Using AWS IoT Door Lock Reference Application with RAFW Connectivity Devices

This document explains using the reference application for door lock with the AWS IoT Core and a mobile application to control the door lock.

Contents

1. Introduction **3**
2. System Overview **4**
2.1 Software Requirements 4
2.2 Hardware Requirements 4
3. Hardware Setup and Configuration **5**
4. Reference Application Setup **6**
4.1 Set UP RA6W1 Wi-Fi Provisioning 6
4.2 Prepare AWS Account 6
4.3 Program Firmware Image 6
5. Run AWS IoT Door Lock Example **7**
5.1 Provision the Wi-Fi 7
5.1.1 Wi-Fi Provisioning in RA6W1 7
5.1.2 Wi-Fi Provisioning in RA6W2 7
5.2 Door Lock Application 9
6. Revision History **14**

Figures

Figure 1. Architecture of AWS IoT reference application 3
Figure 2. Configure ThingName in NVRAM 7
Figure 3. Renesas Wi-Fi Provisioning application steps 8
Figure 4. Open AWS IoT Door Lock app 9
Figure 5. Open the door 9
Figure 6. Message flows of opening door 10
Figure 7. Opening door on mobile app 11
Figure 8. Message flows of closing door 12
Figure 9. Closing door on mobile app 12

© 2026 Renesas Electronics

Terms and Definitions

AP	Access Point
AT	Attention (AT commands)
AWS	Amazon Web Services
EVB	Evaluation Board
FSP	Flexible Software Package
IDE	Integrated Development Environment
IoT	Internet of Things
MCU	Microcontroller Unit
MQTT	Message Queuing Telemetry Transport
PC	Personal Computer
SSID	Service Set Identifier
STA	Station Mode
USB	Universal Serial Bus
Wi-Fi®	Wireless Fidelity

1. Introduction

To get started with AWS IoT connectivity on the RA6W1/RA6W2, begin with a ready-to-use reference application. The provided application project, awsiot_wifi_lock_ek_ra6w1 and awsiot_wifi_ble_lock_ek_ra6w2, includes pre-configured project settings, middleware, and example code, which help save time and ensures that the environment is correctly set up for AWS IoT development. It shows how an application can be created that implements a cloud-connected door lock system in RA6W1/RA6W2 using the AWS IoT platform.
TheRA6W1/RA6W2 connects to a Wi-Fi network and establishes a secure MQTT connection to the AWS IoT Core. The actual door lock hardware is expected to be controlled by the RA6W1/RA6W2. The mobile application communicates with the AWS IoT Core through the internet to remotely control the door lock through theRA6W1/RA6W2.
The following items are the main components of the AWS IoT reference application (see Figure 1):

- RA6W1/RA6W2 evaluation board
- Router: Connection to internet
- Mobile device: Android/iOS application
- AWS IoT Cloud



Figure 1. Architecture of AWS IoT reference application

For more information about RA6W1, see the RA6W1 Datasheet and RA6W1/RRQ61001 Quick Start Guide.
For more information about RA6W2, see the RA6W2 Datasheet and RA6W2/RRQ61051 Quick Start Guide.

2. System Overview

The example project demonstrates how to create an application in the RA6W1/RA6W2 using the AWS IoT platform. It implements a cloud-connected door lock system.
The RA6W1/RA6W2 connects to a Wi-Fi network and establishes a secure MQTT connection to the AWS IoT Core. The actual door lock hardware is expected to be controlled by the RA6W1/RA6W2.
The mobile application connects to the AWS IoT Core over the internet using MQTT. In MQTT, devices communicate using topics, which are like channels or message paths. The mobile app sends (publishes) commands to a specific topic, and the RA6W1/RA6W2 door lock listens (subscribes) to that topic to receive and act on those commands.
This example also uses the Wi-Fi provisioning method to assign the AP details for internet access to the RA6W1/RA6W2 using the same mobile application. In this same process, the mobile application also collects the AWS application mode (AT/non-AT) along with the Thing Name of the device that the user configures.
For provisioning in RA6W1, users must configure it in Provisioning mode. This is explained in the following sections. In Provisioning mode, the RA6W1 temporarily creates a Wi-Fi AP with a predefined SSID. Mobile applications connect to this access point and access a provisioning interface to configure network credentials. After provisioning is complete, the device reboots itself in STA mode, connects to the configured AP, and starts the normal functionality of door lock application.
For provisioning in RA6W2, the application uses Bluetooth LE to connect with the mobile application. This is explained in the following sections. In Provisioning mode, the RA6W2 creates a Bluetooth LE advertisement with a predefined name. Mobile applications connect to this Bluetooth LE and access a provisioning interface to configure network credentials. After provisioning is complete, the device reboots itself, connects to the configured AP, and starts the normal functionality for the door lock application.

2.1 Software Requirements

The following software tools and packages are required to complete the project:

- Renesas Electronics Flexible Software Package (FSP): Version 2.2.0 or later.
- e^2 studio IDE: Version 2026-04.3 or later.
- GCC Arm Embedded Toolchain: Version 13.3.1.arm-13-24.
- Serial Terminal Application: Tera Term or similar tools for logging and monitoring output.

2.2 Hardware Requirements

- Renesas RA6W1/RA6W2 Evaluation Kit (EVK).
- RA6W1/RA6W2 MCUwith ultra-low power consumption.
- USB Type-C Cable: For programming and debugging.
- Jumper Wire (Female to Female): For making proper motherboard configurations for the application.
- Host PC: For programming, viewing console logsFor programming, viewing console logs, opening MQTT test client and giving AT commands.
- Wi-Fi Access Point (AP): With network connectivity.

3. Hardware Setup and Configuration

To connect hardware:

1. Connect the RA6W1/RA6W2 MCU to the Evaluation Kit (EVK) using the extension socket.
2. Connect the USB Debug port on the EVK to the host PC for programming and debugging using a USB Type-C cable.
3. Set up jumpers as following:
   - UART (RX): P0_00
   - UART (TX): P0_01
   - J102: {2-3} [Current Measurement]
   - J105: {2,3} [Voltage Selector]
   - J106: {2,3} [Voltage Selector]
   - J107: {2,3} [Voltage Selector]
   - J210: {1,2} [Reset]
   - SWCLK(c): SWCLK
   - SWDIO(c): SWDIO
4. Additionally for RA6W1, connect BTN1 and P0_10. This configuration is used for switching the device to Wi-Fi Provisioning mode (SoftAP mode).

4. Reference Application Setup

Before starting the application, ensure your hardware environment is set up, your AWS account is prepared, and the firmware is programmed. See the following sections to finalize the application setup.

4.1 Set UP RA6W1 Wi-Fi Provisioning

Follow the steps in the SoftAP-Based Wi-Fi Provisioning Setup section in *Wi-Fi Provisioning with RAFW Connectivity Devices*.

4.2 Prepare AWS Account

Follow the steps in the Connect Devices to AWS IoT section in *RA6W1/RRQ61001 AWS IoT Core Quick Start Guide* or *RA6W2/RRQ61051 AWS IoT Core Quick Start Guide*.
Make changes in the following properties of AWS Stack in `configuration.xml` of the project according to the AWS configurations done here.

- AWSIOT_W Thing Name
- AWSIOT_W Host Name
- AWSIOT_W ROOT CA Certificate
- AWSIOT_W Client Certificate
- AWSIOT_W Client Private Key.

4.3 Program Firmware Image

The firmware image for the reference application is generated and flashed onto the EK-RA6W1/RA6W2 using e^2 studio with J-Link support. e^2 studio provides an integrated build and flash environment, eliminating the need for external flash programming scripts.
To flash the firmware image:

2. Open e^2 studio and connect the EK-RA6W1/RA6W2 to the PC through USB (using the inbuilt SEGGER J-Link).
3. Load the reference application to e^{2}studio. The reference application for AWS IoT Door Lock is in the path `application_projects/r19an0426/AWS_Reference_Application/`. For more information on how to load the template, see the Load Project Template in e^2 Studio section in the *RA6W1/RRQ61001 AWS IoT Core Quick Start Guide* or *RA6W2/RRQ61051 AWS IoT Core Quick Start Guide*.
4. Edit the properties of AWS Stack based on the configurations made in Section 4.2 Prepare AWS Account. For more information on how to edit AWS Stack Properties, see the Load AWS-IoT Reference Application in e^2 Studio section in *RA6W1/RRQ61001 AWS IoT Core Quick Start Guide* or *RA6W2/RRQ61051 AWS IoT Core Quick Start Guide*.
5. Flash the image using debugger. For more information, see the Flash and Debug using e^2 Studio section in the *RA6W1/RRQ61001 Application Development Guide*.

   When flashing is complete, the device starts running the firmware.

5. Run AWS IoT Door Lock Example

5.1 Provision the Wi-Fi

Provisioning a Wi-Fi device is the process of setting up the device to connect to a network and function properly. This involves configuring network settings, ensuring secure authentication, and assigning necessary parameters. The provisioning can be done with the Renesas Wi-Fi Provisioning application on either an Android or an iOS device.

5.1.1 Wi-Fi Provisioning in RA6W1

When the RA6W1 initially starts after flashing the image, it is in Station mode. To change the device into Provisioning mode and to provision the Wi-Fi:

1. Press the factory reset button configured using the jumper connection in .

   The device performs the factory reset and bootup in Soft-AP mode and is ready for Wi-Fi provisioning.
2. Modify the default AWS `thingname` if needed, using the `cli` command, `nvram setenv appcfg thingname <THINGNAME>`.

```
[/RA6W1-RRQ61001/nvram] # nvram setenv appcfg thingname RA6Wx_Demo_ThingName

[/RA6W1-RRQ61001/nvram] #
[/RA6W1-RRQ61001/nvram] # nvram printenv appcfg thingname

appcfg(gid:1301):

        Tag_Name(ID)                                    [Used/MaxLen] [Data]
        ------------------------------------------------------------------------
        thingname(1452) ............................... [20/64] RA6Wx_Demo_ThingName
Total 1/215
```

Figure 2. Configure ThingName in NVRAM

3. To complete Wi-Fi Provisioning, follow the steps in the Provision Wi-Fi through Soft AP Mode section in *Wi-Fi Provisioning with RAFW Connectivity Devices*.

Wi-Fi Provisioning is now completed and the RA6W1 reboots and starts as a Station and connects to the Access Point configured from the application. After the successful connection to the internet, it connects to the AWS server.

5.1.2 Wi-Fi Provisioning in RA6W2

1. Configure the device Thing Name as mentioned in Figure 2.
2. Open the same Renesas WiFi Provisioning app mentioned in *Wi-Fi Provisioning with RAFW Connectivity Devices*.
3. Select **Start provisioning over Bluetooth Low Energy**.
4. Select the device **RRQ-DEVICE** from the scanned Bluetooth LE devices.
5. Next, the application reads the AWS IoT Thing Name and operation mode (AT or non-AT) and connects to Bluetooth® LE device.
6. Select **Scan Wi-Fi network**.
7. Select the required Access Point from the scanned list.
8. Enter the password and click **OK**.
9. Select **Connect to <AP Name>**.
10. After the connection attempt, the application provides a network check result. Review the details and select **Complete**.
11. The device reboots and connects to the configured Access Point.

Figure 3 explains the preceding steps using the screenshots from the mobile application.



Figure 3. Renesas Wi-Fi Provisioning application steps

5.2 Door Lock Application

When provisioning is completed, to open the AWS application on a mobile device, select **AWS IoT** by clicking the menu icon (see Figure 4).

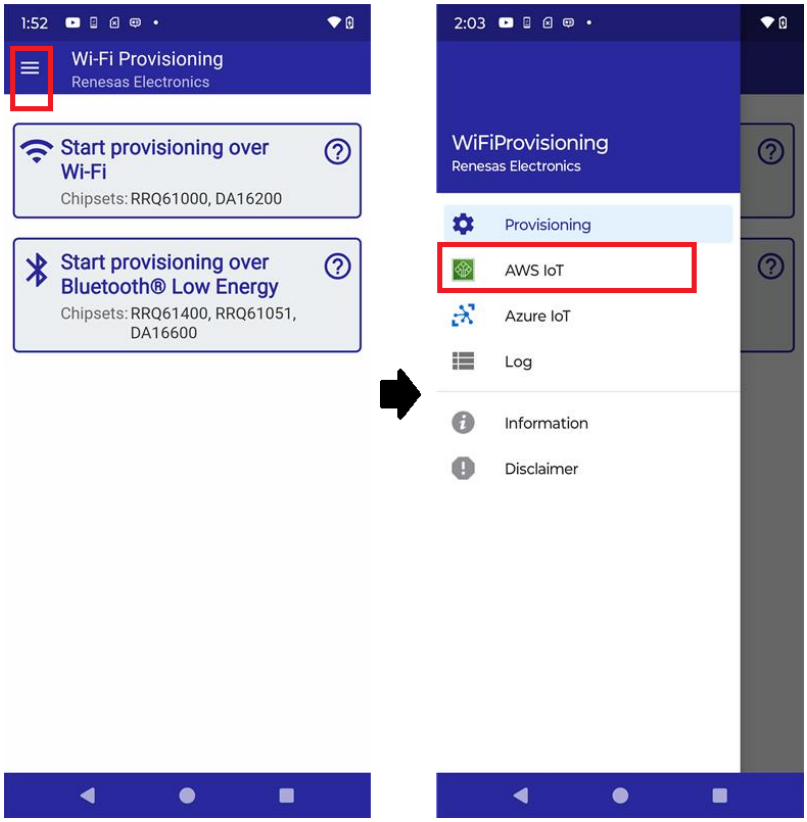


Figure 4. Open AWS IoT Door Lock app

In this application, you can open and close the door by clicking the Lock icon (see Figure 5).



Figure 5. Open the door

The following is the flow for the door opening and closing process:

- When RA6W1/RA6W2 boots up and connects to AWS cloud, it subscribes to the `thingName/AppControl`topic.
- In this stage, when the mobile application boots up and connects to AWS cloud, it is publishing the message "Connected" to the `thingName/AppControl` topic. It also is subscribing to the `thingName/DeviceControl` and the `thingName/DeviceConnect` topics.

- When this "Connected" message is received in RA6W1/RA6W2, it sends the "yes" message to the `thingName/DeviceConnect` topic. It also publishes the initial door state as "closed" to the `thingName/DeviceControl` topic .
- When this "yes" message is received in the mobile application, it knows the device is connected, so the icon of the Device in the UI turns to solid green. Also, when the "closed" message is received, it changes the lock image to closed in the UI.
- Now when the lock icon is pressed, it sends the message, "doorOpen <Number> <Date> <Time>" to the `thingName/AppControl` topic.
- When this message is received in RA6W1/RA6W2, it sends a message "opened" to the `thingName/DeviceControl` topic (When an interface occurs with actual door lock hardware, RA6W1/RA6W2 should do the door unlock process and send the message "opened").
- When this "opened" message is received in the application, it updates the UI by changing the lock image to open state.
- Similarly for door closing, when the lock icon is pressed, it sends the message, "doorClose <Number> <Date> <Time>" to the `thingName/AppControl` topic.
- When this message is received in RA6W1/RA6W2, it sends a message "closed" to the thingName/DeviceControl topic (When we interface with actual door lock hardware, RA6W1/RA6W2 should do the door lock process and send the message "closed").
- When this "closed" message is received in the application, it updates the UI by changing the lock image to close state.

The following is the flow of the updating device shadow process:

- To update the device shadow, the refresh button in the application is clicked.
- Next, the application sends the message "updateSensor" to the `thingName/AppControl` topic.
- When this message is received inRA6W1/RA6W2, it updates the sensor data and publishes the data to the `$aws/things/<ThingName>/shadow/update` shadow topic. It also publishes the message "updated" to the `thingName/DeviceControl` topic. (Currently, the part of reading the sensor data like temperature and battery is not available in the reference application.)

When the message, "updated" is received, the application reads the shadow data from the `$aws/things/<ThingName>/shadow/update` topic and updates the value of temperature and battery in the UI. Figure 6 shows the message flow of opening the door.

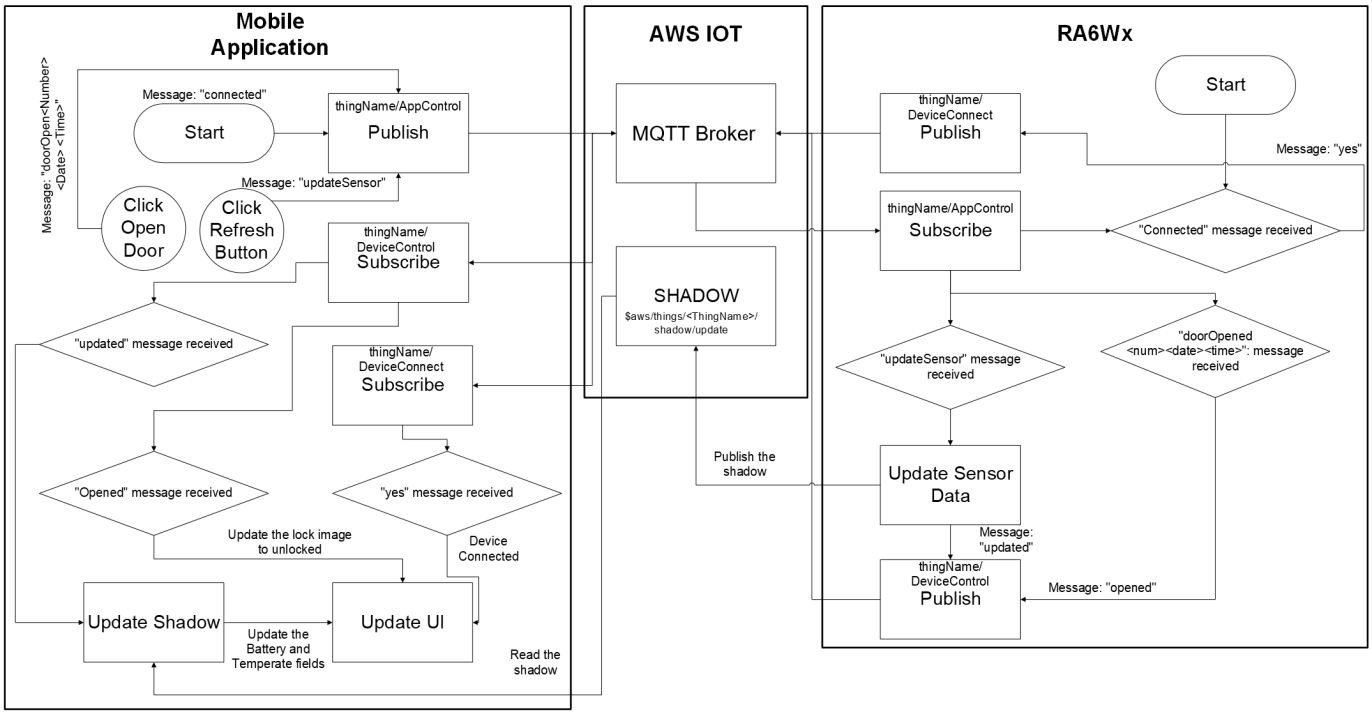


Figure 6. Message flows of opening door

Figure 7 shows the operation of opening the door on the Android application.

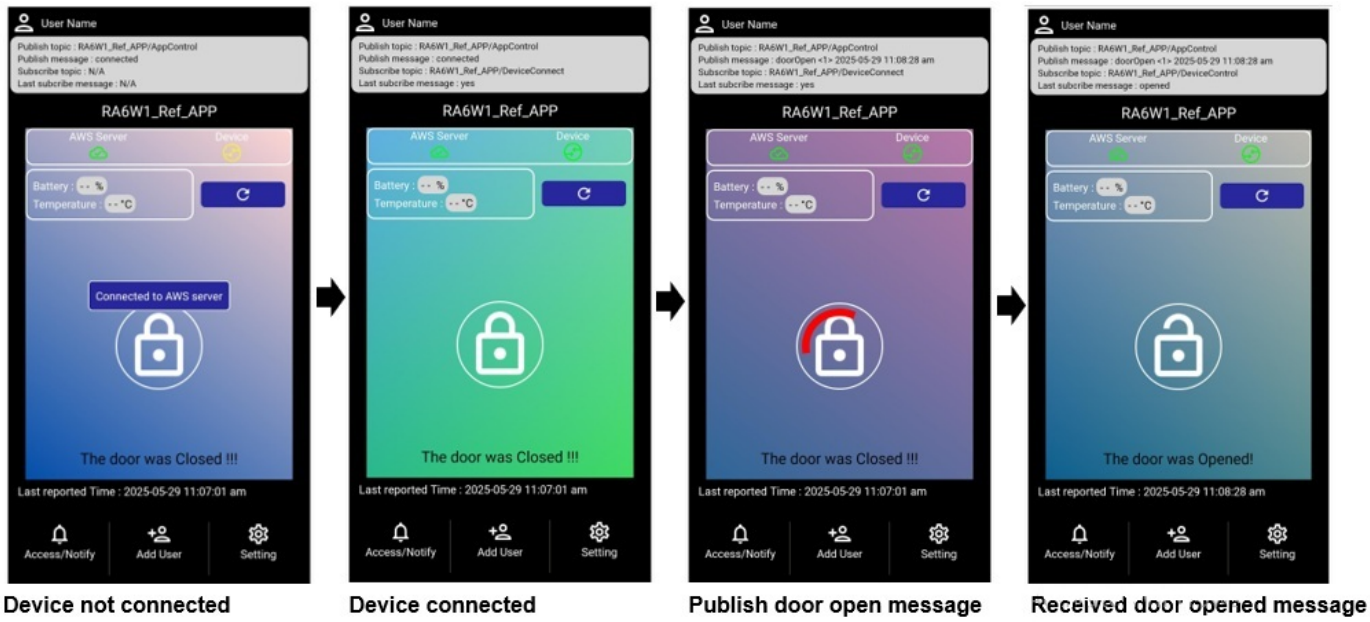


Figure 7. Opening door on mobile app

When the operation of opening door is completed, the console logs of the RA6W1 appear as follows:

```
open comm
[openControl]

DEBUG:   [aws_dpm_app_door_work:1961] previous MQTT result = 0, doorLock CMD (=1: 0-idle, 1-
open, 2-close, 3-auto close)

================================================================================

********************************************************************************

last user Timer ID = 5
last doorOpenFlag state: "true"
last FOTA Stat: 0
last FOTA Url: ""

Sleep mode DPM: KA timer interval(=1800 sec)

[DPM_App_Main] DM_FINISH_DEVICE
recv timeout(=19 ms) set OK (socket=0)
[dpm_keepalive_timer_unregister] OK to deregister AWS DPM's timer(5)
[dpm_keepalive_timer_register] OK to delete AWS DPM's timer(5)
[dpm_keepalive_timer_register] OK to register timer: interval = 1770(sec), tid = 5
>>> Start DPM Power-Down !!!
```

As shown in the log, note that when the Door is opened, the `doorOpenFlag` state changes to "true". Also, `open comm` and `[openControl]` prints can be seen.

Figure 8 shows the message flows of closing the door.

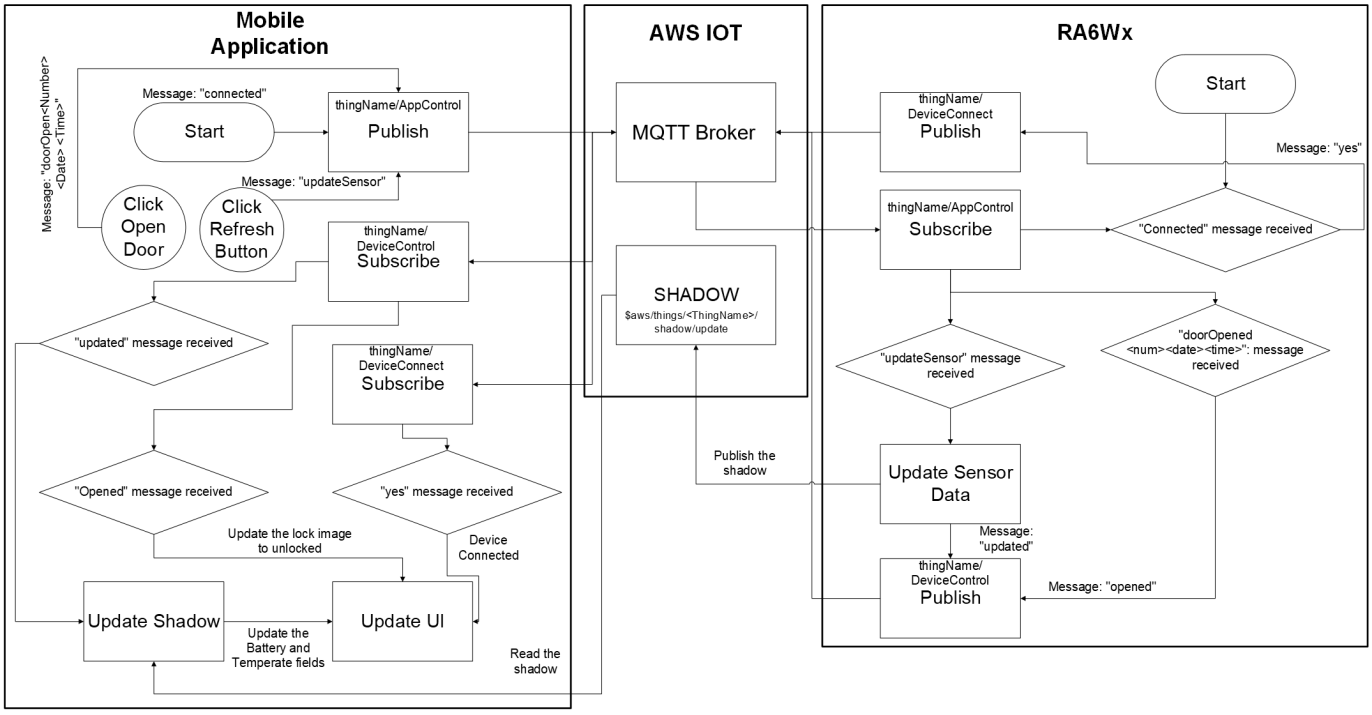
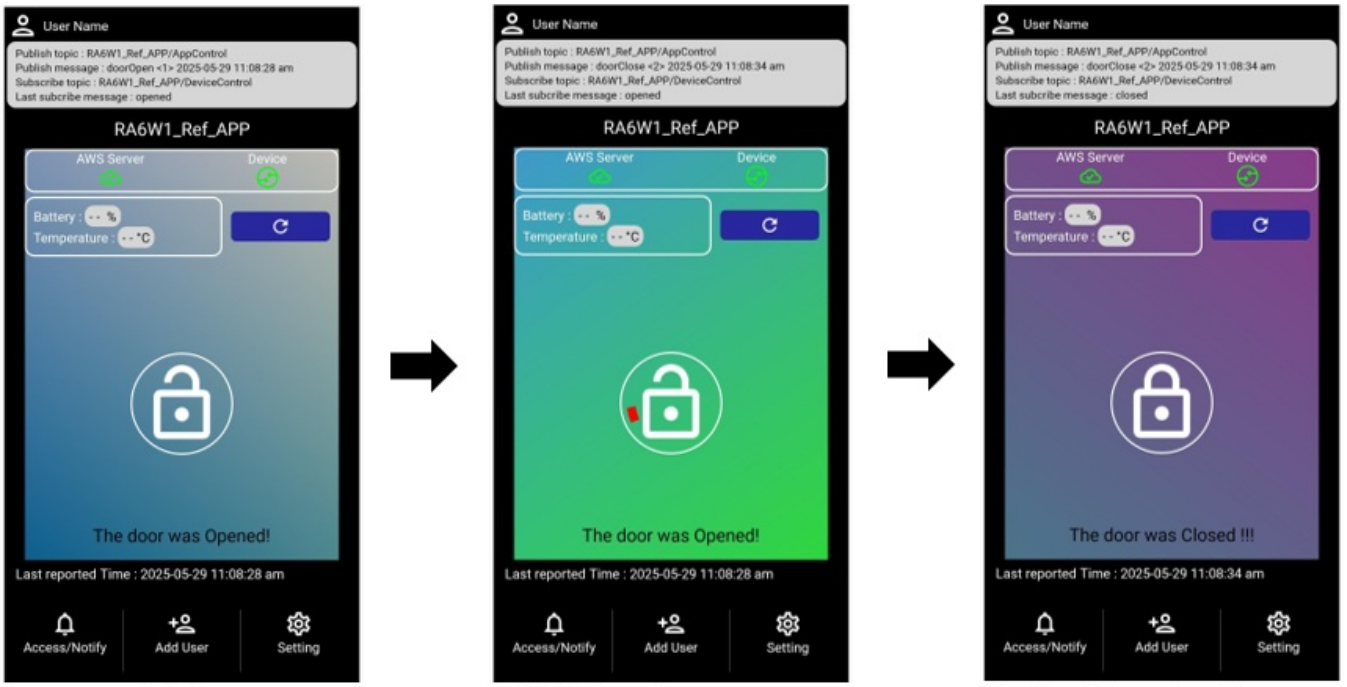


Figure 8. Message flows of closing door

Figure 9 shows the operation of closing the door in Android application.



Door opened state **Publish door close message** **Received door closed message**

Figure 9. Closing door on mobile app

When the operation of closing the door is completed, the console logs of the RA6W1 appears as follows:

```
close comm
[closeControl]

DEBUG:   [aws_dpm_app_door_work:1961] previous MQTT result = 0, doorLock CMD (=2: 0-idle, 1-
open, 2-close, 3-auto close)

================================================================================

********************************************************************************

last user Timer ID = 5
last doorOpenFlag state: "false"
last FOTA Stat: 0
last FOTA Url: ""

Sleep mode DPM: KA timer interval(=1800 sec)

[DPM_App_Main] DM_FINISH_DEVICE
recv timeout(=19 ms) set OK (socket=0)
[dpm_keepalive_timer_unregister] OK to deregister AWS DPM's timer(5)
[dpm_keepalive_timer_register] OK to delete AWS DPM's timer(5)
[dpm_keepalive_timer_register] OK to register timer: interval = 1770(sec), tid = 5
[aws_dpm_app_finish_loop] break finish_loop...
>>> Start DPM Power-Down !!!
```

As shown in the log, note that when the Door is opened, the `doorOpenFlag` state changes to "false". Also, `close comm` and `[closeControl]` prints can be seen.

6. Revision History

Revision	Date	Description
1.02	Sep 21, 2026	Updated for FSP version 2.2.0, Reference document name updated.
1.01	July 21, 2026	Updated for FSP version 2.1.0
1.00	May 29, 2026	Initial release

IMPORTANT NOTICE AND DISCLAIMER

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

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

(Disclaimer Rev.1.01)

CorporateHeadquarters

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

www.renesas.com

Trademarks

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

Contact Information

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

© 2026 Renesas Electronics Corporation. All rights reserved.