WE866Cx Wi-Fi and Bluetooth Network Interface Card (NIC) User Guide

1VV0301545 Rev. 10 – 2020-06-08

TELIT
TECHNICAL
DOCUMENTATION

SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT NOTICE

NOTICE

While reasonable efforts have been made to assure the accuracy of this document, Telit assumes no liability resulting from any inaccuracies or omissions in this document, or from use of the information obtained herein. The information in this document has been carefully checked and is believed to be reliable. However, no responsibility is assumed for inaccuracies or omissions. Telit reserves the right to make changes to any products described herein and reserves the right to revise this document and to make changes from time to time in content hereof with no obligation to notify any person of revisions or changes. Telit does not assume any liability arising out of the application or use of any product, software, or circuit described herein; neither does it convey license under its patent rights or the rights of others.

It is possible that this publication may contain references to, or information about Telit products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Telit intends to announce such Telit products, programming, or services in your country.

COPYRIGHTS

This instruction manual and the Telit products described in this instruction manual may be, include or describe copyrighted Telit material, such as computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and its licensors certain exclusive rights for copyrighted material, including the exclusive right to copy, reproduce in any form, distribute and make derivative works of the copyrighted material. Accordingly, any copyrighted material of Telit and its licensors contained herein or in the Telit products described in this instruction manual may not be copied, reproduced, distributed, merged or modified in any manner without the express written permission of Telit. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit, as arises by operation of law in the sale of a product.

COMPUTER SOFTWARE COPYRIGHTS

The Telit and 3rd Party supplied Software (SW) products described in this instruction manual may include copyrighted Telit and other 3rd Party supplied computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and other 3rd Party supplied SW certain exclusive rights for copyrighted computer programs, including the exclusive right to copy or reproduce in any form the copyrighted computer program. Accordingly, any copyrighted Telit or other 3rd Party supplied SW computer programs contained in the Telit products described in this instruction manual may not be copied (reverse engineered) or reproduced in any manner without the express written permission of Telit or the 3rd Party SW supplier. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit or other 3rd Party supplied SW, except for the normal non-exclusive, royalty free license to use that arises by operation of law in the sale of a product.

USAGE AND DISCLOSURE RESTRICTIONS

I. License Agreements

The software described in this document is the property of Telit and its licensors. It is furnished by express license agreement only and may be used only in accordance with the terms of such an agreement.

II. Copyrighted Materials

Software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law. No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without prior written permission of Telit

III. High Risk Materials

Components, units, or third-party products used in the product described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (High Risk Activities"). Telit and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

IV. Trademarks

TELIT and the Stylized T Logo are registered in Trademark Office. All other product or service names are the property of their respective owners.

V. Third Party Rights

The software may include Third Party Right software. In this case you agree to comply with all terms and conditions imposed on you in respect of such separate software. In addition to Third Party Terms, the disclaimer of warranty and limitation of liability provisions in this License shall apply to the Third Party Right software.

TELIT HEREBY DISCLAIMS ANY AND ALL WARRANTIES EXPRESS OR IMPLIED FROM ANY THIRD PARTIES REGARDING ANY SEPARATE FILES, ANY THIRD PARTY MATERIALS INCLUDED IN THE SOFTWARE, ANY THIRD PARTY MATERIALS FROM WHICH THE SOFTWARE IS DERIVED (COLLECTIVELY “OTHER CODE”), AND THE USE OF ANY OR ALL THE OTHER CODE IN CONNECTION WITH THE SOFTWARE, INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF SATISFACTORY QUALITY OR FITNESS FOR A PARTICULAR PURPOSE.

NO THIRD PARTY LICENSORS OF OTHER CODE SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND WHETHER MADE UNDER CONTRACT, TORT OR OTHER LEGAL THEORY, ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE OTHER CODE OR THE EXERCISE OF ANY RIGHTS GRANTED UNDER EITHER OR BOTH THIS LICENSE AND THE LEGAL TERMS APPLICABLE TO ANY SEPARATE FILES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

APPLICABILITY TABLE

PRODUCTS

- WE866C3
- WE866C6

CONTENTS

FIGURE LIST

1. INTRODUCTION

1.1. Scope

This user guide provides information required to install and evaluate Telit WE866Cx Wi-Fi and Bluetooth NIC card for Linux hosts. It also provides guidelines to prepare the host platform and start testing the module using the Linux supplicant applications.

1.2. Audience

This document is intended for Telit customers, who are integrators and about to implement their applications using Telit Wi-Fi and Bluetooth NIC module.

1.3. Contact Information and Support

For general contact, technical support services, technical questions and report documentation errors, contact Telit Technical Support at:

- TS-SRD@telit.com

Alternatively, use:

https://www.telit.com/contact-us

For detailed information about where you can buy Telit modules or for recommendations on accessories and components visit:

https://www.telit.com

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.4. Text Convention

Danger – This information MUST be followed or catastrophic equipment failure or bodily injury may occur.

Caution or Warning – Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.

Tip or Information – Provides advice and suggestions that may be useful when integrating the module.

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

1.5. Related Documents

Please refer to https://www.telit.com/m2m-iot-products/wifi-bluetooth-modules/ for current documentation and downloads.

2. OVERVIEW

This chapter describes the components and procedures for building a wireless application with a Linux host and Telit WE866Cx Wi-Fi and Bluetooth NIC module.

The Telit WE866Cx module provides IEEE802.11a/b/g/n/ac Wireless LAN and Bluetooth functionalities. It integrates complete MAC, PHY, and RF functionality on a single chip providing a low cost and an easy-to-use solution for adding wireless connectivity to applications.

WE866Cx modules integrate the required components like crystals, regulators, RF front-end components to provide ready-to-use WLAN radio modules that can be used as NIC cards to provide network connectivity to hosts. These modules are regulated, certified and calibrated for easy integration and building applications instantly without requiring any work on radio connectivity modules.

WE866Cx module provides an SDIO port to interface the WLAN module to a variety of hosts. The SDIO interface provides Ultra High Speed (SDR104) interconnection for faster communication with the host systems that are based on processors with 32-bit CPU and MMU which run on Linux operating systems. The Linux host runs the WLAN drivers, the network stack, the supplicant and authenticator 802.11 security applications to establish an 802.11 based Wireless LAN network using the WE866Cx module.

Standard Linux applications such as “wpa_supplicant” and “hostapd” are used for control path communications and standard Linux data path is used for data communications, and no custom software required.

WE866Cx module also provides a UART port to interface the Bluetooth module to the host. The UART interface is connected to the standard “tty” interface in the host.

The WE866Cx Bluetooth module provides the controller functionalities. The Bluetooth Stack and applications should be run in the Host. The Bluetooth software provides custom Bluetooth applications to use the Bluetooth module.

2.1. Hardware Components

The following diagram illustrates the hardware components:

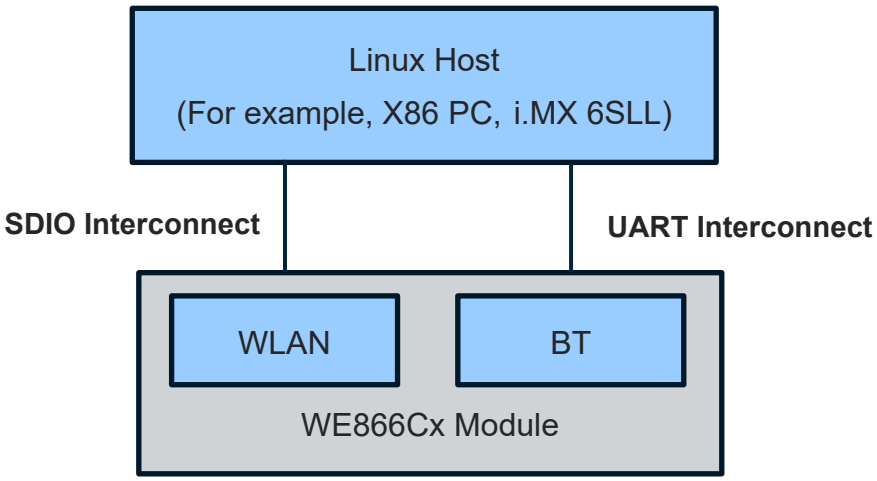

Figure :- Hardware Components

The description to the hardware components are as follows:

2.1.1. Linux Host

The Linux host can be any processor system that can run standard Linux software. It can be a high end X86 PC or an embedded platform like i.MX 6SLL. The host processor should be a 32-bit system running a 32-bit Linux operating system.

NOTE:

Currently, WE866Cx driver is supported in 32-bit Linux environment only.

2.1.2. WE866Cx Modules

WE866Cx modules acts as NIC card, providing IEEE802.11 a/b/g/n/ac MAC functionalities.

2.1.3. SDIO Interconnect

The SDIO interconnect provides connectivity between the host and WE866Cx module processor.

2.1.4. UART Interconnect

The UART pins are exposed in the FPC extension connector. Host applications can use the HCI layer to communicate with “tty” UART interface. A UART-to-USB converter can be used to interface the BT UART with the host having a USB port.

2.2. Software Components

The following diagram illustrates the software components being used. Telit Wi-Fi NIC host software is provided along with the software package –required to be installed in a directory and built, for details refer to 4Building WLAN and Bluetooth Software.

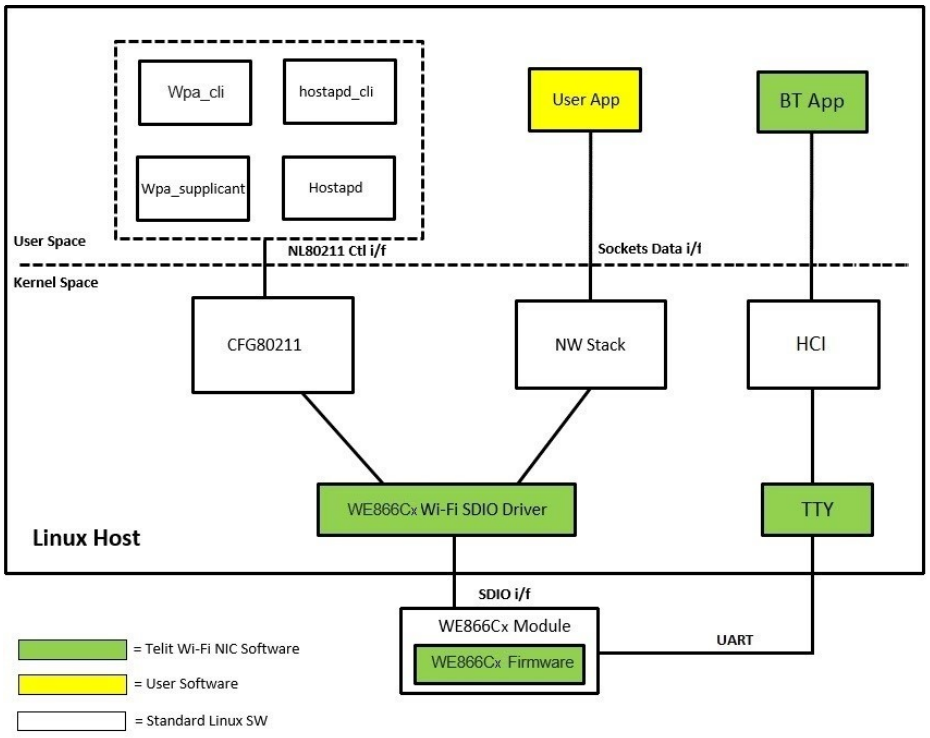

Figure 2-1 Software Components

2.2.1. Linux Application

Standard Linux applications are used for network connection setup and management. Following are the standard Linux application:

2.2.1.1. WPA Supplicant

“wpa_supplicant” is a WPA supplicant for Linux, BSD, Mac OS X, and Windows with WPA and WPA2 (IEEE 802.11i / RSN) support. It is suitable for both desktop/laptop computers and embedded systems. The supplicant is IEEE 802.1X/WPA component that is used in the client stations. It implements key negotiation with a WPA Authenticator, controls the roaming and IEEE 802.11 authentication/association of WLAN driver.

2.2.1.2. WPA CLI

“wpa_cli” is a text-based front-end program interacting with “wpa_supplicant”. It is used to get the current status, change configuration, trigger events, and request interactive user input.

2.2.1.3. Hostapd

“hostapd” is a user-space daemon for access point and authentication servers. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators, RADIUS client, EAP server, and RADIUS authentication server.

2.2.1.4. Hostapd CLI

“hostapd_cli” utility is a text-based frontend program for interacting with hostapd.

2.2.1.5. UserApp

“UserApp” makes use of networking services to setup socket connections and perform data transfer.

2.2.1.6. BT App

WE866Cx Bluetooth (BT) software provides a custom application to test the Bluetooth functionalities.

2.2.2. WE866Cx Wi-Fi and Bluetooth Software Package

2.2.2.1. WE866Cx Wi-Fi Driver

WE866Cx Wi-Fi driver software package provided by Telit, is a kernel module which implements Wi-Fi driver for interfacing the Linux kernel network control and data path to the WE866Cx Wi-Fi device. It is implemented as an IEEE802.11 Soft-MAC driver to establish communication between the Linux kernel and WE866Cx device. It contains SDIO driver adaptation layer for communication between Linux kernel and WE866Cx device over SDIO interconnect. It also performs queuing and flow control.

2.2.2.2. WE866Cx Firmware

WE866Cx firmware binaries are provided as part of the driver software package. These binaries must be placed at a certain location in the Linux host machine file system. These binaries will be accessed by the WE866Cx device to implement the IEEE 802.11 MAC functionality. It handles scan, association, and data transfer functionality and implements other MAC features, PHY, and RF functionalities.

2.2.2.3. Bluetooth Software

WE866Cx Bluetooth software package comprises of the Fluoride stack and some example applications provided to test the various profiles supported in the package. The WE866Cx Bluetooth module uses the UART interface to communicate with the host. WE866Cx Bluetooth software provides the binaries and configuration files to be used along with the applications to interface the Bluetooth module to the host system. The procedure to build

the Bluetooth software stack and applications and to run the applications are provided in the sections below.

Telit Wi-Fi and Bluetooth NIC software package “WE866Cx.tar.gz” consists of Telit Wi-Fi NIC software components and few tools.

Following are the software package contents:

1. apps – This folder contains the supported application software files
2. build – This folder contains scripts and make files to compile the driver software
3. drivers – This folder contains driver source code files and make files
   a. patches – This folder contains required patch files to be used in the Linux host system
   b. firmware – This folder contains binary images needed to run the WE866Cx device
4. bsp – This folder contains the files that support the bringing up of the platform
   a. i.mx6 – This folder contains patch file for i.MX6 (ARM) platform.

3. REFERENCE PLATFORM

This chapter describes the reference platform used to demonstrate the solution, the setup and applications used.

3.1. X86 PC Host Platform

“Lenovo ThinkPad T410i” laptop is used as the X86 PC host for testing the WE866Cx WLAN module. This host machine is booted with Ubuntu 16.04, 32-bit OS and the Linux kernel version 4.9.11.

3.2. Arm Cortex A9 Embedded Platform

WE866Cx module is tested with Arm architecture based on i.MX 6 series embedded host platforms. The products based on the i.MX 6 application processors enable cost effectiveness, rapid development of multimedia applications for Android and Linux operating systems.

3.2.1. MCIMX6SLL-EVK

MCIMX6SLL-EVK board is an embedded host platform based on the i.MX 6SLL processor. i.MX 6SLL application processor is a single Arm® Cortex®-A9, which operates up to a speed of 1GHz. This board has an SD card slot supporting SDIO 3.0 mode to plug WE866Cx target board for testing. Products based on the i.MX 6SLL application processors enables cost effective, rapid development of multimedia applications for Android® and Linux® operating systems.

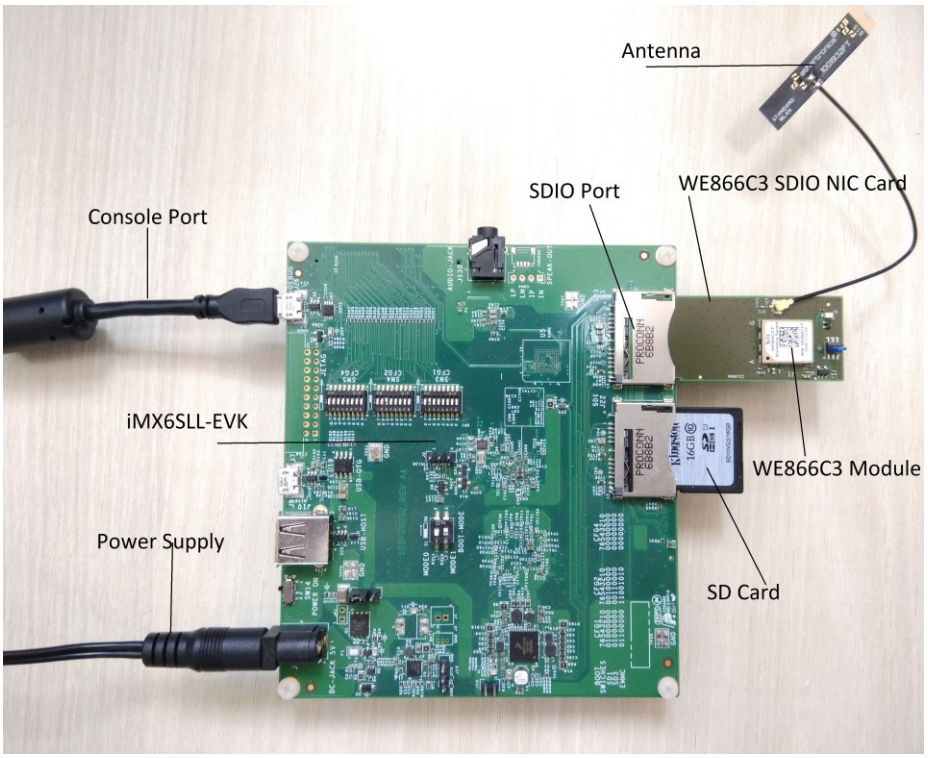

Figure 3-1 i.MX 6SLL EVK Host WLAN Setup

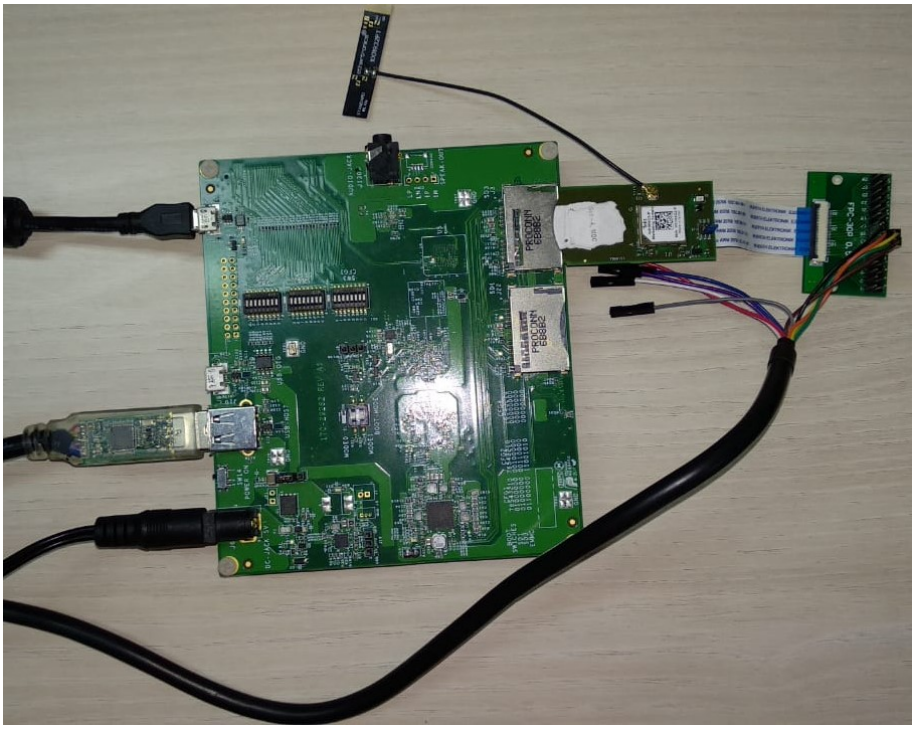

Figure 3-2 i.MX 6SL-EVK Host WLAN and Bluetooth Setup

3.2.2. MCIMX6Q-SDB

MCIMX6Q-SDB is an embedded host board based on the i.MX 6Quad core processor, which belongs to i.MX 6 family of Application processors. i.MX 6Quad processor has four Arm® Cortex®-A9 CPUs, clocking at a speed of 1GHz and using 1GB DDR3 SRAM. This board has an SD card slot capable of supporting SDIO 3.0 mode to plug WE866Cx target board for testing.

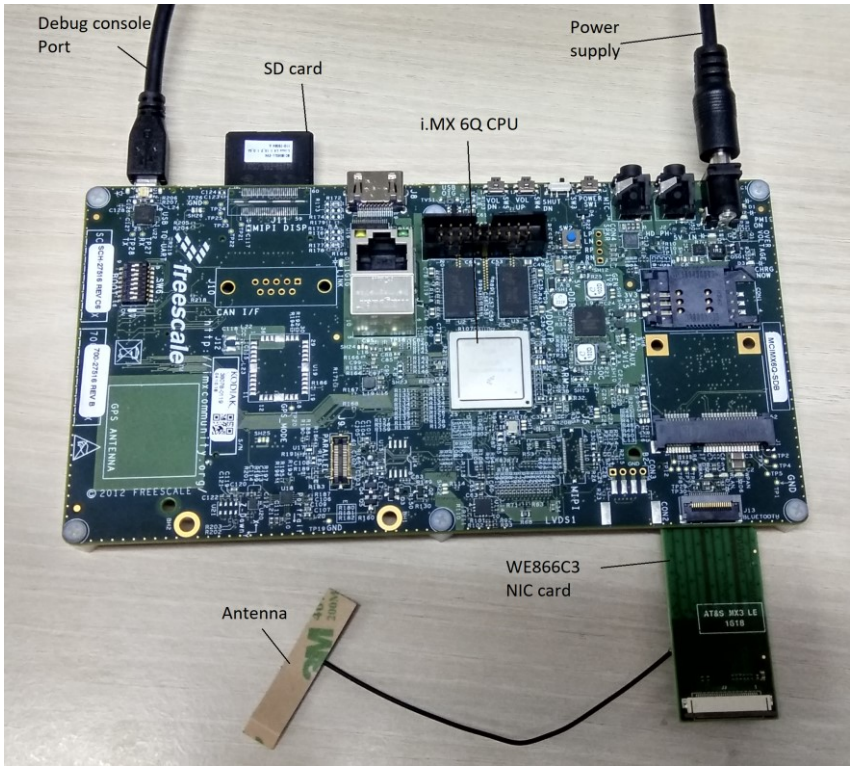

Figure 3-3 MCIMX6Q-SDB Board

3.3. WE866Cx Module NIC Card

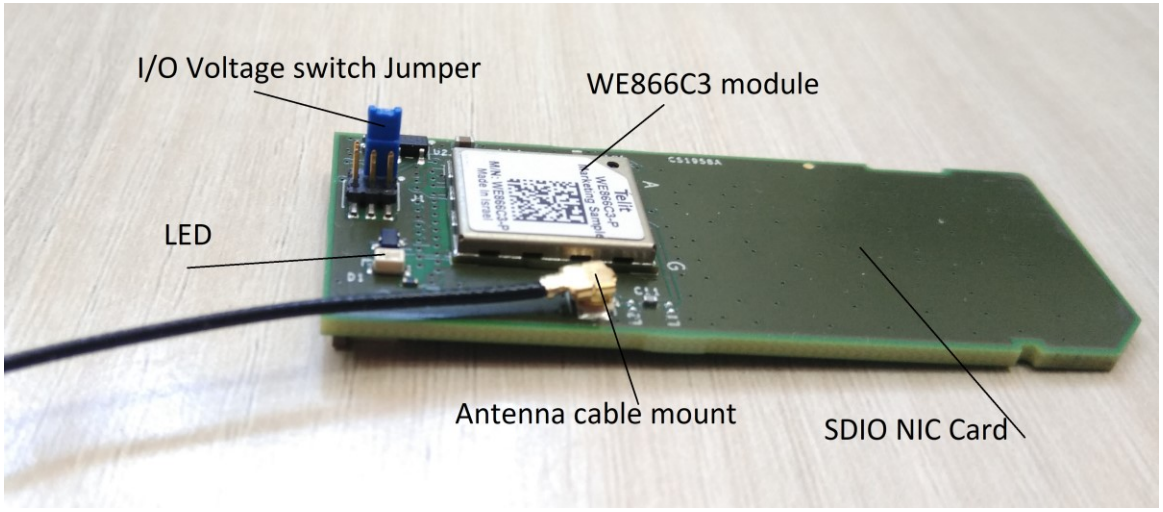

Figure 3-4 WE866Cx Board

The WE866Cx NIC card has an LED indicator and an antenna mount point to plug in the connector cables from the external antennas. It also has a jumper to select between 1.8v and 3.3v SDIO line signal operations, depending on the host platform with SDIO controller capabilities.

If the host SDIO controller is based on SD 2.0 protocol, then it supports only 3.3v I/O operation. So, the jumper should be put into 3.3v position.

If the host SDIO controller is based on SD 3.0 protocol, then it supports 1.8v I/O operation. The jumper should be put in the 1.8v position.

1.8v and 3.3v jumper positions are given in the following figure:

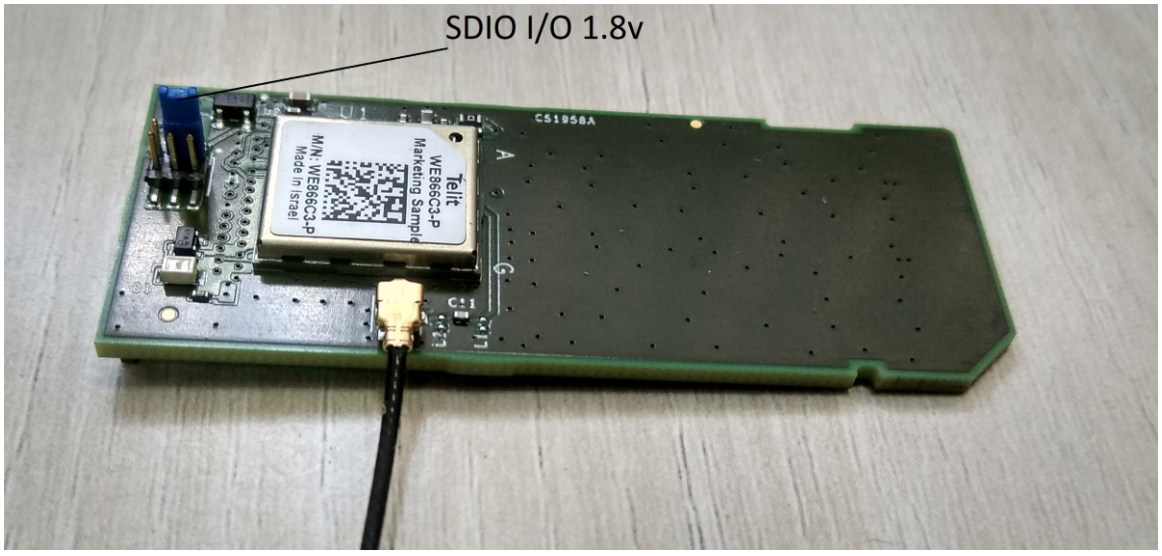

Figure 3-5 Jumper Position for 1.8v SDIO Signal

Figure 3-6 Jumper Position for 3.3v SDIO Signal

The following figure shows the FPC extension connector which provides UART pins to connect to the Host UART interface. Pins 11-15 provide CTS, RTS, RXD, TXD, and GND respectively.

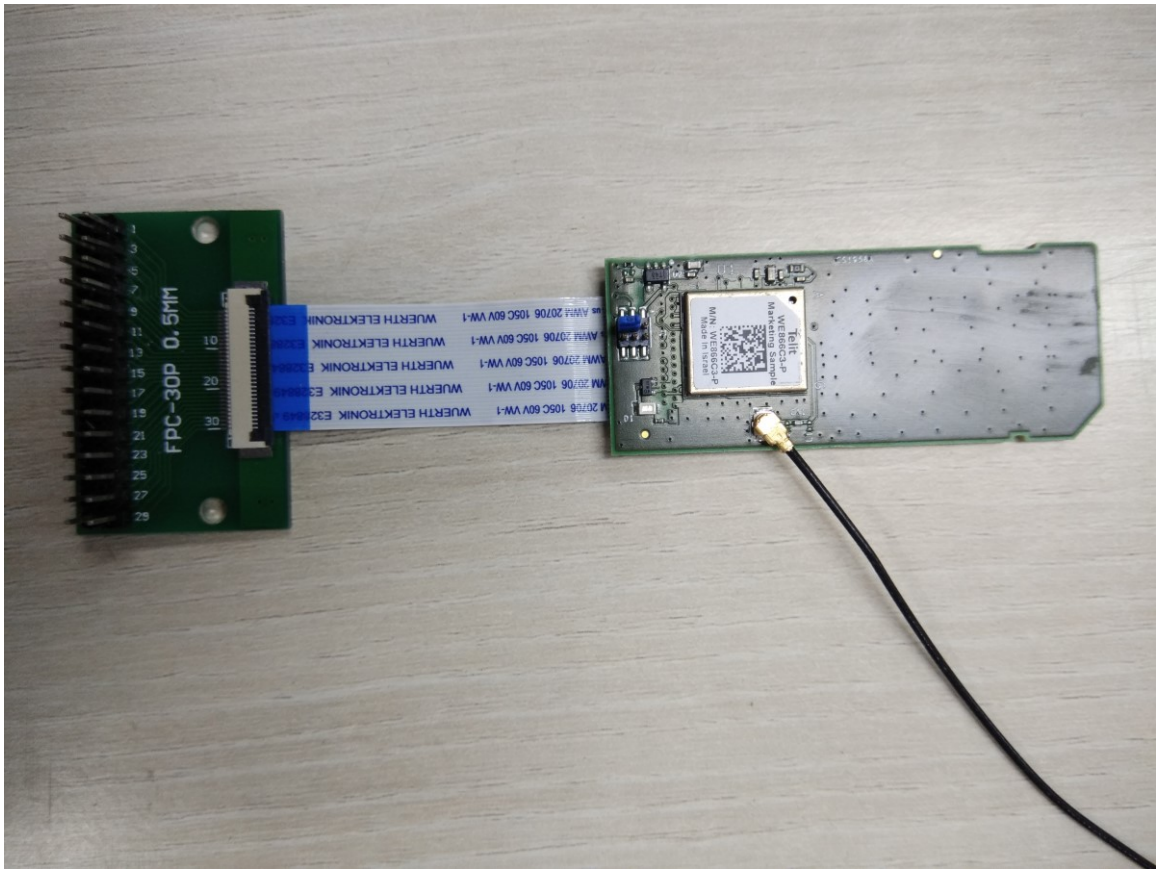

Figure 3-7 WE866Cx Bluetooth Extension Connector

4. BUILDING WLAN AND BLUETOOTH SOFTWARE

This chapter provides the steps for building WE866Cx Linux drivers, loading and running them on a reference platform.

4.1. X86 PC Host Platform

The X86 host machine is loaded with Linux kernel v4.9.11 patched with the kernel patches provided in the software package of WE866Cx driver. Now, X86 host platform is ready to build the driver software. For details refer to Appendix A: Downloading, Building, and Installing Linux Kernel on x86 Ubuntu

4.2. ARM Cortex A9 Embedded Platform

i.MX 6SLL-EVK board must be booted with Linux kernel v4.9.11 built with the required patches. The driver project should be cross compiled for the i.MX 6SLL_EVK platform. For details refer to Appendix B: Building and Installing Linux Kernel and Driver on NXP i.MX 6 Embedded Host

4.3. WE866Cx Wi-Fi and Bluetooth Software Package

Telit WE866Cx Wi-Fi driver software package consists of folders such as apps, build, drivers etc. WE866Cx driver is supported with the Linux kernel version 4.9.11. The Linux kernel 4.9.11 should be patched with the kernel patch files given in the driver software package.

Following steps details how to build procedure for X86 platform:

1. Download the WE866Cx software package into your project directory.

   For example, in X86 host, if "$HOME" is the current project directory.

   ```
   $ cp WE866Cx.tar.gz $HOME
   ```

2. Extract the software package contents to your project directory using the following command.

   ```
   $ cd $HOME
   $ tar -xvf WE866Cx.tar.gz
   ```

4.3.1. Building WE866Cx Wi-Fi Driver

1. To build the driver, go to the build directory in the WE866Cx project.

   ```
   $ cd ~/WE866Cx/build
   ```

2. To clean and build the driver, run the make command with clean option.

   ```
   $ make clean
   ```

3. Run the make command to build the driver files.

   ```
   $ make
   ```

Once WLAN driver code is built, the WLAN driver module (wlan.ko), firmware binaries and configuration files will be available in the “WE866Cx/rootfs-we866cx.build/” directory.

Note: The procedure to build WLAN software for the i.MX 6 platform is explained under B.4 Building WE866Cx Wi-Fi Linux Application for i.MX 6 on EVK Platform

4.3.2. Loading WE866Cx Wi-Fi Driver

1. Copy the WLAN firmware binaries to the host machine filesystem.

   ```
   $ sudo cp  ~/WE866Cx/rootfs-we866cx.build/lib/firmware/WLAN-firmware/*
   /lib/firmware/
   ```

2. Copy the configuration files to the host machine filesystem

```
$ sudo mkdir -p /lib/firmware/wlan/
$ sudo cp  ~/WE866Cx/rootfs-we866cx.build/lib/firmware/wlan/*
/lib/firmware/wlan/
```

3. Before placing the driver object (*wlan.ko*) file, plug the WE866Cx Wi-Fi card in SDIO slot of the host machine.
4. Insert the WE866Cx kernel object (.ko) file into the Linux kernel. WE866Cx driver will load the board data firmware, based on the input parameter "use_cust_board_data".
   - For WE866C3 module,

     ```
     $ sudo insmod rootfs-we866cx.build/lib/modules/wlan.ko
     ```

   - For WE866C6 module,

     ```
     $ sudo insmod rootfs-we866cx.build/lib/modules/wlan.ko
     use_cust_board_data=2
     ```

   Note: For the input argument "use_cust_board_data" with any value other than 2, the default board data firmware corresponding to WE866C3 module will be download.
5. If the WLAN driver is inserted into the kernel, the WE866Cx Wi-Fi NIC card will be listed as a wireless device with the names **wlan0** & **p2p0**, in the network interface list.

```
$ ifconfig -a
```

4.3.3. Building WE866Cx Bluetooth Software

To build the Bluetooth software components, certain kernel headers and library files must be pre-installed. For x86 platform, the kernel headers can be installed directly into the filesystem. Similarly, for x86 systems, the libraries could also be immediately installed.

For i.MX6 platform, the library files required to build the BT files are provided with the release package. For i.MX6 systems, the kernel header files must be copied to the cross-compiler included path, to be used while compiling the Bluetooth applications.

Note: The procedure to build Bluetooth software for the i.MX6 platform is explained under B.5 Building Bluetooth Application for i.MX6 Host.

To compile the Bluetooth application, the kernel headers must be installed. Execute the following commands for Bluetooth Fluoride stack compilation. This step must be executed after the system is rebooted with the new kernel.

```
$ cd <kernel_path>/linux-stable
$ sudo make headers_install ARCH=i386 INSTALL_HDR_PATH=/usr
```

NOTE:

Before building the Bluetooth software, make sure that the required libraries as listed in A.1 Prerequisites are pre-installed.

Next, run the build script files corresponding to the host machine architecture. The script will build the Bluetooth fluoride stack, and other application components.

1. Enter the build directory.

```
$ cd ~/WE866Cx/build
```

2. Export the board type as given in the file (env.makefile)

```
$ export BOARD_TYPE=linux
$ chmod 777 ./scripts/${BOARD_TYPE}/bt_x86.sh
$ sudo ./scripts/${BOARD_TYPE}/bt_x86.sh
```

Once the Bluetooth software is built, the application binaries will be generated in their respective directories. The Bluetooth software configuration files and supporting library files will be available in the *WE866Cx/rootfs-we866cx.build/*directory.

4.3.4. Loading WE866Cx Bluetooth Software

4.3.4.1. Bluetooth HCI UART Interface

No special driver is required for the UART interface. It uses a standard "**tty**" interface in the host. If using a UART-to-USB converter device, after connecting the cable, run *dmesg*, and ensure that the device is detected as **"/dev/ttyUSB0"**.

Note: If the UART-to-USB device does not enumerate as **/dev/ttyUSB0**, then reboot the host once and make sure the interface is detected as **/dev/ttyUSB0**.

4.3.4.2. Bluetooth Firmware

To load the Bluetooth firmware, copy the firmware patch files (nvm and rampatch files) to host filesystem location.

```
$ sudo cp ~/WE866Cx/rootfs-we866cx.build/lib/firmware/BT-firmware/*
/lib/firmware/ar3k/
```

4.3.4.3. Bluetooth Library Files

1. Copy the Bluetooth configuration files to the linux host filesystem

   ```
   $ sudo cp ~/WE866Cx/rootfs-we866cx.build/etc/bluetooth/*  /etc/bluetooth/
   ```

   Note: Before copying the btapp configuration files, make sure to remove the previous meta files located at /etc/Bluetooth.

2. Copy the Bluetooth output binaries to the Linux host

   ```
   $ sudo cp ~/WE866Cx/rootfs-we866cx.build/usr/bin/*  /usr/bin/
   ```

3. Copy the Bluetooth output libraries to the Linux host

   ```
   $ sudo cp ~/WE866Cx/rootfs-we866cx.build/usr/lib/*  /usr/lib/
   ```

5. WIRELESS NETWORK OPERATIONS

This chapter provides step by step procedure to setup wireless network connectivity and perform data transfer.

Following are the steps to connect to the wireless network and test data transfer, once the WLAN interface is configured:

NOTE:

Configuration instructions are only for reference and contain typical example. For more details refer – “Linux documentation”.

5.1. WE866Cx WLAN Interface

If the driver is loaded correctly, WE866Cx device will be listed in the wireless interfaces present in the host system. WE866Cx module will register two wireless interfaces wlan0 & p2p0. This is verified using the following command:

```
$ ifconfig -a
```

The following command is used to find the physical device corresponding to WE866Cx WLAN interface:

```
$ iw dev
```

The following command is used to check the features and commands supported by all devices:

```
$ iw phy
```

5.2. STA Mode

Telit WE866Cx module supports 802.11 b/g/n and a/ac modes in STA operations. STA mode supports connecting to both 2.4 GHz & 5GHz BSS networks.

Following are the steps required to configure and test wireless interface in station mode:

1. Configure DHCP
   a. Open configuration file and make sure DHCP is enabled.
      ```
      vi /etc/network/interfaces
      ```
   b. File should contain below two lines:
      ```
      auto wlan0
      iface wlan0 inet dhcp
      ```
2. Restart networking
   ```
   /etc/init.d/networking restart
   ```
3. Create or update the configuration file. In STA mode, configuration file is common between the 802.11b/g/n or 802.11a/ac networks.

   Typical configuration file contains the following references:

   a. Open Mode (For example, sta_open_con.conf file)
      ```
      ctrl_interface=/run/wpa_supplicant
      update_config=1
      network={
      ssid="MY_AP_OPEN"
      ```

```
scan_ssid=1
key_mgmt=NONE
}
```

b. AES security (For example, sta_aes_con.conf file)

```
ctrl_interface=/run/wpa_supplicant
update_config=1
network={
ssid="MY_AP_AES"
scan_ssid=1
key_mgmt=WPA-PSK
psk="TelitDemo123"
proto=RSN
pairwise=CCMP
group=CCMP
}
```

4. Terminate wpa_supplicant (if its running)

```
$ sudo killall wpa_supplicant
$ sudo rfkill unblock wifi
```

5. Start wpa_supplicant and initiate connection (use the required conf file)

   For example:

```
$ sudo wpa_supplicant -B -i wlan0 nl80211 -c sta_open_con.conf
$ sudo wpa_supplicant -B -i wlan0 nl80211 -c sta_aes_con.conf
```

6. Get IP address by running dhclient

```
$ sudo dhclient wlan0
```

7. To check the IP address and status:

```
$ ifconfig wlan0
```

8. If needed disable power save feature

```
$ sudo iw dev wlan0 set power_save off
$ sudo iw dev wlan0 get power_save
```

9. If required start wpa_cli application

```
$ sudo wpa_cli -i wlan0
```

10. Test data transfer using ping

```
$ ping <AP IP address>
```

5.3. AP Mode

Telit WE866Cx module supports 802.11 b/g/n and a/ac modes in AP operations. AP interface supports starting a BSS in 2.4 GHz and 5GHz network.

Following are the steps required to install, configure, and test wireless interface in AP mode:

1. Install the DHCP server.

   ```
   $ sudo apt-get install isc-dhcp-server
   ```

2. Configure DHCP server
   a. Open configuration file and make sure static configuration

      ```
      vi /etc/network/interfaces
      ```

   b. File content:

      ```
      auto wlan0
      iface wlan0 inet static
      address 10.0.0.1
      netmask 255.255.255.0
      gateway 10.0.0.1
      ```

   c. Restart networking

      ```
      /etc/init.d/networking restart
      ```

3. Open dhcp configuration file and modify if required

   ```
   vi /etc/dhcp/dhcpd.conf
   ```

   Sample configuration given below:

   ```
   ddns-update-style none;
   ignore client-updates;
   authoritative;
   option routers 10.0.0.1;
   option subnet-mask 255.255.255.0;
   option broadcast-address 10.0.0.255;
   option domain-name-servers 10.0.0.1,8.8.8.8,8.8.4.4;
   option time-offset 0;
   default-lease-time 1209600;
   max-lease-time 1814400;
   subnet 10.0.0.0 netmask 255.255.255.0 {
   range 10.0.0.3 10.0.0.13;
   }
   ```

4. Select the “wlan0” interface to start DHCP server. Open the file “/etc/default/isc-dhcp-server” and add “wlan0” to the interfaces list.

   INTERFACES="wlan0"

5. Start DHCP Server in wlan0 interface

   ```
   $ sudo service isc-dhcp-server start
   ```

6. Create or update the hostapd configuration file

Typical configuration file contains the following references:

a. 802.11 b/g/n mode AP with Open security (hostapd_open.conf)

```
interface=wlan0
driver=nl80211
ssid=Test_AP_OPEN
channel=11
hw_mode=g
ieee80211n=1
```

b. 802.11 b/g/n mode AP with AES security (hostapd_aes.conf)

```
interface=wlan0
driver=nl80211
ssid=Test_AP_AES
channel=11
hw_mode=g
ieee80211n=1
wpa=2
wpa_passphrase=TelitDemo123
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

c. 802.11 a/ac VHT=80 mode AP with AES security (hostapd_11ac_aes.conf)

```
interface=wlan0
driver=nl80211
ssid=Test_AP_AES
channel=161
hw_mode=a
ieee80211n=1
ieee80211ac=1
vht_capab=[HTC-VHT80]
vht_oper_chwidth=0
vht_oper_centr_freq_seg0_idx=161
wpa=2
wpa_passphrase=TelitDemo123
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

Note: In 802.11 a/ac mode the VHT parameters should be adjusted according to the requirement.

7. Terminate wpa_supplicant (if running)

```
$ sudo killall wpa_supplicant
$ sudo rfkill unblock wifi
```

8. Start hostapd (use the required conf file). For example:

```
$ sudo hostpd -B -dd hostapd_open.conf
$ sudo hostpd -B -dd hostapd_aes.conf
```

9. Start CLI application to check the status.

```
$ sudo hostapd_cli
```

10. Connect a client to this AP and test data transfer using ping.

5.4. Concurrent Mode Operation

5.4.1. STA - AP Mode Concurrency

This feature is used to enable the STA and AP interfaces operating concurrently in the same channel.

1. Load the driver into the kernel and insert the WE866Cx Wi-Fi module into SDIO slot.
2. Setup a virtual interface for AP mode of operation.

```
$ iw dev wlan0 interface add ap0 type __ap
```

3. Start STA mode in WLAN0 interface and connect to the required AP.

```
$ sudo wpa_supplicant -B -i wlan0 -D nl80211 -c sta_aes_con.conf
```

4. Run dhclient application to get the IP address for the STA interface.

```
$ sudo dhlcient wlan0
```

5. Assign static address to the newly created AP interface ap0 and start DHCP server as shown in the section 5.3 AP Mode.
6. Before starting the AP interface, start the DHCP server using isc-dhcp-server application. Refer section 5.3 AP Mode.
7. AP interface can be started either using the wpa_supplicant or hostapd application.
   a. To start the AP interface using the hostapd application issue the command.

```
$ sudo hostapd -B -dd hostapd_open.conf
```

   Note: The channel mentioned in hostapd config file should be same as the channel in which STA operates.
   b. To start the AP interface using the WPA supplicant application, suitable configuration file is needed.

   For example, concurrent_ap.conf

```
ctrl_interface=/run/wpa_supplicant
update_config=1
network={
ssid="Concurrent_AP"
mode=2
key_mgmt=NONE
}
```

   Note: It is not required to mention the channel in AP configuration file. It follows the channel used by the STA.

```
$ sudo wpa_supplicant -B -i ap0 -D nl80211 -c concurrent_ap.conf
```

8. STA and AP interface can be started using a single wpa_supplicant

```
$ sudo wpa_supplicant -B -i ap0 -D nl80211 -c concurrent_ap.conf -N
-i wlan0 -D nl80211 -c sta_aes_con.conf
```

9. Run the DHCP server and assign IP to STA interface and test data transfer for both STA and AP interfaces.

5.4.2. STA - P2P Mode Concurrency

STA & P2P mode concurrent operation can be tested using the wpa_supplicant application.

1. Start STA mode in wlan0 interface and connect to the required AP. Refer section 5.2 STA Mode.
2. Start P2P functionality, in the p2p0 interface using the wpa_supplicant application. P2P mode supports both GO and client functionalities. Refer section 5.5 P2P Mode for the steps to configure the p2p0 interface.

5.5. P2P Mode

WE866Cx driver supports the P2P mode of operation. WE866Cx device registers a p2p0 physical interface which is used for P2P mode. P2P mode can be started as either P2P client or P2P Group Owner (GO) functionality. P2P mode can be tested using the wpa_supplicant application. Both P2P GO and P2P client functionalities can be achieved using the same configuration file.

Typical P2P configuration file will contain the following settings:

p2p_mode.conf

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
ap_scan=1
device_name=WE866Cx-P2P-Linux
device_type=1-0050F204-1
config_methods=display keypad push_button
p2p_oper_reg_class=81
p2p_oper_channel=1
p2p_listen_reg_class=81
p2p_listen_channel=1
p2p_go_intent=15
persistent_reconnect=1
```

5.5.1. P2P Client Mode

In P2P client mode, the WE866Cx device is connected to a P2P device capable of running as GO. It can connect to an autonomous P2P GO device or it can connect to a normal P2P device, which will become GO during the connection process.

1. For client mode, write the GO intent to a lesser value in the config file.

   ```
   p2p_go_intent=1
   ```

2. Start the wpa_supplicant for the p2p interface using the above configuration file.

   ```
   $ sudo wpa_supplicant -B -D nl80211 -i p2p0 -c p2p_mode.conf
   ```

3. Start the wpa_cli to issue the p2p commands

   ```
   $ sudo wpa_cli -i p2p0
   ```

4. Scan the GO MAC address to connect

   ```
   p2p_find
   ```

5. When the target GO details are listed in the find operation, stop find

   ```
   p2p_stop_find
   ```

6. The MAC address of all the devices found will be listed using the command

   ```
   p2p_peers
   ```

7. Connect the GO MAC address using the PIN method

   ```
   p2p_connect <GO MAC addr> pin go_intent=1
   ```

   This command generates the “PIN” number to be typed in the keypad of the GO device.

8. When the P2P connection is successful, then the command for the status of the device provides the new BSS

   ```
   status
   ```

9. Get the IP address for the P2P client interface

   ```
   $ sudo dhclient p2p0
   ```

   Once the IP address is received, check the data transfer with GO using the Ping test.

5.5.2. P2P GO Mode

WE866Cx module can start P2P GO mode in two ways:

- The module can start as a GO autonomously and a client can connect to it.
- During connection with another P2P device, WE866Cx can be assigned as P2P GO.

1. For P2P GO mode, write the GO intent value to a maximum value in config file

   ```
   p2p_go_intent=15
   ```

2. Before starting the p2p0 interface in GO mode, assign static IP address for the p2p0 interface and start the isc-dhcp-server for the p2p interface. Refer section -5.3 AP Mode.
3. Start the wpa_supplicant for the p2p interface using the above configuration file

   ```
   $ sudo wpa_supplicant -B -D nl80211 -i p2p0 -c p2p_mode.conf
   ```

4. Start the wpa_cli to issue the p2p commands

   ```
   $ sudo wpa_cli -i p2p0
   ```

5. Scan the GO MAC address to connect and issue command for time for scan completion

   ```
   p2p_find 20
   ```

6. The MAC address of all the devices found will be listed using the command

   ```
   p2p_peers
   ```

7. To start a persistent group

   ```
   p2p_group_add persistent freq=2 ht40
   ```

   freq=2 means 2.4 GHz network.

freq=5 means 5GHz network.

8. If the autonomous GO is started successfully, then the P2P GO device will be in connected state. To check the status,

```
status
```

9. Scan from the p2p client device and issue the connect request to GO MAC address. In GO device issue the connect command to the client device address along with the PIN generated by the client address

```
p2p_connect <Client MAC address>  "pin_value"  go_intent=15
```

10. Run the dhclient in the P2P client device and check the data transfer between P2P GO and client devices.

5.6. Throughput Measurement

iPerf is a tool used to measure the maximum TCP/UDP throughput. For peak performance, tests are conducted in RF chamber or in a less crowded environment.

To test the uplink data transfer throughput, the target device should be running iperf client service to transmit data to the iperf server. To test the downlink data transfer throughput, the target device should be running iperf server and another device in the network to should run the iperf client service to transmit data.

For peak performance, tests are conducted in RF chamber or in a less crowded environment.

5.6.1. Throughput Commands

5.6.1.1. UDP server

```
iperf  -u -s -p <port_no> -w10M
```

5.6.1.2. UDP Client

```
iperf -u -c <server_IP> -p <port_no> -t60 -b1000M -w10M -Px
```

5.6.1.3. TCP Server

```
iperf  -s -p <port_no> -w10M
```

5.6.1.4. TCP Client

```
iperf -c <server_IP> -p <port_no> -t60 -w10M -Px
```

Here,

- b is the UDP bandwidth used for the Tx test. It should be a minimum of 450 for VHT-80 mode tests.
- w is the UDP buffer size or TCP window size used for Tx & Rx tests. It should be a minimum of 2MB for VHT-80 mode tests.
- P is the parallel threads used for Tx. It can be increased from 1 to ‘n’ as per the host system requirement.

5.6.2. WE866Cx Throughput Test Requirements

To conduct throughput tests with a WE866Cx module in 802.11ac VHT 80MHz channel mode, the embedded host platforms must be capable to drive the high bandwidth of data via network stacks. Hence, some of the kernel parameters must be adjusted to support the high throughput tests.

5.6.2.1. Configuring Network Stack Parameters

Increase the memory used by the network stack, to run high throughput tests in 802.11ac mode VHT-80MHz. Kernel network memory parameters can be adjusted in run time to test the maximum bandwidth supported by the host platform.

Increase the maximum memory size to support the window size used by the tests.

```
$ sudo echo 'net.core.rmem_max=8388608' >> /etc/sysctl.conf
$ sudo echo 'net.core.wmem_max=8388608' >> /etc/sysctl.conf
$ sudo sysctl -p
```

5.6.2.2. Enabling Single Core Operation in CPU

In some hosts with multiple cores, when all the cores are running the throughput achieved is lesser than the maximum capacity of the host. To estimate the maximum throughput supported by the host, some cores can be turned off before running the tests.

Enable the single core operation in the ARM architecture based embedded CPUs like IMX6Q CPU to avoid the context switching between the throughput test and to estimate the maximum throughput.

```
$ sudo -s
$ echo 0 > /sys/devices/system/cpu/cpu1/online
$ echo 0 > /sys/devices/system/cpu/cpu2/online
$ echo 0 > /sys/devices/system/cpu/cpu3/online
$ mpstat -P ALL
```

5.7. Configuring DFS Master

Configure DFS Master feature to enable or disable WLAN Access Point mode operation in IEEE802.11 DFS channels.

To configure DFS Master mode functionality, edit “/lib/firmware/wlan/qcom_cfg.ini” WLAN configuration file.

- To enable DFS Master functionality for Access Point mode, set the variable

  `gEnableDFSMasterCap=1`

- To disable DFS Master functionality and stop using the DFS channels, reset the variable

  `gEnableDFSMasterCap=0`

Note: By default, the DFS master is enabled**.** The configuration file contains the variable `gEnableDFSMasterCap=1` in the released package.

Note: Configuration file must be modified, before starting WLAN interface using “insmod” command.

6. ENABLING BLUETOOTH FEATURES

This section describes the step by step procedure to enable and test the WE866Cx Bluetooth features.

NOTE:

The user cannot run multiple instances of the btapp. The user can run only one instance.

The user cannot use the kill -9 <btapp pid>. Instead, the user can use kill -2 <btapp

pid>, or CTRL+C.

6.1. Testing Bluetooth

Once the kernel boots up, perform the following steps to verify the Bluetooth functionality.

1. Open two terminals. In the first terminal run the following commands.

   ```
   $ cd ~/WE866Cx/apps/bt_workspace/qcom-opensource/bt/property-ops
   $ sudo ./btproperty
   ```

2. On the other terminal, run the following commands.

   ```
   $ cd ~/WE866Cx/apps/bt_workspace/qcom-opensource/bt/bt-app
   $ sudo ./main/btapp
   ```

By this time, stack and other libraries are loaded.

Press **Enter** and go to **gap_menu > enable**. This enables the Bluetooth, and the following message is displayed.

6.2. Renaming the Bluetooth Device

Rename the Bluetooth device name as per your desire, using the following command

```
$ sudo vi /etc/bluetooth/bt_app.conf
```

6.3. A2DP Sink

1. Go to **gap_menu > enable**.
2. Connect to the DUT using a remote device (For example, from a mobile phone).

3. If the passphrase that appears on the screen matches the passphrase on the mobile phone, type **Yes** and press **Enter**.

   Note: If the passphrase that appears on the screen does not match the passphrase on the mobile phone, type **NO** and press **Enter**, return to **Step 2**.

The on-screen message `A2DP Sink Connected to <remote_bd_address>` indicates a successful connection.

Make sure to use the start, stop, forward, backward, next, previous, volume change, and disconnect functions from the remote device only.

NOTE:

1. Either A2DP sink or A2DP source is supported at any time.
2. Only one device is supported at any time (For example, max a2dp connection is 1)
3. Currently, only the SBC codec is supported.
4. Ensure the following settings in **/etc/bluetooth/bt_app.conf**

   BtA2dpSourceEnable=false

   BtA2dpSinkEnable=true

   BtAvrcpEnable=true

6.4. A2DP Source

1. Go to **gap_menu > enable**.
2. Connect to the DUT using a remote device (For example, from a mobile phone).
3. If the passphrase that appears on the screen matches the passphrase on the mobile phone, type **Yes** and press **Enter**.

   Note: If the passphrase that appears on the screen does not match the passphrase on the mobile phone, type **NO** and press **Enter**, return to **Step 2**.

   The on-screen message `Pairing state for <Device name> is BONDED` indicates a successful connection.

   Or,

1. Go to **gap_menu > enable**.
2. Pair `<space><bt_address of the sink device>` (For example, pair `00:11:22:33:44:55`).
3. If the passphrase that appears on the screen matches the passphrase on the sink device, type **Yes** and press **Enter**.

   Note: If the passphrase that appears on the screen does not match the passphrase on the sink device, type **NO**, press Enter, and return to **Step 2**.

4. Go to **main_menu > a2dp_source_menu**.
5. Perform a profile-level connection by using command `connect<space><bt_address>` (For example, connect `00:11:22:33:44:55`).

   The on-screen message `A2DP Source Connected to <remote_bd_address>` indicates a successful connection.

NOTE:

1. Currently, the player is not integrated into the A2DP source. Only one file can be played. Before starting the source, ensure that the file named **pcm.wav** is copied to the **/usr/local folder**. This file should be 16-bit PCM data and 44.1 kHz. It can also be generated using tool like audacity.
2. Either A2DP sink or A2DP source is supported at any time.
3. Only one device is supported at any time (For example, max a2dp connection is 1).
4. Currently, only the SBC codec is supported.
5. a2dp_source_menu currently supports connect, disconnect, start, stop, suspend, volume up, and volume down.
6. Ensure the following settings in ***/etc/bluetooth/bt_app.conf***:

   BtA2dpSourceEnable=true

   BtA2dpSinkEnable=false

   BtAvrcpEnable=false

6.5. HID

1. Go to **gap_menu > enable**.
2. Start **inquiry**, **cancel_inquiry**, list through **inquiry_list**.
3. Pair **<bd_add>** from **gap_menu**.
4. The passkey shown on the screen must be entered on the keyboard (not a mouse).
5. View this device in **bonded_list**, at this point.
6. Go to **main_menu > hid_menu**.
7. Connect **<bd_addr>** for profile-level HID connection.

Or,

1. Go to **gap_menu > enable**.
2. Start **inquiry**, **cancel_inquiry**, list through **inquiry_list**.
3. Go to **main_menu > hid_menu**.
4. Connect **<bd_addr>** for profile-level HID connection.
5. The passkey shown on the screen must be entered on the keyboard (not a mouse).
6. After connecting, this can be seen in **hid_list** from **gap_menu**.

6.6. HOGP

1. Go to gap_menu > enable.
2. Start **inquiry**, **cancel_inquiry**, list through **inquiry_list**.
3. Go to **main_menu** > **hid_menu**.
4. Use connect **<bd_addr>** from **hid_menu**

The passkey shown on the screen must be entered on the keyboard (not a mouse).

Once the connection is established, this device will be seen in **hid_list** from **gap_menu**. dev

NOTE:

It cannot be paired from the pair of **gap_menu** because LE is over GATT.

To connect BLE devices it is recommended to make the following configuration settings in **/etc/bluetooth/bt_app.conf** before executing **btapp** and **btproperty**.

BtA2dpSourceEnable=false

BtA2dpSinkEnable=false

BtAvrcpEnable=false

BtSppServerEnable=false

BtSppClientEnable=false

BtHidEnable=true

6.7. SPP

Enable SPP (client and server) in **/etc/bluetooth/bt_app.conf** using the below settings:

BtSppServerEnable=true

BtSppClientEnable=true

6.7.1. SPP Server

6.7.1.1. Steps to receive a file

1. Go to **gap_menu > enable**.

   Switch ON Bluetooth from the mobile phone and ensure it is discoverable.

2. Go to **main_menu > spp_server_menu**.
3. Start (this starts the SPP server, and SDP registration happens only after this).
4. Use the *Bluetooth SPP Pro* application (the SPP client application from the mobile phone) to discover the DUT.
5. Initiate pairing from the *Bluetooth SPP Pro* application.
6. After pairing is complete, initiate a connection from the *Bluetooth SPP Pro* application.
7. Once the connection is established, you will see **Select communication** mode in the *Bluetooth SPP Pro* application.
8. On the DUT side, go to **main_menu > spp_server_menu.**
9. Use option **receive <filename>** to receive the file. For example, *Receive /home/administrator/test_sppserver_recv.txt*
10. From the *Bluetooth SPP Pro* application, select **byte stream** mode.
11. Send data from the *Bluetooth SPP Pro* application (use the > button).
12. Exit the *Bluetooth SPP Pro* application (disconnects the connection)
13. You can also disconnect from the DUT side using the disconnect option.
14. Go back to the **main_menu**.

Note: If the remote device (mobile phone) is already paired, initiate scan from *Bluetooth SPP Pro* application, select the DUT from the list of discovered devices. Use the **Connect** option to connect to DUT, then follow procedure from Step 7.

6.7.1.2. Steps to send a file

1. Go to **gap_menu > enable**.

   Switch ON Bluetooth from the mobile phone and ensure it is discoverable.

2. Start inquiry from the DUT.
3. Discover and initiate pairing from the remote device (DUT) using the *Bluetooth SPP Pro* application.
4. After pairing is complete, initiate a connection from the *Bluetooth SPP Pro* application.
5. After the connection is established, you see **Select communication mode** in the *Bluetooth SPP Pro* application.
6. From the *Bluetooth SPP Pro* application, select **Byte stream** mode.
7. On the DUT side, go to main_menu > spp_server_menu.
8. Use option **send <filename>** to send the file. For example, *Send /home/administrator/test_spps_send.txt*
9. Verify that the data is received in the Bluetooth SPP Pro application.
10. Exit the *Bluetooth SPP Pro* application (disconnects the connection).
11. You can also disconnect from the DUT side using the disconnect option.
12. Go back to the **main_menu**.
    Note: If the remote device (mobile phone) is already paired, initiate scan from *Bluetooth SPP Pro* application, select the DUT from the list of discovered devices. Use **Connect** option to connect to DUT, then follow procedure from Step 7.

6.7.2. SPP Client

6.7.2.1. Steps to receive a file

1. Go to **gap_menu > enable**.

   Switch ON Bluetooth on the mobile phone side and ensure it is discoverable.

2. Start **Inquiry** from DUT.
3. Discover and pair remote phone from DUT.
4. Start **BlueSPP** (SPP server application in the phone).
5. On DUT, Go to **main_menu > spp_client_menu**.
6. From DUT initiate connect “connect <BD Addr>”. For example, connect `bc:f5:ac:9c:0d:e6`

   Note: In case you have previously tested using the BlueSPP application, it will load the history (send & receive data) unless you disable the default option.

7. From DUT, use command **receive <file>**. For example, *receive /home/administrator/test_sppclient_recv.txt*
8. From phone “BlueSPP” send data to DUT.
9. Exit the “BlueSPP” application (disconnects the connection) - You can also disconnect from the DUT side using the “disconnect” option.
10. Go back to the **main_menu**

    Note: If the remote device (mobile phone) is already paired, follow procedure from Step 8.

6.7.2.2. Steps to send a file

1. Go to **gap_menu > enable**.
2. Switch ON Bluetooth on the phone side and ensure it is discoverable.
3. Start **inquiry** from DUT.
4. Discover and pair remote phone from DUT.
5. Start “BlueSPP” (SPP server application in the mobile phone).

6. On DUT, go to **main_menu > spp_client_menu**.
7. From DUT initiate connect “connect <BD Addr>”. For example, connect `bc:f5:ac:9c:0d:e6`

   Note: In case you have previously tested using the BlueSPP application, it will load the history (send & receive data) unless you disable the default option.
8. From DUT use command “send <file>”. For example, *send /home/administrator/test_sppclient_send.txt*
9. Verify the data in “BlueSPP” application in mobile phone.
10. Exit the “BlueSPP” connection. (You can also disconnect from the DUT side using the **disconnect** option).
11. Go back to the **main_menu**.

    Note: If the remote device (mobile phone) is already paired, follow procedure from Step 8.

6.8. RSP Menu

RSP Menu uses the underlying GATT protocol to test the BLE mode in the Host controller. It provides two options “rsp_init” and “rsp_start”, to test BLE GATT server mode of operation.

Before you begin, make sure to install the “nRF Connect” (Android/iOS) application in your remote device (mobile phone). This application will be used to evaluate WE866Cx BLE mode of data transfer.

Perform the following steps to test the BLE Mode:

1. Go to **gap_menu > enable**
2. Go to **main_menu > rsp_menu**
3. Type **rsp_init**, to initialize GATT server and start advertising the BLE frames.
   Note: This step is required to be done only once during initialization phase.
4. Switch ON Bluetooth from your test mobile phone and ensure it is discoverable.
5. Now, open **nRF Connect** application in your mobile phone and go to “*SCANNER*” menu and initiate a scan for BLE devices.
6. Search for DUT name **MDM_FLUORIDE** and connect.
7. DUT in BLE GATT server mode will provide a custom service which will contain a custom characteristic with write property enabled.
8. Start writing byte data to this custom characteristic from the mobile phone. This will initiate pairing between DUT and your mobile phone.
9. In DUT side, BTApp will prompt with “BT pairing request”. Type **“yes”** to continue paring process. In mobile phone, a BT pairing prompt will be received. Compare the random number generated and type “**yes”** on both mobile and DUT sides, to confirm pairing.
10. Once pairing is done, test the custom characteristic transmission by writing either byte data, byte array or string data, and so on.
11. If BLE link is disconnected, use **rsp_start** to re-advertise from DUT GATT server.
12. Reconnect from remote mobile phone and restart the custom characteristic writing test.
    In Mobile side, close the previous connection. Rescan and reconnect to DUT.

APPENDIX A: DOWNLOADING, BUILDING, AND INSTALLING LINUX KERNEL ON X86 UBUNTU

This section provides step by step procedure to setup the host platform and install the Linux kernel v4.9.11 required to build and use the WE866Cx Wi-Fi driver package. A host X86 machine, is required to be installed with Linux OS 32-bit with kernel version 4.9.11. And, the kernel v4.9.11 should be patched with the patch files provided in the driver package.

A.1 Prerequisites

Before building the kernel v4.9.11, the X86 host machine must be ready with the following packages installed:

```
$ sudo apt-get update
$ sudo apt-get install ncurses-dev
$ sudo apt-get install libssl-dev
$ sudo apt-get install libnl-3-dev
$ sudo apt-get install libnl-genl-3-dev
$ sudo apt-get install bison
$ sudo apt-get install flex
$ sudo apt-get install automake libtool dpkg-dev libasound2-dev
$ sudo apt-get install tar
$ sudo apt-get install git
```

A.2 Installing Linux Kernel v4.9.11

1. Before downloading the kernel, set the git credentials which will be used while patching the kernel

   ```
   $ git config --global user.name "name"
   $ git config --global user.email "name@mail.com"
   ```

2. Download the latest stable Linux kernel from the kernel tree

   ```
   $ git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
   ```

3. Enter the Linux kernel directory

   ```
   $ cd linux-stable
   ```

4. Get the copy of kernel version 4.9.11 into the Linux kernel directory

   ```
   $ git checkout v4.9.11
   ```

5. Apply the required patches to the downloaded kernel code

   ```
   $ git am ~/WE866Cx/drivers/patches/linux/v4.9.11/*.patch
   ```

   Note: Make sure to provide the correct path of WE866Cx directory to patch Linux kernel.

   Refer section 4.3 WE866Cx Wi-Fi and Bluetooth Software Package for steps to download the package for X86 platform.

6. Configure Linux Kernel by issuing the below command and enable the required settings

```
$ make menuconfig
```

NOTE:

By default, "make menuconfig" command reads the configuration file used by the current booted kernel and uses it to create the new configuration (.config) file.

User is requested to enable the platform specific drivers and configurations, that are needed for the host system functionalities.

Enable the following settings in the kernel:

CFG80211 – To enable Wireless configuration APIs

CONFIG_NL80211_TESTMODE=y – To support commands in test mode

MMC - To enable MMC/SD/SDIO card support

MMC_DEBUG – To enable MMC debugging (If debugging is required)

CFG80211_INTERNAL_REGDB – To use the statically compiled regulatory rule data base

If UART-to-USB converter is used to interface WE866Cx Bluetooth hardware with Linux host, then ensure that the kernel is built with supporting USB serial drivers.

CONFIG_USB_SERIAL=y

CONFIG_USB_SERIAL_GENERIC=y

CONFIG_USB_SERIAL_FTDI_SIO=y

7. Build and install the Linux kernel with the required modules

```
$ make -j8
$ make modules
$ sudo make modules_install
$ sudo make headers_install
$ sudo make install
```

8. After installation, reboot the host to boot with kernel v4.9.11. If the newly built kernel v4.9.11 is not the latest in the host machine (i.e., the host machine is already running with a kernel version higher than v4.9.11), then while rebooting hold the “SHIFT” key to enter inside GRUB and select the desired kernel v4.9.11. After rebooting, check the current kernel version using the below command.

```
$ uname -a
```

APPENDIX B: BUILDING AND INSTALLING LINUX KERNEL AND DRIVER ON NXP I.MX 6 EMBEDDED HOST

This section provides the steps to bring up the host machine with the kernel v4.9.11 and install the WE866Cx driver for the i.MX 6 platform.

B.1 Prerequisites

1. Laptop or a desktop running a recent release of Debian, Fedora or Ubuntu; without OS virtualization software.
2. ARM Cross Compiler

   Linaro toolchain binaries: http://www.linaro.org/downloads/
3. Bootloader

   Das U-Boot – the Universal Boot Loader: http://www.denx.de/wiki/U-Boot

   Source: http://git.denx.de/?p=u-boot.git;a=summary
4. Linux Kernel – imx_4.9.11_1.0.0_ga from Freescale
5. ARM based rootfs
   a. Debian: https://www.debian.org
   b. Ubuntu: http://www.ubuntu.com

B.2 Building the Kernel for i.MX 6 Platform

This section includes the steps to build the kernel for both the hosts - MCIMX6Q-SDB and MCIMX6SLL-EVK.

1. Define a project directory with the name of the host board under test. This will be used as the project directory to build the bootloader and kernel and the driver and so on.
   For MCIMX6SLL-EVK host board,

   ```
   $ export DIR=mcimx6sll-evk
   ```

   For MCIMX6Q-SDB host board,

   ```
   $ export DIR=mcimx6q-sdb
   ```

   Create a directory under the defined board name:

   ```
   $ mkdir ~/$DIR
   $ cd ~/$DIR
   ```
2. Create two directories which holds the final kernel images and other information:

   ```
   $ mkdir ~/$DIR/boot
   $ mkdir ~/$DIR/kernel_modules
   ```
3. Download the pre-built Linaro GCC compiler for cross compilation

   ```
   $ wget -c --no-check-certificate
   https://releases.linaro.org/components/toolchain/binaries/6.4-
   2017.11/arm-linux-gnueabihf/gcc-linaro-6.4.1-2017.11-i686_arm-
   linux-gnueabihf.tar.xz
   $ tar xf gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf.tar.xz
   ```
4. Download the ARM based rootfs

```
$ wget -c https://rcn-ee.com/rootfs/eewiki/minfs/ubuntu-16.04.3-
minimal-armhf-2017-12-09.tar.xz

$ tar xf ubuntu-16.04.3-minimal-armhf-2017-12-09.tar.xz
```

5. Download the Linux source code from Freescale server (Branch -imx_4.9.11_1.0.0_ga)

```
$ git clone -b imx_4.9.11_1.0.0_ga --single-branch
git://source.codeaurora.org/external/imx/linux-imx/
```

6. Download the u-boot - the Universal Boot Loader

```
$ wget -c ftp://ftp.denx.de/pub/u-boot/u-boot-2018.01.tar.bz2

$ tar xf u-boot-2018.01.tar.bz2
```

7. Apply the patch to the u-boot source code,

```
$ cd ~/$DIR/u-boot-2018.01
```

For MCIMX6SLL-EVK host board,

```
$ patch -p1 < ~/$DIR/WE866Cx/bsp/imx6/u-boot/2018.01/mx6sllevk-fixes.patch
```

For MCIMX6Q-SDB host board,

```
$ patch -p1 < ~/$DIR/WE866Cx/bsp/imx6/u-boot/2018.01/mx6sabresd-fixes.patch
```

8. Set the environment variable with following command:

```
$ export ARCH=arm

$ export CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-
linux-gnueabihf/bin/arm-linux-gnueabihf-
```

9. Configure and build the u-boot.

For MCIMX6SLL-EVK host board,

```
$ make mx6sllevk_defconfig ARCH=arm CROSS_COMPILE=~/$DIR/gcc-linaro-
6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

$ make -j4 ARCH=arm CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-
i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

For MCIMX6Q-SDB host board,

```
$ make mx6sabresd_defconfig ARCH=arm CROSS_COMPILE=~/$DIR/gcc-linaro-
6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-

$ make -j4 ARCH=arm CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-
i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

10. Navigate to the Linux kernel source directory. Apply the patches to the kernel to avoid kernel build errors.

```
$ cd ~/$DIR/linux-imx/

$ patch -p1 < ~/$DIR/WE866Cx/bsp/imx6/linux-imx/v4.9.11/wl_cfg80211.patch
```

11. Apply the kernel patches to enable the SDIO 3.0 mode of operation in the SD slot of the host platform.

```
$ patch -p1 < ~/$DIR/WE866Cx/bsp/imx6/linux-imx/v4.9.11/usdhcx_vselect.patch
```

For MCIMX6SLL-EVK host board,

```
$ patch -p1 < ~/$DIR/WE866Cx/bsp/imx6/linux-imx/v4.9.11/mx6sllevk-dtb.patch
```

For MCIMX6Q-SDB host board,

```
$ patch -p1 < ~/$DIR/WE866Cx/bsp/imx6/linux-imx/v4.9.11/mx6sabresd-
dtb.patch
```

12. Apply the patches given along with the WE866Cx driver package to the Linux kernel.

```
$ git am ~/$DIR/WE866Cx/drivers/patches/linux/v4.9.11/*.patch
```

Note: Make sure to provide the correct path of WE866Cx directory to patch Linux kernel.

13. Apply the kernel default configuration, suitable for i.MX platform

```
$ export ARCH=arm
$ export CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
$ make imx_v7_defconfig ARCH=arm  CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Note: Export environment variables, before invoking “make” to cross-compling kernel for ARM platform

14. Modify the kernel configuration for any required changes

```
$ make menuconfig ARCH=arm  CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Enable the following settings in the kernel:

CFG80211 – To enable Wireless configuration APIs

CONFIG_NL80211_TESTMODE=y – To support commands in test mode

MMC - To enable MMC/SD/SDIO card support

MMC_DEBUG – To enable MMC debugging (If debugging is required)

CFG80211_INTERNAL_REGDB – To use the statically compiled regulatory rule data base

Device drivers-> Network device Support-> Wireless LAN-> Intersil devices-> IEEE 802.11 for Host AP

If a UART-to-USB converter is used to interface WE866Cx Bluetooth hardware with host, then ensure the kernel is built with supporting USB serial drivers.

CONFIG_USB_SERIAL=y

CONFIG_USB_SERIAL_GENERIC=y

CONFIG_USB_SERIAL_FTDI_SIO=y

15. Install “lzop” before building the kernel

```
$ sudo apt-get install lzop
```

16. Build the kernel image. This step may take long time

```
$ make bzImage ARCH=arm  CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

17. Copy the kernel image to the boot directory.

```
$ cp arch/arm/boot/zImage  ~/$DIR/boot/
```

18. Build the device tree binaries for the host.

```
$ make dtbs ARCH=arm  CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Copy the corresponding dtb files to the boot directory

For MCIMX6SLL-EVK host board,

```
$ cp arch/arm/boot/dts/imx6sll-evk.dtb ~/$DIR/boot/
```

For MCIMX6Q-SDB host board,

```
$ cp arch/arm/boot/dts/imx6q-sabresd.dtb  ~/$DIR/boot/
```

19. Build the Linux kernel modules and copy it to the “kernel_modules” directory

```
$ make modules ARCH=arm  CROSS_COMPILE=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
$ make modules_install ARCH=arm INSTALL_MOD_PATH=~/$DIR/kernel_modules
```

20. Install the kernel headers

```
$ mkdir -p ~/$DIR/kernel_modules/usr
$ make headers_install ARCH=arm INSTALL_HDR_PATH=~/$DIR/kernel_modules/usr
```

B.3 Setting up the SD Card

Insert the SD card into the laptop

1. Run the “lsblk” command to get the SD card mount name.
   For example, if “mmcblk0” is the mount name used for the card.

```
$ export DISK=/dev/mmcblk0
```

2. To erase the partition table/labels on microSD card, execute the following command:

```
$ sudo dd if=/dev/zero of=${DISK} bs=1M count=50
```

   Now, remove and re-insert the MMC card.

3. Install the bootloader

   For MCIMX6SLL-EVK host board,

```
$ sudo dd if=~/$DIR/u-boot-2018.01/u-boot-dtb.imx of=${DISK} seek=2 bs=512
```

   For MCIMX6Q-SDB host board,

```
$ sudo dd if=~/$DIR/u-boot-2018.01/SPL of=${DISK} seek=1 bs=1k
$ sudo dd if=~/$DIR/u-boot-2018.01/u-boot.img of=${DISK} seek=69 bs=1k
```

4. Create Partition Layout

   util-linux v2.26, sfdisk is rewritten and is based on libfdisk.

```
$ sudo sfdisk -version
If sfdisk >= 2.26.x
$ sudo sfdisk ${DISK} <<-__EOF__
1M,,L,*
__EOF__
If sfdisk <= 2.25.x
$ sudo sfdisk -UNIT M ${DISK} <<-__EOF__
1,,L,*
__EOF__
```

5. Format Partition

```
$ sudo mkfs.ext4 -V
```

```
for: DISK=/dev/mmcblk0
$ sudo mkfs.ext4 -L rootfs ${DISK}p1
for: DISK=/dev/sdX
$ sudo mkfs.ext4 -L rootfs ${DISK}1
```

6. Mount Partition

   Most systems have the partitions auto mounted.

```
$ sudo mkdir -p /media/rootfs/
for: DISK=/dev/mmcblk0
$ sudo mount ${DISK}p1 /media/rootfs/
for: DISK=/dev/sdX
$ sudo mount ${DISK}1 /media/rootfs/
```

7. Copy the root file system

```
$ sudo tar xfvp  ~/$DIR/ubuntu-16.04.3-minimal-armhf-2017-12-
09/armhf-rootfs-ubuntu-xenial.tar  -C  /media/rootfs
$ sudo chown root:root /media/rootfs/
$ sudo chmod 755 /media/rootfs/
```

8. Create an environmental variable with the kernel version

```
$ export KERNEL_VERSION=4.9.11
```

9. Set *uname_r* in */boot/uEnv.txt*

```
$ sudo sh -c "echo 'uname_r=${KERNEL_VERSION}'  >>
/media/rootfs/boot/uEnv.txt"
```

10. Copy the kernel image

```
$ sudo cp -v ~/$DIR/boot/zImage  /media/rootfs/boot/vmlinuz-
${KERNEL_VERSION}
```

11. Copy the device tree binaries

```
$ sudo mkdir -p /media/rootfs/boot/dtbs/${KERNEL_VERSION}/
```

    For MCIMX6SLL-EVK host board,

```
$ sudo cp -v ~/$DIR/boot/imx6sll-evk.dtb
/media/rootfs/boot/dtbs/${KERNEL_VERSION}/
```

    For MCIMX6Q-SDB host board,

```
$ sudo cp -v ~/$DIR/boot/imx6q-
sabresd.dtb/media/rootfs/boot/dtbs/${KERNEL_VERSION}/
```

12. Copy the kernel modules

```
$ sudo cp -rv ~/$DIR/kernel_modules/lib/  /media/rootfs/
```

13. Create static information about filesystem

```
$ sudo sh -c "echo '/dev/mmcblk2p1 / auto errors=remount-ro 0 1' >>
/media/rootfs/etc/fstab"
```

14. Unmount the SD card

```
$ sync
$ sudo umount  /media/rootfs
```

B.4 Building WE866Cx Wi-Fi Linux Application for i.MX 6 on EVK Platform

Consider the path "*~/$DIR/WE866Cx/* " is the location, where the WLAN driver code is cross-compiled for the i.MX 6 platform.

1. Download the WE866Cx software package into your project directory.

   ```
   $ cp WE866Cx.tar.gz $DIR
   ```

2. Extract the software package contents to your project directory using the following command.

   ```
   $ cd $DIR
   $ tar -xvf WE866Cx.tar.gz
   ```

3. Get the board type from the file "**env.makefile**" and export it as a variable

   ```
   $ export BOARD_TYPE=linux
   ```

4. Open the build configuration file and edit the kernel variables to include the cross-compiler path

   ```
   $ sudo vi  ~/$DIR/WE866Cx/build/scripts/${BOARD_TYPE}/config.${BOARD_TYPE}
   ```

   a. Assign the kernel path variable to the location where the Linux kernel v4.9.11 is cross-compiled for iMX6SLL-EVK board

      ```
      export KERNELPATH=~/$DIR/linux-imx/
      ```

   b. Assign the kernel architecture variable with ARM

      ```
      export KERNELARCH=arm
      ```

   c. Assign the tool prefix variable with the path name, in which "Linaro" cross-compile tool chain is located.

      ```
      export TOOLPREFIX=~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
      ```

5. Edit the "**Makefile**" in the folder "*WE866Cx/driver/qcacld-new/*", to assign the kernel source variable with proper path to the kernel source code.

   ```
   $ KERNEL_SRC ?= ~/$DIR/linux-imx/
   ```

6. To build the driver, go to the build directory in the WE866Cx project.

   ```
   $ cd ~/WE866Cx/build
   ```

7. To clean and build the driver, run the make command with clean option.

   ```
   $ make clean
   ```

8. Run the make command to build the driver files.

   ```
   $ make
   ```

Once the WLAN driver code is built, the WLAN driver module (*wlan.ko*), firmware binaries and driver configuration files will be available in the "~/$DIR/*WE866Cx/rootfs-we866cx.build/*" directory. These firmware binaries and configuration files must be copied to the Linux files system mounted on the i.MX 6 host SD card.

9. Insert the i.MX 6 SD card in the host laptop. Consider "/media/rootfs" is the host system directory, where i.MX6 SD card files system is mounted.
10. Copy the WLAN binaries to the location "*/lib/firmware"* in the file system mounted on the SD card.

```
$ sudo cp -rf   ~/$DIR/WE866Cx/rootfs-we866cx.build/lib/firmware/WLAN-
firmware/*    /media/rootfs/lib/firmware/
```

Note: Use proper directory path, where SD card is mounted in your host

11. Copy the WLAN configuration files to the location "*/lib/firmware/wlan/*" in the SD card file system. Create a folder named "wlan" in the target location, if there is no folder.

```
$ sudo mkdir /media/rootfs/lib/firmware/wlan/
```

```
$ sudo cp -rf  ~/$DIR/WE866Cx/rootfs-we866cx.build/lib/firmware/wlan/*
/media/rootfs/lib/firmware/wlan/
```

12. Copy the cross compiled driver folder to the SD card.

```
$ sudo cp -rf ~/$DIR/WE866Cx/ /media/rootfs/home/ubuntu/
```

13. Unmount the SD card and re-insert into i.MX6 target platform. Restart i.MX6 platform.

```
$ cd /home/ubuntu/WE866Cx/
```

14. Insert the WE866Cx kernel object (.ko) file into the Linux kernel.
    For WE866C3 module,

```
$ sudo insmod rootfs-we866cx.build/lib/modules/wlan.ko
```

For WE866C6 module,

```
$ sudo insmod rootfs-we866cx.build/lib/modules/wlan.ko
use_cust_board_data=2
```

B.5 Building Bluetooth Application for i.MX6 Host

This section includes the steps required to build the WE866Cx Bluetooth software for i.MX 6 host platform.

1. Install the kernel headers required to build the Bluetooth applications.

```
$  mkdir -p ~/$DIR/kernel_modules/usr
```

```
$ cd ~/$DIR/linux-imx/
```

```
$  make headers_install ARCH=arm INSTALL_HDR_PATH=~/$DIR/kernel_modules/usr
```

2. Copy the kernel header files to the cross-compiler toolchain path.

```
$  cp -rf  ~/$DIR/kernel_modules/usr/include/*   ~/$DIR/gcc-linaro-
6.4.1-2017.11-i686_arm-linux-gnueabihf/arm-linux-
gnueabihf/libc/usr/include/
```

3. Copy the kernel library headers from the release package to the cross-compiler path.

```
$  cp -rf  ~/$DIR/WE866Cx/bsp/imx6/BT/alsa/usr/include/alsa/
~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/arm-linux-
gnueabihf/libc/usr/include/
```

```
$  cp -rf  ~/$DIR/WE866Cx/bsp/imx6/BT/zlib/usr/include/*
~/$DIR/gcc-linaro-6.4.1-2017.11-i686_arm-linux-gnueabihf/arm-linux-
gnueabihf/libc/usr/include/
```

4. Copy the libraries to the cross-compiler toolchain libraries path

```
$  cp  ~/$DIR/WE866Cx/bsp/imx6/BT/alsa/usr/lib/*   ~/$DIR/gcc-linaro-
6.4.1-2017.11-i686_arm-linux-gnueabihf/arm-linux-
gnueabihf/libc/usr/lib/
```

```
$  cp  ~/$DIR/WE866Cx/bsp/imx6/BT/zlib/usr/lib/*   ~/$DIR/gcc-linaro-
6.4.1-2017.11-i686_arm-linux-gnueabihf/arm-linux-
gnueabihf/libc/usr/lib/
```

5. Edit the script file **bt_armhf.sh**, to add the arm cross-compiler toolchain path.

   a. Enter the WE866Cx project build directory

   ```
   $ cd ~/$DIR/WE866Cx/build/
   ```

   b. Export the board type variable as given in the file **env.makefile**

   ```
   $ export BOARD_TYPE=linux
   ```

   c. Open the script file **bt_armhf.sh**

   ```
   $ vi  scripts/${BOARD_TYPE}/bt_armhf.sh
   ```

   d. Add the tool-chain path

   ```
   $ export TARGETSYSROOT=~/$DIR/gcc-linaro-6.4.1-2017.11-
   i686_arm-linux-gnueabihf
   ```

   e. Save and close the file.

6. Run the script file **bt_armhf.sh** to cross-compile the BT applications.

   ```
   $ chmod 777 ./scripts/${BOARD_TYPE}/bt_armhf.sh
   ```

   ```
   $ sudo ./scripts/${BOARD_TYPE}/bt_armhf.sh
   ```

   The build script will cross-compile the Fluoride stack and hardware libraries and all the required applications. Output files generated during the build will be available in the *~/$DIR/WE866Cx/rootfs-we866cx.build/* directory.

7. Insert i.MX6 SD card in the host laptop. Consider “/media/rootfs” is the host system directory, where i.MX6 SD card files system is mounted.

8. After building the Bluetooth application software, copy the project directory WE866Cx to the i.MX 6 host filesystem mounted on the SD card.

   ```
   $ sudo cp -rf ~/$DIR/WE866Cx/ /media/rootfs/home/ubuntu/
   ```

   Note: Use proper directory path, where SD card is mounted in your host

   a. Copy the Bluetooth firmware patch files to the i.MX 6 host filesystem mounted on the SD card

   ```
   $ sudo cp ~/$DIR/WE866Cx/rootfs-we866cx.build/lib/firmware/BT-
   firmware/*   /media/rootfs/lib/firmware/ar3k/
   ```

   b. Copy the Bluetooth configuration files to the i.MX6 host filesystem mounted on the SD card

   ```
   $ sudo cp ~/$DIR/WE866Cx/rootfs-we866cx.build/etc/bluetooth/*
   /media/rootfs/etc/bluetooth/
   ```

   Note: Before copying the btapp configuration files, make sure to remove the previous meta files located at /media/rootfs/etc/bluetooth.

   c. Copy the Bluetooth output binaries to the host machine root filesystem

   ```
   $ sudo cp ~/$DIR/WE866Cx/rootfs-we866cx.build/usr/bin/*
   /media/rootfs/usr/bin/
   ```

   d. Copy the Bluetooth output libraries to the host machine root filesystem

   ```
   $ sudo cp ~/$DIR/WE866Cx/rootfs-we866cx.build/usr/lib/*
   /media/rootfs/usr/lib/
   ```

   e. Copy the supporting libraries used to build Bluetooth applications to the SD card.

```
$ sudo cp -rf ~/$DIR/WE866Cx/bsp/imx6/BT/alsa/usr/bin/*
/media/rootfs/usr/bin/

$ sudo cp -rf ~/$DIR/WE866Cx/bsp/imx6/BT/alsa/usr/include/alsa
/media/rootfs/usr/include/

$ sudo cp -rf ~/$DIR/WE866Cx/bsp/imx6/BT/alsa/usr/lib/*
/media/rootfs/usr/lib/

$ sudo cp -rf ~/$DIR/WE866Cx/bsp/imx6/BT/alsa/usr/share/*
/media/rootfs/usr/share/

$ sudo cp -rf ~/$DIR/WE866Cx/bsp/imx6/BT/zlib/usr/include/*
/media/rootfs/usr/include/

$ sudo cp -rf ~/$DIR/WE866Cx/bsp/imx6/BT/zlib/usr/lib/*
/media/rootfs/usr/lib/
```

9. Unmount the SD card and re-insert into i.MX6 target platform. Restart i.MX6 platform.

```
$ cd /home/ubuntu/WE866Cx/
```

10. Run the Bluetooth application “**btapp**”

```
$ cd ~/WE866Cx/apps/bt_workspace/qcom-opensource/bt/property-ops

$ sudo ./btproperty &

$ cd ~/WE866Cx/apps/bt_workspace/qcom-opensource/bt/bt-app

$ sudo ./main/btapp
```

7. ACRONYMS

HCI	Host Controller Interface
DUT	Device Under Test
HID	Human Interface Device Profile
A2DP	Advance Audio Distribution Profile
GATT	Generic Attribute Profile
HOGP	HID Over GATT Profile
SPP	Serial Port Pro
DHCP	Dynamic Host Configuration Protocol

8. DOCUMENT HISTORY

Revision	Date	Changes
0	2018-06-21	First issue
1	2018-07-09	Updated section: B.2 Building the Kernel for i.MX 6 Platform 5.3 AP Mode
2	2018-08-27	Updated the document with respect to MCIMX6Q-SDB details.
3	2018-11-26	Updated the document with Bluetooth feature.
4	2018-12-11	Corrected Linux Commands under the following sections: A.2 Installing Linux Kernel v4.9.11 B.2Building the Kernel for i.MX 6 Platform B.5 Building Bluetooth Application for i.MX6 Host
5	2019-07-19	Added section 6.8 RSP Menu Added a note under section A.2 Installing Linux Kernel v4.9.11 Corrected typographical errors, “/DIR/” into “/$DIR/” in the document.
6	2019-08-16	Added a note for removing previous meta files before copying the btapp configuration files under the following sections: B.5 Building Bluetooth Application for i.MX6 Host 4.3.4 Loading WE866Cx Bluetooth Software Changed the commands related to Bluetooth testing under the following section: 6.1 Testing Bluetooth 4.3.4 Loading WE866Cx Bluetooth Software B.5 Building Bluetooth Application for i.MX6 Host Updated steps to test the BLE Mode under section 6.8 RSP Menu
7	2020-02-05	Corrected kernel build command under Appendix: B.2 Building the Kernel for i.MX 6 Platform A.2 Installing Linux Kernel v4.9.11
8	2020-04-20	Added section 5.7 Configuring DFS Master

9	2020-05-27	Changed the document name to “WE866Cx Wi-Fi and Bluetooth Network Interface Card (NIC) User Guide” to incorporate WE866C6 module. Added section 4.3.2 Loading WE866Cx Wi-Fi Driver Updated the following sections: B.2 Building the Kernel for i.MX 6 Platform B.4 Building WE866Cx Wi-Fi Linux Application for i.MX 6 on EVK Platform B.5 Building Bluetooth Application for i.MX6 Host
10	2020-06-08	Updated the following sections: 4.3.2 Loading WE866Cx Wi-Fi Driver B.2 Building the Kernel for i.MX 6 Platform B.4 Building WE866Cx Wi-Fi Linux Application for i.MX 6 on EVK Platform

SUPPORT INQUIRIES

Link to **www.telit.com** and contact our technical support team for any questions related to technical issues.

www.telit.com

Telit Communications S.p.A.
Via Stazione di Prosecco, 5/B
I-34010 Sgonico (Trieste), Italy

Telit IoT Platforms LLC
5300 Broken Sound Blvd, Suite 150
Boca Raton, FL 33487, USA

Telit Wireless Solutions Inc.
3131 RDU Center Drive, Suite 135
Morrisville, NC 27560, USA

Telit Wireless Solutions Co., Ltd.
8th Fl., Shinyoung Securities Bld.
6, Gukjegeumyung-ro8-gil, Yeongdeungpo-gu
Seoul, 150-884, Korea

Telit Wireless Solutions Ltd.
10 Habarzel St.
Tel Aviv 69710, Israel

Telit Wireless Solutions
Technologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil