

# Thor96 Alexa User Manual

## iMX8M- THOR96 Alexa Demo

Date: March 10, 2022 | Version 2.0

---



The Solutions People



**Contents:**

1.	Document Details.....	4
1.1	Document History .....	4
1.2	Definition, Acronyms and Abbreviations .....	4
1.3	References .....	4
2.	Introduction .....	5
2.1	Purpose of the Document .....	5
2.2	Intended Audience .....	5
2.3	Overview .....	5
2.4	Pre-requisite for Demo.....	5
3.	Setup Steps for Alexa demo .....	6
3.1	Setup Steps: .....	6
3.2	Zigbee Web Application setup: .....	8
3.2.1	Login Page:.....	8
3.2.2	Scan and Pair Zigbee Devices .....	9
3.2.3	Zigbee Device Table: .....	9
4.	Run Alexa Demo:.....	11
5.	Execution Workflow .....	12
6.	Web Application Workflow .....	12
7.	Create ALEXA CUSTOM Skill .....	12
7.1	Log in.....	12
7.2	Creating Invocation .....	13
7.3	Creating intent: .....	13
7.4	Creating Utterances: .....	13
7.5	Creating Slot:.....	14
7.6	Endpoint:.....	14
8.	AWS-IOT .....	15
8.1	Create a thing:.....	15
9.	Basics of Alexa and AWS Setup: .....	17
9.1	Create a Lambda Function .....	17
9.2	Set Environment Variable:.....	19
10.	Alexa demo test sequence: .....	22
11.	Working logs: .....	25

## FIGURES

Figure 1: Overview of Demo .....	5
Figure 2: Web App Login Page .....	8
Figure 3: web App Scan page .....	9
Figure 4 : Scanning ZigBee devices.....	9
Figure 5: Scanned Devices.....	10
Figure 6: Paired Zigbee device table .....	10
Figure 7 : ZigBee Light Turn ON.....	11
Figure 8: Alexa custom skill setup .....	12
Figure 9: Create Invocation .....	13
Figure 10: Create Intent .....	13
Figure 11: Create Utterances .....	14
Figure 12: Create Slot.....	14
Figure 13: Slot Types .....	14
Figure 14: Create a Thing .....	15
Figure 15: Create Certificate .....	16
Figure 16: Create a Lambda Function at AWS.....	18
Figure 17: Lambda Functions at AWS.....	18
Figure 18: Add Alexa Skill Kit and Skill ID.....	19
Figure 19: Set Environment variable .....	19
Figure 20: Upload thor96.zip .....	20
Figure 21: Update lambda ID in Endpoint section.....	20
Figure 22: Lambda Cloud Watch .....	21
Figure 23: Voice commands from Simulator .....	23
Figure 24: Zigbee Device ON State.....	23
Figure 25: Zigbee Device OFF State.....	24

## Tables

Table 1 : Documents History .....	4
Table 2 : Description of Changes .....	4
Table 3 : Definition, Acronyms and Abbreviations.....	4
Table 4 : References .....	4

## 1. Document Details

### 1.1 Document History

Version	Author		Reviewer		Approver	
	Name	Date (DD-MM-YYYY)	Name	Date (DD-MM-YYYY)	Name	Date (DD-MM-YYYY)
0.1	Anil Patel	19-Feb-2019	Prajose John	22-Feb-2019	Bhavin Patel	22-Feb-2019
2.0	Deepak Rathore	08-Mar-2022	Snehal Parmar	09-Mar-2022	Snehal Parmar	09-Mar-2022

Table 1 : Documents History

Version	Description Of Changes
0.1	Initial draft
2.0	<ul style="list-style-type: none"> <li>Validated Alexa Demo on NXP Yocto Hardknot release</li> <li>Added steps for Alexa AWS account setup</li> <li>Added steps for Alexa Custom skill setup</li> <li>Added steps to setup Alexa Lambda function</li> </ul>

Table 2 : Description of Changes

### 1.2 Definition, Acronyms and Abbreviations

Definition/Acronym/Abbreviation	Description
cd	Change directory
SCP	Secure copy over the network
Wi-Fi	Wireless fidelity
App	Application
AWS	Amazon Web Service
IOT	Internet Of Things
ASK	Alexa Skills Kit

Table 3 : Definition, Acronyms and Abbreviations

### 1.3 References

No.	Document	Version	Remarks
1	ei_User_Guide_IMX8M_Thor96_L5_10_Rel_4_2.pdf	4.2	

Table 4 : References

## 2. Introduction

### 2.1 Purpose of the Document

Purpose of this document is to use/understand/flash/demonstrate Alexa demo on iMX8M-THOR96 firmware.

### 2.2 Intended Audience

This document is for developers and end-users who want to understand/flash/demonstrate interfaces on iMX8M-THOR96 firmware.

### 2.3 Overview

To control Household or Industrial ZigBee based devices through Voice (Alexa) and the board (THOR96).

For demo purpose, please use ZigBee based Light module.

In this demo, the objective is to control ZigBee based light which is connected to the THOR96 board through Alexa.

Below is the overview for our demo.



Figure 1: Overview of Demo

### 2.4 Pre-requisite for Demo

To run the Zigbee Demo, these are following prerequisites:

- Attach HDMI and Ethernet to board
- Board must have internet connectivity without any firewall. So that board can directly access and communicate with AMAZON server (ALEXA as well as AWS console)
- Zigbee Firmware (bootloader) must be loaded on Zigbee Chip
- Attach Mouse, Keyboard after board successfully boots up. (Currently we observe that when we attach USB devices and reboot board, Zigbee is not powered up successfully)
- ZigBee based light module <https://www.amazon.in/Directly-SmartThings-2-Outlet-Lighting-Appliance/dp/B07RYB3X5F>
- Alexa AWS account - For AWS setup, see the section [9. Basics of Alexa and AWS Setup](#) and for Alexa Custom skill, see the section [7.Create ALEXA CUSTOM Skill](#) of this document
- AWS Lambda function: [AWS](#)

### 3. Setup Steps for Alexa demo

#### 3.1 Setup Steps:

User needs to download firmware package and flash image in Thor96 board. Please follow the steps as mentioned in the [Section 2.3 and 2.4 of Thor96 User Guide](#) to flash firmware image in Thor96 board.

And once Thor96 board boots up with firmware image then do the following setup steps for Alexa demo:

- Update User and Group name in php configuration file

Update the following 2 changes in `/etc/php-fpm.d/www.conf.default` file and restart the php-fpm service.

```
- user = root
- group = root
```

```
$ systemctl restart php-fpm
```

- php-fpm service must be up and running. User can verify it using following command:

```
$ systemctl status php-fpm
```

- Zigbee service must be up and running. User can verify it using following command:

```
$ systemctl status zigbee
```



Note: If Zigbee service is not actively running, then please remove all USB devices and reboot board. (Soft reboot- through reboot command)

- apache2 service must be up and running. User can verify it using following command:

```
$ systemctl status apache2
```

- If it fails, then add certificate files at appropriate location. One can create self-signed certificate with below commands:

```
$ mkdir -p /var/apache2/logs
```

```
$ cd /etc/apache2
```

```
$ openssl req -newkey rsa:2048 -nodes -keyout server.key -x509 -days 365 -out server.crt
```

- Fill appropriate details and restart apache2 server

```
$ systemctl restart apache2
```

```
$ systemctl status apache2
```

- Reboot the board once all 3 services are running up and check again status of these services
- Now create Zigbee Network
- If by any case, ZigBee service is crashed, we must need to create new ZigBee network
- User can create Zigbee network by running this following URL:  
<https://localhost/htdocs/zigbee/startZigbeeNetwork.sh>
- Or User can directly run following script:

```
$ /usr/share/apache2/htdocs/zigbee/startZigbeeNetwork.sh
```

- Please note that if we create new Zigbee network, previous Zigbee network devices are lost, and no connection is possible with previous end devices
- All Zigbee logs are captured inside /var/log/syslog under tag **"Z3GatewayHost\_HMI"**

```
$ tail -f /var/log/syslog | grep Z3GatewayHost &
```

- Please verify your Board's (ZIGBEE) EUI64 with /usr/share/apache2/htdocs/zigbee/EUI64. If is there any mismatch, then correct it
- You can get your board's EUI64 by following line:

```
$ cat /var/log/syslog | grep Z3GatewayHost
```

```
Jan 17 06:41:50 imx8mqevk Z3GatewayHost_HMI: HA Gateweay EUI64 = 90FD9FFFFE1ECE9D
```

- User also need to subscribe MQTT topics for Alexa so that when we have any changes/delta in device states (at AWS), we will capture that delta and turn on/off lights based on that.
- Follow below steps for that:

```
$ cd ~/zigbee
```

```
$ tar-xzf aws-iot-device-sdk-python.tar.gz
```

```
$ cd ~/zigbee/aws-iot-device-sdk-python/
```

```
$ python setup.py install //for installing it first time
```

```
$ cd samples/basicShadow
```

```
$ python basicShadowDeltaListener.py -e a1aiafk5dxd68r-ats.iot.us-east-1.amazonaws.com -r ../root-CA.crt -c ../661540f04f-certificate.pem.crt -k ../661540f04f-private.pem.key -n EIC_ALEXA &
```

- This python script run in listening mode and check for updated device status on every sec
- When we ask Alexa to turn on/off ZigBee device, it will call this script with updated state
- We will listen those state and update our light according to that



Note: Before running the script (basicShadowDeltaListener.py), we need to set up the AWS account and Alexa developer custom skill.

For AWS setup, see the section [9. Basics of Alexa and AWS Setup](#) and for Alexa Custom skill, see the section [7.Create ALEXA CUSTOM Skill](#) of this document.

a1aiafk5dxd68r-ats.iot.us-east-1.amazonaws.com is custom skill id.

root-CA.crt, 661540f04f-certificate.pem.crt and 661540f04f-private.pem.key - these are the certificates of thing. Please follow steps in section [8.AWS IOT](#) to create certificates.

### 3.2 Zigbee Web Application setup:

- To run Web application on board, use chromium browser support in SD card image. User can run chromium browser through SSH or through serial console with following commands:

```
$ chromium-no-sandbox &
```

- It will start web browser on Connected HDMI display
- To open web application, type <https://localhost/ZigBeeWebApplication/index.html> in chromium browser. (Make sure to use secure connection "https" certified localhost if you use self-sign certificate otherwise scanning called got cancel and no scanning device found)

#### 3.2.1 Login Page:



Figure 2: Web App Login Page

- Enter following user credentials for login:

```
$ Username: admin
```

```
$ password: einfochips
```



### 3.2.2 Scan and Pair Zigbee Devices



Figure 3: web App Scan page

### 3.2.3 Zigbee Device Table:

- In this table, we will show the information about ZigBee devices like device name, device type, device EUI64, device node id etc.
- Now click on Scan Button. So, it will scan nearby ZigBee devices for almost 2 and half minutes



Figure 4 : Scanning ZigBee devices

- In order to pair ZigBee device, your device should be in reset mode and ready to pair mode
- Before one starts with scan ZigBee device, **“Scan Button”** is enabled by default and **“Pair Button”**, **“Delete Button”** and **“Help Button”** is disabled
- Click **“Scan Button”** to initiate scan process for ZigBee devices from server as mentioned in the Figure 4
- After successful scan, all scanned ZigBee devices details appears as shown in the following figure:

Industrial Automation				
Select	Device Name	Device ID	Device Type	Device ModelID
<input type="checkbox"/>	Zigbee_0	0x000D6F00053B03C9	ON-OFF Light	0x6532
<input type="checkbox"/>	Zigbee_1	0x000D6F0005B93D9	Color Dimmable Light	0x6531
<input type="checkbox"/>	Zigbee_2	0x000D6F0005B9310	Other devices	0x6531

Figure 5: Scanned Devices

- After successful scan, ‘Scan Button’ is disabled, and “Pair Button” and “Delete Button” is enabled
- Pair Button is useful for pairing ZigBee devices. User needs to decide on which devices is to be paired and should click on Pair Button
- But Before Pairing user must edit correct device name under “device name” section. For example, set device name “kitchen” for light
- Delete Button is useful to remove ZigBee devices from table
- Help Button is useful to get information of ZigBee web application
- After pairing, User can see current Device Status of paired ZigBee devices as shown in the following figure:  
EX. “LED- ON” or “LED- OFF”



Industrial Automation		
Device Name	Device ID	Device Status
Zigbee_0	0x000D6F00053B03C9	LED-OFF 
Zigbee_1	0x000D6F0005B93D9	LED-OFF 

Figure 6: Paired Zigbee device table

#### 4. Run Alexa Demo:

To Run Alexa Demo, User needs to setup Alexa AWS account and Alexa AWS account setup details can be found [here](#):

- After completion of setup, one can test Alexa Demo
- Pair at least one standard ZigBee light device or any other ZigBee device
- Zigbee devices must be paired, before running this demo
- Also, User need to subscribe MQTT topics for Alexa so that changes/delta in device states can be monitored when we turn on/off lights. (As describe in prerequisite)
- Open ALEXA APP in mobile or Alexa simulator online. Your Alexa account must have **“thor96 skill”** published by us
- After enabling skill, user can get response without any issue like:  
**“Alexa, how’s the weather today in Germany”**
- Here, our skill will not be in picture. But it will come into picture when user says  
**“Alexa, Open Thor automation demo”**
- Alexa will respond back –  
**“Welcome to the Thor Automation Demo”**
- Next the session gets created and one can send voice command to the Thor96 board
- Now speak **“turn on the kitchen light”**. (Instead of kitchen light, say whatever light name it is)
- If Alexa sends data successfully to device shadow, the success response shall be  
**“The kitchen light is turned on”**
- If above MQTT listener script (basicShadowDeltaListener.py) is running on board, it will directly call our ZigBee script and the status of the light shall change and Web App UI will be updated with LED ON status as shown following:

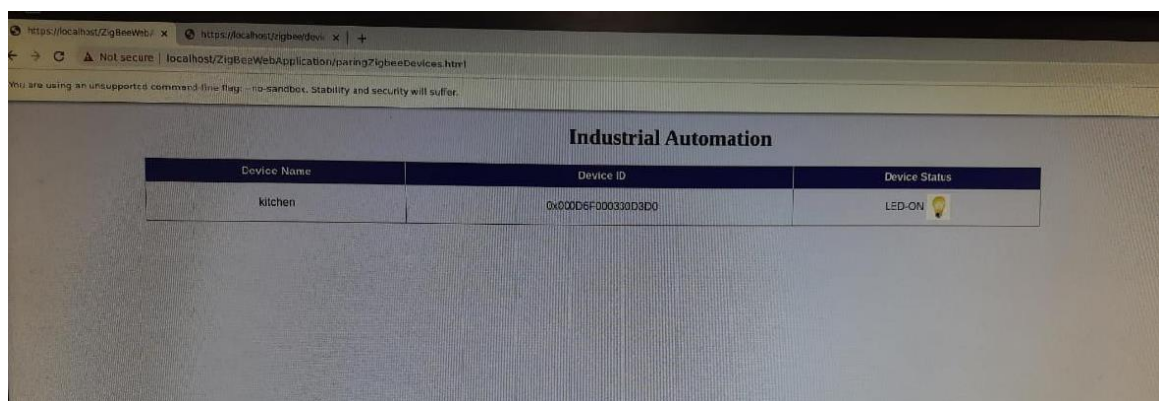


Figure 7 : ZigBee Light Turn ON

- This interactive Alexa session will continue until any error occurred or user says “BYE”
- If session end, user must need to create new session for further test

## 5. Execution Workflow

User → Alexa device/Alexa Simulator → Alexa Custom Skill → Identify Invocation (Thor automation demo) → AWS Lambda Function → REST API Calls → AWS IOT → AWS Shadow → Identify Delta in commands → THOR96 Device [Running Script, Custom ZigBee Binary] → Control ZigBee Lights [ZigBee-End-device ON/OFF]

- The User shall give a voice command to Alexa and Alexa shall respond to this command
- If the user speaks any “invocation” word, then our custom skill section is referred
- The custom skill in turn shall call the AWS lambda function
- AWS lambda function is backend code written in JavaScript and responsible to update the device shadow
- Device shadow maintains the current state of device
- In case of any update, MQTT message is published through AWS IOT platform. On device we subscribe MQTT topic and based on it the status of Zigbee lights is changed

## 6. Web Application Workflow

A web application is available to visualize the ZigBee operations and monitor status of Zigbee devices.

Start ZigBee Network on board → Open Web App on board through ssh or serial console → Insert login details → Scan for Zigbee Devices → Enter desired name for light → Select desired ZigBee Devices for status monitoring → Change status of Zigbee Devices → See updated state on Web app

## 7. Create ALEXA CUSTOM Skill

### 7.1 Log in

- Click on the below link and sign up [It is available for free to create account]:  
**[Amazon Alexa Console - Amazon Alexa Official Site](#)**
- Login to amazon Development Portal
- Navigate to Alexa-> your console -> skills
- Give your skill name like thor96 and keep it default selected options and create a skill. We can see our Skill name in below figure:

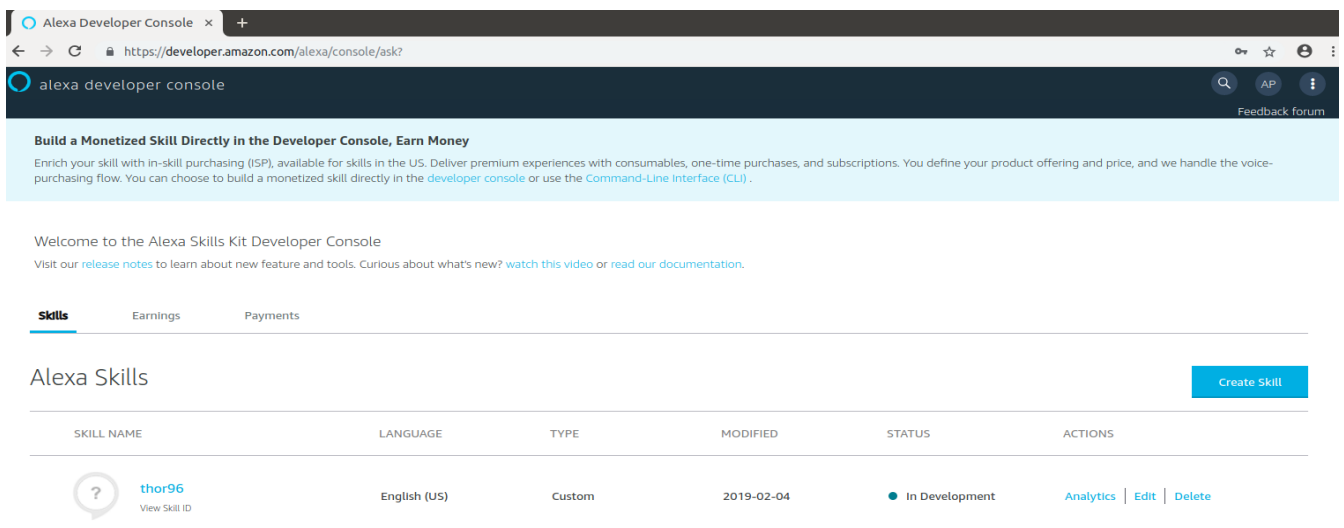


Figure 8: Alexa custom skill setup

## 7.2 Creating Invocation

- This is the command the user should provide in order to start the custom skill
- Users say a skill's invocation name to begin an interaction with a particular custom skill
- For e.g.: - to start one needs to say Alexa: open Thor Automation Demo, then invocation name is **“Thor automation demo”**
- Steps: Click on invocation and fill the name inside skill invocation name as mentioned in the Figure 8

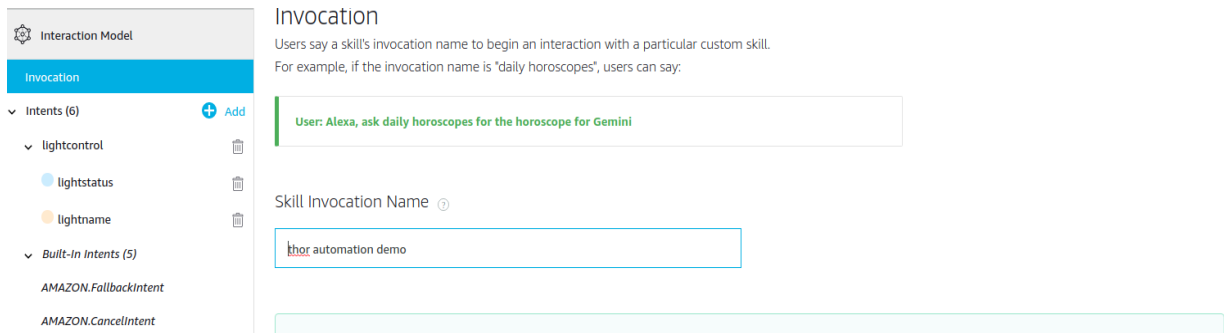


Figure 9: Create Invocation

## 7.3 Creating intent:

- Intents are the individual commands that is provided within skill to provide multiple skills
- To create intent, need to click on add button on page, in our case we made intent named as “lightcontrol”. Inside intent we train Alexa for multiple things such as control bedroom light, control kitchen light, etc.
- Steps: Click on add and put the name of intent [lightcontrol] as shown in the following figure:

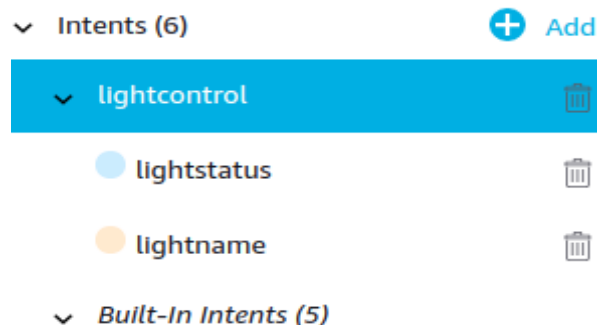


Figure 10: Create Intent

## 7.4 Creating Utterances:

- To create Utterances, provide the different flavors to control a device for similar kind of end device
- In our scenario, command: turn {lightstatus} the {lightname} light, light status and light name provide the flavors for different ways to say same thing

Sample Utterances (4) ? Bulk Edit Export

What might a user say to invoke this Intent? +

switch {lightstatus} {lightname} light	<span>🗑️</span>
turn {lightstatus} {lightname} light	<span>🗑️</span>
switch {lightstatus} the {lightname} light	<span>🗑️</span>
turn {lightstatus} the {lightname} light	<span>🗑️</span>

< 1 – 4 of 4 >

Figure 11: Create Utterances

## 7.5 Creating Slot:

- Slot is like variable which takes any value from given set of values.
- In our case, for utterances {lightstatus}, it has been taken as “on and off”; for {lightname} we have taken multiple values for example: kitchen, bedroom, etc.
- Slot types:
  - Lightstatustype
  - AMAZON.Room

Slot Types / lightstatustype

Slot Values (2) ? Bulk Edit Export Search

Enter a new value for this slot type +

VALUE <span>?</span>	ID (OPTIONAL) <span>?</span>	SYNONYMS (OPTIONAL) <span>?</span>	
off	Enter ID	Add synonym	<span>+</span> <span>🗑️</span>
on	Enter ID	Add synonym	<span>+</span> <span>🗑️</span>

Figure 12: Create Slot

Intent Launch Phrases

- Interaction Model
- Intents (6)
- Annotation Sets New
- Intent History
- Utterance Conflicts (0)
- JSON Editor
- Assets
- Slot Types (2)**
- Multimodal Responses
- Interfaces
- Endpoint

+ Add Slot Type Filter Slot Types 🔍

NAME	SLOT VALUES	SLOT TYPE	ACTIONS
AMAZON.Room	0	Built-in	<a href="#">Edit</a>   <a href="#">Delete</a>
lightstatustype	2	Custom with values	<a href="#">Edit</a>   <a href="#">Delete</a>

< 1 – 2 of 2 Slot Types >

Figure 13: Slot Types

## 7.6 Endpoint:

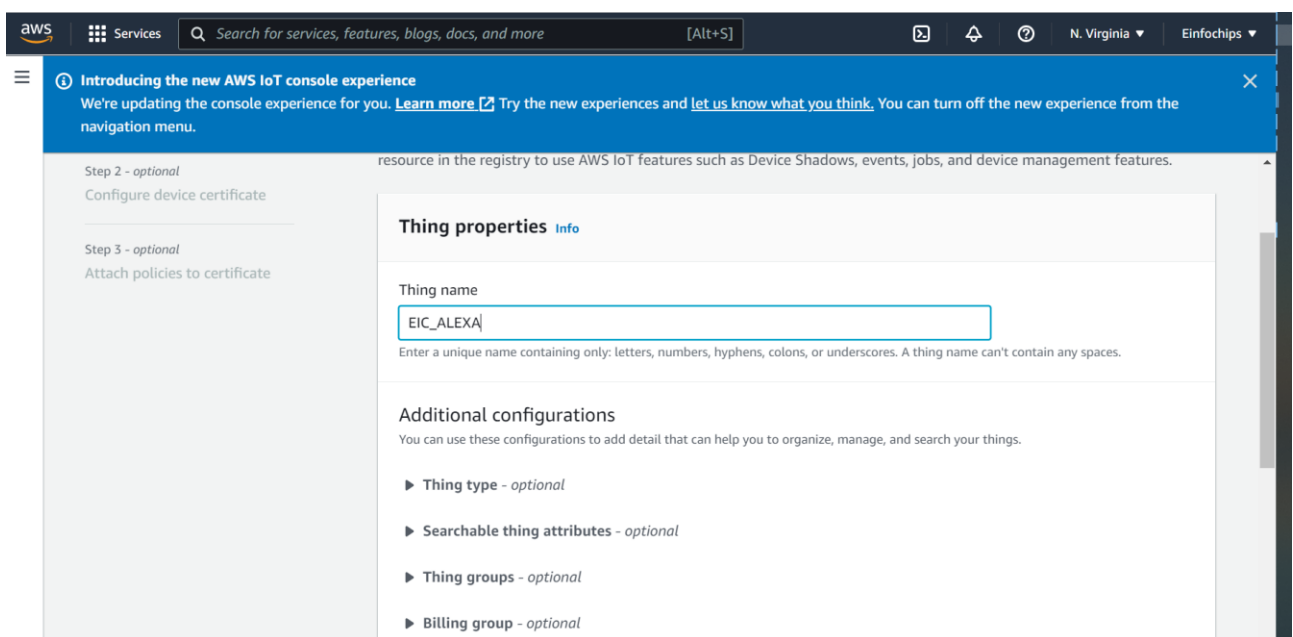
- Copy the lambda ARN id to Endpoint ->Default region - text box, save and build the skill

## 8. AWS-IOT

AWS IOT is a managed cloud platform that connects devices [Thor and Alexa] more easily and securely Here, connection is secured so it needs certificate to get connected to devices. Here AWS manages connection so we neither must provide any IP nor dependent on any given IP

### 8.1 Create a thing:

- Sign into AWS-IOT console <https://console.aws.amazon.com/iot/home?region=us-east-1#/home>
- AWS IOT: AWS IoT - Manage - Things (amazon.com)
- Give the login credential user id and password
- After that search for AWS IOT in search box and find the AWS IOT
- Go to the Manage tab and click on the things section
- Things -> create a thing -> single Thing
- Click on the next button and give the thing name



*Figure 14: Create a Thing*

- Register a thing [for ex.: EIC\_ALEX]
- Create a certificate for your thing [Refer: Figure 15]
- Download Self-generated Licenses
- Activate the License
- Attach a policy [provided name of policy, and fill Action section : iot:\*, and Resource ARN:\*)]
- Now attach Policy to license

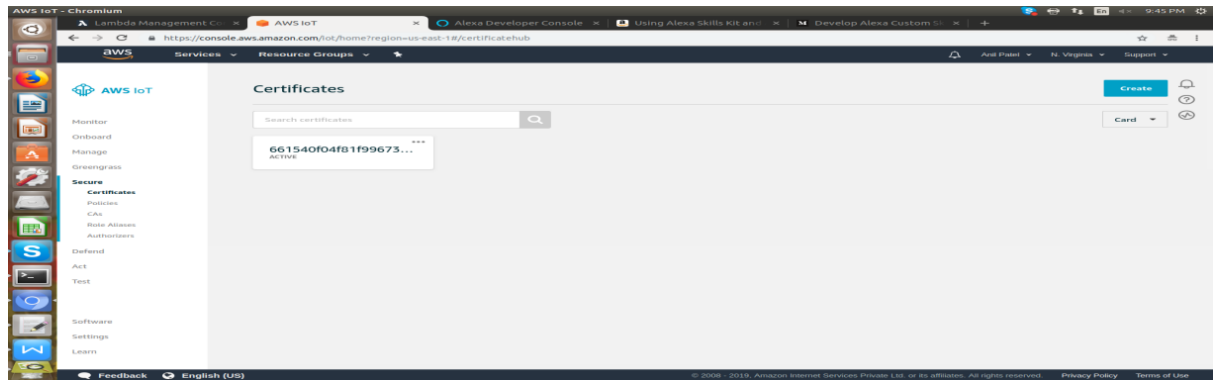


Figure 15: Create Certificate

Self-Generated license is used with Thor96 board:

- root-CA.crt
- 661540f04f-certificate.pem.crt
- 661540f04f-private.pem.key



**Note:** When user creating the thing, user need to download the certificates [ \*.crt, \*.pem.crt and \*.pem.key files] and copy to the board in below path. We have used same thing name and certificates name in this document.

Copy certificates and key files in this path in Thor96 board: **/home/root/zigbee/aws-iot-device-sdk-python/samples**

- It provides a bridge between thor96 and AWS Lambda, with multiple level of security and secured protocol such as MQTT to make all connectivity even stronger
- Following snippet is delta of JSON which gets updated from Lambda and eventually updated on device

```
"delta": {
  "lightstatus": "on",
  "lightname": "kitchen"
}
```



## 9. Basics of Alexa and AWS Setup:

- The brain behind Echo and other Amazon voice-enabled devices like Echo Show, Echo Dot, Echo Spot, and Amazon Tap is **“Alexa - the cloud-based service”** that handles all the speech recognition, machine learning and Natural Language understanding for all Alexa enabled devices
- Alexa provides a set of built-in capabilities, referred to as skills, that define how one can interact with the device.  
For example, Alexa’s built-in skills include playing music, reading the news, getting a weather forecast, and querying Wikipedia.
- So, one could say things like:  
**“Alexa, ask daily horoscopes for the horoscope for Gemini”**
- In addition to these built-in skills, we can program custom skills by using the Alexa Skills Kit (ASK). An Alexa user can then access these new abilities by asking Alexa questions or making requests
- All skills, like web or mobile applications, contain two parts: Interaction Model (the frontend) and the Hosted Service (the backend)
- Interaction Model (frontend) —Voice User Interface (VUI) - It defines what functionalities or behaviors the skill can handle. For example, our light skill for turn on/off lights
- Hosted Service (backend) — The programming logic that responds to user's requests. In our case, AWS LAMBDA function

Let’s understand few terminologies to understand skills:

When User says – **“Alexa, ask daily horoscopes for the horoscope for Gemini”**

Here Alexa is considered the wake word. Now Alexa starts responding as following:

“Daily horoscopes” is SKILL INVOCATION NAME. So here skill which is enabled and whose’ invocation name is “daily horoscopes” will be activated and rest of the words (the horoscope for Gemini) are sent for performing action.

### 9.1 Create a Lambda Function

Lambda function is backend programming logic, which is responsible to update device state on device shadow.

- To develop Lambda Function, it is required to be logged into the AWS Console and go to lambda [It is free to create account on AWS]
- Click on the below link and create a function  
<https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions>
- Go to: Create Function -> Author from Scratch [Author from scratch] -> Write function Name, select run-time, and go to change default execution role section and select existing role
- i.e., lambda\_iot\_execution and click on Create Function

**Basic information**

**Function name**  
Enter a name that describes the purpose of your function.  
thor96\_us  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** info  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.  
Node.js 14.x

**Architecture** info  
Choose the instruction set architecture you want for your function code.  
☒ x86\_64  
☐ arm64

**Permissions** info  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

**Change default execution role**  
**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.  
☐ Create a new role with basic Lambda permissions  
☒ Use an existing role  
☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
lambda\_iam\_execution  
View the lambda\_iam\_execution role on the IAM console.

**Advanced settings**

Cancel Create function

Figure 16: Create a Lambda Function at AWS

- After creating the lambda function, we could see our lambda function in lambda -> function section as below:

**Functions** (1)

Filter by tags and attributes or search by keyword

Function name	Description	Runtime	Code size	Last modified
thor96_us		Node.js 8.10	1.2 kB	57 minutes ago

Feedback English (US) © 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Figure 17: Lambda Functions at AWS

- Now we must add the Alexa Skill Kit, for this click the + Add trigger and search for Alexa Skill Kit and select it
- Add our custom skill id in skill ID test box like below

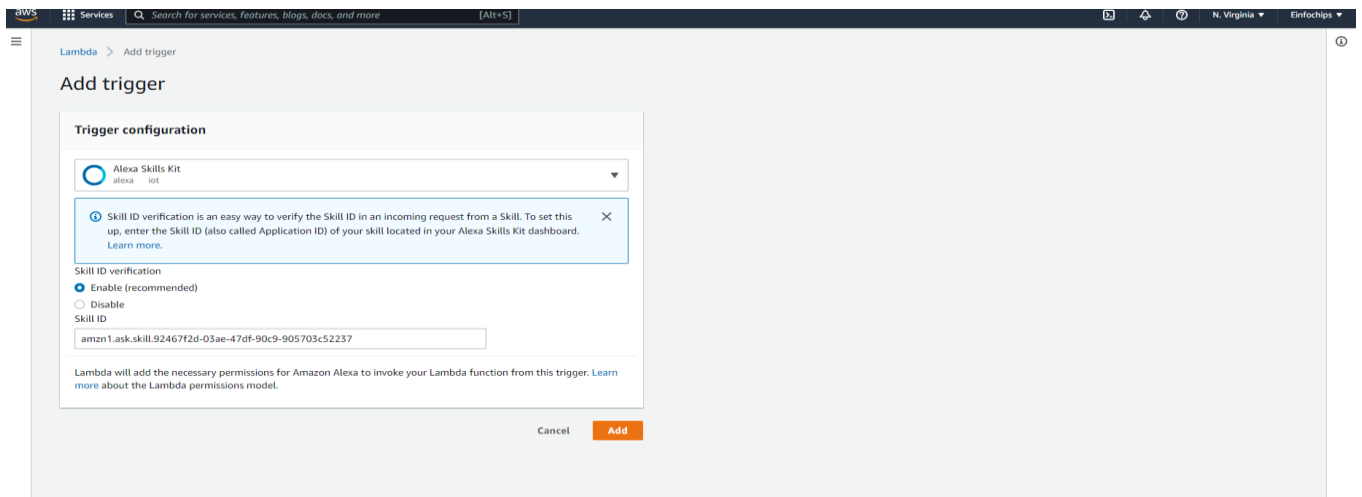


Figure 18: Add Alexa Skill Kit and Skill ID

## 9.2 Set Environment Variable:

- In Configuration tab, select the Environment variable section, add the env variables key and value pair
- See the following Figure 19:

```
AWS_IOT_MQTT_HOST="ec2-3-89-121-147.compute-1.amazonaws.com"
AWS_IOT_MQTT_PORT=8883
AWS_IOT_MQTT_PORT_UPDATE=8443
AWS_IOT_MY_THING_NAME="EIC_ALEXA"
AWS_ALEXA_SKILLS_KIT_ID="amzn1.ask.skill.92467f2d-03ae-47df-90c9-905703c52237"
```

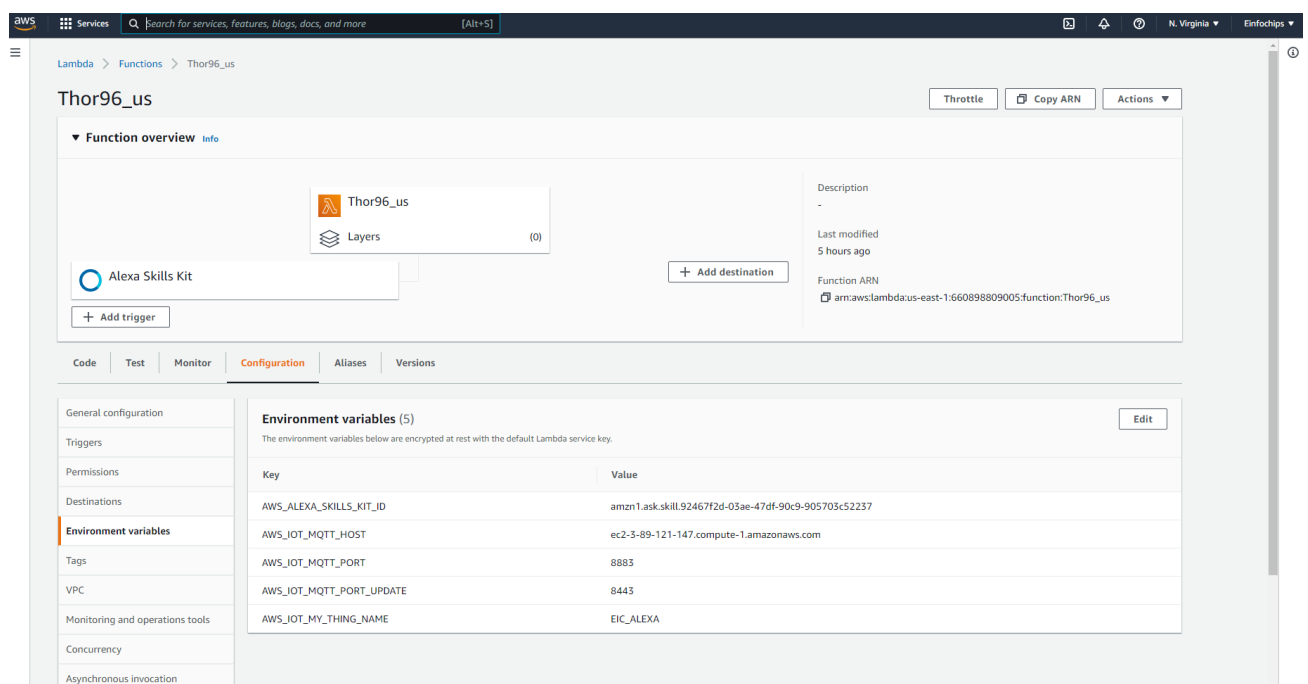


Figure 19: Set Environment variable

- After that upload the *thor96\_AWS\_lambda\_function.zip* file in lambda function

The screenshot shows the AWS Lambda console with the code editor open for the `lambda_function`. The code is a Python handler for an Alexa skill, using the `event_actions` module to handle requests. The code includes comments and docstrings explaining the function's purpose and arguments.

```

1  """
2  lambda_handler
3  """
4  import os
5
6  import event_actions
7
8  alexa_id = os.environ.get('AWS_ALEXA_SKILLS_KIT_ID')
9
10 def lambda_handler(event, context):
11     """
12     Handles event and request from Alexa Skill by using methods form
13     the event_actions module.
14
15     Args:
16         event: Python dict of event, request, and session data
17         context: LambdaContext containing runtime data
18     Returns:
19         Python dict of response message
20     Raises:
21         ValueError
22     """
23     print("event.session.application.applicationId=" +
24           event['session']['application']['applicationId'])
25
26     # Ensure that request is from our skill
27     ...
28
29     if event['session']['application']['applicationId'] !=
30       'amzn1.ask.skill.0'.format(alexa_id):
31         print("amzn1.ask.skill.0".format(alexa_id))
32         raise ValueError("Invalid Application ID")
33     ...
34
35     # Uncomment if storing information in sessions
36     if event['session']['new']:
37         event_actions.on_session_started({'requestId': event['request']['requestId'],
38                                         event['session']})
39
40     request_type = event['request']['type']
41
42     if request_type == 'LaunchRequest':

```

- Copy the function ARN id and paste in the custom skill Endpoint section -> Default Region textbox

Endpoint

MODELS

TOOLS

### Service Endpoint Type

Select how you will host your skill's service endpoint. Best practices in choosing lambda regions. [Learn more.](#)

● AWS Lambda ARN  
(Recommended)

Your Skill ID ⓘ  
arn:aws:lambda:**us-east-1**:660898809005:function:Thor96\_us

COPY Copy to Clipboard

Default Region  
(Required)

arn:aws:lambda:**us-east-1**:660898809005:function:Thor96\_us

North America  
(Optional)

arn:aws:lambda:**us-east-1**:<aws\_account\_id>;function:<lambda\_name>

Europe and India  
(Optional)

arn:aws:lambda:**eu-west-1**:<aws\_account\_id>;function:<lambda\_name>

Far East  
(Optional)

arn:aws:lambda:**ap-northeast-1**:<aws\_account\_id>;function:<lambda\_name>

- We can monitor the Lambda functionality for error(S) or server availability every moment when the Lambda has multiple loads when accessing multiple devices

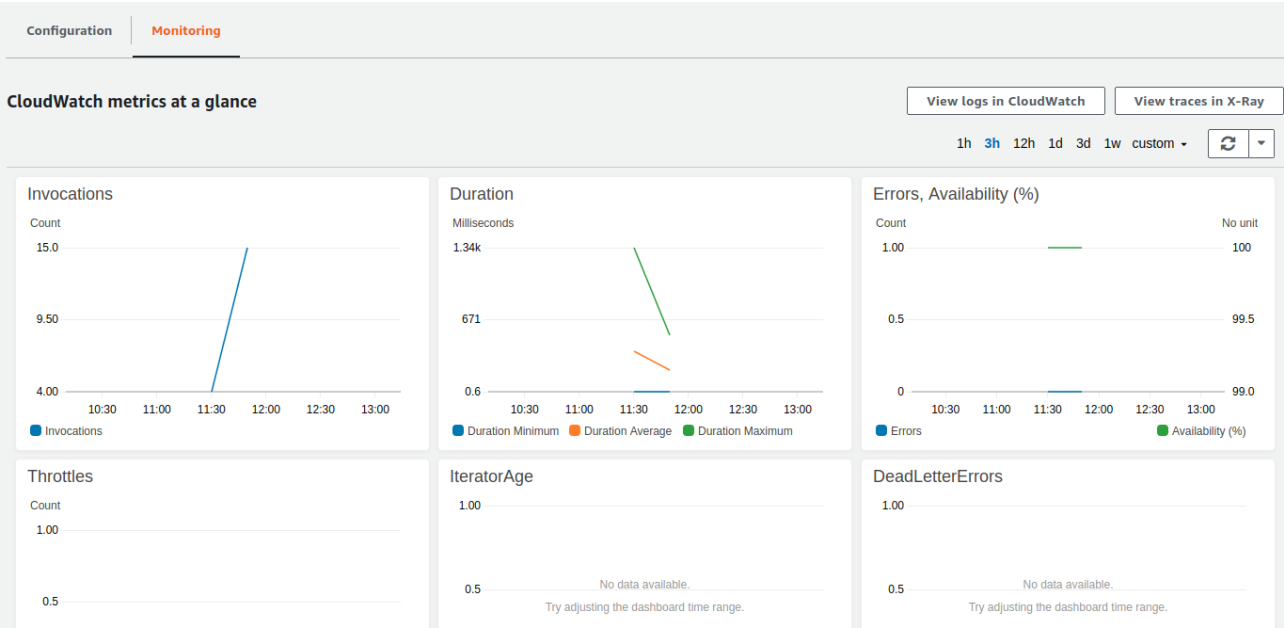


Figure 22: Lambda Cloud Watch

## 10. Alexa demo test sequence:

- Power on the thor96 board. Once board is powered up then check below services:

```
$ systemctl status zigbee
$ systemctl status apache2
$ systemctl status php-fpm
```

- Now create Zigbee Network

```
$ /usr/share/apache2/htdocs/zigbee/startZigbeeNetwork.sh
```

- All Zigbee logs are captured inside /var/log/syslog under tag "Z3GatewayHost\_HMI"

```
$ tail -f /var/log/syslog | grep -i Z3GatewayHost &
```

- Please verify your Board's (ZIGBEE) EUI64 with /usr/share/apache2/htdocs/zigbee/EUI64. If there is any mismatch, then correct it

```
$ cd ~/zigbee
$ tar -xzf aws-iot-device-sdk-python.tar.gz
$ cd ~/zigbee/aws-iot-device-sdk-python/
$ python setup.py install //for installing it first time
```

- Run the chromium

```
$ chromium-no-sandbox &
```

- To open web application, type <https://localhost/ZigBeeWebApplication/index.html> in chromium browser

```
$ https://localhost/ZigBeeWebApplication/index.html
```

- Enter user credentials for login

```
$ Username: admin
$ password: einfochips
```

- Run the AWS service instance, copy the instance and add to the lambda function environment variables and same to board as discussed above  
ex: ec2-3-89-121-147.compute-1.amazonaws.com
- we need to copy the certificates to board using scp command as discussed above in the section 8

```
$ root@imx8mqthor96: ~/zigbee/aws-iot-device-sdk-python/samples/
```

- Run the script from the board in this command we need to provide the Thing Endpoint id: az10lm4ep3yp4-ats.iot.us-east-1.amazonaws.com

```
$root@imx8mqthor96:~/cd zigbee/aws-iot-device-sdk-python/samples/basicShadow
$ root@imx8mqthor96:~/zigbee/aws-iot-device-sdk-python/samples/basicShadow/
```

```
python3 basicShadowDeltaListener.py--endpoint az10lm4ep3yp4-ats.iot.us-east-1.amazonaws.com-
r ../AmazonRootCA1.pem-
c ../54101ba00084229ed132cfa0e5bd947136e72a8a0b34da78726c9b875206631c-certificate.pem.crt-
k ../54101ba00084229ed132cfa0e5bd947136e72a8a0b34da78726c9b875206631c-private.pem.key-n
EIC_ALEXA
```

- Give the following voice command through Alexa app / Simulator:  
**“Open thor automation demo and switch on the kitchen light”**

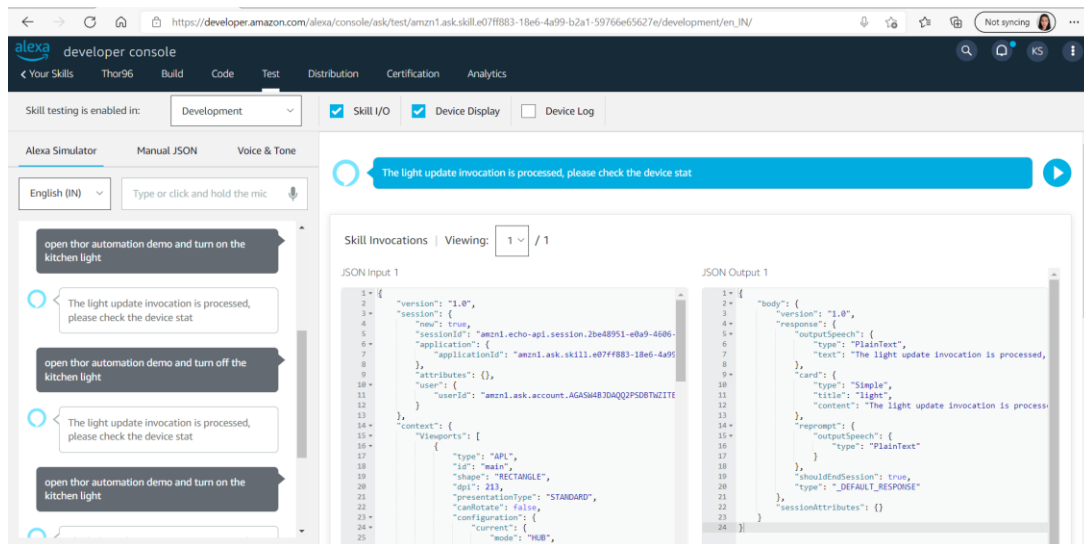


Figure 23: Voice commands from Simulator

- Here we have used Jasco Zigbee Plug-in Smart On/Off Switch device
- When we give the “open thor automation demo and turn on the kitchen light” voice command from the Simulator, ZigBee device is will be turned ON which we can see in the below figure:



Figure 24: Zigbee Device ON State

- And when you give the “open thor automation demo and turn off the kitchen light” voice command in Simulator then ZigBee device will turn off which we can see in the below figure:



*Figure 25: Zigbee Device OFF State*



## 11. Working logs:

```
$ root@imx8mqthor96:~/zigbee/aws-iot-device-sdk-python/samples/basicShadow#
python basicShadowDeltaListener.py --endpoint aoaxcgzjhl8o-ats.iot.us-east-1.amazonaws.com --rootCA root-ca-
cert.pem --cert 2ab1250063-certificate.pem.crt --key 2ab1250063-private.pem.key -n EIC_ALEXA
2022-02-01 09:47:02,797 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Initializing MQTT layer...
2022-02-01 09:47:02,799 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Registering internal event
callbacks to MQTT layer...
2022-02-01 09:47:02,800 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - MqttCore initialized

2022-02-01 09:47:02,800 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Client id: basicShadowDeltaListener
2022-02-01 09:47:02,801 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Protocol version: MQTTv3.1.1
2022-02-01 09:47:02,801 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Authentication type: TLSv1.2
certificate based Mutual Auth.
2022-02-01 09:47:02,801 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring offline requests
queueing: max queue size: 0
2022-02-01 09:47:02,802 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring offline requests queue
draining interval: 0.100000 sec
2022-02-01 09:47:02,803 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring endpoint...
2022-02-01 09:47:02,804 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring certificates...
2022-02-01 09:47:02,804 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring reconnect back off
timing...
2022-02-01 09:47:02,805 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Base quiet time: 1.000000 sec
2022-02-01 09:47:02,805 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Max quiet time: 32.000000 sec
2022-02-01 09:47:02,805 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Stable connection time: 20.000000
sec
2022-02-01 09:47:02,806 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring connect/disconnect
time out: 10.000000 sec
2022-02-01 09:47:02,806 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Configuring MQTT operation time
out: 5.000000 sec
2022-02-01 09:47:02,807 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync connect...
2022-02-01 09:47:02,807 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing async connect...
2022-02-01 09:47:02,808 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Keep-alive: 600.000000 sec
2022-02-01 09:47:02,810 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Event consuming thread
started
2022-02-01 09:47:02,810 - AWSIoTPythonSDK.core.protocol.mqtt_core - DEBUG - Passing in general notification
callbacks to internal client...
2022-02-01 09:47:02,812 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Filling in fixed event
callbacks: CONNACK, DISCONNECT, MESSAGE
```

```

2022-02-01 09:47:05,790- AWSIoTPythonSDK.core.protocol.internal.clients- DEBUG- Starting network I/O thread...
2022-02-01 09:47:06,226- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- Produced [connack] event
2022-02-01 09:47:06,228 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [connack]
event
2022-02-01 09:47:06,229- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- No need for recovery
2022-02-01 09:47:06,229 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event
callback...
2022-02-01 09:47:06,263- AWSIoTPythonSDK.core.protocol.mqtt_core- INFO- Performing sync subscribe...
2022-02-01 09:47:06,263- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- Adding a new subscription
record: $aws/things/EIC_ALEXA/shadow/update/delta qos: 0
2022-02-01 09:47:06,265 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Filling in custom suback
event callback...
2022-02-01 09:47:06,636- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- Produced [suback] event
2022-02-01 09:47:06,637- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- Dispatching [suback] event
2022-02-01 09:47:06,638 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event
callback...
2022-02-01 09:47:06,639- AWSIoTPythonSDK.core.protocol.internal.clients- DEBUG- This custom event callback is
for pub/sub/unsub, removing it after invocation...
2022-02-01 09:47:08,687 - AWSIoTPythonSDK.core.shadow.deviceShadow - INFO- Subscribed to delta topic for
deviceShadow: EIC_ALEXA
2022-02-01 09:47:13,074- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- Produced [message] event
2022-02-01 09:47:13,075 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message]
event
delta/EIC_ALEXA
+++++++DELTA+++++++
2022-02-01 09:47:13,082 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event
callback...
Content-type: text/plain; charset=iso-8859-1

Turn On Zigbee Device...

result = 0
lightstatus: ON
lightname: KITCHEN
+++++++
2022-02-01 09:47:26,229 - AWSIoTPythonSDK.core.protocol.connection.cores - DEBUG - stableConnection:
Resetting the backoff time to: 1 sec.

```

```
2022-02-01 09:47:30,821- AWSIoTPythonSDK.core.protocol.internal.workers- DEBUG- Produced [message] event
2022-02-01 09:47:30,823 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message]
event
delta/EIC_ALEXA
+++++++DELTA+++++++
2022-02-01 09:47:30,831 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event
callback...
Content-type: text/plain; charset=iso-8859-1

Turn Off Zigbee Device...

result = 0
lightstatus: OFF
lightname: KITCHEN
+++++++
```