iMX8M- THOR96 Alexa Demo

Version	1.0
Status	Baseline
Date	22-FEB-2019

Contents

FIGURES

Tables

Document Details

➢ 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

Table 1 : Documents History

Version	Description Of Changes
0.1	*initial draft*

Table 2 : Description of Changes

➢ 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

➢ References

No.	Document	Version	Remarks
1	Refer the User guide V0.3	0.3	

Table 4 : References

Alexa Demo

1. Objective:

To control Household or Industrial ZigBee based devices through Voice (Alexa) and the board (THOR96). For demo purpose, please use ZigBee Light.

2. Overview:

In this demo, the objective is to control ZigBee light connected to the THOR96 board through Alexa. Below is the overview for our demo.

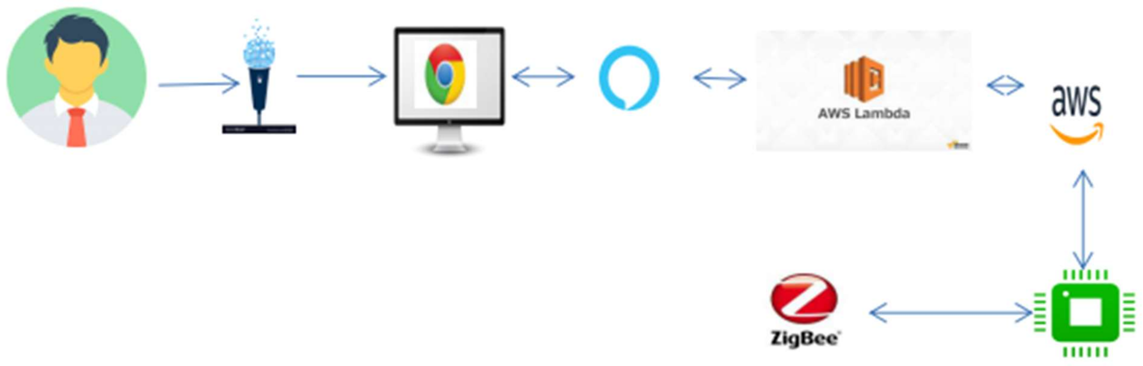


Figure 1: Overview of Demo

Project Demo

3. Prerequisite for Demo

To run the Zigbee Demo - there are a few 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. (Please refer Release User guide for this).
- 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 service must be up and running
- User can verify by Systemctl command

  # systemctl status zigbee

  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

```
# systemctl status apache2
```

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

```
# 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
```

- 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 as mentioned in following URL (on board):

  https://localhost/cgi-bin/zigbee/startZigbeeNetwork.sh

  or

  User can directly run script:

```
# /usr/share/apache2/cgi-bin/zigbee/startZigbeeNetwork.sh
```

- Please note that if we create new Zigbee network, previous Zigbee network devices is 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 –i  Z3GatewayHost &
```

- Please verify your Board's (ZIGBEE) EUI64 with /usr/share/apache2/cgi-bin/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 -i Z3GatewayHost
# Jan 17 06:41:50 imx8mqevk Z3GatewayHost_HMI: HA Gateweay EUI64 = 90FD9FFFFE1ECE9D
```

- Also, User 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.
  For that we need to follow below steps:

```
# 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 ~/zigbee/aws-iot-device-sdk-python/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. So 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.

4. Zigbee Web Application:

Web app help for user friendly scan and to pair ZigBee end devices with THOR96 board, monitor ZigBee devices.

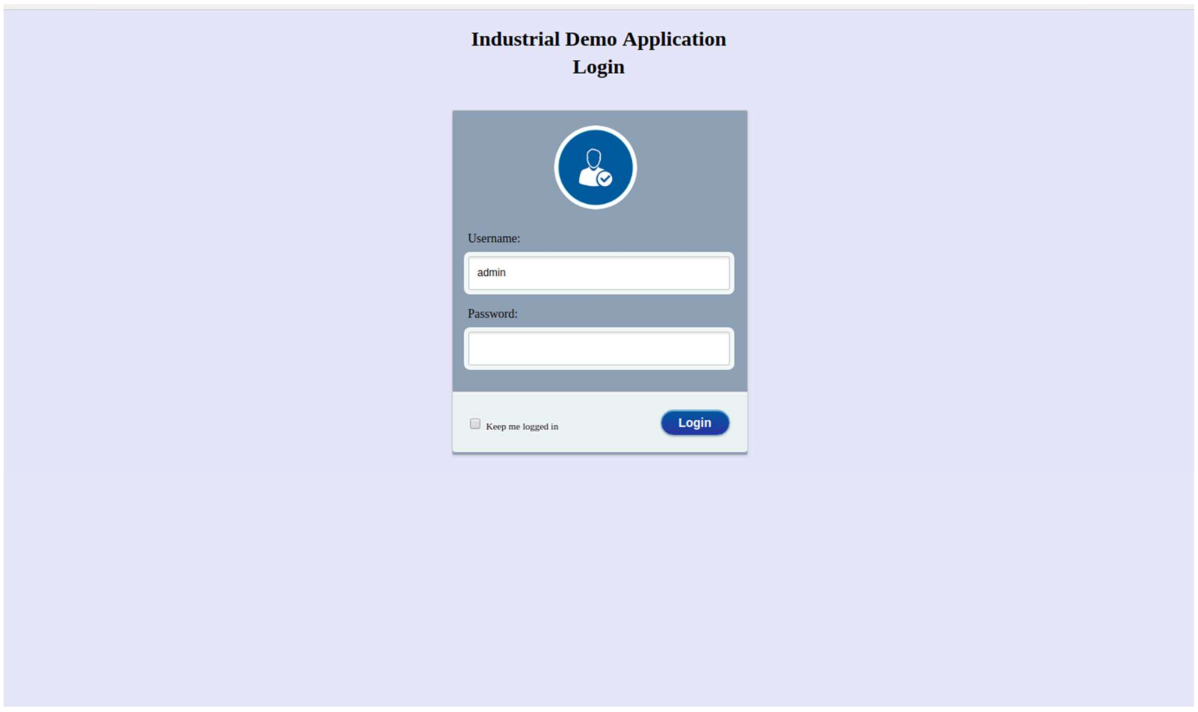
To run Web application on board use chromium browser support in sdcard image. User can run chromium browser through SSH or through serial with following commands:

# 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" and certified localhost if you use self-sign certificate otherwise scanning called got cancel and no scanning device found)

Login Page:


Industrial Demo Application
Login

Username:
admin
Password:

Figure 2: Web App Login Page

Username & Password:

enter user credentials for login.

Username: admin

Password: einfochips

Scan and Pair Zigbee Devices :

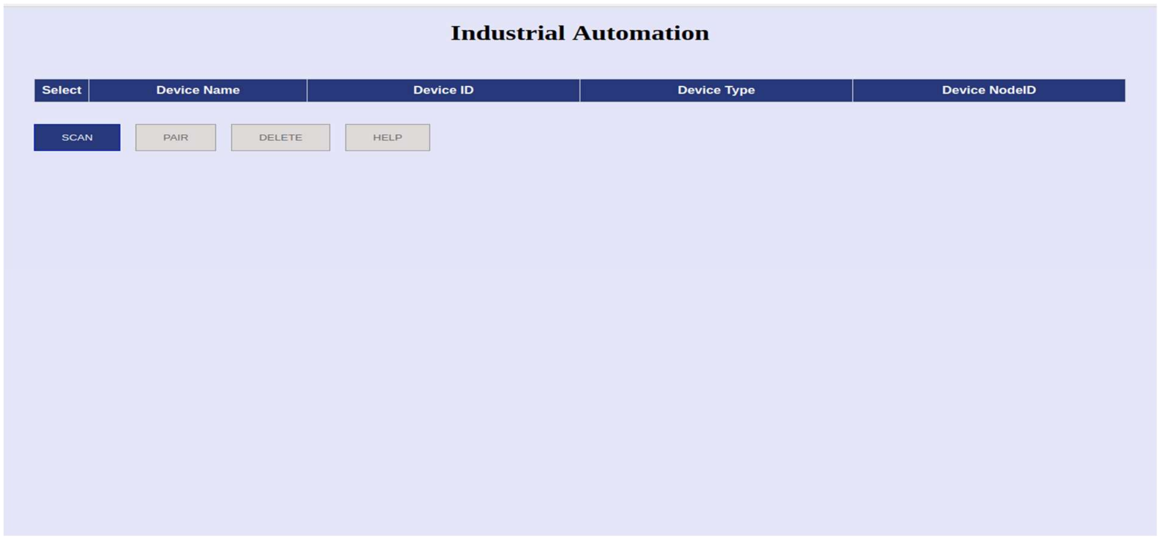


Figure 3: Web App Scan Page

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.

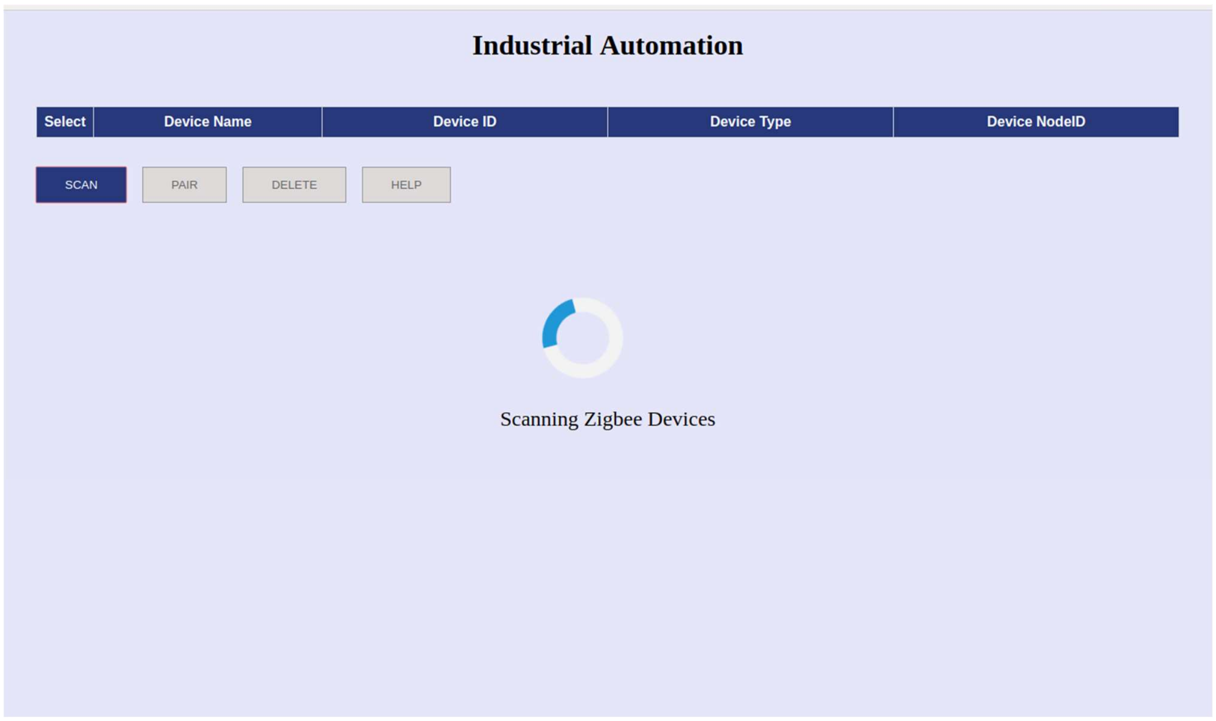


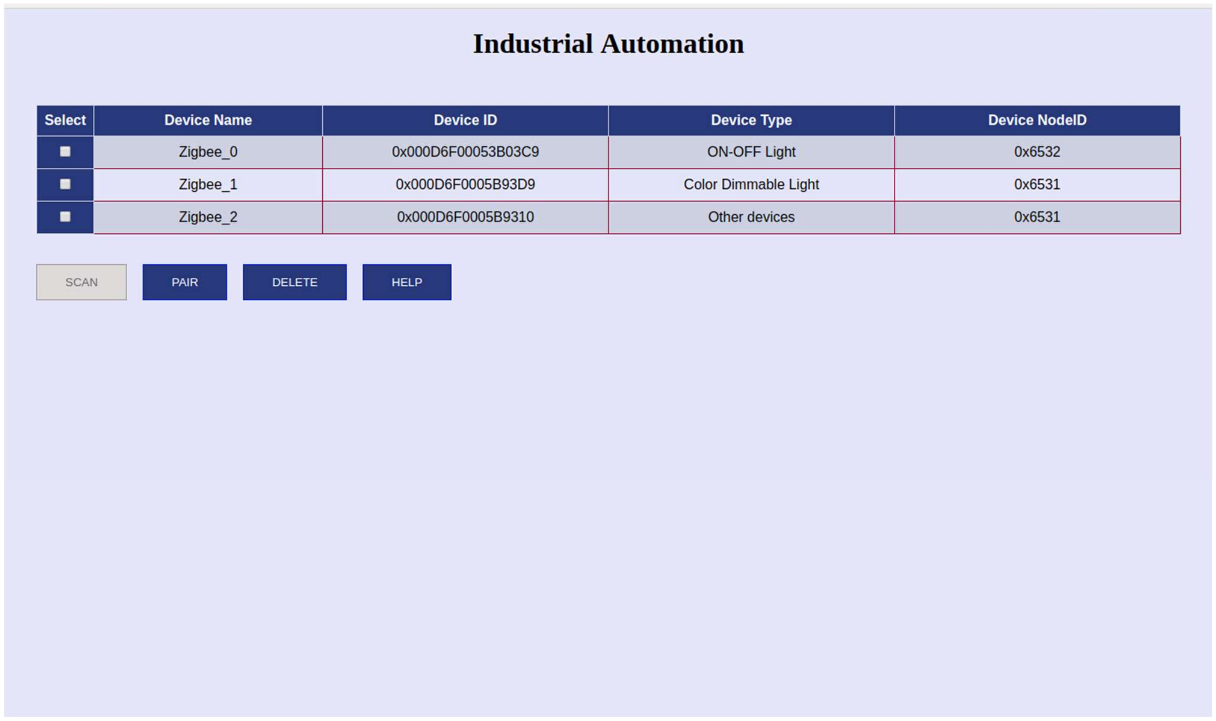
Figure 4: Scanning Nearby Zigbee Devices

In this type 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 described in below image.

After successful scan, all scanned ZigBee devices details appears as shown in table.



Select	Device Name	Device ID	Device Type	Device NodeID
☐	Zigbee_0	0x000D6F00053B03C9	ON-OFF Light	0x6532
☐	Zigbee_1	0x000D6F0005B93D9	Color Dimmable Light	0x6531
☐	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 need 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 "bedroom" 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.

EX. **"LED - ON"** or **"LED - OFF"**

5. Run Alexa Demo:

After completion of setup, one can test Alexa Demo.

Pair at least one standard ZigBee light and any other ZigBee device. We use door / windows sensor (which doesn't accept any output command)

Both 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 "**lightskill**" published by us.

For Testing purpose, anyone can use my Alexa account:

Username : anil.patel@einfochips.com

Password: Einfochips@123

By default, "**lightskill**" is enable in this account.

After enabling skill, user can get daily 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 Thor board.

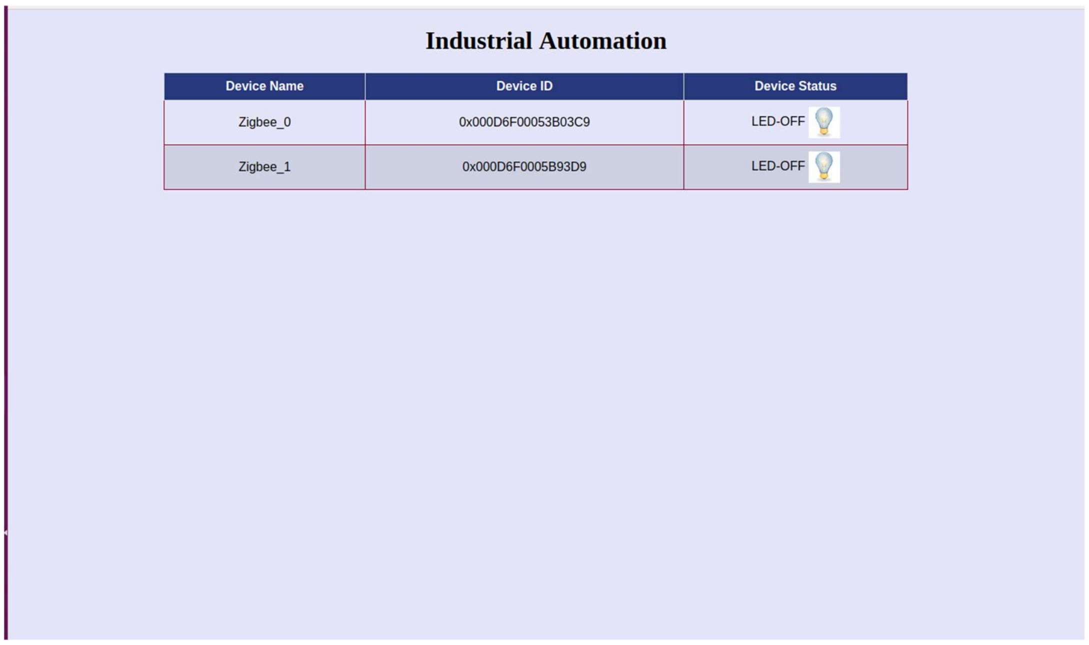
Now speak "**turn on the bedroom light**". (instead of bedroom whatever light name it is)

If Alexa sends data successfully to device shadow, the success response shall be "**The bedroom light is turned on**".

If above MQTT listener script is running on board, it will directly call our ZigBee script and the status of the light shall change.

This interactive Alexa session will continue until **any error occurred** or user says "**BYE**".

If session end, use must need to create new session for further test.



Industrial Automation

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

Figure 6: Paired Zigbee device table

Appendix:

(Following all sections are just for information on how our demo works.)

6. Execution Work Flow

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 the our custom **skill** section is referred.
- The custom skill in turn shall call the **AWS lambda** function.
- AWS lambda function is the 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 are changed.

Web Application Work Flow

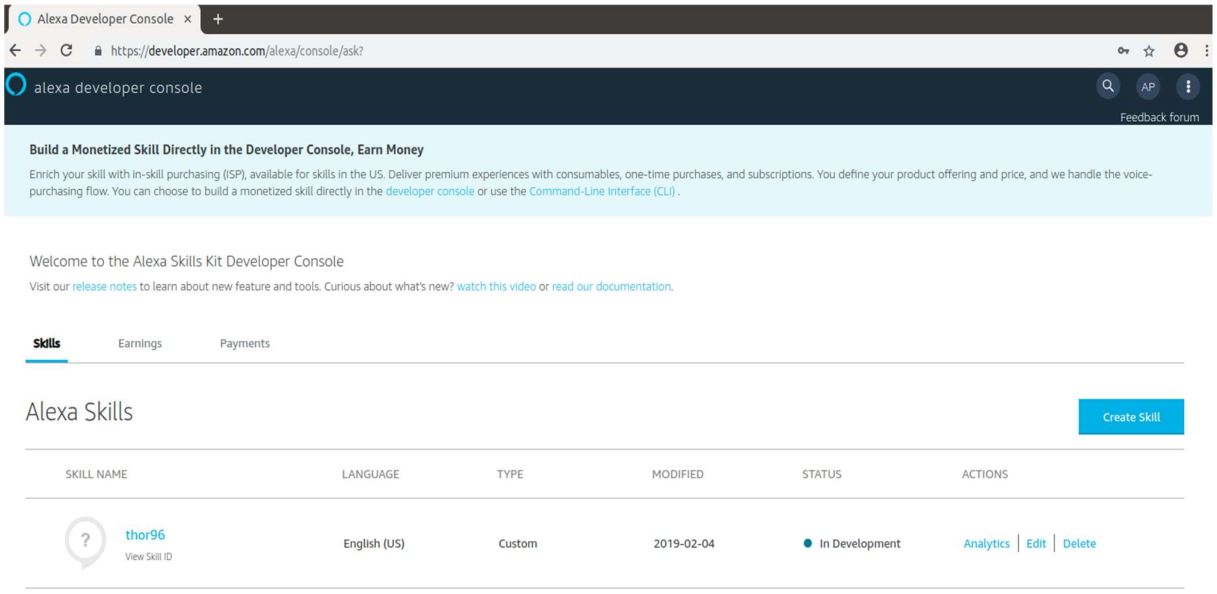
A web application is available to visualize the ZigBee operations and monitor status of Zigbee devices.

<u>Web App WorkFlow</u>

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

Log in to amazon Development Portal [It available for free to create account]. Navigate to Alexa-> your console -> skills.



Alexa Developer Console

https://developer.amazon.com/alexa/console/ask?

alexa developer console

Feedback forum

Build a Monetized Skill Directly in the Developer Console, Earn Money

Enrich your skill with in-skill purchasing (ISP), available for skills in the US. Deliver premium experiences with consumables, one-time purchases, and subscriptions. You define your product offering and price, and we handle the voice-purchasing flow. You can choose to build a monetized skill directly in the developer console or use the Command-Line Interface (CLI) .

Welcome to the Alexa Skills Kit Developer Console

Visit our release notes to learn about new feature and tools. Curious about what's new? watch this video or read our documentation.

Skills | Earnings | Payments

Alexa Skills

Create Skill

SKILL NAME	LANGUAGE	TYPE	MODIFIED	STATUS	ACTIONS
thor96 View Skill ID	English (US)	Custom	2019-02-04	● In Development	Analytics | Edit | Delete

Figure 7: Create Account on Alexa developer

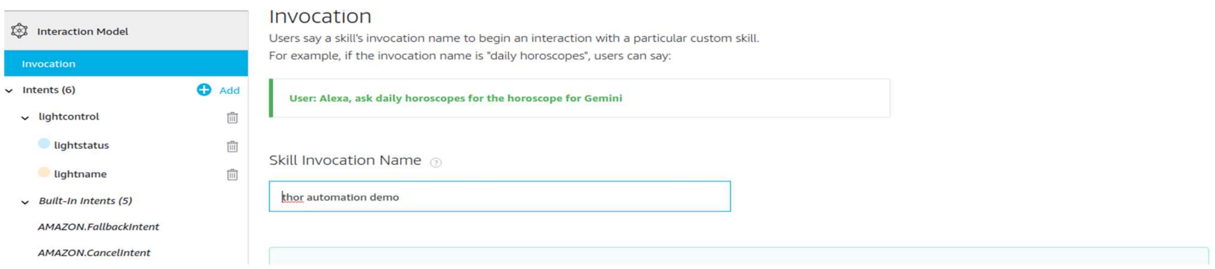
Creating Invocation:

This is the command the user should provide in order to start the custom skill.

Users say a skills' 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**".

<u>Steps</u>: click on invocation and fill the name inside skill invocation name



Invocation

Users say a skill's invocation name to begin an interaction with a particular custom skill.
For example, if the invocation name is "daily horoscopes", users can say:

User: Alexa, ask daily horoscopes for the horoscope for Gemini

Skill Invocation Name

thor automation demo

Figure 8: Create Invocation

Creating intent:

Intents are the individual commands that is provided within skill to provide multiple skills.

To create intent one 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 bed room light, control kitchen light, etc.

Steps: Click on add and put the name of intent [**lightcontrol**]

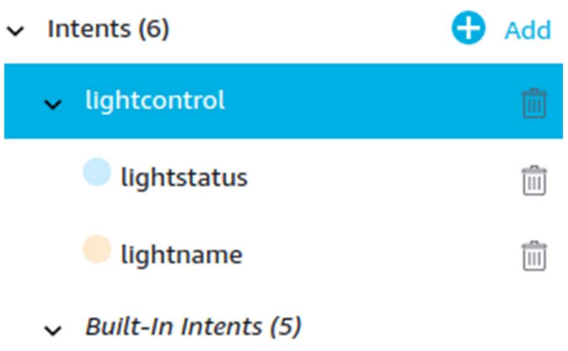
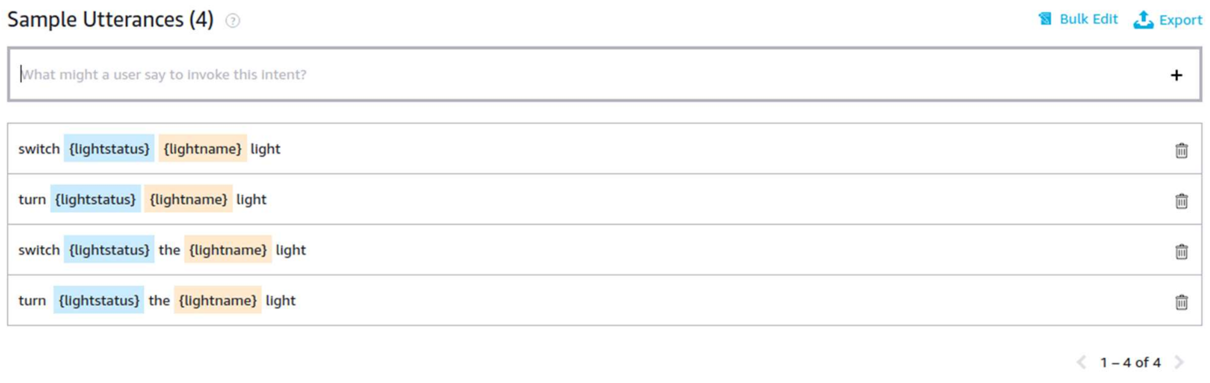


Figure 9: Create Intent

Creating Utterances:

Utterances provide the different flavour 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 flavour for different ways to say same thing.



switch {lightstatus} {lightname} light

turn {lightstatus} {lightname} light

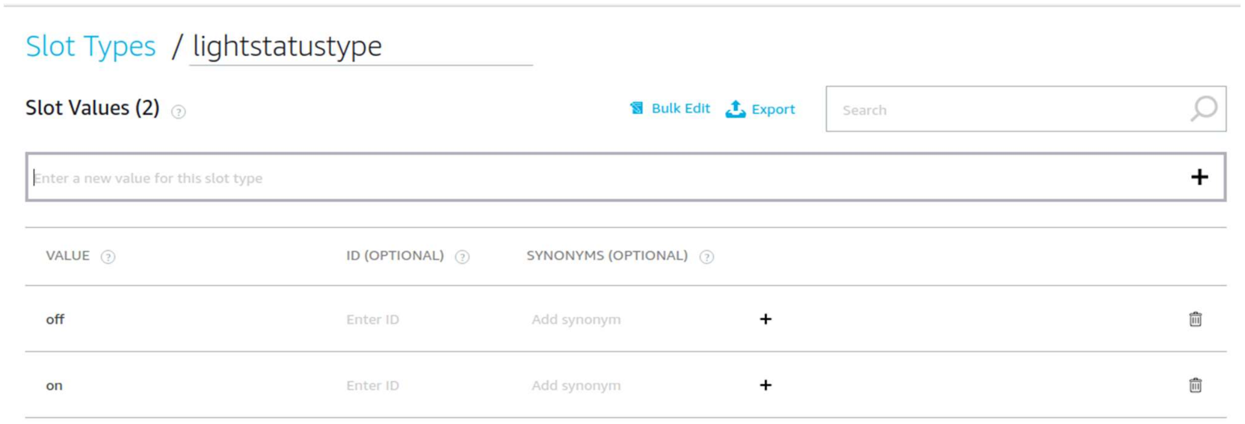
switch {lightstatus} the {lightname} light

turn {lightstatus} the {lightname} light

Figure 10: Create Utterances

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

Slot Values (2)

VALUE	ID (OPTIONAL)	SYNONYMS (OPTIONAL)
off	Enter ID	Add synonym
on	Enter ID	Add synonym

Figure 11: Create Slot

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 have to provide any IP nor dependent on any given IP.

Steps:

1. Sign in to AWS-IOT console,
2. Register a think [It created EIC_ALEX]
3. Create a certificate for your think [Refer: Figure, created certificate for device to AWS-IOT Connectivity].
4. Self-generated Licenses should be downloaded
5. Activate the License
6. Attach a policy [provided name of policy, and fill Action sec: iot:*, and Resource ARN: *].
7. Now Attach Policy to license.

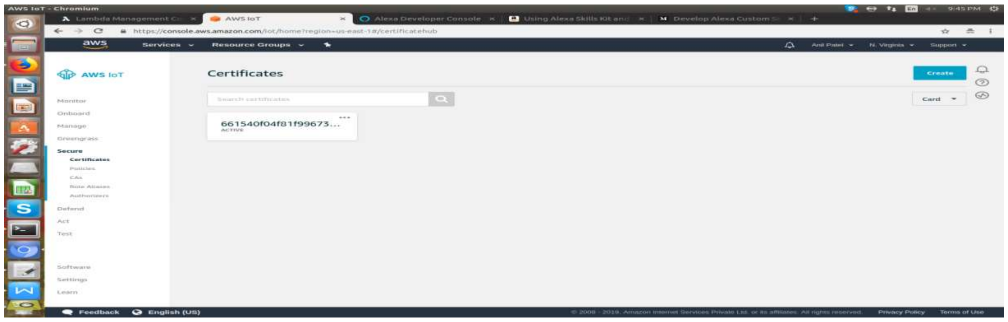


Figure 12: Create Certificate

Self-Generated license is used with Thor board:

1. root-CA.crt

2. 2.661540f04f-certificate.pem.crt -k

3. 661540f04f-private.pem.key

There is a way we can monitor [Refer Fig:11] the connectivity of devices either from backend [Thor96] or frontend [Lambda].

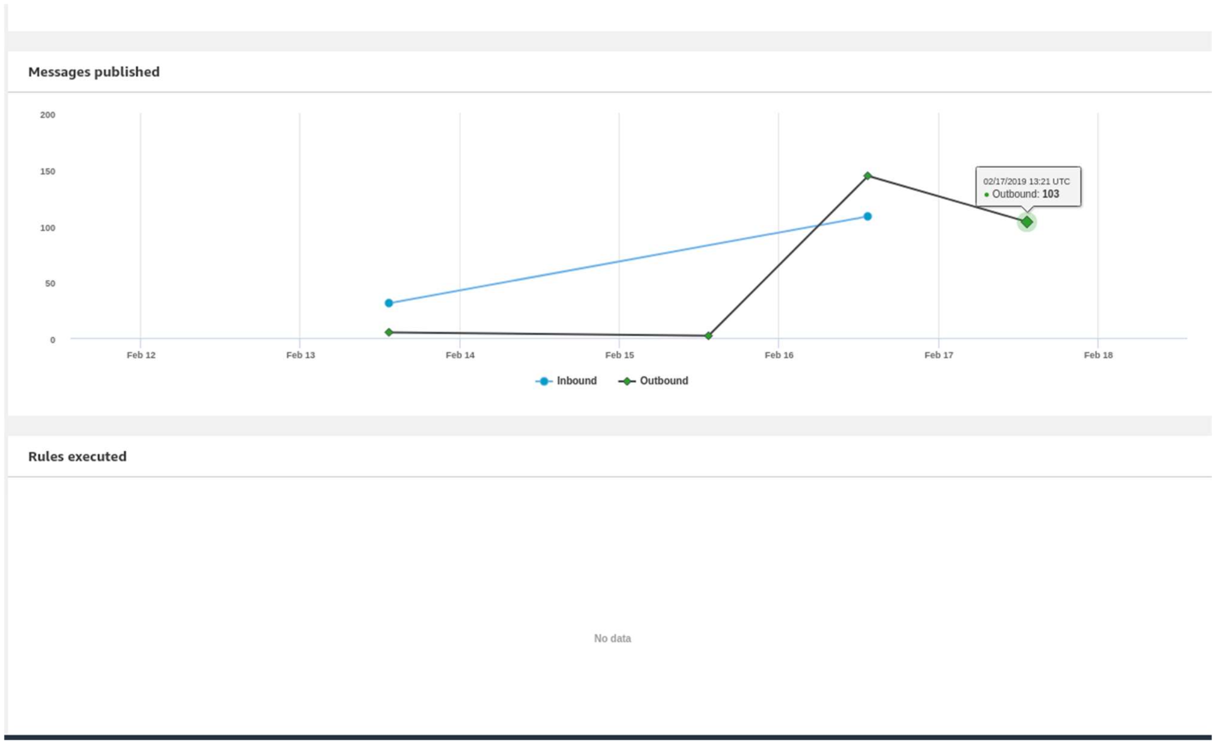


Figure 13: Monitor Device connectivity

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.

The below is delta of json which gets updated from Lambda and eventually updated on device.

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

9. Basics of Alexa:

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 behaviours the skill is able to 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 one says - **Alexa, ask daily horoscopes for the horoscope for Gemini.**

Here **Alexa** is considered the **wake word**. Now **Alexa** starts responding.

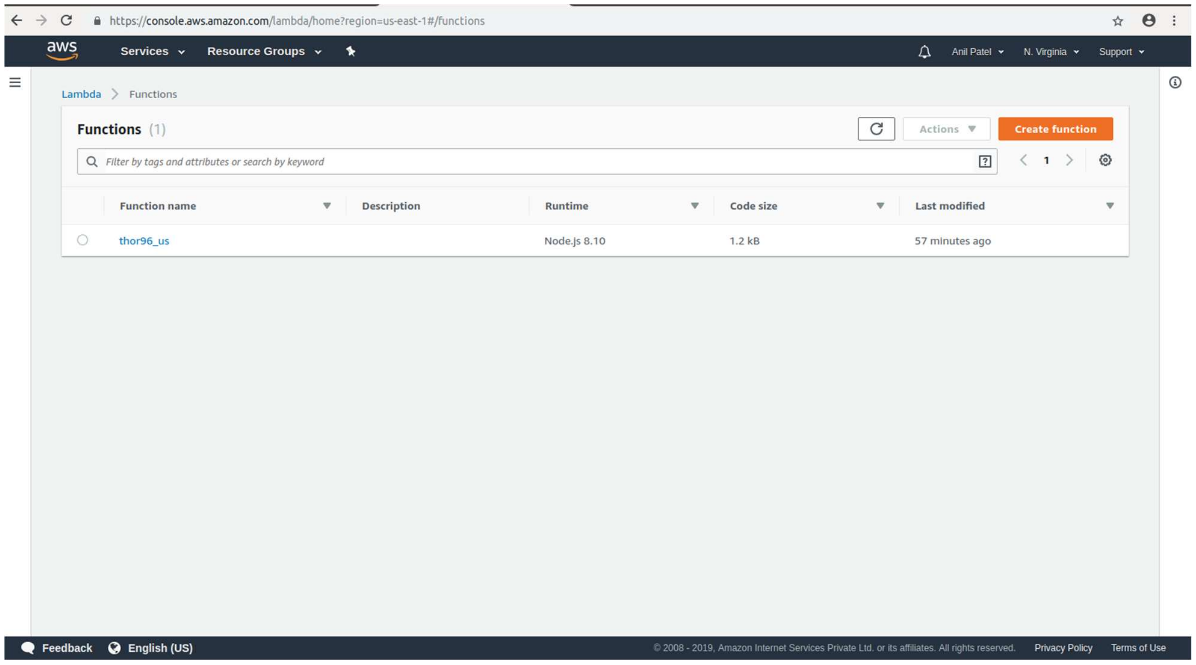
"Daily horoscopes" is **SKILL INVOCATION NAME**. So here skill which is enabled and whose' invocation name is "daily horoscopes" will be activated and rest of the words (**the horoscope for Gemini**) are sent for performing action.

10.AWS Lambda Function:

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

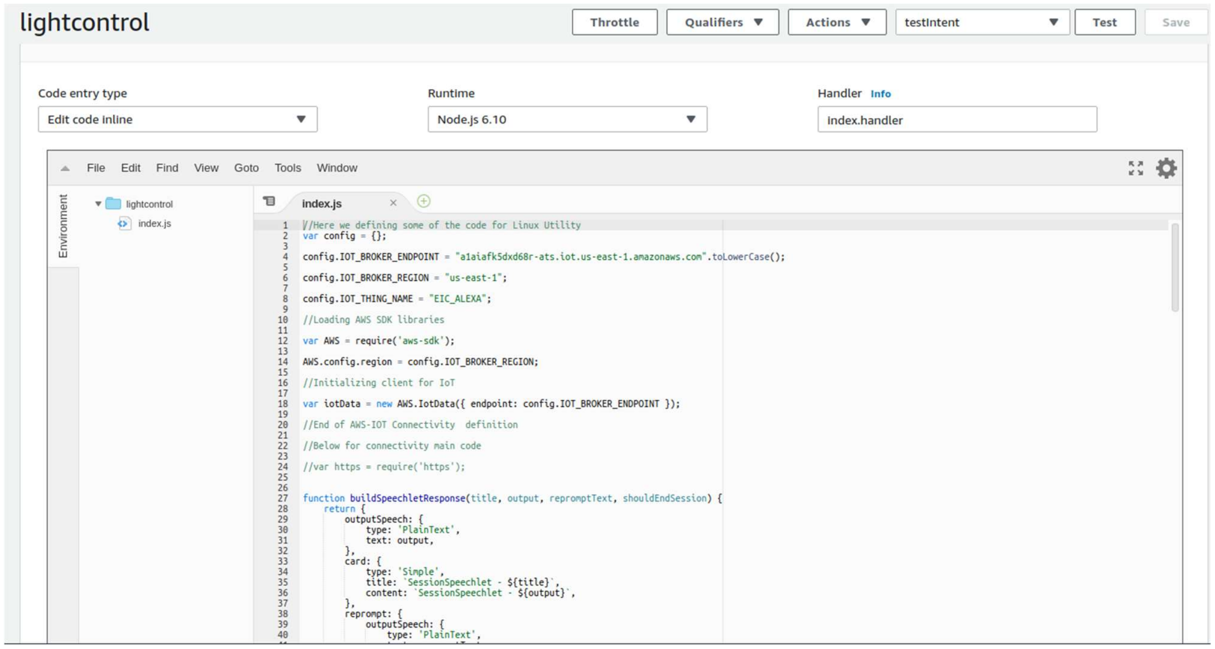
To develop Lambda Function, it is required to be logged into the AWS Console and go to lambda [It is free to create account on AWS].

Go to: Create Function -> Author from Scratch [Author from scratch] -> Write Name, select run-time, role and existing role and click on Create Function [Refer: Fig7].


https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions
aws
Services
Resource Groups
Anil Patel
N. Virginia
Support
Lambda > Functions
Functions (1)
Actions
Create function
Filter by tags and attributes or search by keyword
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

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

Figure 14: How to create lambda function at AWS



Code entry type: Edit code inline

Runtime: Node.js 6.10

Handler Info: index.handler

index.js

```
//Here we defining some of the code for Linux Utility
var config = {};

config.IOT_BROKER_ENDPOINT = "a1aiafk5dxd68r-ats.iot.us-east-1.amazonaws.com".toLowerCase();

config.IOT_BROKER_REGION = "us-east-1";

config.IOT_THING_NAME = "EIC_ALEXA";

//Loading AWS SDK libraries

var AWS = require('aws-sdk');

AWS.config.region = config.IOT_BROKER_REGION;

//Initializing client for IoT

var iotData = new AWS.IotData({ endpoint: config.IOT_BROKER_ENDPOINT });

//End of AWS-IOT Connectivity  definition

//Below for connectivity main code

//var https = require('https');


function buildSpeechletResponse(title, output, repromptText, shouldEndSession) {
    return {
        outputSpeech: {
            type: 'PlainText',
            text: output,
        },
        card: {
            type: 'Simple',
            title: `SessionSpeechlet - ${title}`,
            content: `SessionSpeechlet - ${output}`,
        },
        reprompt: {
            outputSpeech: {
                type: 'PlainText',
```

Figure 15: Define Lambda function

we can monitor the Lambda functionality for error(S) or server availability every moment when the Lambda has multiple load when accessing multiple devices.

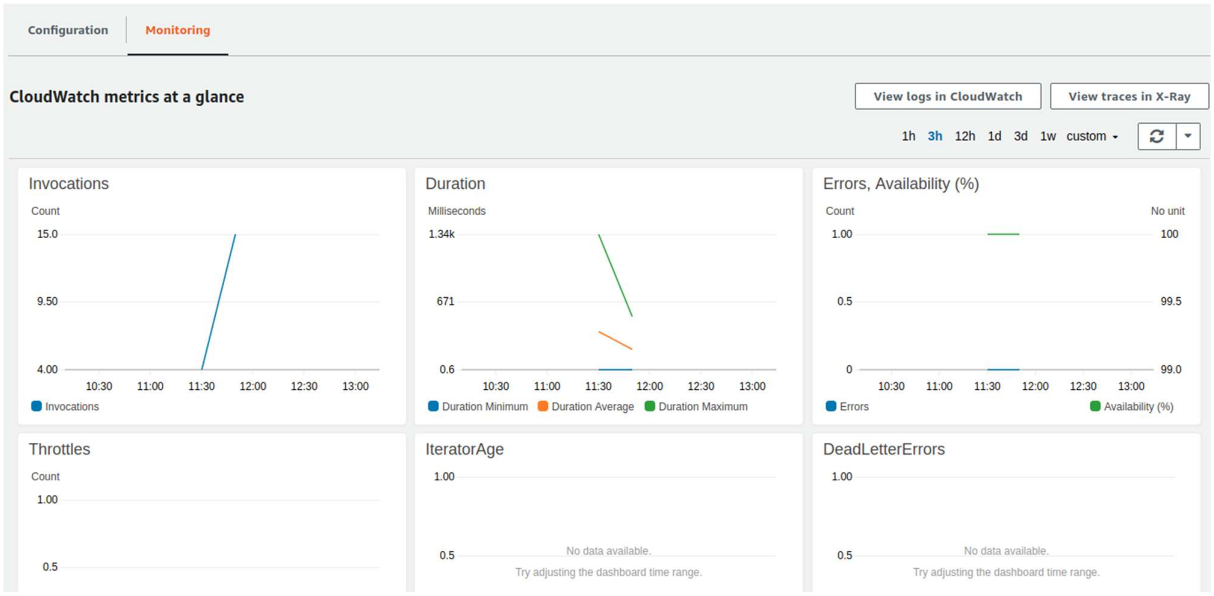


Figure 16: Lambda Cloud Watch