NXP Semiconductors

Document Number: IIoT_UG
Rev. 1.1, 03/2018

Industrial IoT User Guide

Contents

Chapter 1
Introduction

The Industrial IoT (IIoT) project is based on the open source project LEDE (Linux Embedded Development Environment), and has the source branch base tag v17.01.2. It includes four hardware development platforms: LS1021A-IoT, LS1012ARDB, LS1043ARDB, and LS1046ARDB. All these platforms can be built from LEDE. The table below lists the industrial IoT features each platform can support.

Table 1. IoT features supported by NXP hardware platforms

IoT feature / platform	LS1021A-IoT	LS1012ARDB	LS1043ARDB	LS1046ARDB
BLE	Yes	Yes	No	No
Thread	Yes	Yes	No	No
NFC	Yes	Yes	No	No
WiFi	Yes	Yes	Yes	Yes
Docker	No	No	Yes	Yes
ZigBee	No	Yes	No	No
OTA	Yes	Yes	Yes	Yes
4G-LTE	Yes	Yes	Yes	Yes
Sigfox	Yes	No	No	No

This document covers how to prepare the hardware setup and build images using the LEDE project. It also describes how to prepare the hardware setup to demonstrate and verify the available features such as BLE connectivity, thread connectivity, NFC, Wifi, Docker, ZigBee scenario, OTA implementation, 4G-LTE modem and Sigfox.

1.1 LEDE overview

LEDE is a highly extensible GNU/Linux distribution for embedded devices (typically wireless routers). LEDE is based on the OpenWrt project. Unlike many other distributions for routers, LEDE is built to be a to be a full-featured, easily modifiable operating system for embedded devices. In practice, this means that you can have all the features you need with none of the overheads, powered by a modern Linux kernel.

For more information, refer to the site: https://lede-project.org/.

1.2 Abbreviations and acronyms

The following acronyms and abbreviations have been used in this document.

Table 2. Acronyms and abbreviations

Acronym	Description
BLE	Bluetooth® low energy

Table continues on the next page...

Table 2. Acronyms and abbreviations (continued)

Acronym	Description
HCI	Host controller interface
IoT	Internet of things
LEDE	Linux embedded development environment
LPWA	Low-power wide-area
LS	Layerscape
OTA	Over-the-air
QSPI	Quad SPI interface
RCW	Reset configuration word
RDB	Reference design board
SPI	Serial peripheral interface
TCP	Transmission control protocol
UART	Universal asynchronous receiver-transmitter
UDP	User datagram protocol

1.3 Reference documentation

Refer to the following documents:

- Before booting up the LS1012ARDB, LS1021A-IoT, LS1043ARDB, or LS1046ARDB boards, refer to the detailed instructions listed in the *Getting Started Guide* of the respective board. These are:
  - LS1012ARDB Getting Started Guide.
  - LS1021-IoT Getting Started Guide.
  - LS1043ARDB Getting Started Guide.
  - LS1046ARDB Getting Started Guide.
- For details on implementing the ZigBee demonstration scenario based on LS1012ARDB and the JN516x-EK004 Evaluation Kit, refer to *Zigbee Solution on LS1012ARDB for Industrial IoT Quick Start Guide*.
- For a complete description of the various TSN demo scenarios and their results, see *Chapter 7.6,* "*Managing configurations with the sja1105-tool*" of the *Open Industrial Linux User Guide*.

1.4 Setting up the host environment

The following command provides the detailed package list on the Ubuntu or Debian host:

```
$ sudo apt-get install build-essential libncurses5-dev gawk git subversion libssl-dev gettext unzip
zlib1g-dev file python
```

Then, get the LEDE source code from the tarball using the commands below:

```
git clone https://github.com/iiot-gateway/lede-source-17.01.2.git
$ cd lede-source
$ ./scripts/feeds update -a
$ ./scripts/feeds install -a
```

NOTE

When you first run the '`make menuconfig`' command on the host PC, it might display errors for some missing packages. You need to install these packages, as prompted.

1.5 Rebuilding the LEDE project

In order to rebuild the LEDE project, you should clean the project using the following commands:

```
$make clean
$rm -rf tmp .config dl/*
```

1.6 Building and deploying images on NXP platforms

This section describes the switch settings and procedure for building and deploying SD card or QSPI images for the NXP hardware platforms supporting Industrial IoT.

As a prerequisite, refer to the Getting Started Guide of your board for detailed instructions to boot up the respective board.

NOTE

You might need to perform the following steps before booting up the boards described in this section:

- Install the mbed Windows serial port driver in order to obtain the board console. This is a one time activity. Please ignore this step if you have already installed mbed driver on your system (PC or laptop). You can download the mbed Windows serial port driver from the link:
  - https://developer.mbed.org/handbook/Windows-serial-configuration.
- Download and install Tera Term on the host computer from the Internet. After installation, a shortcut to the tool is created on the desktop of the host computer.
- If you are using a Windows 10 machine as a host computer and encountering a serial port unstable issue, you need to disable the *Volume Storage* service of the Windows machine.

1.6.1 LS1021A-IoT platform

This section describes the switch settings and procedure for building and deploying SD card images for the LS1021A-IoT platform.

1.6.1.1 Switch settings for LS1021A-IoT board

The following table lists and describes the switch configuration for the LS1021AIOT board.

Table 3. Switch settings for LS1021A-IoT board

Feature	Settings (OFF=1, ON=0)	Option	Comments
S2.1	OFF	RCW & Boot Source	0: QSPI (not supported on revA board) 1: SDHC (default)
S2.2	OFF	SYSCLK Select	0: DIFF_SYSCLK 1: SYSCLK (default)
S2.3	OFF	Reserved	0: Reserved 1: Reserved (default)
S2.4	OFF	Reserved	0: Reserved 1: Reserved (default)
S2.5	OFF	SGMII2_SATA MUX	0: SerDes Lane 2 - SATA 1: SerDes Lane 2 – SGMII2 (default)
S2.[6 :7]	ON:ON	SYSCLK Frequency Select	00: 100 MHz (default) 01: 99 MHz 10: 96 MHz 11: Reserved
S2.8	OFF	SDA_SWD_EN Control	0: K22 CMSIS-DAP 1: JTAG HEADER (default)

The following table lists the jumper settings.

Table 4. Jumper settings of LS1021A-IoT

Jumpers	Default settings on LS1021A-IoT	Description
J11	OFF	• VDD_LP Source Select • OFF – Battery • ON - +1V0_VDDC
J18	OFF	Reserved
J19	OFF	Reserved
J20	OFF	Reserved

1.6.1.2 Building the SD card image

Use the following commands to compile the image that can be programmed into the SD card:

```
$ cd lede-source
$ make menuconfig
```

Configure the settings as shown below:

```
Target System (NXP Layerscape) --->
           (X) NXP Layerscape
Subtarget (layerscape armv7 boards) --->
           (X) layerscape armv7 boards
Target Profile (ls1021aiot) --->
           (X) ls1021aiot
Target Images --->
           [*] ext4 --->
           [ ] squashfs ----
           [ ] GZip images
           (20) Boot (SD Card) filesystem partition size
                  (This FAT_fs partition is stored for kernel and DTB)
           (256) Root filesystem partition size (in MB)
           (This ext4_fs partition is stored for rootfs)
[*] Advanced configuration options (for developers) --->
           [*] Toolchain Options --->
                C Library implementation (Use glibc) --->

$make -j1 V=s
```

After building, the SD card image for LS1021A-IoT is generated at:

`bin/targets/layerscape/armv7-glibc/lede-layerscape-armv7-ls1021aiot-ext4-firmware.bin`

1.6.1.3 Deploying the SD card image

Deploy the SD card image for LS1021A-IoT board directly into an SD card using the following command:

```
$dd if=bin/targets/layerscape/armv7-glibc/lede-layerscape-armv7-ls1021aiot-ext4-firmware.bin
of=/dev/mmcblk0
```

Once deployment is complete, plug the SD card into the LS1021A-IoT board and boot the board.

1.6.2 LS1012ARDB platform

This section describes the switch settings and procedure for building and deploying SD card and QSPI images for the LS1021ARDB platform.

1.6.2.1 Switch settings for LS1012ARDB board

Table 5. Switch settings of LS1012ARDB

Feature	Settings (ON=1,OFF=0)	Option	Description
S1.1	ON	SW_RCW_SRC1	0 - Hard coded source 3 (Reserved) 1 - QSPI is the RCW source (default)
S1[2-3]	ON:ON	SW_ENG_USE/SW_ENG_USE2	XOSC Transconductance Multiplier 0 0 - 0.21x 0 1 - 0.55x 1 0 - 0.66x 1 1 - 1.00x (default)
S1.6	ON	BATT_OTG_BST_EN_B	OTG 5 V VBUS enable 0 - Enable the VBUS boost regulator of LTC4155 1 - Disable VBUS Boost regulator of LTC4155 (default)1 : Reserved (default)
S1.7	OFF	CFG_UART_MUX_EN_B	0- LS1012A UART1 is connected to K22 UART0 for CMSIS DAP debug. (default) 1 - LS1012A UART1 0 is translated to RS232 levels and is available on 1x3header (J24).
S1.8	OFF	CFG_SERDES_MUX_SEL	SERDES Lane A MUX selection 0 - SGMII 1G to PHY (default) 1 - PCIe TX clock to mini PCIeconnector
S2[1-2]	ON:ON	CFG_MUX_SDHC2_S0/ CFG_MUX_SDHC2_S1	SDHC 2 interface demultiplexer select lines 00 - SDIO WiFi (default) 01 - GPIO (to Arduino) 10 - eMMC Memory 11 - SPI
S2.6	OFF	CFG_RGMII_MUX_EN_B	RGMII interface demultiplexer select 0 -> RGMII enabled (default) 1 -> SAI2 enabled (through Arduino)

Table continues on the next page...

Table 5. Switch settings of LS1012ARDB (continued)

Feature	Settings (ON=1,OFF=0)	Option	Description
S2.7	OFF	CFG_MUX_QSPI_S0/ CFG_MUX_QSPI_S1	QSPI chip-select demultiplexer select CFG_MUX_QSPI_S[1:0] 00 - CS routed to SPI memory bank 1 (default) 01 - CS routed to SPI memory bank 2 10 - CS routed to Emulator 11 - Invalid : Never use this option as it might cause bus contention.

1.6.2.2 Building the SD card and QSPI images for LS1012ARDB

Select the following options to compile the SD card image and QSPI image that can be programmed into QSPI flash:

```
$ cd lede-source
$ make menuconfig
```

Configure the settings as shown below:

```
Target System (NXP Layerscape) --->
                   (X) NXP Layerscape
Subtarget (layerscape 64b boards) --->
                   (X) layerscape 64b boards
Target Profile (ls1012ardb-64b) --->
                   (X) ls1012ardb-64b
Target Images --->
                  [*] ext4 --->
                  [ ] squashfs ----
                  [ ] GZip images
                  (20) Boot (SD Card) filesystem partition size
                   (This FAT_fs partition is stored for kernel and DTB)
                  (256) Root filesystem partition size (in MB)
                   (This ext4_fs partition is stored for rootfs)
[*] Advanced configuration options (for developers) --->
       [*] Toolchain Options --->
        C Library implementation (Use glibc) --->
      Firmware --->
                 <*> rcw-layerscape-ls1012ardb
      Global build settings --->
      Binary stripping method (none) --->

$ make -j1 V=s
```

After building, two images for LS1012ARDB are generated:

- `bin/targets/layerscape/64b-glibc/lede-layerscape-64b-ls1012ardb-ext4-firmware.bin.` :This is an **SD card image**, which includes kernel and file system that need to be written to the SD card.
- `bin/targets/layerscape/64b-glibc/ls1012ardb-64b-rcw-uboot.bin`: This is **a QSPI flash image** that includes RCW and U-Boot that need to be written to the QSPI flash memory.

1.6.2.3 Deploying images for LS1012ARDB board

This section describes the steps required for deploying images for the LS1012ARDB board.

Steps for updating QSPI image

1. LS1012ARDB uses QSPI NOR flash boot. It has two QSPI NOR flash banks. Use the following code to update RCW and U-Boot after starting U-Boot of bank0:

```
=>i2c mw 0x24 0x7 0xfc; i2c mw 0x24 0x3 0xf5
=>sf probe 0:0
=>tftp 84000000 tftpboot/ ls1012ardb-64b-rcw-uboot.bin
=>sf erase 0 +0x300000
=>sf write 84000000 0 $filesize
=>reset
```

Steps for updating the SD card image

1. Use the below command to program the SD card image, `lede-layerscape-64b-ls1012ardb-ext4-firmware.bin` into an SD card directly:

```
$dd if= bin/targets/layerscape/64b-glibc/lede-layerscape-64b-ls1012ardb-ext4-firmware.bin
of=/dev/mmcblk0
```

2. Once deployment is complete, plug the SD card into LS1012ARDB board and boot the board.

1.6.3 LS1043ARDB platform

This section describes the switch settings and procedure for building and deploying SD card images for the LS1043ARDB platform.

1.6.3.1 Switch settings for LS1043ARDB board

The following table lists and describes the switch configuration for booting up the LS1043ARDB board from SD card.

Table 6. Switch settings for LS1043ARDB

Feature	Settings (OFF=0,ON=1)	Description
SW4[1-8] + SW5[1]	00100000_0	

1.6.3.2 Building images

Use the below commands for building images for the LS1043ARDB platform.

```
$ cd lede-source
$ make menuconfig
```

Configure the settings as shown below:

```
Target System (NXP Layerscape)  --->
    (X) NXP Layerscape
Subtarget (layerscape 64b boards)  --->
    (X) layerscape 64b boards
Target Profile (ls1043ardb-64b)  --->
    (X) ls1043ardb-64b
Target Images  --->
```

```
    [*] ext4  --->
       [ ] squashfs  ----
       [ ] GZip images
       (20) Boot (SD Card) filesystem partition size
             (This FAT_fs partition is stored for kernel and DTB)
       (256) Root filesystem partition size (in MB)
          (This ext4_fs partition is stored for rootfs)
[*] Advanced configuration options (for developers)  --->
        [*]   Toolchain Options  --->
            C Library implementation (Use glibc)  --->

$ make -j1 V=s
```

After building, an SD card image for LS1043ARDB is generated at the following location:

`bin/targets/layerscape/64b-glibc/`

1.6.3.3 Deploying images

Use the below command to program the SD card image for LS1043ARDB into an SD card directly:

```
$dd if= bin/targets/layerscape/64b-glibc/lede-layerscape-64b-ls1043ardb-ext4-firmware.bin of=/dev/
mmcblk0
```

Once deployment is complete, plug the SD card into LS1043ARDB board and boot the board.

1.6.4 LS1046ARDB platform

This section describes the switch settings and procedure for building and deploying SD card images for the LS1046ARDB platform.

1.6.4.1 Switch settings

The following table lists and describes the switch configuration for booting up the LS1046ARDB board from the SD card.

Table 7. Switch settings for LS1046ARDB board

Feature	Settings (OFF=0,ON=1)	Description
SW5[1-8] +SW4[1]	00100000_0	

1.6.4.2 Building the image

Use the following commands to build the SD card images:

```
$ cd lede-source
$ make menuconfig
```

Configure the settings as shown below:

```
Target System (NXP Layerscape) --->
                      (X) NXP Layerscape
Subtarget (layerscape 64b boards) --->
                      (X) layerscape 64b boards
Target Profile (ls1046ardb-64b) --->
                      (X) ls1046ardb-64b
```

```
Target Images --->
                        [*] ext4 --->
                        [ ] squashfs ----
                        [ ] GZip images
                        (20) Boot (SD Card) filesystem partition size
                              (This FAT_fs partition is stored for kernel and DTB)
                        (256) Root filesystem partition size (in MB)
                              (This ext4_fs partition is stored for rootfs)
 [*] Advanced configuration options (for developers) --->
          [*] Toolchain Options --->
               C Library implementation (Use glibc) --->
$ make -j1 V=s
```

After building, the flash image for LS1046ARDB is generated at the following location:

`bin/targets/layerscape/64b-glibc/`

1.6.4.3 Deploying images for LS1046ARDB board

The SD card image for LS1046ARDB can be programmed into a SD card directly. Use the code:

```
$dd if= bin/targets/layerscape/64b-glibc/lede-layerscape-64b-ls1046ardb-ext4-firmware.bin of=/dev/
mmcblk0
```

Then, plug the SD card into LS1046ARDB board and start the board.

1.6.5 FRDM-KW41Z

FRDM-KW41Z is a development kit enabled by the Kinetis® W series KW41Z/31Z/21Z (KW41Z) family built on the ARM® Cortex®-M0+ processor with integrated 2.4 GHz transceiver supporting Bluetooth® Smart/Bluetooth® Low Energy (BLE) v4.2, Generic FSK, IEEE® 802.15.4, and Thread.

The FRDM-KW41Z kit contains two Freedom boards that can be used as a development board or a shield to connect to a host processor. Explore more out-of-box demos and download software and tools on:

www.nxp.com/FRDM-KW41Z/startnow

The following sections describe the process for downloading firmware for the LS1021-IoT and the LS1012ARDB platforms.

1.6.5.1 Downloading firmware

Firmware for the FRDM-KW41Z development kit can be downloaded from the `IoT-gateway-images-v1.0.tar.bz2`. Following is the list of KW41Z firmware that can be used for different use cases.

1. `ble_shell_frdmkw41z.bin`: used by end device for BLE demo.
2. `LS1012aRDB_hci_black_box.bin`: used by LS1012ARDB for BLE demo.
3. `LS1021aIOT_hci_black_box.bin`: used by LS1021A-IoT for BLE demo.
4. `LS1012ARDB_host_controlled_device.bin`: used by LS1012ARDB for Thread demo.
5. `LS1012ARDB_router_eligible_device.bin`: used by end device for Thread demo.
6. `LS1021aIOT_host_controlled_device.srec`: used by LS10121A-IoT for Thread demo.
7. `LS1021aIOT_end_device.srec`: used by end device for Thread demo.

1.6.5.1.1 LS1021A-IoT platform

Use the micro-USB interface to connect the KW41Z Freedom Board to the host PC. Following this, the KW41Z board is recognized as a new storage device.

You can now copy the firmware to this storage device, after which the firmware is downloaded into the board.

1.6.5.1.2 LS1012ARDB platform

Using Jlink to load the firmware into KW41Z on LS1012ARDB.

1. Make sure that you install the latest Jlink driver from the link provided below:

   https://www.segger.com/jlink-software.html

2. Connect Jlink to the SWD interface of LS1012ARDB.

Figure 1. Connecting Jlink to LS1012ARDB

3. Use the Jlink commander from the Start menu (created after installing Jlink driver) to download firmware into KW41Z on LS1012ARDB. The figure below lists the commands for flashing the image on the microcontroller.

```
Type "connect" to establish a target connection, '?' for help
J-Link>connect
Please specify device / core. <Default>: MKW41Z512XXX4
Type '?' for selection dialog
Device>MKW41Z512XXX4
Please specify target interface:
  J) JTAG (Default)
  S) SWD
TIF>S
Specify target interface speed [kHz]. <Default>: 4000 kHz
Speed>4000
Device "MKW41Z512XXX4" selected.


Found SWD-DP with ID 0x0BC11477
Found SWD-DP with ID 0x0BC11477
AP-IDR: 0x04770031, Type: AHB-AP
AHB-AP ROM: 0xF0002000 (Base addr. of first ROM table)
Found Cortex-M0 r0p1, Little endian.
FPUnit: 2 code (BP) slots and 0 literal slots
CoreSight components:
ROMTbl 0 @ F0002000
ROMTbl 0 [0]: FFFFE000, CID: B105900D, PID: 001BB932 MTB-M0+
ROMTbl 0 [1]: FFFFF000, CID: B105900D, PID: 0008E000 MTBDWT
ROMTbl 0 [2]: F00FD000, CID: B105100D, PID: 000BB4C0 ROM Table
ROMTbl 1 @ E00FF000
ROMTbl 1 [0]: FFF0F000, CID: B105E00D, PID: 000BB008 SCS
ROMTbl 1 [1]: FFF02000, CID: B105E00D, PID: 000BB00A DWT
ROMTbl 1 [2]: FFF03000, CID: B105E00D, PID: 000BB00B FPB
Cortex-M0 identified.
J-Link>loadbin hci_black_box.bin 0
```

Figure 2. Use Jlink to flash firmware on KW41Z

Chapter 2
BLE demo

This section describes in brief, the Bluetooth demo, hardware requirements to setup the demo for the LS1021A-IoT and the LS1012ARDB boards, the LEDE configuration used, and steps for testing the demo features on both these boards.

2.1 Introduction

The Bluetooth specification has a very well defined interface between the Controller and the Host called the HCI (Host Controller Interface). This interface is defined for and can be used with various transport layers including an asynchronous serial transport layer.

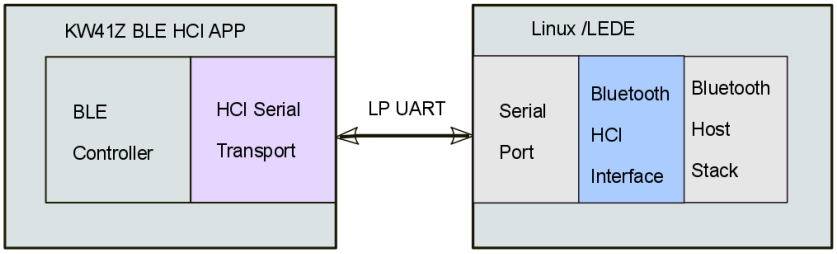

Figure 3. BLE HCI structure

2.2 Hardware requirements for LS1021A-IoT platform

There are two FRDM-KW41Z Freedom boards needed, one is used as a shield to connect to LS1021A-IoT board, another is used as an end device to test BLE connection setup.

Download firmware into FRDM-KW41Z Freedom boards and use as follows:

1. On FRDM-KW41Z board A, which connects to LS1021A-IoT, perform the following steps:
   - Download the firmware named as **LS1021aIOT_hci_black_box.bin** on this board.
   - Plug it in the Arduino interface of LS1021A-IoT.
   - Jump J30 and J31 as shown in the figure below.

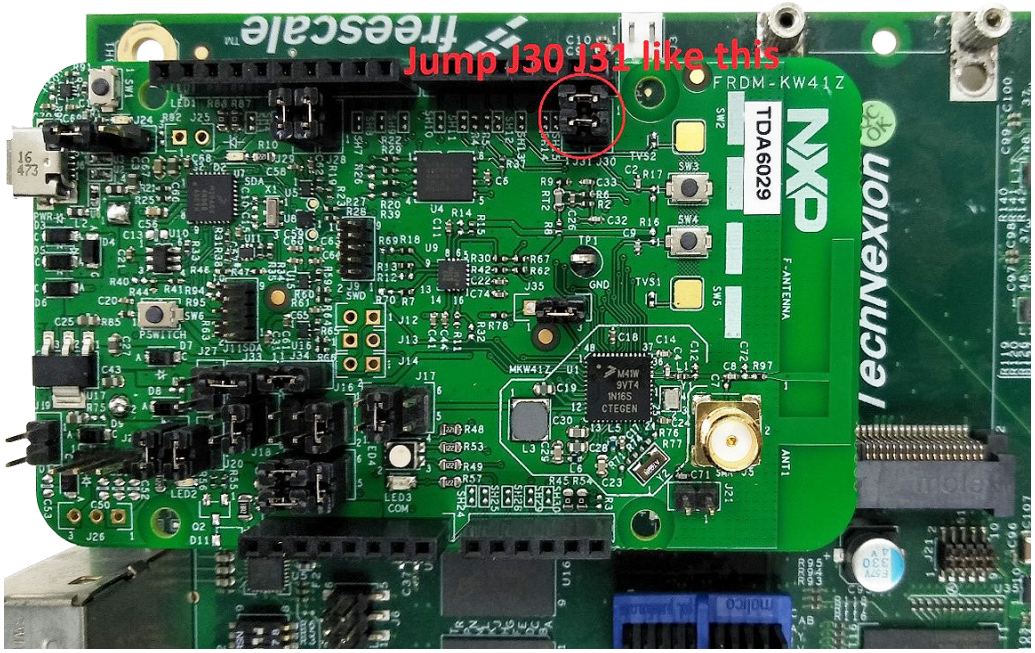

Figure 4. Jumper settings on FRDM-KW41Z board A

2. On FRDM-KW41Z board B, which is used as end device, perform the following steps:
   a. Download firmware named as **ble_shell_frdmkw41z.bin** on this board.
   b. Connect it to the host PC by using a micro-USB interface.
   c. Open a serial terminal to connect FRDM-KW41Z board to open SDA.

   The board start log is displayed in the following figure.

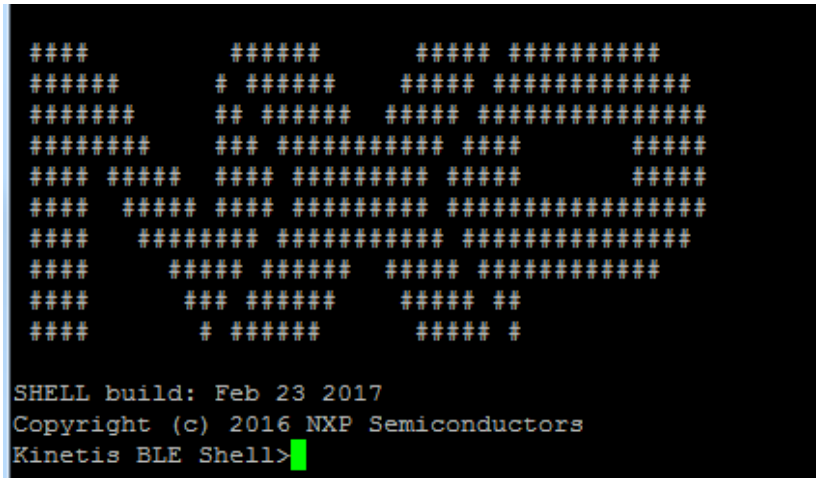

Figure 5. BLE shell start log on FRDM-KW41Z board B

2.3 Hardware requirements for LS1012ARDB platform

LS1012ARDB has a KW41Z device on the board, which is connected by a dual-channel high performance UART bridge. The UART bridge is a SPI slave device and the LS1012A SPI bus is muxed with LS1012A SDHC1 IOs. Hence, muxes on the board are used for selecting the available options.

Configure the SW2 switch as described in the following table.

Table 8. BLE switch settings of LS1012ARDB

Switch	Option	ON/OFF
SW2 [1]	CFG_MUX_SDHC2_S0	ON
SW2 [2]	CFG_MUX_SDHC2_S1	ON

Use Jlink to flash this firmware on KW41 of LS1012ARDB board: **LS1012aRDB_hci_black_box.bin**

Hardware requirements for the FRDM-KW41Z board B, which is used as the end device, are the same as that for the LS1021A-IoT platform. Refer Hardware requirements for LS1021A-IoT platform on page 16.

2.4 LEDE configuration

LEDE configuration for the BLE demo can be done using the command below:

```
$make menuconfig
```

Then, use the following configuration settings:

```
Utilities --->
      <*> bluez-utils
```

2.5 Testing BLE connectivity

This section describes the steps for testing BLE connectivity for the LS1021A-IoT and the LS1012ARDB boards.

2.5.1 LS1021A-IoT platform

After LS1021A-IoT board startup, follow the below instructions to connect LS1021A-IoT to KW41Z board B via the BLE protocol.

1. Reset KW41Z board A by pressing SW1 on KW41Z board A.
2. Connect the Linux Bluetooth stack to a serial HCI interface.

```
$hciattach /dev/ttyLP0 any 115200 noflow nosleep
Device setup complete
```

3. Start the hci0 interface manually.

```
$ hciconfig hci0 up
```

4. Run the hciconfig command with no parameters to check the HCI interface.

```
$ hciconfig
    hci0:   Type: BR/EDR  Bus: UART
        BD Address: 00:04:9F:00:00:15  ACL MTU: 500:20  SCO MTU: 0:0
        UP RUNNING
        RX bytes:462 acl:0 sco:0 events:32 errors:0
        TX bytes:240 acl:0 sco:0 commands:32 errors:0
```

5. Start advertising BLE on KW41Z board B.

```
Kinetis BLE Shell> gap advstart
```

6. Scan for advertising LE devices.

```
$ hcitool -i hci0 lescan
    LE Scan ...
    00:04:9F:00:00:16 (unknown)
```

7. Obtain remote LE Device (KW41Z board B) information.

```
$ hcitool -i hci0 leinfo 00:04:9F:00:00:16
    Requesting information ...
            Handle: 705 (0x02c1)
            LMP Version: 4.2 (0x8) LMP Subversion: 0x121
            Manufacturer: NXP Semiconductors (formerly Philips Semiconductors) (37)
            Features: 0xff 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

8. If KW41Z board B has been disconnected, you need to restart advertising BLE on KW41Z board B.

```
Kinetis BLE Shell> gap advstart
```

9. Scan for advertising LE devices to ensure advertising starts.

```
$ hcitool -i hci0 lescan
```

10. Connect to a remote LE device (KW41Z board B).

```
# gatttool -I
[                ][LE]> connect 00:04:9F:00:00:16
Attempting to connect to 00:04:9F:00:00:16
Connection successful
```

11. After disconnecting from the BLE network, if you want to connect to it again, perform the instructions starting from step 5.

2.5.2 LS1012ARDB platform

After LS1012ARDB board startup, use the following commands to connect the Linux Bluetooth stack to a serial HCI interface:

```
$ hciattach /dev/ttySC1 any 9600 noflow nosleep
Device setup complete
```

Then, perform the steps as described in LS1021A-IoT platform on page 18.

Chapter 3
Thread demo

This section describes the Thread demo, its hardware requirements, LEDE configuration, and steps to test the thread connectivity for the LS1021A-IoT and the LS1012ARDB boards.

3.1 Introduction

Kinetis Thread Stack provides Thread networking components over IEEE-802.15.4 MAC 2006 layer running on Kinetis MCUs which are enabled to use IEEE 802.15.4. It provides a CoAP application profile API.

The Kinetis Thread stack implements a Serial Tunnel Media interface that can be used to exchange FSCI encapsulated IPv6 packets with a host system. To provide connectivity to the host, two components are needed: the TUN/TAP kernel module, which allows the operating system to create virtual interfaces and a program that knows how to encapsulate/decapsulate IP packets to/from FSCI/THCI.

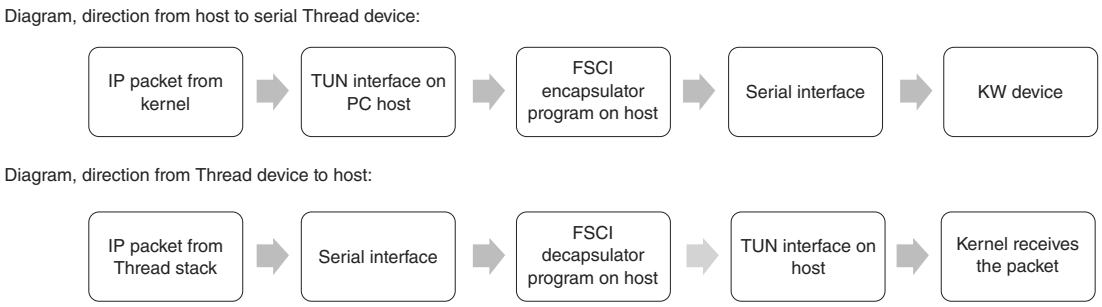

Figure 6. Thread stack structure

3.2 Hardware requirements for LS1021A-IoT platform

This demo has hardware requirements similar to the BLE demo. Refer Hardware requirements for LS1021A-IoT platform on page 16.

Then, download the following firmware onto the FRDM-KW41Z boards:

- On FRDM-KW41Z board A, which connects to LS1021A-IoT: download **LS1021aIOT_host_controlled_device.srec**.
- On FRDM-KW41Z board B, which is used as end device: download **LS1021aIOT_end_device.srec**.

The Board start log is as follows:

```
SHELL starting...
NXP Thread v1.1.0.7
Enter 'thr join" to join a network, or "thr create" to start new network
Enter "help" for other commands
$
```

3.3 Hardware requirements for LS1012ARDB platform

This demo has hardware requirements similar to the BLE demo for LS1021A-IoT platform. Refer Hardware requirements for LS1021A-IoT platform on page 16.

Then, download the following firmware onto the two FRDM-KW41Z boards:

1. Use Jlink to flash the firmware on FRDM-KW41Z board, connected to the LS1012ARDB board:

   LS1012ARDB_host_controlled_device.bin

2. Download the following firmware into the FRDM-KW41Z board B that is used as an end device:

   LS1012ARDB_router_eligible_device.bin

   .

3.4 LEDE configuration

For LEDE configuration, use the command below:

```
$make menuconfig
```

Configure the following settings:

```
Network --->
     Routing and Redirection --->
                <*> ip-full
Utilities --->
      <*> hsdk
```

3.5 Testing thread connectivity

This section describes the steps for testing thread connectivity for the LS1021A-IoT and the LS1012ARDB platform.

3.5.1 LS1021A-IoT platform

After LS1021A-IoT board start-up, perform the following instructions to connect LS1021A-IoT to the KW41Z board B via the Thread protocol.

1. Create a TUN/TAP interface and add IP route tables.

   ```
   $ make_tun.sh
   ```

   This script automates the creation of the TUN interface, while configuring it with the proper IPv6 addresses and routes.

   ```
   $ cat make_tun.sh
   #!/bin/bash
   # Create a new TUN interface for Thread interaction.
   ip -6 tuntap add mode tun fslthr0
   # Assign it a global IPv6 address.
   ip -6 addr add FD01::2 dev fslthr0
   # Add route to default address of Serial TUN embedded interface.
   ```

```
ip -6 route add FD01::1 dev fslthr0
# Add route to Unique Local /64 Prefix via fslthr0.
ip -6 route add FD01:0000:0000:3EAD::/64 dev fslthr0
# The interface is ready.
ip link set fslthr0 up
# Enable IPv6 routing on host.
sysctl -w net.ipv6.conf.all.forwarding=1
# The interface is ready.
ip link set fslthr0 up
```

2. Reset KW41Z board A by pressing SW1 on KW41Z board A.
3. Create a thread network on LS1021A-IoT.

```
$Thread_KW_Tun /dev/ttyLP0 fslthr0 1 15 115200 &
[  170.543616] IPv6: ADDRCONF(NETDEV_CHANGE): fslthr0: link becomes ready
WARNING: Cannot open /usr/share/hsdk/hsdk.conf => FSCI ACKs are disabled
[THR] Factory Reset                 OK!
[THR] Create Network                OK!
[THR] Border Router Add Prefix      OK!
[THR] Border Router Sync Prefix     OK!
[MESHCOP] Start Commissioner        OK!
[MESHCOP] Add Expected Joiner       OK!
[MESHCOP] Sync Steering Data        OK!
```

4. Join thread network on FRDM-KW41Z board B. Do the following after add expected joiner is ok on LS1021a-IOT board.

```
$ thr join
Joining a Thread network...
Commissioning successful
```

5. Test thread connectivity by ping tun/tap interface IP of LS1021A from FRDM-KW41Z board B.

```
$ping FD01::2
$ifconfig
Interface 0: 6LoWPAN
        Mesh local address (ML64): fdc8:9eea:ea7c::8ed:6664:78bf:826f
        Mesh local address (ML16): fdc8:9eea:ea7c::ff:fe00:400
        Unique local address: fd01::3ead:b0f0:e421:57e9:562a
```

6. Ping the remote device IP(use the Unique local address) from LS1021a.

```
$ping fd01::3ead:b0f0:e421:57e9:562a
```

3.5.2 LS1012ARDB platform

After the LS1012ARDB board start-up, perform the same steps as for testing the LS1021A-IoT thread demo. Refer LS1021A-IoT platform on page 21.

The only difference is in using the Thread_KW_Tun command to create a thread network on the LS1012ARDB board. Use the command specified below instead of the Step 3 described in the referred section:

```
$Thread_KW_Tun /dev/ttySC1 fslthr0 1 15 9600 &
```

Chapter 4
NFC demo

This section describes the NFC demo, hardware requirements to setup this demo for the LS1021A-IoT and the LS1012ARDB boards, the LEDE configuration to be used, and steps for testing the demo features.

4.1 Introduction

The PN7150 NFC module package comes with the following items:

1. PN7150 module
2. Arduino Shield Interface board (OM5578)
3. NFC sample tag

Figure 7. NFC module package

LS1021A-IoT and LS1012ARDB platforms use the OM5578 / PN7150ARD module, which connects to the I2C-0 bus as a NFC module through the Arduino interface. The following figures depict the required hardware connection signals between PN7150 NFC module and LS1021A-IoT or LS1012ARDB boards.

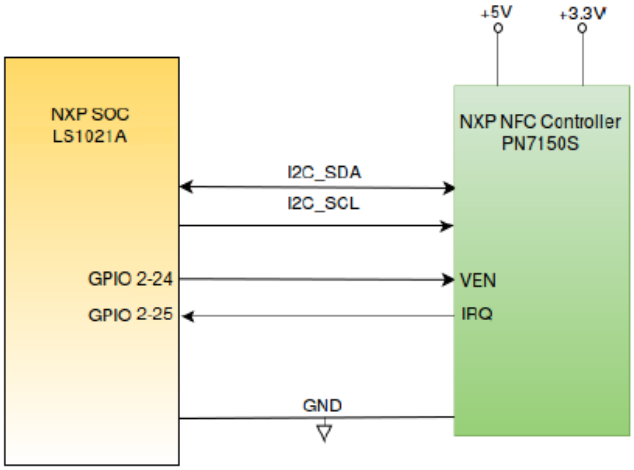

Figure 8. NFC connections to the LS1021A-IoT board

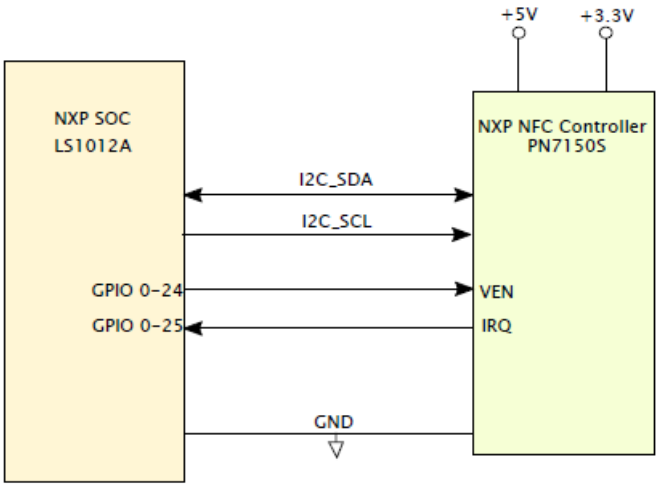

Figure 9. NFC connections to the LS1012ARDB board

4.2 Hardware requirements for LS1021A-IoT platform

The LS1021A-IoT board does not connect GPIO pins to the Arduino interface. Therefore, you need to connect two GPIO pins to the Arduino interface, as shown in the figure below. (Link: J8.1->J502.3, J17.8->J502.5).

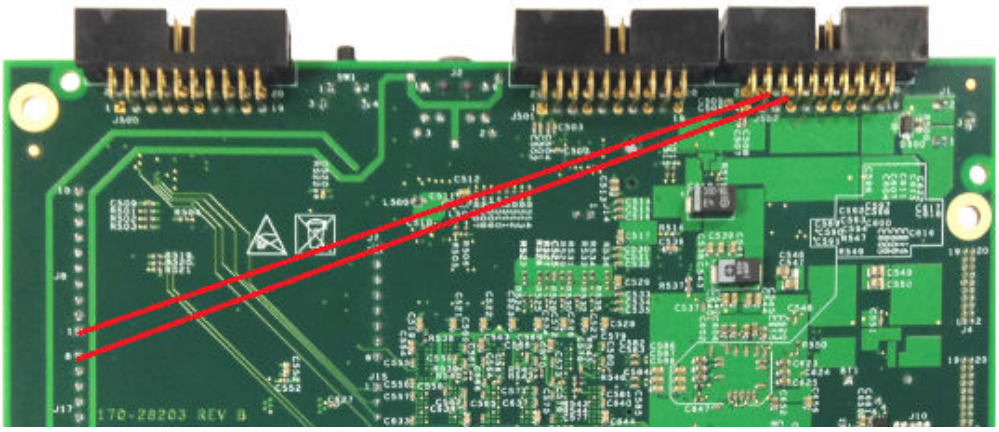

Figure 10. NFC hardware connection on LS1021A-IoT

4.3 Hardware requirements for LS1012ARDB platform

The LS1012ARDB board supports the NFC feature. In order to ensure the NFC feature works correctly on this board, you need to rework the hardware, and short circuit jumpers 16.2 and J17.14, as shown in the figure below.

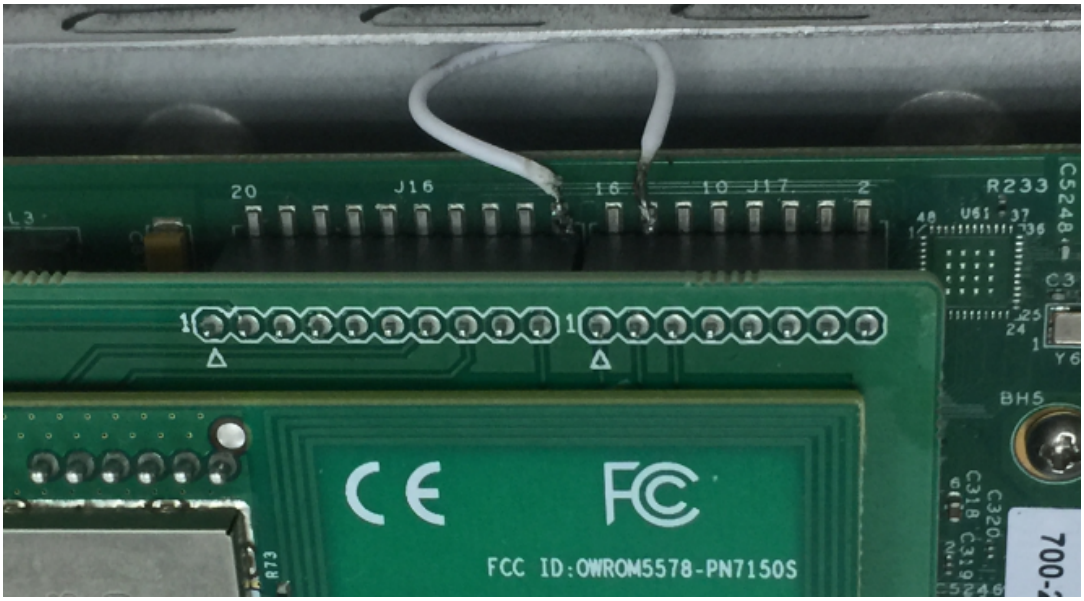

Figure 11. NFC hardware rework on LS1012ARDB board

NOTE

In order to support NFC on LS1012ARDB Rev D board, you need to configure the SW2 switch as: SW2[1-2]: ON/ON before power on, then set the SW2 switch as: SW2[1-2]: ON/OFF after powering on the board.

4.4 LEDE configuration

For the LS1021A-IoT platform, configure the following:

```
$make menuconfig
 Kernel modules --->
   Other modules --->
           <*> kmod-nxp_pn5xx
 Utilities --->
           <*> libnfc-nci
```

For the LS1012ARDB platform, configure the following:

```
$make menuconfig
 Kernel modules --->
    Other modules --->
          <*> kmod-nxp_pn5xx
 Utilities --->
          <*> libnfc-nci
```

NOTE

Using the NFC module on LS1012ARDB changes the RCW to modify mux_GPIO pins. You need to clean the project and rebuild it in order to generate images.

4.5 Testing NFC feature

Make sure the U-Boot identifies the NFC module before starting up the Linux Kernel. For this, use the below command at the U-Boot prompt:

```
=> i2c probe
=> Valid chip addresses: 00 08 09 1E 20 24 25 26 28 40 7C
```

The I2C address '0x28' is for the OM5578/PN7150 module.

NOTE

In case the module is not identified, you need to re-install the hardware and repeat the above mentioned command.

4.5.1 LS1021A-IoT platform

After the LS1021A-IoT board startup, run the command below to start the NFC demo.

```
$ ./data/nfc/nfcDemoApp poll
```

4.5.2 LS1012ARDB platform

Make sure the SW2[1:2] = on:on.

Make sure the below mentioned power sequences are implemented:

1. Connect USB/serial port “CONSOLE” to the PC's USB port.
2. Connect the supplied AC power to the LS1012ARDB board.

4.5.3 Logs

When the application is running, a print log is displayed, as shown in the following figure.

```
root@LEDE:/# nfcDemoApp poll
################################################################################
#########
##                                      NFC demo
       ##
################################################################################
#########
##                                  Poll mo[   23.182571] pn54x_dev_open : 10,58
de activated   [   23.186872] pn54x_dev_ioctl, cmd=1074063617, arg=1
               [   23.192908] pn544_enable power on
               ##
################################################################################
#########
                              ... press enter to quit ...

[   23.413046] pn54x_dev_ioctl, cmd=1074063617, arg=0
[   23.417809] pn544_disable power off
[   23.633045] pn54x_dev_ioctl, cmd=1074063617, arg=1
[   23.637807] pn544_enable power on
BrcmHcpX8103
BrcmHcpR8180
BrcmHcpX810103020304
BrcmHcpR8180
BrcmHcpX8101010194fffffe93ffff
BrcmHcpR8180
BrcmHcpX810204
BrcmHcpR818000
Waiting for a Tag/Device...
```

Figure 12. NFC demo log

The application print log generated after touching a NFC sample tag to the OM5578/PN7150 NFC reader is shown in the following figure.

```
Waiting for a Tag/Device...

       NFC Tag Found

              Type :          'Type A - Mifare Ul'
              NFCID1 :        '04 DC 58 D2 9C 39 80 '
              Record Found :
                              NDEF Content Max size :         '868 bytes'
                              NDEF Actual Content size :      '29 bytes'
                              ReadOnly :                      'FALSE'
                              Type :                          'URI'
                              URI :                           'http://www.nxp.com/demoboard/OM5578'


              29 bytes of NDEF data received :
              D1 01 19 55 01 6E 78 70 2E 63 6F 6D 2F 64 65 6D 6F 62 6F 61 72 64 2F 4F 4D 35 35 37 38

       NFC Tag Lost

Waiting for a Tag/Device...
```

Figure 13. NFC demo log after recognizing sample tag

Chapter 5
Wifi demo

This section outlines the Wifi demo process, hardware requirements, the LEDE configuration, and steps for testing the demo features on the LS1021A-IoT, LS1012ARDB, LS1043ARDB, and LS1046ARDB boards.

5.1 Introduction

A WNC DNXA-H1 card is used for the WiFi verification and the corresponding kernel driver ATH9K is also set as the default. This demo can be used on the LS1021A-IoT, LS1012ARDB, LS1043ARDB, and the LS1046ARDB boards. The LS1021A-IoT board has two mini-PCIe slots. Hence, it allows a maximum of two WiFi cards to be inserted.

NOTE

In case you want to use only one Wi-fi card on the LS1021A-IoT board, you should use the MPCIE slot1. The MPCIE slot2 cannot be used if there is only one WIFI card. See Known issues on page 45.

5.2 Hardware requirements

Insert a WNC DNXA-H1 card as the wi-fi card into mPCIE slot of LS1021A-IoT board as shown in the following figure.

Figure 14. Wi-fi card on LS1021AIOT

5.3 LEDE configuration

For LS1021AIOT board, use the following settings for the LEDE configuration:

```
$make menuconfig
#Set configures as following:
Libraries  --->
         <*>libnfnetlink
         <*>libnftnl
         <*>libnl
Network  --->
         <*>hostapd
         -*- hostapd-common
         <*>hostapd-utils
Kernel modules  --->
   Wireless Drivers  --->
            <*> kmod-ath9k
Base system  --->
            <*>dnsmasq
Luci--->
   Collection  --->
         <*>luci
```

Use the following settings for the LS1012ARDB, LS1043ARDB, and LS1046ARDB boards.

Kernel configuration

```
$make kernel_menuconfig
[*] Networking support --->
         -*- Wireless --->
               [*] Wireless extensions
               [*] WEXT_SPY
               [*] WEXT_PRIV
               <*> cfg80211 - wireless configuration API
               [*] enable powersave by default
               <*> Generic IEEE 802.11 Networking Stack (mac80211)
               [*] Minstrel
               [*] Minstrel 802.11n support
               -*- Enable LED triggers
Device Drivers --->
      [*] Network device support --->
         [*] Wireless LAN --->
               <*> Atheros Wireless Cards --->
                  <*> Atheros 802.11n wireless cards support
                  [*] Atheros bluetooth coexistence support
                  [*] Atheros ath9k PCI/PCIe bus support
                  [*] Atheros ath9k support for PC OEM cards
```

LEDE configuration

```
$make menuconfig
Libraries --->
             <*> libnfnetlink
             <*> libnftnl
             <*> libnl
Network --->
             <*> hostapd
```

```
             -*- hostapd-common
             <*> hostapd-utils
Kernel modules --->
            Wireless Drivers --->
                <*> kmod-ath9k
Base system --->
           <*> dnsmasq
Luci--->
           Collection --->
                <*> luci
```

5.4 Testing WiFi demo

To set WiFi, perform the following instructions while starting up the board for the first time.

1. Generate wifi config file.

```
$wifi config
```

2. Modify wifi config file to delete disable option.

```
$vi /etc/config/wireless
#option disabled '1'
```

3. Set eth1 which is connected to network as wan.

```
$vi /etc/config/network
        config interface 'lan'
                option type 'bridge'
                option ifname 'eth0'
                option proto 'static'
                option ipaddr '192.168.1.1'
                option netmask '255.255.255.0'
                option ip6assign '60'

        config interface 'wan'
                option ifname 'eth1'
                option proto 'dhcp'
```

4. Start wifi up.

```
$ wifi up
[ 317.647622] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 317.655381] device wlan0 entered promiscuous mode
[ 317.669190] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 317.675638] br-lan: port 2(wlan0) entered forwarding state
[ 317.681116] br-lan: port 2(wlan0) entered forwarding state
[ 317.686689] IPv6: ADDRCONF(NETDEV_CHANGE): br-lan: link becomes ready
[ 319.672689] br-lan: port 2(wlan0) entered forwarding state
[ 327.782870] device wlan0 left promiscuous mode
[ 327.787420] br-lan: port 2(wlan0) entered disabled state
[ 328.008825] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 328.017056] device wlan0 entered promiscuous mode
[ 328.021790] br-lan: port 2(wlan0) entered forwarding state
[ 328.027288] br-lan: port 2(wlan0) entered forwarding state
```

```
[ 328.042613] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 330.022688] br-lan: port 2(wlan0) entered forwarding state
```

5. Use the SSID of the Wi-Fi as 'LEDE' to connect the Wi-Fi to your mobile phone.

Chapter 6
Docker demo

This section describes the Docker demo, hardware requirements to setup this demo for NXP's LS1043ARDB and LS1046ARDB boards, the LEDE configuration to be used, and a test case for testing the demo features.

6.1 Introduction

Docker is a tool that makes it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, an application can run on any other Linux machine regardless of any customized settings that machine might, which could differ from the machine actually used for writing and testing the code.

The following sections describe the steps to run Docker on LEDE file system environment using NXP's LS1043ARDB and LS1046ARDB platforms with 64B images.

NOTE

As the LEDE does not include the Docker package, therefore, this demo uses Docker binaries built standalone in 64B and copies them into the LEDE filesystem using the patch:

`0001-lede-docker-Add-docker-support-for-ls1043ardb-ls1046.patch`

This patch has been applied in the IoT release v0.3 LEDE source code.

6.2 Hardware requirements

Ensure to boot up the LS1043ARDB and the LS1046ARDB boards from the SD card using the software settings listed in the following table.

Table 9. Settings for booting the board

Platform	Boot	Board software setting
LS1043ARDB	SD card	SW4[1-8] +SW5[1] = 0b'00100000_0
LS1046ARDB	SD card	SW5[1-8] +SW4[1] = 0b'00100000_0

6.3 Docker test case

Following are the steps to test the Docker demo.

- Obtain the IoT release v0.3 LEDE source code

```
cd lede-source
./scripts/feeds update -a
./scripts/feeds install -a
```

- Use the lede-1701-docker-config as the LEDE config.

```
cd lede-source
mv lede-1701-docker-config .config
```

- Check the menuconfig.

```
cd lede-source
make menuconfig
# choose the ls1043ardb 64b or ls1046ardb 64b
Utilities  --->
    <*> docker
# exit and save
```

- Compile the LEDE and get the final image.

```
cd lede-source
make -j8 V=s
# get the binary image for ls1043ardb or ls1046ardb at
# bin/targets/layerscape/64b-glibc/
# named 'lede-layerscape-64b-ls1043ardb-ext4-firmware.bin' or
# 'lede-layerscape-64b-ls1046ardb-ext4-firmware.bin'
```

- Program the binary image into a SD card under a host Linux machine.

```
sudo dd if=./lede-layerscape-64b-ls1043ardb-ext4-firmware.bin of=/dev/sdx
# check the SD card name "/dev/sdx" in your machine and replace the # "sdx" in the above
command
```

- Insert the programmed SD card into the LS1043ARDB or LS1046ARDB board and boot it up.
- Set the LEDE system network (following is an example on the LS1043ARDB board).

```
# set the Ethernet interface IP addr
root@LEDE:/# ifconfig eth2 10.192.208.230
# set the default gateway
root@LEDE:/# route add default gw 10.192.208.254
# set the DNS server
root@LEDE:/# vi /etc/resolv.conf
# add the namespace in resolv.conf like:
    nameserver 10.192.130.201
    nameserver 10.228.49.200
    nameserver 10.201.141.100

# save and exit
```

- Run docker and aarch64/hello-world docker image.

```
root@LEDE:/# mkdir docker
root@LEDE:/# docker daemon --graph=/docker&
root@LEDE:/# date -s "xxxxx"  /* set the current time to make certificate valid */
root@LEDE:/# docker pull aarch64/hello-world
root@LEDE:/# docker images
REPOSITORY            TAG                 IMAGE ID            CREATED             SIZE
aarch64/hello-world   latest              70a49d30fa8f        3 months ago        2.088 kB
root@LEDE:/# docker run --net=host aarch64/hello-world
```

Chapter 7
ZigBee scenario

The ZigBee demonstration scenario is based on LS1012ARDB platform and the JN516x-EK004 Evaluation Kit. For the complete description, refer to the *Zigbee Solution on LS1012ARDB for Industrial IoT Quick Start Guide*.

Chapter 8
OTA implementation

This section provides an introduction to OTA use cases, scripts and configuration settings for OTA implementation, LEDE configuration, server preparation, and an OTA test case. OTA is supported by NXP's LS1021-IoT, LS1012ARDB, LS1043ARDB, and LS1046ARDB platforms. It also lists the OTA features supported by each hardware platform.

8.1 Introduction

OTA refers to a method of updating U-Boot, kernel, file system, and even the full firmware to devices through the network. If the updated firmware does not work, the device can rollback the firmware to the latest version automatically.

NOTE

While updating U-Boot, there is no hardware method to rollback the device automatically, hence the device might not be rolled back, once the U-Boot is not working.

- **version.json**: This is a JSON file which saves the board name and version of each firmware. Below is an example of version.json.

```
{
"updatePart":"kernel", /* Name of firmware image which has been updated. */
"updateVersion":"1.0", /* Version of firmware image which has been updated. */
"all":"1.0", /* version of the full firmware image which has been used now */
"u-boot":"1.0", /* version of the u-boot image which has been used now */
"kernel":"1.0", /* version of the kernel image which has been used now */
"filesystem":"1.0", /* version of the filesystem image which has been used now */
"boardname":"ls1021aiot" /* used to get the corresponding firmware from server*/
"URL":"https://www.nxp.com/lgfiles/iiot" /* used to get the corresponding firmware from
server*/
}
```

- **update.json**: This file is stored in server, it saves the name and version of firmware image which will be updated. Below is a sample update.json file:

```
{
"updateStatus":"yes", /* set yes or no to tell devices is it need to update. */
"updatePart":"kernel", /* name of update firmware. */
"updateVersion":"1.0", /* version of update firmware */
}
```

- **ota-update**: This script can get a JSON file named update.json from server, then parse the file and get the new firmware version to confirm whether to download it from server or not. It finally writes the firmware into the SD card instead of the old one. After that, save the "updatePart" and "updateVersion" into version.json, and mark the update status on 4080 block of SD card to let U-Boot know it.
- **ota-versioncheck**: This script checks if the firmware has been updated, then updates the version of the update part in version.json, and cleans the flag of update status on 4080 block of SD card. This script runs automatically each time the system restarts.
- **ota-rollback**: This script runs on the ramdisk filesystem after the filesystem update fails. It gets the old firmware version from the version.json file and then updates it from the server.

8.2 LEDE configuration

Configure the settings as described below:

```
$make menuconfig
Network --->
      File Transfer --->
         <*> wget
Libraries --->
      <*> libustream-openssl
```

Before performing the actual compilation, set the URL of the server path on the `target/linux/layerscape/image/backup/version.json` file as the following:

```
URL":"http://www.nxp.com/lgfiles/iiot/"
```

You can also use the URL of your own server.

8.3 Platform support for OTA demo

The OTA demo is supported by four NXP hardware platforms. Following is the list of features supported by each platform:

1. **LS1021A-IoT**
   - Full SD card firmware update
   - U-Boot image update kernel image update
   - File system image update
   - Full SD card firmware update
2. **LS1012ARDB**
   - Full SD card firmware update
   - RCW and U-Boot image update on QSPI flash
   - Kernel image update and rollback
   - File system image update and rollback
3. **LS1043ARDB**
   - Full SD card firmware update
   - U-Boot image update
   - Kernel image update and rollback
   - File system image update and rollback
4. **LS1046ARDB**
   - Full SD card firmware update
   - U-Boot image update
   - Kernel image update and rollback
   - File system image update and rollback

8.4 Server requirements

This demo provides a sample server to update images for the v1.0 release. In case you want to use another server, you need to change the URL to your own server path at "target/linux/layerscape/image/backup/version.json" such as the following:

```
"URL":"https://www.nxp.com/lgfiles/iiot/"
```

The server must include a JSON file named update.json that can send information to device boards. Below is a sample update.json file.

```
{
    /* set yes or no to tell devices is it need to update. */
        "updateStatus":"yes",

    /* which part to update, you can write "all", "u-boot", "kernel",  "filesystem"  */
        "updatePart":"filesystem",

    /* version of update firmware */
        "updateVersion":"1.0",
}
```

Images for OTA are stored in the path:

<updateVersion>/<boardname>/

where the <boardname> can be one of these: ls1021aiot, ls1012ardb-64b, ls1012ardb-32b, ls1043ardb-64b, ls1043ardb-32b, ls1046ardb-64b, or ls1046ardb-32b.

Images must be named as following:

- u-boot.bin: U-Boot image for update. In ls1012ardb folder, this image includes RCW and U-Boot.
- uImage: kernel image for update
- rootfs.ext4: filesystem image for update
- firmware_sdcard.bin: a full firmware of SD card image.

8.5 OTA test case

1. Plug network cable into Eth1 on the board. This enables the network after the system is running.
2. Update U-Boot using the following steps:
   - Update the .json on server as shown in the following example:

```
 {
                    "updateStatus":"yes",
                   "updatePart":"u-boot",
                    "updateVersion":"1.0",
}
```

   - Upload the u-boot image on server path: 1.0/<boardname>/u-boot.bin
   - Run ota-update command on device board.
3. Updating the file system:

- Set the "updatePart" to "filesystem" in update.json.
- Upload the filesystem image on server path: 1.0/<boardname>/rootfs.ext4
- Run ota-update command on the device board.

4. Updating full firmware
   - Set the "updatePart" to "all" in update.json.
   - Upload the full firmware image on server path: 1.0/<boardname>/firmware_sdcard.bin
   - Run ota-update command on device board.
5. Rollback test:
   - The Kernel and file system can use a wrong image to upload on the server and test update on device.

Chapter 9
4G-LTE modem

The 4G-LTE feature is supported by NXP's LS1021A-IoT, LS1012A-RDB, LS1043A-RDB, and LS1046A-RDB hardware platforms.

This section describes the 4G-LTE feature demo, hardware requirements to setup this demo for the LS1021A-IoT board, the LEDE configuration to be used, and steps for testing the 4G USB modem's link to the internet.

9.1 Introduction

A HuaWei E3372 USB Modem is used for the 4G network verification.

9.2 Hardware requirements

Insert a USB Modem into the USB slot of LS1021A-IoT board.

9.3 LEDE configuration

Use the following configuration settings:

```
$make menuconfig
      Utilities --->
               <*> usb-modeswitch
       Kernel modules --->
               USB Support --->
                     <*> kmod-usb-net
                     <*> kmod-usb-net-cdc-eem
                     <*> kmod-usb-net-cdc-ether
                     <*> kmod-usb-net-cdc-ncm
```

9.4 Testing the 4G USB modem link to the Internet

Perform the following instructions to set up the 4G modem.

1. Set eth3 connected to network.

```
root@LEDE:/# vi /etc/config/network
# add the wan in network as shown below:
config interface 'wan'
        option ifname 'eth3'
        option proto 'dhcp'

# save and exit
```

2. Test the 4G modem link to the internet.

```
 oroot@LEDE:/# ping www.nxp.com
PING www.nxp.com (210.192.117.231): 56 data bytes
64 bytes from 210.192.117.231: seq=0 ttl=52 time=60.223 ms
64 bytes from 210.192.117.231: seq=1 ttl=52 time=95.076 ms
64 bytes from 210.192.117.231: seq=2 ttl=52 time=89.827 ms
64 bytes from 210.192.117.231: seq=3 ttl=52 time=84.694 ms
64 bytes from 210.192.117.231: seq=4 ttl=52 time=68.566 ms
64 bytes from 210.192.117.231: seq=5 ttl=52 time=89.809 ms
```

Chapter 10
Sigfox demo

This section describes the Sigfox demo, hardware requirements to setup this demo for the LS1021A-IoT board, the LEDE configuration to be used, and steps for testing the Sigfox base station.

10.1 Introduction

Sigfox is an operated telecommunication Low-Power Wide-Area (LPWA) public network, dedicated to the Internet of Things. An OL2385 Sigfox device is used on LS1021A-IoT board to connect to the Sigfox network. This section explains how to use OL2385 device on the LS1021A-IoT platform to send messages to the Sigfox backend.

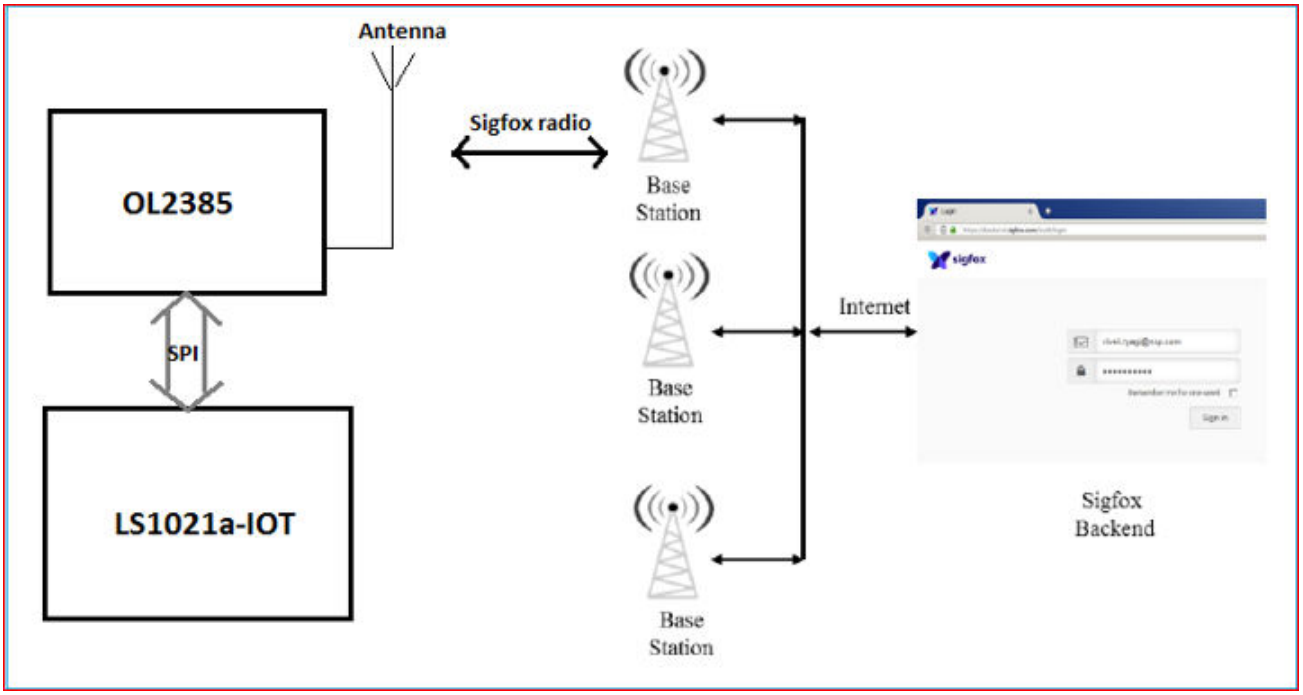

Figure 15. Sigfox demo

10.2 Hardware requirements

The OL2385 needs a wire to respond to ACK, therefore you need to connect a GPIO pin to the Arduino interface (Link: J8.2->J502.9) as shown in the below figure.

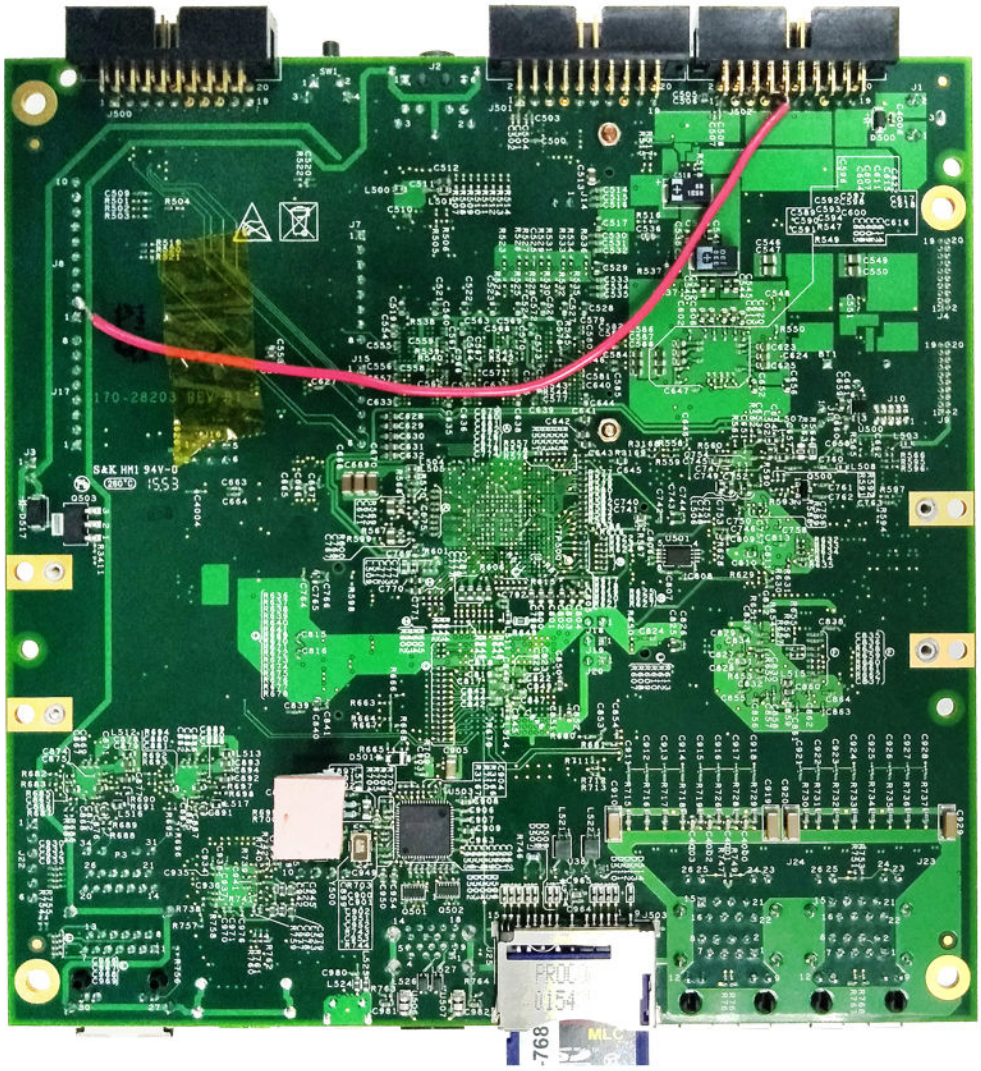

Figure 16. Hardware requirements for Sigfox demo

Then, plug OL2385 into Arduino of LS1021A-IoT board, so that ls1021A-IoT can communicate with OL2385 through the SPI bus. Attach the PCB antenna by snapping the µFL connector on the antenna to the µFL connector on OL2385 as shown in the figure below.

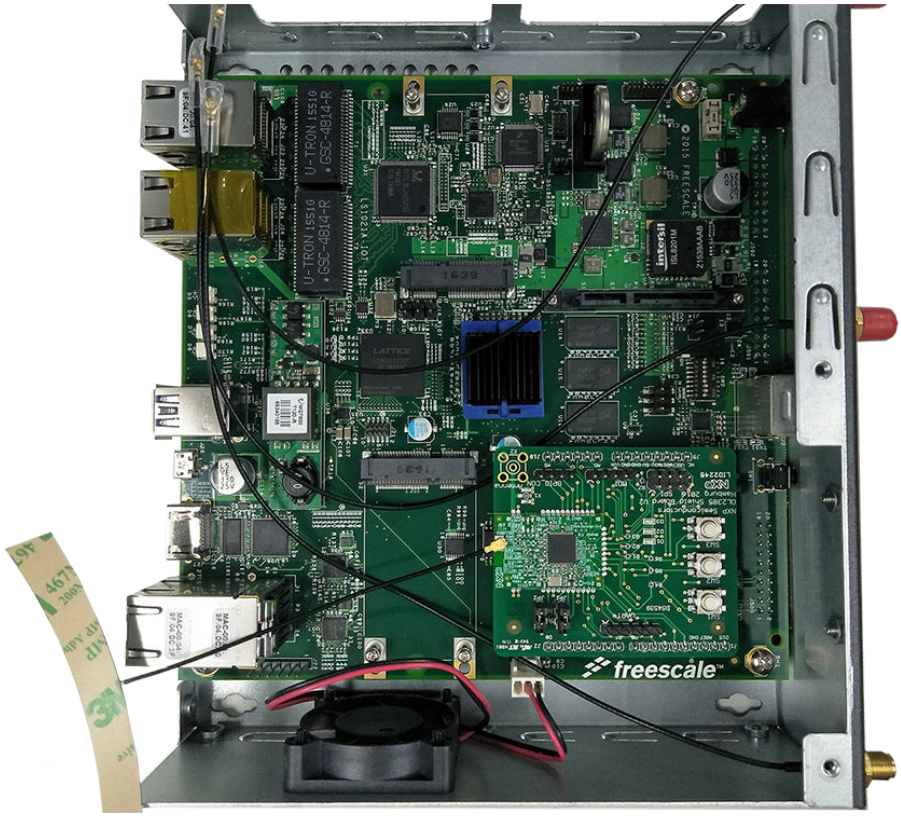

Figure 17. Using the µFL connector for Sigfox demo

10.3 LEDE configuration

Set the configuration as the following:

```
$make menuconfig
 Utilities  --->
           <*> nxp-sigfox
 Kernel modules  --->
   SPI Support  --->
           <*>  kmod-spi-dev
```

10.4 Testing on Sigfox Base Station

Use the following steps for testing the Sigfox base station:

1. **Register the Sigfox device on the network.** Take a note of Device ID and PAC on the back of OL2385. Then, activate the device by registering it on https://backend.sigfox.com/activate. If in case, it issues a warning that the Device Id is invalid, you can write a mail to devrelations@sigfox.com to get it added.
2. **Edit Sigfox configuration**.

   Enter Linux on LS1021A-IoT platform, edit the file, `/etc/sigfox.conf`, select **Network Standard** to an appropriate value:

   a. 1 - RCZ1 ETSI Europe (863 MHz – 870 MHz)

b. 2 – RCZ2 FCC US (902 MHz – 928 MHz)

c. 3 – RCZ3 ARIB Japan, Korea (915 MHz – 930 MHz)

d. 4 – RCZ4 FCC Latin America, Australia, New Zealand (902 MHz – 915 MHz).

3. **Run the command**`sigfox-demo` on LS1021A-IoT platform to send the message to the server.
4. Go to https://backend.sigfox.com/auth/login to log in to your just created account. Click the **Device** tab to see your devices. Click the **Message** tab to view the messages that you have sent before.

Chapter 11
Known issues

The following table lists the known issues for this release.

Table 10. Known issues

S.No	Description
1	The ath9k wifi card used for the Wifi demo might be disabled after rebooting. To make it work you need to power-off the LS1021A-IoT board.
2	The Huawei E3372 4G LTE dongle with PID 157C is not recognized as an Ethernet Interface. Hence, it is not able to connect to the network.
3	For the Wi-fi demo, in case you are using only one Wi-fi card on the LS1021A-IoT board, you should use the MPCIE slot1. The MPCIE slot2 cannot be used if there is only one Wifi card on LS1021A-IoT board.

Chapter 12
Revision history

The table below summarizes revisions to this document.

Table 11. Revision history

Date	Revision	Topic-reference	Change
30.03.2018	1.1	Building and deploying images on NXP platforms on page 6	Added the note for mbed driver information in this section.
22.12.2017	1.0	Sigfox demo on page 41	Added the chapter.
-	0.1	-	Initial release.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

IIoT_UG
Rev. 1.1
03/2018