NUC980 Simple OOB Simulation

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example demonstrates how to use the NUC980 series microprocessor (MPU) to build a simple remote monitoring and control application, featuring out-of-band (OOB) capabilities.
BSP Version	NUC980 buildroot_2024
Hardware	NuMaker-RTU-NUC980 (Chili) or NuMaker-IIoT-NUC980, NuMaker-IIoT-NUC980G1, or NuMaker-IIoT-NUC980G2

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 NuMicro microcontroller and 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. Overview

This demo showcases the integration of IoT technologies on the Nuvoton NuMaker NUC980 evaluation board, running Embedded Linux built with Buildroot. The example, written in C and using the Paho MQTT library, enables the board to communicate with an external MQTT broker over Ethernet. It reports system status, button states, and allows remote control of an onboard LED.

1.1 Principle

When running the example on the NuMaker NUC980 board, it connects to test.mosquitto.org (a free MQTT broker for testing) using the MQTT protocol. The NUC980 then subscribes to the following topics:

- board/status: Indicates whether the NuMaker NUC980 board is online or offline.
- board/leds/1: Displays the status of the on-board LED (LED3 on NuMaker-IIoT-NUC980 boards or LED1 on NuMaker-RTU-NUC980 (Chili) board).
- board/buttons/1: Reflects the pressed or released status of the K1 button on board.
- board/temperature/1: Reports temperature data from MAX31875 sensor.

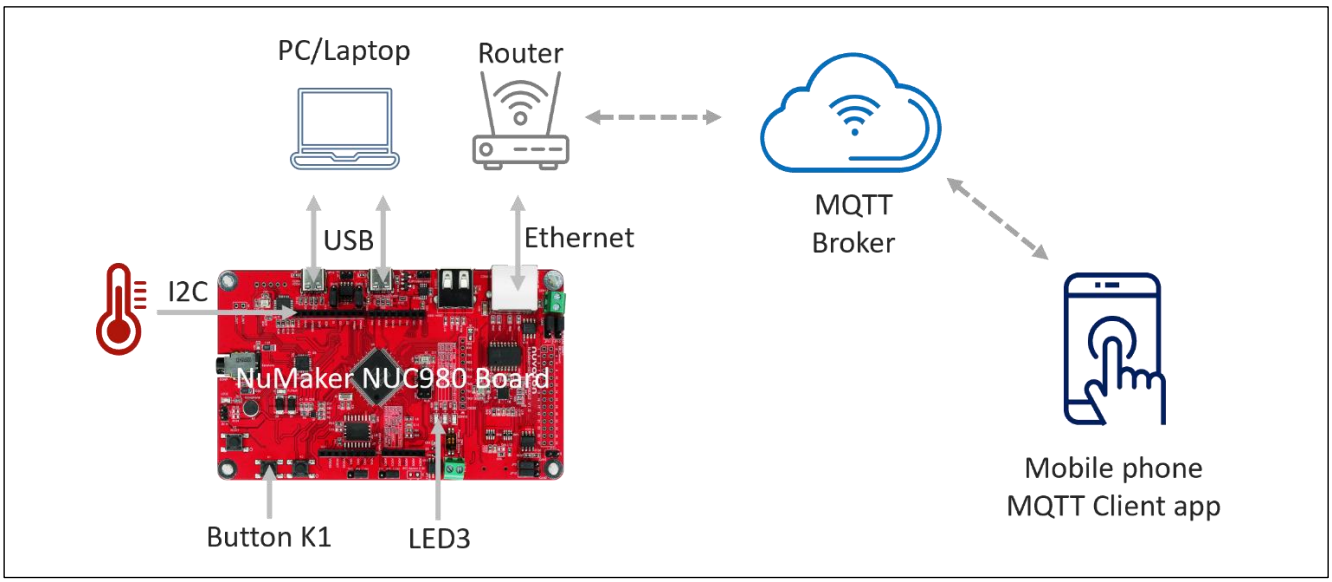


Figure 1-1 Demo Architecture Diagram

1.2 Prerequisite

To set up the demo, first refer to the Software and Hardware Requirements chapter to prepare the necessary components. Then, follow the instructions in the Example Code Execution chapter to build the example code, program it onto the board, and configure the MQTT App.

Pre-built files are included in this example ZIP. Refer to the Directory Information chapter for detailed description.

1.3 Demo Result

After programming the firmware onto the NuMaker NUC980 board and configuring the MQTT App, you can begin running the demo.

- Reboot the NuMaker NUC980 board, and use “root” as the account to log in.
- Enable Ethernet and obtain IP address from DHCP server.

```
# udhcpc
```

- Run the rcdemo.

```
# ./rcdemo
```

- The NuMaker NUC980 board connects to MQTT broker, subscribes to topics, and publishes its status.

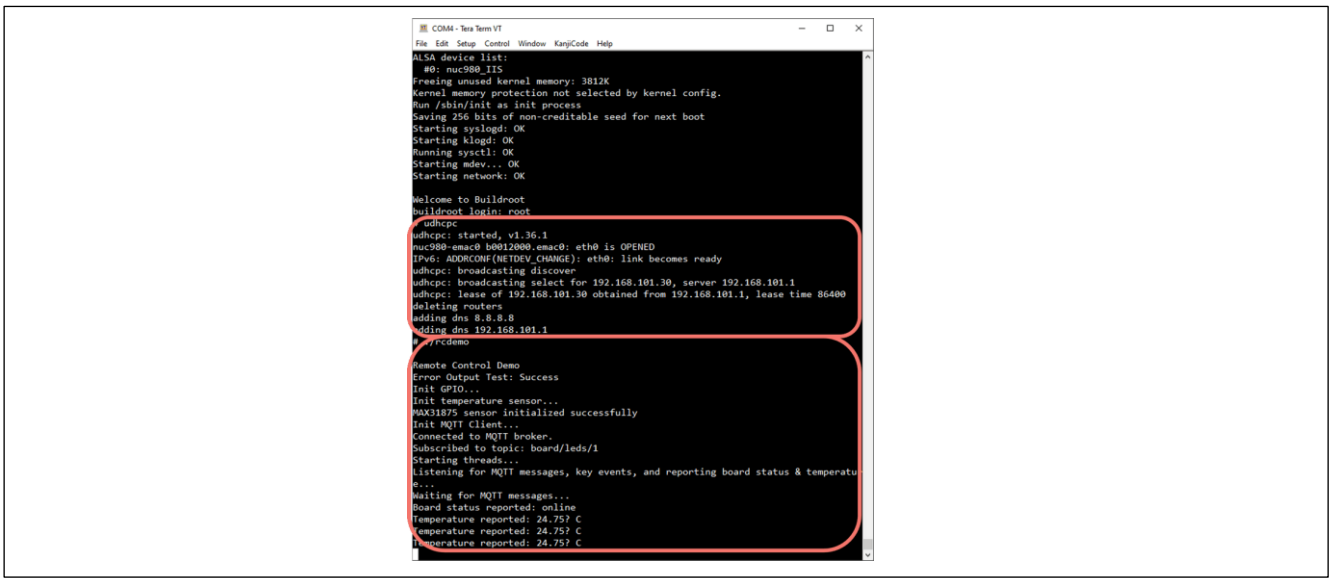


Figure 1-2 Messages on NUC980 Console

- Open MQTT App on Android mobile phone.
- Make sure the App connects to MQTT broker. The cloud icon should be orange. If not, tap it to connect again.
- When the NuMaker board is running the demo and successfully connects to the MQTT broker, the **Board Status** will show as **online**.
- You can now try the following controls:

- Press and hold **K1** on the NuMaker NUC980 board; the **Button status** will display as **pressed**. Release **K1** and **released** will appear.
- Tap the **LED 1** switch in the App to control the onboard **LED**.
- The App will display a history of temperature readings.

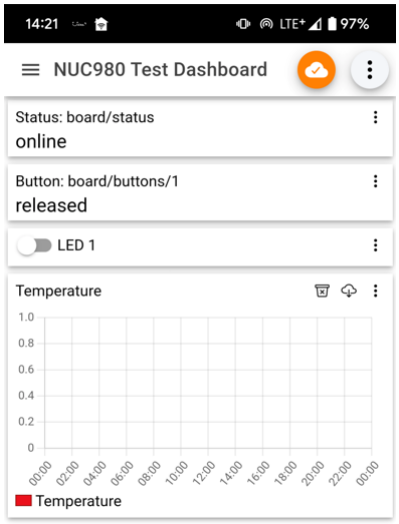


Figure 1-3 Screen Displayed by MQTT App

2. Code Description

All code for this example code is in the *rcdemo.c*. NUMAKER_BOARD is used to specify the board used for LED, Key I/O and I²C port.

```
#define NUC980_IOT      1      // For NuMaker-IIoT-NUC980/G1/G2 board
#define NUC980_CHILI    2      // For NuMaker-RTU-NUC980 (Chili) board

#ifndef NUMAKER_BOARD
#define NUMAKER_BOARD NUC980_IOT    // Default to IoT series board if not defined.
#endif
```

The URL and port of MQTT broker is assigned to ADDRESS.

```
#define ADDRESS     "tcp://test.mosquitto.org:1883"
#define CLIENTID    "NUC980Client"
#define QOS         1
#define TIMEOUT     10000L
```

The demo source creates different threads to process key (key_monitor_thread), board status (board_status_thread), temperature (temperature_thread), and LED (mqtt_client_thread).

```
    // Start threads
    printf("Starting threads...\n");
    pthread_create(&key_thread, NULL, key_monitor_thread, NULL);
    pthread_create(&status_thread, NULL, board_status_thread, NULL);
    pthread_create(&temp_thread, NULL, temperature_thread, NULL);
    pthread_create(&mqtt_thread, NULL, mqtt_client_thread, NULL);
```

The Paho MQTT library makes it easy to create a client and subscribe to the MQTT broker.

```
    // Initialize MQTT client
    printf("Init MQTT Client...\n");
    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS)
    {
        printf("Failed to connect to MQTT broker, return code %d\n", rc);
        cleanup_gpio();
        exit(EXIT_FAILURE);
    }
    printf("Connected to MQTT broker.\n");

    if ((rc = MQTTClient_subscribe(client, TOPIC_LED, QOS)) != MQTTCLIENT_SUCCESS)
    {
        printf("Failed to subscribe topic %s, return code %d\n", TOPIC_LED, rc);
```

```
    }
    else
    {
        printf("Subscribed to topic: %s\n", TOPIC_LED);
    }
```

3. Software and Hardware Requirements

3.1 Software Requirements

- PC running Ubuntu
  - A physical machine or a virtual machine (VM) running Ubuntu is okay.
- Buildroot
  - Available at https://github.com/OpenNuvoton/buildroot_2024.
- Terminal Emulator on PC
  - Putty, Realterm, or TeraTerm, etc.
- MQTT Application
  - IoT MQTT Panel (available on Google Play)
  - Or any other MQTT client Apps.

3.2 Hardware Requirements

- One of the following NuMaker NUC980 boards
  - NuMaker-IIoT-NUC980 boards, including:
    - NuMaker-IIoT-NUC980
    - NuMaker-IIoT-NUC980G1
    - NuMaker-IIoT-NUC980G2
  - NuMaker-RTU-NUC980 (Chili)
- LAN cable
- Two USB cables
  - MicroUSB for NuMaker-IIoT-NUC980 or Chili
  - Type-C for NuMaker-IIoT-NUC980G1 or NuMaker-IIoT-NUC980G2
- Wi-Fi Router with LAN
- Thermo 6 Click board
  - For MAX31875 temperature sensor

Refer to the diagrams below to connect the demo environment.

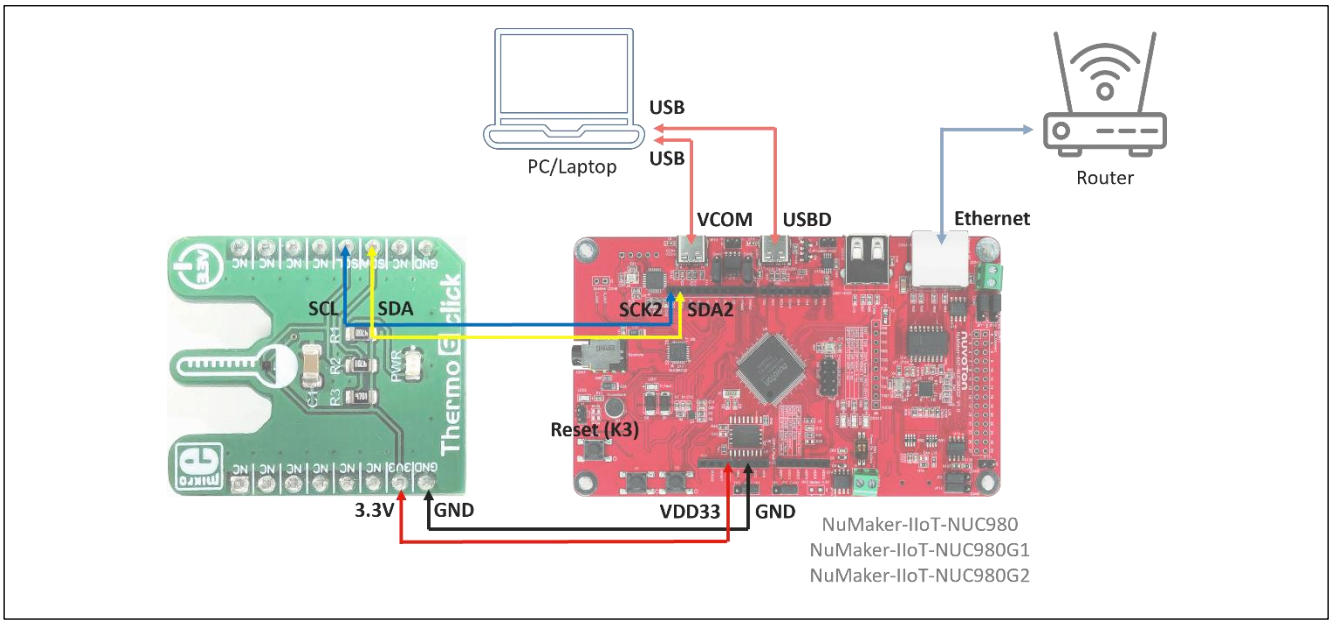


Figure 3-1 NuMaker-IIoT-NUC980 Board Connections

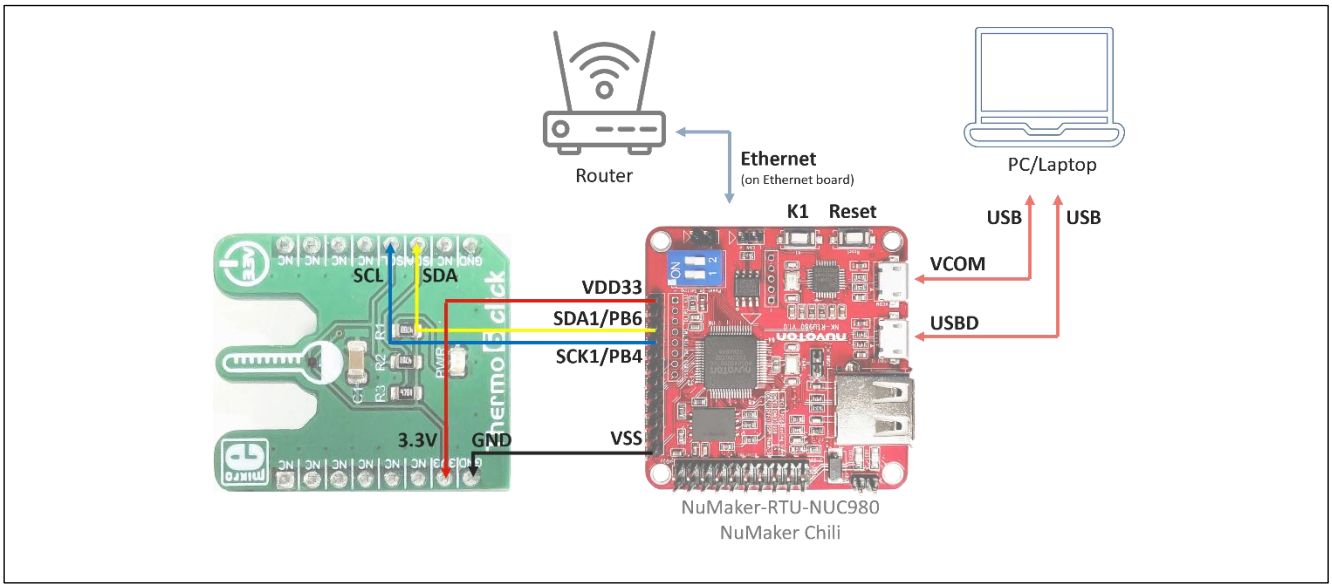


Figure 3-2 NuMaker-RTU-NUC980 Board Connections

4. Directory Information

The directory structure is shown below.

EC_NUC980_Simple_OOB_Simulation_V1.00	
numaker-nuc980-iot	U-boot, system image, and device tree for NuMaker-IIoT-NU980 boards
numaker-nuc980-rtu(chili)	U-boot, system image, and device tree for NuMaker-RTU-NU980 board
env.txt	Environment file
rcdemo.c	Example source code

Figure 4-1 Directory Structure

5. Example Code Execution

Follows the steps to create and configure the development environment using NUC980 Buildroot then build the example code.

5.1 Building Firmware

- Buildroot setup in Ubuntu terminal.
  - Clone the buildroot_2024 for NUC980.

```
$ git clone https://github.com/OpenNuvoton/buildroot_2024
$ cd buildroot_2024
```

  - Configure Buildroot for NuMaker NUC980 boards (for NuMaker-IIoT-NUC980 or NuMaker-IIoT-NUC980G1).

```
$ make nuvoton_nuc980_iot_defconfig
```

    For NuMaker-IIoT-NUC980G2.

```
$ make nuvoton_nuc980_iot_g2_defconfig
```

    For NuMaker-RTU-NUC980 (Chili).

```
$ make nuvoton_nuc980_chili_defconfig
```

  - Configure Buildroot to include related packages.

```
$ make menuconfig
```

  - Include Paho MQTT library.

```
Target packages ➔ Libraries ➔ Networking ➔ [*] paho-mqtt-c
```

  - Then click <Save> to save configuration, and click <Exit> to exit configuration.
  - Configure Linux kernel to support NUC980 keyboard events.

```
$ make linux-menuconfig
```

```
Device Drivers ➔ Input device support ➔ <*> Event interface
Device Drivers ➔ Input device support ➔ [*] keyboards ➔ [*] Nuvoton keys
support
```

- Then click <Save> to save configuration, and click <Exit> to exit Linux kernel configuration.
- Build the system image.

```
$ make
```

- Modify device tree to enable I²C on J2 on **NuMaker-RTU-NUC980 (Chili)** only.

```
$ vi output/build/linux-custom/arch/arm/boot/dts/nuc980-chili.dts
```

```
    i2c1: i2c1@b0081000 {
            status = "okay";
            pinctrl-0 = <&pinctrl_i2c1_PB>;
    };
```

- The rcdemo.c source code is included in the example ZIP file. Simply copy it into the Buildroot directory.

```
$ cd ..
$ cp your-directory-of-EC_NUC980_Simple_OOB_simulation_V1.00/rcdemo.c .
```

- Set up cross-compiler environment.

```
$ source buildroot_2024/output/host/environment-setup
```

- Compile the example source code (rcdemo.c) for **NuMaker-IIoT-NUC980G1** or **NuMaker-IIoT-NUC980G2.**

```
$ $CC rcdemo.c -o rcdemo -Ibuildroot_2024/output/staging/usr/include \
  -Lbuildroot_2024/output/staging/usr/lib -Wl,-Bstatic -lpaho-mqtt3c -Wl,-Bdynamic
```

- Compile the example source code (rcdemo.c) for **NuMaker-RTU-NUC980 (Chili).**

```
$ $CC rcdemo.c -o rcdemo -Ibuildroot_2024/output/staging/usr/include \
  -DNUMAKER_BOARD=2 -Lbuildroot_2024/output/staging/usr/lib -Wl,-Bstatic \
  -lpaho-mqtt3c -Wl,-Bdynamic
```

- Copy executable file to the root user's home directory.

```
$ cp rcdemo buildroot_2024/output/target/root
```

- Rebuild the image to include rcdemo executable.

```
$ cd buildroot_2024
$ make
```

- After build is completed, retrieve the following files from the Buildroot subdirectories.

Files	IIoT-NUC980 IIoT-NUC980G1	IIoT-NUC980G2	RTU-NUC980 (Chili)
./output/images/**u-boot.bin**	●	●	●
./output/build/uboot-custom/spl/**u-boot-spl.bin**	●	●	
./output/images/**nuc980-iot-v1.0.dtb**	●		
./output/images/**nuc980-iot-g2-v1.0.dtb**		●	
./output/images/**nuc980-chili.dtb**			●
./output/images/**uImage**	●	●	●

- Next, use a plain text editor to create the *env.txt* file with the following content. For convenience, *env.txt* is also included in the example ZIP file.

```
baudrate=115200
bootdelay=1
stderr=serial
stdin=serial
stdout=serial
setspi=sf probe 0 30000000
loadkernel=sf read 0x7fc0 0x200000 0xC00000
loaddtb=sf read 0x1800000 0xE00000 0x10000
bootcmd=run setspi;run loadkernel;run loaddtb;bootm 0x7fc0 - 0x1800000
```

5.2 Programming Firmware to Board

- Download the **NuWriter** tool to program firmware to board.
  - Clone or download the ZIP from https://github.com/OpenNuvoton/NUC980_NuWriter/
  - NuWriter Document is the PDF file in ".\" directory.
  - NuWriter Windows Driver is located in ".\NuWriter\Driver" directory. If not installed, please install it first.
  - NuWriter Tool is located in ".\NuWriter\NuWriter\Release" directory. No need to be installed.
- Open Board Console
  - Use any terminal tool on your PC or laptop, such as TeraTerm or Putty, to open a terminal for the NUC980 VCOM, labeled "USB Serial Device".
  - Select baud rate to 115200, non-parity, 8-bit, one stop bit.
- Program Firmware for **NuMaker-IIoT-NUC980** boards.
  - Set NuMaker-IIoT-NUC980G1 or NuMaker-IIoT-NUC980G2 board to boot from USB.
  - Set both **No.1** and **No.2** switches on **SW1** to **ON**.

- Press **Reset** (**K3**) button on board to boot from USB.
- Use the **NuWriter** to program files with the following types and addresses.
- DDR parameter: Select **NUC980DK61Y.ini** / **NUC980DK63YC.ini** / **NUC980DK71YC.ini** according to the NUC980 part number on board.
- Choose Type: **SPI NAND**.

Files	Type	Address
u-boot-spl.bin	Loader	0x200
u-boot.bin	Data	0x100000
uImage	Data	0x200000
nuc980-iot-v1.0.dtb (NUC980/G1) nuc980-iot-g2-v1.0.dtb (NUC980G2)	Data	0xE00000
env.txt	Environment	0x80000

- Program Firmware for **NuMaker-RTU-NUC980 (Chili)** board.
  - Set NuMaker-RTU-NUC980 (Chili) board to boot from USB.
  - Set both **No.1** and **No.2** switches on **SW1** to **ON**.
  - Press **Reset** button on board to boot from USB.
  - Use the **NuWriter** to program files with the following types and addresses.
  - DDR parameter: Select **NUC980DR61YC.ini** / **NUC980DR63YC.ini** according to the NUC980 part number on board.
  - Choose Type: **SPI NOR**.

Files	Type	Address
u-boot.bin	Loader	0xE00000
uImage	Data	0x200000
nuc980-chilli.dtb	Data	0xE00000
env.txt	Environment	0x80000

- Set the NuMaker NUC980 boards to boot from SPI Flash:
  - Set both **No.1** and **No.2** switches on **SW1** to **OFF**.
  - Press the **Reset** button to reboot the board from SPI Flash.
- Log in using the “root” account.

5.3 Configuring MQTT App

- Execute IoT MQTT Panel App on Android mobile phone.
- Add a new connection and enter the MQTT broker information.
  - **Connection name**: Give this connection a name, such as **NUC980-MQTT-Test**
  - **Client ID**: Leave blank and it will be automatically generated.
  - Broker Web/IP address: Fill in test.mosquitto.org

- Port: 1883
- Network protocol: TCP
- Click **+** to add a dashboard, and assign a dashboard name, such as **NUC980 Test Dashboard**
- Then tap **CREATE**

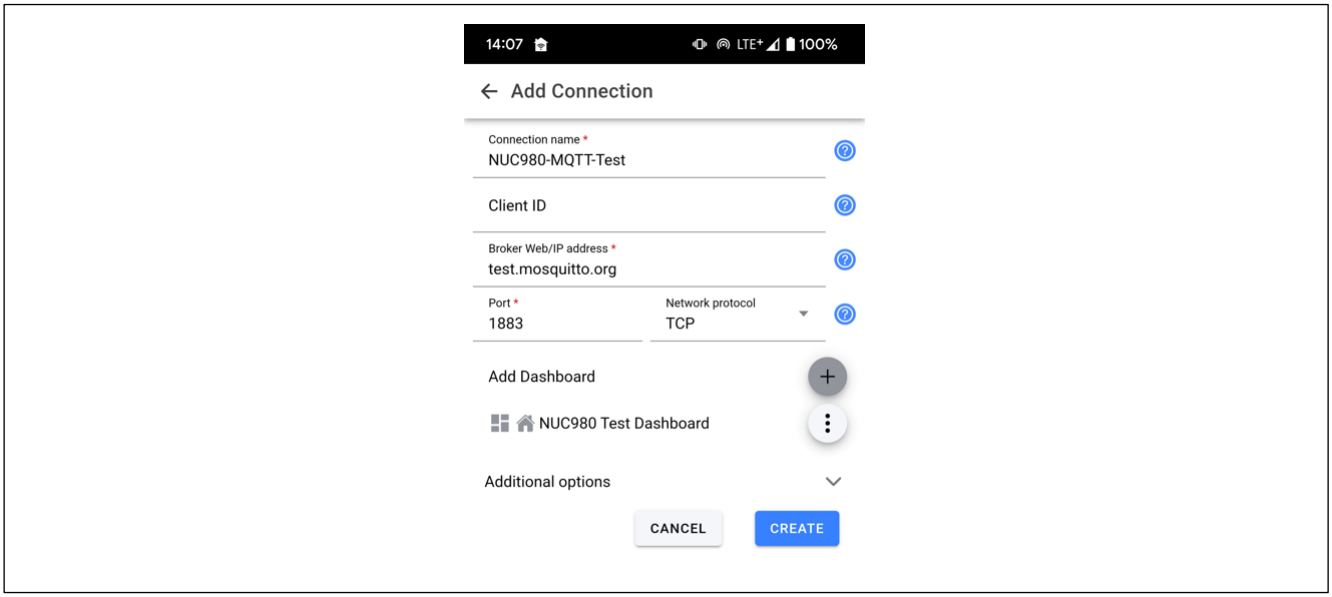


Figure 5-1 Create a New Connection through MQTT App

- Back to App main screen, click Connection Name (i.e., NUC980-MQTT-Test in the demo) to add panels.
- Add 4 panels for the connection.

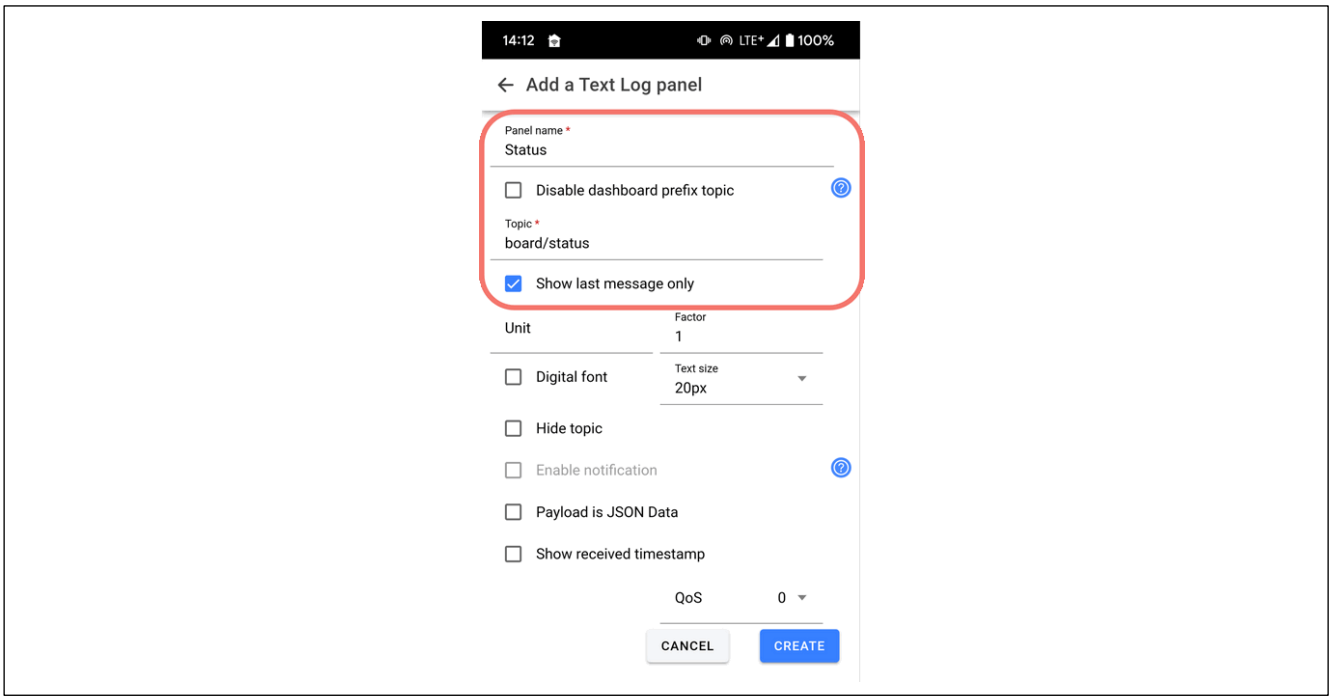


Figure 5-2 Add the First Panel through MQTT App

- Add the first panel.
  - ♦ Select Text Log
  - ♦ Panel name: Status
  - ♦ Topic: board/status
  - ♦ Check Show last message only
  - ♦ Then tap CREATE

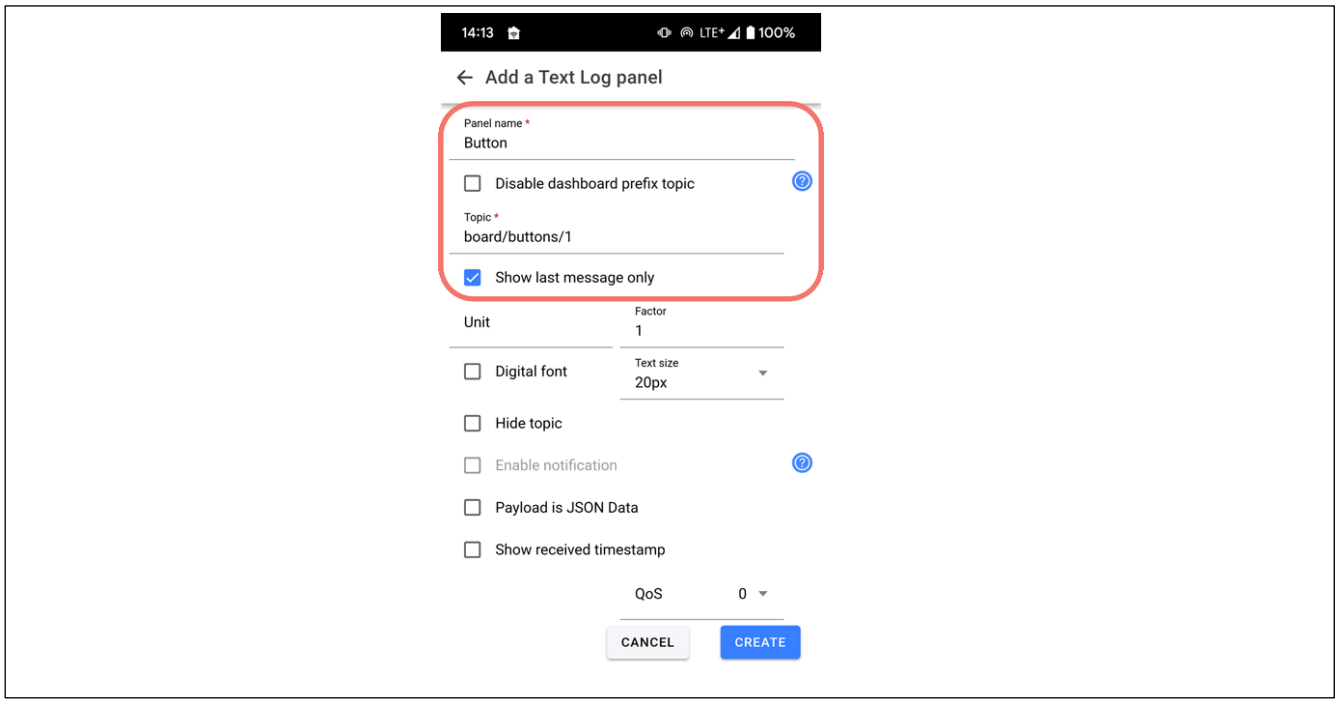


Figure 5-3 Add the Second Panel through MQTT App

- Add the second panel.
  - ♦ Select Text Log
  - ♦ Panel name: Button
  - ♦ Topic: board/buttons/1
  - ♦ Check Show last message only
  - ♦ Then tap CREATE

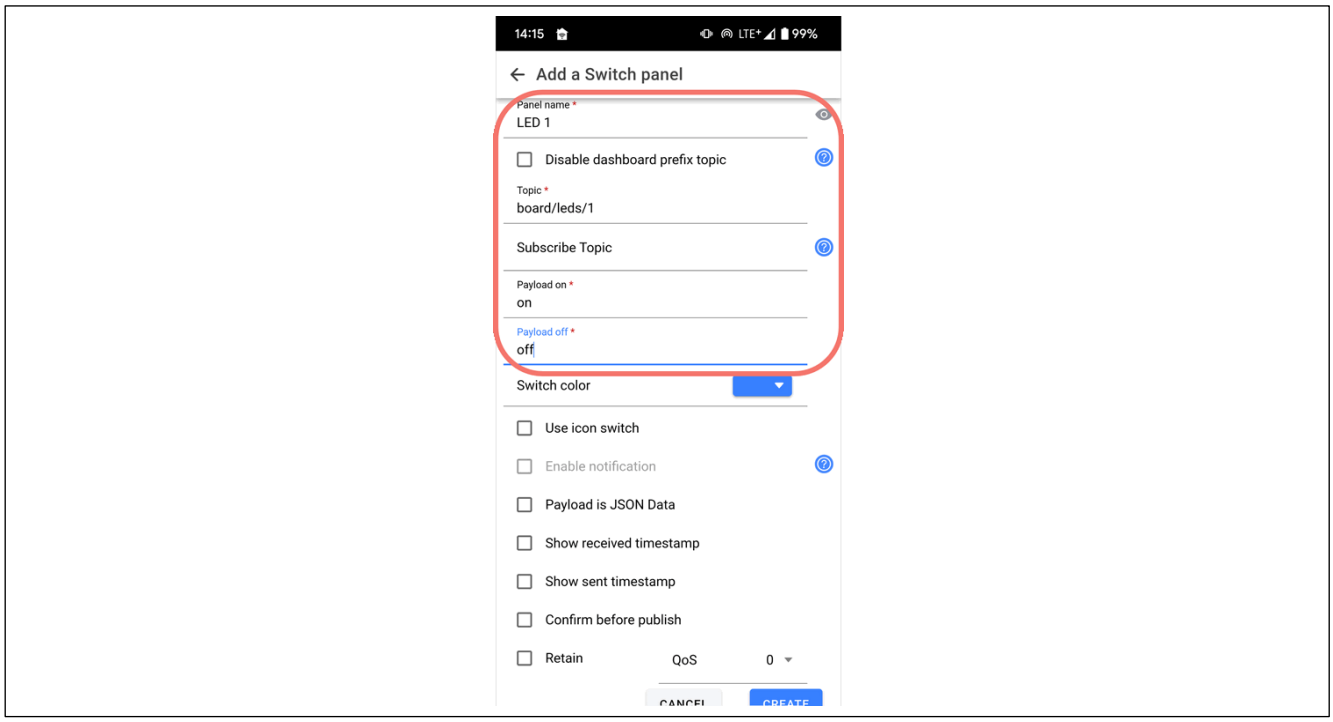


Figure 5-4 Add the Third Panel through MQTT App

- Add the third panel.
  - ♦ Select Switch
  - ♦ Panel name: LED 1
  - ♦ Topic: board/leds/1
  - ♦ Payload on: on
  - ♦ Payload off: off
  - ♦ Then tap CREATE

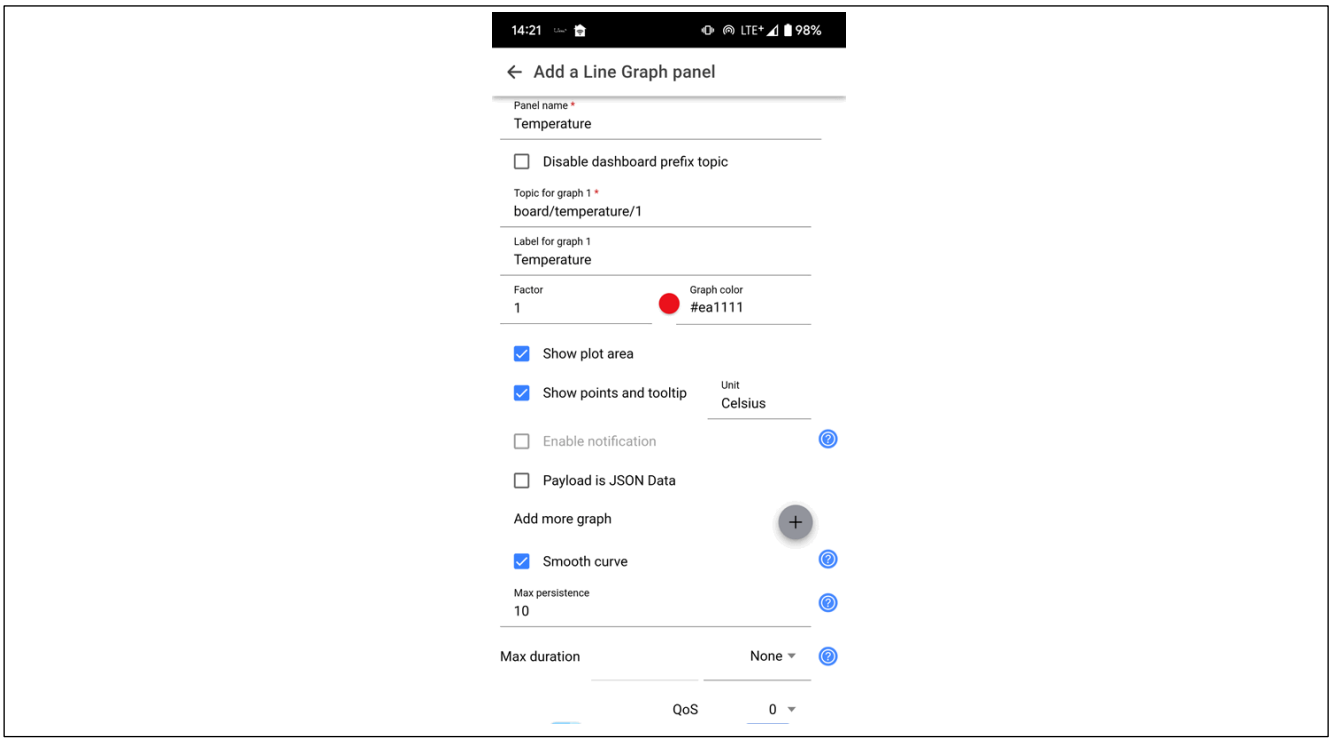


Figure 5-5 Add the Fourth Panel through MQTT App

- Add the fourth panel.
  - ♦ Select Line Graph
  - ♦ Panel name: Temperature
  - ♦ Topic: board/temperature/1
  - ♦ Label for graph 1: Temperature
  - ♦ Factor: 1
  - ♦ Graph color: #ea1111
  - ♦ Check Show plot area, Show points and tooltip, and Smooth curve
  - ♦ Unit: Celsius
  - ♦ Max persistence: 10
  - ♦ Then tap CREATE

6. Conclusion

This demonstration highlights how the Nuvoton NuMaker NUC980 boards, running Embedded Linux, enables remote control via the MQTT protocol and a mobile IoT control panel App.

Using C and the Paho MQTT library, the demo effectively demonstrates how to remotely control onboard peripherals (such as LEDs and buttons) and monitor environmental data (like temperature).

This project demonstrates the seamless integration of hardware, software, and cloud services, providing a practical solution for IoT-based remote control applications.

7. Revision History

Date	Revision	Description
2025.06.21	1.00	Initial version.

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.