User Guide: Alibaba Cloud (Aliyun) iotkit Integration of Z3GatwayFreeRTOS for LCGW

This article provides a guidance of integrating Alibaba Cloud (Aliyun) iotkit-embedded SDK souce codes on Realtek's AmebaD Wi-Fi SoC SDK.

Realtek IoT Wi-Fi SoC RTL872xCS/DN serves as the targeted NCP host porting hardware platfrom of Z3GatewayFreeRTOS with Silicon Labs's EFR32MG21/13 as the NCP.

Please note that this document is aligned with the Zigbee 3.0 RTOS gateway sample codes running on the Low-cost Gateway (LCGW) development board.

KEY FEATURES

- Source Codes and Compilation
- Device Authentication
- Mobile Application Usage
- TMall Genie Conrol Zigbee Devices
- Port Alibaba iotkit from Scratch

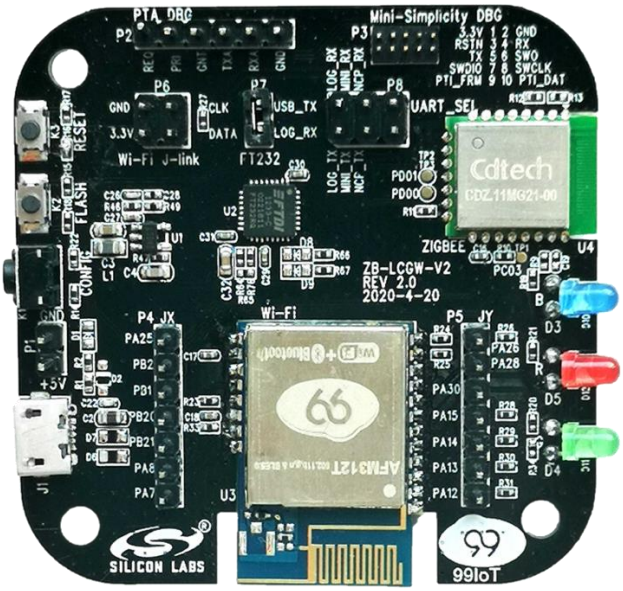
Table of Contents

1. Source Codes and Compilation

1.1 Prerequisite

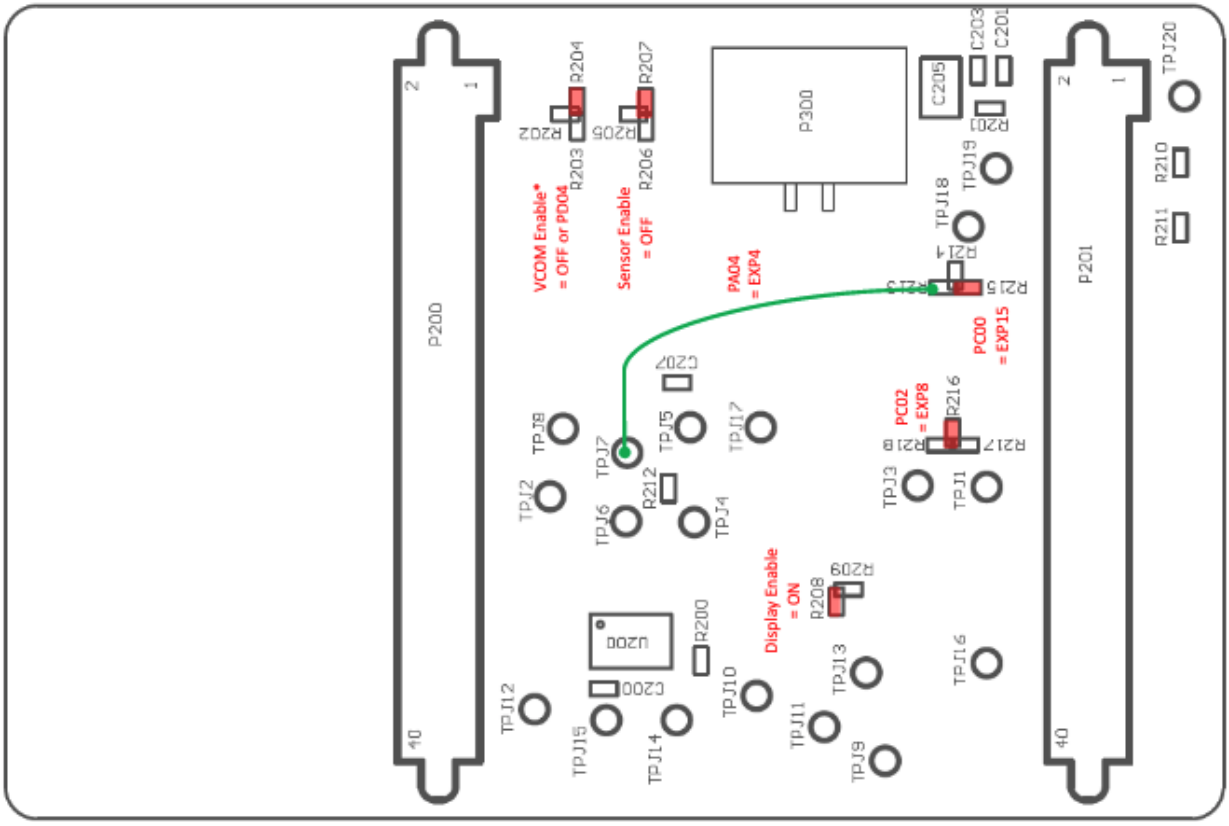
Silicon Labs' Z3GatewayFreeRTOS sample app is targeted to run on Realtek's RTL872xCS/DN Ameba series of Wi-Fi/BLE combo SoCs which commonly rely on Realtek's AmebaD SDK for Wi-Fi application development.

This Aliyun Zigbee Gateway example has been ported and verified on the Low-cost Gateway (LCGWv2) kit hardware. The LCGWv2 kit has integrated Silicon Labs' EFR32MG21 Zigbee module from CDTech and Realtek's RTL8720CSM Wi-Fi/BLE module from 99IoT.

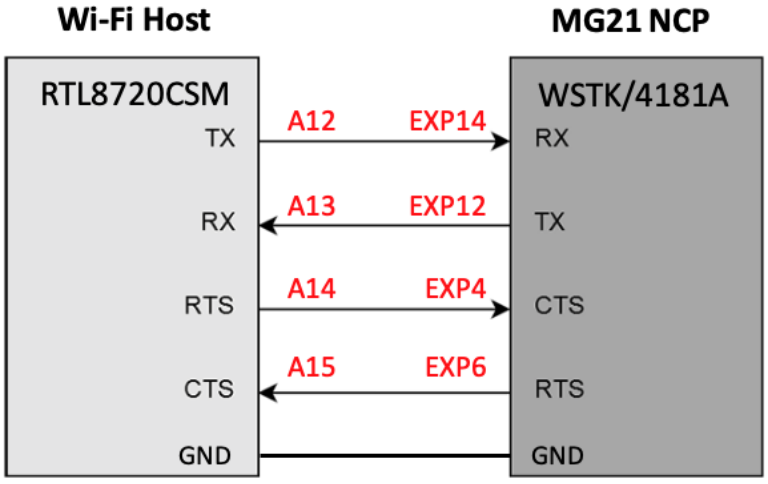


Alternatively, the RTOS gateway can be prototyped using Silicon Labs WSTK with BRD4181A (MG21) radio board plus Realtek's official RTL8720CSM (QFN48) development board, where the RTL8720CSM board can be purchased from Wi-Fi module partner Shenzhen 99IOT from Taobao:
https://item.taobao.com/item.htm?&id=603250166829
Realtek official Ameba IoT website also provides the links to buy the AmebaD development boards:
https://www.amebaiot.com/en/where-to-buy-link/

Rework the BRD4181A radio board as below:



Add below wiring connections between the AmebaD development-board and the MG21 WSTK. PTA connections may be added: i.e. REQUEST A26–EXP15, GRANT A27–EXP13 and PRIORITY B1–EXP11



Please note that AmebaD SDK gcc version released by Realtek has both public and NDA versions. Our Z3GatewayFreeRTOS sample app is built and verified on the public version AmebaD SDK. *Silicon Labs has worked with customer who has the NDA version AmebaD SDK and the SW-PTA patch. PTA functionality and performance of EFR32 NCP Zigbee and AmebaD Wi-Fi have been verified working good*. Custom may reach out Realtek, its official distributors or IDHs to obtain the NDA version Ameba SDK and the corresponding SW-PTA patch.

Please read Realtek's AmebaD SDK app note document "00016461-AN0400-Ameba-D-Application-Note-v12.pdf" which is downloadable at: https://www.amebaiot.com/en/sdk-download-manual-8722dm/ for setting up the Ubuntu Linux (recommended) or the Windows Cygwin gcc build environment and the basic build instructions.

1.2 Z3GatewayFreeRTOS Application

The Z3GatewayFreeRTOS Alibaba Cloud (Aliyun) gateway example package is a source-code patch overriding the AmebaD SDK and its default example project “Realtek_amebaD_va0_example”

This AmebaD SDK Aliyun gateway example project includes the Z3GatewayFreeRTOS sample generated by Silicon Labs’s Znet SDK as the Z3Controller library, i.e. lib_z3ctrl.a, also the ported version of Alibaba’s iotkit-embedded SDK as the iotkit library, i.e. lib_iotkit.a.

- lib_z3ctrl library source code is located at:
  project/5ealtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/make/project/lib_z3ctrl/
- lib_iotkit library source code is located at:
  project/5ealtek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/make/project/lib_iotkit/

The interactions between the Z3Controller and the IoT Kit are implemented in the high-level cloud gateway application located at:
project/5ealtek_amebaD_va0_example/src/src_hp/

This cloud gateway application implements below functions and more:

- Retrieve the Triple-unit-group info of the gateway itself and register gateway to Aliyun
- Retrieve the Triple-unit-group info of the underlying Zigbee child devices and register them to Aliyun
- Manage the underlying Zigbee child devices for registration/deregistration to Aliyun
- Dispatch the uplink/downlink events and messages between cloud and child devices
- Message format conversion between Zigbee Cluster Library (zcl) and Alibaba ICA TSL models (aka A-Link) formats

1.3 Alibaba Iotkit Embedded SDK

The source code of Alibaba’s Iotkit embedded SDK v3.0.1 stable version can be obtained from Alibaba’s GitHub repository:
https://github.com/aliyun/iotkit-embedded/tree/v3.0.1
Or downloaded from Alibaba web site to get the LTS v3.01 patched source code package:
https://help.aliyun.com/document_detail/96623.html

The ported version of this lib_iotkit code is based on Aliyun’s iotkit SDK V3.0.1 branch commit 82270c6 dated 2020-01-14

1.4 Realtek AmebaD SDK

There are two versions of Realtek’s AmebaD FreeRTOS SDK, namely the public version and the NDA version:

- Public version: github on https://github.com/ambiot/ambd_sdk
- NDA version: Require signed NDA to obtain, please contact Realtek, its distributors or its IDHs for details

Silicon Labs’ Z3GatewayFreeRTOS library and the Aliyun gateway example could be built on top of both the public version AmebaD SDK and the NDA SDK. The difference is that the RTL872xCS/DN Wi-Fi and EFR32MG13/MG21 Zigbee 2.4GHz ISM band PTA coexistence is only available on the AmebaD NDA SDK through a SW-PTA patch applied on it.

Please refer to Quick Start Guide: qsg_Z3GatewayFreeRTOS_Aliyun_gw_example_build_for_LCGW_rx.x.pdf for how to obtain the NDA version of AmebaD SDK and the corresponding SW-PTA patch and other critical patches, also on how to setup the build environment for the AmebaD SDK and our gateway host app example.

Note that the Zigbee 3.0 LCGW release package is targeted to be built onto the public version of AmebaD SDK only. It requires a slightly different LCGW release package in order to be buildable on the NDA version AmebaD SDK. Customer please contact Silicon Labs to obtain the corresponding LCGW release package specific to the NDA version AmebaD SDK.

1.5 Iotkit Folder Structure

The iotkit embedded SDK source is placed in the *GCC-RELEASE/project_hp/asdk/make/project/lib_iotkit/* folder as a customer library. See below the folder structure.

`Project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/asdk/make/project/`

```
└──lib_iotkit
    │   Makefile
    │   sdk_include.h
    │
    ├──certs
    │
    ├──CoAPPacket
    │
    ├──coap_server
    │   └──server
    │
    ├──dev_bind
    │   └──impl
    │       ├──awss_reset
    │       └──os
    │
    ├──dev_model
    │
    ├──dev_reset
    │
    ├──dev_sign
    │
    ├──infra
    │
    ├──mqtt
    │
    ├──ota
    │
    ├──wifi_provision
    │   ├──dev_ap
    │   └──frameworks
    │       ├──aplist
    │       ├──ieee80211
    │       ├──statics
    │       └──utils
    │
    └──wrappers
        │   wrappers_defs.h
        │
        ├──external_libs
        │   └──mbedtls
        │       └──include
        │           └──mbedtls
        │
        ├──freertos
        │
        └──tls
```

The folder *lib_iotkit/* contains the extracted iotkit-embedded SDK source codes while the FreeRTOS related wrapper functions are implemented in *lib_iotkit/wrapper/* folder. *Lib_iotkit* library's *Makefile* is added in *lib_iotkit/ f*older. Note that some modifications of the iotkit source code are required in order to resolve some symbol conflicts when builting in the AmebaD SDK.

The iotkit SDK is scalable and customizable. If more features need to be added, please refer to below link for further information and the "*Porting Alibaba iotkit-embedded to AmebaD SDK from scratch*" session for details.
https://help.aliyun.com/document_detail/111026.html

1.6 AmebaD Zigbee Gateway Project Compilation

The Zigbee 3.0 gateway Aliyun application source folder structure is shown below:

```
/project/Realtek_amebad_va0_example/
    |
    ├──inc_hp
    |       board_io.h
    |       build_info.h
    |       cloud_gateway.h
    |       cloud_gateway_interface.h
    |       FreeRTOSConfig.h
    |       main.h
    |       platform_autoconf.h
    |       platform_opts.h
    |       platform_opts_bt.h
    |
    ├──inc_lp
    |       build_info.h
    |       FreeRTOSConfig.h
    |       main.h
    |       platform_autoconf.h
    |       platform_opts.h
    |
    ├──src_hp
    |       board_io.c
    |       cloud_gateway.c
    |       cloud_gateway_interface.c
    |       main.c
    |       Makefile
    |       wifi_prov.c
    |
    └──src_lp
            main.c
            Makefile
```

Each AmebaD project consists of two Make targets, one for its KM0 baseband core and the other for the KM4 application core. The cloud gateway application is integrated into project_hp for the KM4 application target. The cloud gateway header files are placed in the *inc_hp* folder, the source files and the *Makefile* are placed in the *src_hp* folder.

- ***main.c*** implements main() which calls calls start_button_polling() and start_cloud_gateway() to start button polling and cloud gateway initialization tasks
- ***cloud_gateway.c*** defines the Triple-unit-groups for the GW and its child devices' registration to Cloud
  - Cloud gateway initization: initializes WDT, waits Wi-Fi connected, calls startGateway(), delays 3 seconds and forms the Zigbee network, then registers GW to cloud, creates cloud_msg_center and stack_msg_center tasks
  - Implements iotkit tasks user_dispatch_yield(), cloud_msg_center(), stack_msg_center() to handle cloud events and Zigbee devices management (register to cloud) + device control/report.
  - Implements button_evt_polloing_center() and calls button_evt_hdl() in cloud_msg_center task for handling
- ***could_gateway_interface.c***
  - Implements the cloud and Zigbee stack message/event queues dispatch handlers
  - Implements zcl <-> Alink data format conversion, interfaces with lib_z3ctrl library and calls its APIs
- ***wifi_prov.c*** contains initialization functions of Wi-Fi provisioning mechanisms via mobile phone App:
  - Alibaba's Wi-Fi provisioning by device-ap method
- ***board_io.c***
  - Defines RTL8720CS gpio ports for Zigbee NCP reset and boot control, Config button reading, and R/G/B LED control
  - Implements the Button polling and the LED blinking drivers
- *cloud_gateway.h* contains the compile options for a number of gateway features (see 1.6 for details)
- *platform_opts.h* contains the compile options of various AmebaD system features and application example features (see 1.6 for the newly added options)
- *platform_otps_bt.h* contains the compile options for AmebaD BLE roles and features on/off.

1.7 Gateway Compile Options

In *platform_opts.h*, some platform compilation options related to Z3GatewayFreeRTOS are defined below.

```
#define CONFIG_Z3_CONTROLLER    1   //1: Enable AmebaD SDK support code for Z3Controller
                                    //0: Disalbe AmebaD support codes
#define CLOUD_GATEWAY_ENABLE    1   //1: Running Cloud Gateway
                                    //0: Not running Cloud Gateway
#define LOGSEV_ZIGBEE_CLI_CMD   1   //1: Enable the Zigbee CLI in log_service
                                    //0: Disable the Zigbee CLI in log_service
#define LOGSEV_PROCESS_UP_MSG   0   //1: msg processed by log_service and cloud layer won't get the msg
                                    //0: msg processed by the cloud layer and some CLI z2 cmd won't work
```

In *cloud_gateway.h*, some compilation options for gateway app features are defined.

```
#define WIFI_PROVISION_ENABLE           1       ///< enable wifi provisioning for GW
#define WIFI_PROV_AWSS_DEV_AP_ENABLE    1       ///< wifi provisioning method: device AP, need
                                                     Aliyun's Intelligent Cloud App to go with
#define BOARD_IO_ENABLE                 1       ///< enable board io control
```

- The CONFIG_Z3_CONTROLLER compile option adds the Z3GatewayFreeRTOS library into AmebaD SDK and should always be 1.

- If CLOUD_GATEWAY_ENABLE is set to 0, the cloud feature is not included.
  When running the Aliyun Gateway example, we need to set `CLOUD_GATEWAY_ENABLE` to 1, and also LOGSEV_PROCESS_UP_MSG to 0 such that Z3GatewayFreeRTOS up messages will be processed by the cloud layer. Please note that some CLI z2 commands will not work if LOGSEV_ZIGBEE_CLI_CMD is set to 1 in this case.

- If LOGSEV_ZIGBEE_CLI_CMD is set to 1, the corresponding "Z2" and "Z3" Zigbee CLI commands will be added into AmebaD's log uart console. This is useful to test the low-level Z3Controller library features and the corresponding "Z2" and "Z3" commands. It is recommended to set LOGSEV_PROCESS_UP_MSG to 1 to have the full z2 command set and to set CLOUD_GATEWAY_ENABLE to 0 to have CLI only mode for feature testing.

- Setting `WIFI_PROVISION_ENABLE` to 1 to enable wi-fi provisioning for the gateway. Specific mobile phone App is used to provision the Wi-Fi network info to the gateway device. There is one Wi-Fi providing mechanism supported by current LCGW platform currently:

- Provision over Wi-Fi: gateway acts as soft-access point (dev-ap) and mobile phone as Wi-Fi station:
  Alibaba's Cloud Intelligence mobile phone App (Android and iOS) supports this mechanism to add the gateway device to it. Setting `WIFI_PROV_AWSS_DEV_AP_ENABLE` to 1 to enable this feature (default setting)

1.8 Substitution of Gateway and Test Devices' Default Triple-unit-groups

It is highly recommended that LCGW developer registers his own Aliyun cloud account to create a new IoT cloud project and the corresponding product/device of the LCGW and associated products/devices for the underlying child Zigbee devices. Through this process, developer can visualize the interactions between the cloud and the gateway as well as the underlying devices. Please refer to section 2.1 on how to attain the corresponding your own triple-unit-groups for this purpose.

Furthermore, if any other user has bound these default Triple-unit-groups with his own gateway device and Zigbee test devices by Alibaba's Cloud Intelligence mobile phone App before, later user will fail to add his own devices as these default Triple-unit-groups have been bound and registered to Aliyun already. The former user needs to unbind his devices manually to free them for the other users.

Modify /project/Realtek_amebad_va0_example/src_hp/cloud_gateway.c

1. Substitute the supported product key below in RED:

```
char *product_type_cloud_define[PRODUCT_TYPE_MAX] = {
   "a1cSTtZ7TXM",
   "a1pDuSGqiOf",
};
```

2. Substitute LCGW's Triple-unit-group below in RED:

```
#ifdef DEVICE_MODEL_ENABLED
char _product_key[IOTX_PRODUCT_KEY_LEN + 1]       = "a1VtEtCqVsx";
char _product_secret[IOTX_PRODUCT_SECRET_LEN + 1] = "bbr85gLslEx0GSYn";
char _device_name[IOTX_DEVICE_NAME_LEN + 1]       = "alink_gw_0001";
char _device_secret[IOTX_DEVICE_SECRET_LEN + 1]   = "IoL5eEaE1YoC3vZWeEEnuUP7HrrpDw8F";
#else
```

3. Substitute any test Zigbee device(s)' Triple-unit-groups below in RED:

```
#if Z3_CONTROLLER_ENABLE
// Silabs Z3 device example triple-unit-groups for https://living.aliyun.com
const CLOUD_DEV_META_T subdevArr[EXAMPLE_SUBDEV_MAX_NUM] = {
{
    "a1cSTtZ7TXM",
    "66HP98S8G6f5GlrB",
    "alilink_light_0001",
    "ys3PgPFBuFoSpizlhxSJSQL5Wil3XYG4",
},
```

```
{
    “a1cSTtZ7TXM”,
    “66HP98S8G6f5GlrB”,
    “alilink_light_0002”,
    “MqeuOAymX7IgeDARtAlWx4IiRCzArzpP”,
},
{
    “a1cSTtZ7TXM”,
    “66HP98S8G6f5GlrB”,
    “alilink_light_0003”,
    “1O8A5lI1DtjbaPfa2TD5dnRzvw1o1A4r”,
},
{
    “a1pDuSGqiOf”,
    “T0Iwt2HYZqfhzFHK”,
    “alink_wallswitch_0001”,
    “2egYRt9PlIZNeL5yMKcVsKLIXAGUxvAq”,
},
```

1.9 Compilation

Setup either the Ubuntu 18.04 Linux (recommended) or Windows 10 Cygwin build environment according to Realtek’s AmebaD SDK app note document.

Note that for the Ubuntu 18.04 Linux build environment, you need to install the 32-bit version of build packages on your 64-bit system as below:

```
> sudo apt-get update
> sudo apt-get install build-essential gcc-multilib gcc-4.8-multilib g++-multilib g++-4.8-multilib
lib32z1 lib32ncurses5 libc6-dev libgmp-dev libmpfr-dev libmpc-dev
> sudo dpkg –add-architecture i386
```

Build and generate the result AmebaD Cortex-M0 and Cortex-M4 firmware binaries as below

1. Build the project_lp target of KM0 baseband core:

```
> $ cd project/Realtek_amebad_va0_example/GCC-RELEASE/project_lp/
> $ make clean
> $ make all
```

   If build is passed, km0_boot_all.bin will be generated in below folder:
   `project/Realtek_amebad_va0_example/GCC-RELEASE/project_lp/asdk/image/`

2. Build the project_hp target of KM4 application core:

```
> $ cd project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/
> $ make clean
> $ make all
```

   If build is passed, km4_boot_all.bin and km0_km4_image2.bin will be generated in below folder:
   `project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/asdk/image/`

1.10 Flashing Image Binaries

Follow the AmebaD app note chapter 8 to flash the firmware binaries over USB-UART by Windows ImageTool (recommended) into the LCGW development board or Realtek AmebaD dev-board

Alternatively, user may flash the firmware binaries over SWD by a J-Link debugger (V9 or higher version) in Ubuntu Linux environment. Follow AmebaD app note instructions in section 1.4.1.2 to setup J-Link debug driver and the SWD wiring connection, section 1.4.2.2 to set up the J-Link GDB server, finally section 1.5 to use “make flash” to download image and flash over SWD.

2. Device Authentication

Every identity, including the gateway device and the Zigbee devices, to be connected to the Aliyun should have its unique Triple-unit-group, which includes the Product Key, Device Name and Device Secret info, to complete the authentication process. All these elements are attained from Aliyun Control Center by creating products and adding (generating) devices.

Alibaba offers two IoT cloud platforms, namely the Living IoT Platform (生活物联网平台) (https://living.aliyun.com/) and the IoT Platform (物联网平台) (https://cn.aliyun.com/). The former Living IoT Platform offers rather completed smart-home device connectivity to Aliyun and value-added services, e.g. Aliyun's "Cloud Intelligence" mobile phone App allows to control and manage the gateway and the underlying Zigbee devices. While the Zigbee devices can also be associated to Tmall Genie (天猫精灵) intelligent speaker for controlling the underlying Zigbee devices. Alibaba does charge the Triple-unit-group per device connecting to Living IoT Platform. On the other hand, customers need to development its own mobile App and/or web services for their gateways and IoT devices on the latter IoT Platform.

For simplicity, Silicon Labs has chosen the Living IoT Platform for easy illustrating our LCGW and the underlying Zigbee devices connecting to Aliyun, demonstrating their features through the Cloud Intelligence (云智能) mobile phone App and the associated Tmall Genie intelligent speaker.

2.1 Triple-unit-group Attaining

The following will show the steps to attain the device's triple-unit-group.

For the detailed information of the project and product management on https://living.aliyun.com/,
please refer to the following link https://help.aliyun.com/document_detail/126365.html

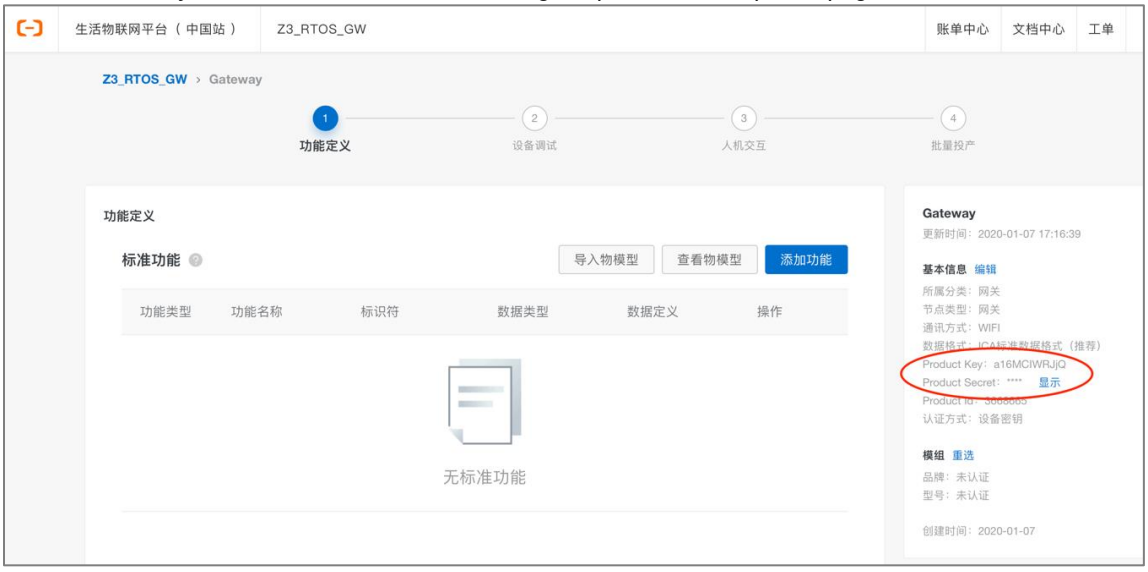
1) Sign up and log into https://living.aliyun.com/

2) Create new project, for example, create Z3_RTOS_GW project.

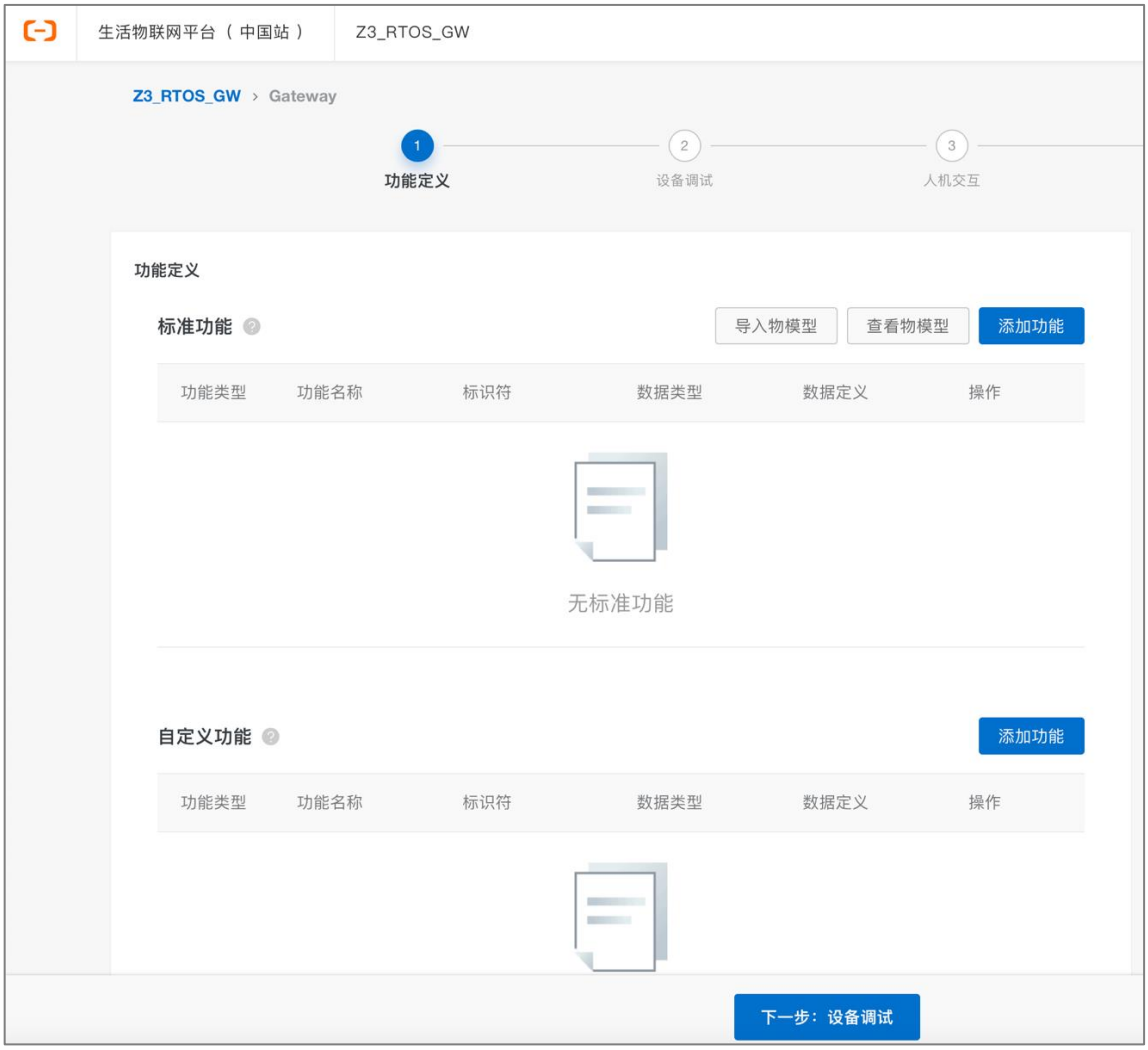
3) Create new product and get the Product Key and Product Secret.
   - Create new product

   - Define product. Gateway, Light and Switch are shown as examples below.
     - Gateway: Category “Gateway”, node-type “Gateway”, connectivity “WiFi”, data-format “Allink JSON” and enable-ID^2 “No”
     - Zigbee Light: Category “Light”, node-type “Device”, connect-through-gateway “Yes”, connect-protocol “Zigbee” , data-format “Allink JSON” and enable-ID^2 “No”
     - Zigbee Switch: Category “Wall Switch”, node-type “Device”, connect-through-gateway “Yes”, connect-protocol “Zigbee” , data-format “Allink JSON” and enable-ID^2 “No”

- Find Product Key and Product Secret after entering the product development page



- Define product's functions
  For Gateway product, there's no standard function to be chosen from. One can skip to the next step.



For Light product, besides the default standard function, LightSwitch, there're some standard functions to be chosen from, like Brightness and Color Temperature. Brightness is added in the example below.

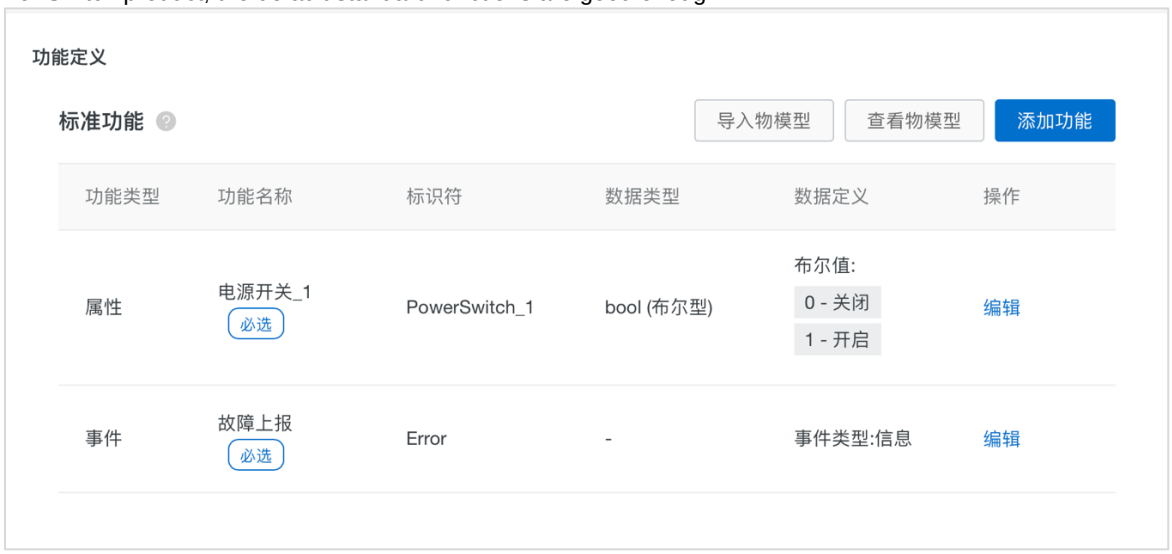
功能定义

标准功能

导入物模型 | 查看物模型 | 添加功能

功能类型	功能名称	标识符	数据类型	数据定义	操作
属性	主灯开关 必选	LightSwitch	bool (布尔型)	布尔值: 0 - 关闭 1 - 开启	编辑
属性	明暗度 可选	Brightness	int32 (整数型)	取值范围:0 ~ 100	编辑 删除
事件	故障上报 必选	Error	-	事件类型:故障	编辑

For Switch product, the default standard functions are good enough.



4) Get the Triple-unit-group for each device
There are two authentication info programing (burning) methods for devices:
   (1) **[1-Device-1-Secret]**: programing (burning) the unique Device Secret into the permanent storage memory of each device at the production stage, which requires that the developer or the manufacturer registers each device to the Aliyun platform and get its Device Secret beforehand.
   (2) **[1-Product-1-Secret]**: programing (burning) the same Product Secret into the permanent storage memory for the same product type of devices at the production stage, which requires less work effort in production, and requires that the developer or the manufacturer registers each device's Device Name to the Aliyun platform beforehand. The device will get its Device Secret when it logs onto the Aliyun Cloud at the first time only with product information (Product Key, Product Secret) and Device Name, which is called the Dynamic Registration.

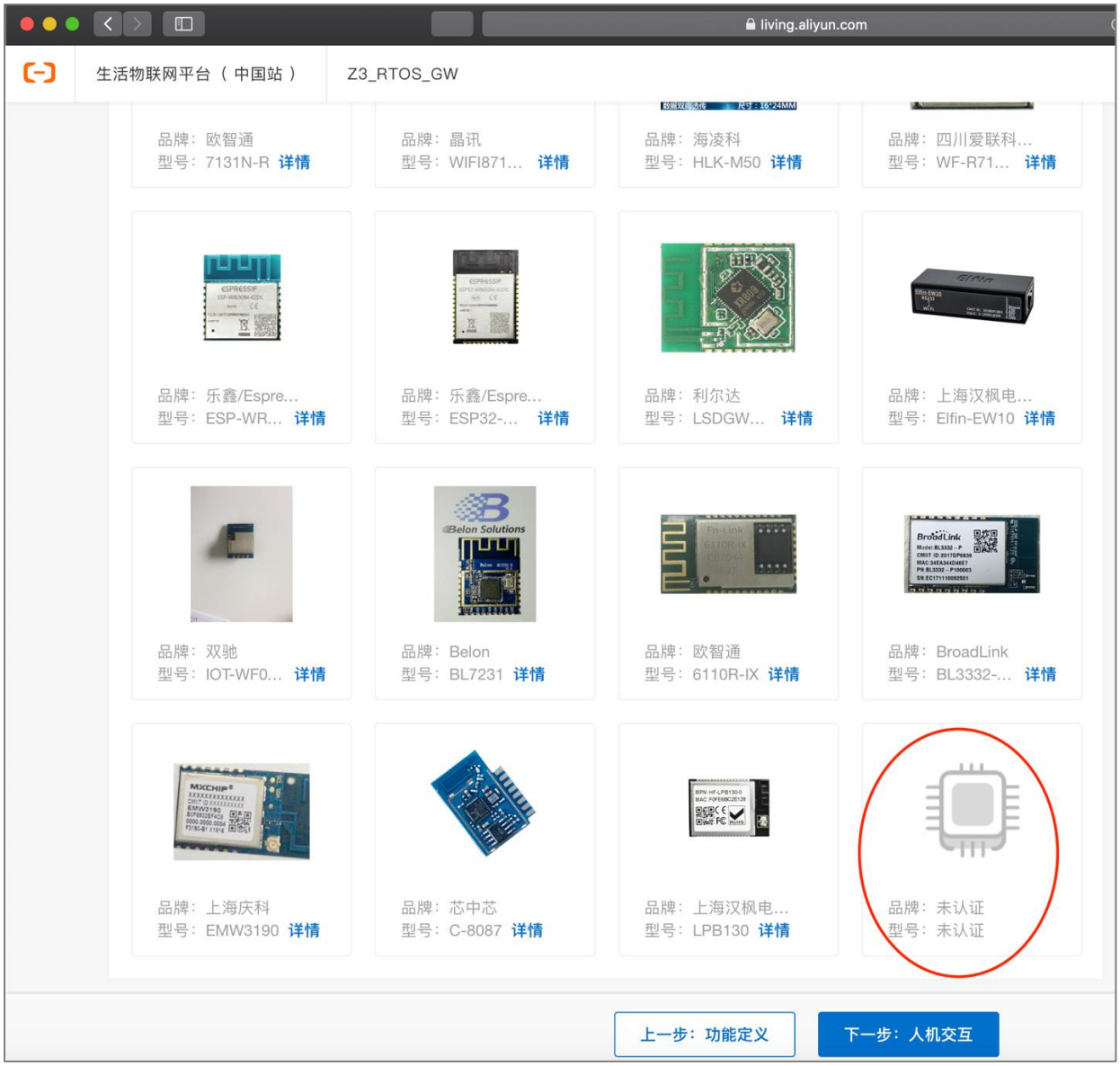
Dynamic Register process can only be executed once per device, so the Device Secret must be stored permanently in the device's memory after the Dynamic Registration process. Note that the Dynamic Register (Registration) feature is not implemented in current LCGW solution yet.

Customer can adopt the "Dynamic Registration" method which uses the device's EUI64 address as the Device Name to look up the corresponding Device Secret from Aliyun for device registration in the actual gateway product. Please refer to section 5 "Porting Alibaba iotkit-embedded to AmebaD SDK from Scratch" for more details

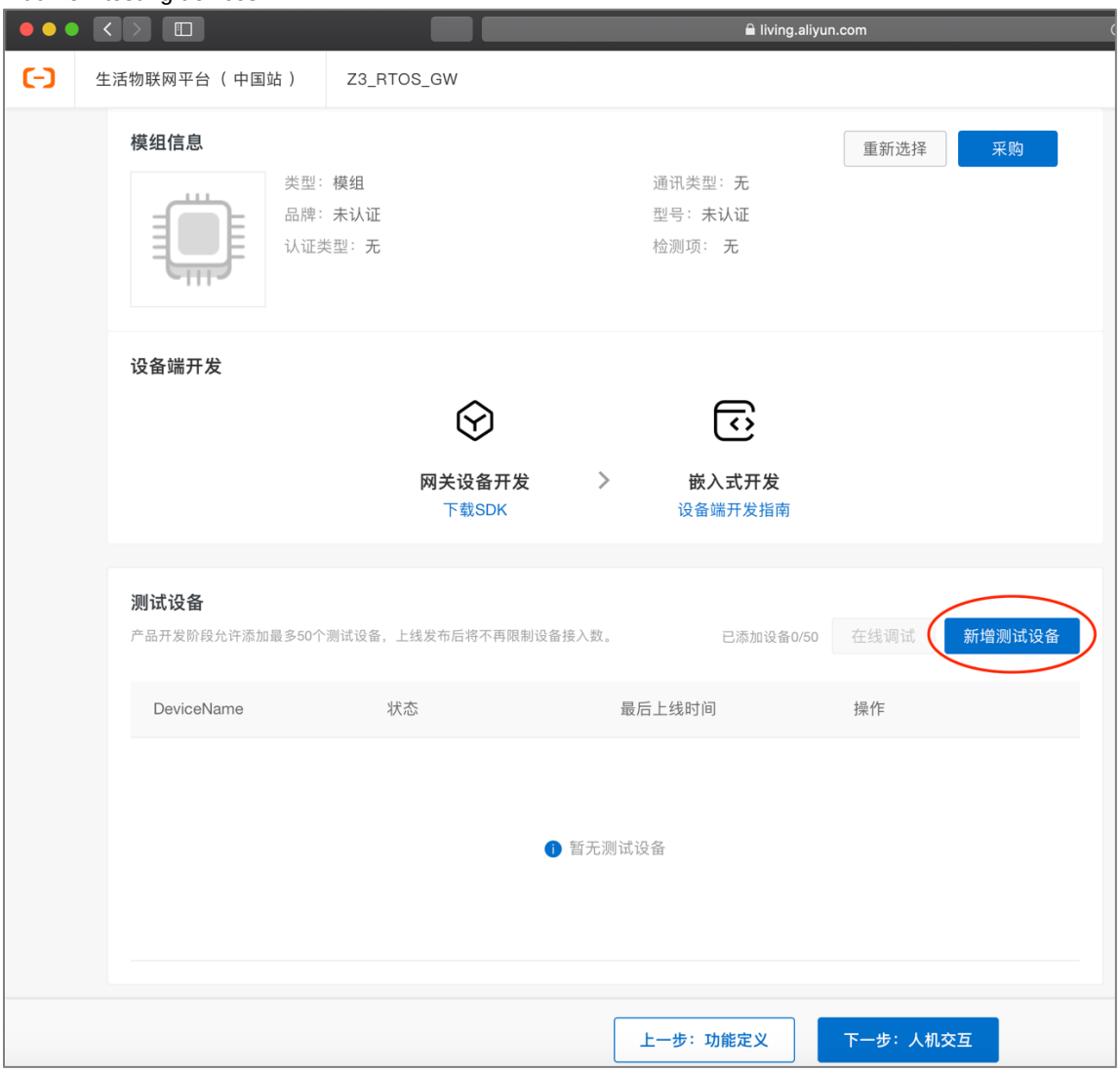
The Aliyun Gateway example in our current LCGW solution is using **1-Device-1-Secret** method, which means we need to apply for the Device Secret for every device beforehand.

The following will take the gateway as an example to show the steps to add new testing devices to get the triple-unit-groups on the Device Debug page. One could follow the same steps to add new testing devices for Light and Switch products.

- For Gateway, the chipset needs to be chosen first. The un-certificated one could be chosen if it's not in the list.


living.aliyun.com
生活物联网平台（中国站）
Z3_RTOS_GW
品牌：欧智通
型号：7131N-R 详情
品牌：晶讯
型号：WIFI871... 详情
品牌：海凌科
型号：HLK-M50 详情
品牌：四川爱联科...
型号：WF-R71... 详情
品牌：乐鑫/Espre...
型号：ESP-WR... 详情
品牌：乐鑫/Espre...
型号：ESP32-... 详情
品牌：利尔达
型号：LSDGW... 详情
品牌：上海汉枫电...
型号：Elfin-EW10 详情
品牌：双驰
型号：IOT-WF0... 详情
品牌：Belon
型号：BL7231 详情
品牌：欧智通
型号：6110R-IX 详情
品牌：BroadLink
型号：BL3332-... 详情
品牌：上海庆科
型号：EMW3190 详情
品牌：芯中芯
型号：C-8087 详情
品牌：上海汉枫电...
型号：LPB130 详情
品牌：未认证
型号：未认证
上一步：功能定义
下一步：人机交互

- Add new testing devices



- Enter the name of the new testing device

- Get the Triple-unit-group

2.2 Triple-unit-group Deploying

In general, the Triple-unit-group information, no matter for 1-Device-1-Secret method or 1-Product-1-Secret method, sooner or later will be programmed into the devices for permanent storage. There should be a way for the gateway to obtain the device's Triple-unit-group information from the device If the device needs the gateway to help it log onto the Cloud.

One approach is that when the Zigbee device joins the Zigbee network, the gateway reads the Triple-unit-group information from the device using manufacturer specific cluster. This means both the Zigbee devices and the gateway need application-level customization and related production process changes.

Obviously, with the above approach, a standard Zigbee product in the market without such firmware customization can't log onto the Aliyun (Cloud). To achieve standard Zigbee 3.0 devices connecting to Aliyun, the LCGW example has implemented the Triple-unit-group Proxy method to allow customers to evaluate the LCGW without firmware modifications on the Zigbee devices.

2.2.1 Triple-unit-group Proxy Method

All the pre-defined Triple-unit-groups are stored in the gateway, instead of in each device. When the Zigbee device joins the Zigbee network, the gateway will act as a proxy and assign one Triple-unit-group to it and help it log onto the cloud. The pre-defined Triple-unit-group can be assigned to any Zigbee device, and the gateway will not validate the joined device's EUI64 address (or Zigbee device's MAC address) with the pre-stored Triple-unit-groups, but one needs to make sure that the product type that the Triple-unit-group is referring to is the same as the device. Otherwise the cloud can't control the device properly.

2.2.2 Triple-unit-group Table Replacement

The Triple-unit-group of the gateway, some lighting devices and one switch device are pre-defined in the code. One could replace them with his/her own product definition. Please refer to section 1.8 for details.

Note that current LCGW example adopts the "1-Device-1-Secret" method and Triple-unit-group Proxy method for illustration purpose only.

2.3 Zigbee ZCL to Alibaba/ICA TSL Models Conversion

Aliyun uses Alibaba/ICA defined TSL (Thing Specification Language) models and its implementation in iotkit which is called the Alink protocol. The Zigbee devices follow Zigbee Alliance defined ZCL (Zigbee Cluster Library) as the application protocol.

To simplify the device message data conversion and manipulation in the cloud, our RTOS GW selects the Alink format instead of pass-through format during the LCGW and Zigbee devices product creation stage in Aliyun Control Center. Therefore, two-way ZCL to Alink data format conversion is implemented in cloud gateway application layer. i.e.
`project/Realtek_amebad_va0_example/src_hp/cloud_gateway_interface.c`

The code below shows some downlink property setting handle functions and their corresponding set_interface functions where the Alink to ZCL data conversions are implemented

```
/*******************************************************************************
*  property setting handler used by cloud msg center
*******************************************************************************/
static int16_t lightswitch_property_set(uint16_t value, EUI64_T eui64, uint8_t endpoint)
{
    // send msg to stack
    if (-1 == lightswitch_set_interface(value, eui64, endpoint)) {
        ERROR_TRACE("failed.");
        return -1;
    }
    return 0;
}

static int16_t brightness_property_set(uint16_t value, EUI64_T eui64, uint8_t endpoint)
{
    // send msg to stack
    if (-1 == brightness_set_interface(value, eui64, endpoint)) {
        ERROR_TRACE("failed.");
        return -1;
    }
    return 0;
}
```

Where:

```
int16_t lightswitch_set_interface(uint16_t value, Eui64_t eui64, uint8_t endpoint)
{
    uint16_t on_off;

    if (0 == value) {
        on_off = CLUSTER_ON_OFF_CMD_ID_OFF;
    } else {
        on_off = CLUSTER_ON_OFF_CMD_ID_ON;
    }

    if (-1 == cmd_ctrl(&eui64, CLUSTER_ID_ON_OFF, on_off, NULL)) {
        ERROR_TRACE("failed.");
        return -1;
    }
    return 0;
}

int16_t brightness_set_interface(uint16_t value, Eui64_t eui64, uint8_t endpoint)
{
    USER_TRACE("Set brightness to %d", value);
    if (-1 == cmd_ctrl(&eui64, CLUSTER_ID_LVL_CTRL, CLUSTER_LVL_CTRL_CMD_ID_MTL, value)) {
        ERROR_TRACE("failed.");
        return -1;
    }
    return 0;
}
```

The uplink attribute reporting message handler and data conversion will be handled in the similar manner. When a further new device type support is added, developer is responsible for adding the corresponding new handler and interface functions accordingly.

3. Mobile Phone App Usage

The Aliyun's Cloud Intelligence mobile phone App (public version) could be used to add gateway and Zigbee devices to Aliyun. Current Cloud Intelligence v3.1 has been verified with our Z3GatewayFreeRTOS Aliyun example. It is recommended to use the Android Cloud Intelligence mobile phone App for evaluation and development due to its better user experiences. Because the iPhone Cloud Intelligence mobile phone App currently has issue to add the LCGW successfully.

3.1 App Configuration

Before using the App to add devices, we still need to configure the human-machine interface for each product type.

3.1.1 App Configuration for LCGW

Enter the HM Interface page of the LCGW product.

- Use Public App (使用公版 App 控制产品) = Yes
- Select Panel (面板选择): Set the Control Panel
- QR code of Wi-Fi provisioning and App download (配网+App 下载二维码): Save it for adding the same type of device by QR code scanning later
- Sharing Type (分享方式)：When using the public App, the default is Authorized and can't be changed
- Multi-language (多语言管理): Set the product name and function names for displaying in App in different languages

- Wi-Fi provisioning (配网引导):
  - Default Wi-Fi Provisioning Method (默认配网方式) = Device AP
  - Alternative Wi-Fi Provisioning Method (备选配网方式) = None
  - Scan QR Code/ Auto Local Discovery (扫码/本地自动发现): Checked by default
  - Support Manual Select from Product List/ Search (支持手动选择产品列表/搜索): Checked
    (Only the product that is in the Mass Production stage and whose name has passed the inspection of Aliyun platform will be shown in the Product List.)
  - Edit the Guide page (编辑引导页面): Set the picture and descriptions in the Guide page

- Connect to Tmall Genie (天猫精灵接入) = Yes

3.1.2 App Configuration for Light

Enter the HM Interface page of the Light product.

- Use Public App (使用公版 App 控制产品) = Yes
- Select Panel (面板选择): Set the Control Panel
- QR code of network provisioning and App download (配网+App 下载二维码): Save it for adding the same type of device by QR code scanning later
- Sharing Type （分享方式）: When using the public App, the default is Authorized and can't be changed
- Multi-language (多语言管理): Set the product name and function names for displaying in App in different languages

- Network provisioning (配网引导):
  - Scan QR Code/ Auto Local Discovery (扫码/本地自动发现): Checked by default
  - Support Manual Select from Product List/ Search (支持手动选择产品列表/搜索): Checked
    (Only the product that is in the Mass Production stage and whose name has passed the inspection of Aliyun platform will be shown in the Product List.)
  - Guide Picture (配网引导图): Set the picture in the Guide page
  - Provisioning Description (配网文案): Set the description for network provisioning
  - Button Description (按钮文案): Set the description for button

- Automation and Timing (自动化和定时): Select the product's properties to be used in the automatic device controlling, and determine whether the timer or countdown timer can be set in the product itself.
  Here we allow LightSwitch and Brightness to be chosen as both the Condition and the Action.

- Connect to Tmall Genie (天猫精灵接入) = Yes

3.1.3 App Configuration for Switch

Enter the HM Interface page of the Switch product.

- Use Public App (使用公版 App 控制产品) = Yes
- Select Panel (面板选择): Set the Control Panel
- QR code of network provisioning and App download (配网+App 下载二维码): Save it for adding the same type of device by QR code scanning later
- Sharing Type (分享方式)：When using the public App, the default is Authorized and can't be changed
- Multi-language (多语言管理): Set the product name and function names for displaying in App in different languages

- Network provisioning (配网引导):
  - Scan QR Code/ Auto Local Discovery (扫码/本地自动发现): Checked by default
  - Support Manual Select from Product List/ Search (支持手动选择产品列表/搜索): Checked
    (Only the product that is in the Mass Production stage and whose name has passed the inspection of Aliyun platform will be shown in the Product List.)
  - Guide Picture (配网引导图): Set the picture in the Guide page
  - Provisioning Description (配网文案): Set the description for network provisioning
  - Button Description (按钮文案): Set the description for button

- Automation and Timing (自动化和定时): Select the product's properties to be used in the automatic device controlling, and determine whether the timer or countdown timer can be set in the product itself.
  Here we allow PowerSwitch_1 to be chosen as the Condition.

- Connect to Tmall Genie (天猫精灵接入) = Yes

3.2 Cloud Intelligence App Download

Scan the QR code below to download Alibaba's Cloud Intelligence mobile application (User or Developer editions are available). The User edition is recommended for demo and evaluation while the Developer edition is recommended for development.

User Edition Developer Edition

3.3 Mobile Phone App Provision Wi-Fi to LCGW and Add it to Aliyun

The mobile phone App will provision Wi-Fi network to the gateway so that the gateway can connect to the Wi-Fi AP.

Device AP method is chosen and implemented. The user needs to make sure that the gateway enters the Device AP mode.

For Silicon Labs LCGW hardware platform:

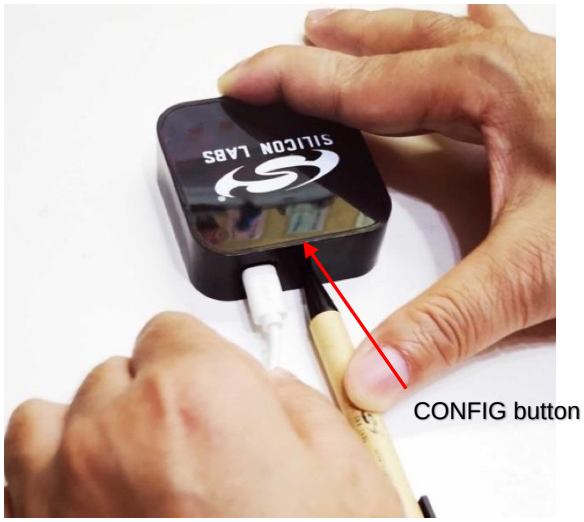
- Press and hold the Config button and then press the internal Reset button and release Reset button then the Config button to force the gateway to enter the device AP mode, OR
- Press and hold the Config button and plug the gateway demo unit into the main power supply to power it up to enter the device AP mode.

Note:

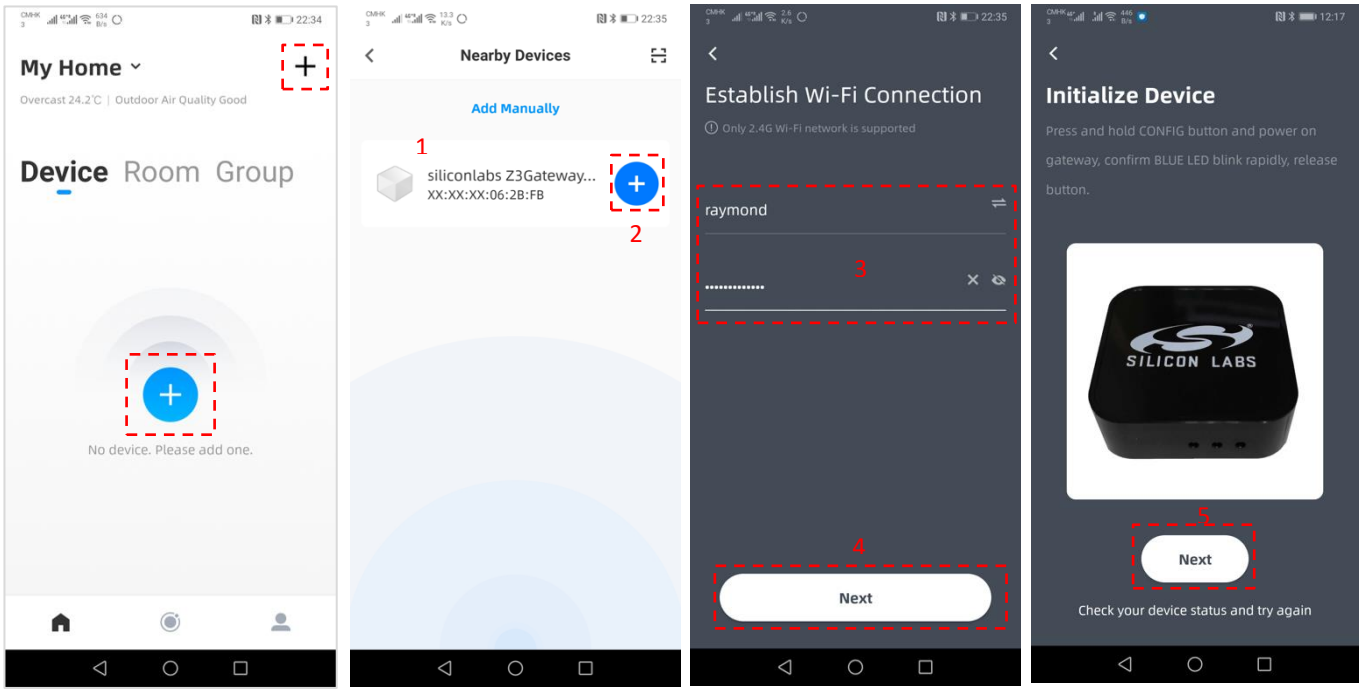
- For Realtek AmebaCS or AmebaD dev-boards, GPIO header pin B2 (silkscreen B2) is corresponding to the Config button.
- For Shenzhen 99IoT AFM30xT EVB, K1 is the Reset button and K2 (P_B2) is corresponding to the Config button.

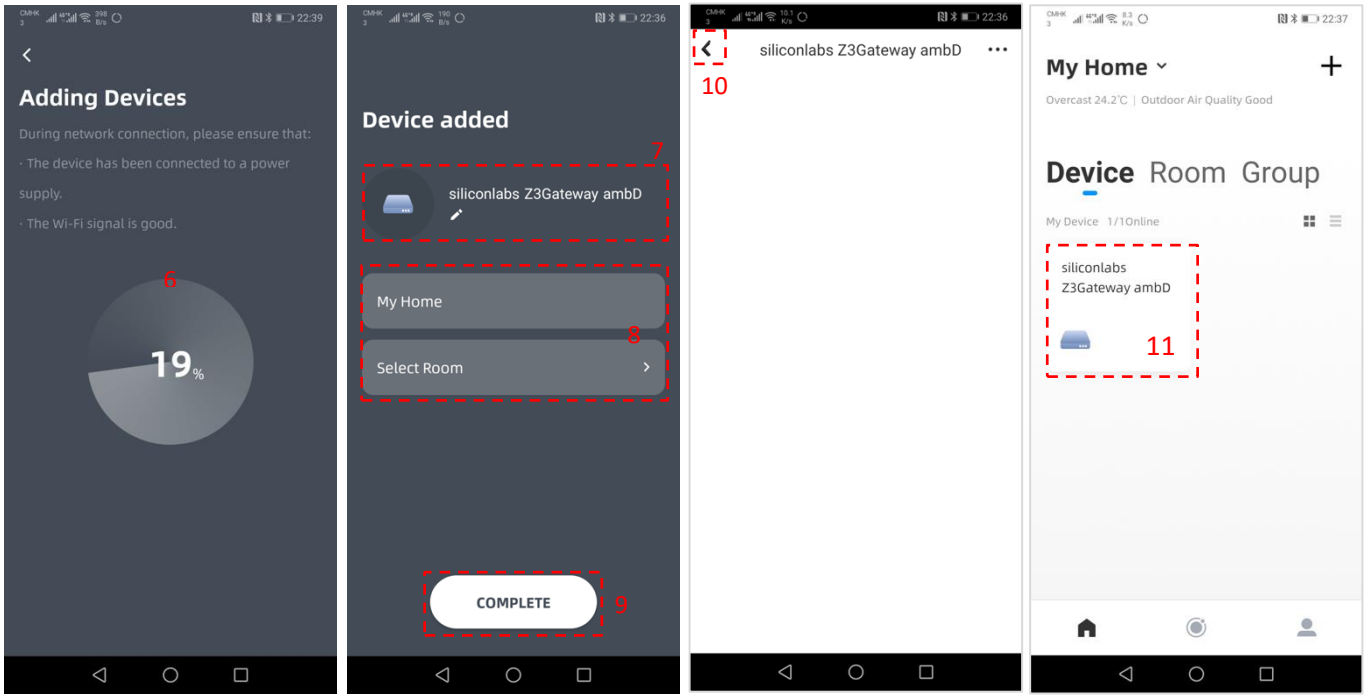
Currently Silicon Labs has implemented the Device AP method on the RTOS gateway RTL8720CSM platform for illustration purpose. Customer can select other Wi-Fi provisioning methods at the Human-machine Interface phase of the project development on https://living.aliyun.com/, see below the screenshot.

- Step 1: Get the LCGW into the Wi-Fi provisioning mode
  1. Unplug the micro-USB cable from the LCCWv2's USB port
  2. Use the ball pen tip to punch the hidden COFIG button at the back and hold the button
  3. Re-plug the micro-USB cable to LCGWv2 to power it up from PC's USB port
  4. Release the CONFIG button, Blue LED indicator is fast blinking as the indication of GW Wi-Fi provisioning mode

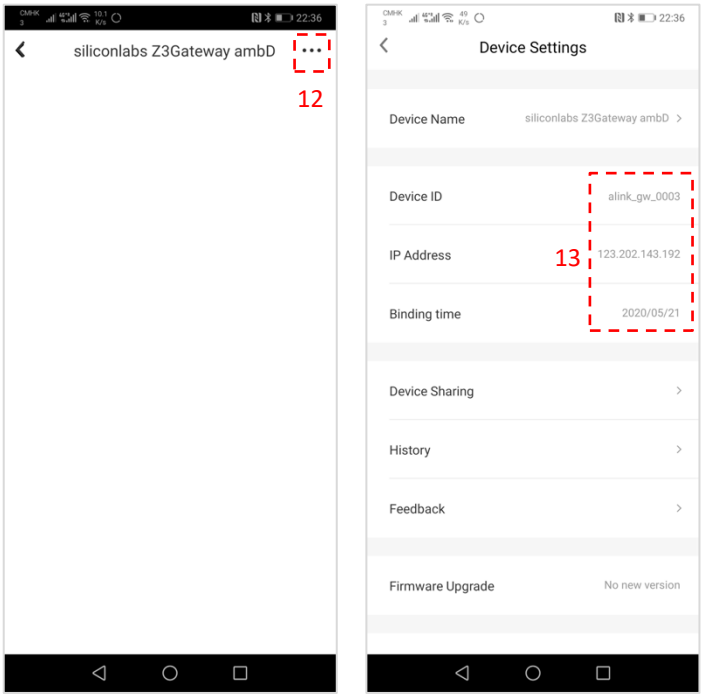


- Step 2: Connect your mobile phone to the Wi-Fi network that the LCGWv2 is targeted to be connected to internet
  Note: The mobile phone Wi-Fi must be connected to a 2.4GHz Wi-Fi network, otherwise provisioning will fail.
- Step 3: Tap to run “Cloud Intelligence” App, tap the top-right “+” icon to scan for nearby Alibaba Wi-Fi/BT-mesh IoT devices…
  1. On Android phones, LCGW will be discovered automatically…
  2. “siliconlabs Z3Gateway ambD” appears, tap right-side blue “(+)” icon
  3. Enter Wi-Fi SSID and Password
  4. Tap “Next” button
  5. Make sure that the GW Wi-Fi (BLUE) indicator is fast blinking, tap “Next” button to continue…





6. You will see Adding Devices in progress…
7. Wait LCGW device added in less than 1 minute. Optional to rename GW.
8. Optional to rename the house and select room
9. Tap “Complete” button
10. In GW panel, tap “<” icon to return main page. See the LED indicators:
    - If GW connected to Aliyun: Cloud (GREEN) indicator turns on
    - Wi-Fi (BLUE) and Zigbee (RED) indicators restore to slow blinking
11. “siliconlabs Z3Gateway ambD” appears in the device list, tap device icon
12. The GW Control Panel is blank as nothing to control. Tap top-right “…” icon to view GW’s settings page
13. Check GW’s device ID, IP address, bonding time, etc. information

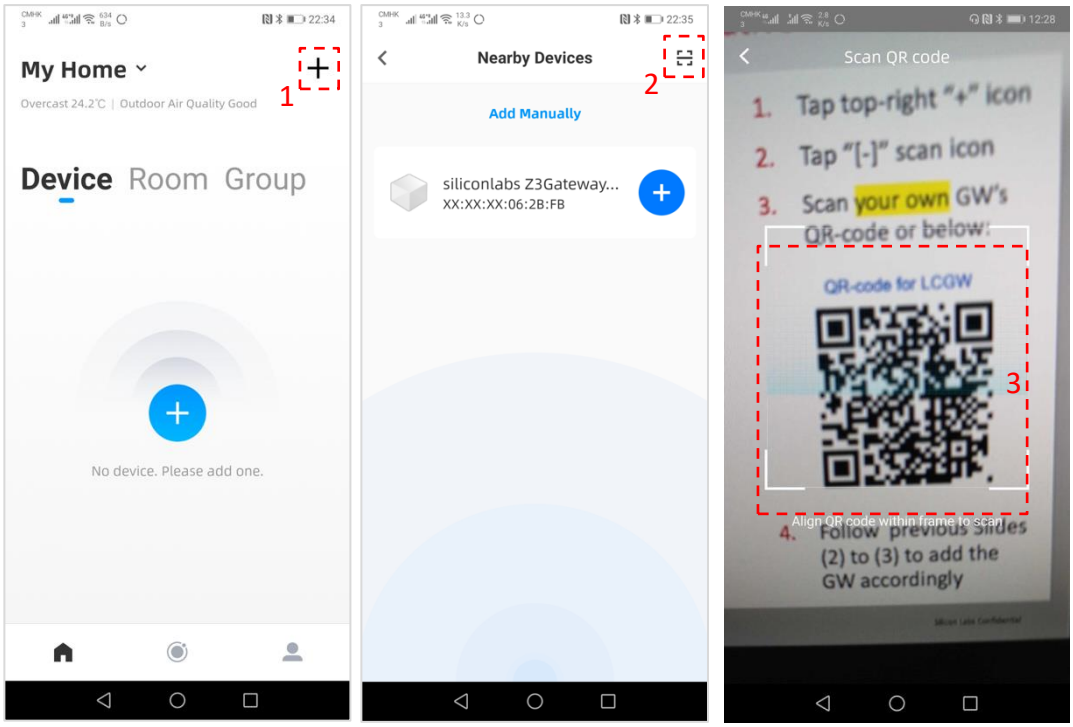


Alternative Method 1: Scan the LCGW's QR-code

1. Tap top-right "+" icon
2. Tap "[-]" scan icon
3. Scan LCGW's QR-code or below:

QR-code for LCGWv2

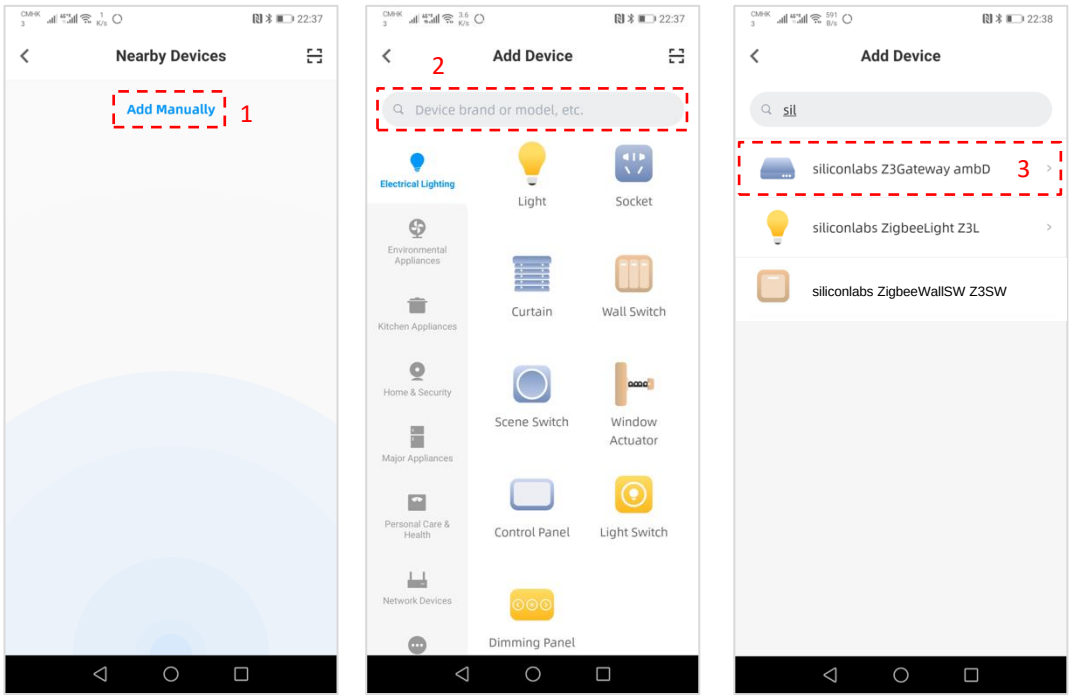
4. Follow sub-step (3) to (10) of previous Step 3 to add the LCGW accordingly



Alternative Method 2: Manually add the LCGW device

Note: Only the product that is in the Mass Production stage and whose name has passed the inspection of Aliyun platform will be shown in the Product List.

1. Tap top-middle “Add Manually” icon
2. In “Add Device” panel search box, enter “siliconlabs” (or your device name) to filter our devices
3. Tap from result listed “siliconlabs Z3Gateway ambD” (or your defined product) to add the GW
4. Follow sub-step (3) to (10) of previous Step 3 to add the LCGW accordingly



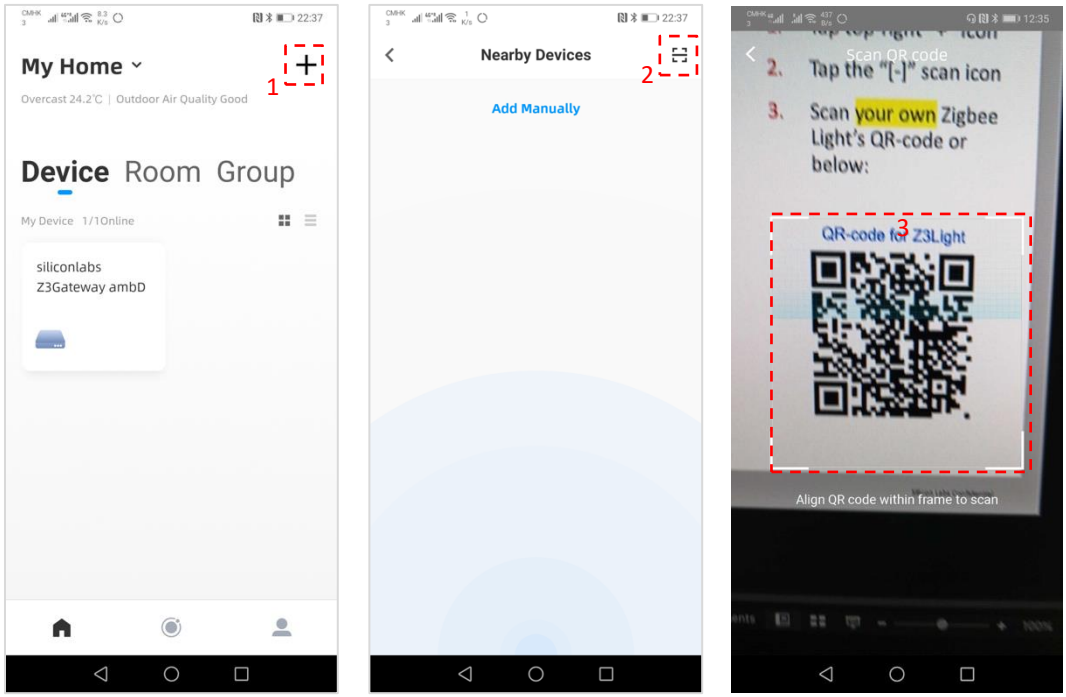
3.4 Adding Zigbee Light to Network and Aliyun

There're three different methods to add Zigbee sub-devices:
(1) Scan device's product QR-code generated from living.aliyun.com
(2) Manually select a device product from the full product list
(3) LCGW single CONFIG button to permit Zigbee device joining in 180s using default global link-key.
Note: Method 3 CONFIG button method requires the first device of a product has been added via QR-code successfully.

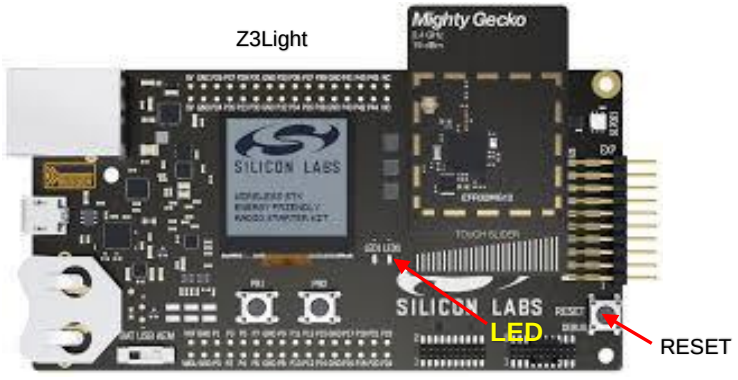
Method 1: Scan Zigbee Light's product QR-code

- Before adding a Zigbee device, the gateway should have automatically formed a Zigbee network.
- Step 1: Scan the device's QR-code by the Cloud Intelligence mobile phone App
  1. Tap top-right "+" icon
  2. Tap "[-]" scan icon
  3. Scan device's QR-code or below:

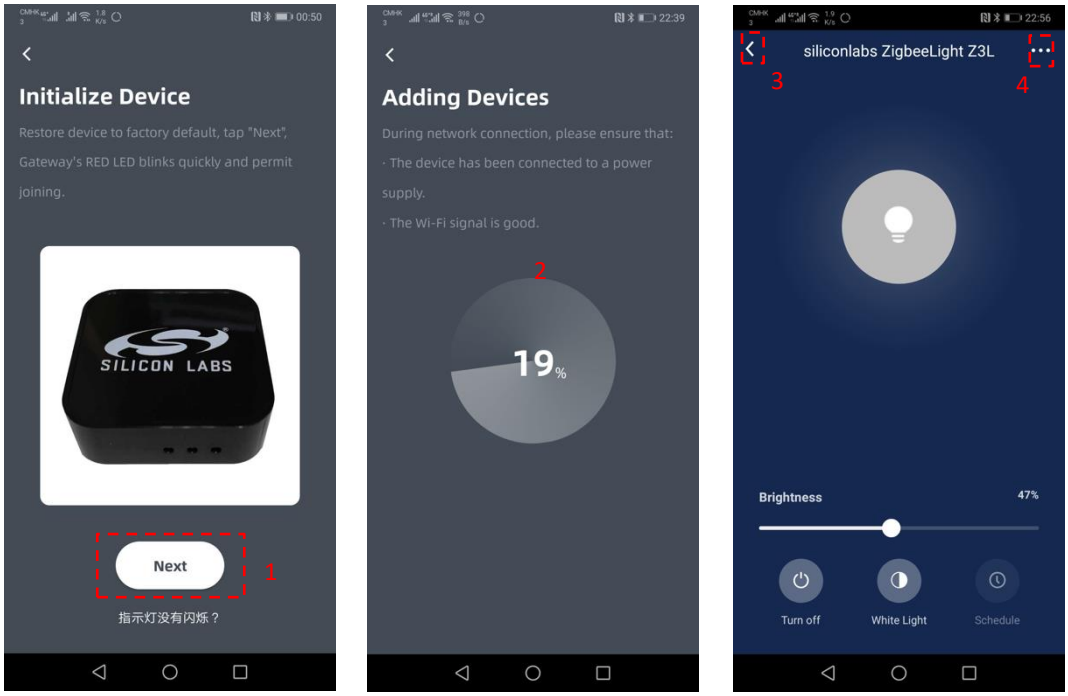
QR-code for Z3Light



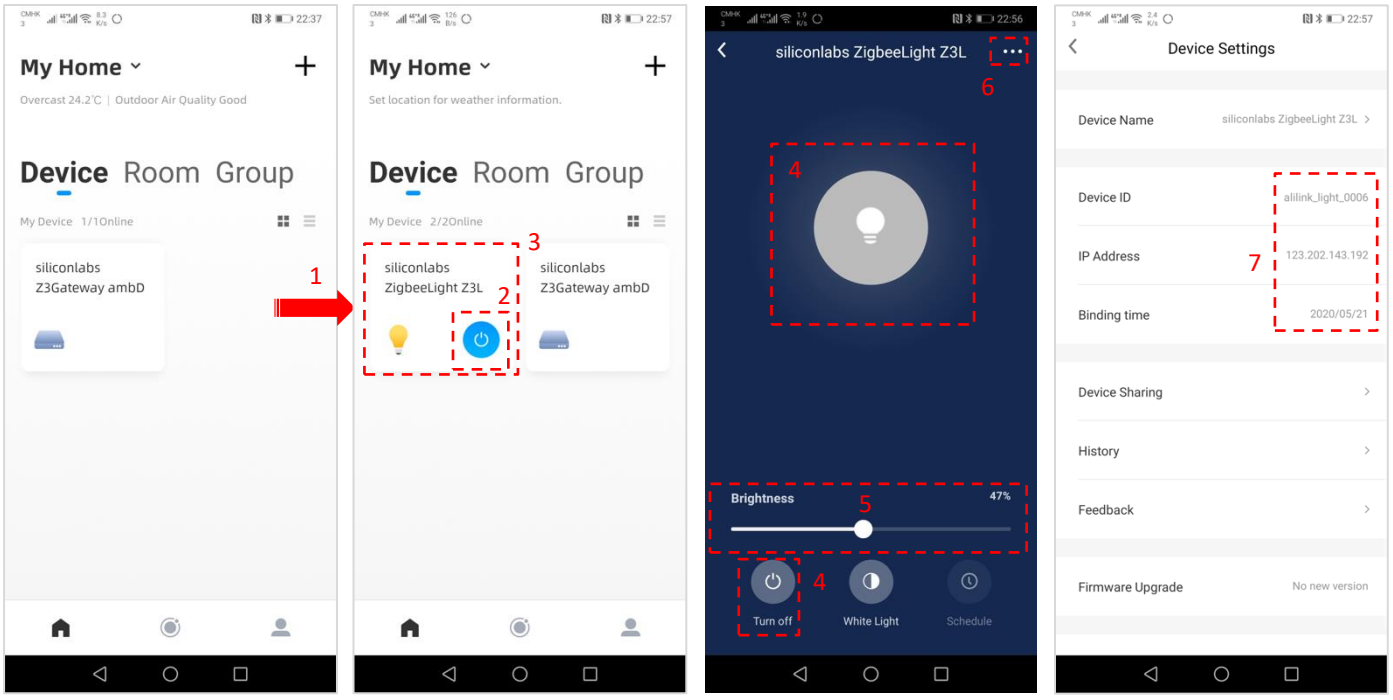
- Step 2: Run the Z3LightSoC FW on WSTK to scan Zigbee network:
  1. Connect WSTK with BRD4161/4162A radio board flashed with Z3LightSoC firmware to PC via USB
  2. In Simplicity Studio Debug Adapters, right click the Z3Light WSTK and select "launch Console"
  3. Go to "Serial 1" window, press "Enter" in terminal until you see the prompts, e.g. "Z3LightSoC>"
  4. Type "network leave" then press "Enter" to force Z3Light to leave network
  5. Press WSTK's Target RESET button, Z3Light will start searching network after reset
     (One could check the logs in terminal to confirm device has joined network successfully.)



- Step 3: Add Z3Light device
  1. Tap “Add Device” button on the App
  2. You will see Adding Devices in progress and GW Zigbee (RED) indicator is fast blinking, 100ms on 100ms off. Wait till adding device process completes in less than 1 minute. Once the device joins successfully or the 180s expires, the Zigbee (RED) indicator returns back to slow blinking (100ms on 1900ms off)
  3. Tap “<” to return main page, or
  4. Tap “…” to view the Light settings page



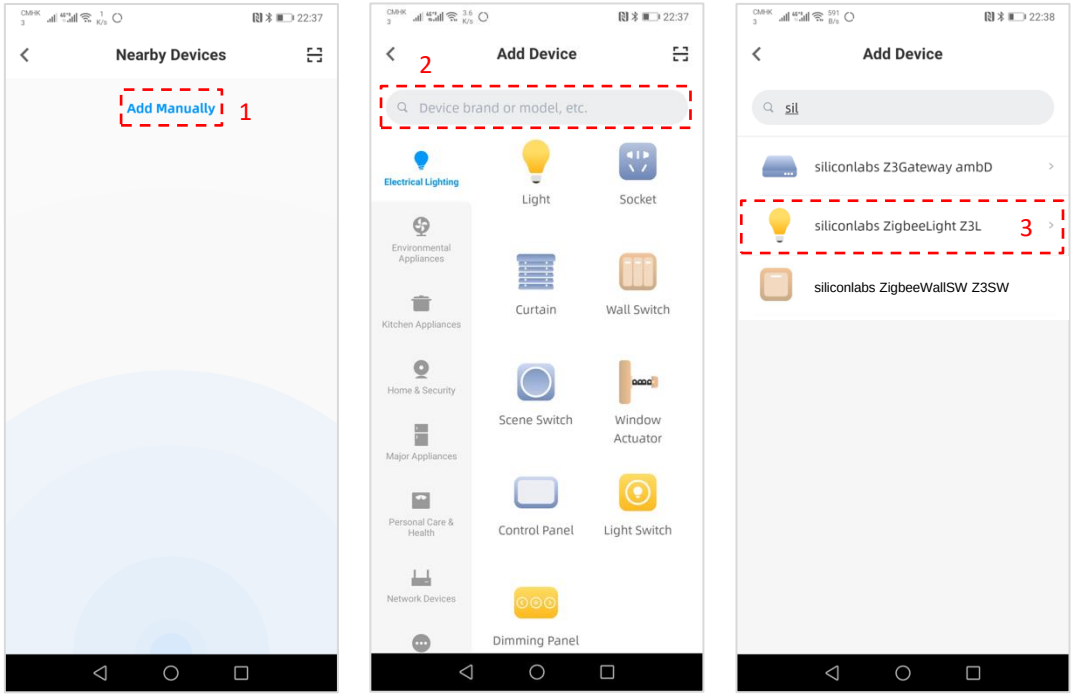
- Step 4: Z3Light Control and Settings
  1. After successful adding the device, it'll appear on the Device page of the App. (If not, please refresh the page.)
  2. Tap the "On/Off" button to toggle the Z3 Light on/off
  3. Tap "siliconlabs ZigbeeLight Z3L" icon itself to open the Light Control panel
  4. Tap middle "Bulb" icon or bottom "On/Off" button to toggle on/off
  5. Drag the Brightness bar to adjust brightness from 0 to 100% (Note that the WSTK's LED doesn't support the brightness adjustment.)
  6. Tap "…" to view "Device Settings" page
  7. In "Device Settings" page, check the device ID, IP address, bonding time, etc. information



Method 2: Manually add the Zigbee Light device

Note: Only the product that is in the Mass Production stage and whose name has passed the inspection of Aliyun platform will be shown in the Product List.

1. Tap top “Add Manually” icon
2. In “Add Device” panel search box, enter “siliconlabs” to filter our devices
3. Tap from result list “siliconlabs ZigbeeLight Z3L” to add the Zigbee Light
4. Follow previous Steps 2 to 4 to add the device accordingly and do the device controlling



Method 3: Single CONFIG button open network for Zigbee device joining

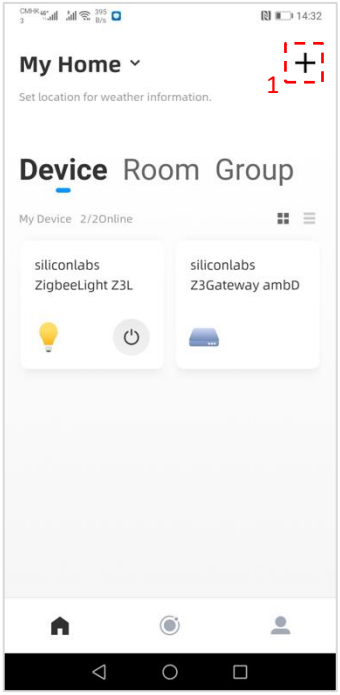
1. Follow previous Step 2 to force the device SoC on WSTK to leave network and scan Zigbee network
2. Set LCGWv2 to “Open” network for device joining:
   - Use a ball pen tip to punch the hidden COFIG button once, i.e. less than 500ms
   - Release the CONFIG button, Zigbee (RED) indicator is fast blinking (100ms on 100ms off) as the indication
   - Gateway “Permit Join” with default global link-key for 180s
3. Once the device joins successfully or the 180s expires, the Zigbee (RED) indicator returns back to slow blinking (100ms on 1900ms off)
4. If the device joins successfully, follow the previous Step 4 to do the device controlling.

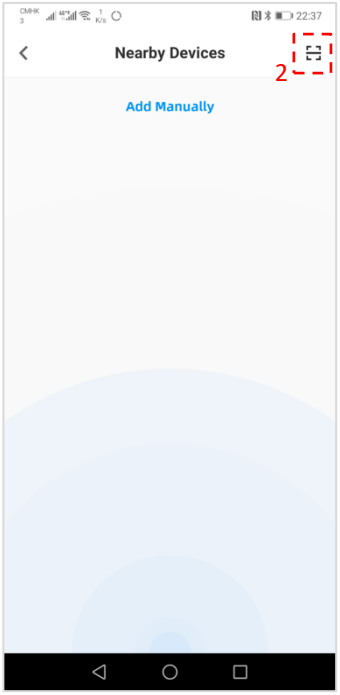
3.5 Adding Zigbee Switch to Network and Aliyun

Method 1: Scan Zigbee Switch's product QR-code

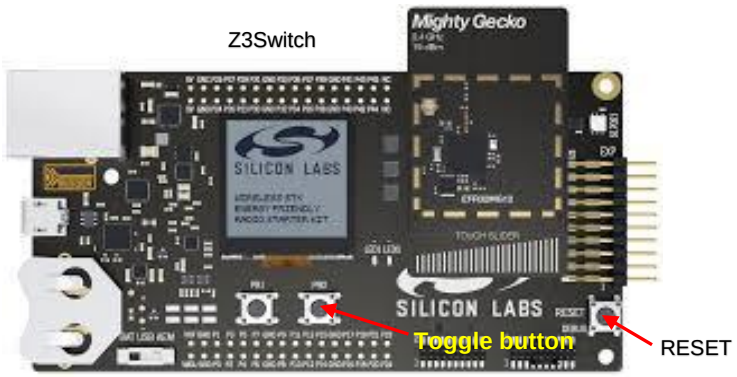
- Before adding a Zigbee device, the gateway should have automatically formed a Zigbee network.
- Step 1: Scan the device's QR-code by the Cloud Intelligence mobile phone App
  1. Tap top-right "+" icon
  2. Tap "[-]" scan icon
  3. Scan device's QR-code or below:

QR-code for Z3Switch

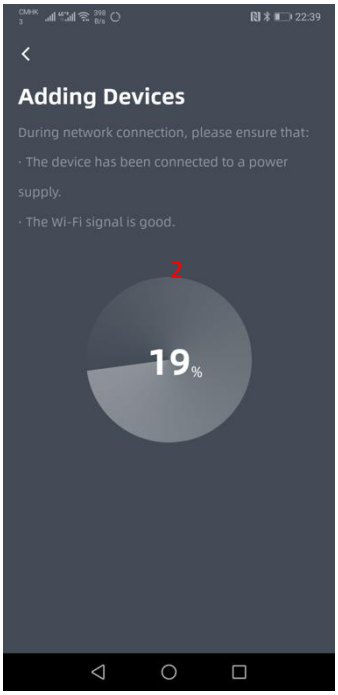


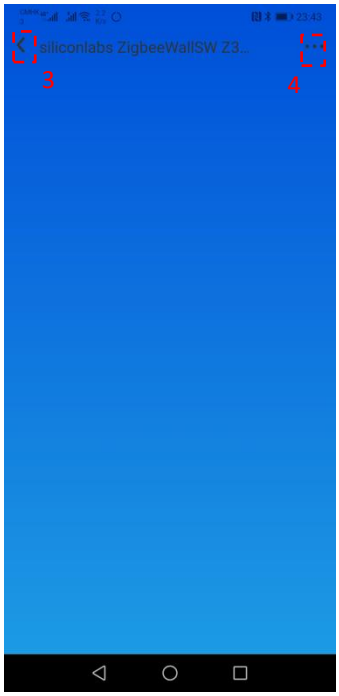


- Step 2: Run the Z3SwitchSoC FW on WSTK to scan Zigbee network:
  1. Connect WSTK with BRD4161/4162A radio board flashed with Z3SwitchSoC firmware to PC via USB
  2. In Simplicity Studio Debug Adapters, right click the Z3Switch WSTK and select "launch Console"
  3. Go to "Serial 1" window, press "Enter" in terminal until you see the prompts, e.g. "Z3SwitchSoC>"
  4. Type "network leave" then press "Enter" to force Z3Switch to leave network
  5. Type "plugin network-steering start 60" then press "Enter" to command Z3Switch to start searching network (One could check the logs in terminal to confirm device has joined network successfully.)

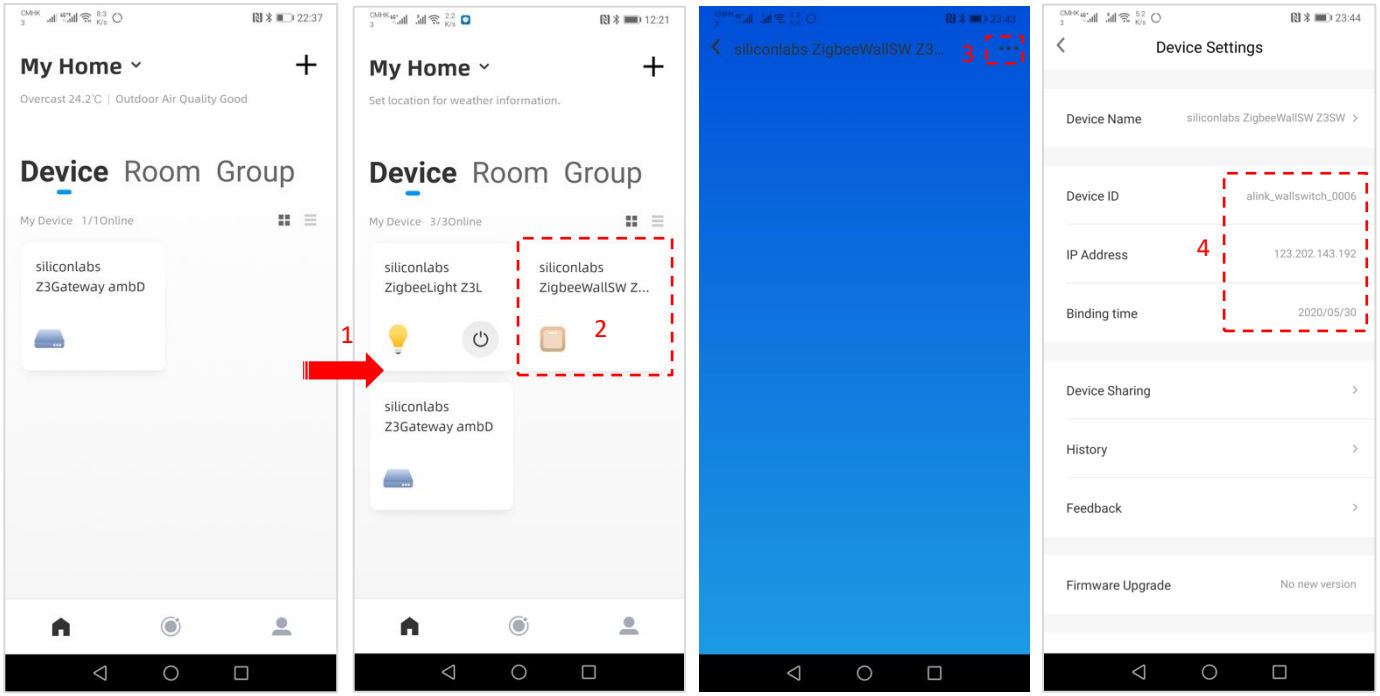


- Step 3: Add the Z3Switch device
  1. Tap “Add Device” button
  2. You will see Adding Devices in progress and GW Zigbee (RED) indicator is fast blinking, 100ms on 100ms off. Wait till adding device process completes in less than 1 minute. Once the device joins successfully or the 180s expires, the Zigbee (RED) indicator returns back to slow blinking (100ms on 1900ms off)
  3. Tap “<” to return main page, or
  4. Tap “…” to view “Device Settings” page





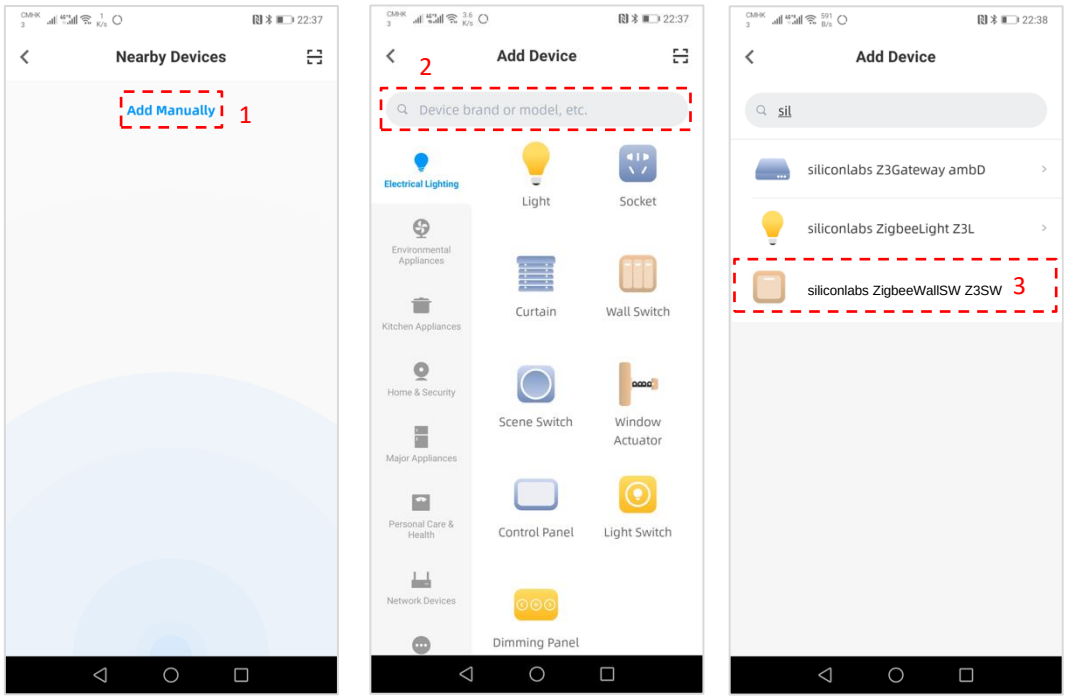
- Step 4: Z3Switch Settings
  1. After successful adding the device, it'll appear on the Device page of the App. (If not, please refresh the page.)
  2. Tap "siliconlabs ZigbeeWallSW Z3SW" icon itself to open the Switch Control panel
  3. Tap "…" to view "Device Settings" page
  4. In "Device Settings" page, check the device ID, IP address, bonding time, etc. information



Method 2: Manually add the Zigbee Switch device

Note: Only the product that is in the Mass Production stage and whose name has passed the inspection of Aliyun platform will be shown in the Product List.

1. Tap top “Add Manually” icon
2. In “Add Device” panel search box, enter “siliconlabs” to filter our devices
3. Tap from result list “siliconlabs ZigbeeWallSW Z3SW” to add the Zigbee Switch
4. Follow previous Steps 2 to 4 to add the device accordingly and do the device controlling

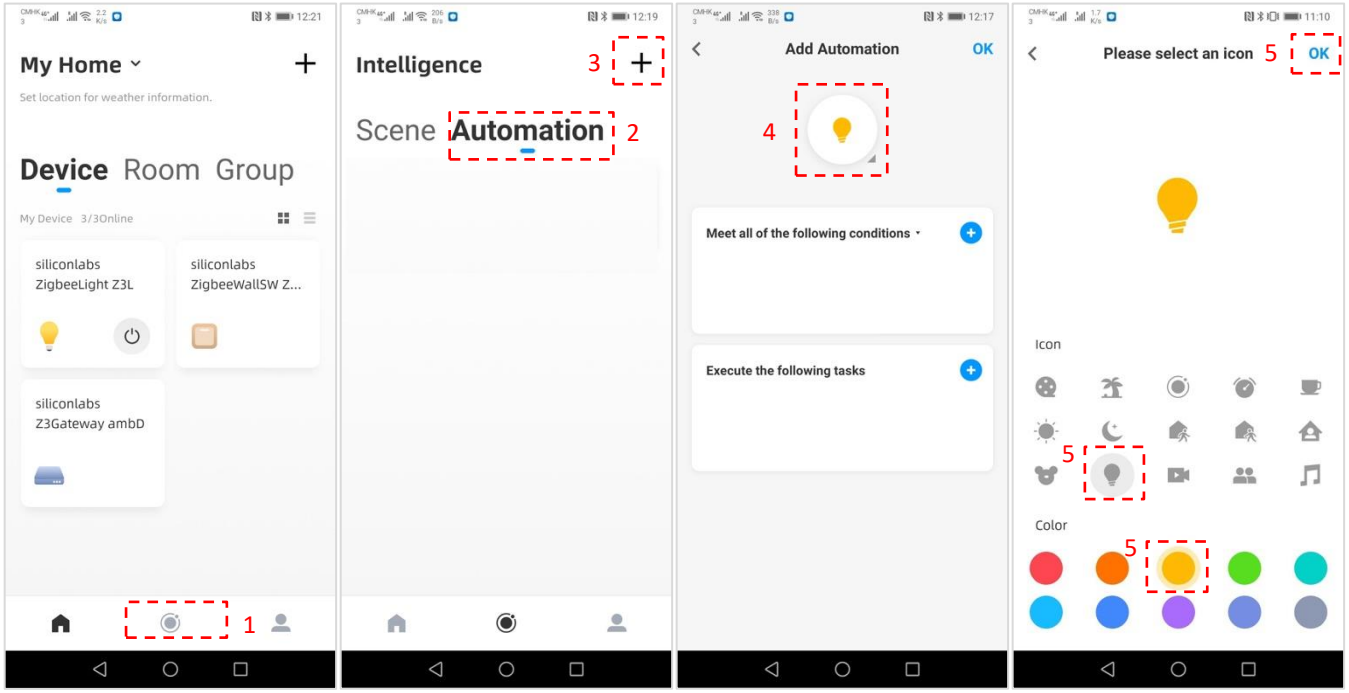


Method 3: Single CONFIG button open network for Zigbee device joining

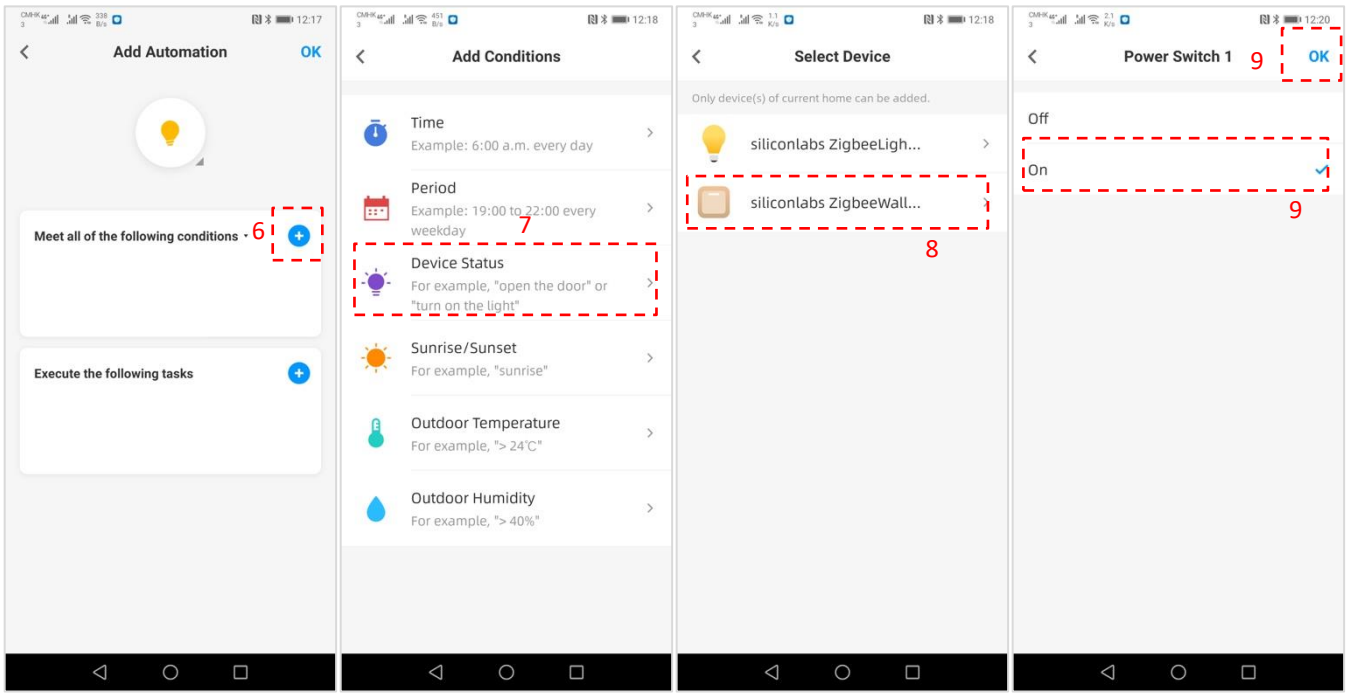
1. Follow previous Step 2 to force the device SoC on WSTK to leave network and scan Zigbee network
2. Set LCGWv2 to “Open” network for device joining:
   - Use a ball pen tip to punch the hidden COFIG button once, i.e. less than 500ms
   - Release the CONFIG button, Zigbee (RED) indicator is fast blinking (100ms on 100ms off) as the indication
   - Gateway “Permit Join” with default global link-key for 180s
3. Once the device joins successfully or the 180s expires, the Zigbee (RED) indicator returns back to slow blinking (100ms on 1900ms off)
4. If the device joins successfully, follow the previous Step 4 to do the device controlling.

3.6 Setting Up the Automation

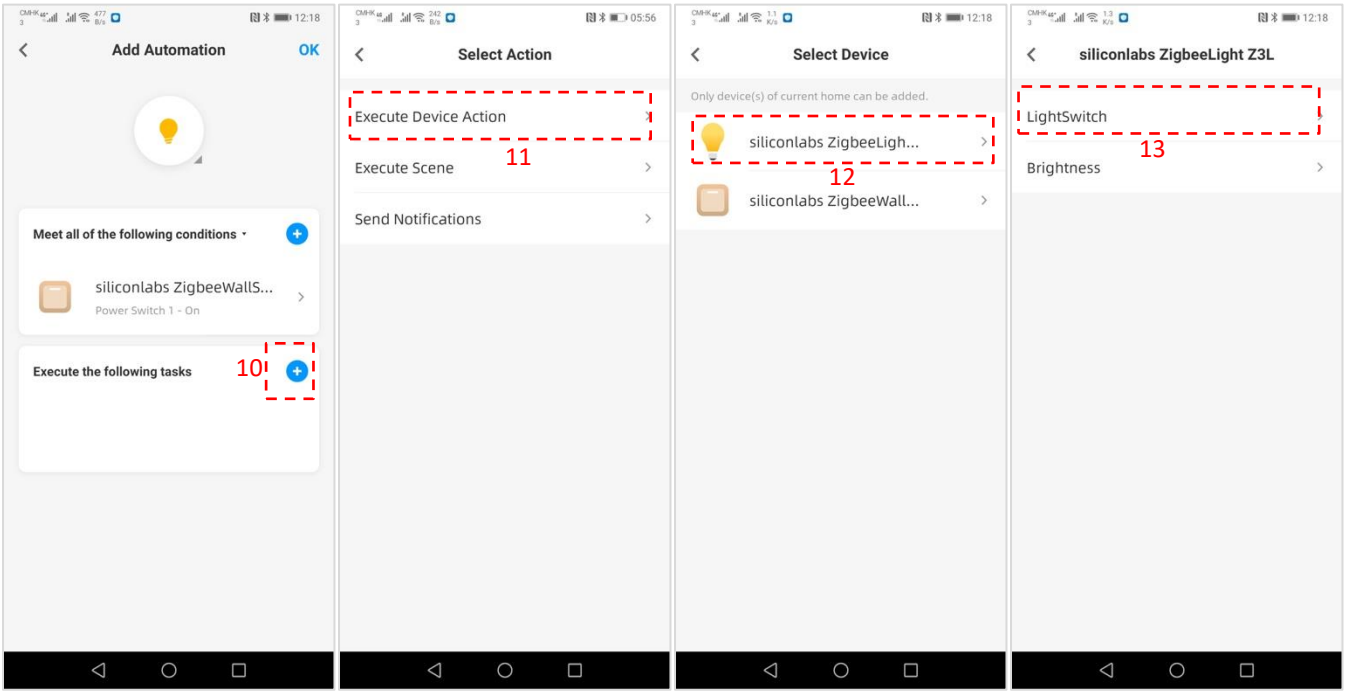
- On the Cloud Intelligence mobile phone App, user may set up the cloud-based rules for gateway's sub-devices to realize automation.
- For example, setting up the rules for the Z3 Switch to control the Z3 Light, i.e. press the Toggle Button on the Z3Switch WSTK to control the LED on/off on the Z3Light WSTK.
- Set up the on-on rule:
  1. Tap the circle icon on the bottom middle to go to the "Intelligence" page
  2. Tap the "Automation" tag
  3. Tap the "+" on the top right to the "Add Automation" page
  4. Tap the middle icon to customize the icon
  5. Choose the icon and the color, then tap "OK"



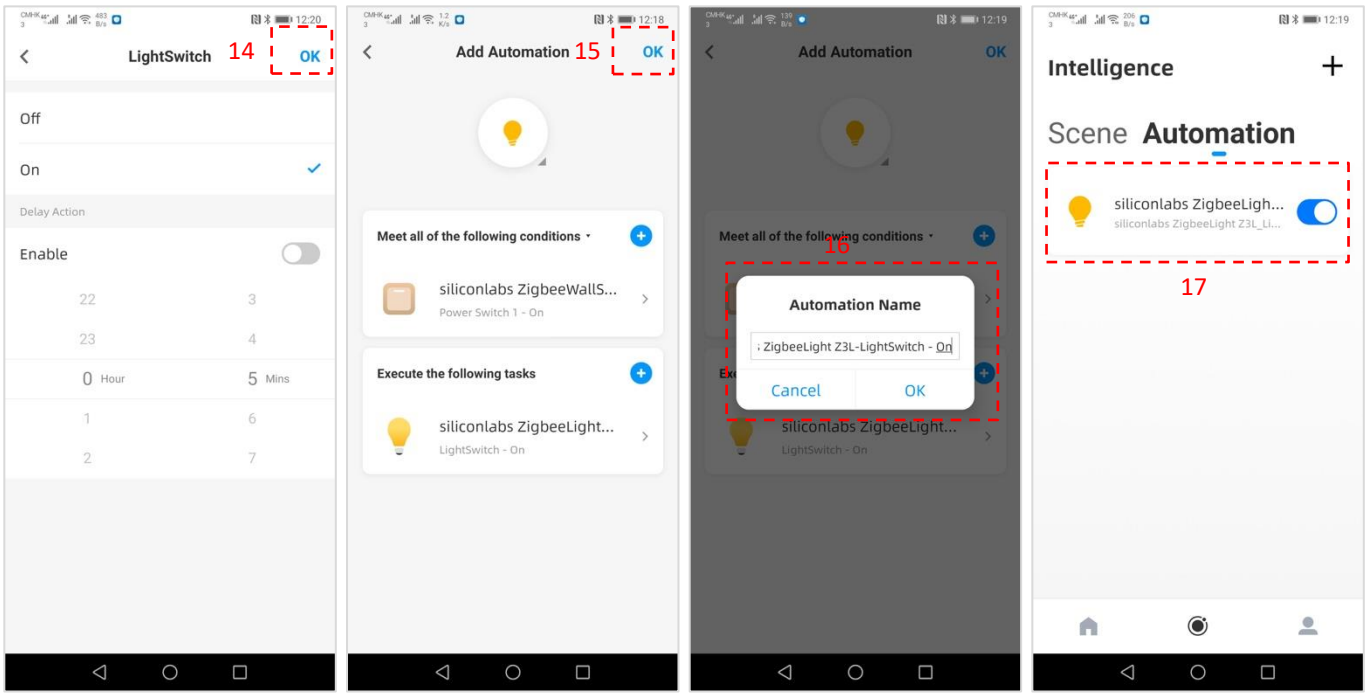
6. Tap the “+” to open the “Add conditions” page
7. Select the “Device Status” and go to the “Select Device” page
8. Select the desired switch device, “siliconlabs ZigbeeWallSW Z3SW”
9. Select “On” and tap “OK”



10. Tap the “+” to open the “Select Action” page
11. Select “Execute Device Action”
12. Select the desired light device, “siliconlabs ZigbeeLight Z3L”
13. Select “LightSwitch”

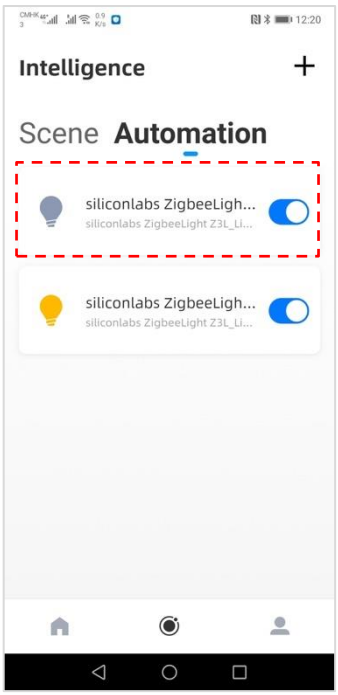


14. Select “On” and tap “OK”
15. Tap the top right “OK” to save
16. Enter the name of the automation rule and tap “OK”
17. The newly created automation rule appears on the “Automation” page



- Set up the off-off rule:

Follow the above steps to create the off-off rule.



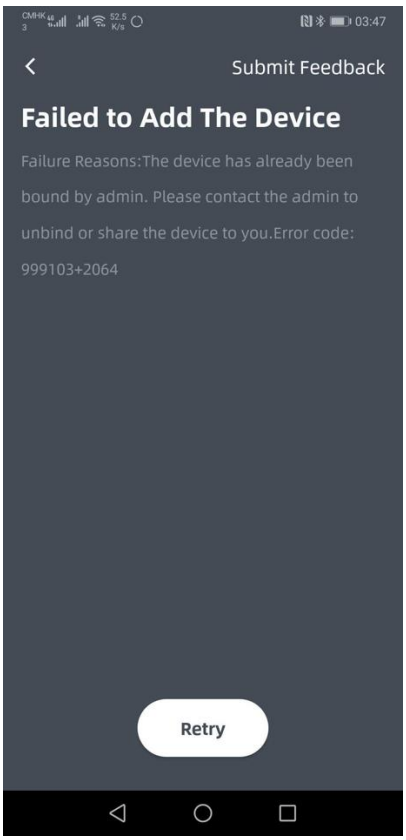
- Press the Toggle Button on the Z3Switch WSTK to control the LED on/off on the selected Z3Light WSTK. The selected Z3Light device's on/off state change can be observed on the mobile App.

Note:

- The Alink TSL Light model only accepts setting its On/Off property while the Z3Switch WSTK sends out the Toggle attribute when the toggle button is pressed. The toggle value is not support by the Alink light model. Thus the Aliyun Zigbee Gateway example code needs to translate the Zigbee Toggle attribute value to the specific On or Off property for Alink light operation.
- However, LCGW doesn't know which Z3Light the Z3Switch is trying to control as the Automation Rule was set on the Cloud. So current translation relies on a static variable to remember the toggle state which may not equal to the actual light state. As a result, the first pressing the Toggle button may not result in actual LED's On/Off change.

3.7 Important Notice

By using the public Cloud Intelligence mobile phone App, no matter the User edition or the Developer edition, once a device, including the gateway itself, is added (bound) to one user account, the other user account will be unable to add (bind) the same device again, i.e. the Triple-unit-group can't be bound unless it's unbound beforehand. See below error message on the mobile App.

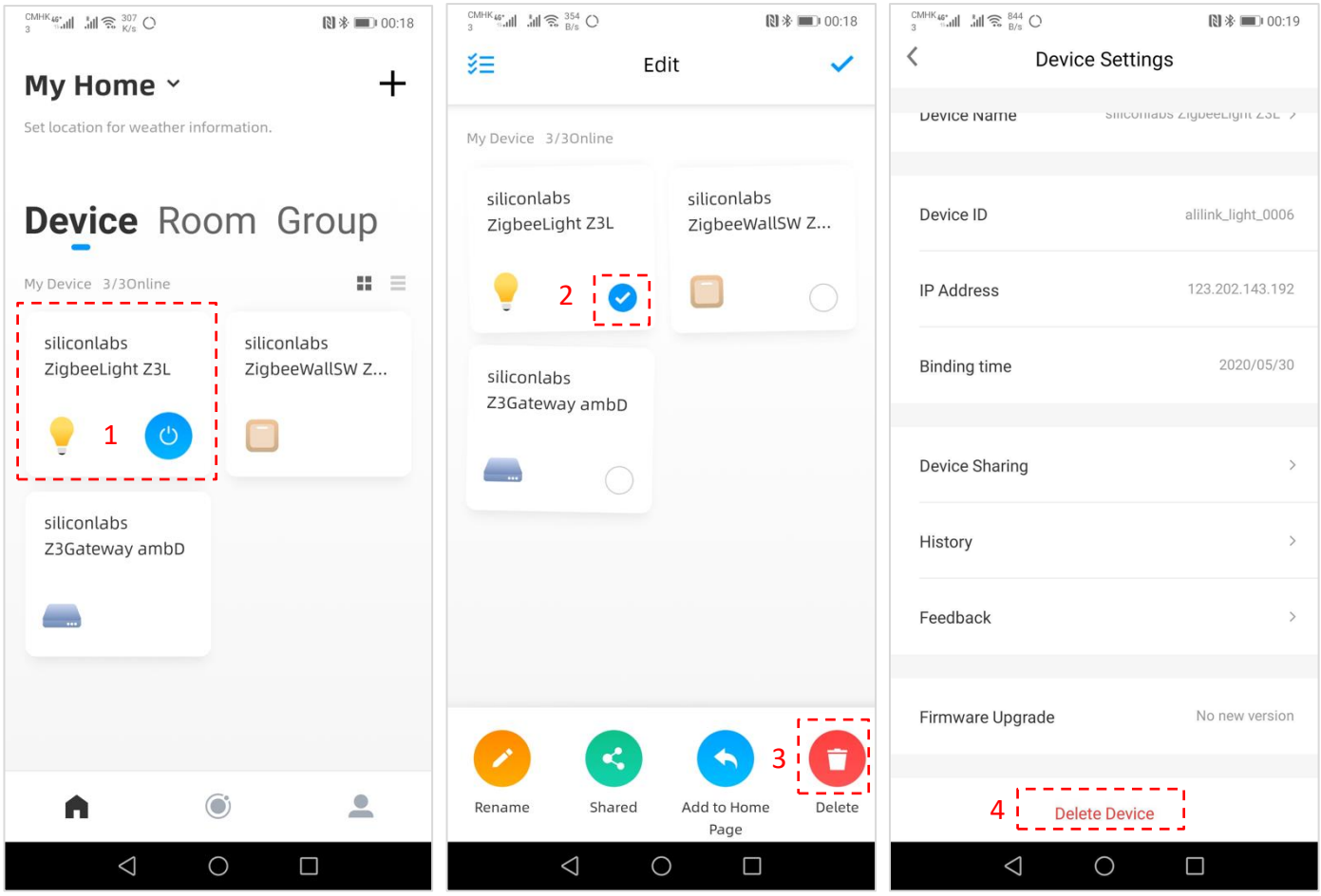


- This is very important when the default pre-defined Triple-unit-groups of the gateway and the test devices have been used in the gateway project sample code in a shared manner among multiple parties. To avoid conflicts of the default pre-defined Triple-unit-groups, a user should
  - use the shared test account below to log into the Cloud Intelligence App.
    - username: silabs_tester@sina.com
    - password: Silabs2014
  - or use his/her own device Triple-unit-groups in the sample code and log into the Cloud Intelligence App by his/her personal account. (Recommended)
- If one does add the LCGW with the default pre-defined Triple-unit-groups into the App logged in with other account rather than the shared test account, one should unbind the LCGW and the underlying devices which are associating with the default pre-defined Triple-unit-groups after his/her evaluation in the mobile phone App. The following will show how to unbind a device.

1. Tap and hold a device from the main page device list to switch to the Edit mode
2. Tap the “Radio” button(s) to select the device(s) for next operation
3. Tap the bottom “Delete” icon button to delete the device, i.e. unbind the device from the mobile phone App

Alternatively, user may tap a device icon in the main page device list and go to its Device Control panel, then tap the top-right “…” icon to open its Device Settings panel, then

4. Scroll down to the very bottom, tap “Delete Device” button and tap again “Delete Device” in the confirmation dialog



The device then will be unbound to the mobile App so that the associated Triple-unit-group is free to be used by other user accounts. After unbinding all the underlying Zigbee devices, we need to unbind the gateway itself too in a similar manner to free its associated Triple-unit-group.

3.8 LCGWv2 CONFIG Button Usage Tips

- Punch COFIG button once, i.e. less than 500ms, to permit device joining, RED (Zigbee) indicator starts fast blinking (100ms on,100ms off)
- Punch CONFIG button again to close device joining. RED indicator restores slow blinking (100ms on, 1900ms off)
- Press and hold CONFIG button for more than 3s until both Blue and Green indicators are on then release it. LCGW will force all joined devices left and create a new Zigbee network. During the resetting process, the Blue and Green indicators will blink together (100ms on, 100ms off).
- Press and hold CONFIG button for more than 8s until three LEDs are all on then release it to reset LCGW to factory default, i.e. unregister cloud & leave network. During the resetting process, the three LEDs will blink together (100ms on, 100ms off). After the resetting process completes, the three LEDs remain on and the LCGW is no longer operable. User needs to press and hold CONFIG button and power cycle LCGW to start Wi-Fi provisioning, see Chapter 3.2.

4. Tmall Genie Control LCGW underlying Zigbee Devices

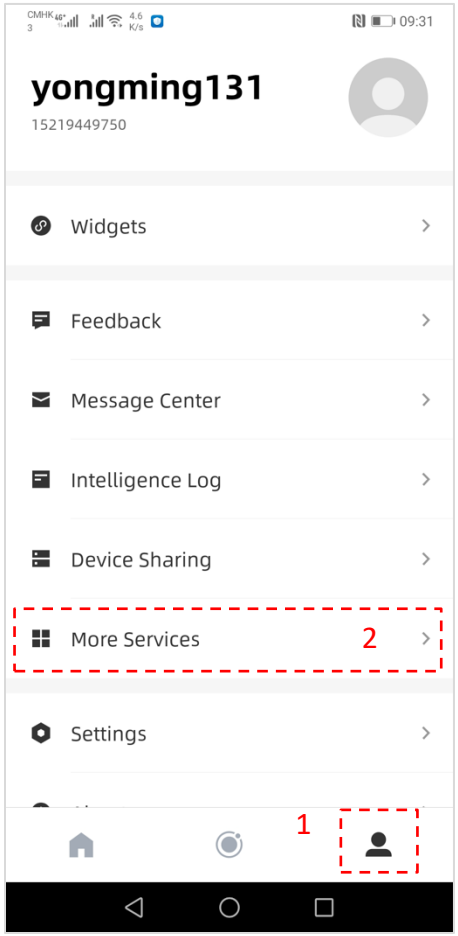
Referring to below Aliyun help document, use the public Cloud Intelligence (云智能) mobile phone App to configure the LCGW underlying Zigbee devices to be controlled by Tmall Genie (天猫精灵) Intelligent speaker.
https://help.aliyun.com/document_detail/131285.html?

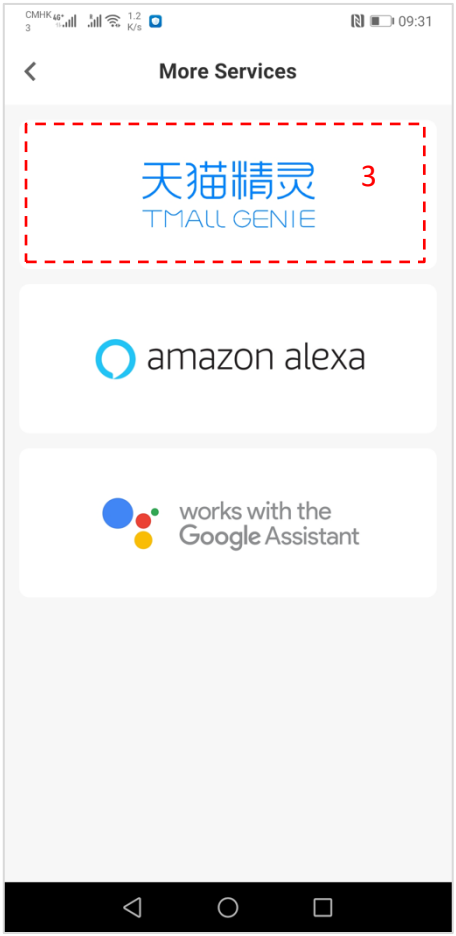
4.1 Prerequisite

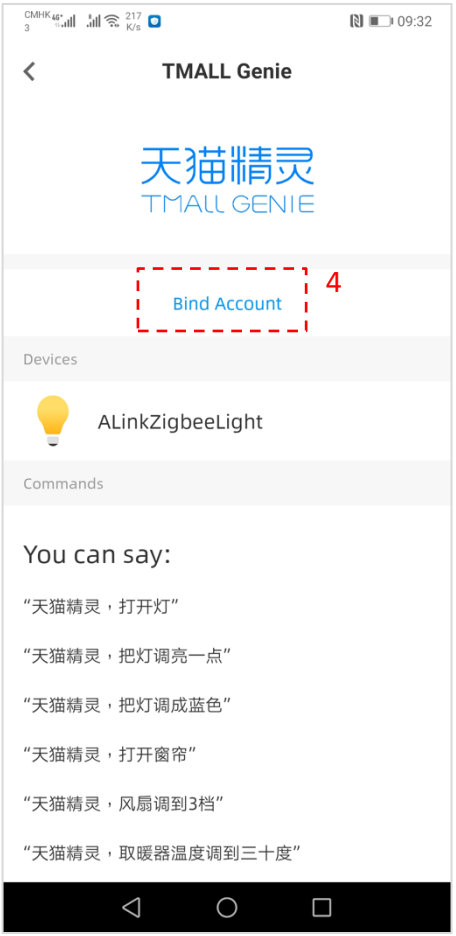
In Aliyun Control Center, the gateway product and the test device products need to be configured to support “Tmall Genie” in their corresponding “Human-machine interface” property settings.

4.2 Procedure

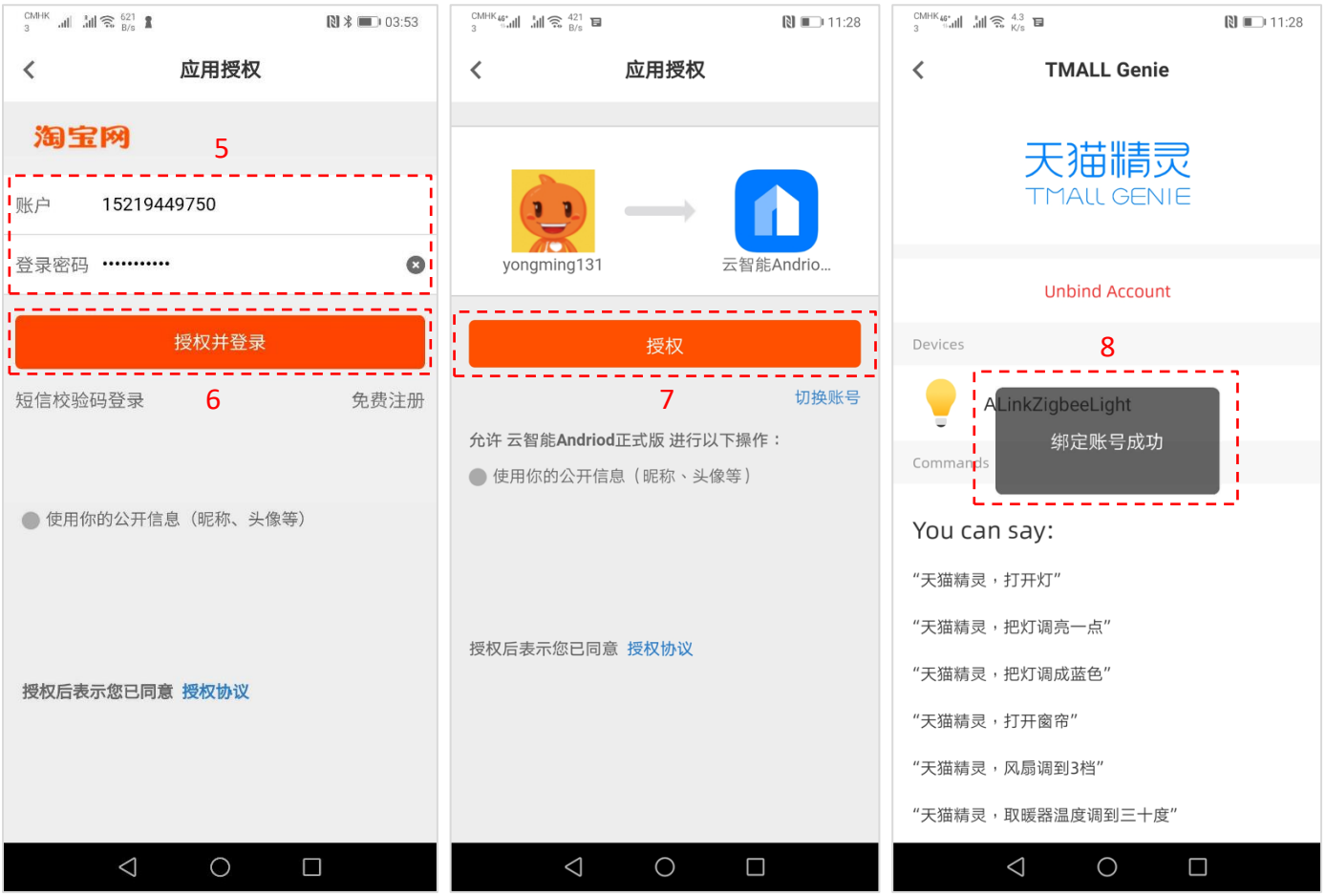
1. Tap the bottom right tap icon “My account” to switch to “My Settings” page
2. Tap from the list the item “More Services” to open corresponding panel
3. Tap from the list the item “TMALL GENIE” to configure the Tmall Genie service
4. In Tmall Genie services configuration dialog, Tmall Genie controllable devices are being listed. Tap the “Bind Account” button







5. Enter the Taobao user account related credentials
6. Tap below “Authenticate and login” (授权並登录) button to continue
7. Tap “Authenticate” (授权) button again in the next dialog showing both Taobao and the Cloud Intelligence accounts figures
8. “Binding Account Success” (绑定帐号成功) is prompted



After this, open the Tmall Genie mobile phone App, tap from the main page bottom right corner “My Home” tap button to switch to the “My home” panel, you should see the LCGW underlying controllable Zigbee devices are listed in this page for confirmation.

You may now try to speak to the Ali Gennie Intelligent Speaker, e.g. “Tmall Genie, turn on all lights” to voice control the Zigbee test light devices.

5. Porting Alibaba iotkit-embedded to AmebaD SDK from Scratch

The source code of Alibaba's Iotkit embedded SDK v3.0.1 stable version can be obtained from Alibaba's GitHub repository, or downloaded from Alibaba web site to get the LTS v3.01 patched source code package:
https://github.com/aliyun/iotkit-embedded/tree/v3.0.1
https://help.aliyun.com/document_detail/96623.html

Aliyun iotkit SDK V3.0.1 branch commit 82270c6 dated 2020-01-14 is used in this AmebaD LCGW lib_iotkit library code porting. Please refer to below Aliyun online document to understand the iotkit feature configuration, code extraction and porting concepts:
https://code.aliyun.com/edward.yangx/public-docs/wikis/user-guide/linkkit/Port_Guide/Extract_Example

5.1 Porting iotkit-embedded to AmebaD SDK

In the future, customer may need to update the iotkit to newer release version or make changes of the iotkit features. Please follow below steps for porting the iotkit-embedded sources to AmebaD SDK as a user library:

1) Setup Ubuntu 18.04 Linux environment or Windows 10 bash environment

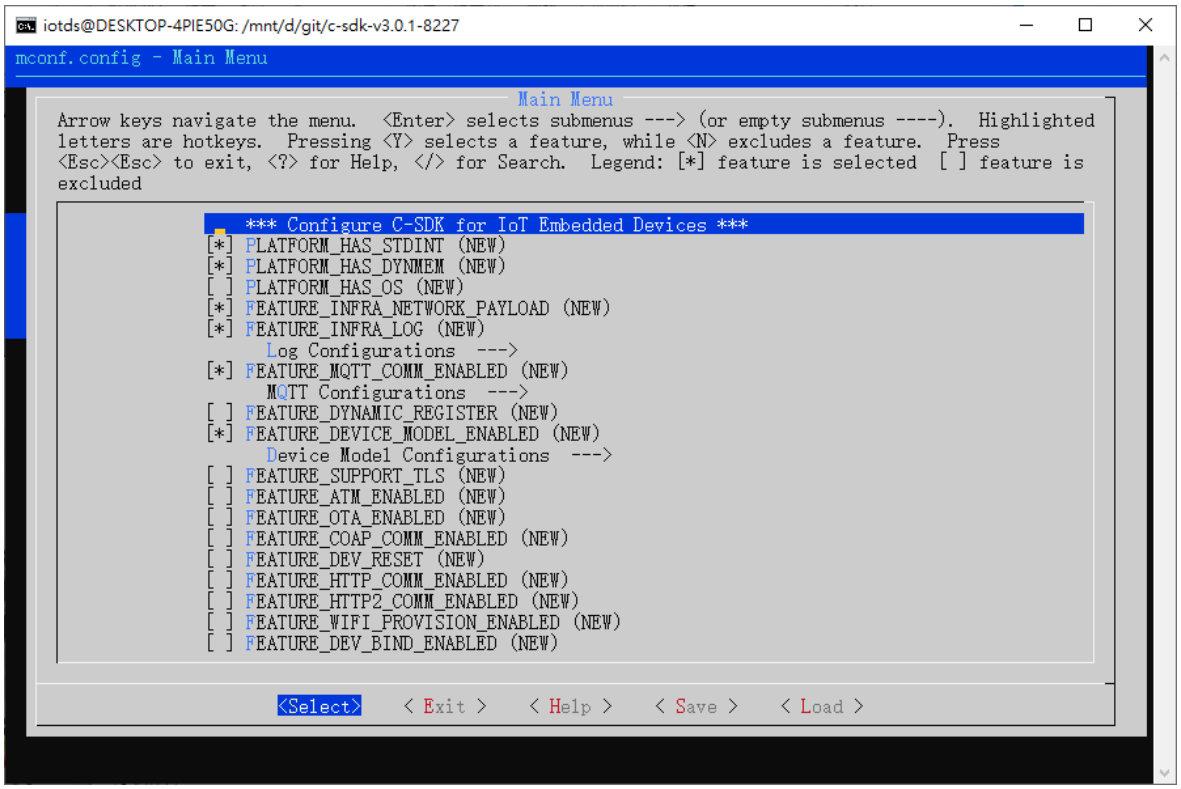
2) Git clone or download the source code Aliyun iotkit SDK V3.0.1 locally

```
> $ git clone https://github.com/aliyun/iotkit-embedded.git
```

3) Run menuconfig to configure iotkit features

```
> $ cd iotkit-embedded
> $ make menuconfig
```

The following configuration menu will be shown. You may navigate through the menu to enable/disable different features.



Below list shows the Iotkit configuration of current LCGW, enable/disable the same feature set in the menuconfig accordingly and save your config setting when existing the config.

```
*** Configurate C-SDK for IoT Embedded Devices ***
[*] FEATURE_HAS_STDINT (NEW)
[*] FEATURE_HAS_DYNMEM (NEW)
[ ] FEATURE_HAS_OS (NEW)
[*] FEATURE_INFRA_NETWORK_PAYLOAD (NEW)
[*] FEATURE_INFA_LOG (NEW)
    Log Configurations -→
        [*] MUTE LEVEL of FLOW (6) (NEW)
        [*] MUTE LEVEL of FLOW (5) (NEW)
        [*] MUTE LEVEL of FLOW (4) (NEW)
        [*] MUTE LEVEL of FLOW (3) (NEW)
        [*] MUTE LEVEL of FLOW (2) (NEW)
        [*] MUTE LEVEL of FLOW (1) (NEW)
[*] FEATURE_MQTT_COMM_ENABLED (NEW)
    MQTT Configurations --->
        [*] FEATURE_MQTT_DEFAULT_IMPL (NEW)
        [ ] FEATURE_MQTT_DIRECT (NEW)
        [ ] FEATURE_ASYNC_PROTOCOL_STACK (NEW)
[ ] FEATURE_DYNAMIC_REGISTER
[ ] FEATURE_DEVICE_MODEL_ENABLE
    Device Model Configurations --->
       [*] FEATURE_DEVICE_MODEL_GATEWAY
       [ ] FEATURE_ALCS_ENABLED (NEW)
       [ ] FEATURE_SUB_PERSISTENCE_ENABLED (NEW)
       [*] FEATURE_DEVICE_MODEL_SUBDEV_OTA
       [ ] FEATURE_DEVICE_MODEL_SHADOW (NEW)
-*- FEA TURE_SUPPORT_TLS
[ ] FEATURE_ATM_ENABLED (NEW)
[*] FEATURE_OTA_ENABLED
[ ] FEATURE_COAP_COMM_ENABLED
[*] FEATURE_DEV_RESET
[ ] FEATURE_HTTP_COMM_ENABLED (NEW)
[ ] FEATURE_HTTP2_COMM_ENABLED (NEW)
[*] FEATURE_WIFI_PROVISION_ENABLED
     WiFi Provision Configurations --->
        [ ] FEATURE_AWSS_SUPPORT_SMARTCONFIG
        [ ] FEATURE_AWSS_SUPPORT_ZEROCONFIG (NEW)
        [ ] FEATURE_AWSS_SUPPORT_AHA, ADHA (NEW)
        [*] FEATURE_AWSS_SUPPORT_DEV_AP
-*- FEATURE_DEV_BIND_ENABLED
```

Next time when you need to modify the iotkit config settings, you may load from file “make.setting”, make related change and save the settings back.

4) Run code extraction and compile the iotkit examples x86 executables.

```
> $ make clean
> $ make
```

Note that source files code extraction and compilation is shown in the building output. Furthermore, the custom iotkit dependent HAL/Wrapper functions are listed for reference.

```
HAL/Wrapper Function List:
001 HAL_Aes128_Cbc_Decrypt
002 HAL_Aes128_Cfb_Decrypt
……
058 HAL_Wifi_Get_IP
059 HAL_Wifi_Get_Mac
```

The x86 example executables build is generated in folder: `iotkit-embedded/output/release/bin/`
The extracted source codes for platform porting are located in folder: `iotkit-embedded/output/eng/`

5) Verify the x86 sample gateway executable connection to Aliyun

Run the x86 example gateway executable:

```
> $./output/release/bin/linkkit-example-gateway
```

Check from the console output traces to see if the iotkit gateway works and is able to connect to Aliyun successfully

6) Platform porting of the extracted iotkit sources to AmebaD SDK

While the extracted source codes are located: `iotkit-embedded/output/eng/`, contents inside will be used to replace the contents inside `lib_iotkit/` shown below in red dotted boarder:

```
project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/asdk/make/project/lib_iotkit/
    │
    └─lib_iotkit
        │  Makefile
        │  sdk_include.h
        ├─certs
        ├─CoAPPacket
        ├─coap_server
        │  └─server
        ├─dev_bind
        │  └─impl
        │      ├─awss_reset
        │      └─os
        ├─dev_model
        ├─dev_reset
        ├─dev_sign
        ├─infra
        ├─mqtt
        ├─ota
        ├─wifi_provision
        │  ├─dev_ap
        │  └─frameworks
        │      ├─aplist
        │      ├─ieee80211
        │      ├─statics
        │      └─utils
        └─wrappers
            │  wrappers_defs.h
            ├─external_libs
            │  └─mbedtls
            │      └─include
            │          └─mbedtls
            ├─freertos
            └─tls
```

In step 4, the build console output has already listed out the HAL/Wrapper functions, and dummy wrapper functions are generated after code extraction for passing the initial cross-compilation. These wrapper functions need to be implemented on this FreeRTOS target ported platform.

In the `lib_iotkit/wrappers/` folder of Z3GatewayFreeRTOS Alibaba Cloud (Aliyun) example software, the AmebaD FreeRTOS platform specific wrapping functions have already been implemented, and are not required to be further modified unless new features have been enabled or updated in iotkit config and new wrapper functions need to be implemented.

Note that a diff GUI tool, e.g. WinMerge on Windows, Meld on Linux or DiffMerge on multi-platforms, is needed to merge with new version or feature extracted iotkit sources.

In order to get the original iokit sources successfully built in the AmebaD SDK, a few small modifications need to be made:

1. infra/infra_config.h
   - Add the #inclu“e "platform_opt”.h" line to include AmebaD platform options to achieve platform specific compilation
   - Comment out below iotkit features as they are not implemented in current LCGW release yet:

```
//#define DEVICE_MODEL_SUBDEV_OTA //not implemented
//#define HAL_KV //not implemented
……
//#define OTA_ENABLED //not implemented
```

2. infra/infra_compat.h

   Add the PLATFORM_FREERTOS compilation switch lines as the LIST_XXXX, list_xxxx macros and static functions have already been defined in AmebaD SDK, simply use AmebaD marcos directly:

```
#ifdef PLATFORM_FREERTOS
#define list_entry_number               dlist_entry_number
#else
#define LIST_HEAD                       AOS_DLIST_HEAD
#define LIST_HEAD_INIT                  AOS_DLIST_INIT
……
#endif //PLATFORM_FREERTOS
```

3. infra/infra_list.h

   Add the PLATFORM_FREERTOS compilation switch lines to exclude the duplicated list implementation in iotkit source.

```
#ifdef PLATFORM_FREERTOS
#inclu“e "dlis”.h"
typedef struct list_head dlist_t;
#else
/*
 * Get offset of a member variable.
….
#endif //PLATFORM_FREERTOS
```

4. dev_bind/impl/awss_reset/awss_dev_reset.c

   Add the HAL_KV compilation switch lines to exclude the not-implemented HAL_kv_Set() wrap functions.

```
    #ifdef HAL_   HAL_Kv_Set(AWSS_KV_RST, &rst, sizeof(rst), 0);
#endif
```

5. dev_bind/impl/os/os_misc.c

   Add the PLATFORM_FREERTOS compilation switch lines to exclude the os_zalloc() already defined in AmebaD SDK, simply use AmebaD version directly:

```
#ifndef PLATFORM_FREERTOS
    void *os_zalloc(uint32_t size)   void *ptr = HAL_Malloc(size);
    if (ptr != NULL) {
            memset(ptr, 0, size     return ptr;
}
#endif //PLATFORM_FREERTOS
```

6. wifi_provision/dev_ap/awss_dev_ap.c

   Add the PLATFORM_FREERTOS compilation switch lines to specify longer wait time on AmebaD platform:

```
    #ifdef PLATFORM_FREERT   HAL_SleepMs(4000);  /* wait for dev ap to work well */
    #el   HAL_SleepMs(1000);  /* wait for dev ap to work well */
#endif //PLATFORM_FREERTOS
```

Modify the lib_iotkit ***Makefile*** accordingly for c source files addition/removal and dependent header files include paths

`project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/asdk/make/project/lib_iotkit/Makefile`

7) lib_iotkit Library Compilation

Build the project_hp target of Cortex-M4 application core:

```
> $ cd project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/
> $ make clean
> $ make all
```

If build is passed, the lib_iotkit.a will be generated in below folder:

```
Proje54ealtektek_amebaD_va0_example/GCC-RELEASE/project_hp/asdk/lib/application/
```

You may need to address further building missing dependences and linkage conflicts related to the high-level cloud gateway application codes.

After the entire build is passed, km4_boot_all.bin and km0_km4_image2.bin will be generated in below folder:

```
project/Realtek_amebad_va0_example/GCC-RELEASE/project_hp/asdk/image/
```

5.2 Integrating another Cloud to LCGW

In general, developers interested in using the AmebaD LCGW gateway solution to bridge to another cloud service provider other than Aliyun can refer to the concepts and direction in chapter 5 about how Alibaba iotkkit-embedded is ported.

1) Prerequisite

Another Cloud service provider should provide:

- A full source standard C Cloud SDK provided for platform porting and integration, C++ could be support in AmebaD SDK
- The Cloud SDK should support gateway type device feature already and be able to manage child devices connecting to cloud
- The Cloud SDK should have OS/kernel dependences abstracted out for platform specific implementation

2) Platform porting

Most of current AmebaD Alibaba iotkit wrapper functions can be reused after slightly modification for this purpose. Those abstracted OS/kernel features like memory alloc/free management, task/thread creation/termination, semaphores, mutex, wait/timers, persistent storage/filesystem, tcp/udp sockets, tls/ssl and etc.

3) Cloud Gateway application integration

In theory, Silicon Labs's Z3Controller library (lib_z3ctrl.a) should be self-contained and provide the required API for generally any cloud services integration.

However, as mentioned earlier, each Cloud SDK may have their own gateway and device-to-cloud data communication protocol and data format. Files like cloud_gateway.c and cloud_gateway_interface.c need to be modified and new data format conversion needs to be implemented accordingly.

wifi_prov.c needs to be modified accordingly if different cloud-specific Wi-Fi provisioning mechanism is adapted other that AmebaD supported Bluetooth-LE provisioning method.

6. Document Revision history

Revision 0.1

December 31, 2019

- Initial release

Revision 0.2

February 29, 2020

- Update document to LCGW ported to Realtek AmebaD SDK
- Added document revision history

Revision 0.3

April 20, 2020

- Released with LCGW v1.0 RC1 software package
- Added TOC, pictures for Realtek RTL8720CS and LCGWv2 boards and delete CLI commands

Revision 1.0

May 31, 2020

- Released with LCGW v1.0 GA software package
- Updated 1.1 contents of LCGW kit recent photo and AmebaD board prototyping RTOS gateway
- Enriched 3.2, 3.3 mobile app LCGW and Z3Light adding more screenshots and details
- Added 3.4 Z3Switch operation and 3.5 mobile app Automation Setup

Revision 1.0.1

Jun. 17, 2020

- Added 1.4 Realtek AmebaD SDK
- Revised some 1.7 Gateway Compile Options
- Corrected 1.9 Compilation

Revision 1.0.2

Jun. 23, 2020

- Added 3.1 App Configuration
- Revised 3.7 Important Notice
- Minor formatting and revise