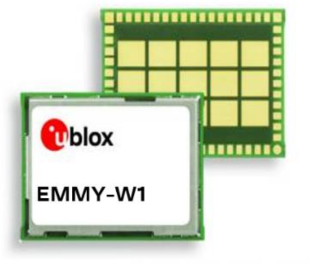
S

EMMY-W1 series

Android 9 integration

Application note



Abstract

This application note describes the procedure for integrating EMMY-W1 multiradio modules with Wi-Fi and Bluetooth in an Android framework.

www.u-blox.com
UBX-19035432 - R01

Document information

Title	**EMMY-W1 series**	
Subtitle	Android 9 integration	
Document type	Application note	
Document number	UBX-19035432	
Revision and date	R01	30-Oct-2019
Disclosure restriction		

This document applies to the following products:

Product name	Type number	Firmware version	PCN reference	Product status
EMMY-W1 series			N/A	Initial production

Contents

1 Introduction

This document defines the essential steps to follow when integrating EMMY-W1 series modules (based on a Marvell chipset) with the Android 9.0 package. The information contained in this document and its references will be helpful to successfully integrate the Android software package with the EMMY-W1 modules.

Note:

- Even though this document is written with EMMY-W1 as the reference module, the same procedure can be extended to any Marvell-based chipsets in the future. It might need some additional changes with respect to the Android package.
- The description of the integration is based on the MCIMX6Q-SDB host platform from NXP.
- The Android package is provided by NXP. It is a customized version of the Android open source package pie 9.0.0_2.2.0.
- The procedure remains the same independent of the platform and package. Every vendor who manufactures boards for Android development will have to ensure that they provide the right set of kernel package and Android sources.

2 Device setup and components

2.1 Host platform

The hardware platform used in this guide is MCIMX6Q Smart Application Blueprint for Rapid Engineering (SABRE) Development board. It has the following features:

- Industrial grade NXP i.MX 6Quad processor
- Supports both Yocto Linux and Android
- Extensive peripherals that can meet most evaluation scenarios
- Evolving update of BSP for the board

Detailed information about the board can be found at the following URL:
https://www.nxp.com/design/development-boards/i.mx-evaluation-and-development-boards/sabre-board-for-smart-devices-based-on-the-i.mx-6quad-applications-processors:RD-IMX6Q-SABRE

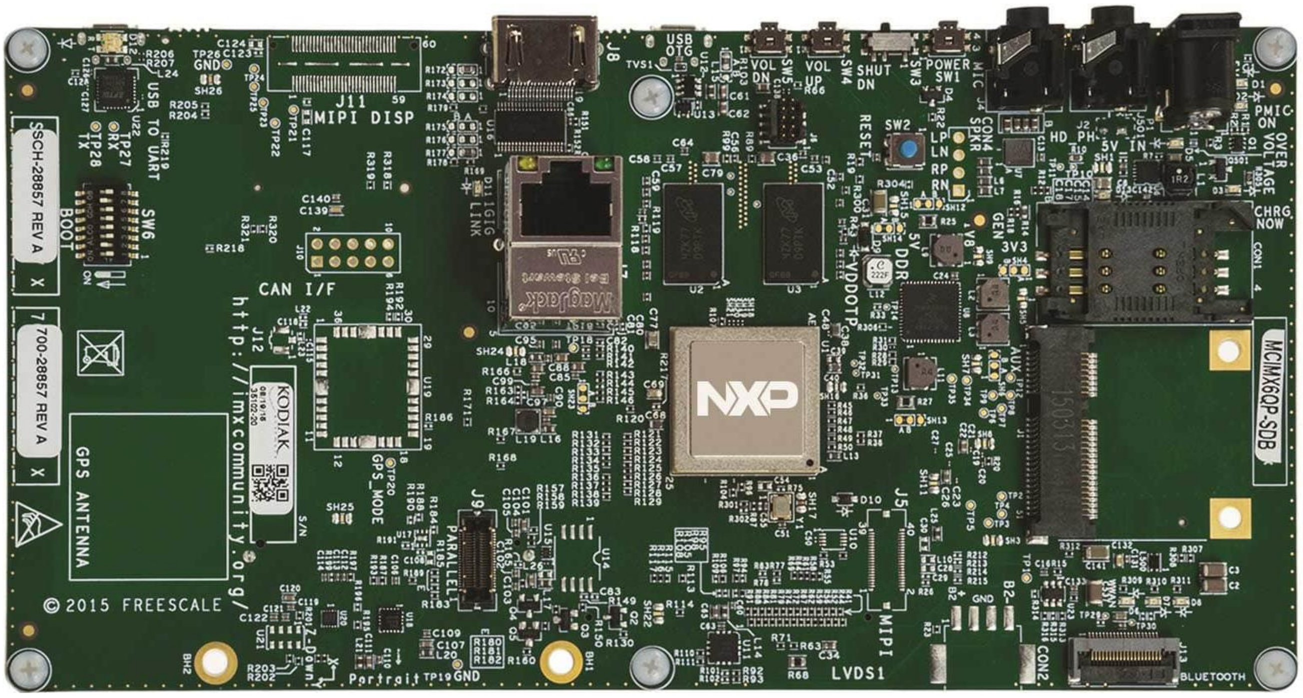


Figure 1: NXP MCIMX6Q Sabre board

2.2 Build system

1. A Linux host (for example Ubuntu 14.04) is required. This is needed to compile the Android code. Recommended hardware requirement is:
   - 8 GB RAM
   - 500 GB hard disk
2. A Windows PC with Windows OS version 7, 8, or 10 installed. This PC is needed to initially flash the SD card with the image built using AOSP.

2.3 Software packages

Different packages and their sources are explained in the following sections.

2.3.1 Android AOSP

The following software packages are available for the board:

- Android 8.1.0
- Android 9.0.0

This document assumes that Android 9.0.0 (Android Pie) is used for integrating EMMY-W1. The customized variant of the open source package is provided by NXP along with the EVK.

The version of the Linux kernel integrated with the image is: *4.14.98-dirty.*

Follow the steps below to build the image for the hardware:

1. Setup for download

   For using the manifest repository, the "repo" tool must be installed first.

   ```
   $: mkdir ~/bin
   $: curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
   $: chmod a+x ~/bin/repo
   $: export PATH=${PATH}:~/bin
   ```

2. Download the manifest

   Download and extract the source directory from the link:

   https://www.nxp.com/webapp/Download?colCode=P9.0.0_2.2.0_GA_ANDROID_SOURCE&appType=license

   It will have the necessary configurations to fetch the packages from different repositories.

3. Setup Source

   Run the setup script to download Android open source and copy proprietary source code to different folders. The proprietary code is updated on top of Android open source by NXP to cater to the platforms supported. This step will typically take 5-6 hours depending on the network speed and machine.

   ```
   $: source ./imx-p9.0.0 2.2.0-ga/imx android setup.sh
   ```

☞ Ensure that the script is run from the above-mentioned folder for the setup script to work properly. At this point, all the files and directories needed for Android image building are downloaded. Integrate the device driver and make necessary changes for integrating Wi-Fi and Bluetooth before starting the build procedure.

4. Setup build environment

   This configures the current terminal for Android build:

   ```
   $: source build/envsetup.sh
   ```

5. Execute the Android lunch command.

   The lunch command can be issued with an argument or without the argument through a selection menu. Select different types of images based on the requirement:

   ```
   $: lunch sabresd_6dq-eng
   ```

6. Execute the make command to generate the image:

```
$: make -j BUILD_TARGET_DEVICE=sd 2>&1 | tee build-log.txt
```

There are three provisions to build the image depending on the need.

Build type	Description
user	Production-ready image, no debug
userdebug	Provides image with root access and debug, similar to "user"
eng	Development image with debug tools

Table 1: Image building options for EVK

2.3.2 Driver package

The reference driver developed by Marvell is distributed by u-blox to those customers who have signed the limited use license agreement (LULA-M) with u-blox. Contact u-blox support for your area as listed in the Contact section for information about how to obtain the driver package.

The 88W8887 chip is used in the EMMY-W1 module. Updated driver packages should also work in the similar way as the release mentioned below. A brief description of the package can be found below:

```
SDUART.8887.U16-MMC-W15.44.19.p19-15.100.19.p19-C4X15632_A2-MGPL (04 January 2019)
├── FwImage                          ┐
│   ├── sd8887_wlan_a2.bin           │      sduart8887_combo_a2.bin is used for operating
│   ├── sduart8887_combo_a2.bin      ┤      Wi-Fi and Bluetooth modules in Android. Rest
│   └── uart8887_bt_a2.bin           ┘      are needed for individual radios.
├── SD-UAPSTA-8887-U16-MMC-W15.44.19.p19-C4X15632_A2-app-src.tgz
├── SD-UAPSTA-8887-U16-MMC-W15.44.19.p19-C4X15632_A2-MGPL-src.tgz
├── SD-UAPSTA-8887-U16-MMC-W15.44.19.p19-C4X15632_A2-mlan-src.tgz
└── UART-BT-8887-U16-X86-15.100.19.p19-2.2-M2614100-GPL-src.tgz
```

2.3.3 Wi-Fi vendor HAL

Similar to driver package vendor HAL is also distributed by u-blox to the customers. We have used vendor HAL Version 004 in this integration. But the structure should remain same for the packages coming up in the future. Extracted contents from the libraries must be kept in specific path to compile along with the Android system build.

```
Marvell_Vendor_hal_004
├── libbt                   → Source code for vendor HAL for Bluetooth
│   └── conf                → Conf file specific to Bluetooth of the combo chip
└── wifi_hal_004
    ├── 1.2                 → Service to integrate with native Wi-Fi framework
    ├── libwifi-hal            Library which provides services specific to scanning
    │   └── wifi_hal-mrvl      collection of stats used by the host framework.
    └── wlan_lib            → Glue layer between wpa_supplicant and the framework
```

3 Wi-Fi integration

3.1 Wi-Fi architecture

Android application uses managers to access system services like Wi-Fi. These managers use various Hardware Abstraction Layer (HAL) to access some of the vendor-specific functionalities in each layer. As shown below, different layers are implemented to access the Wi-Fi hardware.

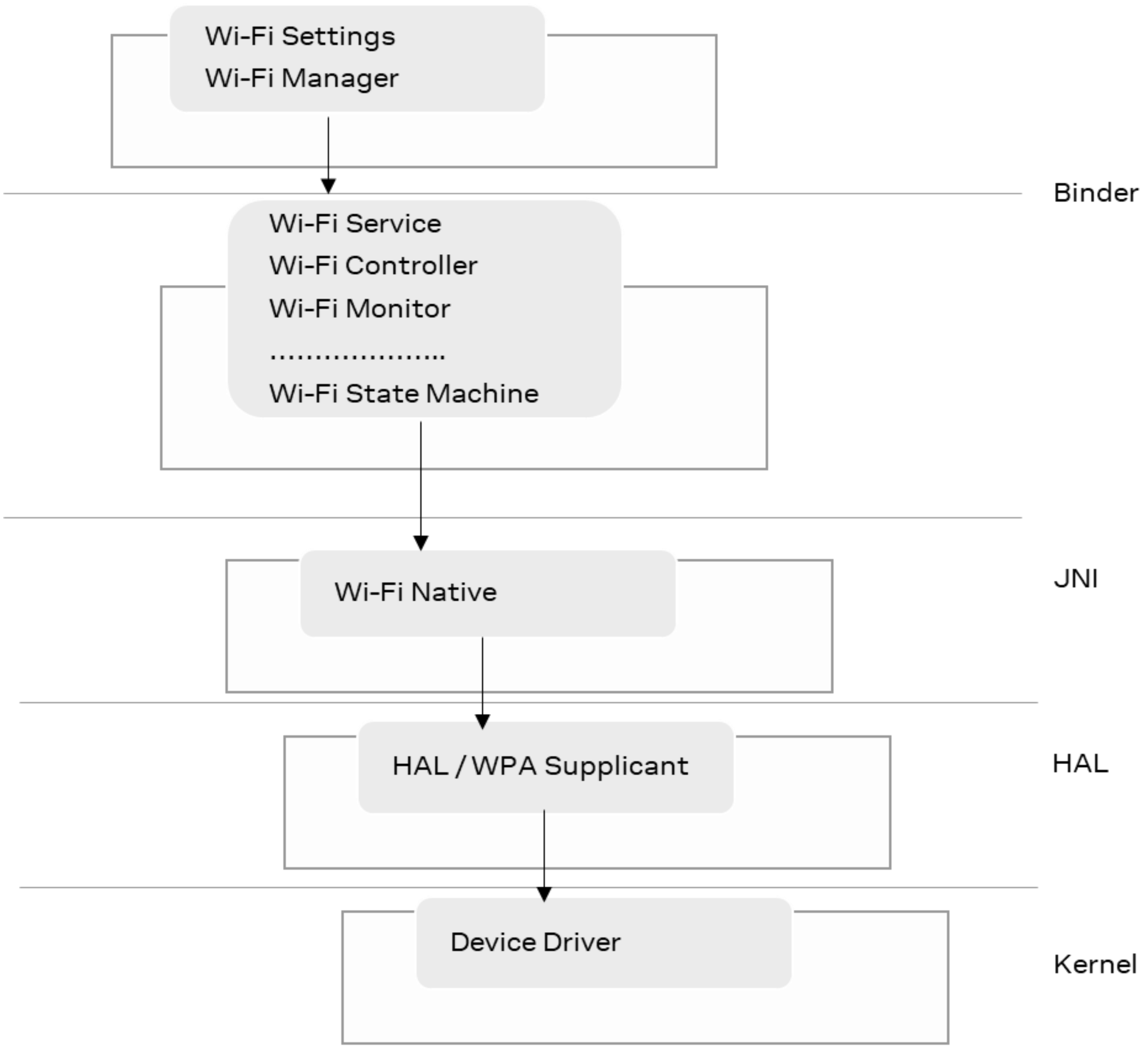


Figure 2: Android Wi-Fi architecture

WifiSettings is an application in the default AOSP build used to control Wi-Fi connections. It uses WifiManager to get access to Wi-Fi services. WifiManager provides the following functionalities:

1. Provides a list of configured networks
2. Monitors the current active Wi-Fi network
3. Provides results of access point scans with enough information to prioritize and decide upon the best choice.

The callbacks and handling in Android Wi-Fi state machine are explained in Figure 3. Every action done in the context of Wi-Fi will cut across these different layers.

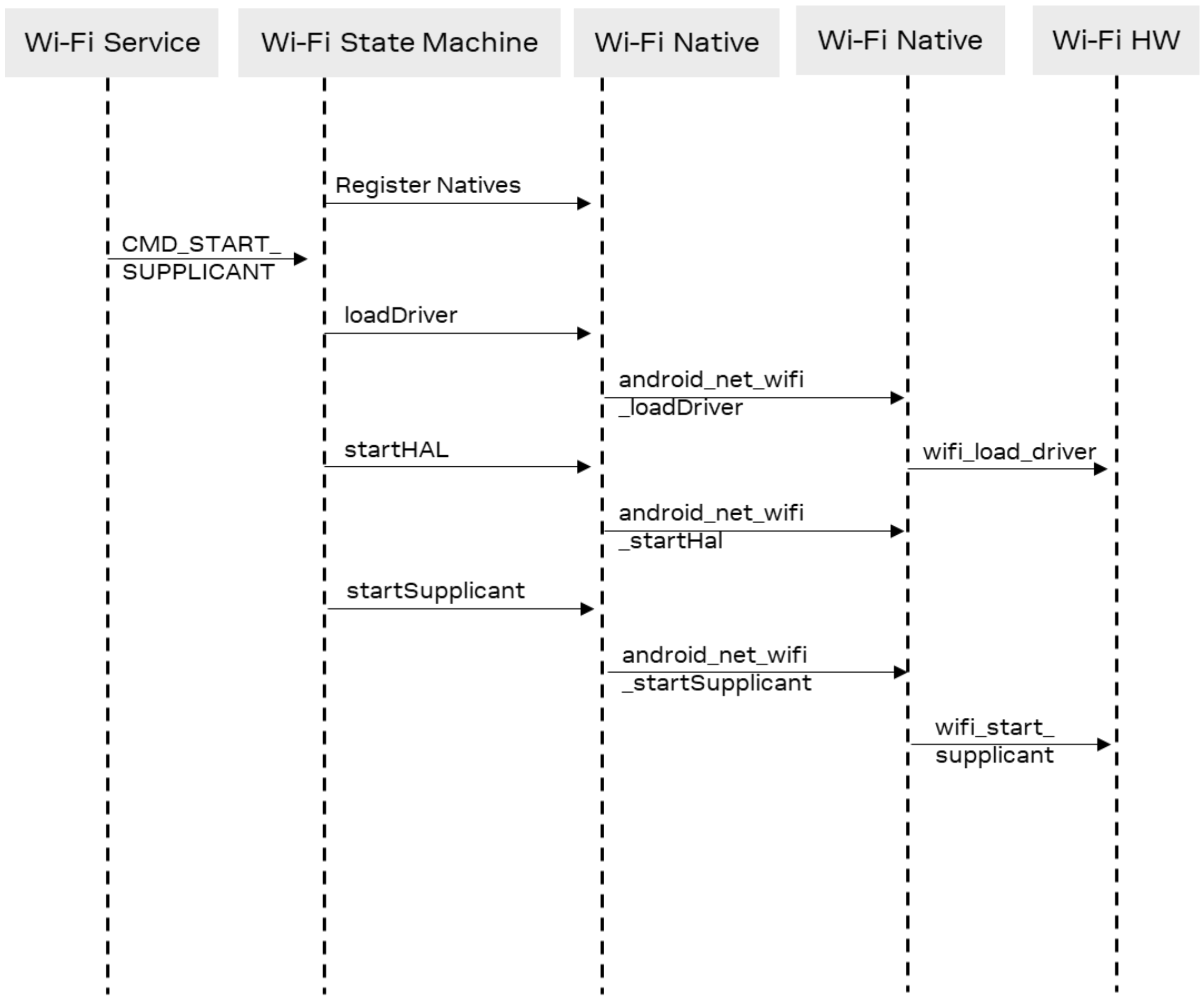


Figure 3: Sequence diagram of Wi-Fi initialization

☞ Most of the problems are encountered during the time of loading. The above-mentioned Wi-Fi state machine should help in debugging such issues. Once the Wi-Fi is enabled, further operations like scan, join etc. are carried out smoothly since they do not need any external intervention from vendor-specific layers.

To get the Wi-Fi driver up and running, some changes must be made to the files and configurations. Different files are provided by the chip vendor and some files must be altered to cater to our needs.

3.2 Components

3.2.1 Manifest

These changes are needed to choose the right HIDL interface among the options integrated with AOSP. The Wi-Fi HAL selected in the manifest here uses HIDL interface to talk to WifiServices. The manifest should contain the tags shown below; add the same if they are not present:

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/manifest.xml
    <hal format="hidl">
       <name>android.hardware.wifi</name>
       <transport>hwbinder</transport>
       <version>1.2</version>
       <interface>
           <name>IWifi</name>
           <instance>default</instance>
       </interface>
    </hal>
    <hal format="hidl">
       <name>android.hardware.wifi.supplicant</name>
       <transport>hwbinder</transport>
       <version>1.1</version>
       <interface>
           <name>ISupplicant</name>
           <instance>default</instance>
       </interface>
    </hal>
    <hal format="hidl">
        <name>android.hardware.wifi.hostapd</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>IHostapd</name>
            <instance>default</instance>
        </interface>
    </hal>
```

3.2.2 Driver source

From the package received from Marvell, contents of different folders must be arranged in a particular format for the build. The AOSP package already has an open source driver from Marvell. This section explains the procedure to integrate the latest proprietary driver with Android.

```
├── muart_src      → Copy the contents of UART-BT-8887-XX into this folder
└── wlan_src       → Copy the contents of SD-UAPSTA-8887-XX into this folder
    ├── Makefile
    ├── mapp                     → Applications for configurations
    ├── mlan                     ┐ Driver code for Wi-Fi module.
    ├── mlinux                   ┘ 2 modules are built out of this source sd8xxx, mlan.
    ├── README                   ┐
    ├── README_MLAN              │
    ├── README_OPENWRT           ├ Readme files with instructions and configurations
    ├── README_UAP               │
    ├── README_WIFIDIRECT        ┘
    └── script
```

Copy the extracted folder to the destination directory as mentioned below:

```
Source: Marvell package as extracted above

Destination Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887
```

3.2.2.1 Enable Android feature

Enable Android kernel compilation if not already done in the package downloaded from Marvell.

```
Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887/wlan_src/Makefile

CONFIG_ANDROID_KERNEL=y
```

3.2.2.2 Build driver

Select the MRVL8887 driver in the kernel configuration for the board. Check for the macro used for selecting the config file.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/BoardConfig.mk

TARGET_KERNEL_DEFCONFIG := imx_v7_android_defconfig
```

☞ The macro used for kernel config file might be different based on the Android versions. But it will be defined in the Boardconfig file.

The file mentioned as kernel configuration is present in the kernel source tree.

```
File: AndroidSrc/vendor/nxp-
opensource/kernel_imx/arch/arm/configs/imx_v7_android_defconfig

CONFIG_WLAN_VENDOR_MARVELL=y
CONFIG_MRVL8887=m
```

Once the driver is selected, to build the module, enable the kernel module in the files from the below-mentioned location:

```
Path: AndroidSrc/vendor/nxp-opensource/kernel_imx/drivers/
/net/wireless/marvell/Makefile
obj-$(CONFIG_MRVL8887) += mrvl8887/wlan_src/
obj-$(CONFIG_MRVL8887) += mrvl8887/muart_src/

Path: AndroidSrc/vendor/nxp-opensource/kernel_imx/drivers/net/wireless/marvell/Kconfig
source "drivers/net/wireless/marvell/mrvl8887/Kconfig"
```

Once the driver is copied correctly, it also needs a Makefile and Kconfig so that the drivers are built along with the Android package along with the kernel. Two samples listed below can be used for compiling the drivers.

```
Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887/Kconfig

config MRVL8887
        tristate "Marvell WiFi Driver for SD8887"
        default m
        depends on CFG80211
        ---help---
          This adds support for wireless adapters based on Marvell
          802.11n/ac chipsets. If you choose to build it as a module,
          it will be build 2 modules sd8xxx.ko and mlan.ko
```

☞ Kconfig file presented above is written for completeness of the build procedure. It does not add any extra configurations to the driver.

```
Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887/Makefile

ANDROID_CROSS_COMPILE:=$(abspath .)/prebuilts/gcc/linux-x86/arm/arm-linux-androideabi-
4.9/bin/arm-linux-androideabi-
MRVL_ANDROID_SRC_WLAN:= $(LOCAL_PATH)/wlan_src
MRVL_ANDROID_SRC_MUART:= $(LOCAL_PATH)/muart_src

default:
        $(MAKE) -j -C $(KDIR) M=$(MRVL_ANDROID_SRC_WLAN) ARCH=arm
CROSS_COMPILE=$(ANDROID_CROSS_COMPILE) SOURCE_DIR=$(MRVL_ANDROID_SRC_WLAN) modules
        $(MAKE) -j -C $(KDIR) M=$(MRVL_ANDROID_SRC_MUART) ARCH=arm
CROSS_COMPILE=$(ANDROID_CROSS_COMPILE) SOURCE_DIR=$(MRVL_ANDROID_SRC_MUART) modules

clean:
        $(MAKE) -C $(MRVL_ANDROID_SRC_WLAN) clean
        $(MAKE) -C $(MRVL_ANDROID_SRC_MUART) clean
```

☞ Some of the paths mentioned in the Makefile above might vary depending on the build and Android package from the host platform vendor. Ensure that changes are made accordingly.

Once all the files are copied, the final directory should look exactly as shown below:

```
Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887

├── Kconfig   ┐     New files as mentioned above for compiling drivers.
├── Makefile  ┘
├── muart_src
└── wlan_src
    ├── mapp
    ├── mlan
    └── mlinux
```

3.2.3 Firmware binary

Different variants of the firmware are added in the package by default. Update the package from Marvell for latest features and bug fixes. Choose one of the images from below based on the use case scenario. Important images that are used for operation are mentioned in Table 2.

Firmware Image	Description
sd8887_wlan_a2.bin	Supports only Wi-Fi operation
sduart8887_combo_a2.bin	Supports both Wi-Fi and Bluetooth operation
uart8887_bt_a2.bin	Supports only Bluetooth over UART

Table 2: Firmware images used with the EMMY-W1 module

To copy the final image, firmware files must be copied to the following path:

```
Source: SDUART.8887.U16-MMC-W15.44.19.p19-15.100.19.p19-C4X15632_A2-MGPL/FwImage

Destination Path: AndroidSrc/vendor/nxp/imx-firmware/mrvl/88W8887/
```

☞ Choose the firmware binary based on the usage of the module for applications.

3.2.4 Vendor HAL

Marvell provides a vendor HAL library to be integrated for vendor-specific implementation of the native commands used in the upper layers. Copy the entire directory from the extracted package as mentioned below:

```
Source: Marvell_Vendor_hal_004

Destination Path: AndroidSrc/hardware/marvell/
```

Once extracted and copied the folder should look like below:

```
AndroidSrc/hardware/marvell/
├── bt
│   ├── Android.mk
│   └── libbt-vendor
└── wlan
    ├── 1.2
    ├── Android.mk
    ├── config                  → Folder to keep the p2p and wpa config files
    ├── libwifi-hal
    └── wlan_lib
```

3.2.5 Configuration files

WPA supplicant is a configuration utility that uses some basic parameters from a configuration file in the root file system. Contents of the configuration file should be as shown below:

```
Path: AndroidSrc/hardware/marvell/wlan/config/wpa_supplicant.conf

update_config=1
eapol_version=1
ap_scan=1
fast_reauth=1
pmf=1
p2p_add_cli_chan=1
```

Similar to station mode operation P2P mode also needs a configuration file as shown below:

```
Path: AndroidSrc/hardware/marvell/wlan/config/p2p_supplicant.conf

update_config=1
eapol_version=1
ap_scan=1
fast_reauth=1
pmf=1
p2p_add_cli_chan=1
p2p_no_group_iface=1
```

3.3 Integration with Android

3.3.1 Board configuration

Enable Marvell supplied components to be used with the Android. This includes defining some libraries, drivers etc.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/BoardConfig.mk

BOARD_WLAN_DEVICE                         := mrvl
BOARD_HAVE_BLUETOOTH_MRVL                 := true
BOARD_HOSTAPD_PRIVATE_LIB                 := lib_driver_cmd_mrvl
BOARD_WPA_SUPPLICANT_PRIVATE_LIB          := lib_driver_cmd_mrvl
BOARD_VENDOR_KERNEL_MODULES += \
        $(KERNEL_OUT)/drivers/net/wireless/marvell/mrvl8887/muart_src/hci_uart.ko \
        $(KERNEL_OUT)/drivers/net/wireless/marvell/mrvl8887/wlan_src/mlan.ko \
        $(KERNEL_OUT)/drivers/net/wireless/marvell/mrvl8887/wlan_src/sd8xxx.ko
```

☞ Along with defining Marvell components, ensure that configurations pertaining to other Wi-Fi and Bluetooth vendors are disabled in this file. For example, `BOARD_HAVE_BLUETOOTH_BCM`, `WIFI_DRIVER_FW_PATH_PARAM` must be disabled to avoid compilation and run-time errors.

3.3.2 Enable HAL library

Enable Marvell provided HAL implementation library.

```
File: AndroidSrc/frameworks/opt/net/wifi/libwifi_hal/Android.mk

LIB_WIFI_HAL := libwifi-hal-mrvl
```

3.3.3 File placement

The board-specific Makefile in the Android source code must include the desired changes in order to copy the files to the right directories to be used.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/sabresd_6dq.mk

# Wi-Fi and Bluetooth Firmware
PRODUCT_COPY_FILES += vendor/nxp/imx-
firmware/mrvl/88W8887/sduart8887_combo_a2.bin:$(TARGET_COPY_OUT_VENDOR)/firmware/mrvl/
sduart8887_combo_a2.bin

# WiFi Marvell HAL libraries
PRODUCT_PACKAGES += \
    libbt-vendor \
    lib_driver_cmd_mrvl

# Supplicant config files
PRODUCT_COPY_FILES += \
hardware/marvell/wlan/config/wpa_supplicant.conf:vendor/etc/wifi/wpa_supplicant.conf \
hardware/marvell/wlan/config/p2p_supplicant.conf:vendor/etc/wifi/p2p_supplicant.conf
```

3.3.4 Additional packages

The Wi-Fi hardware abstraction layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework. It has the following two different HIDL packages:

- Vendor HAL
- Supplicant HAL

When a framework API makes a call to access device hardware interface, the Android system loads the library module for that hardware component. These abstraction layers are already part of the AOSP and are chosen as part of the manifest changes in section 3.2.1.

A few more packages that are needed are listed below:

1. **android.hardware.wifi@1.0-service**: This is the root of the HAL module and is the service invoked by the native framework.

2. **Wifilogd**: This is a logging module used for debug and analysis
3. **Wificond**: It communicates with the Wi-Fi driver over standard `nl80211` commands.

Add the components in the board specific file as shown below:

```
File: /AndroidSrc/device/fsl/imx6dq/sabresd_6dq/sabresd_6dq.mk

# Wi-Fi additional packages
PRODUCT_PACKAGES += \
    android.hardware.wifi@1.0-service \
    wifilogd \
    wificond
```

These are part of AOSP. They must be enabled during compilation if not already done by default.

3.3.5 Define interfaces

Set the global interfaces for Wi-Fi operation during board initialization

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/init.rc

    setprop wlan.driver.status ok
    setprop wifi.interface wlan0
    setprop wifi.direct.interface p2p0
```

3.3.6 Starting supplicant

Add a service to invoke the supplicant when triggered by the upper host layers.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/init.rc

service wpa_supplicant /vendor/bin/hw/wpa_supplicant \
    -ip2p0 -Dnl80211 -c/vendor/etc/wifi/p2p_supplicant.conf \
    -puse_multi_chan_concurrent=1 \
    -N -iwlan0 -Dnl80211 -c/vendor/etc/wifi/wpa_supplicant.conf \
    -puse_multi_chan_concurrent=1 \
    -C /data/vendor/wifi/wpa/sockets \
    -e/data/misc/wifi/entropy.bin -g@android:wpa_wlan0
    class main
    socket wpa_wlan0 dgram 660 wifi wifi
    user root
    group root
    disabled
    oneshot
```

3.3.7 Loading the Wi-Fi driver

Modules are loaded using an insmod script which uses a configuration file with paths to the kernel modules along with the configuration parameters. Both script and configuration file are put up in the excerpts below.

```
Path: AndroidSrc/device/fsl/common/init/init.insmod.sh
#! /vendor/bin/sh

# cfg file format:
# [path for modules along with parameters]

cfg_file=$1

if [ -f $cfg_file ]; then
  while IFS=" " read -r line
  do
    insmod $line
  done < $cfg_file
fi

# set property even if there is no insmod config
# as property value "1" is expected in early-boot trigger
setprop $2 1
```

Configuration file for the insmod script.

```
Path: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/early.init.cfg
vendor/lib/modules/mlan.ko
vendor/lib/modules/sd8xxx.ko fw_name=mrvl/sduart8887_combo_a2.bin cal_data_cfg=none
cfg80211_wext=0xf sta_name=wlan wfd_name=p2p p2p_enh=1 drv_mode=0x5
vendor/lib/modules/hci_uart.ko
```

Once the script and the configuration files are placed in, they must be invoked during init time.

```
Path: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/init.rc
on early-init
    start early_init_sh

service early_init_sh /vendor/bin/init.insmod.sh /vendor/etc/early.init.cfg
sys.all.early_init.ready
    class main
    user root
    group root system
    disabled
    oneshot
```

4 Bluetooth integration

4.1 Bluetooth architecture

To enable Bluetooth, execute the right set of configuration and files. Overview of the stack is shown in Figure 4.

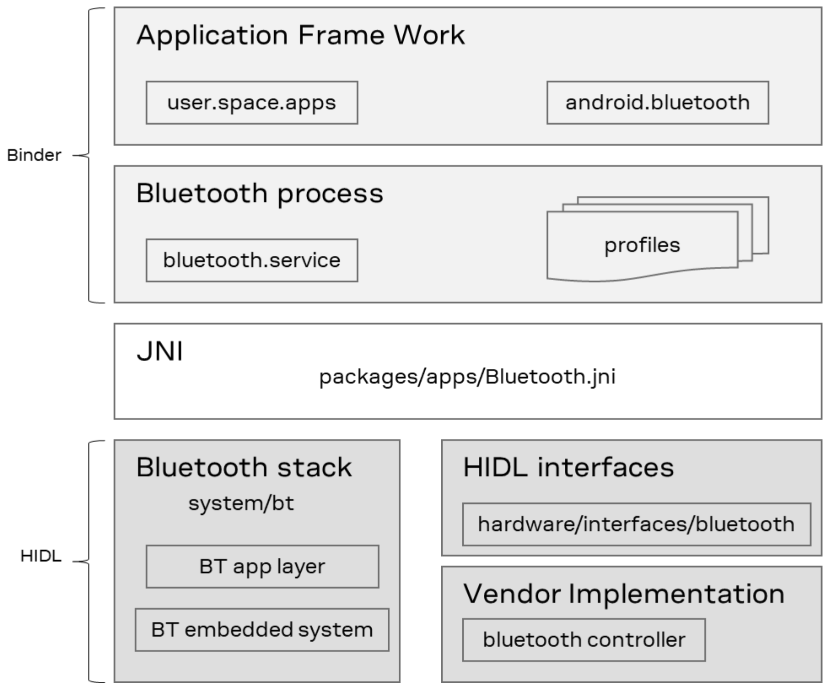


Figure 4: Bluetooth architecture

4.2 Components

4.2.1 Driver source

AOSP package comes by default with a HCI UART driver embedded into the kernel. Marvell provides a modified version of HCI UART code, which is needed for the EMMY-W1 series modules. Make changes to enable HCI UART driver provided by Marvell. These entries defined might already be present with different compile options like (m or y). Make sure they are changed as mentioned below. Configuration option CONFIG_USB_SERIAL_FTDI_SIO is needed only if you are accessing UART interface over FTDI.

```
File: AndroidSrc/vendor/nxp-opensource/kernel_imx/arch/arm/configs/
imx_v7_android_defconfig
CONFIG_BT_HCIUART=m
CONFIG_USB_SERIAL=y
CONFIG_USB_SERIAL_FTDI_SIO=y
```

Compilation part of the HCI UART driver is provided in section 3.2.2.2. It will take care of building the HCI UART proprietary driver. The information provided in section 3.3.1 will copy the drivers built to the specific location to load during initialization phase of Android.

4.3 Integration with Android

As mentioned above, `Marvell_Vendor_hal_004` contains the vendor library for Bluetooth. If it is copied in the right path mentioned below, it will be built along with the other modules.

```
Path: AndroidSrc/hardware/marvell/
       ├── libbt ➔ Vendor HAL for bluetooth
       └── wlan
```

4.3.1 Makefile changes

To specify the vendor for Bluetooth, include the variables shown below in the BoardConfig.mk file:

```
Path: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/BoardConfig.mk
BOARD_HAVE_BLUETOOTH := true
BOARD_HAVE_BLUETOOTH_MRVL := true
```

Ensure that Bluetooth configurations from other vendors are removed from BoardConfig.mk.

4.3.2 Compile time parameters

Default variables needed for Bluetooth driver build are stored in the file below. Change the default values based on the need and the application. They will be used by the libbt-vendor library.

```
Path: AndroidSrc/hardware/marvell/bt/libbt-vendor/conf/bt_vendor_8887.conf

mchar_port = /dev/ttyUSB0
pFileName_helper = /vendor/etc/firmware/mrvl/helper_uart_3000000.bin
pFileName_image = /vendor/etc/firmware/mrvl/uart8887_bt.bin
baudrate_bt = 3000000
baudrate_default = 115200
baudrate_dl_helper = 115200
baudrate_dl_image = 3000000
iSecondBaudrate = 0
download_helper = 1
uart_sleep_after_dl = 700
enable_download_fw = 1
uart_break_before_dl = 1
```

Some of the options mentioned above might not be relevant to the build and can be ignored.

4.3.3 SE-Linux labelling

Assign permission to the device node *ttyUSB0.* Even though permissions are modified using chmod at the beginning of the init, Android policy will prevent the access. Explicit permission can be imparted as below:

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/sepolicy/file_contexts

/dev/ttyUSB0            u:object_r:tty_device:s0
```

4.3.4 Permission settings

Bluetooth is connected over UART using USB. Make necessary changes to set the appropriate permissions so that the Android stack can access the port for read-write.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/init.rc

setprop hw.bluetooth 1
# /dev/ttyUSB0 is the connected port on iMX6 sabre_sd board
chmod 0666 /dev/ttyUSB0
```

☞ "/dev/ttyUSB0" is an example port that is used when the UART interface is connected over USB. This might be different based on the type of connection with the module. The port in this configuration should be in sync with the port mentioned in the vendor configuration file. See section 4.3.2 for additional information.

4.3.5 Additional packages

AOSP provides a default Bluetooth stack that supports both Bluetooth BR/EDR and Bluetooth low energy. The "`android.hardware.bluetooth`" package provides the needed interface between the Bluetooth controller and stack. Include the variable shown below in the build configuration to enable the package:

```
File: /AndroidSrc/device/fsl/imx6dq/sabresd_6dq/sabresd_6dq.mk
# Bluetooth HAL
PRODUCT_PACKAGES += \
    libbt-vendor \
    android.hardware.bluetooth@1.0-impl \
    android.hardware.bluetooth@1.0-service
```

☞ Check with your system platform provider to check whether an rfkill driver on your platform has been implemented for Bluetooth on/off control. The platform BSP should implement an rfkill driver to allow Android Bluetooth stack to control Bluetooth on/off through /sys/class/rfkill/rfkill<id>/state.

The type of rfkill<id> should be "bluetooth" to indicate that it is for Bluetooth (on/off) control. A configuration setting is required in the system init.rc file to properly set the permission of rfkill<id> so that the Bluetooth process can access it.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/init.rc

# Configure bluetooth
chmod 0660 /sys/class/rfkill/rfkill0/state
chown bluetooth bluetooth /sys/class/rfkill/rfkill0/state
```

5 Integrating SDIO-SDIO driver

This section explains the explicit changes needed different from SDIO-UART variant of the driver for EMMY with Android 9.

5.1 Driver sources and compilation

MBTC driver from SDIO-SDIO package is needed for opening the mbtchar0 device node which is used by the Bluetooth stack to communicate with the module. Even though the drivers for Wi-Fi are same, they should be built from the SDIO-SDIO package for 88W8887 chipset.

Driver package must be arranged as explained in the section 3.2.2. We have considered package SD-WLAN-SD-BT-8887-U16-MMC-W15.68.19.p22-15.26.19.p22-C4X15635_A2-MGPL.zip from Marvell to integrate with Android.

```
SD-WLAN-SD-BT-8887-U16-MMC-W15.68.19.p22-15.26.19.p22-C4X15635_A2-MGPL
├── mbtc_src        → Copy the contents of SD-BT-CHAR-8887-XX into this folder
└── wlan_src        → Copy the contents of SD-UAPSTA-8887-XX into this folder
    ├── Makefile
    ├── mapp        → Applications for configurations
    ├── mlan        ⎫  Driver code for Wi-Fi module.
    ├── mlinux      ⎭  2 modules are built out of this source sd8xxx, mlan.
    └── README
```

Copy the extracted folder to the destination directory as mentioned below:

```
Source: Marvell package as extracted above

Destination Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887
```

Once the sources are copied they need to be selected to be compiled as mentioned below.

```
Path: AndroidSrc/vendor/nxp-opensource/kernel_imx/drivers/
/net/wireless/marvell/Makefile
obj-$(CONFIG_MRVL8887) += mrvl8887/wlan_src/
obj-$(CONFIG_MRVL8887) += mrvl8887/mbtc_src/

Path: AndroidSrc/vendor/nxp-opensource/kernel_imx/drivers/net/wireless/marvell/Kconfig
source "drivers/net/wireless/marvell/mrvl8887/Kconfig"
```

☞ Note that CONFIG_MRVL8887 should be enabled before selecting the mrvl8887 folder in the kernel sources. Refer to section 3.2.2.2 for details. Makefile needs some changes to pick up the right MBTC driver path as mentioned below. Kconfig remains same.

```
Path: AndroidSrc/vendor/nxp-
opensource/kernel_imx/drivers/net/wireless/marvell/mrvl8887/Makefile

ANDROID_CROSS_COMPILE:=$(abspath .)/prebuilts/gcc/linux-x86/arm/arm-linux-androideabi-
4.9/bin/arm-linux-androideabi-
MRVL_ANDROID_SRC_WLAN:= $(LOCAL_PATH)/wlan_src
MRVL_ANDROID_SRC_MBTC:= $(LOCAL_PATH)/mbtc_src

default:
        $(MAKE) -j -C $(KDIR) M=$(MRVL_ANDROID_SRC_WLAN) ARCH=arm
CROSS_COMPILE=$(ANDROID_CROSS_COMPILE) SOURCE_DIR=$(MRVL_ANDROID_SRC_WLAN) modules
        $(MAKE) -j -C $(KDIR) M=$(MRVL_ANDROID_SRC_MBTC) ARCH=arm
CROSS_COMPILE=$(ANDROID_CROSS_COMPILE) SOURCE_DIR=$(MRVL_ANDROID_SRC_MBTC) modules

clean:
        $(MAKE) -C $(MRVL_ANDROID_SRC_WLAN) clean
        $(MAKE) -C $(MRVL_ANDROID_SRC_MBTC) clean
```

5.2 Integration with Android

5.2.1 File placement

Copy the firmware in the designated location. The Firmware binaries should be copied in the android source as mentioned in the section 3.2.3.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/sabresd_6dq.mk

# Wi-Fi and Bluetooth Firmware
PRODUCT_COPY_FILES += vendor/nxp/imx-
firmware/mrvl/88W8887/sd8887_uapsta_a2.bin:$(TARGET_COPY_OUT_VENDOR)/firmware/mrvl/sd8
887_uapsta_a2.bin
```

Include the drivers. Additionally mbt8xxx.ko is needed for the Bluetooth.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/BoardConfig.mk

BOARD_VENDOR_KERNEL_MODULES += \
      $(KERNEL_OUT)/drivers/net/wireless/marvell/mrvl8887/mbtc_src/mbt8xxx.ko \
      $(KERNEL_OUT)/drivers/net/wireless/marvell/mrvl8887/wlan_src/mlan.ko \
      $(KERNEL_OUT)/drivers/net/wireless/marvell/mrvl8887/wlan_src/sd8xxx.ko
```

5.2.2 Loading the driver

Follow the instructions in the section 3.3.7 for loading the driver. The configuration file for the SDIO drivers changes as below.

```
Path: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/early.init.cfg

vendor/lib/modules/mlan.ko
vendor/lib/modules/sd8xxx.ko fw_name=mrvl/sd8887_uapsta_a2.bin cal_data_cfg=none
cfg80211_wext=0xf sta_name=wlan wfd_name=p2p p2p_enh=1 drv_mode=0x5
vendor/lib/modules/mbt8xxx.ko
```

5.2.3 SE-Linux labelling

Explicit permission to use *mbtchar0* device node can be imparted as below:

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/sepolicy/file_contexts

/dev/mbtchar0            u:object_r:tty_device:s0
```

5.2.4 Permission settings

SDIO driver for Bluetooth creates a separate device node *mbtchar0* for operating with the hardware. For the vendor HAL to communicate to the driver, some permissions are needed as mentioned below.

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/init.rc

# /dev/mbtchar0 is the device node created
chmod 0666 /dev/mbtchar0
```

☞ Before enabling Bluetooth, make sure that this device node is created.

6 Flashing the SD Card

After performing the steps for Wi-Fi and Bluetooth integration and building the images as described in section 2.3.1, follow the steps below to flash the resulting images to an SD card:

1. Download the mfgtools from the below-mentioned URL and unzip it in Windows: https://www.nxp.com/support/developer-resources/evaluation-and-development-boards/sabre-development-system/sabre-board-for-smart-devices-based-on-the-i.mx-6quadplus-applications-processors:RD-IMX6QP-SABRE?tab=Design_Tools_Tab
2. Copy the following images that are built in while compiling Android OS (see section 2.3.1).

```
Source path (on build server): AndroidSrc/out/target/product/sabresd_6dq/
       boot-imx6q.img
       partition-table.img
       recovery-imx6q.img
       system.img
       u-boot-imx6q.imx
       vendor.img

Destination path (on windows machine):
android_O8.0.0_1.0.0_tools\mfgtools\Profiles\Linux\OS
Firmware\files\android\sabresd
```

☞ Note:

1. Do not delete the empty file placeholder.
2. Even though the name has android_O8.X.X, it can be used to flash Android Pie.

3. Put an 8 GB SD card into the SD card slot SD3. Use USB cable to connect the USB OTG port to the PC. Set the SW6 jumper to boot mode, that is, 00001100 (from 1-8 bit).
4. Power on the board, and then run the script "mfgtool2-android-mx6q-sabresd-sd.vbs" to flash the images to SD card. Power off the board once the flashing is complete.
5. Disconnect the USB cable from USB OTG and connect a USB hub with mouse and keyboard to the USB OTG port. Connect USB debug port to a PC with a USB cable. Connect HDMI port of the board to a monitor or TV with HDMI cable. Set the SW6 jumper to boot from SD3, that is, 01000010 (from 1-8 bit).
6. Open a USB debug console with a serial client program such as Teraterm. Power on the board. Interrupt the boot and enter uboot console. All these configurations are done to set the video and boot options. Boot arguments:

```
U-Boot > setenv fastboot_dev mmc1
U-Boot > setenv bootcmd boota mmc1
U-Boot > saveenv
U-Boot > boot
```

☞ It might not be needed to select the mmc1 to boot the machine. By default, it will choose based on the dip-switch configuration.

Additional settings:

It is not mandatory to use HDMI connection with the monitor if console interface is sufficient for design, build and production. If not, include the below-mentioned command along with other settings as explained in step 6 in the above-mentioned section.

```
U-Boot > setenv bootargs console=ttymxc0,115200 androidboot.console=ttymxc0
consoleblank=0 vmalloc=128M init=/init
video=mxcfb0:dev=hdmi,1920x1080M@60,bpp=32 video=mxcfb1:off video=mxcfb2:off
video=mxcfb3:off androidboot.hardware=freescale cma=448M
galcore.contiguousSize=33554432
```

1. When flashing the SD card for the first time, use mfgtool. In the later instances, shell script provided by NXP can be used, which is faster and easier.

```
Path: AndroidSrc/out/target/product/sabresd_6dq/
Command: ./fsl-sdcard-partition.sh -f imx6q /dev/sdX
```

2. If you use HDMI to VGA adapter and face display issues on the iMX8 board, it is recommended to use an HDMI to HDMI cable and also change the resolution as shown below:

```
File: AndroidSrc/device/fsl/imx6dq/sabresd_6dq/BoardConfig.mk
BOARD_KERNEL_CMDLINE += video=HDMI-A-1:1280x800@60
```

7 Run time usage

7.1 Validation

Once the board boots up after flashing the SD card, ensure that the following files are present in the specified location for Wi-Fi operation:

Type	File name	Path
Firmware	sduart8887_combo_a2.bin	/vendor/firmware/mrvl/
SUPPLICANT	wpa_supplicant	/vendor/bin/hw/
CLIENT	wpa_cli	/vendor/bin/
WPA CONF file	wpa_supplicant.conf	/vendor/etc/wifi/
P2P CONF file	p2p_supplicant.conf	/vendor/etc/wifi/
HOSTAPD	hostapd	/vendor/bin/hw/

Table 3: List of files

7.2 Wi-Fi bring up

Wi-Fi can be used in the following modes:

1. Station mode
2. Access point mode
3. Wi-Fi Direct mode

All the modes have been tested and verified to work with Android 9 Pie and the EMMY-W1 module.

Upon successful integration and placement of all the components, the following logs will be displayed during boot time. This will ensure that the driver and firmware are loaded correctly.

```
SettingsProvider: Notifying for 0: content://settings/global/wifi_on
WifiP2pService: Wifi enabled=false, P2P Interface availability=true
android.hardware.wifi@1.0-service: Wifi HAL started
WifiVendorHal: Vendor Hal started successfully
WifiP2pNative: Registering for interface available listener
WifiP2pNative: P2P InterfaceAvailableListener true
wpa_supplicant: Successfully initialized wpa_supplicant
wpa_supplicant: rfkill: Cannot open RFKILL control device
EthernetTracker: interfaceLinkStateChanged, iface: wlan0, up: false
EthernetTracker: interfaceLinkStateChanged, iface: wlan0, up: false
SupplicantStaIfaceHal: Completed initialization of ISupplicant.
android.hardware.wifi@1.0-service: Adding interface handle for p2p0
android.hardware.wifi@1.0-service: Adding interface handle for wlan0
android.hardware.wifi@1.0-service: Failed to register radio mode change callback
WifiNative: Interface state changed on Iface:{Name=wlan0,Id=1,Type=STA}, isUp=true
CommandListener: Clearing all IP addresses on wlan0
WifiNative: Successfully setup Iface:{Name=wlan0,Id=1,Type=STA}
WifiClientModeManager: sending scan available broadcast: false
WifiScanningService: wifi driver unloaded
WifiStateMachine: entering ConnectModeState: ifaceName = wlan0
WifiStateMachine: setupClientMode() ifacename = wlan0
wpa_supplicant: the nl80211 driver cmd is MACADDR
WifiStateMachine: Setting OUI to DA-A1-19
WifiScanningService: wifi driver loaded with scan capabilities: max buckets=16
WifiVendorHal: Driver: 15.44.19.p19 Firmware: 15.44.19.p19
```

Expected interfaces after booting the host platform:

```
wlan0     Link encap:Ethernet  HWaddr d4:ca:6e:00:1b:16  Driver wlan_sdio
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:0
p2p0      Link encap:Ethernet  HWaddr d6:ca:6e:00:1b:16  Driver wlan_sdio
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 TX bytes:0
```

Ensure that all the modules needed for Wi-Fi and Bluetooth are loaded at the boot time. This will make sure that the hardware is ready for the commands from the host.

```
sabresd_6dq:/ # lsmod
Module                  Size  Used by
hci_uart               49152  0
sd8xxx                626688  0
mlan                  471040  1 sd8xxx
```

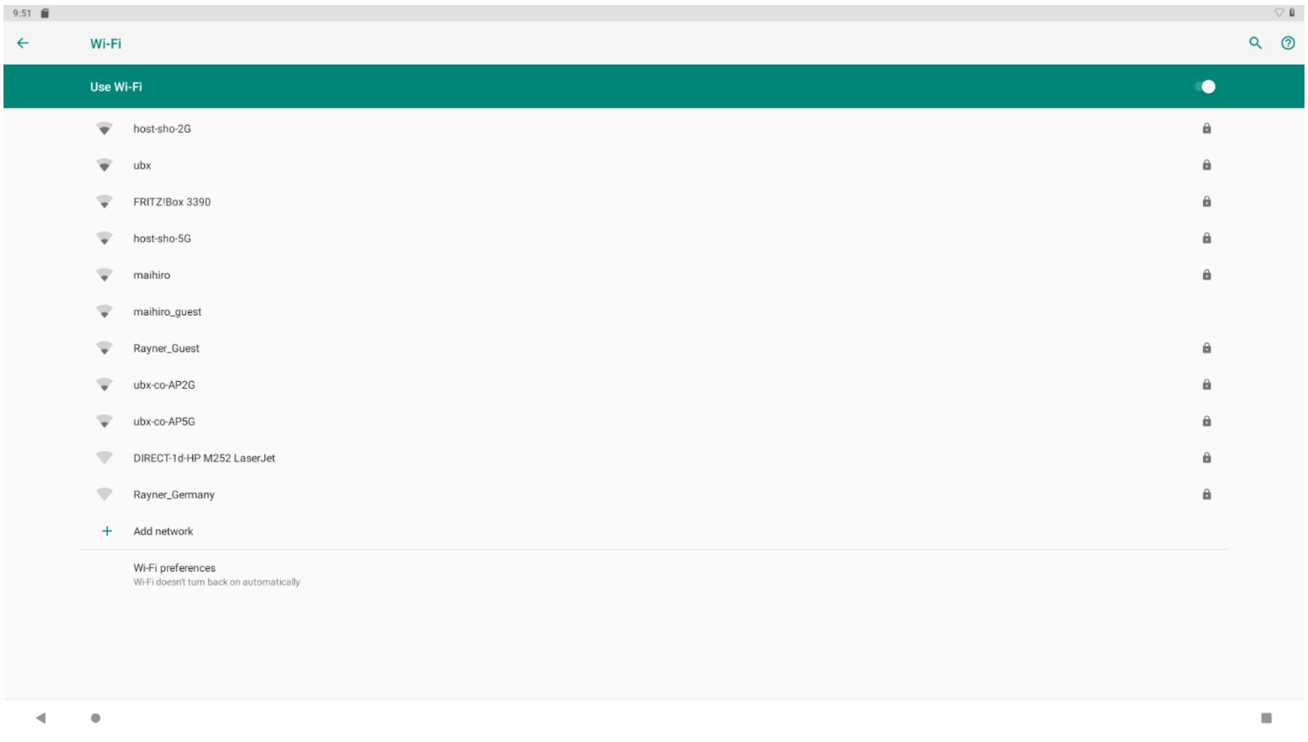
☞ Upon power-up, the Wi-Fi module will be loaded. However, this does not mean that the Wi-Fi is enabled. To switch on the Wi-Fi, Android service must be started. This in turn, translates to switching on the Wi-Fi using GUI.

```
svc wifi enable
```

This can also act as a method to verify that the firmware is loaded and Wi-Fi is started in the right manner across the layers of Android without any errors. Also, wpa_supplicant will be started at this point. Check this by searching in the currently running processes using `pgrep wpa_supplicant`.

7.3 Station mode

Using the GUI, scan and connect to the APs in the environment.



9:51

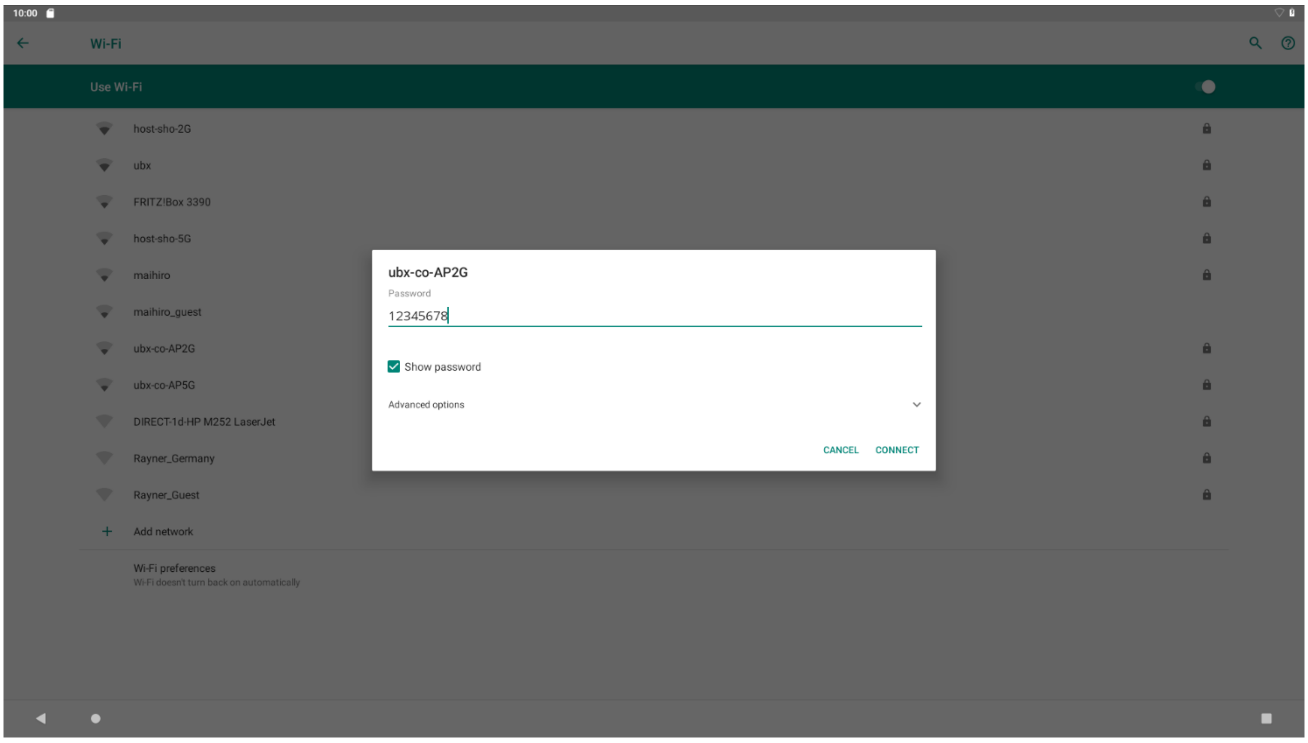
Wi-Fi

Use Wi-Fi

host-sho-2G
ubx
FRITZ!Box 3390
host-sho-5G
maihiro
maihiro_guest
Rayner_Guest
ubx-co-AP2G
ubx-co-AP5G
DIRECT-1d-HP M252 LaserJet
Rayner_Germany
Add network

Wi-Fi preferences
Wi-Fi doesn't turn back on automatically

Figure 5: Wi-Fi scanning



10:00

Wi-Fi

Use Wi-Fi

host-sho-2G
ubx
FRITZ!Box 3390
host-sho-5G
maihiro
maihiro_guest
ubx-co-AP2G
ubx-co-AP5G
DIRECT-1d-HP M252 LaserJet
Rayner_Germany
Rayner_Guest
Add network

Wi-Fi preferences
Wi-Fi doesn't turn back on automatically

ubx-co-AP2G

Password

12345678

✓ Show password

Advanced options

CANCEL CONNECT

Figure 6: Wi-Fi connecting to a secured AP (WPA2-PSK)

7.4 Access point mode

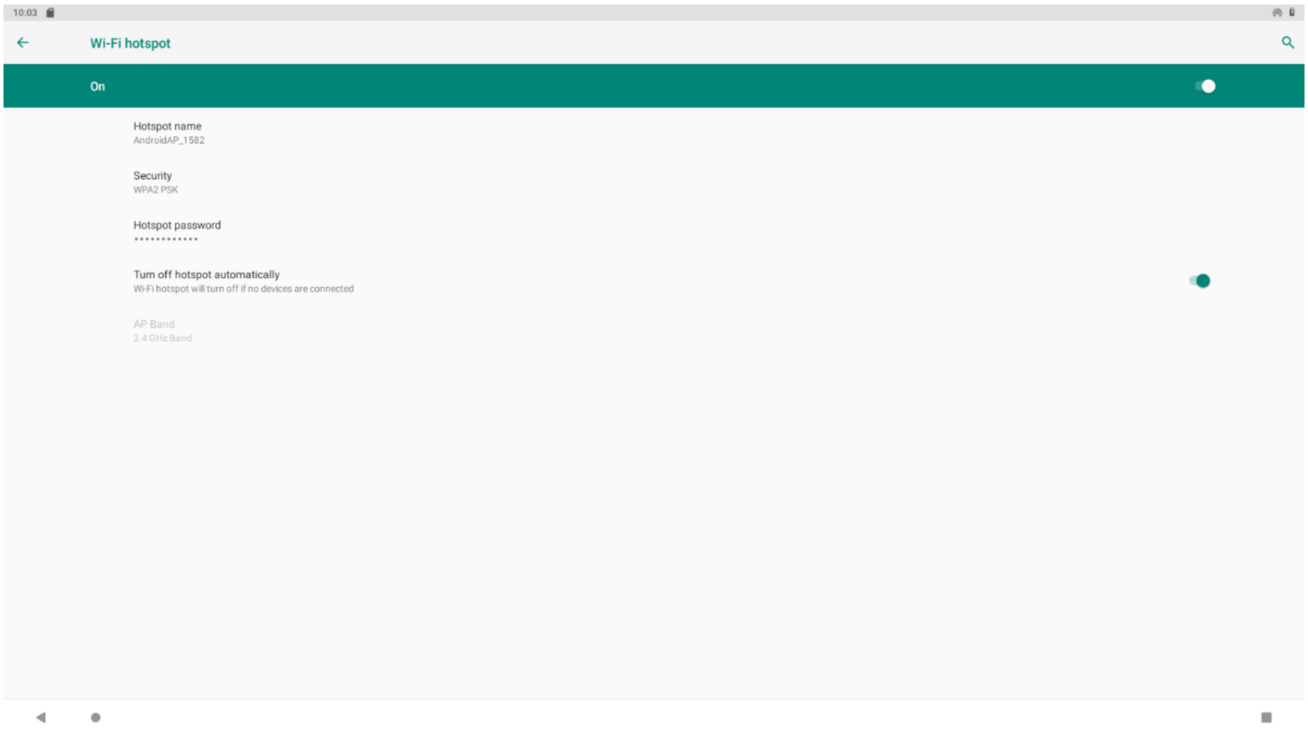
Access point mode is used for tethering the internet bandwidth from cellular with the other peers in the environment or sharing big data files between the devices.

Access point by default uses 2.4 GHz. It has a dependency on the location settings, which will determine if 5 GHz can be used. Add a SIM card so that the 5 GHz option is available in the hotspot UI.

The kernel needs some additional changes to enable 5 GHz operation with Android as mentioned below:

```
Path: AndroidSrc/vendor/nxp-opensource/kernel_imx/net/wireless/Kconfig
config CFG80211_INTERNAL_REGDB
        bool "use statically compiled regulatory rules database" if EXPERT
        default y
        depends on CFG80211
```

Along with this, a country database must also be provided while building the kernel. For that, copy db.txt file in the same location. It contains the list of channels supported in different countries. Further information can be found in the link mentioned in [5].



On

Hotspot name
AndroidAP_1582

Security
WPA2 PSK

Hotspot password
••••••••••••

Turn off hotspot automatically
Wi-Fi hotspot will turn off if no devices are connected

AP Band
2.4 GHz Band

Figure 7: Wi-Fi hotspot creation

7.5 Throughput measurements

Typically, iperf is used for measuring network throughput. An Android application can also be used for measurements over the air. If EMMY-W1 EVK is used, then conducted measurements can be performed.

Link for iperf APK: https://apkpure.com/iperf-for-android/com.magicandroidapps.iperf

7.6 Wi-Fi Direct

Wi-Fi direct enables many different use cases, including Wi-Fi display and others which work between two peers.

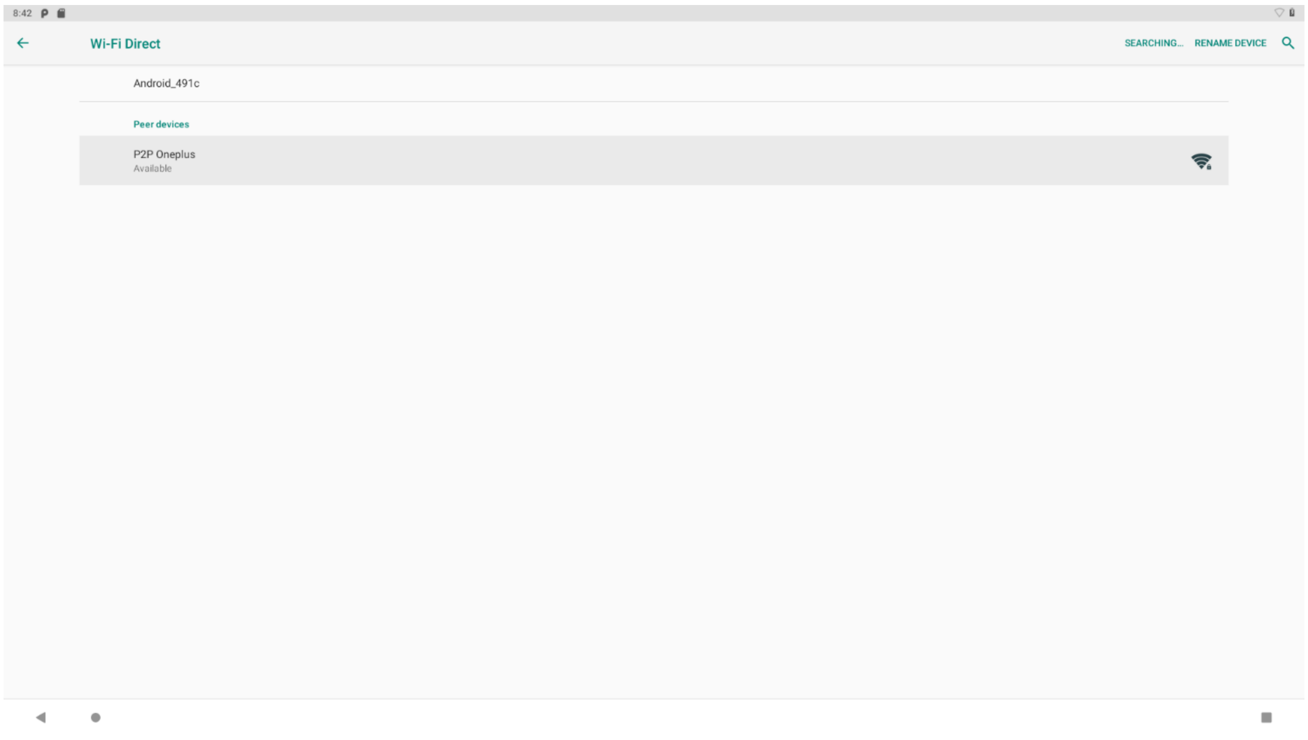
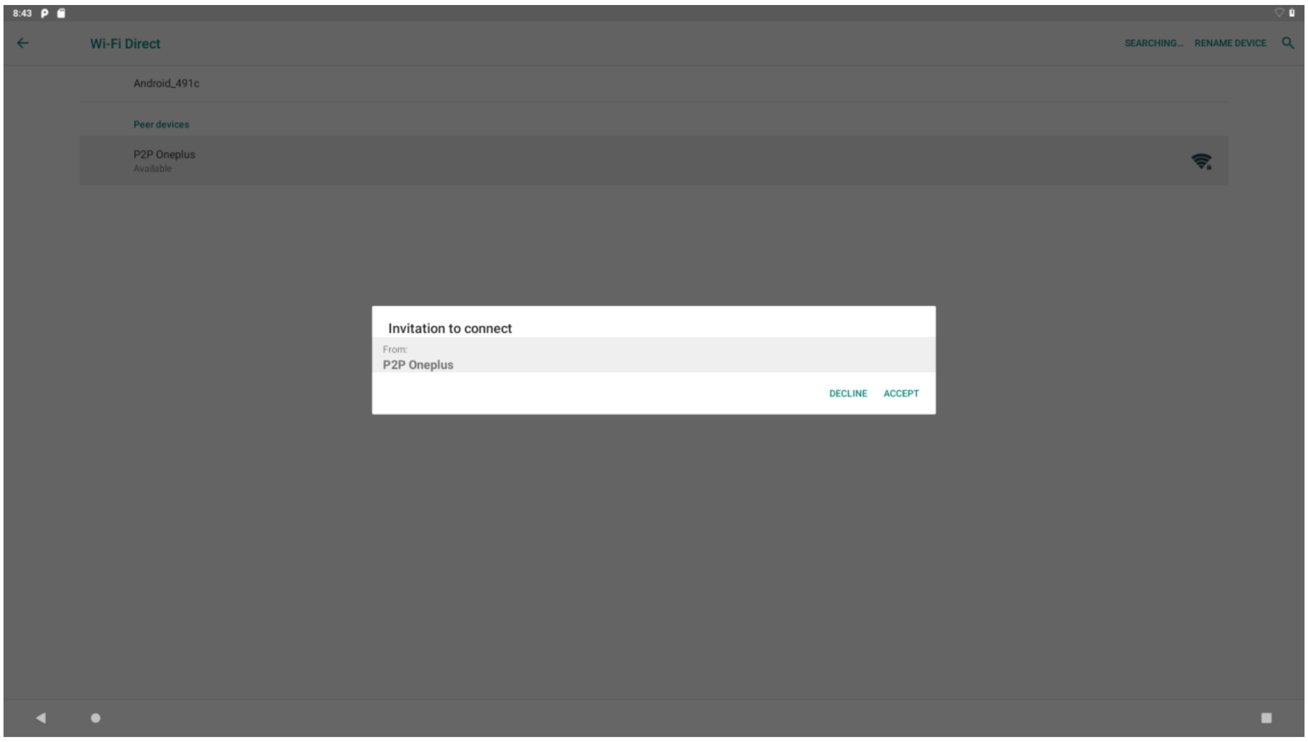


Figure 8: Wi-Fi Direct scanning



Invitation to connect

From:
P2P Oneplus

Figure 9: Wi-Fi Direct accept

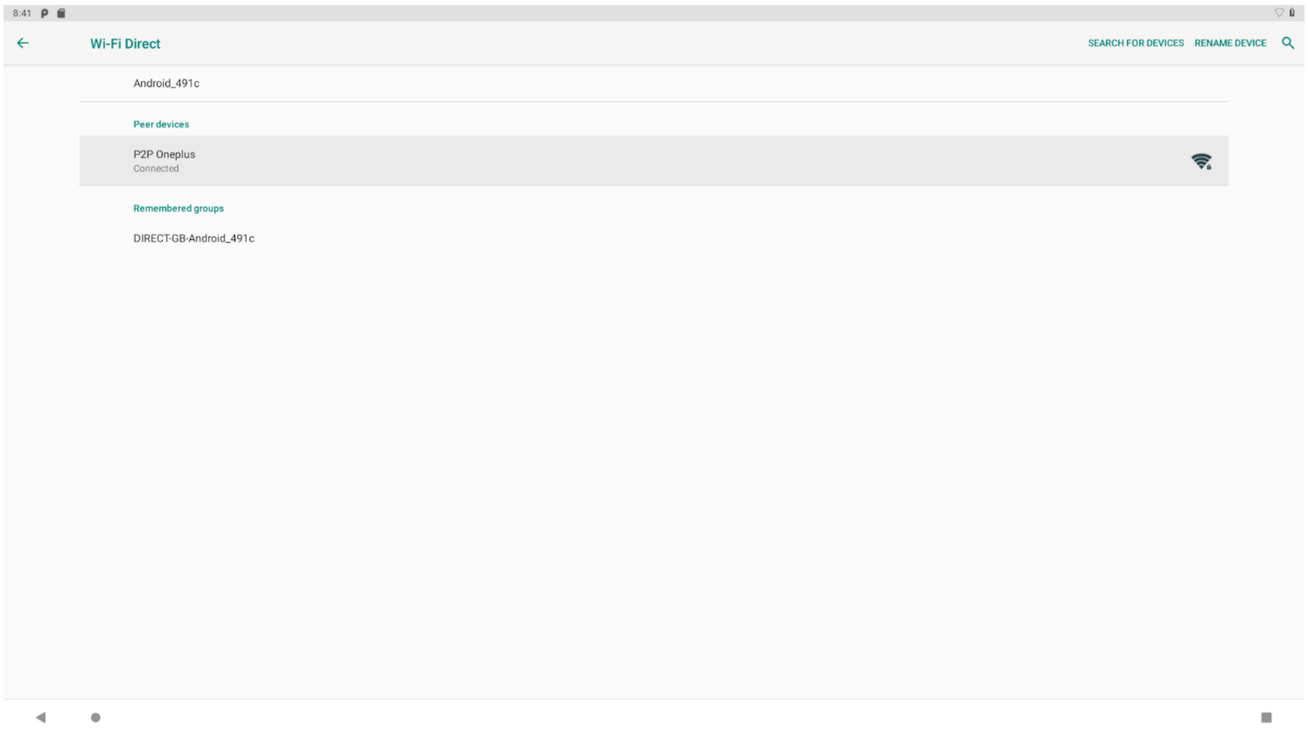


Figure 10: Wi-Fi Direct connected state

☞ Wi-Fi display functionality is not supported by the Android image provided by NXP yet. It needs some more application level libraries along with the Wi-Fi direct protocol.

7.7 Bluetooth bring up

Once the files are placed in the expected paths, Bluetooth will be enabled. Android services can also be used to enable Bluetooth using the below-mentioned command.

```
# Enable Bluetooth via command line
service call bluetooth_manager 6
# Disable Bluetooth via command line
service call bluetooth_manager 8
```

Some logs from the Android system when Bluetooth is successfully enabled are provided below:

```
bt_stack_manager: event_start_up_stack is bringing up the stack
bt_core_module: module_start_up Starting module "btif_config_module"
bt_core_module: module_start_up Started module "btif_config_module"
bt_core_module: module_start_up Starting module "btsnoop_module"
bt_core_module: module_start_up Started module "btsnoop_module"
bt_core_module: module_start_up Starting module "hci_module"
bt_hci  : hci_module_start_up
bt_osi_thread: run_thread: thread id 1896, thread name hci_thread started
bt_hci  : hci_initialize
bt_hci  : hci_module_start_up starting async portion
bt_hci  : hci_initialize: IBluetoothHci::getService() returned 0xa4086ce0 (remote)
android.hardware.bluetooth@1.0-impl: BluetoothHci::initialize()
android.hardware.bluetooth@1.0-impl: Open vendor library loaded
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L414): opcode = 0
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L426): power on --------------------------------------
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L414): opcode = 3
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L443): open serial port--------------------------------
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L449): baudrate_bt 3000000
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L450): baudrate_default 115200
android.hardware.bluetooth@1.0-impl: OnFirmwareConfigured result: 0
android.hardware.bluetooth@1.0-impl: Firmware configured in 0.000s
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L414): opcode = 5
android.hardware.bluetooth@1.0-impl: OnFirmwareConfigured: lpm_timeout_ms 0
bt-vnd-mrvl: bt_vnd_mrvl_if_op(L414): opcode = 6
android.hardware.bluetooth@1.0-impl: low_power_mode_cb result: 0
android.hardware.bluetooth@1.0-impl: OnFirmwareConfigured Calling
StartLowPowerWatchdog()
bt_hci  : event_finish_startup
bt_core_module: module_start_up Started module "hci_module"
```

7.8 Bluetooth verification

Basic Bluetooth involves a discovery scan followed by connection between the two devices. The same connection can be used for different applications such as file sharing or audio streaming. File transfer using images between the devices works successfully and has been verified.

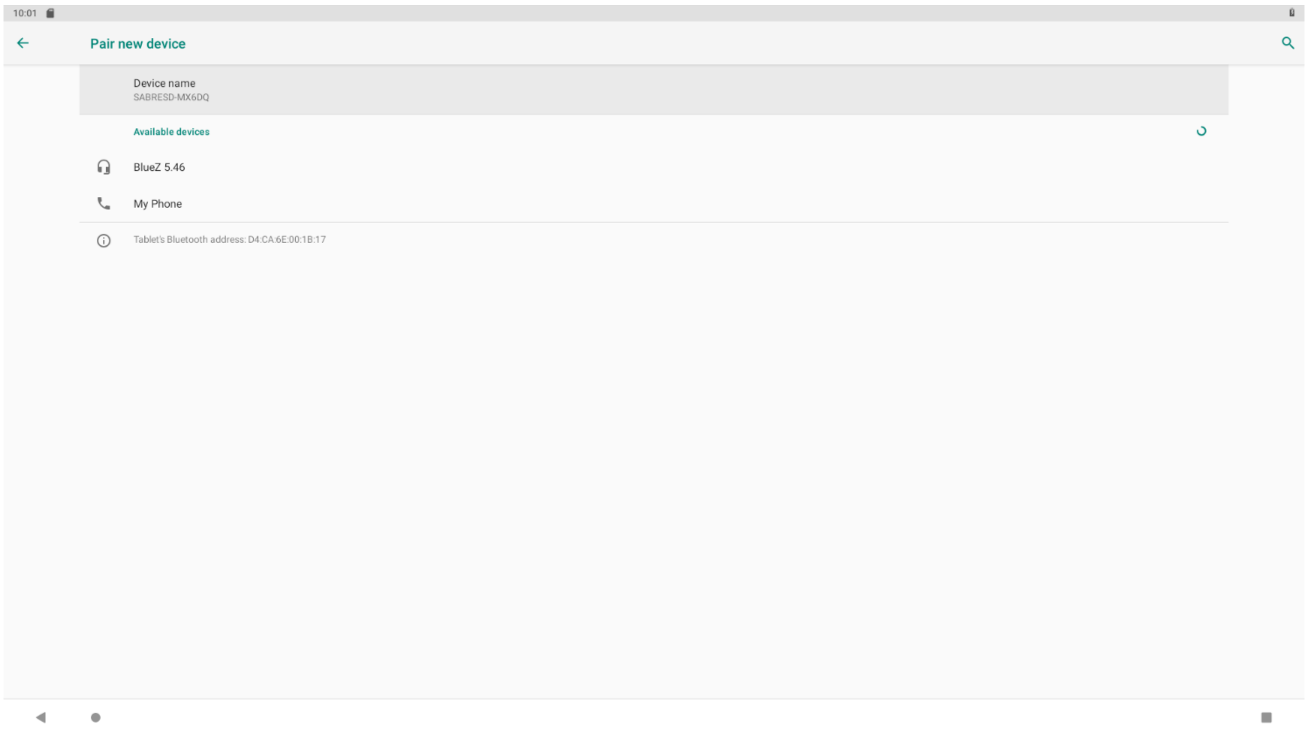


Figure 11: Bluetooth scanning

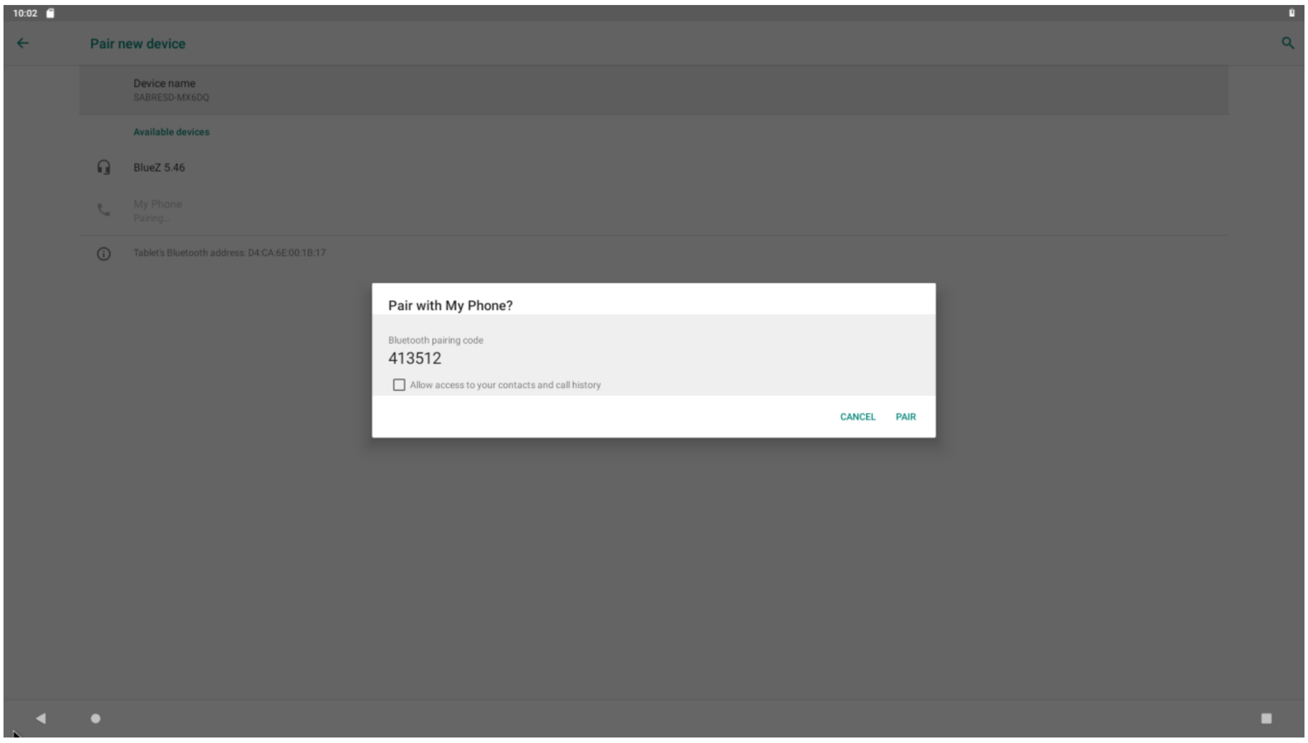


Figure 12: Bluetooth connect

☞ Bluetooth is not brought up with Vendor HAL 004 on SDIO-UART interface. Some changes were made to enable the functionality while using SDIO-SDIO interface and they are working but, only with SDIO interface.

Appendix

A Glossary

Abbreviation	Definition
ASCII	American Standard Code for Information Interchange
ARM	Arm (Advanced RISC Machines) Holdings
AEC	Automotive electronics council
BBR	Battery backed RAM
BER	Bit error rate
CPU	Central processing unit
UTC	Coordinated Universal Time
DTE	Data terminal equipment
DC	Direct current
DRX	Discontinuous reception
DDC	Display data channel
DL	Down link (Reception)
BT	Bluetooth
Wi-Fi	Wireless Fidelity

Table 4: Explanation of the abbreviations and terms used

Related documents

[1] EMMY-W1 series system integration manual, doc. no. UBX-15024929
[2] EVK-EMMY-W1 user guide, doc. no. UBX-15012713
[3] EMMY-W1 series data sheet, doc. no. UBX-15011785
[4] Cross reference platform for Android code browsing: http://aosp.opersys.com/xref/android-9.0.0_r47/
[5] Documentation of internal regulatory domain usage in kernel: https://wireless.wiki.kernel.org/en/developers/regulatory/wireless-regdb
[6] Android support forum for open source code, building and any other queries: https://source.android.com/setup
[7] Implementing SELinux: https://source.android.com/security/selinux/implement
[8] Android Pie integration guide from Marvell: MV-S302867-00B_Android P Vendor HAL porting
[9] Makefile architecture and reference: https://developer.android.com/ndk/guides/android_mk
[10] System architecture of Wi-Fi for Android: https://source.android.com/devices/tech/connect/wifi-overview

☞ For regular updates to u-blox documentation and to receive product change notifications, register on our homepage (www.u-blox.com).

Revision history

Revision	Date	Name	Comments
R01	30-Oct-2019	aheg	Initial release.

Contact

For complete contact information, visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180
E-mail: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640
E-mail: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185
E-mail: support@u-blox.com

Headquarters
Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44
E-mail: info@u-blox.com
Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811
E-mail: info_ap@u-blox.com
Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 2 8448 2016
E-mail: info_anz@u-blox.com
Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083
E-mail: info_cn@u-blox.com
Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 405 092 00
E-mail: info_in@u-blox.com
Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850
E-mail: info_jp@u-blox.com
Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861
E-mail: info_kr@u-blox.com
Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090
E-mail: info_tw@u-blox.com
Support: support_tw@u-blox.com