ARM926EJ-S™ Based

32-bit Microprocessor

NuDesign NK-980IoT
使用手冊

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NUC980 microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 概論 3
2 快速使用 NuDesign NK-980IoT 4
2.1 安裝BSP 4
2.2 編譯 U-Boot 5
2.3 編譯Linux 內核 5
2.4 燒錄Linux Kernel及U-Boot 至 SPI NAND Flash 6
2.5 硬體設置 9
2.6 完成Linux內核開機 9
2.7 連接 WiFi Access Point 11
2.8 連接 NB IoT 11
2.9 連接 LTE 11
3 AWS IoT 12
3.1 編譯 AWS IoT 範例程式 12
3.2 執行 AWS IoT 範例程式 15
4 Pelion Device Management 16
4.1 創建新的開發者憑證 16
4.2 編譯 mbed-cloud-client 範例程式 17
4.3 執行 mbed-cloud-client 範例程式 18
5 結論 20
6 版本歷史 21

1 概論

新唐的NuDesign NK-980IoT演示板是基於使用新唐NUC980DK61Y芯片，包括1 套10/100 Mbps乙太網端口，兩組高速 USB Host，一個 1Gb SPI NAND，麥克風輸入，耳機輸出，以及 Arduino 兼容接口。本平台支援 Linux 作業系統，以及各式 IoT 平台所需的通訊協議，例如 AWS Client / mbed-cloud Client / MQTT / Web Server 等。

用戶可以按照本文件快速啟動NuDesign NK-980IoT演示板及開發並快速開發 IoT 相關應用程式。

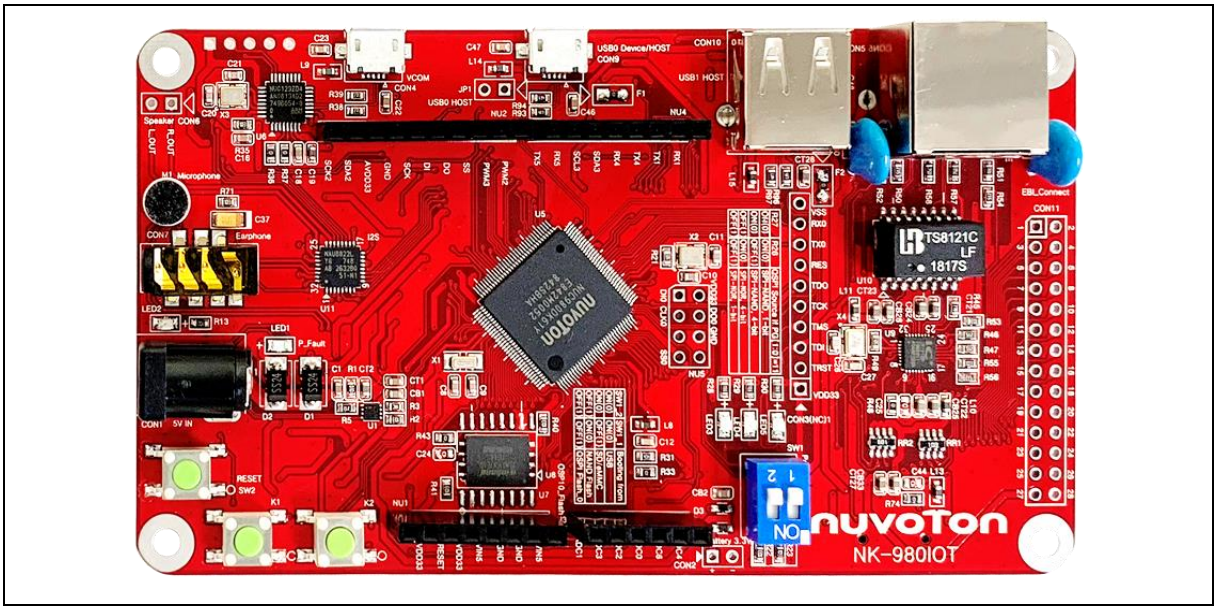

Figure 1-1 NuDesign NK-980IoT 演示板 – 正面

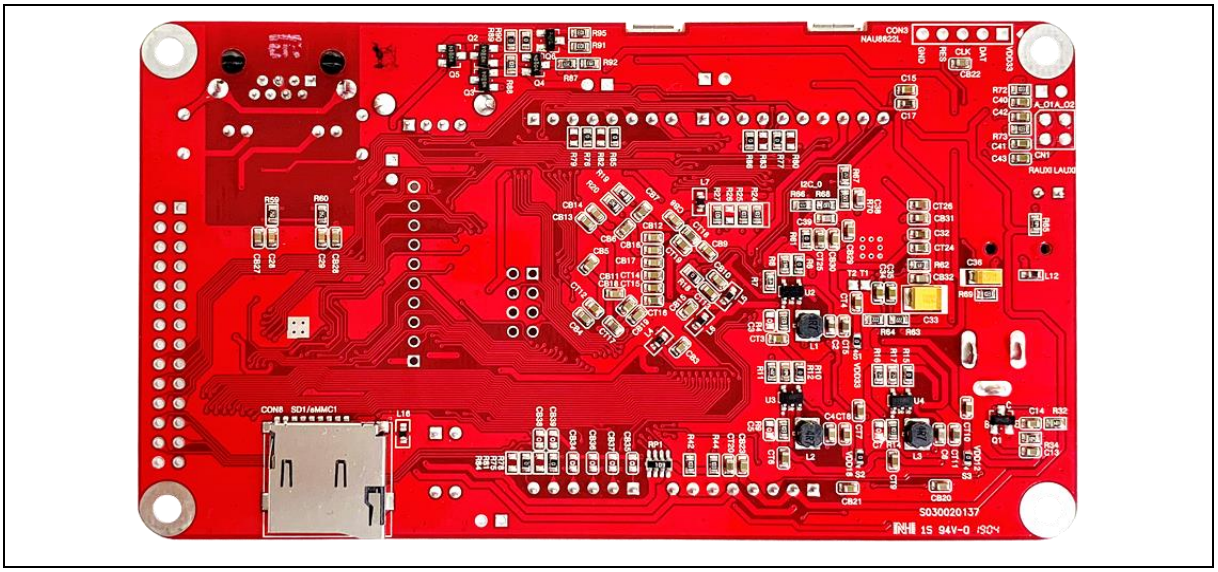

Figure 1-2 NuDesign NK-980IoT 演示板 – 背面

2 快速使用 NuDesign NK-980IoT

2.1 安裝 BSP

Linux BSP 包含了三個目錄。各目錄的內容列在下表:

目錄名稱	內容
BSP	一個包含了交叉編譯工具, 預編映像檔以及根文件系統的壓縮包。
Documents	BSP 相關文件。
Tools	Windows 上的NuWriter燒錄工具以及驅動程式. 以及 SD Writer。

Table 2-1 BSP 包內容

BSP 的源碼可以使用 repo 工具下載. 以下使使用 repo 下載的方式.
確認帳號主目錄下存在 bin/ 目錄, 且路徑有加入path 環境變數中.

```
$ mkdir ~/bin
$ export PATH=~/bin:$PATH
```

下載 repo 工具並設定可執行屬性.

```
$ curl https://storage.googleapis.com/git-repo-downloads/repo >
~/bin/repo
$ chmod a+x ~/bin/repo
```

產生一個空的工作目錄.

```
$ mkdir WORKING_DIR
$ cd WORKING_DIR
```

設定 git 使用的姓名及電子郵件地址.

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

使用以下兩個命令其中之一下載 NUC980 BSP 的 manifest 檔案. 第一個是從 Github 下載, 第二個是從 Gitee 下載. 使用者依據所在的的連線速度擇一使用即可

```
$ repo init -u git://github.com/OpenNuvoton/manifest.git -b nuc980-
2019.09 -m github.xml
```

或

```
$ repo init -u https://gitee.com/OpenNuvoton/manifest.git -b nuc980-
2019.09 -m gitee.xml
```

接著用以下命令下載源碼.

```
$ repo sync
```

下載完源碼後, 請將 BSP 目錄中的壓縮包複製到 Linux 開發機器上，並使用以下的命令解壓縮：

```
$ tar zxvf nuc980bsp.tar.gz
```

在此目錄中有安裝腳本 install.sh，此腳本需要管理者權限才可以執行。可以選擇使用 “su” 命令切換到管理者來執行。

```
$ su
Password: (Enter password of root)
# ./install.sh
```

或是使用 sudo 來執行安裝腳本 (若是安裝的 Linux 沒有開放 root 權限，例如 Ubuntu，則可以使用本方式來安裝 BSP) 。

```
# sudo ./install.sh
```

以下為安裝過程的控制台輸出, 過程中輸入的目錄需要跟之前設定的 WORKING_DIR 相同.

```
Now installing arm_linux_4.8 tool chain to /usr/local/
Setting tool chain environment
Installing arm_linux_4.8 tool chain successfully
Install rootfs, applications, u-boot and Linux kernel
Please enter absolute path for installing(eg:/home/<user name>) :
BSP will be installed in /<path you input>/nuc980bsp
/home/someone
Extract rootfs and pre-build images
...
...
NUC980 BSP installation complete
```

2.2 編譯 U-Boot

用戶可以按照以下步驟編譯 NuDesign NK-980IoT 使用的 U-Boot。

進入U-Boot資料夾。

```
# cd u-boot-2016.11
```

配置NuDesign NK-980IoT演示板的 U-Boot 設定。

```
$ make nuc980_iot_defconfig
```

編譯U-Boot。編譯完成後，會產生 u-boot.bin 及 spl/u-boot-spl.bin

```
$ make
......
  COPY    u-boot.bin
  SYM     u-boot.sym
./scripts/check-config.sh u-boot.cfg \
                ./scripts/config_whitelist.txt . 1>&2
```

2.3 編譯 Linux 內核

用戶可以按照以下步驟編譯 kernel image。

進入kernel資料夾。

```
# cd linux-4.4.y
```

設定NuDesign NK-980IoT演示板的內核配置。

```
# make nuc980_iot_defconfig
```

編譯內核。

```
$ make uImage
......
    Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
  GZIP    arch/arm/boot/compressed/piggy.gzip
  AS      arch/arm/boot/compressed/piggy.gzip.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
  Kernel: arch/arm/boot/Image is ready
cp arch/arm/boot/Image   ../image/980image
  Kernel: arch/arm/boot/zImage is ready
  UIMAGE  arch/arm/boot/uImage
Image Name:   Linux-4.4.115
Created:      Tue Dec 25 16:15:24 2018
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    8039480 Bytes = 7851.05 kB = 7.67 MB
Load Address: 00008000
Entry Point:  00008000
  Image arch/arm/boot/uImage is ready
cp arch/arm/boot/uImage   ../image/980uimage
$ ls ../image/
980image 980uimage
```

2.4 燒錄 Linux Kernel 及 U-Boot 至 SPI NAND Flash

本節介紹如何將U-Boot和內核燒錄至SPI NAND Flash。

A. 安裝NuWriter驅動程序。(請參閱 “NUC980 NuWriter User Manual”)

B. 將SW1(Power On Setting) 設置成從 USB 開機。將高速 USB Device 接口 接到PC。

SW1.2/SW1.1 Switch State	Function
ON/ON	Boot from USB
ON/OFF	Boot from SD/eMMC
OFF/OFF	Boot from SPI Flash

Table 2-2 Power On Setting

C. 打開NuWriter，將Select target chip 設置成 “NUC980 series” 並將 Select DDR parameter 設置成 “NUC980DK61Y.ini” 。再按下 “Continue” 按鈕，並在主視窗下搭是選單中選擇 “SPI NAND” 模式。

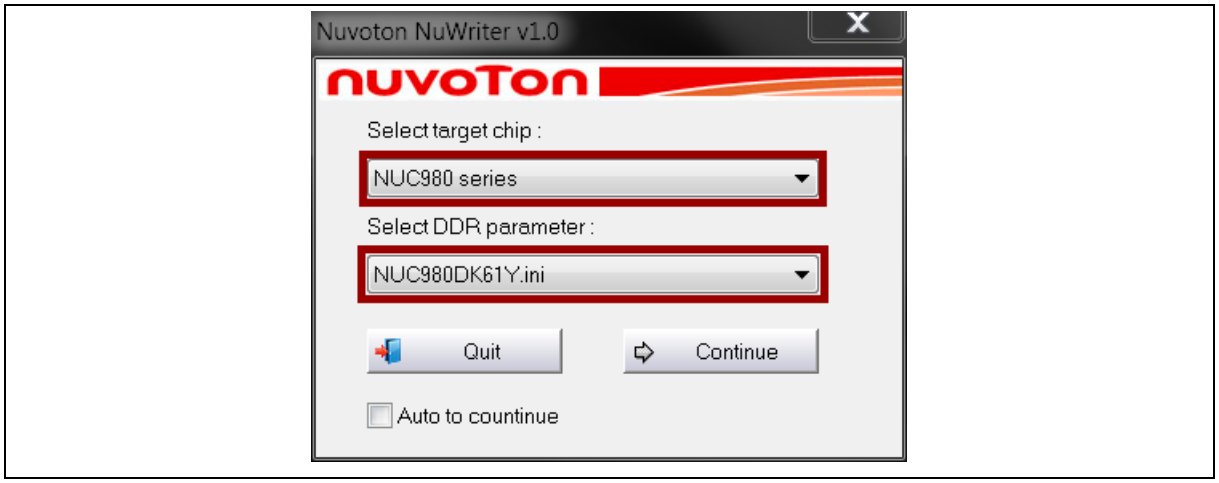

Figure 2-1 NuWriter 設置

D. 燒錄“u-boot-2016.11/spl” 資料夾內的 u-boot-spl.bin 步驟:

1. 輸入參數
   - **Image Name** : u-boot-spl.bin (參照2.2節)
   - **Image Type** : Loader
   - **Image execute address** : 0x200
2. 按下 “Program” 鍵
3. 等待燒錄完成

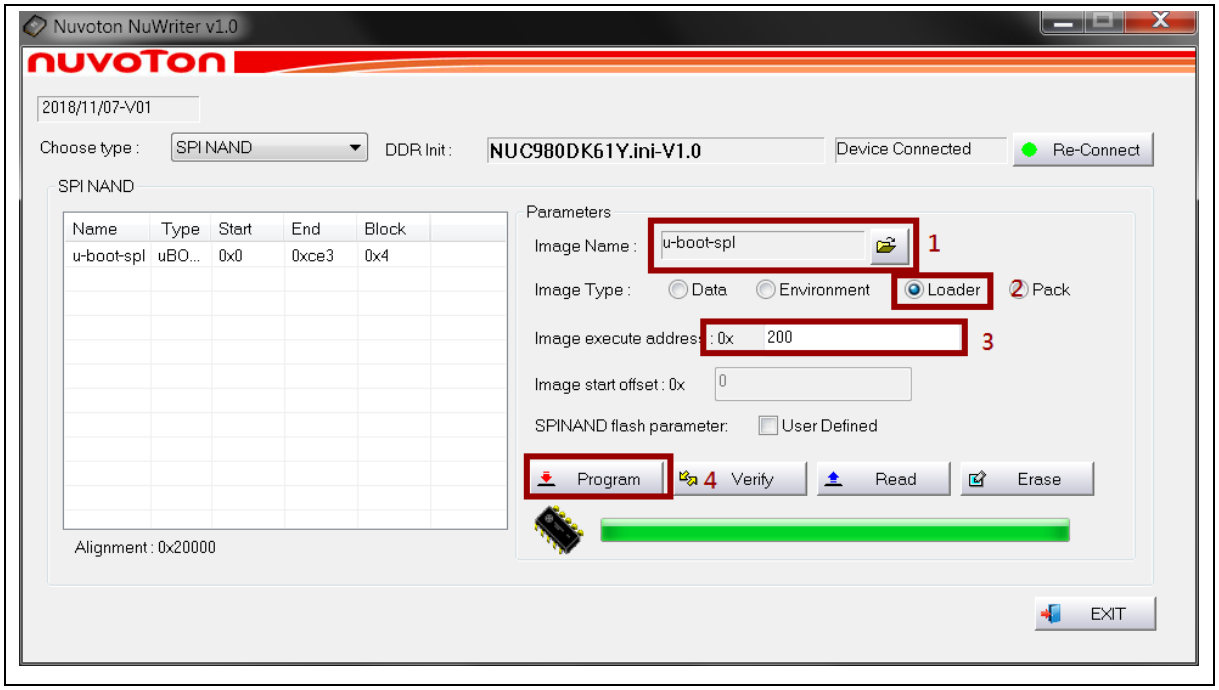

Figure 2-2 燒錄 u-boot-spl

E. 燒錄“u-boot-2016.11” 資料夾內的 u-boot.bin 步驟:

1. 輸入參數
   - **Image Name** : u-boot.bin (參照2.2節)
   - **Image Type** : Data
   - **Image start offset** : 0x100000
2. 按下 “Program” 鍵
3. 等待燒錄完成

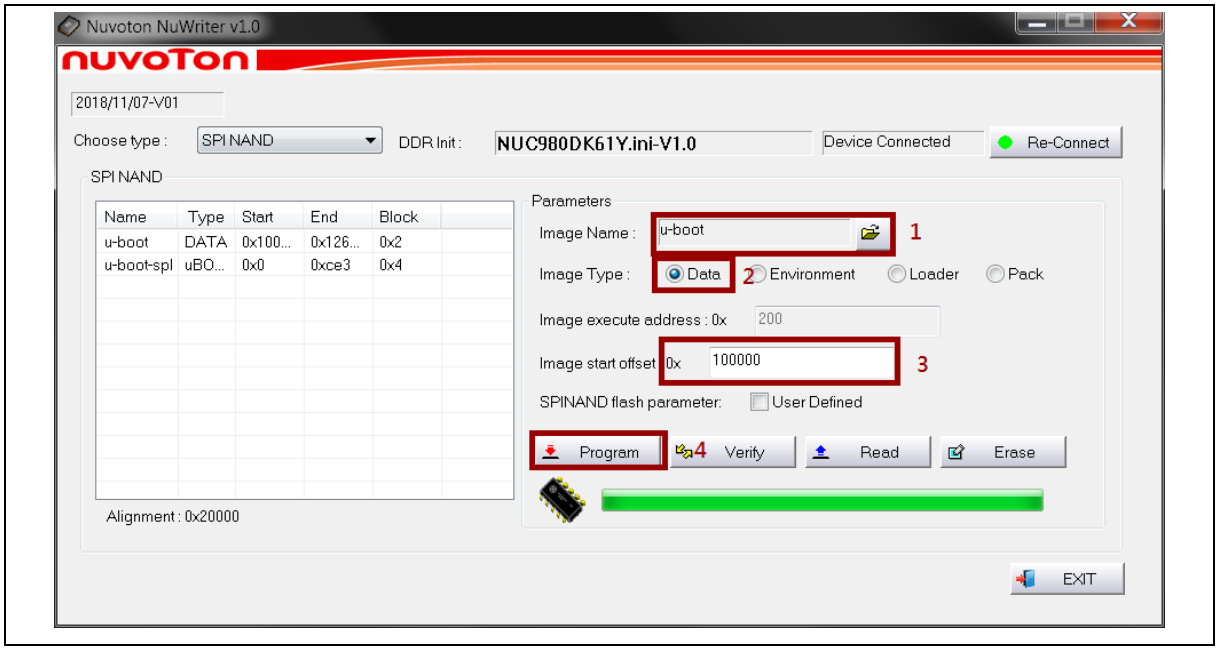

Figure 2-3 燒錄 u-boot

F. 燒錄 kernel image:

1. 輸入參數
   - **Image Name** : 980uimage (參照2.3節)
   - **Image Type** : Data
   - **Image start offset** : 0x200000
2. 按下 “Program” 鍵
3. 等待燒錄完成

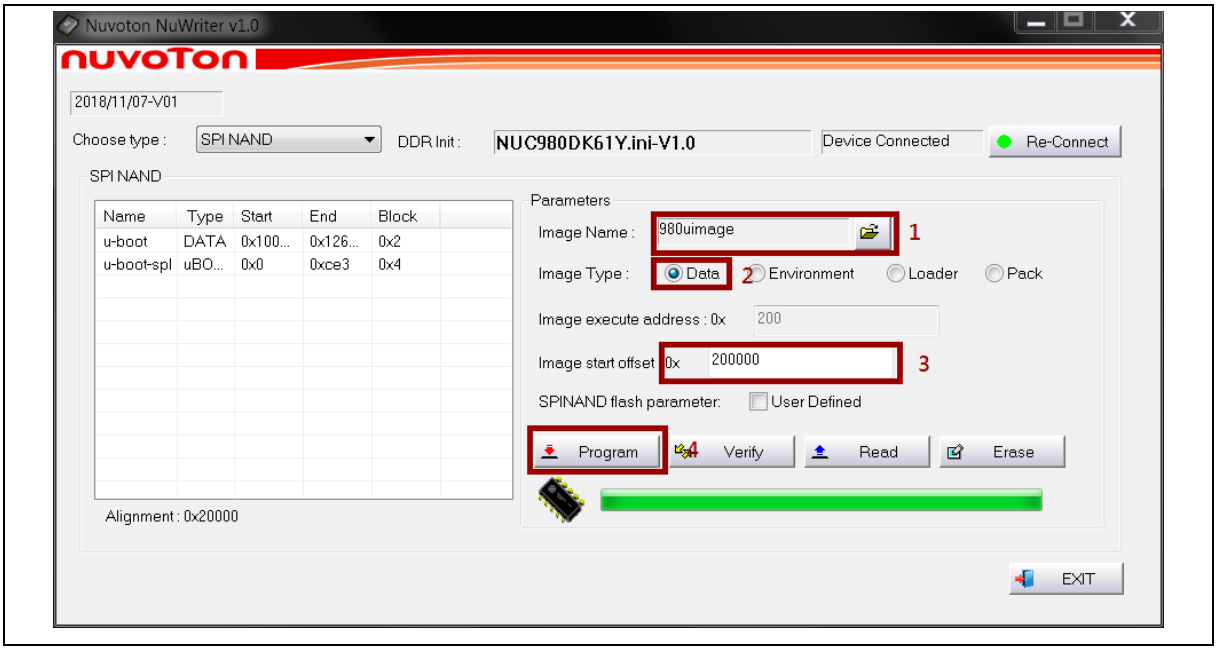

Figure 2-4 燒錄 Kernel image

2.5 硬體設置

A. 將SW1(Power On Setting) 設置成從QSPI 0 Flash開機(參照 Table 2-2) 。

B. 連接 USB CDC Debug Port 作為 console。連線設定為 1152008n1。

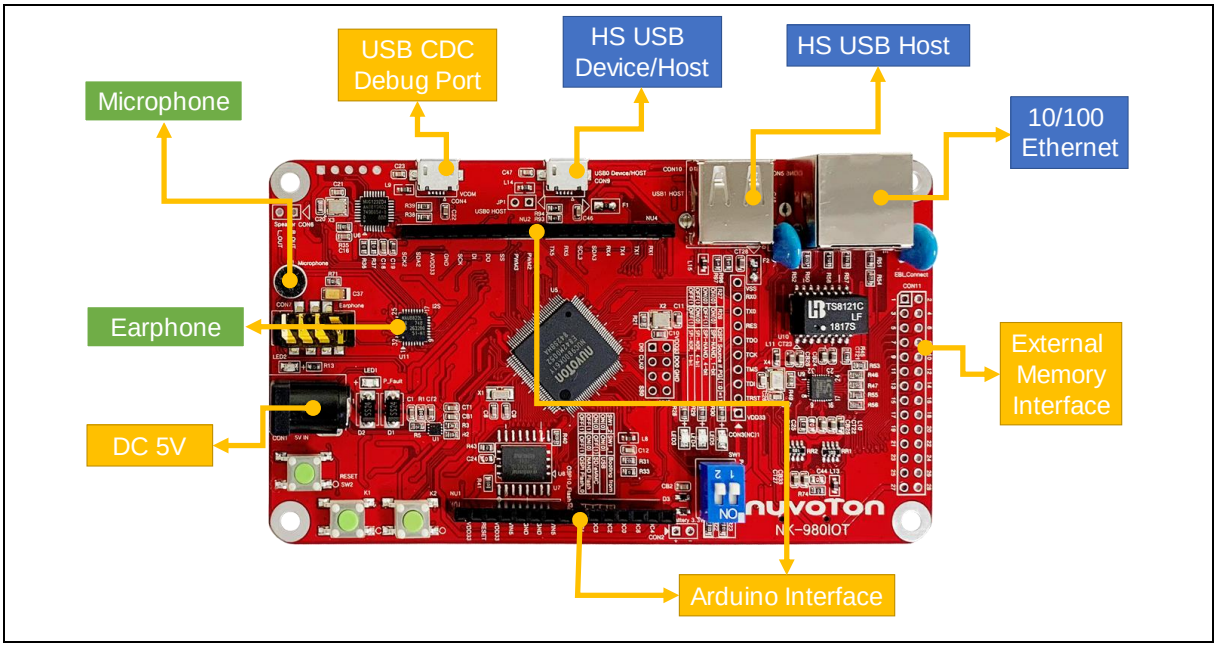

Figure 2-5 NuDesign NK-980IoT Board 設置

2.6 完成 Linux 內核開機

本節說明如何從SPI NAND 啟動Linux內核。

A. 將SW1(Power On Setting) 設置成從QSPI 0 Flash開機 (參照 Table 2-2)。

B. 按下演示板上的Reset 鍵,從 UART console 打印訊息，可以發現系統進入了U-Boot. 用戶需要按照以下步驟由U-Boot 帶起內核。

- 輸入 “sf probe 0 75000000” 去設定 SPI 的速度 (可忽略)
- 輸入 “sf read 0x7FC0 0x200000 0x800000” 將kernel image從SPI NAND Flash讀到DDR內
- 輸入 “bootm 0x7FC0” 啟動 Linux kernel image

```
U-Boot 2016.11-g82e8631-dirty (Jan 30 2019 - 14:55:45 +0800)


CPU: NUC980
Board: NUC980
DRAM:  64 MiB
SF: Detected W25N01GV with page size 2 KiB, erase size 128 KiB, total
128 MiB
In:    serial
Out:   serial
Err:   serial
Net:   Net Initialization Skipped
No ethernet found.
=>
=> sf probe 0 75000000
SF: Detected W25N01GV with page size 2 KiB, erase size 128 KiB, total
128 MiB
=> sf read 0x7FC0 0x200000 0x800000
device 0 offset 0x200000, size 0x800000
SF: 7733248 bytes @ 0x200000 Read: OK
=> bootm 0x7FC0
## Booting kernel from Legacy Image at 00007fc0 ...
   Image Name:   Linux-4.4.115+
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    7573624 Bytes = 7.2 MiB
   Load Address: 00008000
   Entry Point:  00008000
   Verifying Checksum ... OK
   XIP Kernel Image ... OK

Starting kernel ...
```

C. 從打印訊息可以發現 Linux kernel 完成開機。

```
nuc980-i2c0 nuc980-i2c0: i2c-0: nuc980 I2C adapter
nuc980-i2c2 nuc980-i2c2: i2c-2: nuc980 I2C adapter
nuc980_sd_probe - pdev = nuc980-sdh
```

```
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
nuc980-nadc: nuc980 Normal ADC adapter
nuc980-audio nuc980-audio: nau8822-hifi <-> nuc980-audio-i2s mapping ok
NET: Registered protocol family 17
ALSA device list:
  #0: nuc980_IIS
Freeing unused kernel memory: 5304K

BusyBox v1.22.1 (2016-02-03 14:11:04 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ # random: nonblocking pool is initialized
```

D. 關於詳細的Linux kernel 編譯和設定，用戶可以參閱 “NUC980 Linux 4.4 BSP User Manual” 。

2.7 連接 WiFi Access Point

要使用WiFi 連線，使用者在進入shell 後須先安裝驅動程式，以 RTL8188EU 為例，執行以下命命。

```
# insmod /lib/modules/8188eu.ko
```

接著執行 wpa_supplicant，參數中的 /etc/wpa_supplicant 需帶有 access point 的 SSID，PSK...等資訊。連線完成後，再透過 DHCP 取得 IP 位址。

```
# /usr/wpa_supplicant -Dwext -iwlan0 -c/etc/wpa_supplicant.conf -B
# udhcpc -i wlan0
```

2.8 連接 NB IoT

要使用NB-IoT 連線，使用者可以透過使能以下選項使用 buildroot 交叉編譯撥號軟件pppd。

```
Target Packages --->
    Networking applications --->
        [*] pppd
```

在 rootfs/etc/ppp/peers/quectel-chat-connect 裡面設置 APN 的名稱，並在 rootfs/etc/ppp/peers/quectel-ppp 裡面設置使用者名稱及密碼。完成後執行以下命令撥號。

```
# /usr/sbin/pppd call quectel-ppp &
```

2.9 連接 LTE

若要使用LTE連線，使用者可以透過位於 usr/ 目錄下的quectel-CM 程式撥號。此撥號程式可如以下範例接收參數設定 APN，使用者名稱及密碼，認證協議，以及 PIN 碼。本範例中的 0 表示無認證協議，其他的選項為1(PAP) ，2(CHAP) ，及 3(MSCHAPv2) 。

```
# /usr/quectel-CM -s my_apn my_username my_password 0 -p 1234 &
```

3 AWS IoT

在本章節裡，我們將說明如何下載、編譯及測試AWS IoT範例程式。

3.1 編譯 AWS IoT 範例程式

由以下的 URL 下載 AWS IoT device SDK using embedded C 的源碼: https://github.com/aws/aws-iot-device-sdk-embedded-C。

```
# git clone https://github.com/aws/aws-iot-device-sdk-embedded-C.git -b release
# cd aws-iot-device-sdk-embedded-C
```

複製證書，私鑰，公鑰，及 root CA 檔案到 “certs/” 目錄中。這些檔案可以在產生 IoT thing 的證書時下載。

由以下的 URL 下載最新版的MbedTLS 程式庫源碼: https://github.com/ARMmbed/mbedtls 至 AWS SDK 的 “external_libs/mbedTLS” 目錄中，並編譯 mbedTLS 程式庫。

```
# git clone https://github.com/ARMmbed/mbedtls.git external_libs/mbedTLS
# cd external_libs/mbedTLS
# export CC=arm-linux-gcc
# export AR=arm-linux-ar
# make
```

進入 “samples/linux/subscribe_publish_sample” 目錄。編輯 “Makefile” 將 CC 設定成 arm-linux-gcc。

```
CC = arm-linux-gcc
```

編輯相同目錄中的 “aws_iot_config.h” 來設定 IoT 資訊。找到裡面由 “Get from console” 註釋包圍的程式區塊。

```
// Get from console
// =============================================
#define AWS_IOT_MQTT_HOST              ""
#define AWS_IOT_MQTT_PORT              443
#define AWS_IOT_MQTT_CLIENT_ID         "c-sdk-client-id"
#define AWS_IOT_MY_THING_NAME          "AWS-IoT-C-SDK"
#define AWS_IOT_ROOT_CA_FILENAME       "rootCA.crt"
#define AWS_IOT_CERTIFICATE_FILENAME   "cert.pem"
#define AWS_IOT_PRIVATE_KEY_FILENAME   "privkey.pem"
// =============================================
```

AWS_IOT_MY_THING_NAME 是在 Amazon cloud IoT service 上建立的 IoT thing 名稱。AWS_IOT_MQTT_HOST 是由 “interact” 頁面所獲得的網址。AWS_IOT_MQTT_CLIENT_ID 可為任意的獨特名稱。

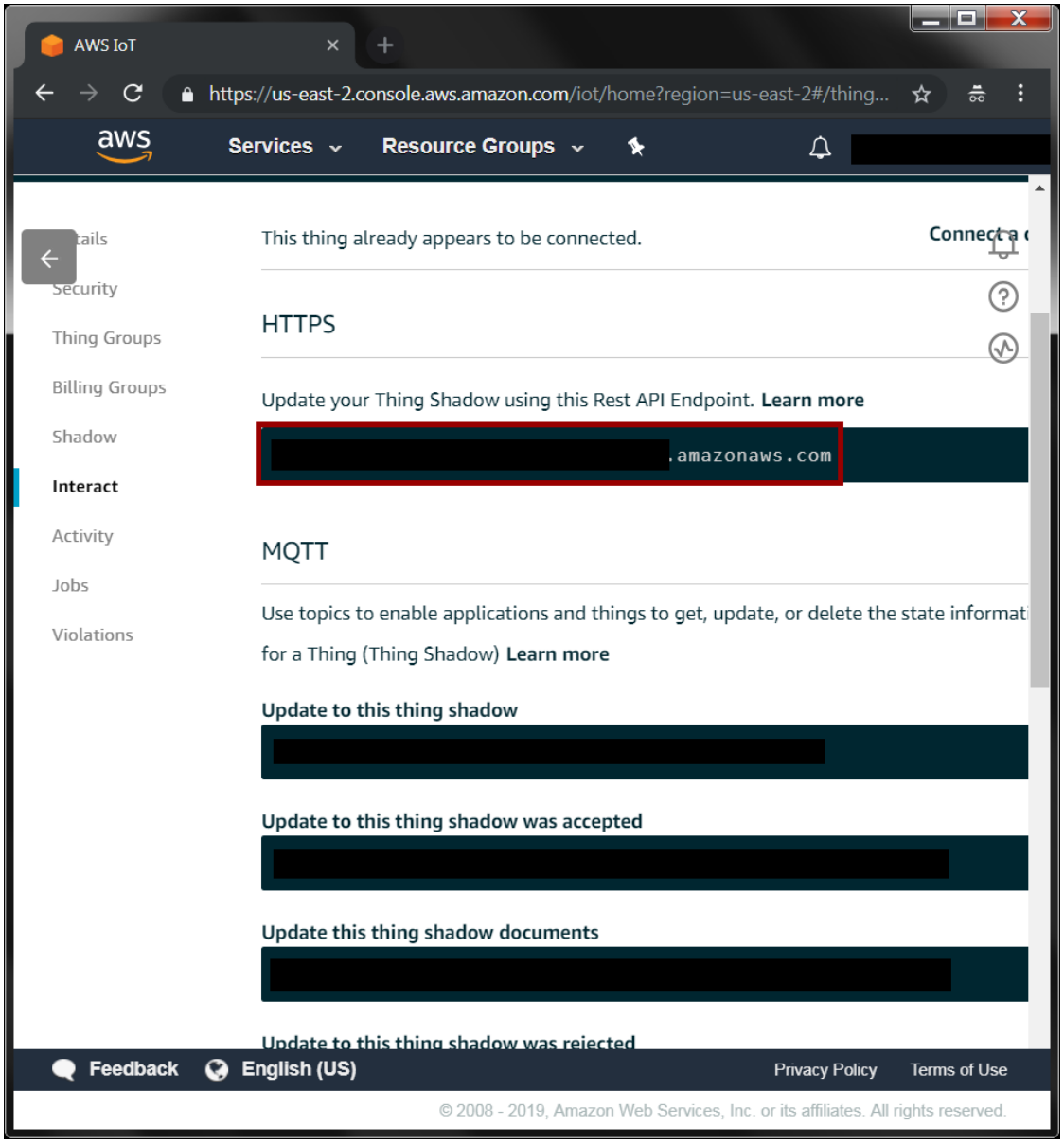

Figure 3-1 取得 AWS_IOT_MQTT_HOST

AWS_IOT_ROOT_CA_FILENAME ， AWS_IOT_CERTIFICATE_FILENAME ， 以 及 AWS_IOT_PRIVATE_KEY_FILENAME 是在取得 IoT thing 證書時下載的證書檔案。

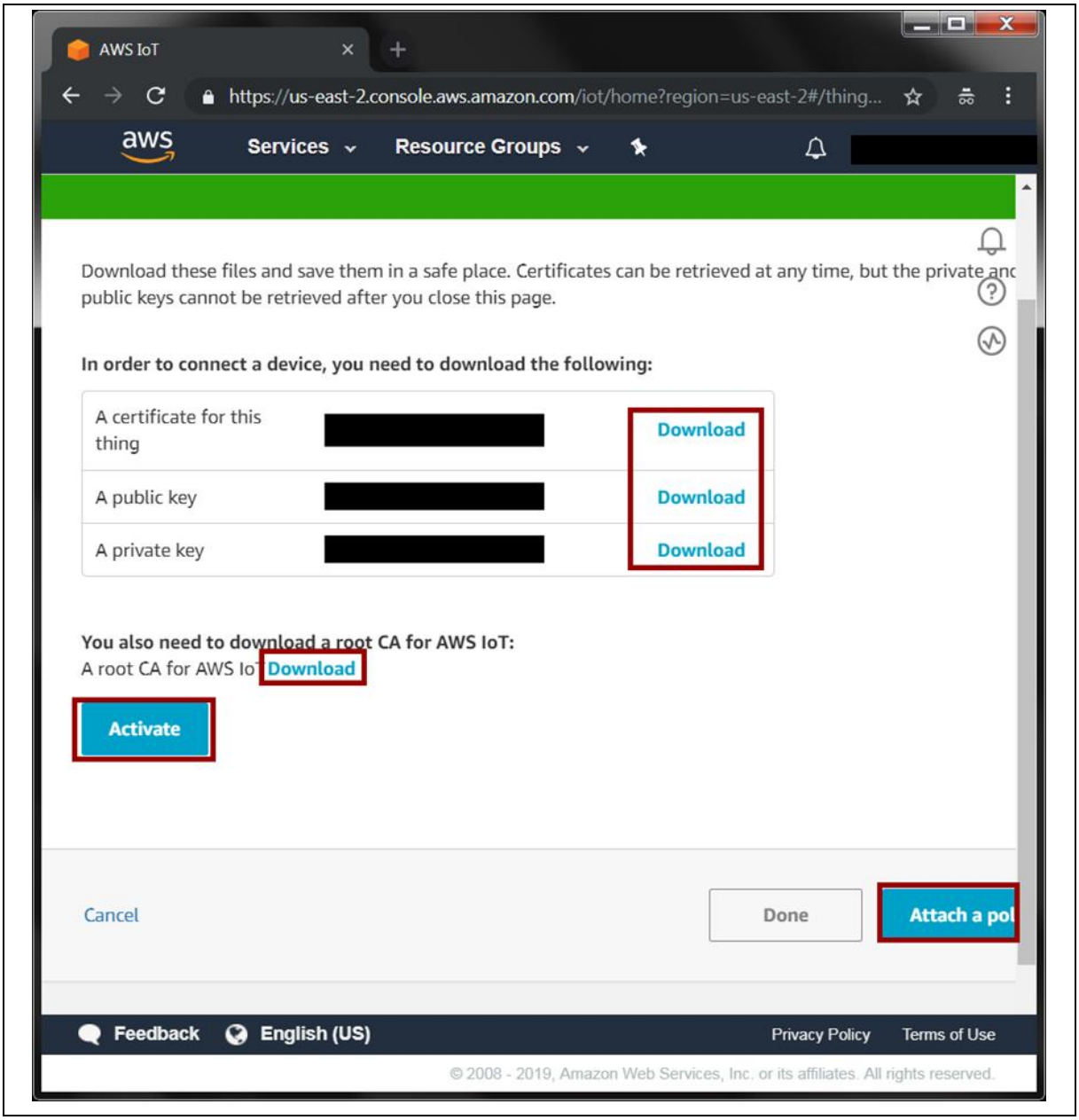

Figure 3-2 取得證書

編輯 “subscribe_publish_sample.c” 裡的 certDirectory[]，這個字串指到了執行時可以找到證書的目錄。假設執行檔“subscribe_publish_sample”放在 “/usr/bin” 目錄中, 證書放在 “/usr/certs” 目錄，則 “certDirectory[]” 應設定成 “../certs” 。

```
static char certDirectory[PATH_MAX + 1] = "../certs";
```

接著在進入 “samples/linux/ubscribe_publish_sample” 目錄，執行“make” 指定來編譯 IoT 客戶端程序 “subscribe_publish_sample” ，並將執行檔及證書放入根文件系統，

若有需要，可以至以下的網頁獲取更多關於 AWS IoT device SDK using embedded C 的訊息。

https://docs.aws.amazon.com/iot/latest/developerguide/iot-embedded-c-sdk.html.

3.2 執行 AWS IoT 範例程式

首先, 必須要建立網路連線，才可以連線到 Amazon 雲端服務。例如可以使用之前提到的 WiFi 或是 NB IoT。若要使用乙太網，則可以使用以下命令帶起網口並透過網關分配 IP 地址給 NUC980，及提供一組 DNS 伺服器地址。

```
# ifconfig eth0 up
# udhcpc –i eth0
```

參考3.1編譯 “subscribe_publish_sample” 範例。然後執行本範例連線到雲端。

請注意, 執行本範例之前，請先註冊 AWS IoT thing 並且激活認證。

除了使用 AWS IoT device SDK (MQTT 客戶端) 外，你也可以使用 HTTP 協議傳送訊息到 AWS 伺服器。以下是使用curl 命令指定證書及私鑰傳送指定訊息到雲端的方式。

```
# cd usr
# ./ curl --tlsv1.2 --cacert root-CA.crt --cert 4b7828d2e5-
certificate.pem.crt --key 4b7828d2e5-private.pem.key -X POST -d
"{ \"serialNumber\": \"G030JF053216F1BS\", \"clickType\": \"SINGLE\",
\"batteryVoltage\": \"2000mV\" }" "https://a1pn10j0v8htvw.iot.us-east-
1.amazonaws.com:8443/topics/iotbutton/virtualButton?qos=1"
```

以下描述個參數的作用:

- --tlsv1.2

使用 TLSv1.2 (SSL) 。必須要安裝 OpenSSL，且 TLS 版本一定要選擇1.2 版。

- --cacert <filename>

用來校驗連線端的 CA 校驗檔案名稱。

- --cert <filename>

客戶端的校驗檔案名稱。

- --key <filename>

The private key filename。

- -X POST

請求的類別 (本例中使用 POST) 。

- -d <data>

要發佈的 HTTP POST 資料，本例中使用了按鍵觸發的狀態。

- "https://..."

使用的 URL。本例中使用了thing 的 REST API 端點。

4 Pelion Device Management

在本章節裡，我們將說明如何下載、編譯及測試 mbed-cloud-client 範例程式。

4.1 創建新的開發者憑證

在 NK-980IoT 裝置連接至 Pelion Device Management Server 之前，需要事先創建一個開發者憑證，以供連接測試。 Pelion Device Management 網址為 https://www.pelion.com/。請登入並進入 ***Device Management Console***。成功登入之後，你可以在頁面上的 ***Device identity***，找到 ***Certificates*** 選項，然後依序點選 ***Certificates***, ***New certificate*** 和 ***Create a developer certificate*** 選項，並在彈跳出的視窗中，輸入憑證附加說明; 若沒有任何的附加說明，可以直接點選 ***Create Certificate*** 按鍵，直接產生憑證。在創建結束後，可在個人憑證列表找到這筆憑證記錄，並點選這筆記錄。在彈跳的視窗中，點選 ***Download Developer C file*** 選項，並儲存這個檔案，這個檔案將用於 mbed-cloud-client-example 建置編譯。

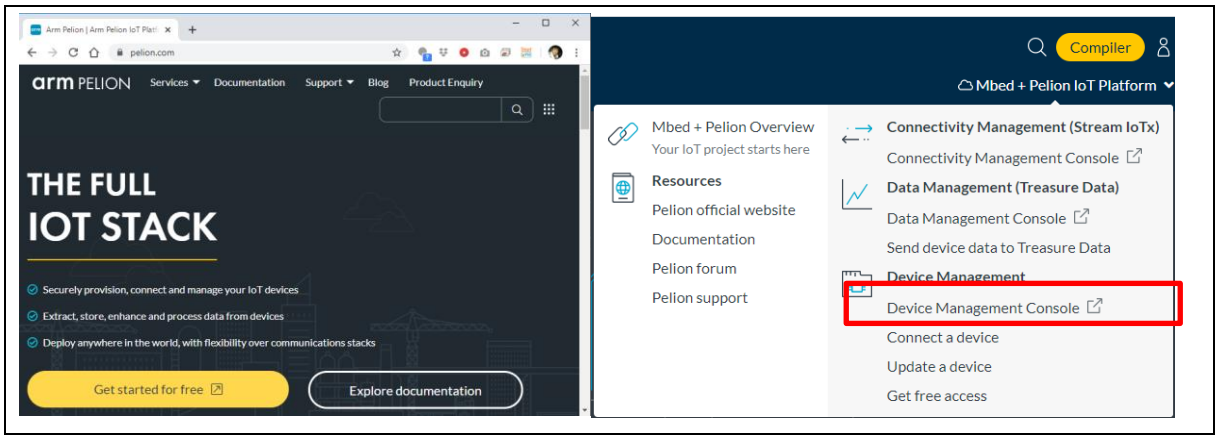

Figure 4-1 進入 Device Management Console

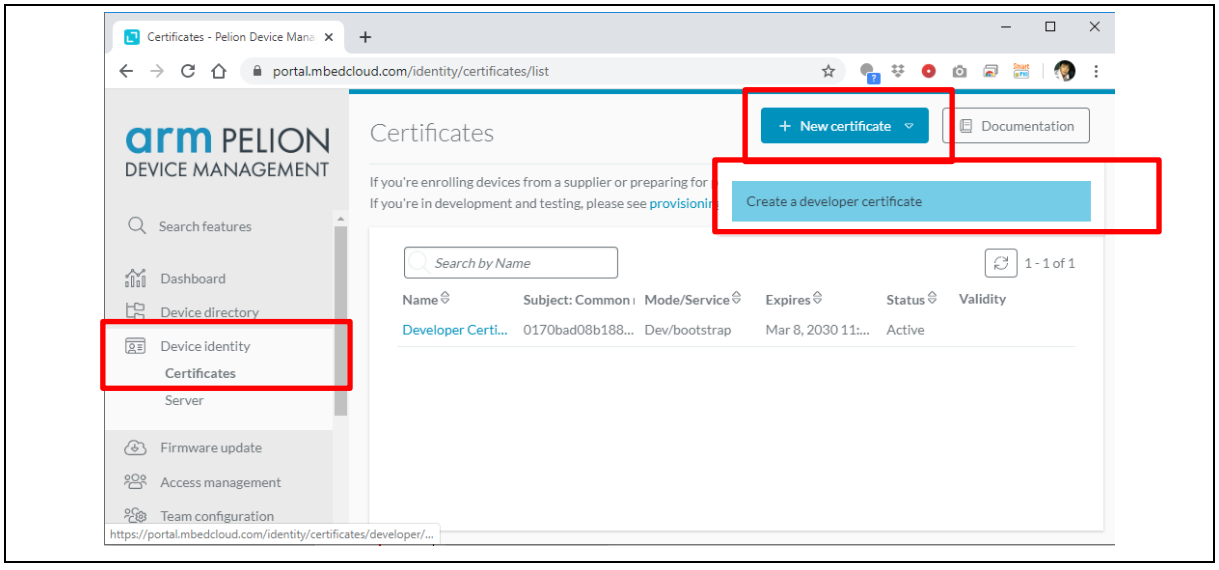

Figure 4-2 創建一個開發者憑證

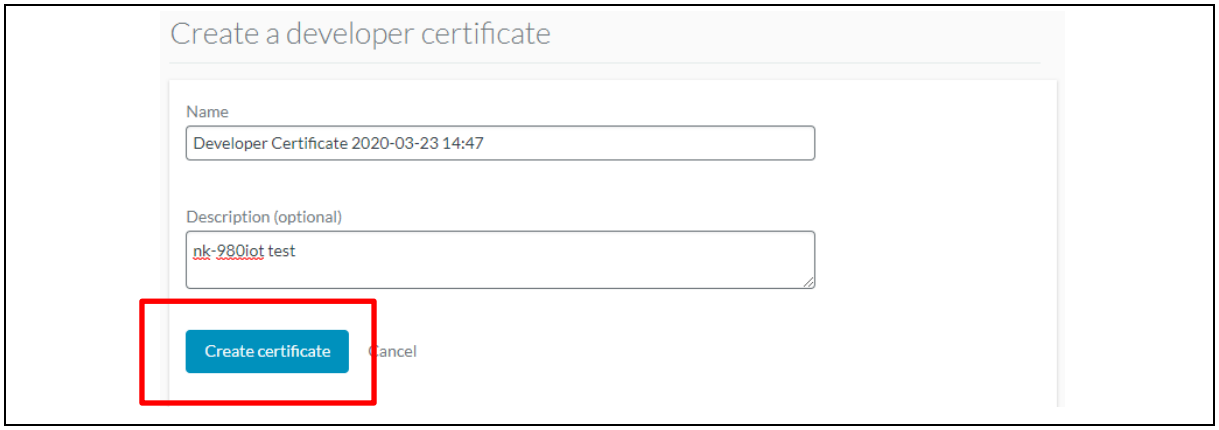

Figure 4-3 輸入附加說明和創建憑證

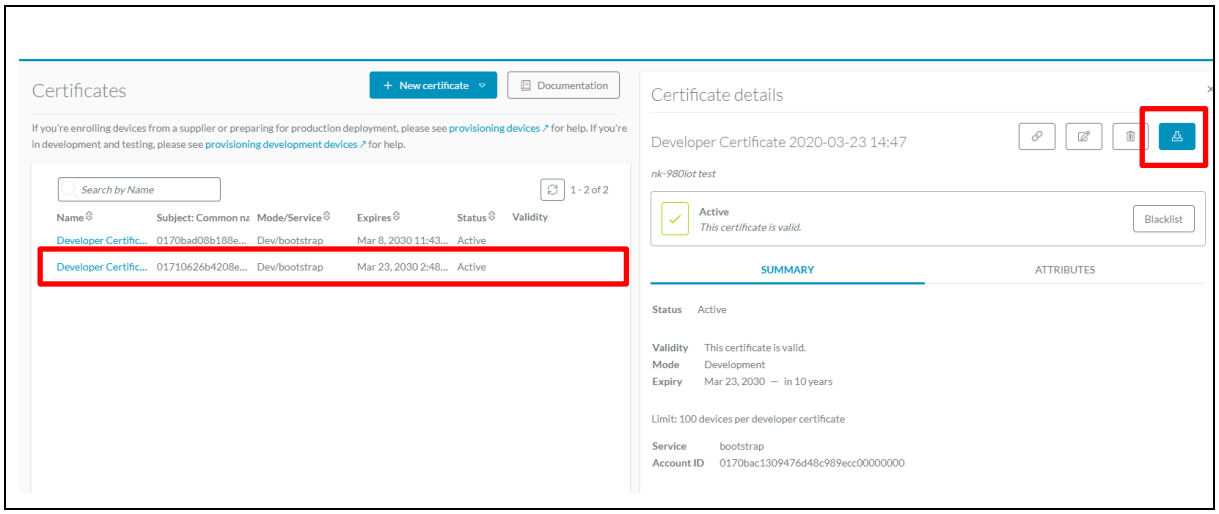

Figure 4-4 下載憑證 C 語言程式檔案

4.2 編譯 mbed-cloud-client 範例程式

從 Github上 的 OpenNuvoton 的官方社群下載 mbed-cloud-client-example 開發包。

網址: https://github.com/OpenNuvoton/mbed-cloud-client-example

```
# sudo apt install python2.7 python-pip git mercurial virtualenv cmake
# git clone https://github.com/OpenNuvoton/mbed-cloud-client-example -b NK-980IOT
# cd mbed-cloud-client-example
```

複製剛下載的開發者憑證 C 語言程式檔案 - *mbed_cloud_dev_credentials.c* 到根目錄(C 語言檔案是從 Pelion Device Management 上創建、下載到本地端) 。然後，執行下列的指令安裝 virtualenv 及部署相依的程式庫。最後執行 '*./build_nuc980.sh*' 編譯範例程式。編譯產生的執行檔將被放置在 *__ARM_NUC980_mbedtls/Debug/mbedCloudClientExample.elf.elf*。

```
# pip2 install virtualenv
# mkdir -p ~/virtualenvs
# virtualenv ~/virtualenvs/nk-980iot
```

```
# . ~/virtualenvs/nk-980iot/bin/activate
(nk-980iot)# pip install mbed-cli
(nk-980iot)# mbed deploy
(nk-980iot)# ./build_nuc980.sh
(nk-980iot)# ls __ARM_NUC980_mbedtls/Debug/mbedCloudClientExample.elf.elf
```

4.3 執行 mbed-cloud-client 範例程式

首先，必須要建立網路連線，才可以連線到 mbed-cloud 服務。例如可以使用之前提到的 WiFi 或是 NB-IoT。若要使用乙太網，則可以使用以下命令帶起網口並透過網關分配 IP 地址給 NUC980及提供DNS 伺服器地址。同時，在執行範例程式之前，也需要更新系統時間，以避免憑證過期導致連接失敗。

```
# ifconfig eth0 up
# udhcpc -i eth0
# date -s "2020-03-23 15:16:00"
Mon Mar 23 15:16:00 UTC 2020
```

測試時，可將 mbedCloudClientExample.elf.elf 執行檔存放一張 SD 儲存卡內，並於 NK-980IoT 開機完成後，插入這張 SD 並掛載到指定目錄下。最後執行 mbedCloudClientExample.elf.elf 並且觀察在 console 是否出現 “Client registered” 訊息，這個訊息代表成功連接 mbed-cloud。

```
mmc0: new SD card at address e624
mmcblk0: mmc0:e624 SU128 120 MiB

# mkdir -p /mnt/mbed-pelion/
# mount -t vfat /dev/mmcblk0 /mnt/mbed-pelion/
# cd /mnt/mbed-pelion/
# ./mbedCloudClientExample.elf.elf
In single-partition mode.
Creating path ./pal
Start Device Management Client
Using hardcoded Root of Trust, not suitable for production use.
Starting developer flow
Generating random from /dev/random, this can take a long time!
Developer credentials already exist, continuing..
Application ready. Build at: Mar 20 2020 16:41:22
Network initialized, registering...
Client registered
Endpoint Name: 0170e7feb99e0000000000010010ea50
Device ID: 0170e7feb99e0000000000010010ea50
Button resource automatically updated. Value 1
......
```

連接成功後，也可以在mbed-cloud裝置目錄清單查看裝置發出的即時事件記錄。在***Device***

directory 選單上，點選 ***Devices***，**找到對應的** Device ID並且點選它。點選後，在 ***Device details*** **視窗內，找到** ***RESOURCES*****頁面**，並點選頁面中的 ***/3200*** 連結，您將可以看到該裝置送出的數位輸入事件。

Figure 4-5 Device – Digital Input Event

5 結論

依照本文件的操作流程，可以簡易地讓 NK-980IoT 連上不同的雲端服務。用戶可以基於本平台快速的開發 IoT網關應用程序。

6 版本歷史

版本號	日期	說明
1.00	Mar. 20, 2019	Initial release
1.01	Apr. 22, 2019	Editorial change.
1.02	Sept. 09, 2019	Add LTE setting.
1.03	Apr. 04, 2020	Add Pelion Device Management chapter.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.