Lab: Add Voice Control Feature to AWS IoT Solutions

Overview	AWS IoT Device Shadow provides the ability to connect devices with your apps so that you can control the devices in your app. And you can create an Alexa Skill Kit (ASK) with AWS IoT Device Shadow to add voice-powered experiences to your connected devices. In this lab, you will learn how to add voice control features to your AWS IoT Solutions.

AWS IoT

Objectives	After completing this lab, you will be able to: • Create a custom Alexa Skill Kit • Create a Lambda function to integrate with AWS IoT Service Devcie Shadow

---

Pre-requisites	This lab requires: • An AWS account. • An Amazon Developer Account. • Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat). • For Microsoft Windows users: Administrator access to the computer. • An Internet browser such as Chrome, Firefox. With camera support would be prefer.

Duration	40 to 60 Minutes

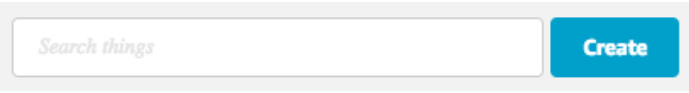
v.1.0.4.0605

Task 1: Create a device in AWS IoT Service

Overview	To connect a device to AWS IoT, we recommend you first create a device in the thing registry. This registry allows you to keep a record of all of the devices that are connected to your AWS IoT account.  AWS IoT

Task 1-1: Create a Device in the Thing Registry

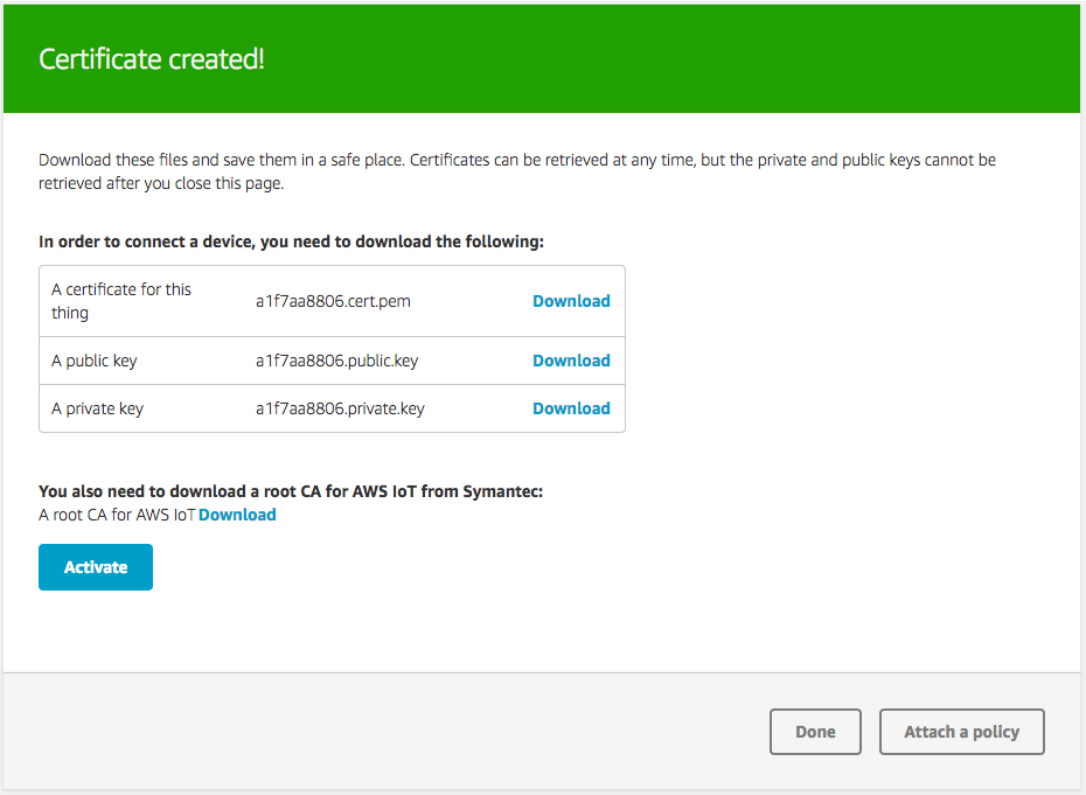
Overview	In this task you will create a Device in the Thing Registry for further usage in this lab..

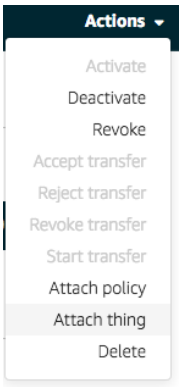
Step	Instruction
1.1.1	Sign in AWS Console, make sure you are using **US East (N. Virginia)** region, and the select AWS IoT from the menu: 
1.1.2	In the AWS IoT panel in the left, select **Registry → Things**，and then click on the **Create** button on the top right window： 
1.1.3	Enter " **myLight**" in the name column，and then click **Create Thing** button。

1.1.4	Now you will see the Thing window of myLight. Click the **Interact** in the left panel and you can see the device related information on the right, including REST API end poinr and MQTT Topic….etc.
1.1.5	Click on **Shadow** in the left panel. You can see or edit the shadow document (in JSON format) in this window if necessary. Usually we don't have to maintain it manually here. In this workshop, let's try to create a shadow document for myLight. Click on the **Create shadow document** button.
1.1.6	Click **Edit** in the right and then update the shadow document as below: **{** **"desired": {},** **"reported": {"led":0}** **}**
1.1.7	Click on **Save** when finished, and you will see the new shadow document and Shadow metadata.
1.1.8	On top left of the window, select the back () button. Finish create a device in the thing registry.

Task 1-2: Creating and Activate an AWS IoT Device Certificate and Policy	
Overview	Communication between your AWS IoT button and AWS IoT is protected through the use of X.509 certificates. AWS IoT can generate a certificate for you or you can use your own X.509 certificate. AWS IoT policies are used to authorize your button to perform AWS IoT operations, such as subscribing or publishing to MQTT topics. Your button will present its certificate when sending messages to AWS IoT. To allow your button to perform AWS IoT operations, you must create an AWS IoT policy and attach it to your device certificate. In this task we will use AWS IoT to generate an X.509 certificate and activate it for you. Also we will create an AWS IoT policy and attach it to the device certificate.

Step	Instruction
1.2.1	On AWS IoT left panel, select **Security → Certificates** and the click on the Crreate button on the top right of the window.

1.2.2	Click **Create certificate** button, and AWS IoT will create the keys and certificate for myLight：  **[Note] PLEASE DO NOT CLOSE THIS PAGE before finish downloading the keys and certificate.**
1.2.3	Create an **IoTCerts** folder in your filesystem, and the click on the **FOUR Download** links on the page. Save the **Four keys and certificate files** to your computer including the **root CA** file.
1.2.4	Click on the **Active** button, and then click the **Done** button.

1.2.5	Select the certificate which you just created in the Certificates window. Then select **Attach thing** in the **Actions** menu: 
1.2.6	In the Attach things to certificate window, search **myLight** and then click on the checkbox. Then select **Attach** button and back to the Details of the certificate.
1.2.7	On top left of the window, select the back () button. Back to the AWS IoT console.
1.2.8	Click on **Security → Policies** on the left panel. And the click on the Create button on the top righ tof the window to create a new policy for myLight device.
1.2.9	In the Create a policy window, type **myLightpolicy** in the **Name** column. From the **Action** menu, choose **iot:***. In the **Resource ARN** field, type * to allow access to all resources. Select the **Allow** check box in the Effect and the click the **Create** button.
1.2.10	On top left of the window, select the back () button. Back to the AWS IoT console. Now we need to attach this policy to the device certificate. Repeat 1.2.5 to find the certificate you created, and then select **Attach policy** from the **Actions** menu.
1.2.11	In the Attach policies to certificate(s) window, seach **myLightPolicy** in the and select it by clicking the check box. Click the **Attach** button.

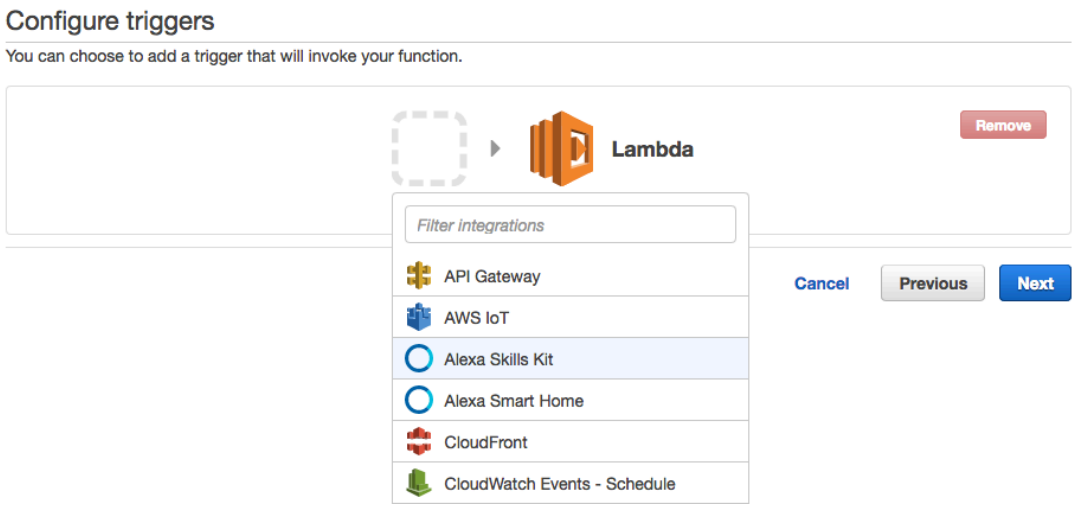
Task 2: Create an Alexa Skill Kit

Overview	AWS IoT Service can integrate with your applications to let users to manage their devices. In this task, you will learn how to create an Alexa Skill Kit to control your device via AWS IoT.

Task 2-1: Create a Lambda function for Alexa Skill Kit

Overview	The easiest way to build the cloud-based service for a custom Alexa skill is by using AWS Lambda, so there is no need to provision or continuously run servers. AWS Lambda supports JavaScript and you upload the code for your Alexa skill to a Lambda function and Lambda does the rest, executing it in response to Alexa voice interactions and automatically managing the compute resources for you.

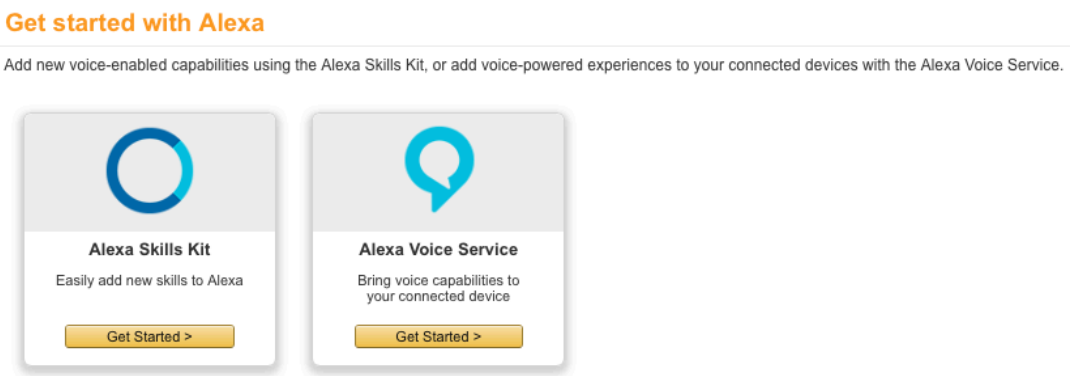
Step	Instruction
2.1.1	Open AWS Lambda Console, and click on **Create a Lambda function → Blank Function**.

2.1.2	In the Configure triggers page, choose **Alexa Skills Kit** as the trigger of the new lambda function. Then click the **Next** button. 
2.1.3	In the Name field, input **myLightController** as the lambda function name and make sure the Runtime is **Node.js**. Please download the Alexa Skill Lambda template from: **https://bit.ly/alexa-iot-lambda.** Open the file downloaded and copy/paste the contents of the file to Lambda function code.
2.1.4	In the Role field, select **Create new role → Create a custom role**. The AWS console will open a new window for the new role.
2.1.5	In the new window, select **Create a new IAM Role** in the IAM Role field. Enter **lambda_basic_execution_myLight** in the Role name field and then click the Allow button on the bottom right corner of the window.
2.1.6	In the Lambda console, select **Next → Create function**.
2.1.7	Copy the ARN of the Lambda function on the right-top corner. The ARN format should be like this: **arn:aws:lambda:us-east-1:XXXXXXXXXX:function:skill-light-control**
2.1.8	Open **IAM** in the AWS console and then select **Roles** in the left panel.

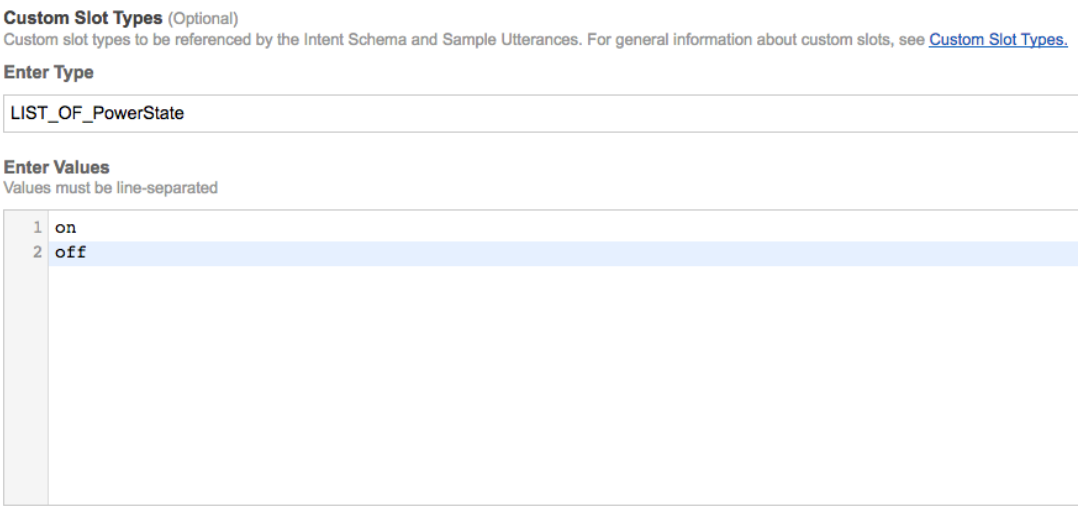
2.1.9	Select and open the **lambda_basic_execution_myLight** role, and click the **Attach Policy** button in the Managed Policies. Select **AWSIoTDataAccess** and then click on the **Attach Policy** button. 

Task 2-2: Create an Alexa Skills Kit

Overview	In this task, we will create a new Alexa Custom Skills Kit to trigger the Lambda function

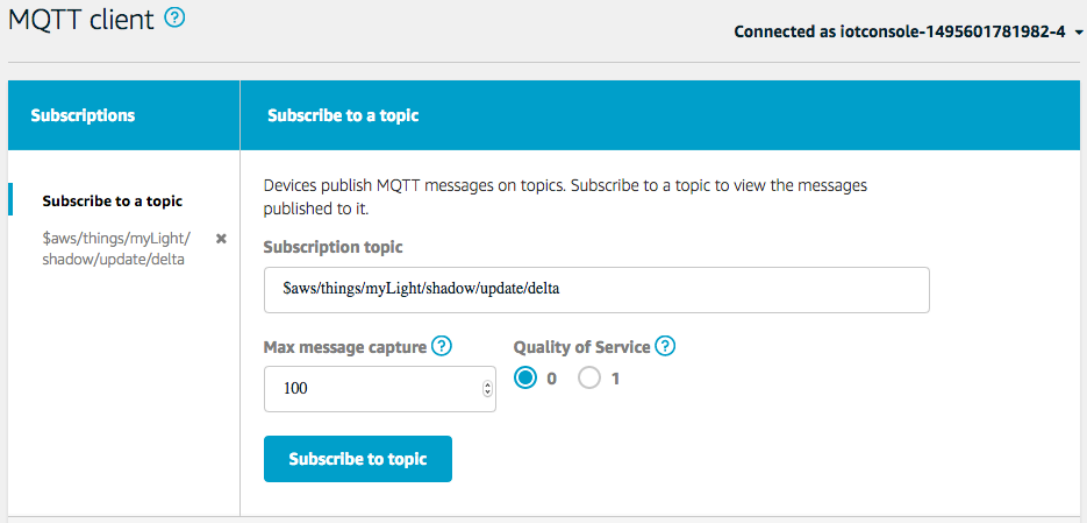
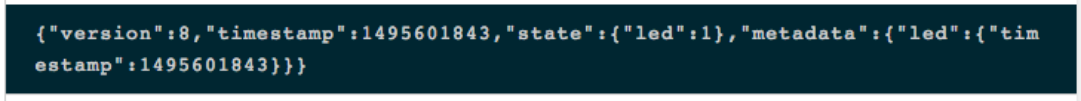
Step	Instruction
2.2.1	Sign up and login Amazon Developer Portal (https://developer.amazon.com) with your Amazon account.
2.2.2	Click on **Alexa**, and then click on **Get Started** of Alexa Skill Kit. 

2.2.3	Select **Add a New Skill**.
2.2.4	In the Create a New Alexa Skill page, select **Custom Interaction Model** as the skill type. Enter **Light Control** in the name field and **Jarvis** in the invocation name field. Click **Save → Next** when finished.
2.2.5	In the Interaction Model window, enter below script into Intent Schema: **{** **"intents": [** **{** **"intent": "LightIntent",** **"slots": [** **{** **"name": "Power",** **"type": "LIST_OF_PowerState"** **}]** **}]** **}**

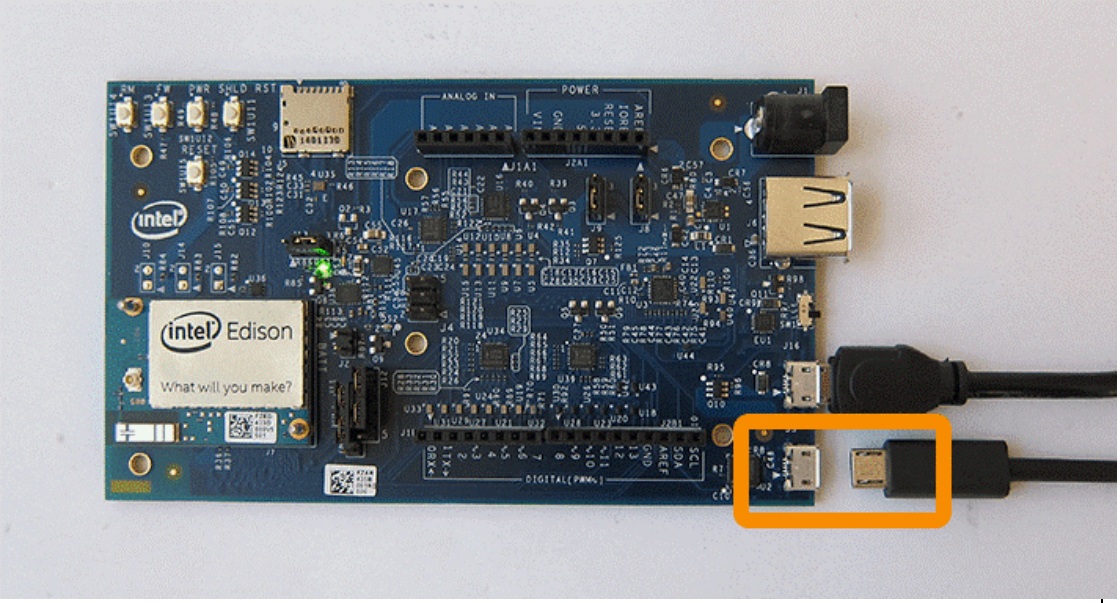
2.2.6	Enter **LIST_OF_PowerState** as the name of the Custom Slot Types. And then enter **on** and **off** as the value of the type: 
2.2.7	Click on the **Add** button to add the new custom slot type.
2.2.8	Enter below script into Sample Utterances and the click **Next**: **LightIntent turn {Power} the light.**
2.2.9	In the Configuration page, select **AWS Lambda ARN** and select **North America**. Paste the ARN which you copied in the step 2.1.7. It should be similar as below: **arn:aws:lambda:us-east-1:xxxxxxxxxx:function:myLightController**
2.2.10	Select **No** on the account linking, and then click **Next**.

2.2.11	Now we can test the voice control feature in the Test page. In the Service Simulator section, enter "**Alexa, ask Javis to turn on the light.**" Into the Enter Utterance field and click on the **Ask Light Control** button. Then you will found "The light has been turned on!" output speech message in the Lambda Response as below:
	
	Lambda Response
	
	`{`
	`  "version": "1.0",`
	`  "response": {`
	`  "outputSpeech": {`
	`  "type": "PlainText",`
	`  "text": "The light has been turned on!"`
	`  },`
	`  "card": {`
	`  "content": "SessionSpeechlet - The light ha`
	`  "title": "SessionSpeechlet - LightIntent",`
	`  "type": "Simple"`
	`  },`

Task 2-3: View Device MQTT Messages with the AWS IoT MQTT Client	
Overview	Before connected to a real device, you can use AWS IoT MQTT client to better understand the MQTT messages sent by a device. Devices publish MQTT messages on topics. You can use the AWS IoT MQTT client to subscribe to these topics to see the content of these messages.

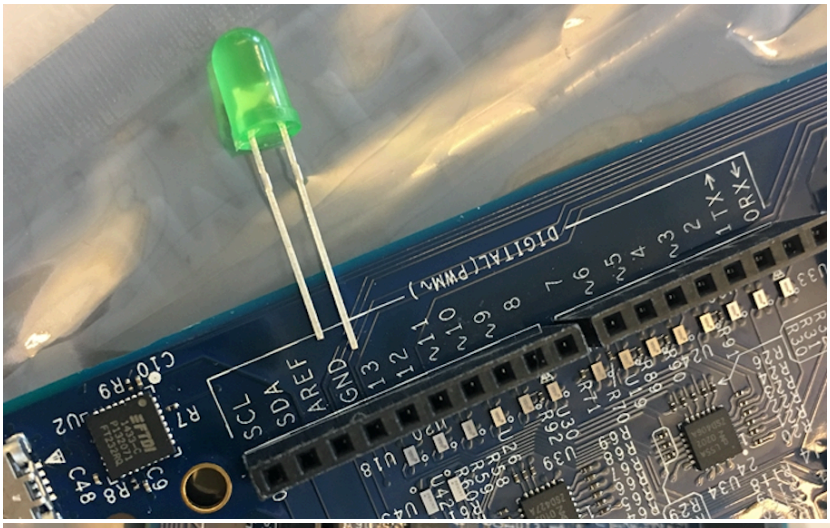
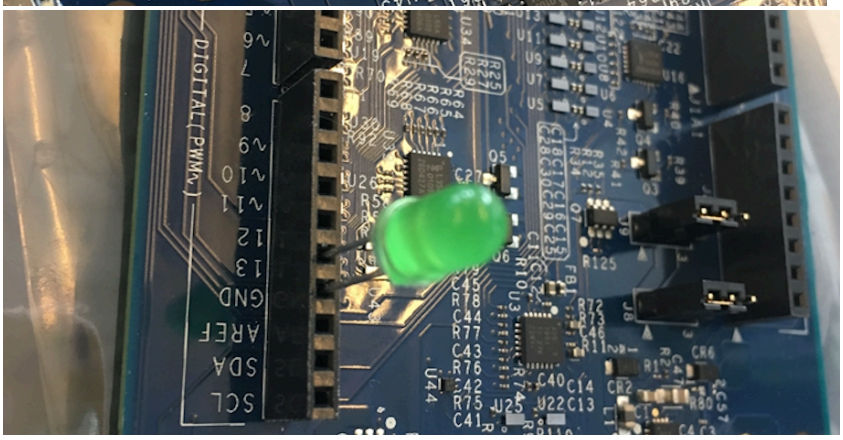
Step	Instruction
2.3.1.	In the AWS IoT console, click **Test** on the left panel..
2.3.2.	Then subscribe to the topic on which your thing publishes. Enter **$aws/things/myLight/shadow/update/delta** to the Subscription topic field, and then choose **Subscribe to topic** button. 
2.3.3.	Click on the subscription on the left panel. Execute step 2.2.11 again, then MOTT Client will receive the delta message as below: 

Task 2-4: Test the results on Intel Edison	
Overview	Now, we are going to test the results on a Intel Edison board. You can control the LED light by Alexa once you successfilly connect your Intel Edison board to AWS IoT.

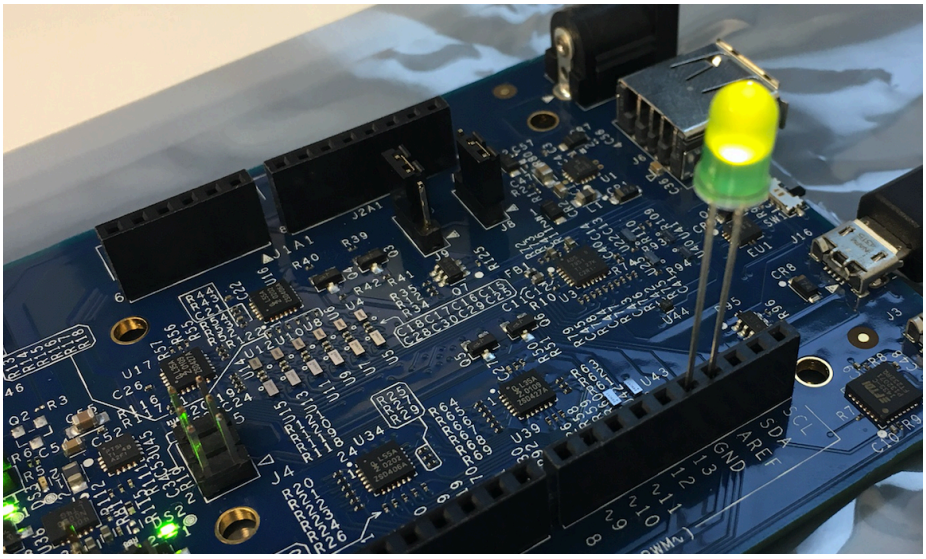
Step	Instruction
2.4.1.	Please refer to the Intel Edison manual to power your board and establish serial communication via the two micro-USB ports. https://software.intel.com/en-us/appendix-connectors-on-the-intel-edison-board 

2.4.2.	Depends on your OS, use one of below links to set up a serial terminal to your Intel Edison board. Use **root** as login id and **password** as password to login. Windows: https://software.intel.com/en-us/setting-up-serial-terminal-on-system-with-windows Mac OS X: https://software.intel.com/en-us/setting-up-serial-terminal-on-system-with-mac-os-x Linux: https://software.intel.com/en-us/setting-up-serial-terminal-on-system-with-linux
2.4.3.	Use **configure_edison --wifi** to connect your Intel Edison board to Wi-Fi. Remember the IP address of your board. You can refer to detail instructions here: https://software.intel.com/en-us/connecting-your-intel-edison-board-using-wifi
2.4.4.	Download and install FileZilla on your computer. You can get it from here: https://filezilla-project.org/
2.4.5.	Open FileZilla and enter the IP address of your board as the Host. Use port 22 to establish the connection.
2.4.6.	Use FileZilla to open **aws_certs** subfolder in the root folder on your board. And then copy the certificate files on your computer which you downloaded from AWS IoT on step 1.2.3 to the new **aws_certs** folder
2.4.7.	In the terminal window, use **vi homegateway.js** command to open the file in the root folder.

2.4.8.	Click on " **i** " key on your keyboard to enter insert mode. Update the value of below two variables based on your certificate file name:
	
	`var awsClientCert = "Your device certicate file name";`
	`var awsClientPrivateKey = "Your device private key file name";`
	`var topicName = "lightcontrol";`

2.4.9.	Click on " **esc "** key on your keyboard again to leave insert mode. Enter **:wq!** to close vi tool and overwrite the homegateway.js file.
2.4.10.	Connect the long leg (+) of the LED to the 13 digital pin and the short leg (-) to the GND pin on the board.  

2.4.11.	In the terminal window, enter **node homegateway.js** to execute the IoT client. You will see below output:
	
	`>> Initial LED......`
	`Device Shadow 初始化中，請稍候.....`
	`Device Shadow 連接中，請稍候.....`
	`received accepted on myLight:`
	`{"state":{"reported":{"red":187,"green":114,"blue":222,"displaytext":"Hello!","relay":0,"relay2":0,"led":0}},……..`

2.4.12.	Repeat step 2.2.11, you will see below output: **received delta on myLight:** {"timestamp":1496283987,"state":{"led":1},"metadata":{"led":{"timestamp":1496283987}}} LED is on.... received accepted on myLight: {"state":{"reported":{"red":187,"green":114,"blue":222,"relay":0,"relay2":0,**"led":1**}},…… And the LED will be turned on as well: 
2.4.13.	Repeat step 2.2.11, you can try "**turn off the light**" and the LED will be turned off.
2.4.14.	Clear the resources created in the AWS IoT console.

