4096 Jul  1 13:41 ..
-rwxr-xr-x 1 pi pi    218 Jul  1 13:41 copy
-rwxr-xr-x 1 pi pi 228664 Jul  1 13:41 nrc7292_cspi.bin
-rwxr-xr-x 1 pi pi 228664 Jul  2 16:07 uni_s1g.bin
-rw-r--r-- 1 root root 228664 Jul  2 16:07 /lib/firmware/uni_s1g.bin
Config for STA is done!
done
[2] Loading module
sudo insmod ~/nrc_pkg/sw/driver/nrc.ko fw_name=uni_s1g.bin disable_cqm=0 hifspeed=16000000
[3] Set tx power
nrf txpwr 17
success
[4] Set aggregation number
Input TID is (0)
set maxagg 1 8
success
[5] Set guard interval
set gi long
success
[6] Start wpa_supplicant
Successfully initialized wpa_supplicant
wlan0: SME: Trying to authenticate with 00:10:40:39:78:62 (SSID='halow_demo' freq=5220 MHz)
wlan0: Trying to associate with 00:10:40:39:78:62 (SSID='halow_demo' freq=5220 MHz)
wlan0: Associated with 00:10:40:39:78:62
wlan0: CTRL-EVENT-CONNECTED - Connection to 00:10:40:39:78:62 completed [id=0 id_str=]
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
[7] Connect and DHCP
Waiting for IP
ip_address=192.168.200.31
IP assigned. HaLow STA ready
Done.
```

**Figure 3.4 Results of running STA**

## 3.4 Configure static IP address

The DHCP client on a STA can obtain an IP address after successful connection to the AP. Or the users can assign a static IP address to the STA by using the script file provided with software package. Figure 3.5 presents the script (~/nrc_pkg/script/conf/etc/CONFIG_IP) and indicates the fields to set.

```
# Config for HaLow STA's IP and Defautl GW
USE_HALOW_STA_STATIC_IP=N      # Use STATIC IP :  Y(use Static IP) or N(use Dynamic IP(DHCP))
HALOW_STA_IP=192.168.200.11    # only valid when using static IP
HALOW_STA_NETMASK=24           # only valid when using static IP
HALOW_STA_DEFAULT_GW=192.168.200.1 # only valid when using static IP
```

**Figure 3.5 Configure static IP address for STA**

In addition, users can configure static IP addresses in the same way as the STA configuration. Figure 3.6 shows the fields to configure.

```
# Config for HaLow AP's IP and DHCPS configuration
HALOW_AP_IP=192.168.200.1
HALOW_AP_NETMASK=24
HALOW_AP_DHCPS_CONFIG=192.168.200.10,192.168.200.50,255.255.255.0,24h
```

**Figure 3.6 Configure static IP address for AP**

# 4 NRC7292 EVK performance evaluation

## 4.1 Performance test

Users can evaluate throughput performance by using the iperf3 tool.

**1) Run iperf3 server at AP side**

```
pi@raspberrypi:~/nrc_pkg/script $ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.200.37, port 48108
[  5] local 192.168.200.1 port 5201 connected to 192.168.200.37 port 33605
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec   320 KBytes  2.62 Mbits/sec  6.193 ms  0/40 (0%)
[  5]   1.00-2.00   sec   352 KBytes  2.88 Mbits/sec  9.022 ms  0/44 (0%)
[  5]   2.00-3.00   sec   352 KBytes  2.88 Mbits/sec  10.556 ms  0/44 (0%)
[  5]   3.00-4.00   sec   368 KBytes  3.01 Mbits/sec  7.511 ms  0/46 (0%)
[  5]   4.00-5.00   sec   376 KBytes  3.08 Mbits/sec  7.605 ms  0/47 (0%)
[  5]   5.00-6.00   sec   368 KBytes  3.01 Mbits/sec  8.038 ms  0/46 (0%)
[  5]   6.00-7.00   sec   368 KBytes  3.01 Mbits/sec  7.942 ms  0/46 (0%)
[  5]   7.00-8.00   sec   400 KBytes  3.28 Mbits/sec  6.621 ms  0/50 (0%)
[  5]   8.00-9.00   sec   384 KBytes  3.15 Mbits/sec  8.250 ms  0/48 (0%)
[  5]   9.00-10.00  sec   360 KBytes  2.95 Mbits/sec  9.083 ms  0/45 (0%)
[  5]  10.00-10.24  sec  96.0 KBytes  3.22 Mbits/sec  9.234 ms  0/12 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-10.24  sec  0.00 Bytes  0.00 bits/sec  9.234 ms  0/468 (0%)
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

**Figure 4.1 Run iperf3 server**

**2) Run iperf3 client at STA side (users can use other iperf3 options as well)**

```
pi@raspberrypi:~/nrc_pkg/script $ iperf3 -c 192.168.200.1 -u -b 10m -t 10
Connecting to host 192.168.200.1, port 5201
[  4] local 192.168.200.37 port 33605 connected to 192.168.200.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   416 KBytes  3.41 Mbits/sec  52
[  4]   1.00-2.00   sec   360 KBytes  2.95 Mbits/sec  45
[  4]   2.00-3.00   sec   344 KBytes  2.82 Mbits/sec  43
[  4]   3.00-4.00   sec   368 KBytes  3.01 Mbits/sec  46
[  4]   4.00-5.00   sec   376 KBytes  3.08 Mbits/sec  47
[  4]   5.00-6.00   sec   368 KBytes  3.01 Mbits/sec  46
[  4]   6.00-7.00   sec   368 KBytes  3.01 Mbits/sec  46
[  4]   7.00-8.00   sec   392 KBytes  3.21 Mbits/sec  49
[  4]   8.00-9.00   sec   392 KBytes  3.21 Mbits/sec  49
[  4]   9.00-10.00  sec   360 KBytes  2.95 Mbits/sec  45
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  3.66 MBytes  3.07 Mbits/sec  9.234 ms  0/468 (0%)
[  4] Sent 468 datagrams

iperf Done.
```

**Figure 4.2 Run iperf3 client**

## 4.2 Enable/Disable A-MPDU

Aggregate MAC protocol data unit (A-MPDU) is a MAC frame with multiple MAC sub-frames and single PHY header. By using A-MPDU, users can improve throughput because overhead is decreased in PHY header and inter-frame space (IFS).

Users can enable/disable the A-MPDU by changing the value of the *maxagg_num* in the 'start.py' script as shown in Figure 4.3. Setting "0" or "1" to *maxagg_num* disables the A-MPDU. The value "2" enables the A-MPDU packed with two sub-frames. Up to eight sub-frames can be aggregated in an A-MPDU.

```
##############################################################################
# Default Configuration (you can change value you want here)
model       = 7292      # 7292 or 7192
hif_speed   = 16000000  # HSPI Clock
gain_type   = 'phy'     # 'phy' or 'nrf(legacy)'
txpwr_val   = 17        # TX Power
maxagg_num  = 8         # 0(AMPDU off) or  >2(AMPDU on)
cqm_off     = 0         # 0(CQM on) or 1(CQM off)
fw_download = 1         # 0(FW Download off) or 1(FW Download on)
fw_name     = 'uni_s1g.bin'
bd_download = 0         # 0(Board Data Download off) or 1(Board Data Download on)
bd_name     = 'nrc7292_bd.dat'
guard_int   = 'long'    # 'long'(LGI) or 'short'(SGI)
supplicant_debug = 0    # WPA Supplicant debug option : 0(off) or 1(on)
hostapd_debug    = 0    # Hostapd debug option    : 0(off) or 1(on)
max_cpuclock     = 1    # RPi Max CPU Clock : 0(off) or 1(on)
relay_type       = 0    # 0 (wlan0: STA, wlan1: AP) 1 (wlan0: AP, wlan1: STA)
power_save       = 0    # power save : 0(off) or 1(on)
bss_max_idle_enable = 0 # 0(bss_max_idle off) or 1(bss_max_idle on)
bss_max_idle = 10       # number of keepalives (0 ~ 65535)
##############################################################################
```

**Figure 4.3 Enable/disable A-MPDU**

## 4.3 Enable/Disable power save

A user can enable/disable the power save option by changing the value of the *power_save* in the 'start.py' script as shown in Figure 4.3. Setting "0" disables the power save and value "1" enables the power_save.

```
##################################################################################
# Default Configuration (you can change value you want here)
model       = 7292      # 7292 or 7192
hif_speed   = 16000000  # HSPI Clock
gain_type   = 'phy'     # 'phy' or 'nrf(legacy)'
txpwr_val   = 17        # TX Power
maxagg_num  = 8         # 0(AMPDU off) or  >2(AMPDU on)
cqm_off     = 0         # 0(CQM on) or 1(CQM off)
fw_download = 1         # 0(FW Download off) or 1(FW Download on)
fw_name     = 'uni_s1g.bin'
bd_download = 0         # 0(Board Data Download off) or 1(Board Data Download on)
bd_name     = 'nrc7292_bd.dat'
guard_int   = 'long'    # 'long'(LGI) or 'short'(SGI)
supplicant_debug = 0    # WPA Supplicant debug option : 0(off) or 1(on)
hostapd_debug    = 0    # Hostapd debug option    : 0(off) or 1(on)
max_cpuclock     = 1    # RPi Max CPU Clock : 0(off) or 1(on)
relay_type       = 0    # 0 (wlan0: STA, wlan1: AP) 1 (wlan0: AP, wlan1: STA)
power_save       = 0    # power save : 0(off) or 1(on)
bss_max_idle_enable = 0 # 0(bss_max_idle off) or 1(bss_max_idle on)
bss_max_idle = 10       # number of keepalives (0 ~ 65535)
##################################################################################
```

**Figure 4.4 Enable/disable power save**

## 4.4 Enable/Disable board data

A user can enable/disable the option to download board data by changing the bd_download value in the 'start.py' script.

Setting BD Data is an enhanced Tx Power scheme, and Tx Power is set differently according to the rate table of board data. “0” disables downloading board data with the firmware and value “1” enables downloading board data with the firmware. Currently only works in Country US, default value is 0.

```
##################################################################################
# Default Configuration (you can change value you want here)
model       = 7292      # 7292 or 7192
hif_speed   = 16000000  # HSPI Clock
gain_type   = 'phy'     # 'phy' or 'nrf(legacy)'
txpwr_val   = 17        # TX Power
maxagg_num  = 8         # 0(AMPDU off) or  >2(AMPDU on)
cqm_off     = 0         # 0(CQM on) or 1(CQM off)
fw_download = 1         # 0(FW Download off) or 1(FW Download on)
fw_name     = 'uni_s1g.bin'
bd_download = 0         # 0(Board Data Download off) or 1(Board Data Download on)
bd_name     = 'nrc7292_bd.dat'
guard_int   = 'long'    # 'long'(LGI) or 'short'(SGI)
supplicant_debug = 0    # WPA Supplicant debug option : 0(off) or 1(on)
hostapd_debug    = 0    # Hostapd debug option    : 0(off) or 1(on)
max_cpuclock     = 1    # RPi Max CPU Clock : 0(off) or 1(on)
relay_type       = 0    # 0 (wlan0: STA, wlan1: AP) 1 (wlan0: AP, wlan1: STA)
power_save       = 0    # power save : 0(off) or 1(on)
bss_max_idle_enable = 0 # 0(bss_max_idle off) or 1(bss_max_idle on)
bss_max_idle = 10       # number of keepalives (0 ~ 65535)
##################################################################################
```

**Figure 4.5 Enable/disable board data**

## 4.5 Enable/Disable BSS MAX IDLE element

A user can enable/disable the option to download bss_max_idle with value by changing the bss_max_idle_enable in the 'start.py' script.

The bss_max_idle is used to check whether the STA is deactivated, and if the keep alive packet or any data is not received from the STA within the bss_max_idle value, the AP determines that the STA is no longer connected. If the AP is determined that the STA cannot be reached, the STA information is cleared through disassoc frame / deauthentication frame. The default is 10 seconds and can be set up to 65535. The default 'bss_max_idle_enable' is 0

```
################################################################################
# Default Configuration (you can change value you want here)
model       = 7292      # 7292 or 7192
hif_speed   = 16000000  # HSPI Clock
gain_type   = 'phy'     # 'phy' or 'nrf(legacy)'
txpwr_val   = 17        # TX Power
maxagg_num  = 8         # 0(AMPDU off) or  >2(AMPDU on)
cqm_off     = 0         # 0(CQM on) or 1(CQM off)
fw_download = 1         # 0(FW Download off) or 1(FW Download on)
fw_name     = 'uni_s1g.bin'
bd_download = 0         # 0(Board Data Download off) or 1(Board Data Download on)
bd_name     = 'nrc7292_bd.dat'
guard_int   = 'long'    # 'long'(LGI) or 'short'(SGI)
supplicant_debug = 0    # WPA Supplicant debug option : 0(off) or 1(on)
hostapd_debug    = 0    # Hostapd debug option    : 0(off) or 1(on)
max_cpuclock     = 1    # RPi Max CPU Clock : 0(off) or 1(on)
relay_type       = 0    # 0 (wlan0: STA, wlan1: AP) 1 (wlan0: AP, wlan1: STA)
power_save       = 0    # power save : 0(off) or 1(on)
bss_max_idle_enable = 0 # 0(bss max idle off) or 1(bss max idle on)
bss_max_idle = 10       # number of keepalives (0 ~ 65535)
################################################################################
```

**Figure 4.6 Enable/disable BSS MAX IDLE**

# 5 Internet connection

If NAT configuration is complete with the start script in chapter 3.2 and AP is connected to the Internet through its Ethernet interface, then the STA connected to the AP can access the Internet via AP. After boot-up, AP can obtain IP address from DHCP server.

On successful Wi-Fi connection, users can verify the reachability to the internet by using a ping program.

```
pi@raspberrypi:~/nrc_pkg/script $ ping google.com -c 3
PING google.com (172.217.24.206) 56(84) bytes of data.
64 bytes from hkg12s13-in-f14.1e100.net (172.217.24.206): icmp_seq=1 ttl=50 time=37.9 ms
64 bytes from hkg12s13-in-f14.1e100.net (172.217.24.206): icmp_seq=2 ttl=50 time=37.3 ms
64 bytes from hkg12s13-in-f14.1e100.net (172.217.24.206): icmp_seq=3 ttl=50 time=38.7 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 37.380/38.016/38.716/0.547 ms
```

**Figure 5.1 Result of ping**

Once the internet connection is confirmed by STA, users can surf the web and play YouTube.

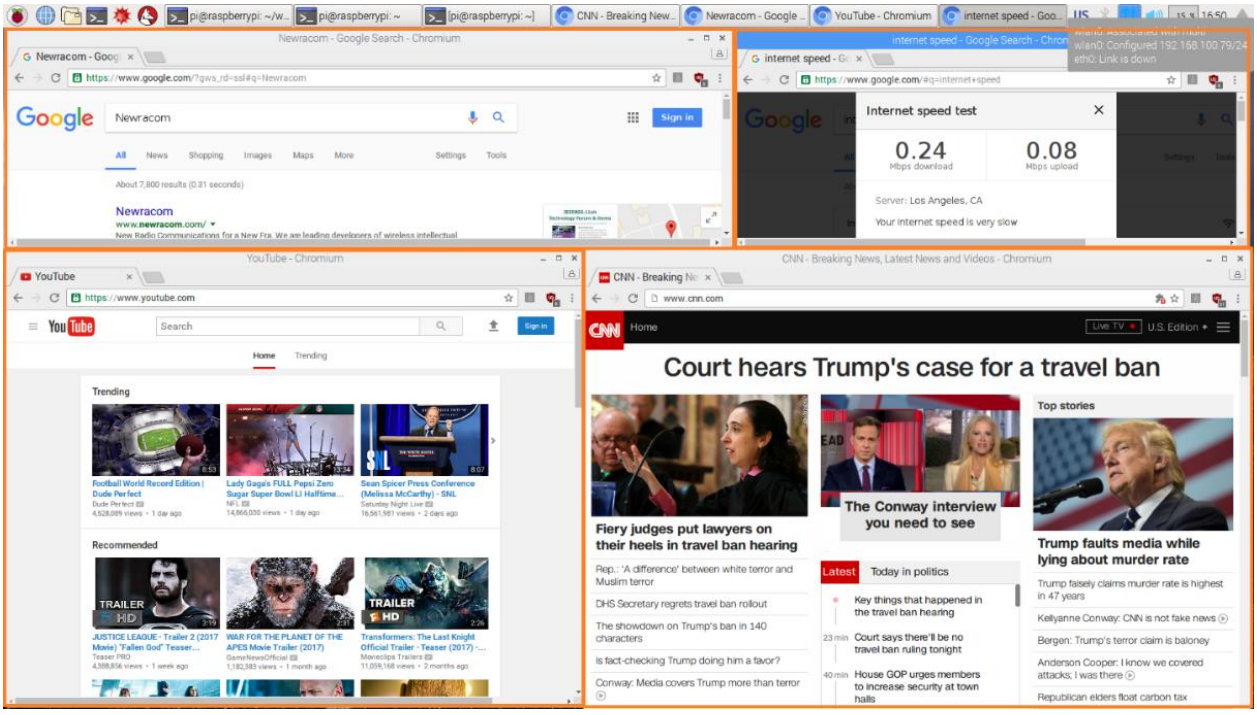



**Figure 5.2 Internal connection**

# 6 Change configuration

NRC7292 supports 1/2/4 MHz channel bandwidth (only 1 MHz for JP and 1/2 MHz for EU). The default channel bandwidth and frequency are different for every country (e.g. 2 MHz BW at 909 MHz for US, 1MHz BW at 917 MHz for JP, 2 MHz BW at 843.5 MHz for TW. Users can change the frequency band and channel bandwidth by editing only the frequency band index of the configuration file of AP named COUNTRY/*ap_halow_open.conf* (Users must attempt to change the channel on the AP). Table 6.1 to Table 6.6 represent the frequency band index, corresponding frequency band and channel bandwidth.

```
ctrl_interface=/var/run/hostapd
country_code=US
interface=wlan0
#ssid=halow_demo
ssid=halow
hw_mode=a

#1MHz
#909.5MHz
#channel=37
#915.5MHz
#channel=43
#922.5MHz
#channel=150

#2MHz
#909MHz
#channel=153
#915MHz
#channel=156
#921MHz
#channel=159
#925MHz
channel=161

#4MHz
#910MHz
#channel=162
#914MHz
#channel=163
#922MHz
#channel=165

ieee80211h=1
ieee80211d=1
ieee80211n=1
#wmm_enabled=1
macaddr_acl=0
driver=nl80211
beacon_int=100
#disable_sgi=0
#ap_max_inactivity=3
ap_max_inactivity=16780
# USF=0, UI=2 : 3
# USF=1, UI=2 : 16780
# USF=0, UI=10: 11
# USF=1, UI=10: 16788
#listen_interval=2
```

```
ctrl_interface=/var/run/hostapd
country_code=JP
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#917MHz
channel=36
#918MHz
#channel=37
#919MHz
#channel=38
#920MHz
#channel=39
#921MHz
#channel=40
#922MHz
#channel=41
#923MHz
#channel=42
#924MHz
#channel=43
#925MHz
#channel=44
#926MHz
#channel=45
#927MHz
#channel=46

ieee80211h=1
ieee80211d=1
ieee80211n=1
#wmm_enabled=1
macaddr_acl=0
driver=nl80211
beacon_int=100
#disable_sgi=0
#ap_max_inactivity=3
ap_max_inactivity=16780
# USF=0, UI=2 : 3
# USF=1, UI=2 : 16780
# USF=0, UI=10: 11
# USF=1, UI=10: 16788
#listen_interval=2
```

```
ctrl_interface=/var/run/hostapd
country_code=TW
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#839MHz
#channel=36
#843MHz
#channel=40
#850MHz
#channel=47

#2MHz
#839.5MHz
#channel=149
#843.5MHz
channel=151
#845.5MHz
#channel=152
#849.5MHz
#channel=154

#4MHz
#840.5MHz
#channel=155
#844.5Hz
#channel=156
#848.5MHz
#channel=157

ieee80211h=1
ieee80211d=1
ieee80211n=1
#wmm_enabled=1
macaddr_acl=0
driver=nl80211
beacon_int=100
#disable_sgi=0
#ap_max_inactivity=3
ap_max_inactivity=16780
# USF=0, UI=2 : 3
# USF=1, UI=2 : 16780
# USF=0, UI=10: 11
# USF=1, UI=10: 16788
#listen_interval=2
```

**Figure 6.1 Contents of ap_halow_open.conf file (US/JP/TW)**

```
ctrl_interface=/var/run/hostapd
country_code=KR
interface=wlan0
ssid=halow_demo
hw_mode=a

#tx_queue_data2_aifs=3
#tx_queue_data2_cwmin=15
#tx_queue_data2_cwmax=63
#tx_queue_data2_burst=30

#1MHz
#918MHz
#channel=36
#919MHz
#channel=37
#920MHz
#channel=38
#921MHz
#channel=39
#922MHz
#channel=40
#923MHz
#channel=41
#################
#942.8MHz
#channel=46
#943.8MHz
#channel=47
#944.8MHz
#channel=48
#945.8MHz
#channel=149

#2MHz
#918.5MHz
#channel=42
#920.5MHz
#channel=43
#922.5MHz
channel=44
#################
#943.3MHz
#channel=150
#945.3MHz
#channel=151

#4MHz
#921.5MHz
#channel=45
#################
#944.3MHz
#channel=152
```

```
ctrl_interface=/var/run/hostapd
country_code=DE
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#863.5MHz
#channel=36
#864.5MHz
#channel=37
#865.5MHz
#channel=38
#866.5MHz
#channel=39
#867.5MHz
#channel=40

#2MHz
#864MHz
channel=41
#866MHz
#channel=42
```

```
ctrl_interface=/var/run/hostapd
country_code=CN
interface=wlan0
ssid=halow_demo
hw_mode=a

#1MHz
#755.5MHz
#channel=36
#759.5MHz
#channel=40
#763.5MHz
#channel=44
#767.5MHz
#channel=48
#770.5MHz
#channel=151
#779.5MHz
#channel=152
#780.5MHz
#channel=153
#781.5MHz
#channel=154
#782.5MHz
#channel=155
#783.5MHz
#channel=156
#784.5MHz
#channel=157
#785.5MHz
#channel=158
#786.5MHz
#channel=159

#2MHz
#780MHz
channel=160
#782MHz
#channel=161
#784MHz
#channel=162
#786MHz
#channel=163

#4MHz
#781MHz
#channel=164
#785MHz
#channel=165
```

**Figure 6.2 Contents of ap_halow_open.conf file (KR/EU/CN)**

**Table 6.1 Available frequency band and corresponding channel bandwidth (US)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| 37 | 1 | 909.5 |
| 43 | 1 | 915.5 |
| 150 | 1 | 922.5 |
| 153 | 2 | 909 |
| 156 | 2 | 915 |
| 159 | 2 | 921 |
| **161(Default)** | **2** | **925** |
| 162 | 4 | 910 |
| 163 | 4 | 914 |
| 165 | 4 | 922 |

**Table 6.2 Available frequency band and corresponding channel bandwidth (JP)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| **36 (Default)** | **1** | **917** |
| 37 | 1 | 918 |
| 38 | 1 | 919 |
| 39 | 1 | 920 |
| 40 | 1 | 921 |
| 41 | 1 | 922 |
| 42 | 1 | 923 |
| 43 | 1 | 924 |
| 44 | 1 | 925 |
| 45 | 1 | 926 |
| 46 | 1 | 927 |

**Table 6.3 Available frequency band and corresponding channel bandwidth (TW)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| 36 | 1 | 839 |
| 40 | 1 | 843 |
| 47 | 1 | 850 |
| 149 | 2 | 839.5 |
| **151 (Default)** | **2** | **843.5** |
| 152 | 2 | 845.5 |
| 154 | 2 | 849.5 |
| 155 | 4 | 840.5 |
| 156 | 4 | 844.5 |
| 157 | 4 | 848.5 |

**Table 6.4 Available frequency band and corresponding channel bandwidth (KR)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| 36 | 1 | 918 |
| 37 | 1 | 919 |
| 38 | 1 | 920 |
| 39 | 2 | 921 |
| 40 | 2 | 922 |
| 41 | 2 | 923 |
| 46 | 2 | 942.8 |
| 47 | 1 | 943.8 |
| 48 | 1 | 944.8 |
| 149 | 1 | 945.8 |
| 42 | 2 | 918.5 |
| 43 | 2 | 920.5 |
| **44(Default)** | **2** | **922.5** |
| 150 | 2 | 943.3 |
| 151 | 2 | 945.3 |
| 45 | 4 | 921.5 |
| 152 | 4 | 944.3 |

**Table 6.5 Available frequency band and corresponding channel bandwidth (KR-MIC)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| 36 | 1 | 925.5 |
| 37 | 1 | 926.5 |
| 38 | 1 | 927.5 |
| 39 | 1 | 928.5 |
| 40 | 1 | 929.5 |
| 41 | 1 | 930.5 |
| 42 **(Default)** | 2 | 927.0 |
| 43 | 2 | 929.0 |

**Table 6.6 Available frequency band and corresponding channel bandwidth (EU)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| 36 | 1 | 863.5 |
| 37 | 1 | 864.5 |
| 38 | 1 | 865.5 |
| 39 | 1 | 866.5 |
| 40 | 1 | 867.5 |
| **41(Default)** | 2 | **864** |
| 42 | 2 | 866 |

**Table 6.7 Available frequency band and corresponding channel bandwidth (CN)**

| Available frequency band index | Bandwidth (MHz) | Sub 1 GHz frequency (MHz) |
|---|---|---|
| 36 | 1 | 755.5 |
| 40 | 1 | 759.5 |
| 44 | 1 | 763.5 |
| 48 | 1 | 767.5 |
| 151 | 1 | 770.5 |
| 152 | 1 | 779.5 |
| 153 | 1 | 780.5 |
| 154 | 1 | 781.5 |
| 155 | 1 | 782.5 |
| 156 | 1 | 783.5 |
| 157 | 1 | 784.5 |
| 158 | 1 | 785.5 |
| 159 | 1 | 786.5 |
| **160(Default)** | 2 | **780** |
| 161 | 2 | 782 |
| 162 | 2 | 784 |
| 163 | 2 | 786 |
| 164 | 4 | 781 |
| 165 | 4 | 785 |

# 7 NRC7292 EVK software

Table 7.1 and Table 7.2 show AP and STA's file directory on the Raspberry PI3 host. NRC7292 EVK contains some scripts to start AP/STA operation, a configuration file for operation, Linux Wi-Fi module driver, and NRC7292 firmware.

**Table 7.1 Files in AP**

| File or Folder | Path | Description |
|---|---|---|
| ap_halow_open.conf | NRC_PKG/script/conf/COUNTRY | AP configuration file (open mode) |
| ap_halow_wpa2_conf | NRC_PKG/script/conf/COUNTRY | AP configuration file (WPA2-PSK) |
| ap_halow_owe_conf | NRC_PKG/script/conf/COUNTRY | AP configuration file (WPA3-OWE) |
| ap_halow_sae_conf | NRC_PKG/script/conf/COUNTRY | AP configuration file (WPA3-SAE) |
| start.py | NRC_PKG/script | start script |
| stop.py | NRC_PKG/script | stop script |
| CONFIG_IP | NRC_PKG/script/conf/etc | IP and DHCP configuration |
| ip_config.sh | NRC_PKG/script/conf/etc | IP and DHCP configuration script |
| clock_config.sh | NRC_PKG/script/conf/etc | RPi Max Clock configuration script |
| nrc.ko | NRC_PKG/sw/driver | NRC7292 Host Wi-Fi driver |
| uni_s1g.bin | NRC_PKG/sw/firmware | NRC7292 firmware |

※ NRC_PKG = /home/pi/nrc_pkg/

**Table 7.2 Files in STA**

| File or Folder | Path | Description |
|---|---|---|
| sta_halow_open.conf | NRC_PKG/script/conf/COUNTRY | STA configuration file (open mode) |
| sta_halow_wpa2_conf | NRC_PKG/script/conf/COUNTRY | STA configuration file (WPA2-PSK) |
| sta_halow_owe_conf | NRC_PKG/script/conf/COUNTRY | STA configuration file (WPA3-OWE) |
| sta_halow_sae_conf | NRC_PKG/script/conf/COUNTRY | STA configuration file (WPA3-SAE) |
| start.py | NRC_PKG/script | start script |
| stop.py | NRC_PKG/script | stop script |
| CONFIG_IP | NRC_PKG/script/conf/etc | IP and DHCP configuration |
| ip_config.sh | NRC_PKG/script/conf/etc | IP and DHCP configuration script |
| clock_config.sh | NRC_PKG/script/conf/etc | RPi Max Clock configuration script |
| nrc.ko | NRC_PKG/sw/driver | NRC7292 Host Wi-Fi driver |
| uni_s1g.bin | NRC_PKG/sw/firmware | NRC7292 firmware |

※ NRC_PKG = /home/pi/nrc_pkg/

# 8 NRC7292 EVK Sniffer operation

NRC7292 EVK can be used to capture 11ah frames in the air like other Wi-Fi sniffer devices in the market. For the installation of software package and operation for sniffer, please refer to “NRC7292 EVK User Guide (NewraPeek™)”.

# 9 Revision history

| Revision No | Date | Comments |
|---|---|---|
| Ver 1.0 | 11/01/2018 | Initial version for customer release created |
| Ver 1.1 | 03/25/2019 | Description updated according to the new script and sniffer mode operation added |
| Ver 1.2 | 04/12/2019 | CN Table added for China updated |
| Ver 1.3 | 07/02/2019 | Manipulation NRC7292 EVK updated<br>WPA3-OWE, WPA3-SAE added<br>Method to enable/disable A-MPDU added |
| Ver 1.4 | 07/12/2019 | NOTE for “start.py” execution added<br>Static IP address assignment for AP and STA added |
| Ver 1.5 | 11/14/2019 | Update IP configuration using CONFIG_IP & power save |
| Ver 1.6 | 12/04/2019 | |
| Ver1.7 | 06/29/2020 | Version 2.0 HW appearance updated |
| Ver1.8 | 08/05/2020 | Update Board data and KR MIC Channel |
| Ver1.9 | 08/26/2020 | Update BSS MAX IDLE element |

# Appendix A.
# Upgrade hostapd & wpa_supplicant for supporting WPA3

## A.1 Overview

WPA3 is the next generation of Wi-Fi security and provides state-of-the-art security protocols to the market. So, all WPA3 networks:

- Use the latest security methods
- Disallow outdated legacy protocols
- Require use of Protected Management Frame (PMF)

WPA3-Personal brings better protections by providing robust password-based authentication. This capability is enabled by Simultaneous Authentication of Equals (SAE), which replaces Pre-Shared Key (PSK) in WPA2-Personal.

WPA3-Enterprise has two modes. Basic mode is based on WPA2-Enterprise and PMF. An optional mode using 192-bit security protocols is also defined in WPA3-Enterprise, but this is not adequate for the IoT application. So, it is not supported in NRC7292 EVK.

However, NRC7292 EVK supports Wi-Fi Enhanced Open mode, which is based on Opportunistic Wireless Encryption (OWE) and replaces open mode.

In summary, NRC7292 EVK supports following WPA3 security modes.

- Wi-Fi Enhanced Open (OWE mode)
- WFA3-Personal (WPA3-SAE mode)

By the way, it is necessary recommendation to upgrade hostapd and wpa_supplicant to version 2.8 for the full support of WPA3 protocols. Please follow the steps listed below to upgrade version.

## A.2 Upgrade hostapd

### A.2.1 Download hostapd v2.8 and install required libraries

Please follow the procedure below.

```
$ wget https://w1.fi/releases/hostapd-2.8.tar.gz
$ tar zxf hostapd-2.8.tar.gz
$ sudo apt-get update
$ sudo apt-get install libnl-3-dev libnl-genl-3-dev libssl-dev
```

### A.2.2 Build and install hostapd v2.8

Please follow the procedure below.

```
$ cd hostapd-2.8/hostapd
$ cp defconfig .config
$ vi .config
Enable followings:
CONFIG_IEEE80211N=y
CONFIG_OWE=y
Insert following:
CONFIG_SAE=y
$ make
$ sudo make install
```

## A.3 Upgrade wpa_supplicant

### A.3.1 Download wpa_supplicant v2.8 and install required libraries

Please follow the procedure below.

```
$ wget https://w1.fi/releases/wpa_supplicant-2.8.tar.gz
$ tar zxf wpa_supplicant-2.8.tar.gz
$ sudo apt-get update
$ sudo apt-get install libnl-3-dev libnl-genl-3-dev libssl-dev
$ sudo apt-get install libdbus-1-dev libdbus-glib-1-dev
```

### A.3.2 Build and install wpa_supplicant v2.8

Please follow the procedure below.

```
$ cd wpa_supplicant-2.8/wpa_supplicant
$ cp defconfig .config
$ vi .config
Enable followings:
CONFIG_IEEE80211N=y
CONFIG_OWE=y
CONFIG_SAE=y
$ make
$ sudo make install
```