

Enable Amazon Alexa Voice Service (AVS)* on the Intel® NUC

Tutorial

January 2017



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: <http://www.intel.com/design/literature.htm>

Intel® NUC and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

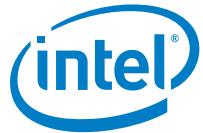
*Other names and brands may be claimed as the property of others.

Copyright © 2017, Intel Corporation. All rights reserved.



Contents

| | |
|---|----------|
| 1.0 Tutorial Steps..... | 5 |
| 1.1 Setup Arduino 101* and Firmata..... | 5 |
| 1.2 Wire up the sensors | 5 |
| 1.3 Set Up Firmata* on Arduino 101* | 6 |
| 1.4 Setup Gateway as Echo..... | 9 |
| 1.5 Reboot the Intel® IoT Gateway | 9 |
| 1.6 Generate SSL Key for Intel® NUC..... | 9 |
| 1.7 Create an AWS Account: | 9 |
| 1.8 Security profile setup for Intel® NUC on Amazon* Developer console..... | 10 |
| 1.9 Add Your Credentials to Creds.py..... | 10 |
| 1.10 Create an AWS IoT Device: | 11 |
| 1.11 Edit main.py: | 16 |
| 1.12 Create a Lambda Function..... | 17 |
| 1.13 Configure IAM access policy..... | 22 |
| 1.14 Configure Custom Alexa Skill | 24 |
| 1.15 Run the program | 29 |



Revision History

| Date | Revision | Description |
|--------------|----------|------------------|
| January 2017 | 1.0 | Initial release. |



1.0 **Tutorial Steps**

This tutorial explains steps to port Amazon Alexa Voice Service (AVS)* on Intel® NUC Gateway with Wind River Linux* operating system. This tutorial assumes you have already done some basic setup on your Intel® IoT Gateway and are familiar with its operation.

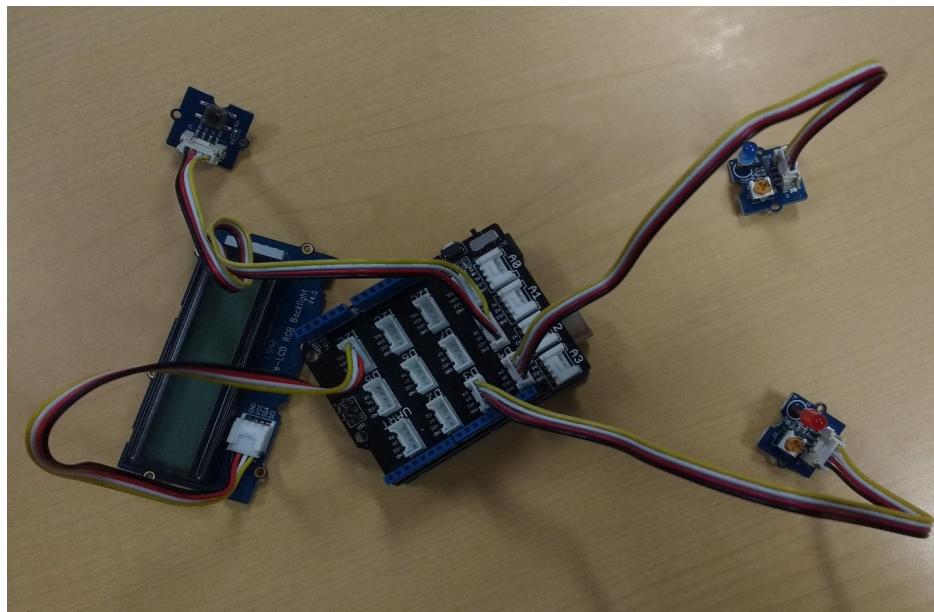
1.1 **Setup Arduino 101* and Firmata**

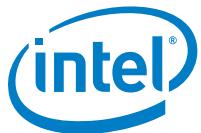
Clone the GitHub* repository onto your Intel® IoT Gateway by logging into the gateway and running:

```
git clone https://github.com/SSG-DRD-IOT/Alexa-on-Intel-NUC
```

1.2 **Wire up the sensors**

Connect Grove* button to pin D8, one LED to pin D3 and other one to Pin D4 as shown in the following picture using the cables from the Grove* starter sensor kit box.



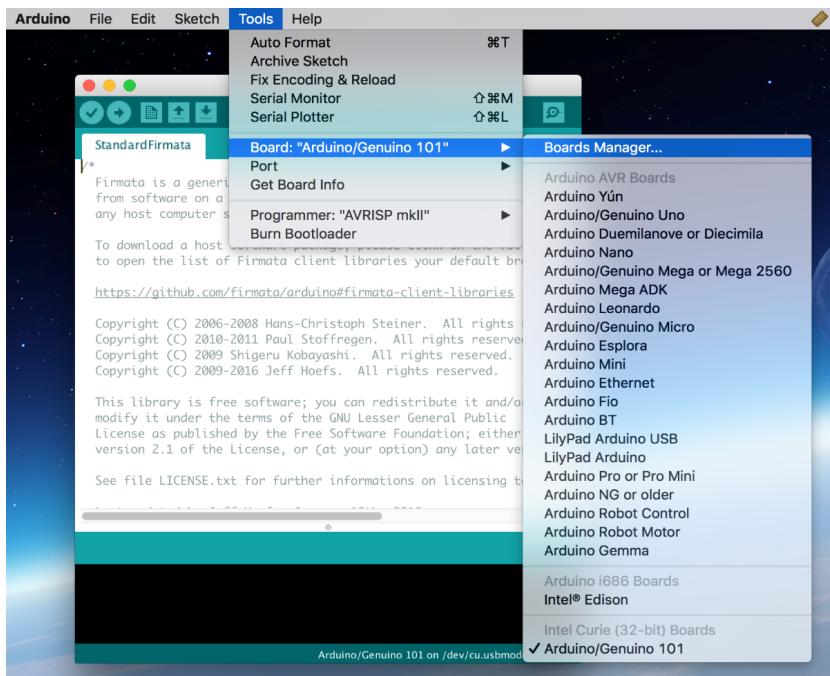


| Sensor | Details | Pin Connection |
|---------------------------|---------|----------------|
| Grove - Red LED | | D3 |
| Grove - Blue LED | | D4 |
| Button | | D8 |
| Grove - LCD RGB Backlight | | I2C |

1.3

Set Up Firmata* on Arduino 101*

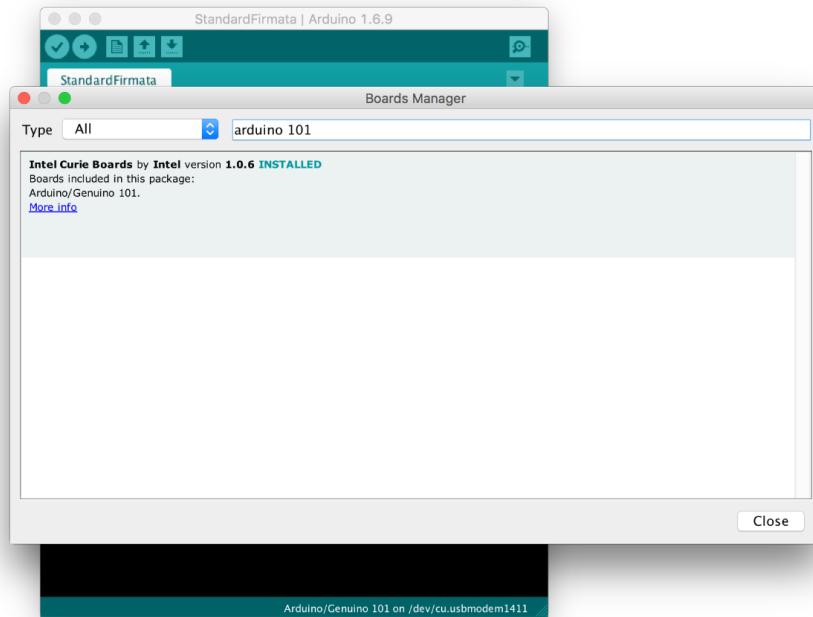
- a. Download Arduino* IDE: <https://www.arduino.cc/en/Main/Software>
- b. Connect Arduino 101* (branded Genuino 101* outside the U.S.) to a host computer via USB
- c. Launch Arduino IDE
- d. Open boards manager:



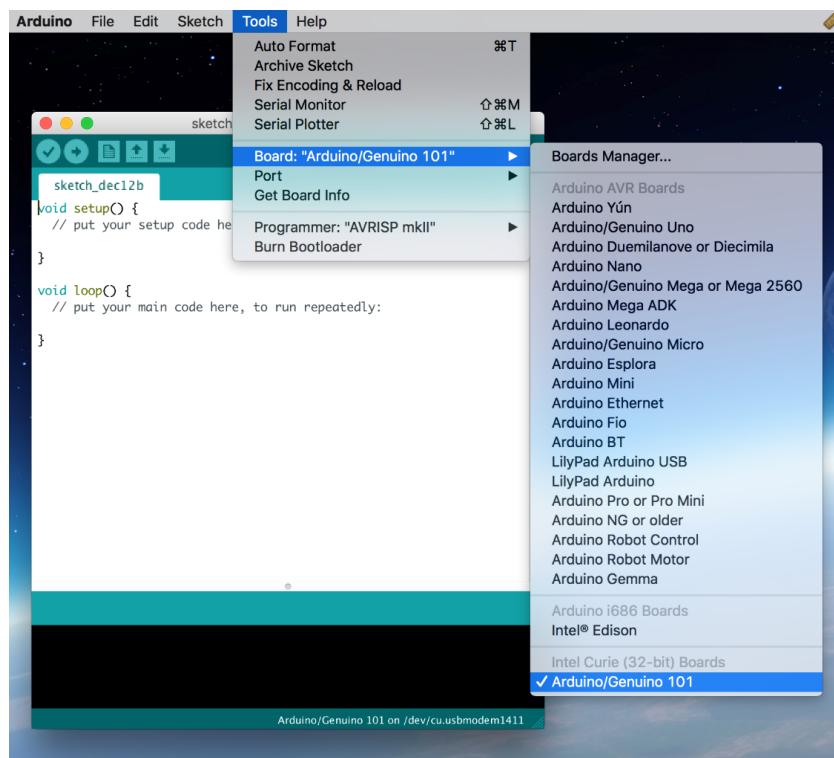
Tutorial Steps



- e. Search for "Arduino 101" and install the board definition:

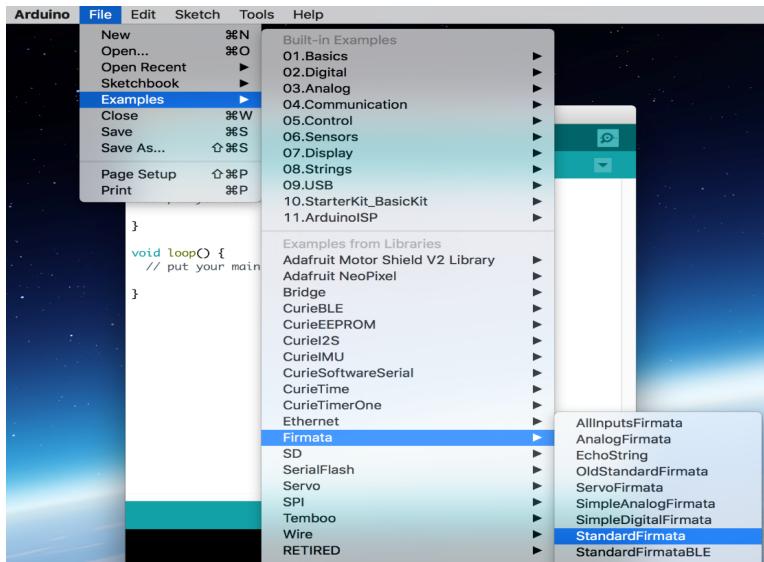


- f. Select "Arduino/Genuino 101" from the Boards menu and the correct port. If you are unsure which port to use disconnect the Arduino 101 and see which port disappears.

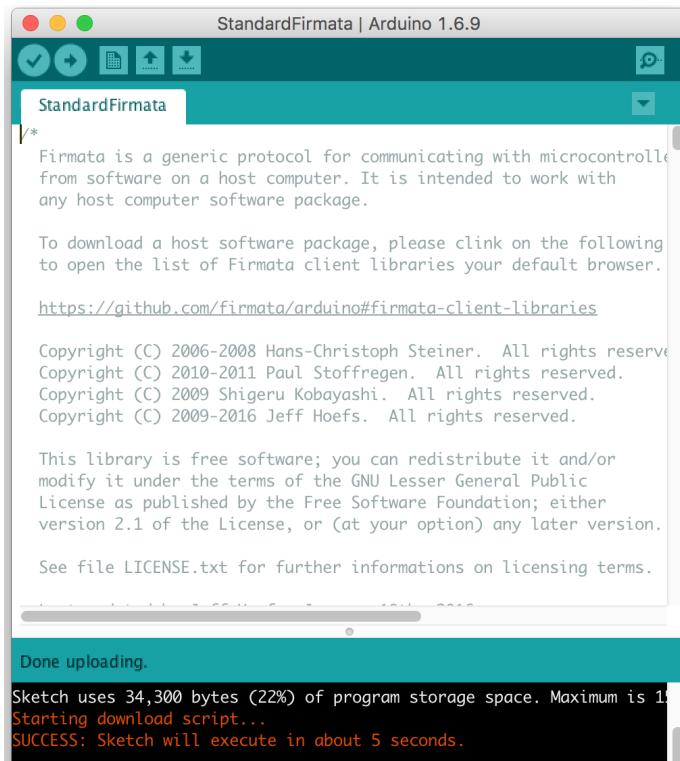




g. Load StandardFirmata Sketch:



h. Click Upload and wait for "Sketch will execute about 5 seconds"





1.4 Setup Gateway as Echo

- a. SSH onto the Gateway:

```
ssh root@IP.OF.GATE.WAY
```

Where IP.OF.GATE.WAY is the IP address of your gateway

Use “root” as the password

- b. CD into the Repo:

```
cd ~/Alexa-on-Intel-NUC
```

- c. Run setup.sh:

```
chmod +x ./setup.sh
```

```
./setup.sh
```

The script will ask you for confirmation to install a number of packages, enter “y” and hit enter for all.

The setup script will install the dependencies needed for the project and will take around 5 minutes to finish.

1.5 Reboot the Intel® IoT Gateway

1.6 Generate SSL Key for Intel® NUC

- a. Run these commands:

```
openssl genrsa -out /root/privkey.pem 2048
```

```
openssl req -new -x509 -days 365 -key
/root/privkey.pem -out /root/cert.pem
```

Fill out your information for the SSL certificate.

1.7 Create an AWS Account:

Follow the instructions at this link to create an Amazon Web Services (AWS)* account and security profile: <https://github.com/alexa/alexa-avs-sample-app/wiki/Linux>



1.8

Security profile setup for Intel® NUC on Amazon* Developer console

- a. Login to Amazon Developer Portal - developer.amazon.com
- b. Go to APPS & SERVICES -> Security Profile
- c. Click Web setting tab

The screenshot shows the Amazon Developer Console interface. At the top, there's a navigation bar with links like DASHBOARD, APPS & SERVICES (which is highlighted), ALEXA, REPORTING, SUPPORT, DOCUMENTATION, SETTINGS, and user-specific links for PRIYANKA, INTEL, SIGN OUT, and ENGLISH. Below the navigation bar, there's a secondary menu with links for My Apps, App Testing Service, Promotions, Security Profiles (highlighted in orange), Login with Amazon, Dash Replenishment Service, Alexa New, GameCircle, and PC / Mac & Web Instant Access. At the bottom of this menu, there are links for Tester Management, Advertise Your App, and Mobile Ads.

The screenshot shows the Security Profile Management page for the "Alexa Voice Service Sample App Security Profile". The title is "Alexa Voice Service Sample App Security Profile - Security Profile". There are tabs at the top: General (selected), Web Settings (highlighted in red), Android/Kindle Settings, and iOS Settings. Below the tabs, there's a note about specifying allowed JavaScript origins or return URLs for Login with Amazon. Under "Allowed Origins", the URL "https://localhost:3000" is listed. Under "Allowed Return URLs", the URL "https://localhost:3000/authresponse" is listed, and below it, the URL "https://192.168.1.197:5000/code" is listed and highlighted with a red box labeled "2". To the right of the URLs, there's a "More Information" section with links to Login with Amazon, GameCircle, and Device Messaging. At the bottom right, there's an "Edit" button with a red box labeled "1" around it.

- d. Click Edit button on Web settings tab.
- e. Get IP address of your Intel® NUC using "ifconfig" command.
- f. Add the URL in the form https://ip_address_for_NUC:5000/code to Allowed Return URLs
- g. e.g. <https://192.168.1.197:5000/code>

1.9

Add Your Credentials to Creds.py

- a. Navigate to the General tab on the Security Profile Management page:

Tutorial Steps



The screenshot shows the Amazon Developer Console interface. At the top, there's a navigation bar with links like DASHBOARD, APPS & SERVICES (which is highlighted), ALEXA, REPORTING, SUPPORT, DOCUMENTATION, and SETTINGS. Below the navigation bar, there are several service links: My Apps, App Testing Service, Promotions, Security Profiles (which is highlighted in orange), Login with Amazon, Dash Replenishment Service, Alexa (with a 'New' badge), GameCircle, and PC / Mac & Web Instant Access. At the bottom of this section, there are links for Tester Management, Advertise Your App, and Mobile Ads.

Security Profile Management

[More Information](#)
[Login with Amazon](#)
[GameCircle](#)
[Device Messaging](#)

Alexa Voice Service Sample App Security Profile - Security Profile

[General](#) [Web Settings](#) [Android/Kindle Settings](#) [iOS Settings](#)

These settings apply to all the apps using this security profile. Your security profile credentials — Client ID and Client Secret — allow your app to securely identify itself to Amazon services. [Learn More](#)

Security Profile Name: Alexa Voice Service Sample App Security Profile
Security Profile Description: Alexa Voice Service Sample App Security Profile Description
Security Profile ID: [REDACTED]
Client ID: [REDACTED]
Client Secret: [REDACTED]
Consent Privacy Notice URL: <http://example.com>
Consent Logo Image: [REDACTED]

- b. Open creds.py using vi with: vi creds.py
- c. Hit "a" to enter insert mode
- d. Enter your information into ProductID, Client_ID, etc.
- e. Hit "esc", type ":wq", and hit enter to save the file

Creating Custom Skills For The Gateway to control sensors:

1.10

Create an AWS IoT Device:

In order for the gateway to communicate with the AWS* cloud we will create an AWS IoT Device to act as a bridge. Alexa* Voice Services will then push updates to the device shadow (a virtual persistent state) and the gateway will get updates from the device shadow.

- a. Create an AWS account: <https://console.aws.amazon.com>
- b. From Services select AWS IoT:

Make sure that your region is selected as N. Virginia (us-east-1). The AWS services are not yet online for all regions.



Tutorial Steps

The screenshot shows the AWS Management Console homepage. The top navigation bar includes the Intel logo, the URL https://console.aws.amazon.com, the AWS IoT icon, and the region N. Virginia. The main content area displays a grid of AWS services. The 'Internet Of Things' service, specifically 'AWS IoT', is highlighted with a red box. Other visible services include Compute, Storage, Database, Networking & Content Delivery, Migration, Developer Tools, Management Tools, Analytics, Application Services, Artificial Intelligence, Business Productivity, Game Development, Mobile Services, and Messaging.

The dashboard should look like this:

The screenshot shows the AWS IoT Dashboard. The top navigation bar includes the AWS IoT icon, the URL https://console.aws.amazon.com/iotv2/home?region=us-east-1#/dashboard, and the region N. Virginia. The main content area is titled 'Dashboard' and features a chart titled 'Successful connections'. The chart shows a line graph with data points for each day from December 15 to December 21. The values are approximately: 15. Dec: 10, 16. Dec: 15, 17. Dec: 13, 18. Dec: 10, 19. Dec: 10, 20. Dec: 40, 21. Dec: 10. Below the chart, there is a section titled 'Messages'.

- c. Go to: Registry -> Things and click "Create" to the right of the search bar. Follow the screens to create AWS IoT device.

Tutorial Steps

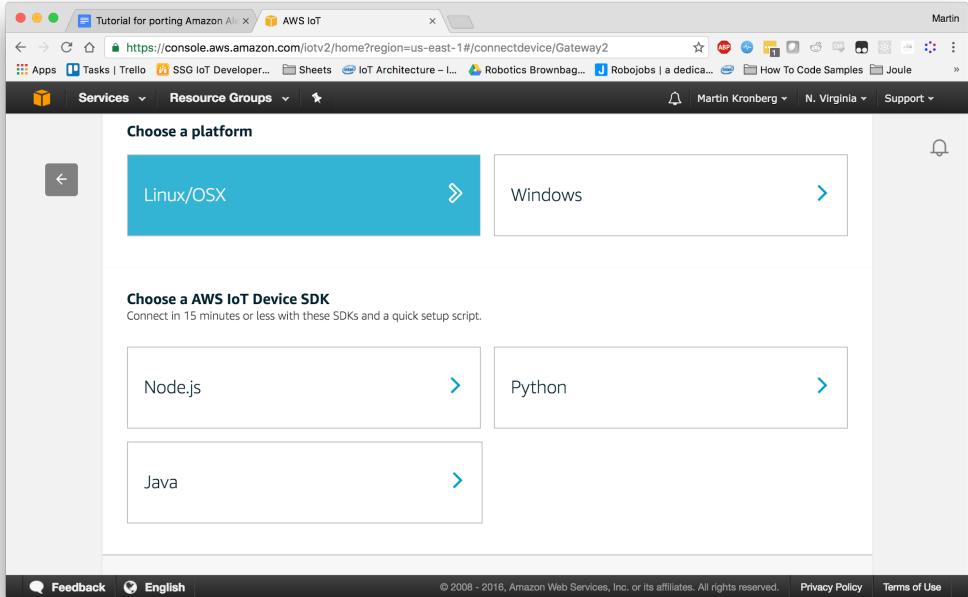


The screenshot shows the AWS IoT Things page. On the left, there's a sidebar with options like Dashboard, Connect, Registry (Things, Types), Security, Rules, Test, Settings, Learn, Feedback, and English. The main area is titled "Things" and contains a grid of 12 device cards. The devices listed are: frontdoor, window4, window3, window2, window1, tv1, backlight, frontlight, light4, light3, light2, light1, and NUC-Gateway. Each card has a "..." button. A search bar at the top right says "Search things" and a "Create" button.

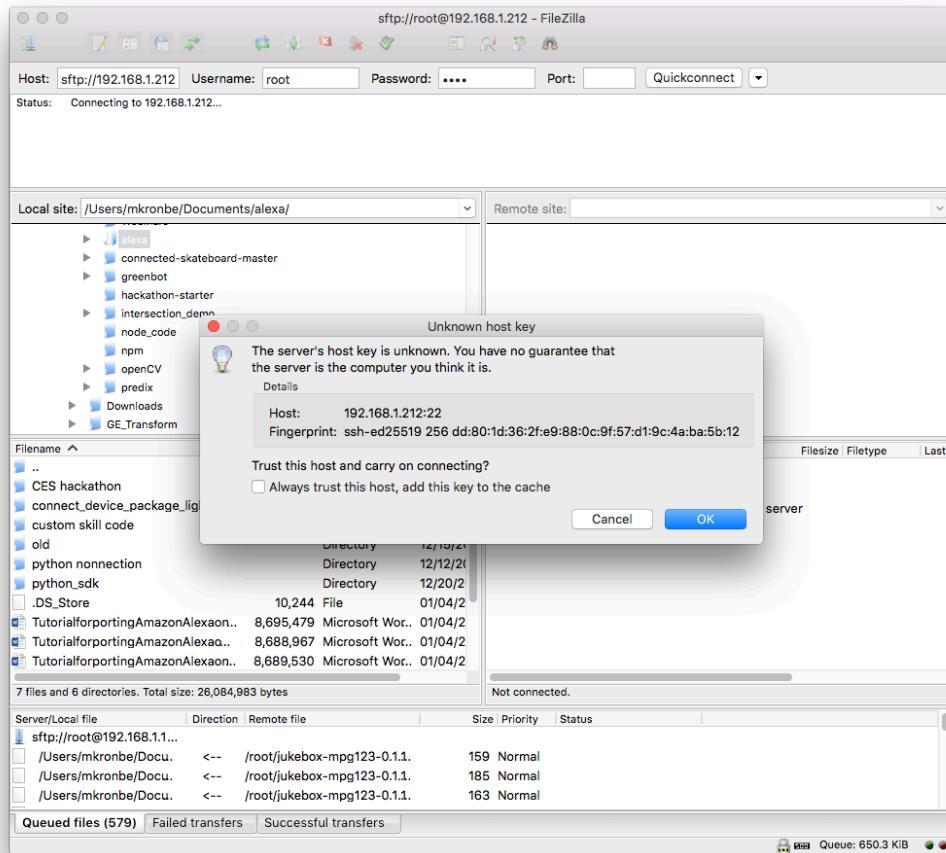
- d. On the device page navigate to Interact and click on “Connect a Device”

The screenshot shows the AWS IoT Thing details page for "Gateway2". The left sidebar has "Interact" selected. The main area shows "THING Gateway2 NO TYPE". It says "This thing already appears to be connected." and has a "Connect a device" button. Under "HTTPS", it says "Update your Thing Shadow using this Rest API Endpoint" and provides the URL "a2la7zf3kffmr.rf.iot.us-east-1.amazonaws.com". Under "MQTT", it says "Use topics to enable applications and things to get, update, or delete the state information for a Thing (Thing Shadow)" and provides the topic "\$aws/things/Gateway2/shadow/update".

- e. Choose your platform – Linux*/OSX* or Windows*, and choose Python*.



- f. Click "Getting Started" and download the connection kit for your selected OS. This package will contain a private key and certificate which we will use to connect the gateway to the AWS IoT service. It also contains a start.sh script which will run a basic connection test, install the SDK, and download the root-CA.crt.
- g. Transfer the zip file to the gateway using FTP from the command line, or FileZilla:
 - i. Go to <https://filezilla-project.org/> to download and install the FileZilla client
 - ii. Open Filezilla and connect to the IP of your Gateway using "root" as username, "root" as password, and 22 as the port:



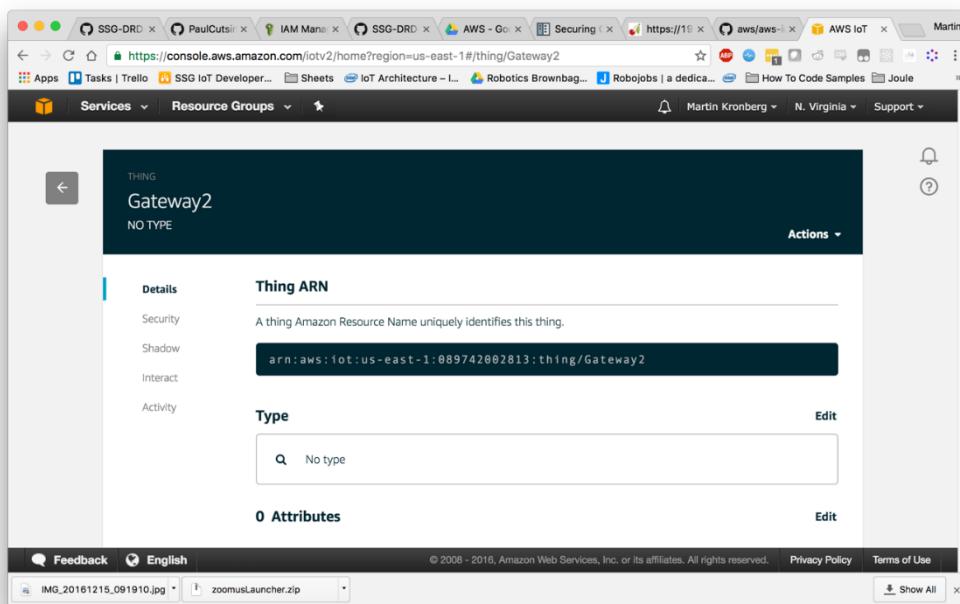
- iii. When asked about unknown host key hit OK
- iv. Navigate to the zip file on the left panel and the Alexa-on-Intel-NUC repository folder on the right panel. Double click the zip file to transfer it.



1.11 Edit main.py:

Main.py contains code for both sending voice commands to Alexa* and connecting to AWS IoT. Now that you have created a device you can add its connection details to main.py.

- a. Using VI open main.py and replace AWS_IOT_ARN with the Thing arn from the Details section of the AWS IoT device console:



- b. Replace REST_API_ENDPOINT with the Rest API Endpoint under the HTTPS section of the Interact section of the AWS IoT Console:



1.12

Create a Lambda Function

The lambda function is a function that runs in a virtual run time environment in AWS. We will use it to do two things: set up a handler for requests coming from Alexa* Voice Services, and setup a connection to AWS IoT to update the Gateway device shadow.

The Github* repository contains /lambda/index.js which you will have to edit in order to connect to your AWS IoT Device.

- a. On line 4 of index.js replace REST_API_ENDPOINT with the endpoint of your IoT Device. The endpoint can be found in the AWS IoT device console under **Interact**



The screenshot shows the AWS IoT console interface. At the top, there's a navigation bar with tabs like 'Services', 'Resource Groups', and 'Actions'. Below that, a sidebar on the left lists 'Details', 'Security', 'Shadow', 'Interact' (which is currently selected), and 'Activity'. The main content area is titled 'THING' and 'NUC-Gateway'. It says 'NO TYPE'. Under the 'Interact' tab, there's a 'HTTPS' section with a sub-section for 'MQTT'. A callout box highlights the URL 'iot.us-east-1.amazonaws.com'.

- b. Once you have edited index.js make a zip file containing index.js and /lambda/node_modules. You will upload this to Lambda later on.
- c. Navigate to the AWS Lambda service:
<https://console.aws.amazon.com/lambda/>
- d. Click 'Create Function' use the Blank Function blueprint:

Tutorial Steps



The screenshot shows the 'Select blueprint' step in the Lambda Management Console. The user is presented with a list of 70 available blueprints. The visible options include:

- Blank Function
- kinesis-firehose-syslog-to-json
- alexa-skill-kit-sdk-factskill
- kinesis-firehose-apachelog-to...
- cloudfront-modify-response-h...
- s3-get-object-python

A filter bar at the top left allows selecting a runtime. A status bar at the bottom indicates 'Viewing 1-9 of 70'.

e. Select "Alexa Skills Kit" as the trigger:

The screenshot shows the 'Configure triggers' step in the Lambda Management Console. The user has selected the 'Alexa Skills Kit' trigger. The configuration screen displays:

- A visual representation of the trigger connection between 'Alexa Skills Kit' and 'Lambda'.
- A note explaining that choosing 'Submit' will create a resource policy allowing the Alexa service to call the Lambda function.
- Navigation buttons: 'Cancel', 'Previous', and 'Next'.

At the bottom, there are links for 'Feedback', 'English', and legal notices: '© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.', 'Privacy Policy', and 'Terms of Use'.



- f. Give your function a name and description and choose “Upload a .ZIP file”

The screenshot shows the 'Configure function' page in the Lambda Management Console. The 'Name' field is filled with 'Test_Function'. The 'Description' field contains 'test'. The 'Runtime' dropdown is set to 'Node.js 4.3'. In the 'Lambda function code' section, the 'Code entry type' dropdown is set to 'Upload a ZIP file'. Below this, there is a 'Function package' section with an 'Upload' button. A note below the upload button states: 'For files larger than 10 MB, consider uploading via S3.' At the bottom of the form, there is a checkbox labeled 'Enable encryption helpers'.

- g. Click Upload and select the zip file you made earlier.
h. Enter index.handler for the Handler – this is the default handler for Lambda
i. Choose “Create new role from template”
j. Select "Simple Microservice Permissions" from "Policy Templates"

Tutorial Steps



Lambda Management Console

https://console.aws.amazon.com/lambda/home?region=us-east-1#/create/configure-function

Lambda function handler and role

Handler* index.handler

Role* Create new role from template(s)

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name* Lambda_IoT_Role

Policy templates Simple Microservice per...

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)* 128

Timeout* 0 min 3 sec

AWS Lambda will automatically retry failed executions for asynchronous invocations. You can additionally optionally configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ), such as an SQS queue or an SNS topic. Learn more about Lambda's [retry policy](#) and [DLQs](#). Please ensure your role has appropriate permissions to access the DLQ resource.

- k. Leave everything else as default and click next.
- l. Review the function on the next page and click "Create Function"



1.13 Configure IAM access policy

- a. Navigate to AWS IAM console: <https://console.aws.amazon.com/iam>
- b. Navigate to the Roles list from the left side menu:

The screenshot shows the AWS IAM Management Console interface. The left sidebar is titled 'Services' and has a 'Resource Groups' dropdown. Under 'Resource Groups', the 'Roles' option is selected and highlighted with an orange border. The main content area is titled 'Create New Role' and 'Role Actions'. It includes a 'Filter' input field and a table titled 'Showing 7 results'. The table has two columns: 'Role Name' and 'Creation Time'. The data in the table is as follows:

| Role Name | Creation Time |
|------------------------|----------------------|
| aws_iot_dynamoDB | 2016-06-08 11:02 PST |
| aws_iot_sns | 2016-05-26 13:12 PST |
| dynamoDB_role | 2016-06-09 10:09 PST |
| IAM_Debug | 2016-12-09 14:31 PST |
| LambdaIoT | 2016-12-14 16:09 PST |
| lambda_basic_execution | 2016-06-09 15:00 PST |
| Lambda_IoT_Role | 2016-12-21 13:53 PST |

- c. Select the role policy you created for your Lambda function
- d. Select "Attach Policy"

Tutorial Steps



Summary

Role ARN: arn:aws:iam::089742002813:role/service-role/Lambda_IoT_Role

Instance Profile ARN(s)

Path: /service-role/

Creation Time: 2016-12-21 13:53 PST

Permissions Trust Relationships Access Advisor Revoke Sessions

Managed Policies

The following managed policies are attached to this role. You can attach up to 10 managed policies.

Attach Policy

| Policy Name | Actions |
|---|---|
| AWSLambdaBasicExecutionRole-12e8af73-170e-4bba-928e-9c8fb311c655 | Show Policy Detach Policy Simulate Policy |
| AWSLambdaMicroserviceExecutionRole-17c67a9c-bb2b-4a48-9f9a-cb197e2990b9 | Show Policy Detach Policy Simulate Policy |

e. Select “AWSIoTFullAccess” and click Attach Policy:

Attach Policy

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type AWS Showing 98 results

| Policy Name | Attached Entities | Creation Time | Edited Time |
|--|-------------------|----------------------|----------------------|
| aws-iot-role-logging-18... | 1 | 2016-12-09 14:31 PST | 2016-12-09 14:31 PST |
| <input checked="" type="checkbox"/> AWSIoTFullAccess | 1 | 2015-10-08 08:19 PST | 2015-10-08 08:19 PST |
| AWSLambdaBasicExecu... | 1 | 2016-12-14 16:09 PST | 2016-12-14 16:09 PST |
| AWSLambdaMicroservic... | 1 | 2016-12-14 16:09 PST | 2016-12-14 16:09 PST |
| AmazonEC2RoleforAWS... | 0 | 2015-05-19 11:10 PST | 2015-05-19 11:10 PST |
| AWSAccountActivityAcc... | 0 | 2015-02-06 10:41 PST | 2015-02-06 10:41 PST |
| AWSAccountUsageRepo... | 0 | 2015-02-06 10:41 PST | 2015-02-06 10:41 PST |
| AWSAgentlessDiscovery... | 0 | 2016-08-01 18:35 PST | 2016-08-01 18:35 PST |
| AWSApplicationDiscover... | 0 | 2016-05-11 14:38 PST | 2016-05-11 14:38 PST |

Cancel Attach Policy



1.14 Configure Custom Alexa Skill

Finally, we will create a custom Alexa skill that will pass events to the handler running in the Lambda function.

- a. Navigate to the Alexa developer portal:

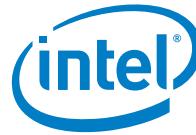
<https://developer.amazon.com/edw/home.html#/skills/list> and select “Get Started” under Alexa Skills Kit

The screenshot shows a web browser window for the Amazon Alexa Skills Kit developer console. The URL is https://developer.amazon.com/edw/home.html#/skills/list. The page title is "Building Alexa Skills with the Alexa Skills Kit". It features a navigation bar with links for DASHBOARD, APPS & SERVICES, ALEXA (which is highlighted in orange), REPORTING, SUPPORT, DOCUMENTATION, and SETTINGS. On the right side of the header, it shows "MARTIN KRONBERG - INTEL" and "SIGN OUT". Below the header, there's a button labeled "Add a New Skill". The main content area displays a table titled "Your skills" with two rows of data. The columns are: Skill Name, Language, Skill Type, Modified, Status, and Actions. The first row shows "gateway_control_skill" in English (U.S.) as a Custom skill modified on 12/19/16 with status "Development" and actions "Edit" and "Delete". The second row shows "SmartHouseExamplePaul" in English (U.S.) as a Custom skill modified on 12/14/16 with status "Development" and actions "Edit" and "Delete". At the bottom of the page, there are links for FAQs, Contact Us, App Distribution Agreement, Trademark Guidelines, Terms of Use, and Job Opportunities.

| Skill Name | Language | Skill Type | Modified | Status | Actions |
|-----------------------|----------------|------------|----------|-------------|-------------|
| gateway_control_skill | English (U.S.) | Custom | 12/19/16 | Development | Edit Delete |
| SmartHouseExamplePaul | English (U.S.) | Custom | 12/14/16 | Development | Edit Delete |

- b. Select “Add a New Skill”. On the following screen add a name for your skill and an invocation name. The invocation name is how you will call the skill from Alexa e.g “Alexa tell gateway to turn on blue LED” where “gateway” is the invocation name.

Tutorial Steps



The screenshot shows the 'Skill Information' step of the Alexa skill creation wizard. The 'Skill Type' section is expanded, showing options for 'Custom Interaction Model' (selected), 'Smart Home Skill API', and 'Flash Briefing Skill API'. The 'Name' field is set to 'Gateway Skill'. The 'Invocation Name' field is set to 'Gateway'. Under 'Global Fields', the 'Audio Player' section is shown with the option 'Does this skill use the audio player directives?' set to 'No'. At the bottom, there are 'Save' and 'Next' buttons.

c. Click "Next"

- d. On the next screen, we will configure the Intent Schema, add custom Slots, and set the utterance. The Intent Schema describes the events that are sent to Lambda and the slots associated with each. Slots can be thought of as variables for the device state, e.g. "blue led off" is a slot that describes the intended state of our gateway. The utterance is what people say to interact with your skill.
- e. In Custom_Skill/Intent_Schema you will find the intent schema that we will use for this project. This schema describes an intent called "DeviceStateIntent" with Slots called "DeviceState" of Type "Device_States". Note that DeviceStateIntent is the event name used in the handler lambda function for switching the state of the gateway – if you rename the event name in the handler function you must also rename it in the intent schema. There are also some default helper intents in the schema that send events to the handler function when the user says things like "help".



The screenshot shows the 'Intent Schema' section of the Alexa Skills Kit developer console. The schema is defined as follows:

```
1 {
  2   "intents": [
  3     {
  4       "intent": "DeviceStateIntent",
  5       "slots": [
  6         {
  7           "name": "DeviceState",
  8           "type": "Device_States"
  9         }
 10      ]
 11    },
 12  ],
 13}
```

Below the schema, there is a 'Custom Slot Types (Optional)' section with a table for defining custom slot types. A single entry is shown:

| Type | Values |
|---------------|--|
| Device_States | all on all off blue led on red led on blue led off red led off |

At the bottom, there is a 'Sample Utterances' section containing the following example phrase:

```
1 DeviceStateIntent turn {DeviceState}
```

- f. Device States is not a standard slot type so we will have to define them. Select Add Slot Type, enter Device States as the type and add the values separated by line, i.e. hit enter between each value. These values are the different cases passes to the DeviceStateIntent handler function. On line 56 of the lambda function you can see **case "all on"**: which defines what the function does when "all on" is passed by the DeviceStateIntent event. We will use "all on", "all off", "blue led on", "blue led off", "red led on", "red led off". You may add your own, but you must define what slot does in the lambda handler function.
- g. Finally add a Sample Utterance. The first term in the sample utterance defines which intent to tie the utterance to, i.e. the user does not say "DeviceStateIntent turn on blue led", only "turn blue led on" and then "blue led on" is passed as the slot for DeviceStateIntent. Any terms in curly braces are slot names defined in the intent schma. Use the simplest form for now "Device StateIntent turn {DeviceState}" **Note:** Alexa is clever enough to substitute "turn" with "switch" and other contextual terms, no need to explicitly define it.
- h. Much more info on intent schema setup can be found here:
[https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/defining-the-voice-interface#The Sample Utterances File](https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/defining-the-voice-interface#The%20Sample%20Utterances%20File)
- i. Click next to continue to the configuration.

Tutorial Steps



The screenshot shows the 'Global Fields' configuration page for an Alexa skill. On the left, there is a sidebar with several sections, each marked with a green checkmark: 'Skill Information', 'Interaction Model', 'Configuration', 'Test', 'Publishing Information', and 'Privacy & Compliance'. The main area is titled 'Global Fields' and contains the following fields:

- Service Endpoint Type:** A radio button is selected for 'AWS Lambda ARN (Amazon Resource Name)', which is described as a serverless compute service that runs code in response to events and automatically manages the underlying compute resources.
- Pick a geographical region that is closest to your target customers:** A radio button is selected for 'North America'.
- Account Linking:** A question asks if users can create or link accounts, with 'Yes' selected.

At the bottom of the page are three buttons: 'Save', 'Submit for Certification', and a large yellow 'Next' button.

- j. Here select AWS Lambda ARN (Amazon Resource Name) as the endpoint type, and enter the ARN found in your Lambda Function Dashboard in the upper right:

The screenshot shows the 'Test_Function' configuration page in the AWS Lambda Management Console. The top navigation bar shows the URL as https://console.aws.amazon.com/lambda/home?region=us-east-1#functions/Test_Function. The page has a sidebar with 'AWS Lambda', 'Dashboard', and 'Functions' options. The main content area is titled 'Lambda > Functions > Test_Function' and shows the ARN: arn:aws:lambda:us-east-1:089742002813:function:Test_Function.

The 'Code' tab is selected. The 'Code entry type' dropdown is set to 'Upload a .ZIP file'. There is a 'Function package*' section with a 'Upload' button and a note about file size. Below that, there is a section for 'Environment variables' with two input fields for 'Key' and 'Value'.



- k. Click next to go to test your new skill:

The screenshot shows the Amazon Appstore Dev console interface. At the top, there are three tabs: 'Amazon Apps & Services Dev', 'Amazon Apps & Services Dev', and 'Defining the Voice Interface'. Below the tabs, there's a search bar with the text 'turn blue led on' and two buttons: 'Ask gateway_control_skill' and 'Reset'. The main area is divided into two sections: 'Lambda Request' and 'Lambda Response'. The 'Lambda Request' section contains the following JSON code:

```
1 {
2     "session": {
3         "sessionId": "SessionId.4f301a1d-2566-483",
4         "application": {
5             "applicationId": "amzn1.ask.skill.899460ba-0a43-41b7-8bf8-5a8...",
6         },
7         "attributes": {},
8         "user": {
9             "userId": "amzn1.ask.account.AHJLJE22ZE",
10        },
11        "new": true
12    },
13    "request": {
14        "type": "IntentRequest",
15        "requestId": "EdwRequestId.2ae78522-98bc-",
16        "locale": "en-US"
17    }
}
```

The 'Lambda Response' section contains the following JSON code:

```
1 {
2     "version": "1.0",
3     "response": {
4         "outputSpeech": {
5             "type": "SSML",
6             "ssml": "<speak> blue L E D on </speak>",
7         },
8         "shouldEndSession": true
9     },
10    "sessionAttributes": {}
11 }
```

At the bottom of the interface, there are two buttons: 'Submit for Certification' and 'Next'. The 'Next' button is highlighted with a yellow border. The footer of the page includes links for 'FAQs', 'Contact Us', 'App Distribution Agreement', 'Trademark Guidelines', 'Terms of Use', and 'Job Opportunities', along with a copyright notice: '© 2010-2016, Amazon.com, Inc. or its affiliates. All Rights Reserved.'

- l. Enter an utterance like “turn blue led on” and click Ask. IF everything is working correctly you should see the lambda request and response with “Turn Blue LED on” as the output speech.



1.15 Run the program

- a. You are now ready to run the program. Launch “Python main.py” from the root folder of the repository.
- b. Wait 15 seconds for everything to initialize.
- c. Click the button, say “Tell gateway to turn blue led on”
- d. The Blue LED should turn on and Alexa will respond with “Blue LED on”

