QC-IOT4 DA16600 Software

This document describes the QC-IOT4 module's configuration switch settings and system setup interconnection on hardware components such as the RZ/G2L SMARC EVK. It also explains the steps and procedures used for building system software on a Linux host using Yocto.

For demonstration purposes, the RZ/G2L SMARC EVK is used throughout this document where the following is described:

- The methods used for flashing system binaries and booting the system
- The prerequisites for the QC-IOT4 module when in Linux mode and AT command mode
- Loading the driver module and setting up configuration files for Wi-Fi to be in Station mode and Access Point mode
- Performing tests to validate Wi-Fi and BLE drivers

Contents

Figures

Tables

1. Overview

The QC-IOT4 is a combo wireless PMOD module having Wi-Fi and Bluetooth® Low Energy (BLE) connectivity. The QC-IOT4 PMOD module is designed based on highly integrated ultra-low-power Wi-Fi + Bluetooth Low Energy using a DA16600 combo module solution. Two PMOD connectors are available for Wi-Fi and BLE communication with SPI and UART interfaces, respectively. All of the PMOD signals are also made available when the QCIOT4 module is interfaced with an Arduino shield connector.

1.1 Targeted Devices

- RZ/G2L SMARC EVK with QC-IOT4 module

1.2 Acronyms

Name	Description
AP	Access Point
AT Command	Attention Command
BLE	Bluetooth Low Energy
BSP	Board Support Package
DMA	Direct Memory Access
EVK	Evaluation Kit
GPIO	General Purpose Input Output
GPT	General Purpose Timer
HCI	Host Controller Interface
JTAG	Joint Test Action Group
MAC	Media Access Control
PMOD	Peripheral Module
QC-IOT4	Quick Connect IoT4
SMARC	Smart Mobility Architecture
SoC	System on Chip
SoM	System on Module
SDK	Software Development Kit
SPI	Serial Peripheral Interface
STA	Station
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity
RF	Radio Frequency

2. Operating Environment

Table 1 describes the software operating environment.

Table 1. Software Operating Environment

Item	Description
Demonstration Board RZ/G2L SAMRC EVK	RTK9744L23C01000BE RTK97X4XXXB00000BE
Microprocessor	RZ/G2L (R9A07G044L23GBG)
BSP	BSP V 3.0.1 – RTK0EF0045Z0021AZJ-v3.0.1.zip
OS	Linux (Kernel – 5.10V)
Build	Yocto
Boot Media	SPI Flash (Kernel and FS from Micro SD)
Tool Chain	Poky 3.1.4 – GCC
Combo Module	DA16600 MOD
Host Build PC	Ubuntu – 20.04

3. QC-IOT4 Module Details

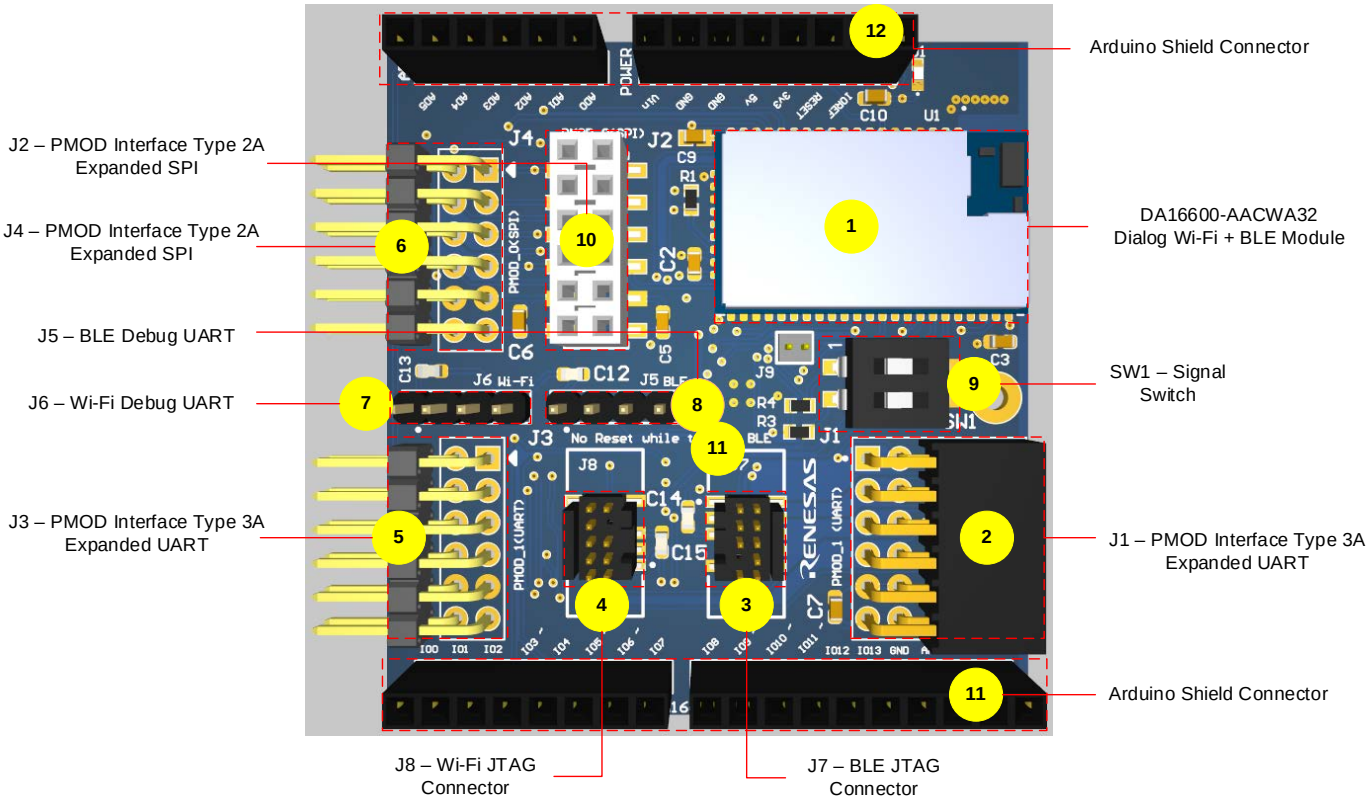


Figure 1. QC-IOT4 Hardware Module Details – Top

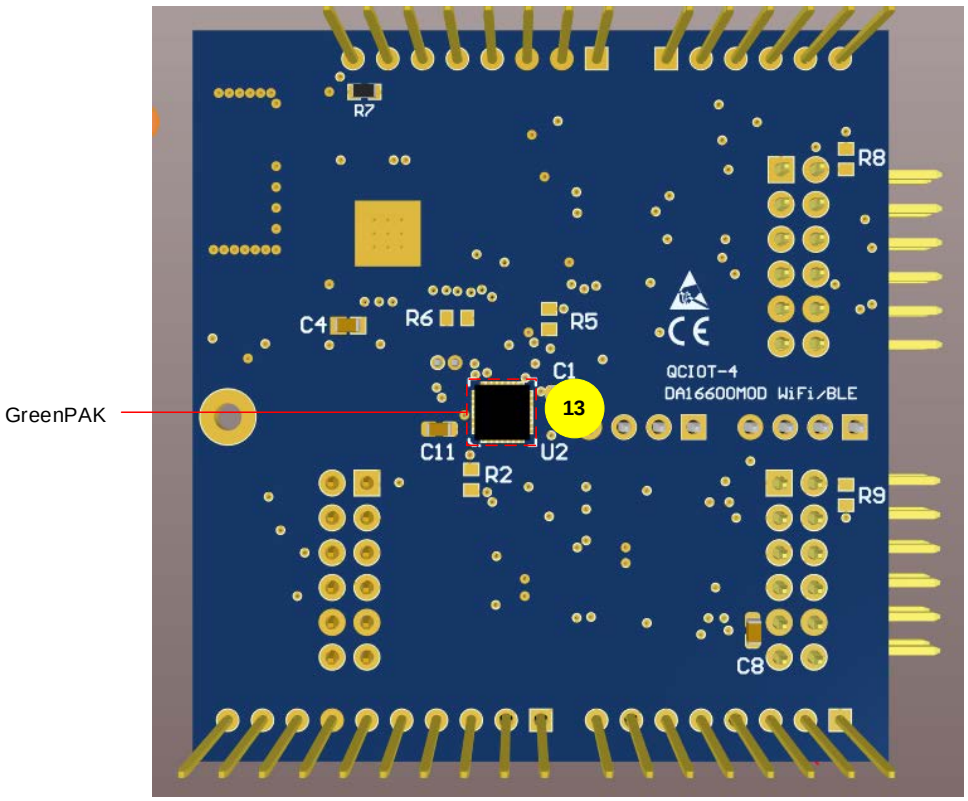
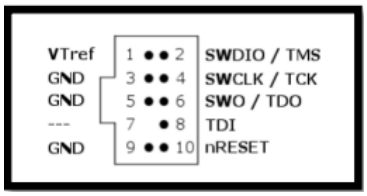
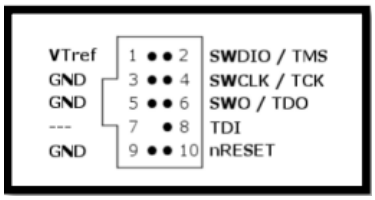


Figure 2. QC-IOT4 Hardware Module Details – Bottom

Table 2. QC-IOT4 Hardware Module Descriptions

Label No.	Name	Description
1	DA16600MOD-AACWA32	Dialog Wi-Fi and Bluetooth LE module.
2	PMOD (UART)	Expanded UART PMOD Type 3A connector.
3	J7 – BLE JTAG Connector	Connector for IARs I-jet JTAG debugger. 
4	J7- Wi-Fi JTAG Connector	Connector for IARs I-jet JTAG debugger. 
5	PMOD Connector UART	Expanded UART PMOD Type 3A connector.
6	PMOD Connector SPI	Expanded SPI PMOD Type 2A connector.
7	BLE Debug UART	Provides UART for BLE debug.

Label No.	Name	Description
8	Wi-Fi Debug UART	Provides UART for Wi-Fi debug.
9	Switch	Signal switch-to-switch between Linux and AT command mode.
10	PMOD Interface Type 2A (expanded SPI)	Expanded SPI PMOD Type 2A connector.
11, 12	Arduino Shield	Arduino shield connector.
13	SLG46880V	GreenPAK programmable mixed-signal matrix with asynchronous state machine and dual supply.

3.1 Configuration and Settings

3.1.1. Linux Mode

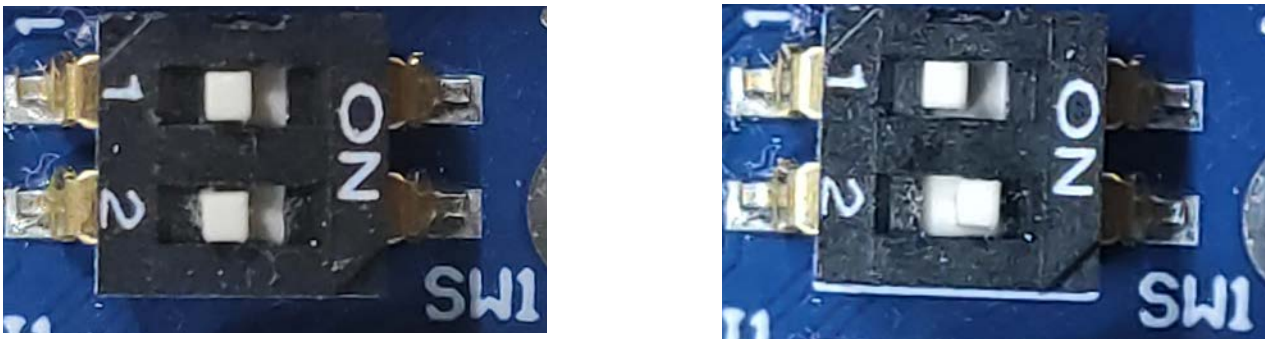


Figure 3. Linux Mode Switch Settings

3.1.2. AT Command Mode

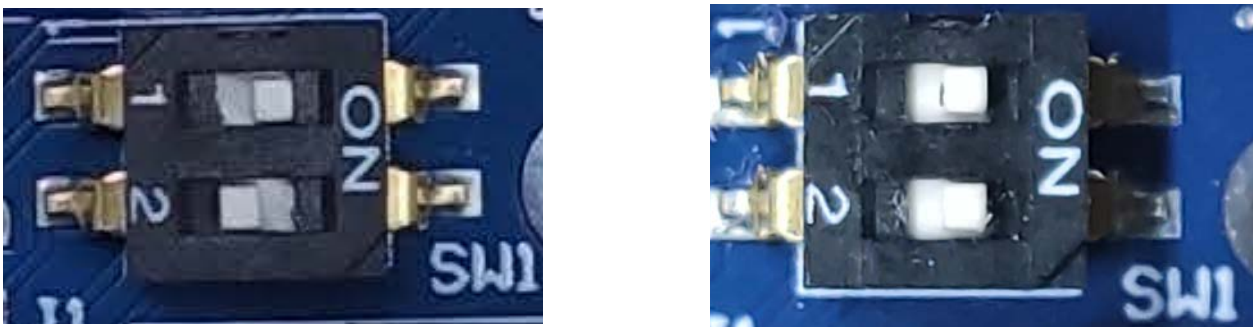


Figure 4. AT Command Mode Switch Settings

Table 3. QC-IOT4 Mode Descriptions

SW1	SW2	Mode	Description
ON	ON/OFF	Linux Mode	QC-IOT4 module can be controlled to Linux host with Wi-Fi over SPI and BLE over UART.
OFF	ON	AT Command Mode – SPI	QC-IOT4 module can be controlled through AT command over SPI.
OFF	OFF	AT Command Mode – UART	QC-IOT4 module can be controlled through AT command over UART.

3.2 Connection Setup

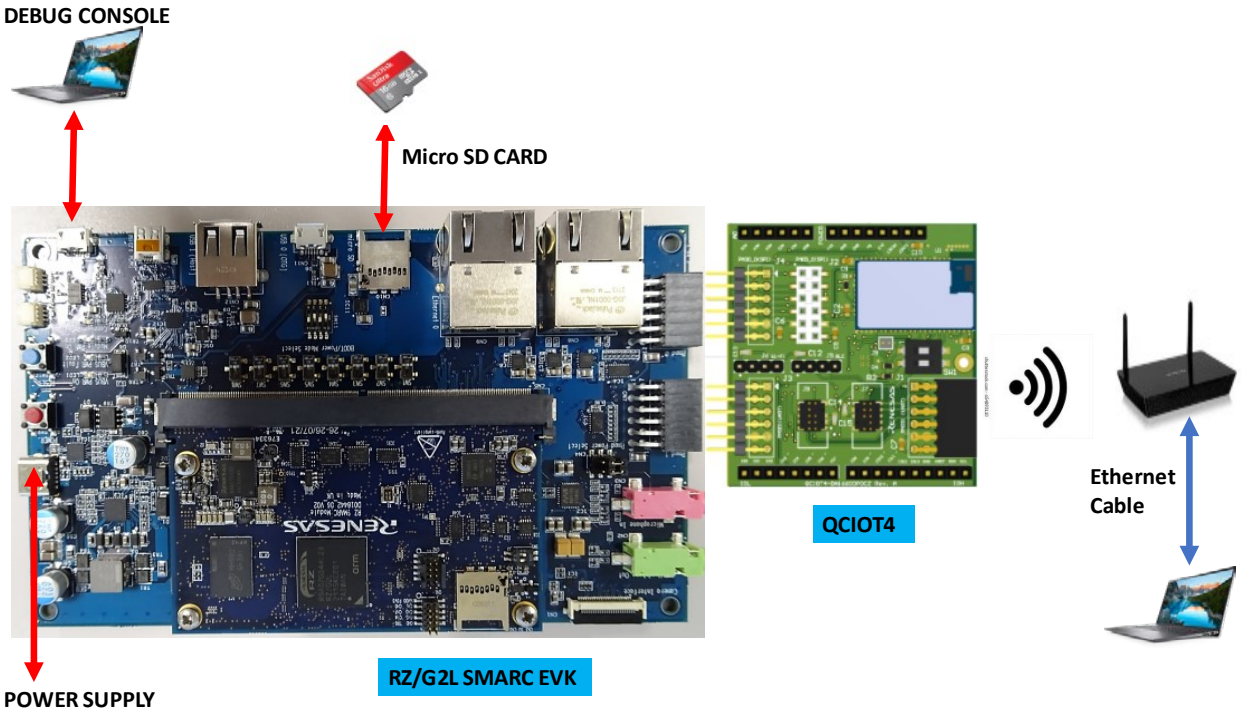


Figure 5. Connection Setup

1. Connect QC-IOT4 module to PMOD connector of RZ/G2L SMARC EVK platform.
2. Connect micro-SD card to the slot present in carrier board connection (see Figure 5).
3. Connect debug console to serial terminal application software (see Figure 5).
4. Wi-Fi Router that connects wireless to QC-IOT4.
5. Wi-Fi Router that connects wired/wireless to Desktop/Laptop.
6. Power on using USB-C type cable.

4. Software Build

The DA16200 is a highly integrated Wi-Fi system on chip (SoC) that allows users to develop Wi-Fi solutions using a single chip. The DA16600 also includes DA14531 BLE solution. The DA16200 (DA16600) provides the Linux driver for the SPI interface and BLE HCI driver for the UART interface so that it can be used as a MAC-only chip on the RZ/G2L Linux system.

The following sections are a quick-start guide to help new or existing developers develop Wi-Fi and BLE systems with the DA16200/DA16600 chipset on RZ/G2L Linux systems using the DA16200/DA16600's Linux driver source code.

4.1 RZ/G2L SMARC EVK

The Wi-Fi Linux driver is based on the Renesas RZ/G2L SMARC EVK with SD card image for Linux kernel and file system. The SD card image is written to the SD card using the MS Windows “Win32 Disk Imager” tool or BalenaEtcher tool.

4.2 Yocto Build

Builds the Linux BSP for RZ/G2L SMARC EVK to get kernel image, DTB and rootfs filesystem, along with bootloaders. The QC-IOT4 driver is built, tested, and validated using the RZ/G Verified Linux Package v3.0.1. Download the “RZ/G Verified Linux Package v3.0.1” package from RTK0EF0045Z9006AZJ-v3.0.1.zip.

Use the following steps to build system binaries/images.

1. Update the system with all host dependent packages.

```
$ sudo apt-get update
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib
\ build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-
utils \ debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa
libsdl1.2-dev \ pylint3 xterm
```

2. Configure the GIT.

```
$ git config --global user.email "you@example.com"
$ git config --global user.name "Your Name"
```

3. Unzip the downloaded package for RZ/G2L SMARC EVK.

```
$unzip RTK0EF0045Z0021AZJ-v3.0.1.zip
```

4. Untar the Renesas BSP file in RTK0EF0045Z0021AZJ-v3.0.1 folder (top directory).

```
RTK0EF0045Z0021AZJ-v3.0.1$tar -xvzf rzg_bsp_v3.0.1.tar.gz
```

5. Copy the meta-da16600.tar.gz to RTK0EF0045Z0021AZJ-v3.0.1 folder and Untar the BLE meta-layer in top directory.

```
RTK0EF0045Z0021AZJ-v3.0.1$ tar -xvzf meta-da16600.tar.gz
```

6. Set up the build environment.

```
RTK0EF0045Z0021AZJ-v3.0.1$ source poky/oe-init-build-env
```

7. Update the configuration file for RZ/G2L SMARC EVK machine.

```
RTK0EF0045Z0021AZJ-v3.0.1/build$cp ../meta-renesas/docs/template/conf/
smarc-rzg2l/*.conf ./conf/
```

8. Add below line for BLE meta-layer in bblayers.conf file in the build/conf folder under the line BBLAYERS ?= " \ and save.

```
${TOPDIR}/../meta-da16600 \
```

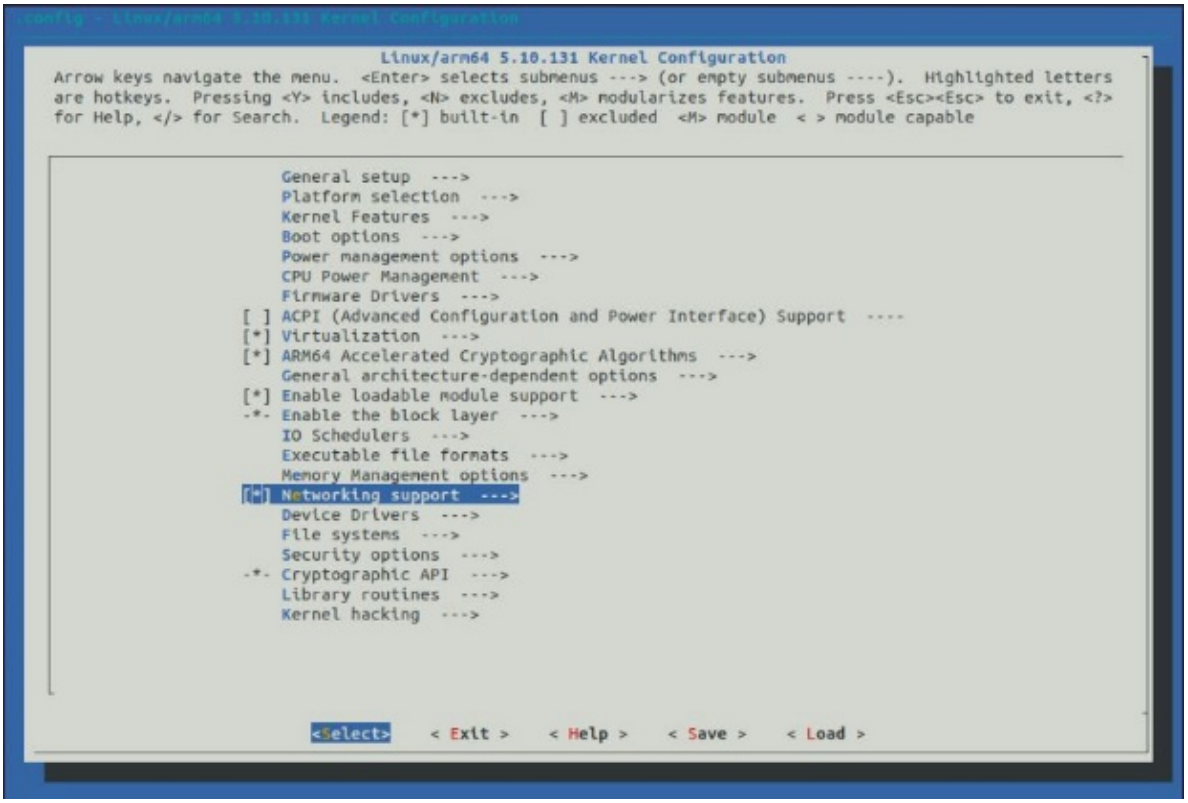
9. Update/Add below lines for DA16600 BLE in local.conf file.

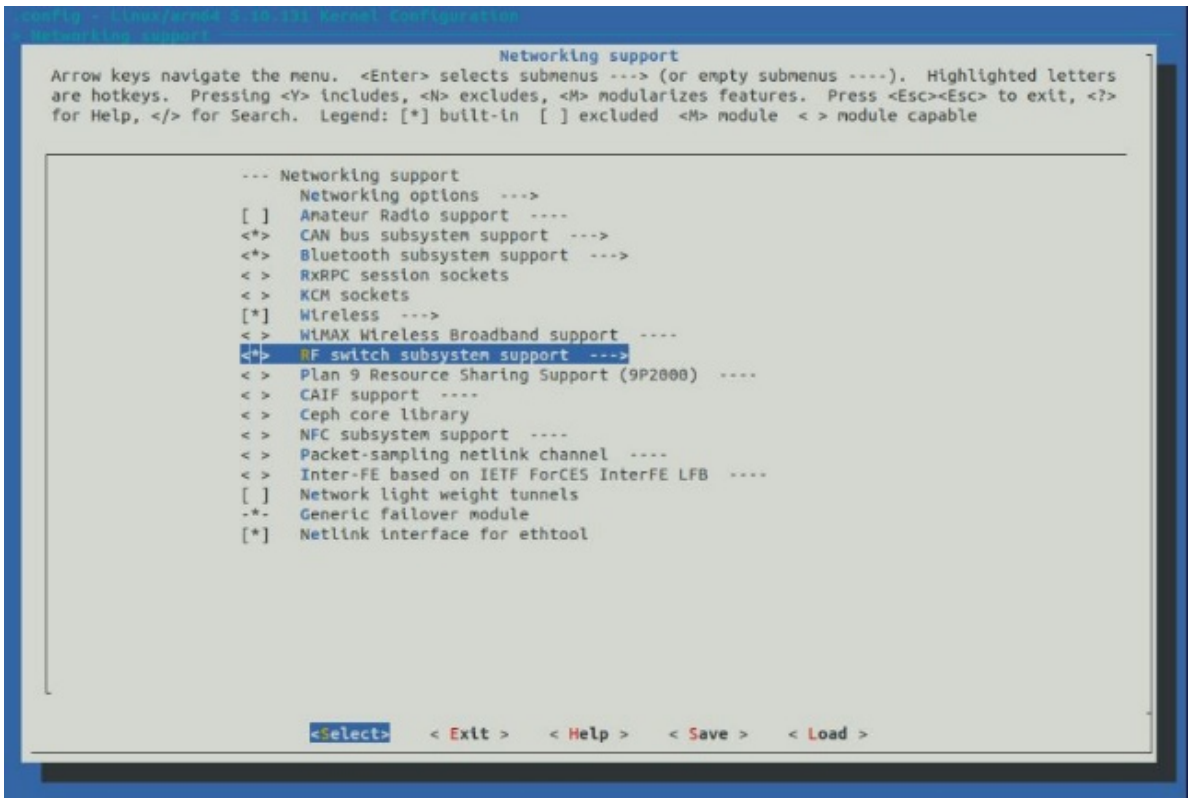
```
MACHINE_FEATURES += "bluetooth"
MACHINE_ESSENTIAL_EXTRA_RDEPENDS += " \
   linux-firmware-da14531 \
"
IMAGE_INSTALL_append = " kernel-modules bluez5 dhcpcd hostapd dnsmasq iw git
perltools python3-dbus python3-pygobject "
```

10. Configure the kernel.

```
RTK0EF0045Z0021AZJ-v3.0.1/build $ bitbake -c menuconfig virtual/kernel
```

a. Add RFKILL (see the three screens in Figure 6 for selections).





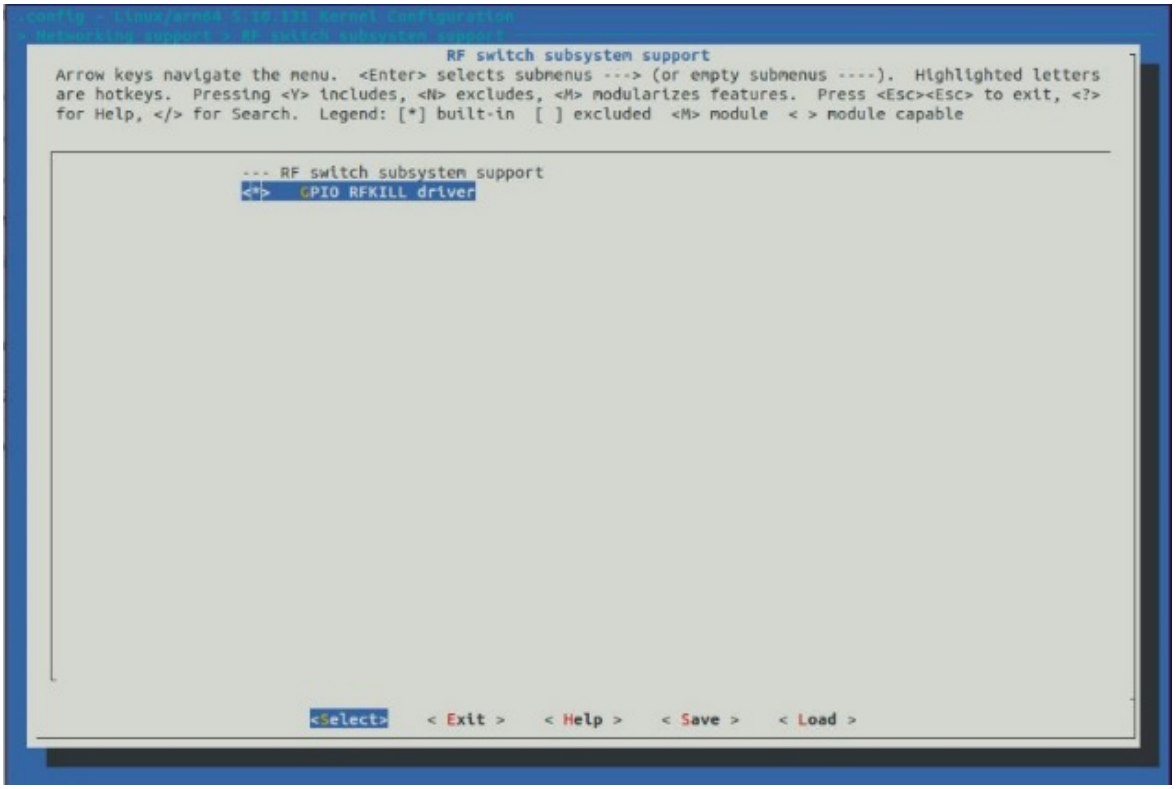
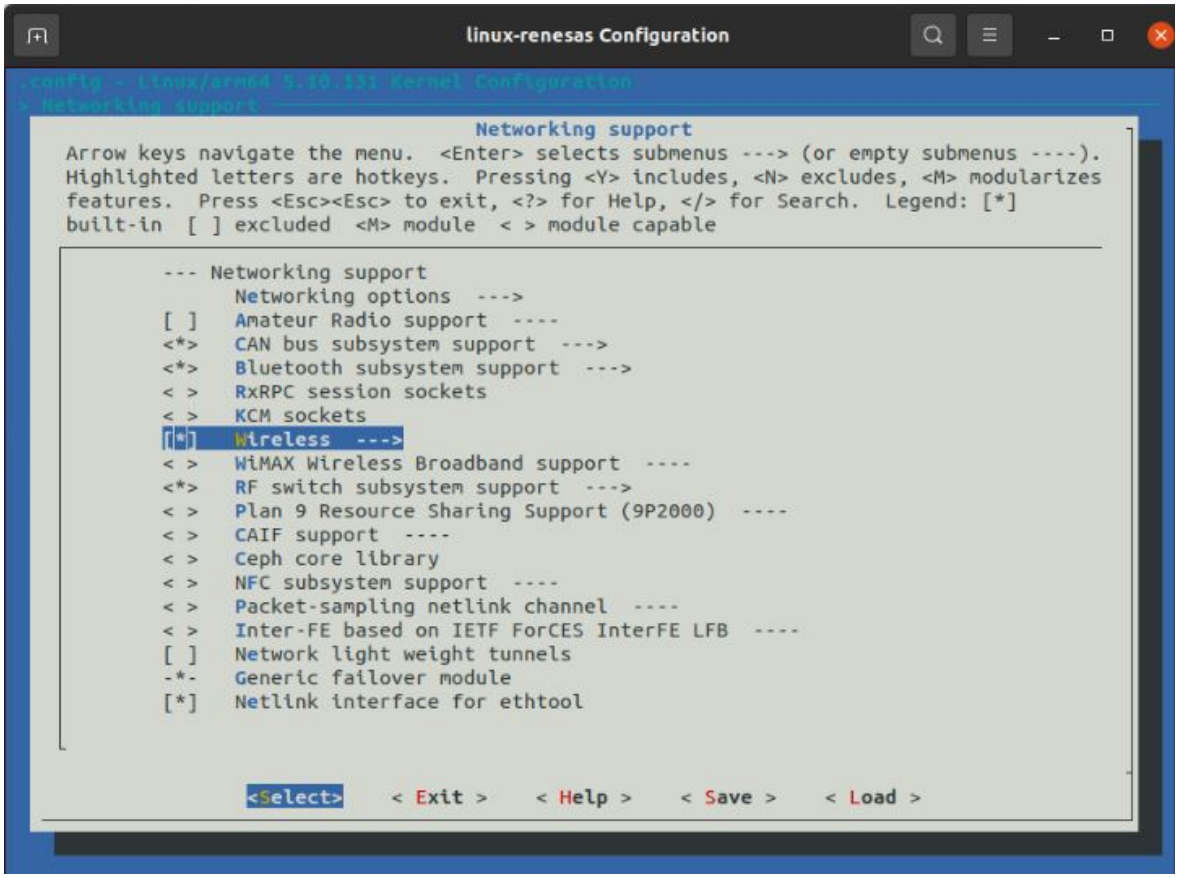


Figure 6. RFKILL Kernel Configuration

b. Wireless Driver support 802.11 (see the three screens in Figure 7 for selections).



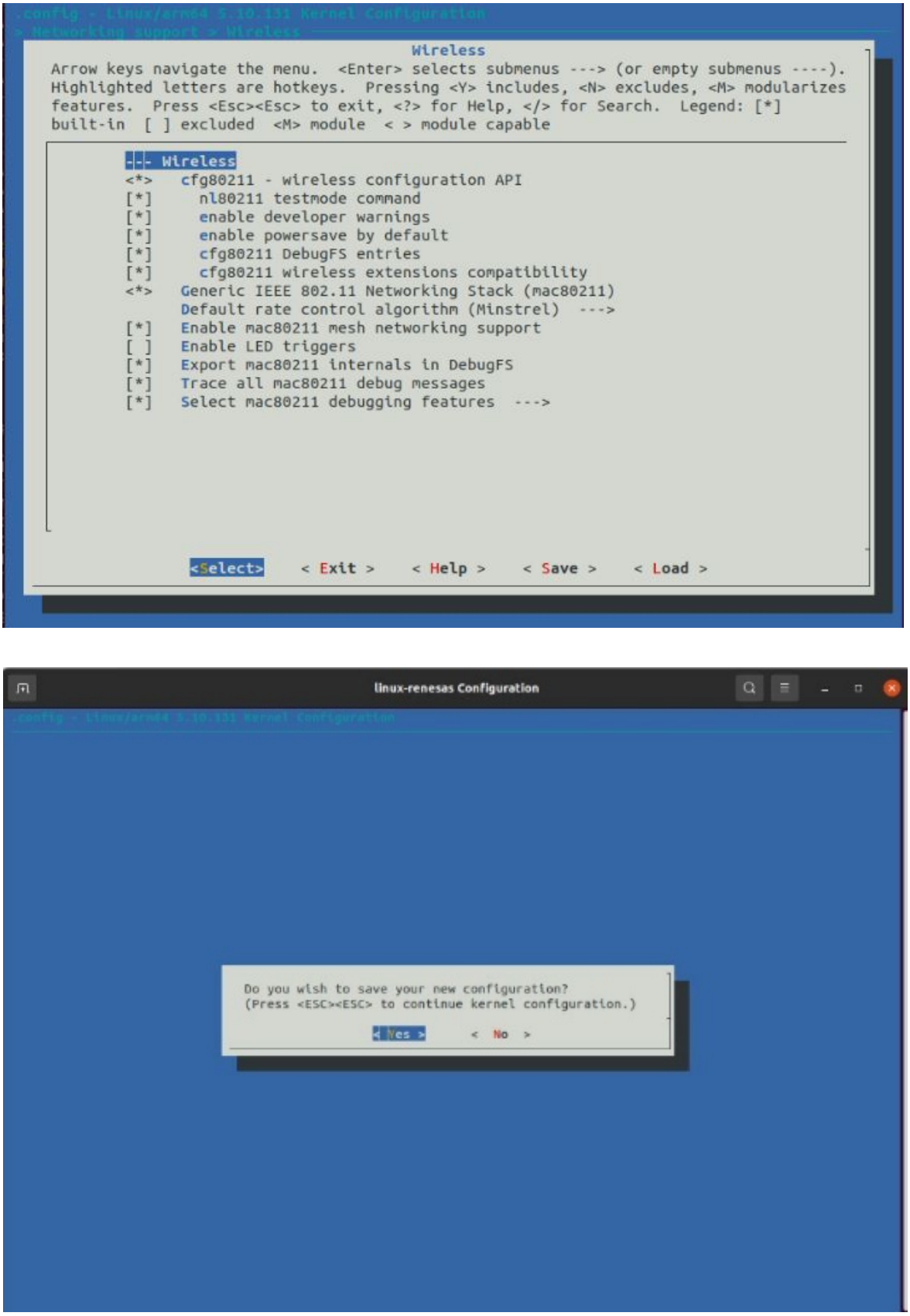


Figure 7. Wireless Kernel Configuration

11. Update DTSI file for Renesas BLE HCI driver.

    File: ${KER_DIR}/arch/arm64/boot/dts/renesas/rzg2l-smarc.dtsi

    Add the highlighted lines in yellow under PMOD1_SER0 macro.

```
#if PMOD1_SER0
&scif2 {
     pinctrl-0 = <&scif2_pins>;
     pinctrl-names = "default";
     uart-has-rtscts;
     status = "okay";
     bluetooth {
     compatible = "renesas,DA14531";
     reset-gpios = <&pinctrl RZG2L_GPIO(47, 2) GPIO_ACTIVE_LOW>;
     };
};
#endif
```

12. Update DTSI file for Renesas Wi-Fi driver.

    File: ${KER_DIR}/arch/arm64/boot/dts/renesas/rz-smarc-common.dtsi

    Add the highlighted lines in yellow under spi1 node section.

```
&spi1 {
   dma-names = "tx", "rx";

   status = "okay";
   da16xxx@0 {
        compatible = "renesas,da16xxx";
        reg = <0>;
        spi-max-frequency = <5000000>;
        reset-gpios = <&pinctrl RZG2L_GPIO(43, 2) GPIO_ACTIVE_LOW>;
        irq0-gpios = <&pinctrl RZG2L_GPIO(43, 1) GPIO_ACTIVE_LOW>;
        irq1-gpios = <&pinctrl RZG2L_GPIO(41, 0) GPIO_ACTIVE_LOW>;
   };
```

13. Disable the GPT4 support.

    File: ${KER_DIR}/arch/arm64/boot/dts/renesas/rzg2l-smarc-dev.dtsi

```
#define GPT4_SUPPORT            0
```

    Protect gpt4 node under GPT4_SUPPORT macro as shown below:

    File: ${KER_DIR}/arch/arm64/boot/dts/renesas/rzg2l-smarc-som.dtsi

```
#if GPT4_SUPPORT
&gpt4 {
   pinctrl-0 = <&gpt4_pins>;
   pinctrl-names = "default";
   channel = "both_AB";
#if (POEGD_SUPPORT)
   poeg = <&poega &poegb &poegc &poegd>;
#endif
   status = "okay";
};
#endif
```

14. Configuration Change.

    Update “IMAGE_INSTALL_append” setup as shown below:

```
File : build/conf/local.conf
IMAGE_INSTALL_append += " dhcpcd hostapd dnsmasq iw"
```

15. Enable serial interface on PMOD1 connector.

```
File: ${KER_DIR}/arch/arm64/boot/dts/renesas/rzg2l-smarc.dtsi
#define PMOD1_SER0     1
```

16. Build the target image for RZ/G2L SMARC EVK.

```
RTK0EF0045Z0021AZJ-v3.0.1/build $ bitbake core-image-weston
```

Note: For example, kernel directory location can be as follows:

```
KER_DIR = “RTK0EF0045Z0021AZJ-v3.0.1/build/tmp/work-shared/smarc-rzg2l/kernel
source”
```

Note: Use space bar for configuration option selection [‘*’, ‘M’, ‘ ‘], Left /Right Arrow Keys for navigation on menu options and configuration options. Press Enter key to get into configuration screens and menu screens.

4.3 Wi-Fi Driver Build

Renesas supports the DA16200 Linux driver source code, `rwnx_drv.tar.gz`. The driver is cross-compiled using "RZ/G Verified Linux Package V3.0.1" included in the RZ/G2L SDK. To compile the driver source, refer to “Linux Interface Specification Yocto recipe Start-Up Guide” to install the RZ/G2L SDK. To build the host driver, the path set in KDIR should be changed to the path of the build environment in use.

1. Navigate to Wi-Fi driver source directory.

```
$cd rwnx_drv
```

2. Update the Makefile.

   a. Update KDIR variable to the kernel directory location. Example is given below:

```
vi Makefile
# rzg_bsp_v3.0.1 released by Renesas
KDIR ?= RTK0EF0045Z0021AZJ-v3.0.1/build/tmp/work-shared/smarc-rzg2l/kernel
source
```

Note: The KDIR can vary due to configuration. The alternate is at kernel-build-artifacts.

3. Set the environment variables using source command.

   a. Use the poky version installed as given in the below command:

```
$ source /opt/poky/3.1.14/environment-setup-aarch64-poky-linux
```

4. Build the driver using make utility/command.

```
$ make
```

5. When compilation is complete, the “rwnx_drv.ko” kernel module is created in the rwnx_drv/softmac folder. Copy “rwnx_drv.ko” file into the rootfs file system for loading the driver as module.

5. Flashing/Programming

Refer to the SMARC EVK of RZ/G2L, RZ/G2LC, RZ/G2UL, RZ/V2L, and RZ/Five Start-up Guide for programming the following system binaries/images.

5.1 Build Binaries

5.1.1. SPI Flash

- **Flash_Writer_SCIF_RZG2L_SMARC_PMIC_DDR4_2GB_1PCS.mot**: Flash Writer Tool
- **bl2_bp-smarc-rzg2l_pmic.srec**: Boot loader
- **fip-smarc-rzg2l_pmic.srec**: Boot Loader

5.1.2. Micro SD Window Partition

- **Image**: Kernel Image
- **r9a07g044l2-smarc.dtb**: Device Tree Binary

5.1.3. Micro SD Linux Partition

- **core-image-weston-smarc-rzg2l.tar.gz** (alternative formats also available): rootfs file system

Place the *lmacfw_spi.bin* Wi-Fi firmware for DA16600 in the `/lib/firmware/` folder of rootfs file system.

Important: When loading the DA16600 Wi-Fi driver module, ensure to place the *rwnx_drv.ko* file in the location where the module is inserted. For example, you can copy to the `/home/root/` folder of rootfs file system.

Board Information: RZ/G2L, RZ/G2LC, RZ/G2UL SMARC Board by Renesas - Renesas.info

5.2 Pre-built Binaries

The micro-SD card can also be prepared from a pre-built image from the release binaries. Follow the below procedure/steps to program the SD card.

1. Prepare a 16GB microSD card with SD card reader and insert it into your laptop.
2. Run the flash writing tool on a laptop (for example, “BalenaEtcher” program).
   a. Installing the BalenaEtcher tool on Linux is shown below:

```
#curl -1sLf 'https://dl.cloudsmith.io/public/balena/etcher/setup.deb.sh' |
 sudo -E bash
#sudo apt install balena-etcher-electron
```

3. Click the *Flash from file* icon and select the Demo image file (QCIOT4_SDCARD_16GB.img). Then, click *Flash* (see Figure 8).

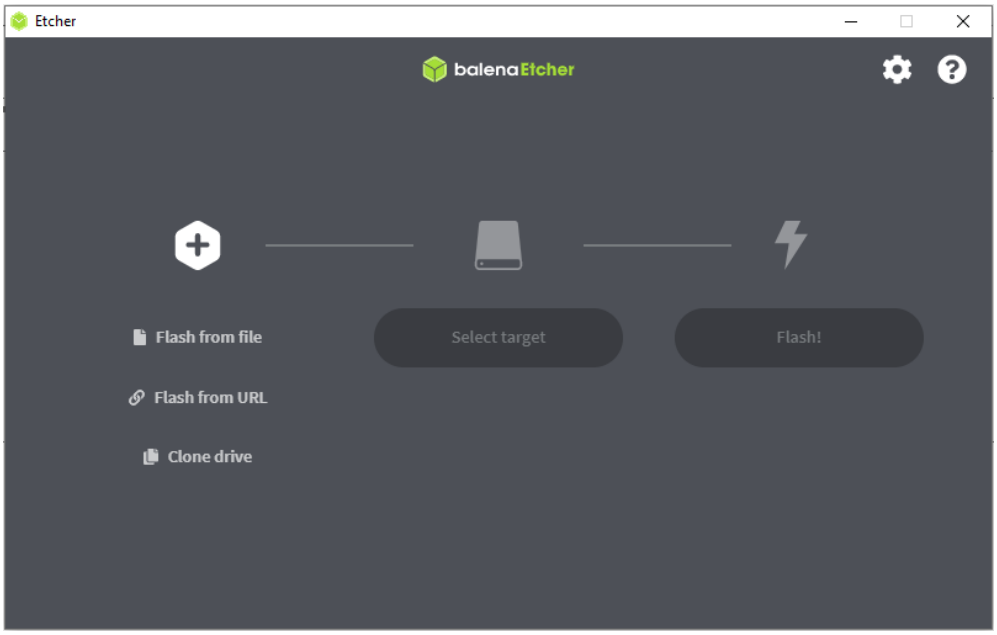


Figure 8. BalenaEtcher Tool

6. Boot

1. Insert the programmed micro-SD card in the slot as shown in the connection setup diagram (see Figure 5).
2. Power on the setup with a USB-C type power adaptor (see Figure 5).
3. Open a serial debug console terminal application and set the serial communication parameters. An example of the *Tera Term* serial port setup and configuration is given below.

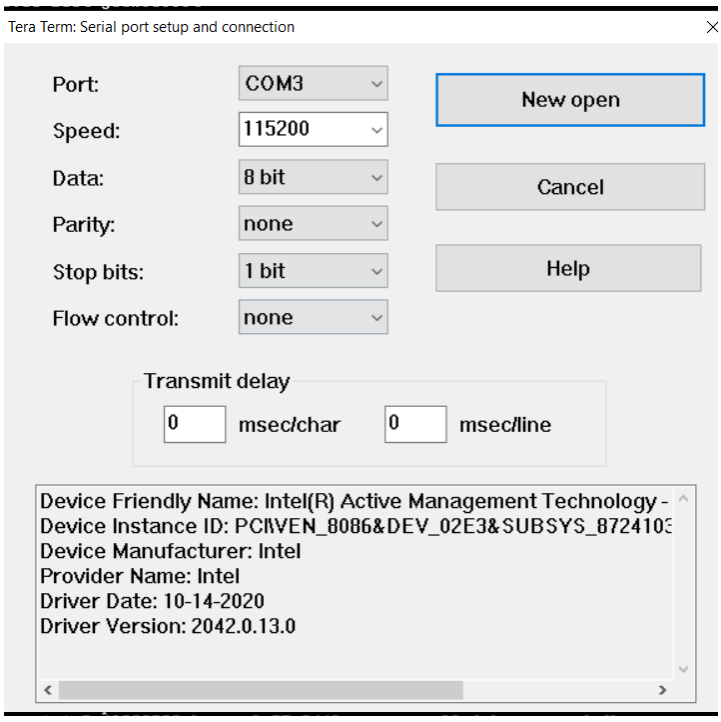


Figure 9. Terminal Configuration

4. Because the Linux kernel image, Device Tree Blob DTB and rootfs file system are placed/stored in the micro-SD card, the U-Boot environment arguments must be modified accordingly as shown in Figure 10. Verify that the modified boot arguments and filenames match. Once verified, boot the Linux kernel.

```
=> setenv sd_boot1 'mmc dev 1 ; fatload mmc 1:1 0x48080000 Image ; fatload mmc 1:1
0x48000000 /Image-r9a07g044l2-smarc.dtb'

=> setenv sd_boot2 'setenv bootargs 'root=/dev/mmcblk1p2 rootwait' ; booti 0x48080000
- 0x48000000'

=> setenv bootcmd 'run sd_boot1 sd_boot2'

=> saveenv

 Saving Environment to MMC... Writing to MMC(0)….OK
```

Figure 10. U-Boot Arguments Settings – µSD

5. Log in as root user to view the Linux kernel debug console (see Figure 11).

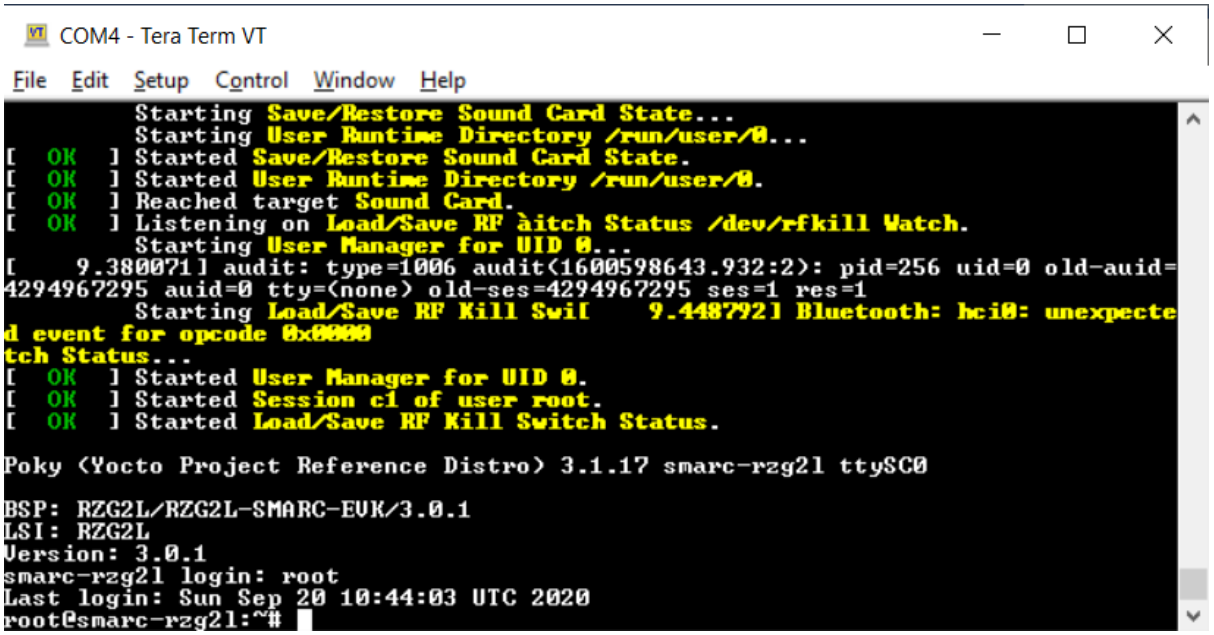


Figure 11. Boot Console Log

Important: Ensure that the QC-IOT4's flash is erased before connecting it to the host system, such as the RZ/G2L SMARC EVK.

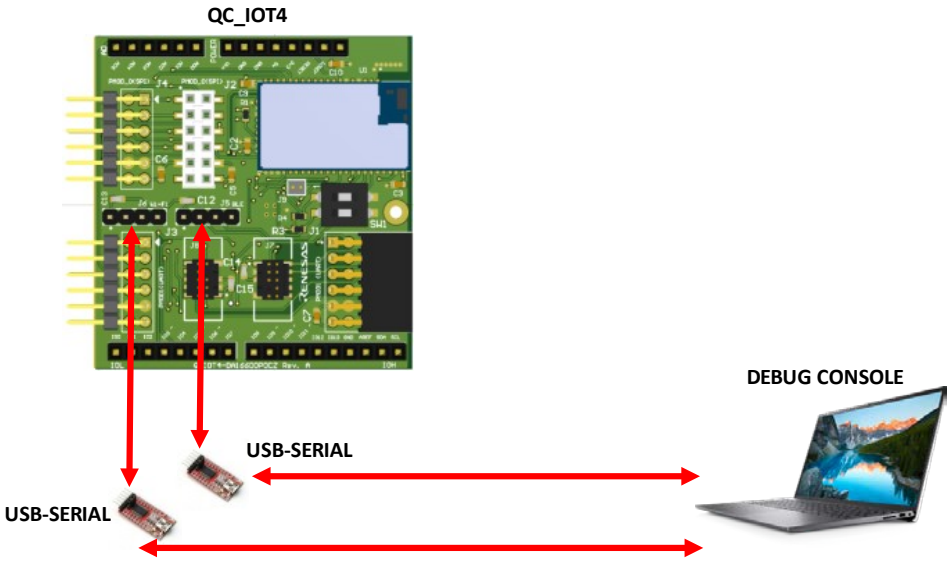


Figure 12. QC-IOT4 Flash Erase – Setup

6. After connecting to DA16200 or DA166600 through UART, execute the command below at the prompt.

```
[combo] reset
[MROM] sflash erase 0 1000
[MROM] sflash read 0 100
```

7. Check to see if the last line, FF FF FF FF… is printed (see Figure 13). If so, the memory removal was successful.

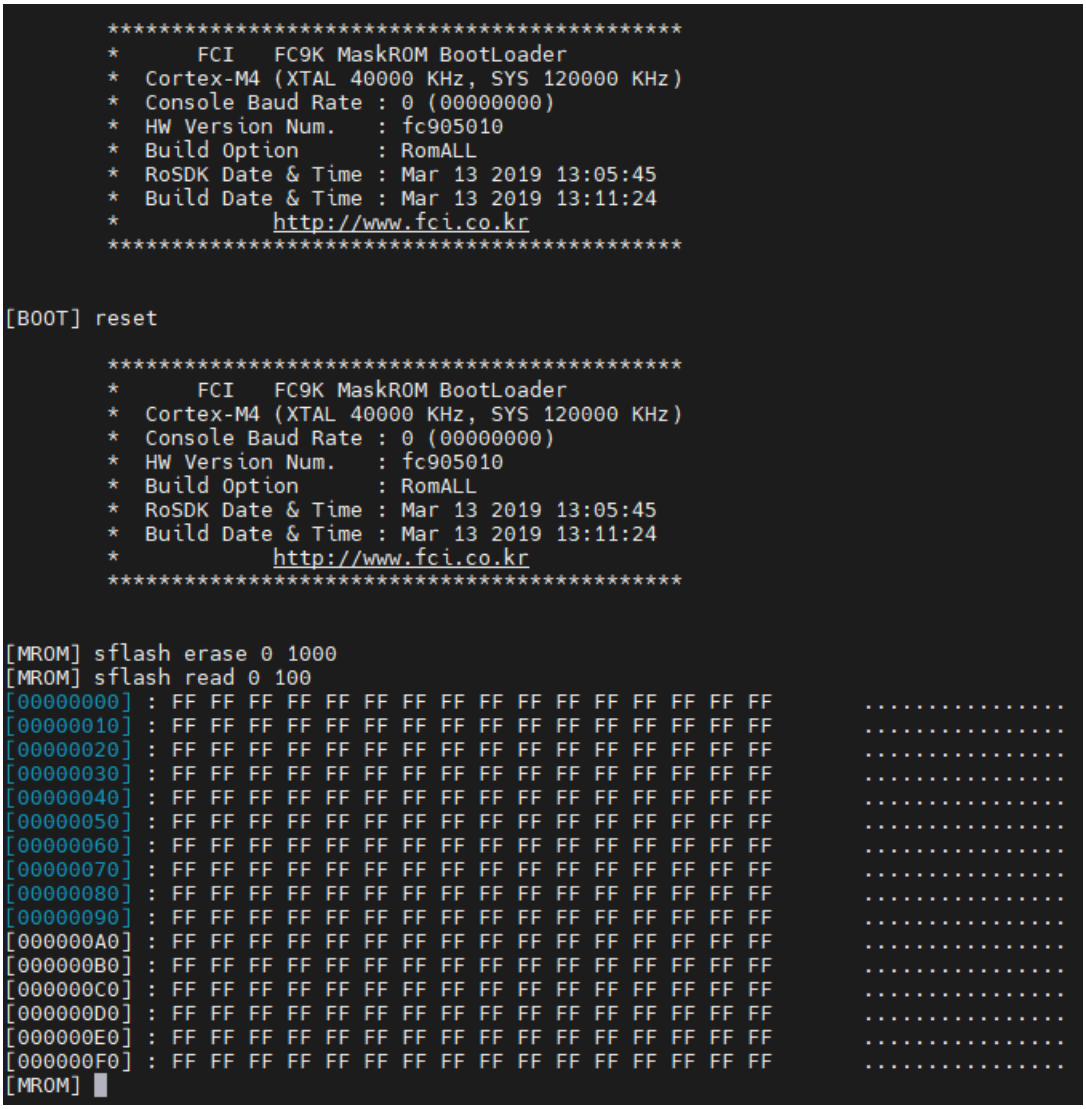


Figure 13. QC-IOT4 Flash Erase – Debug Console

7. Test

- Prepare the setup as shown in the connection setup diagram (see Figure 5).
- Power on the system (see section 6).

7.1 Wi-Fi

7.1.1. QC-IOT4 STA Mode Test

1. Edit WPA supplicant configuration file with credentials of Wi-Fi Router AP.

   ```
   File: /etc/wpa_supplicant.conf
   ```

   Example:

   ```
   network={
      ssid=“SSID_name"
      key_mgmt.=NONE
   }

   network={
      ssid="SST-2.4G"
      scan_ssid=1
      key_mgmt=WPA-PSK
      psk="REIN_sst@!123"
   }
   ```

2. Load the Wi-Fi Linux driver module.

   ```
   $insmod rwnx_drv.ko
   ```

3. Execute DHCP client.

   ```
   $dhcpcd -q -b
   ```

4. Enable all wireless device, If required and necessary (blocked case).

   ```
   $rfkill unblock all
   ```

5. Execute WPA supplicant for key negotiation with authenticator counterpart.

   ```
   $wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf &
   ```

6. Install iperf application utility program on laptop/desktop.
7. Execute iperf server application in laptop/desktop.

   ```
   #iperf3 -s -I 1
   ```

8. Execute iperf client application in RZ/G2L SMARC EVK

   ```
   #iperf3 -c 192.168.1.3 -t 1000 -i 1
   ```

   (Use `ip addr` command to get IP address of the laptop/desktop. Example: 192.168.1.3).

```
root@smarc-rzg2l:~# iperf3 -c 192.168.1.3 -i 1
Connecting to host 192.168.1.3, port 5201
[  5] local 192.168.1.2 port 43752 connected to 192.168.1.3 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   345 KBytes  2.83 Mbits/sec    0   35.4 KBytes
[  5]   1.00-2.00   sec   187 KBytes  1.53 Mbits/sec    0   35.4 KBytes
[  5]   2.00-3.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   3.00-4.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   4.00-5.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   5.00-6.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   6.00-7.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   7.00-8.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   8.00-9.00   sec   249 KBytes  2.04 Mbits/sec    0   35.4 KBytes
[  5]   9.00-10.00  sec   249 KBytes  2.04 Mbits/sec    1   24.0 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  2.46 MBytes  2.07 Mbits/sec    1             sender
[  5]   0.00-10.03  sec  2.35 MBytes  1.97 Mbits/sec                  receiver

iperf Done.
root@smarc-rzg2l:~#
```

```
madhusudhan@madhusudhan-X270:~$ iperf3 -s -i 1
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.1.2, port 43746
[  5] local 192.168.1.3 port 5201 connected to 192.168.1.2 port 43752
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec   225 KBytes  1.84 Mbits/sec
[  5]   1.00-2.00   sec   235 KBytes  1.92 Mbits/sec
[  5]   2.00-3.00   sec   240 KBytes  1.97 Mbits/sec
[  5]   3.00-4.00   sec   249 KBytes  2.04 Mbits/sec
[  5]   4.00-5.00   sec   238 KBytes  1.95 Mbits/sec
[  5]   5.00-6.00   sec   221 KBytes  1.81 Mbits/sec
[  5]   6.00-7.00   sec   245 KBytes  2.00 Mbits/sec
[  5]   7.00-8.00   sec   257 KBytes  2.11 Mbits/sec
[  5]   8.00-9.00   sec   267 KBytes  2.19 Mbits/sec
[  5]   9.00-10.00  sec   226 KBytes  1.85 Mbits/sec
[  5]  10.00-10.03  sec  7.07 KBytes  2.13 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-10.03  sec  2.35 MBytes  1.97 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

Figure 14. Wi-Fi – STA iperf Test Log

7.2 BLE

1. Use `hciconfig` command to list the device that is present on the HCI interface (see Figure 15).

```
root@smarc-rzg2l:~# hciconfig
hci0:   Type: Primary  Bus: UART
        BD Address: 48:23:35:A1:B6:C8  ACL MTU: 251:9  SCO MTU: 0:0
        UP RUNNING
        RX bytes:285 acl:0 sco:0 events:23 errors:0
        TX bytes:19546 acl:0 sco:0 commands:22 errors:0

root@smarc-rzg2l:~#
```

Figure 15. BLE – HCI Interface

2. Make the device present on interface HCI0 up and scan for proximity BLE devices (see Figure 16).

```
root@smarc-rzg2l:~# hciconfig hci0 up
root@smarc-rzg2l:~# hcitool lescan
LE Scan ...
18:DF:C0:93:E4:7D (unknown)
00:B8:5E:20:CA:56 (unknown)
16:73:42:43:EF:AE (unknown)
05:A1:6C:54:CD:20 (unknown)
1A:76:DF:7A:A6:41 (unknown)
1D:26:ED:93:22:99 (unknown)
19:03:CE:7B:7C:A4 (unknown)
0D:C3:99:D2:2B:B7 (unknown)
7A:B4:DC:E3:E7:8C (unknown)
7A:B4:DC:E3:E7:8C (unknown)
FE:33:F0:B9:09:59 (unknown)
08:A4:0C:59:D7:42 (unknown)
29:AB:A9:E9:57:CD (unknown)
```

Figure 16. BLE – Scan

3. A connection with one of the peripheral devices in the scanned lists can be made using the `hcitool lecc` command. See Figure 17 for an example.

```
root@smarc-rzg2l:~# hcitool lescan
LE Scan ...
66:BE:1B:88:12:68 (unknown)
EB:86:71:13:C6:A0 GAZe3728
EB:86:71:13:C6:A0 (unknown)
88:0F:10:93:F0:BE (unknown)
7D:01:65:1A:7C:51 (unknown)
E0:9F:2A:E2:23:45 DA16600-5EC6
E0:9F:2A:E2:23:45 (unknown)
4A:79:B0:2A:1E:CB (unknown)
4A:79:B0:2A:1E:CB (unknown)
4D:3B:FD:DC:2F:4D (unknown)
88:0F:10:93:F0:BE MI_SCALE
4D:3B:FD:DC:2F:4D (unknown)
^Croot@smarc-rzg2l:~# hcitool lecc E0:9F:2A:E2:23:45
Connection handle 0
```

Figure 17. BLE – Connect

4. GATT Example:

   The target RZ/G2L board should have network connectivity. For example, RJ45 connector can be used for this purpose. Connect an Ethernet cable to the RJ45 port.

   In the Linux console, execute following command:

   ```
   # git clone https://github.com/Jumperr-labs/python-gatt-server.git
   # cd python-gatt-server
   # python3 gatt_server_example.py
   ```

   If HCI interface is blocked, unblock it with the following command:

   ```
   # rfkill unblock all
   ```

   Install Renesas' "SmartBond" mobile application. An example of device scanning and connection through the mobile app is shown in Figure 18.

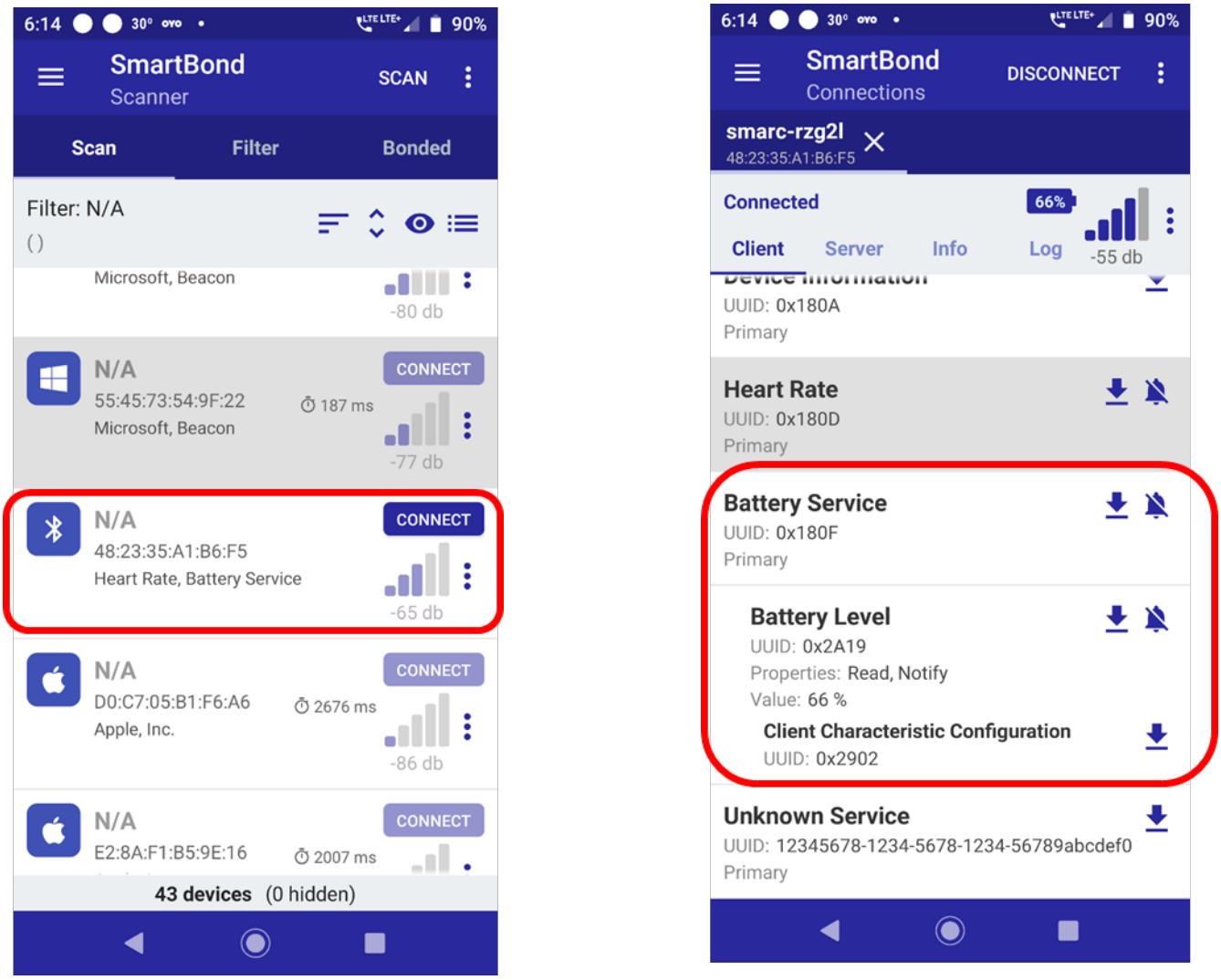


Figure 18. BLE GATT – Mobile App

8. QC-IOT4 Applications Schematic

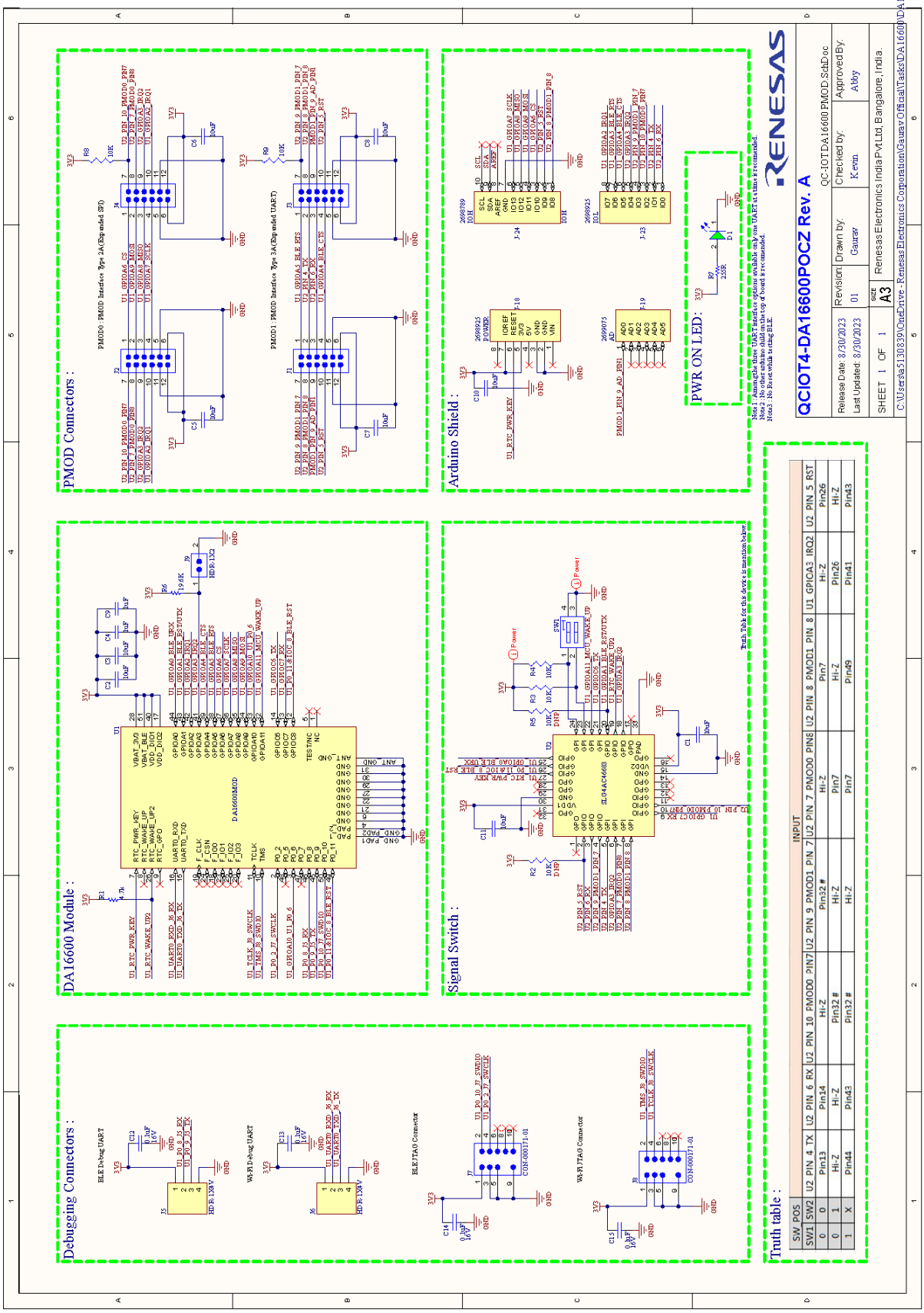


Figure 19. QC-IOT4 Applications Schematic

9. Revision History

Revision	Date	Description
1.00	Sep 13, 2023	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use o any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.0 Mar 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.