Using a Smart Home IoT Application with EMSK

Overview

embARC is an open software platform designed to help accelerate the development and production of embedded systems based on DesignWare© ARC© processors.

This article provides instructions on how to establish connection between the EMSK and Amazon Web Services Internet of Things (AWS IoT) cloud in a simulated smart home application. AWS IoT is a managed cloud platform that lets connected devices securely interact with cloud applications and other devices. It supports Message Queue Telemetry Transport (MQTT) and provides authentication and end-to-end encryption.

NOTE: This article assumes the reader is already familiar with embARC. If this is your first project with embARC, please start by first reading our "Quick Start" article to ensure your development environment is properly set before you begin.

Please visit https://www.embarc.org/index.html for more information on embARC and https://aws.amazon.com/iot/ for more information on AWS IoT.

Development Environment

The development environment used in this article is the following:

Development host operating system:

- **Windows 7**

embARC Version:

- **embARC 2016.05**

Development Toolchain for Target Platform:

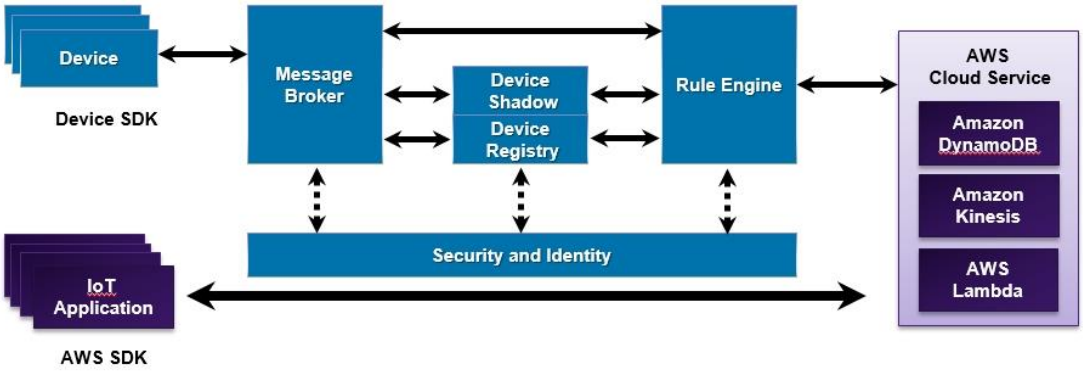
- **GNU Toolchain for DesignWare ARC Processors, version 2015.12**

Target platform:

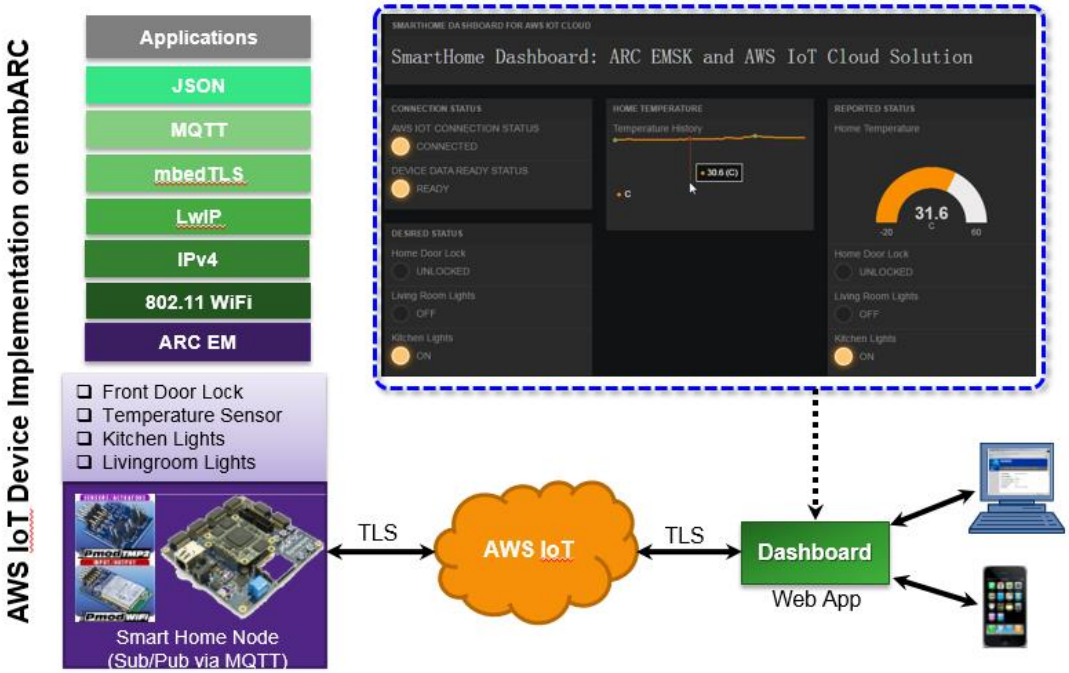
- **ARC EM Starter Kit (EMSK), version 2.2**

AWS IoT

AWS offers IT infrastructure services to businesses in the form of cloud computing. AWS IoT provides secure, bi-directional communication between Internet-connected things and the AWS cloud. A secure mechanism is supported to publish and receive messages with using the MQTT protocol.



MQTT is a machine-to-machine (M2M)/IoT connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. The message broker in AWS IoT can route messages with MQTT topics from publishing clients (devices) to subscribing clients (devices).



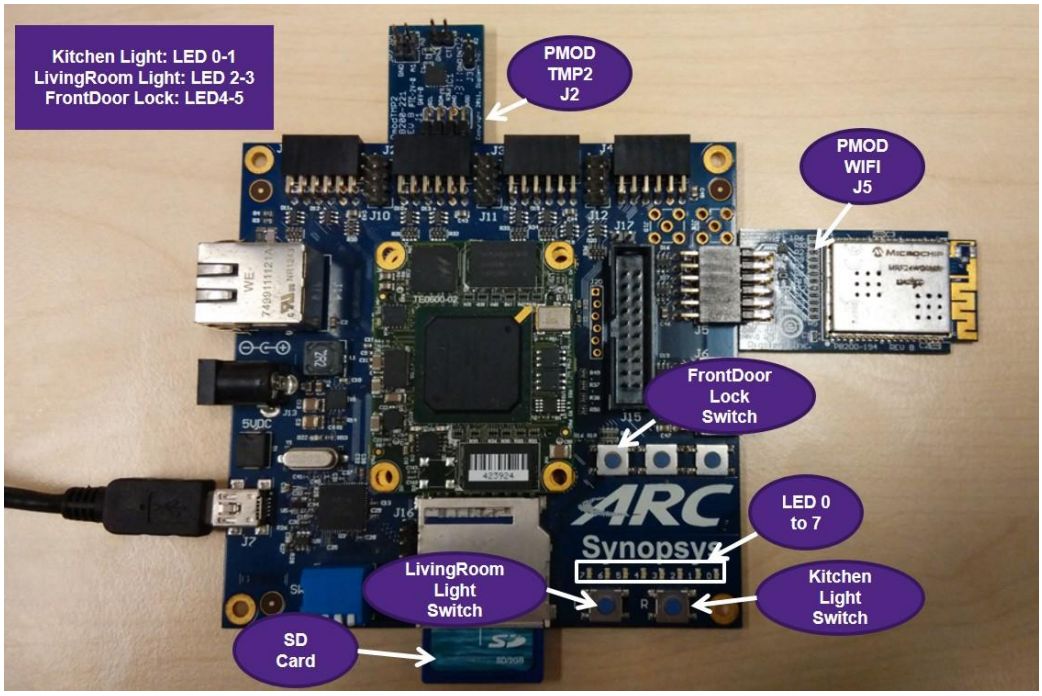
Devices are registered in the device registry. JavaScript Object Notation (JSON) documents are created to store and retrieve device state information. All communications to and from AWS IoT must be encrypted over TLS.

The AWS IoT Device C SDK for embedded platforms has been ported and optimized for use with embARC. The EMSK-based smart home node implementation simulates functionality of a front door lock, temperature sensor, kitchen lights and living room lights. A dedicated freeboard-based HTML5 Web App was developed to provide a dashboard to monitor and control the smart home node.

Please visit http://aws.amazon.com/documentation/iot/ for more information on AWS IoT.

Preparation

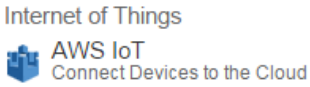
1) Hardware: EMSK 2.2, PmodTMP2, PmodWiFi, SD Card.
2) The Dashboard Web App is located at http://foss-for-synopsys-dwc-arc-processors.github.io/freeboard/ .The JSON file **dashboard-smarthome-singlething.json** in **embARC\example\freertos\iot\aws\smarthome_demo must be loaded into the Dashboard Web App** using web browsers supporting HTML5.
3) Set WiFi network & hotspot names to default values: SSID: **embARC**, WPA2 PSK Password: **qazwsxedc**. User can also modify sources to watch their desired access point configuration.



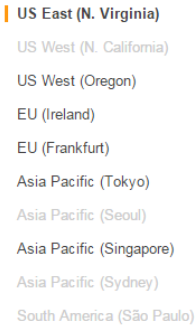
Creating and setting smart home node

1) Create an AWS account in http://aws.amazon.com. Amazon offers various account levels, including a free tier for AWS IoT.

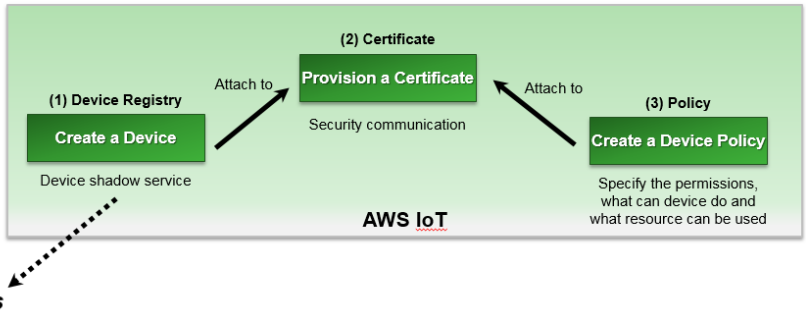
2) Log into AWS console and choose AWS IoT.



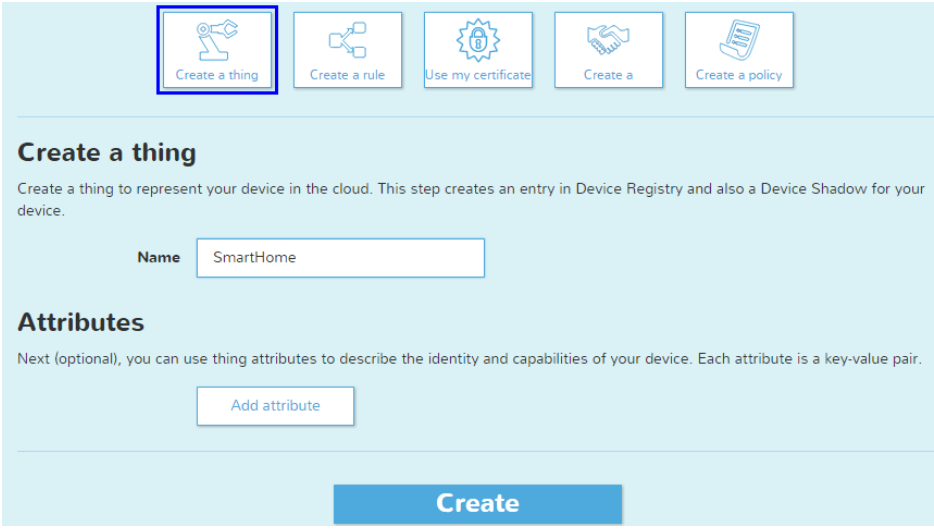
3) Choose an appropriate IoT server in the top right corner of the AWS IoT console page.



4) Create your smart home node in the thing registry and generate X.509 certificate for the node. Create an AWS IoT policy. Then attach your smart home node and policy to the X.509 certificate.

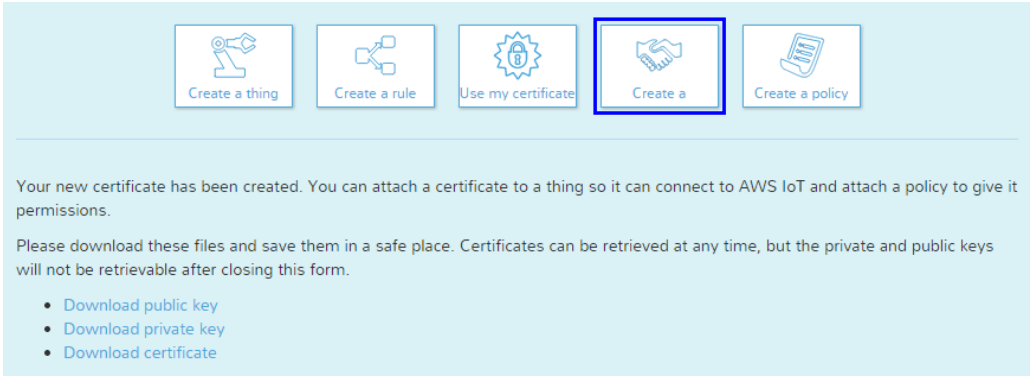


a) Click "**Create a resource**" and "**Create a thing**" to create your smart home node named "**SmartHome**".

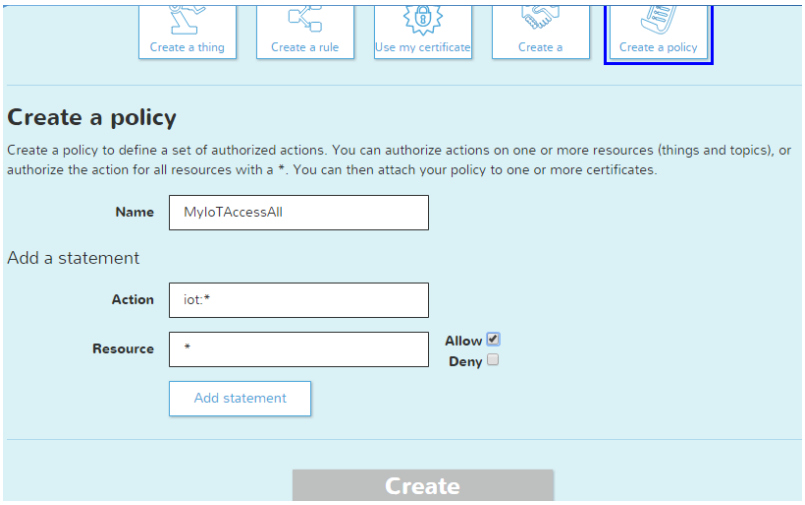


b) Create a certificate ("**1-Click certificate create**") to authenticate your smart home node connection to AWS IoT. Click "**Activate**" option to activate the generated certificate. Download the three files, public key, private key and certificate named **public.pem.key**,

private.pem.key and **certificate.pem.crt**. Rename **private.pem.key** as **privateKey.pem** and **certificate.pem.crt** as **cert.crt** respectively.

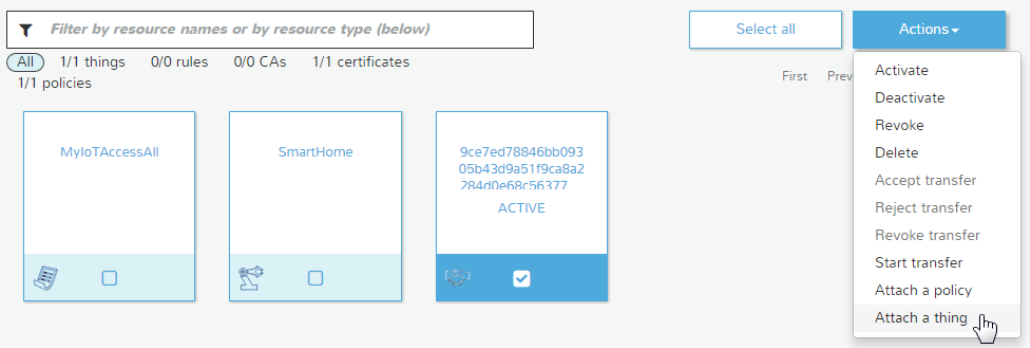


c) Choose "**Create a policy**" to create an AWS IoT policy. Type "**MyIoTAccessAll**" in the **Name** field. Type "**iot:***" and "*" in the **Action** and **Resource** field to allow access to all AWS IoT resources. Select the "**Allow**" check box and choose "**Add Statement**". Then choose "**Create**".



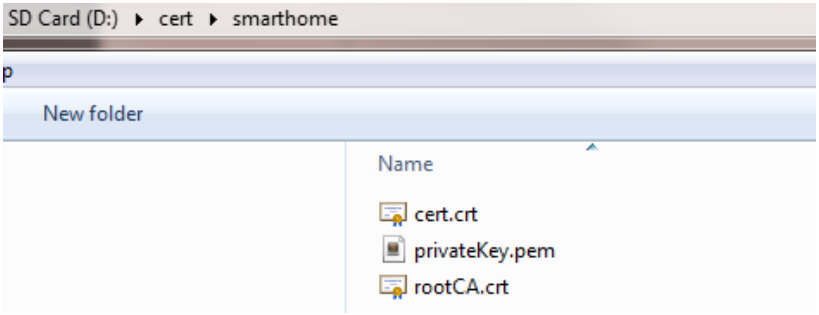
d) Choose your device certificate in the AWS IoT console. Choose "**Attach a policy**" from the "**Actions**" menu. Then type the name of the AWS IoT policy "**MyIoTAccessAll**" in the confirm dialog box. Choose "**Attach**" to complete the setting.

e) Choose your device certificate in the AWS IoT console. Choose "**Attach a thing**" from the "**Actions**" menu. Then type the name of the thing "**SmartHome**" in the confirm dialog box. Choose "**Attach**" to complete the setting.

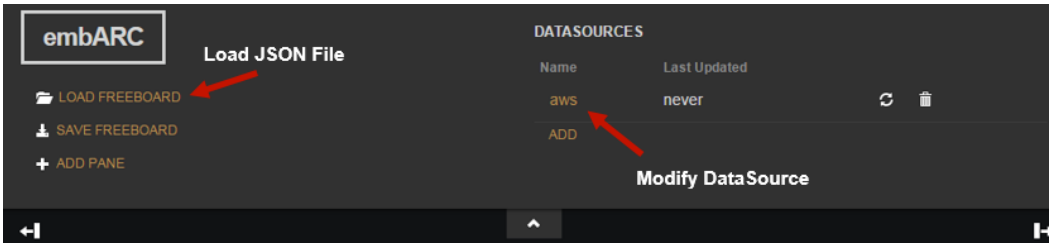


Note: See **Getting Started with AWS IoT** https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html for more information.

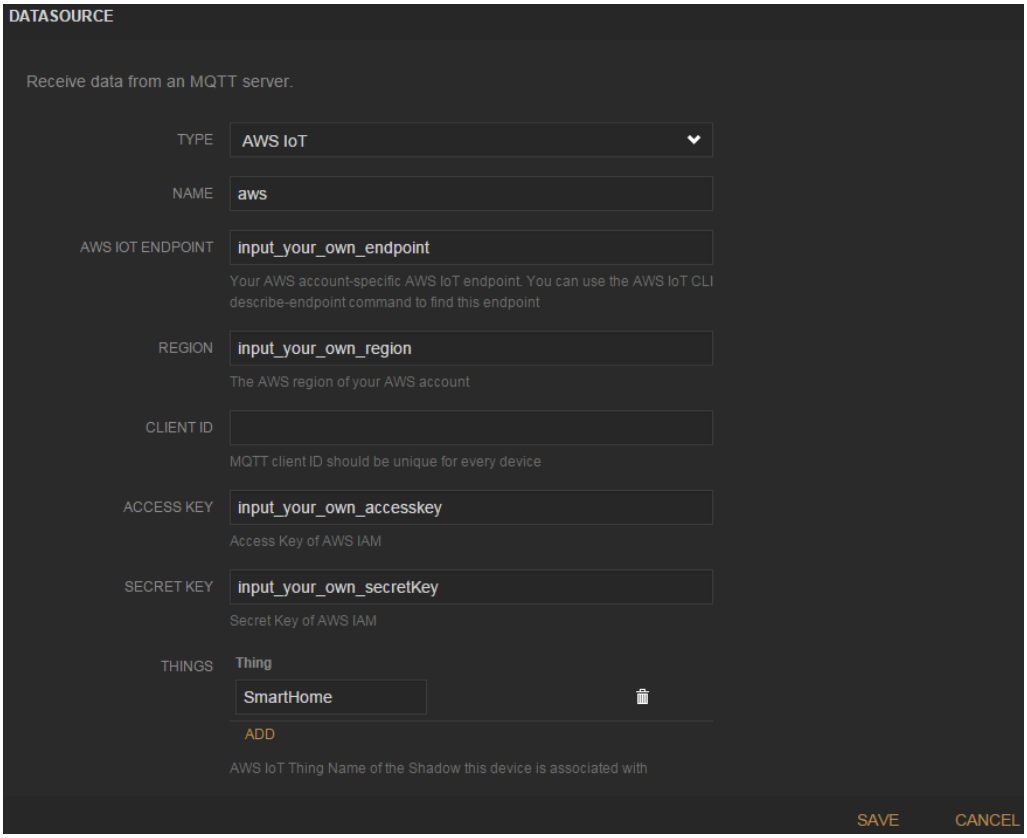
5) Download the root CA certificate from https://www.symantec.com/content/en/us/enterprise/verisign/roots/VeriSign-Class%203-Public-Primary-Certification-Authority-G5.pem . Rename it **rootCA.crt**.
Copy the certificate files **cert.crt**, **privateKey.pem** and **rootCA.crt** to folder **cert\smarthome**.
Insert the SD card to your PC, and copy the certificate folder **cert** to the SD Card root.



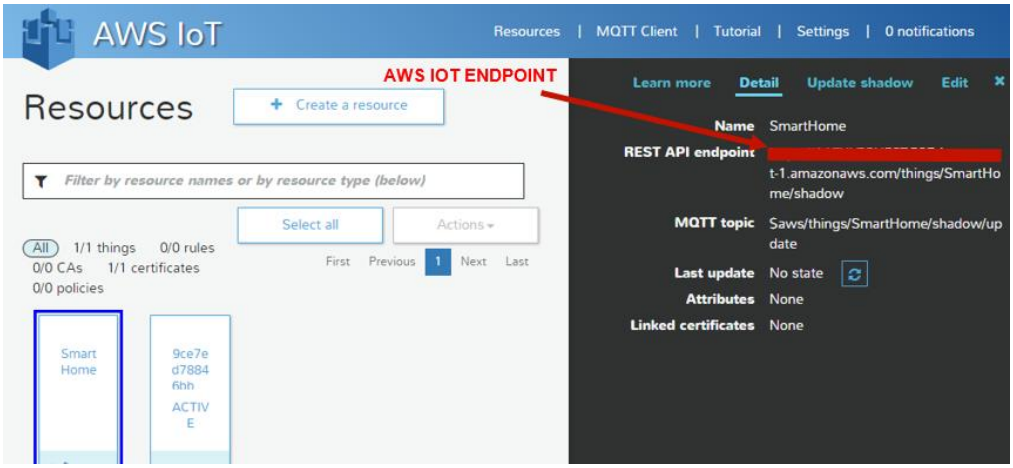
6) Open the Web App in a web browser and load the configuration file **dashboard-smarthome-singlething.json** obtained from **embARC\example\freertos\iot\aws\smarthome_demo**.
The dashboard can be loaded automatically.



7) Click "**ADD**" to go to **DATASOURCE** page and fill up the forms.
   a) TYPE: Choose **AWS IoT**.
   b) NAME: Name is **aws**.



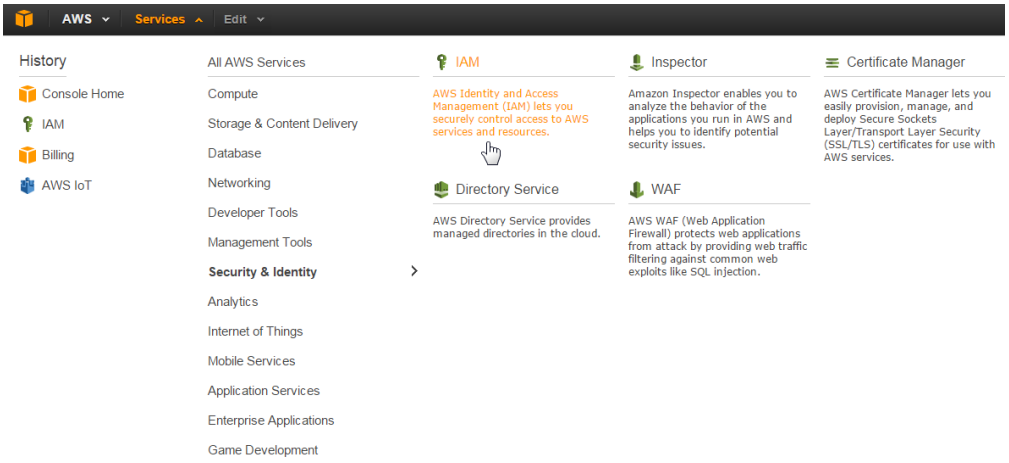
c) AWS IOT ENDPOINT: Go to AWS IoT console and click your smart home node "**SmartHome**". Copy the content **XXXXXXXXXXXXXX.iot.us-east-1.amazonaws.com** in REST API endpoint to AWS IOT ENDPOINT.



d) REGION: Copy the AWS region of your smart home node in REST API endpoint to REGION. For example, **https://XXXXXXXXXXXXXX.iot.us-east-1.amazonaws.com/things/SmartHome/shadow**. REGION is **us-east-1**.

e) CLIENT ID: Leave it blank as default.

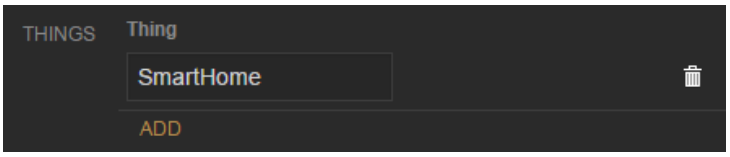
f) ACCESS KEY and SECRET KEY: Go to AWS Services page and click "**IAM**".



Go to User page and click "**Create New Users**". Enter User Names "**AWSIoTUser**". Then download user security credentials, **Access Key ID** and **Secret Access Key**. Copy **Access Key ID** to ACCESS KEY and **Secret Access Key** to SECRET KEY.

Go to User page and click "**AWSIoTUser**". Click "**Attach Policy**" to attach "**AWSIoTDataAccess**" to "**AWSIoTUser**".

g) THINGS: AWS IoT thing name "**SmartHome**".



h) Click "**Save**" to finish the setting.

Building and running AWS IoT smart home example

1) The AWS IoT thing SDK for C has been ported to embARC. Check the above steps in order for your IoT application to work smoothly.
   Go to **\embARC\example\freertos\iot\aws\smarthome_demo**. Modify **aws_iot_config.h** to match your AWS IoT configuration.
   The macro **AWS_IOT_MQTT_HOST** can be copied from the REST API endpoint in AWS IoT console. For example, **https://XXXXXXXXXXXXXX.iot.us-east-**

1.amazonaws.com/things/SmartHome/shadow. **AWS_IOT_MQTT_HOST** should be **XXXXXXXXXXXXX.iot.us-east-1.amazonaws.com**.

```
// Get from console
// ================================================
#define AWS_IOT_MQTT_HOST              "XXXXXXXXXXXXX.iot.us-east-1.amazonaws.com" ///< Cus
#define AWS_IOT_MQTT_PORT              8883 ///< default port for MQTT/S
#define AWS_IOT_MQTT_CLIENT_ID         "csdk-SH" ///< MQTT client ID should be unique for ev
#define AWS_IOT_MY_THING_NAME          "SmartHome" ///< Thing Name of the Shadow this device
#define AWS_IOT_ROOT_CA_FILENAME       "rootCA.crt" ///< Root CA file name
#define AWS_IOT_CERTIFICATE_FILENAME   "cert.crt" ///< device signed certificate file name
#define AWS_IOT_PRIVATE_KEY_FILENAME   "privateKey.pem" ///< Device private key filename
// ================================================
```

Note: The macro **AWS_IOT_MY_THING_NAME** must be "**SmartHome**".

2) Use USB cable to connect the EMSK board. Set the baud rate of the terminal emulator to 115200.

   NOTE: See application note **Quick Start Guide** https://www.embarc.org/help.html#notes for more information.

3) Insert the SD Card into the EMSK board SD Card slot. Run the AWS IoT application using JTAG.

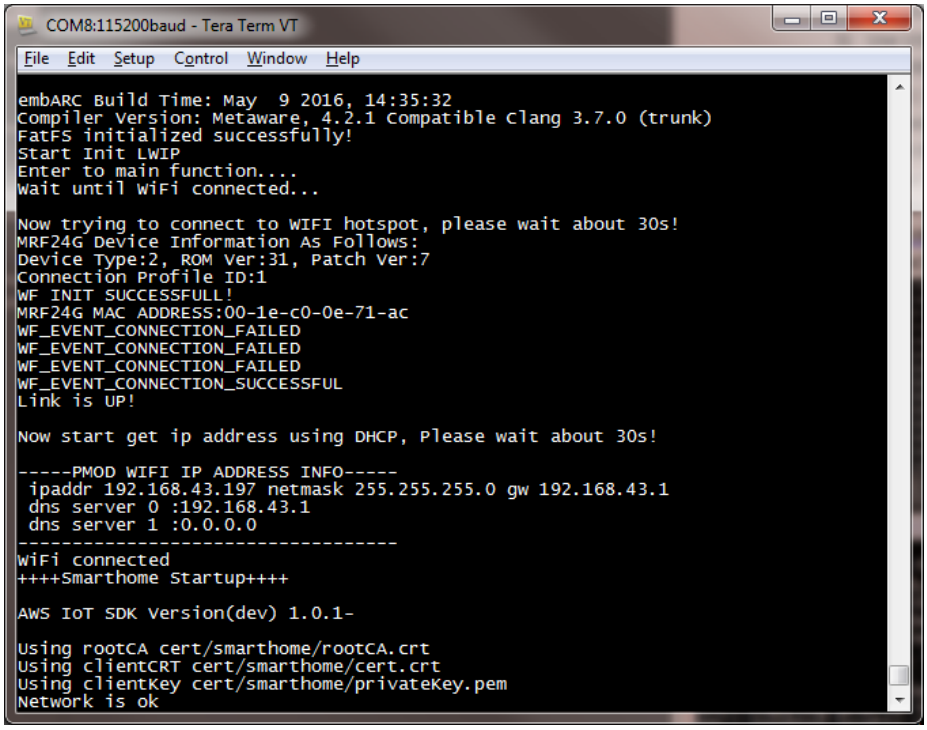
   a. Go to **\embARC\example\freertos\iot\aws\smarthome_demo** in command line.

   b. Enter "***make TOOLCHAIN=gnu BD_VER=22 CUR_CORE=arcem7d***".

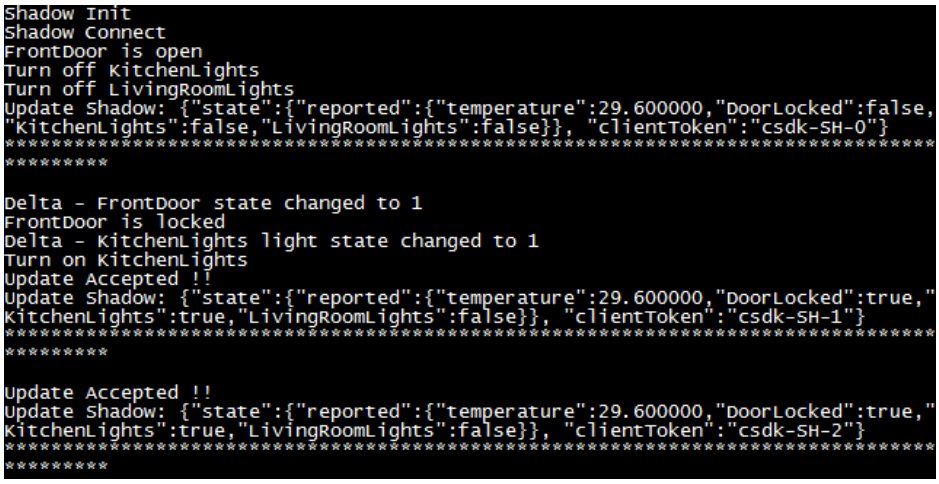
      NOTE: Make sure you have selected the right configuration on EMSK via dipswitches and that you have reset the board (button above "R") and confirmed ARC EM7D configuration.

```
"Compiling          : " ../../../../../middleware/mbedtls/embARC/timing_alt.c
"Compiling          : " ../../../../../middleware/mbedtls/embARC/net_alt.c
"Archiving          : " obj_emsk_22/gnu_arcem7d/libmidmbedtls.a
"Creating Directory : " obj_emsk_22/gnu_arcem7d/freertos
"Compiling          : " ../../../../../os/freertos/portable/Synopsys/ARC_EM/arc_freertos_exceptions.c
"Compiling          : " ../../../../../os/freertos/portable/Synopsys/ARC_EM/port.c
"Compiling          : " ../../../../../os/freertos/portable/Synopsys/ARC_EM/freertos_tls.c
"Compiling          : " ../../../../../os/freertos/portable/heap_sel.c
"Compiling          : " ../../../../../os/freertos/timers.c
"Compiling          : " ../../../../../os/freertos/tasks.c
"Compiling          : " ../../../../../os/freertos/event_groups.c
"Compiling          : " ../../../../../os/freertos/list.c
"Compiling          : " ../../../../../os/freertos/croutine.c
"Compiling          : " ../../../../../os/freertos/queue.c
"Assembling         : " ../../../../../os/freertos/portable/Synopsys/ARC_EM/arc_support.s
"Archiving          : " obj_emsk_22/gnu_arcem7d/libfreertos.a
"Creating Directory : " obj_emsk_22/gnu_arcem7d/arc
"Compiling          : " ../../../../../arc/arc_timer.c
"Compiling          : " ../../../../../arc/arc_cache.c
"Compiling          : " ../../../../../arc/arc_exception.c
"Compiling          : " ../../../../../arc/arc_udma.c
"Assembling         : " ../../../../../arc/arc_startup.s
"Assembling         : " ../../../../../arc/arc_exc_asm.s
"Archiving          : " obj_emsk_22/gnu_arcem7d/libarc.a
"Creating Directory : " obj_emsk_22/gnu_arcem7d/library/clib
"Compiling          : " ../../../../../library/clib/fatfs_dirent.c
"Compiling          : " ../../../../../library/clib/malloc.c
"Compiling          : " ../../../../../library/clib/embARC_sbrk.c
"Compiling          : " ../../../../../library/clib/embARC_misc.c
"Compiling          : " ../../../../../library/clib/embARC_syscalls.c
"Compiling          : " ../../../../../library/clib/embARC_target.c
"Archiving          : " obj_emsk_22/gnu_arcem7d/liblibclib.a
"Archiving          : " obj_emsk_22/gnu_arcem7d/libembarc.a
"Linking            : " obj_emsk_22/gnu_arcem7d/freertos_iot_aws_smarthomedemo_gnu_arcem7d.elf
```

   c. Enter "***make TOOLCHAIN=gnu BD_VER=22 CUR_CORE=arcem7d run***" in command line to run the **smarthome_demo** program on EMSK. FreeRTOS-based runtime environment can be loaded automatically. Wait for WiFi initialization and connection establishment (30 seconds or less) until the "*WiFi connected*" message is shown in the terminal emulator. "*Network is ok*" will be shown after the certificate files **cert.crt**, **privateKey.pem** and **rootCA.crt** are validated.



d. Then the device shadow of **smarthome_demo** is initialized and connected with the AWS IoT cloud. The information in "*reported": {}* is the state of the EMSK-based smart home node. "*Updated Accepted !!*" means the connection works between the smart home node and AWS IoT cloud.

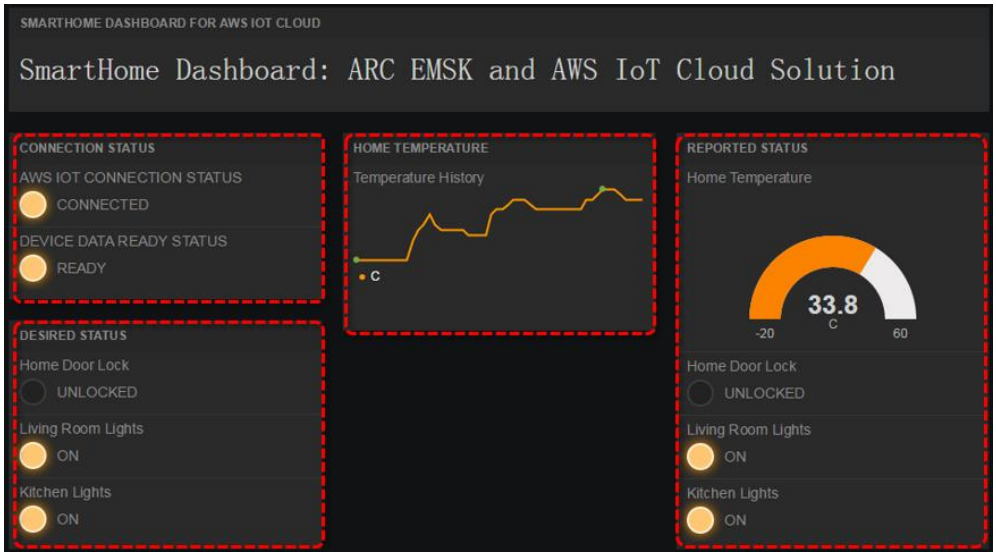


e. The dashboard contains four parts, **CONNECTION STATUS**, **DESIRED STATUS**, **HOME TEMPERATURE** and **REPORTED STATUS**.

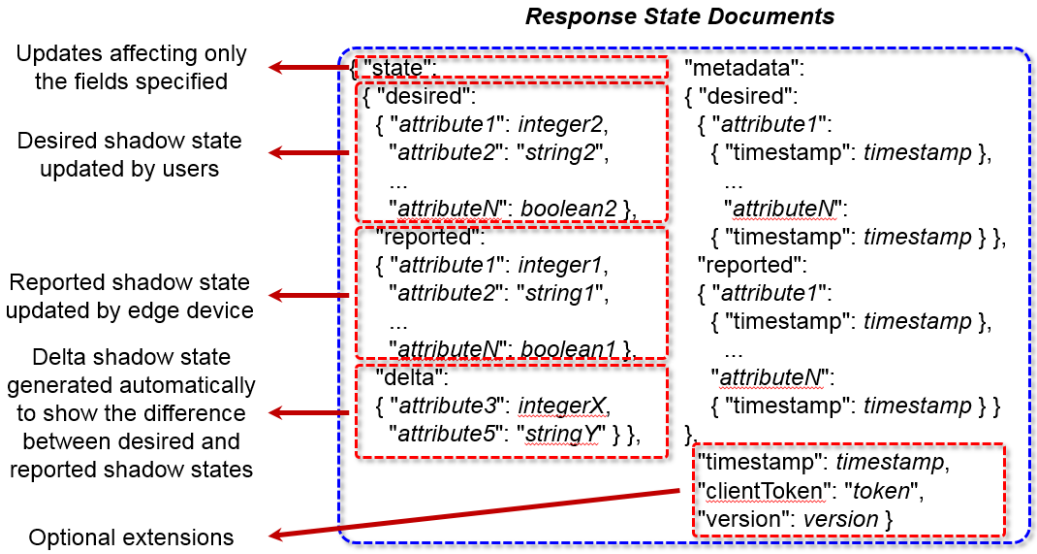
In **CONNECTION STATUS**, **AWS IOT CONNECTION STATUS** shows the connections status of dashboard and AWS IoT cloud. **DEVICE DATAREADY STATUS** shows the connection status of dashboard and smart home node.

DESIRED STATUS is used to send command (desired status) to AWS IoT cloud to control the smart home node. **REPORTED STATUS** is the feedback from smart home node. The temperature data is figured in **HOME TEMPERATURE**.

Click the buttons under **AWS IOT CONNECTION STATUS** and **DEVICE DATAREADY STATUS**. The **CONNECTED** and **READY** items mean the dashboard has established connection with AWS IoT cloud and smart home node. Then the smart home node can be controlled by the buttons of **Home Door Lock**, **Living Room Lights** and **Kitchen Lights** in **DESIRED STATUS**.



4) Go to AWS IoT console and click your smart home node "**SmartHome**". Check the detailed information in the right side. The JSON document in **Shadow state** and **Shadow metadata** is used to store the smart home node state information in the AWS IoT cloud.



NOTE:

1) The multiple node example in **embARC\example\freertos\iot\aws\smarthome_nodes** can be used in the same way. The only difference is that the multiple node example implements a model where multiple EMSKs each emulate one "thing".

2) NOTE: The secondary bootloader can be used to auto-load the bin file of **smarthome_demo** from SD card to memory at boot time and start the application. See application note **Using a secondary bootloader on the EMSK** https://www.embarc.org/help.html#notes for more information.

For any additional support on embARC, please post a question on embARC Forums at https://forums.embarc.org/