



Sub 1-GHz Sensor to Cloud Industrial IOT Gateway Reference Design Application Note

Prepared by:
stackArmor
10411 Motor City Drive, Suite 410A
Bethesda MD 20817
Phone: 888-964-1644
solutions@stackarmor.com
www.stackArmor.com

Applicability:
Sub-1 GHz Sensor to Cloud Industrial IoT Gateway
Reference Design
<http://www.ti.com/tool/TIDEP0084>

The information in this document is provided on an as-is basis and no warranty of any kind is expressed or implied. All copyrights and trademarks are acknowledged. The described design and reference implementation is for prototype and non-production demonstration purposes only.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 1.1 | Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Overview | 4 |
| 1.2 | Overview of AWS Cloud and IoT Components..... | 4 |
| 2 | Connecting with AWS IoT Gateway..... | 5 |
| 2.1 | Software Development Kit (SDK) for Devices..... | 10 |
| 2.2 | Thing | 10 |
| 2.3 | Topic..... | 12 |
| 3 | TI Industrial IoT Gateway to Cloud Interface | 13 |
| 3.1 | Message Types | 17 |
| 3.1.1 | Network Information Message Type (From TI IoT Gateway to the Cloud) | 18 |
| 3.1.2 | Device Information Message Type (From TI IoT Gateway to the Cloud) | 18 |
| 3.1.3 | Update Network State Message Type (From Cloud to TI IoT Gateway) | 18 |
| 3.1.4 | Device Actuation Message Type (From Cloud to TI IoT Gateway) | 19 |
| 3.2 | Data Flows and Rules Engine..... | 19 |
| 3.2.1 | Network Information Sent to the Cloud..... | 19 |
| 3.2.2 | Device Information Sent to the Cloud | 19 |
| 3.2.3 | Update Network State Message Sent to the TI IoT Gateway..... | 20 |
| 3.2.4 | Device Actuation Message Sent to the TI IoT Gateway | 21 |
| 4 | Build a TI IOT Solution on AWS..... | 21 |
| 4.1 | Configure AWS IOT Gateway | 22 |
| 4.2 | Setup and Configure BeagleBone Black (BBB) | 22 |
| 5 | References and Resources..... | 22 |

1 Introduction

Texas Instruments (TI) provides a broad spectrum of offerings for enabling Internet of Things (IoT) solutions in the marketplace. TI offers a variety of sensors, processors, microcontrollers and connectivity solutions to meet the diverse needs of various Industrial IoT and other use cases. In order to harvest and analyze IoT data generated from TI networks and sensors, the ability to rapidly connect and process messages at scale is critical. Cloud platforms are well suited to provide a cost-effective and ready-to-go solution for processing and analyzing such IoT data.

This application note enables developers with their own AWS accounts to develop IoT applications using the AWS IoT platform that works with the Sub1-GHz Sensor to Cloud Industrial IoT Gateway Reference Design. . *Developers who have their own AWS account can use this application note as a reference.* This application note is a companion to the Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Reference Design available at <http://www.ti.com/tool/TIDEP0084>. The TIDEP0084 reference design demonstrates how to connect TI IoT sensor and network components to the cloud over a long-range Sub-1 GHz wireless network, suitable for industrial settings such as building control and asset tracking.

Figure 1 below provides a high-level overview of the system based on the Sub 1 GHz IIOT Gateway reference design.

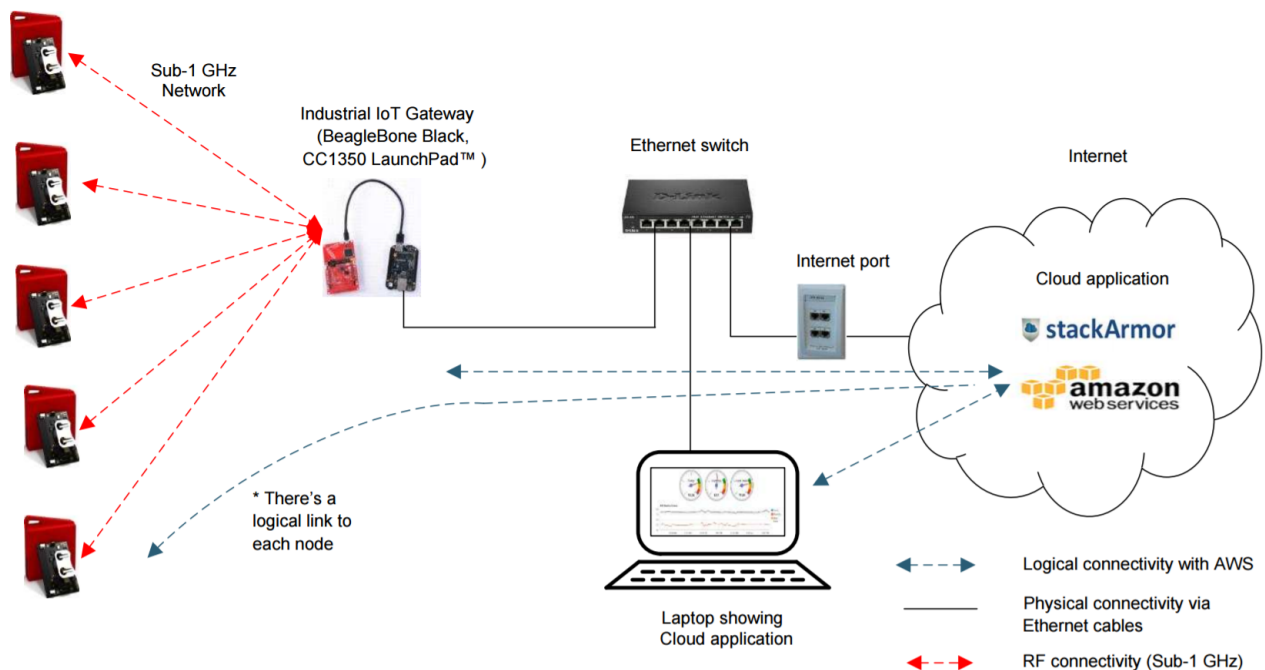


Figure 1: Diagram providing a high-level overview of the Sub 1-GHz IIOT Gateway Reference Design and Architecture

On the left half of the diagram is the Sub1-GHz wireless network, based on the TI 15.4-Stack part of the [SimpleLink CC13x0 SDK](#), with sensor devices transmitting data to the Industrial IoT Gateway using the BeagleBone Black and the CC1310/50 Launchpad. The right half of the diagram depicts the interaction with a cloud-hosted web application using the Amazon Web Services (AWS) cloud platform.

The reference design has been developed to provide developers with a easy to reuse and build-on framework to develop complete sensor-to-cloud solutions that includes the Sub 1-GHz IIOT Gateway solution integrated with the AWS IoT Cloud Gateway. For understanding the TI 15.4-Stack based Sub1-GHz network and sensors design please refer to the Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Reference Design Guide available at <http://www.ti.com/tool/TIDEP0084>. This application note is companion document to the TIDEP0084 design guide document and focuses on the right side of Figure 1 above and describes the AWS IoT Gateway and the cloud application design.

1.1 Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Overview

The Sub-1 GHz Sensor to Cloud Industrial IoT Gateway reference design demonstrates how to connect sensors to the cloud over a long-range Sub-1 GHz wireless network. The Sub-1 GHz wireless network is suitable for industrial settings such as building control and asset tracking. It is powered by a TI Sitara™ AM335x processor and the SimpleLink™ Sub-1 GHz CC1310/CC1350 devices. The reference design pre-integrates the TI 15.4-Stack software development kit (SDK) for Sub-1 GHz star network connectivity and the Linux® TI Processor software development kit (SDK). Key features of the reference design are:

- Large network to cloud connectivity enabling long range, up to 1 km line of sight (LOS)
- IEEE 802.15.4e/g standards based Sub-1 GHz solution with the TI 15.4-stack SDK
- Based on proven hardware designs enabling quick time to market with out-of-the-box ready to use demonstration software
- TI Processor SDK for Linux provides scalability across multiple Sitara processors such as AM437x and AM57x

The reference design includes connecting to a cloud platform for the purposes of displaying and interacting with the Network and Devices. The subsequent sections provide an overview of the Amazon cloud IoT platform and its various components.

1.2 Overview of AWS Cloud and IoT Components

TI Design Network partner stackArmor implemented the cloud application services for cloud connectivity and visualization of the sensor node data hosted on the Amazon Web Services (AWS) cloud platform. The sensor data is collected and analyzed in a web application dashboard hosted on the Amazon Elastic Cloud Computing (EC2) platform and the AWS IoT Gateway. Key features of the cloud application reference design include:

- Ability to connect with multiple networks and display data rapidly
- Ability for bi-directional data exchange: from sensor to cloud for displaying sensor information as well as sending actuation messages to control the network from the cloud-based web application
- Enables implementing scalable and light-weight cloud-based web application using modern components and open source libraries such as AngularJS, Node,JS and NoSQL databases

Amazon Web Services (AWS) IoT is a cloud service with all components necessary for pushing data from the BeagleBone Black board to the AWS hosted application and vice versa. Figure 2 below provides an overview of the AWS based application developed by stackArmor.

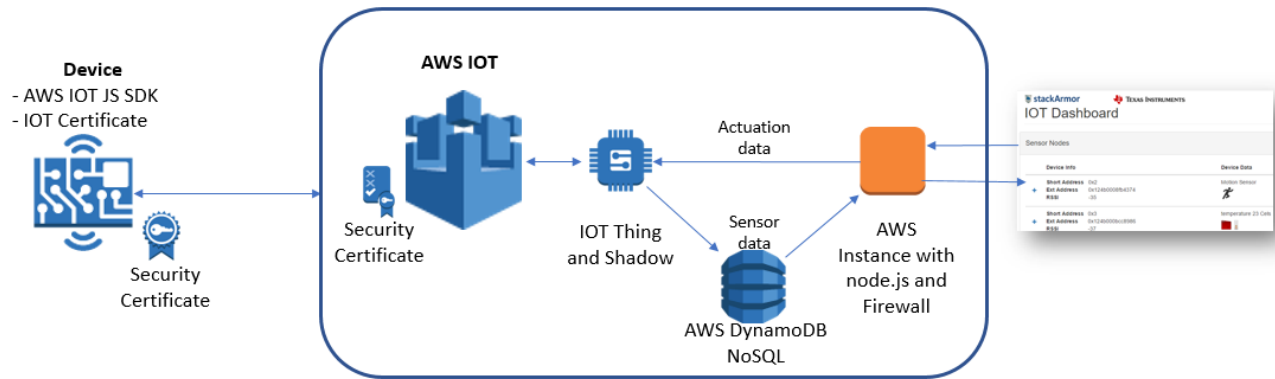


Figure 2: High-level architecture of AWS cloud based web application for collecting and displaying IoT data

The communication link between the Device and the AWS Cloud is established using the AWS IoT service. The AWS IoT service provides a AWS IoT Device Software Development Kit (SDK) which helps to easily and quickly connect the device to the AWS IoT and includes various AWS IoT specific features. This SDK is included in the example/reference application released as part of the TI Sub1-GHz Sensor to Cloud Reference design which runs on the BeagleBone Black [here](#). In addition to the AWS IoT Device SDK, the security certificate are required which allow the AWS IoT cloud service to receive and accept encrypted messages. The Node.js based AWS IoT Device SDK was selected to help write the code to provide the necessary commands for connecting and sending as well as receiving MQTT messages from the AWS IoT cloud service.

The data from the sensors is collected and processed by the AWS IoT Gateway and stored in a NoSQL database service called AWS DynamoDB. The web application is hosted on a AWS cloud instance using a Node.js instance and a NGINX proxy for security purposes.. Subsequent sections in this document help describe the steps necessary to configure and setup the AWS IoT Gateway services as well as some of the associated data processing and web hosting components.

2 Connecting with AWS IoT Gateway

In order to get started with the AWS IoT platform, the developer must have an AWS Account. The account can be obtained by navigating to <https://aws.amazon.com/console> and entering the requested information including credit card information. Once, you are logged into the AWS Console, then you must click on the AWS IoT link which will navigate you to the AWS IoT Dashboard. Figure 3 below displays a screenshot of the AWS IoT Dashboard along with the various actions on the left side.

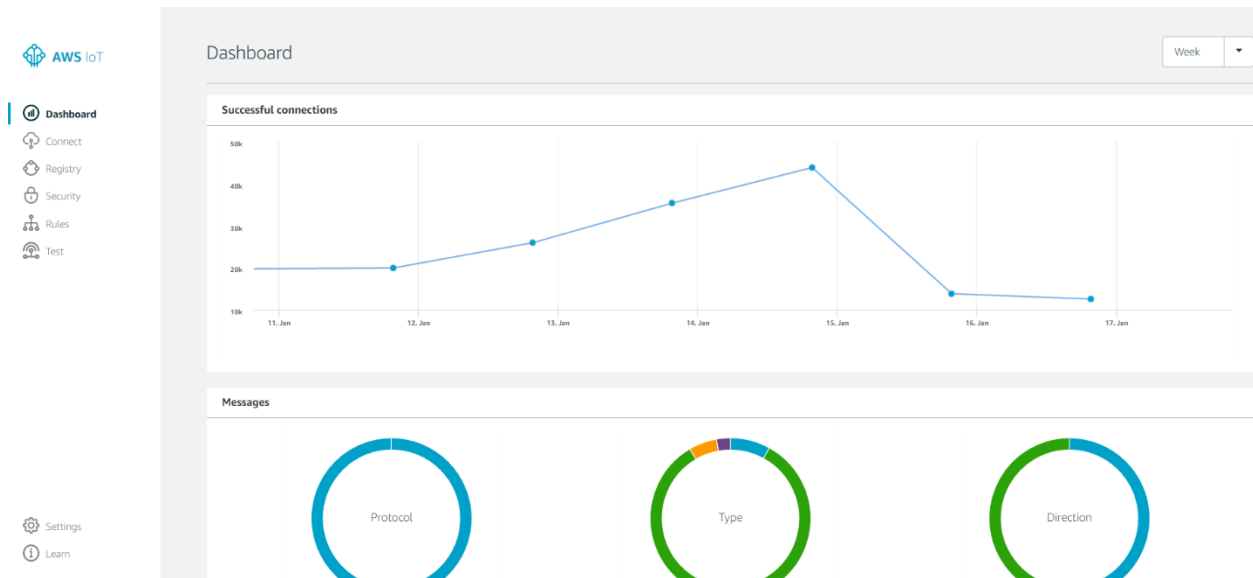


Figure 3: AWS IoT Dashboard screenshot displayed once you log into the console and view the AWS IoT service.

The AWS IoT Dashboard provides the initial entry point towards configuring and setting up the cloud service to receive and transmit IoT messages. Once in the AWS IoT dashboard, click on the **Connect** link on the left. This will navigate you to a screen similar to the one below shown in Figure 4 below with choices on the Thing to connect with.

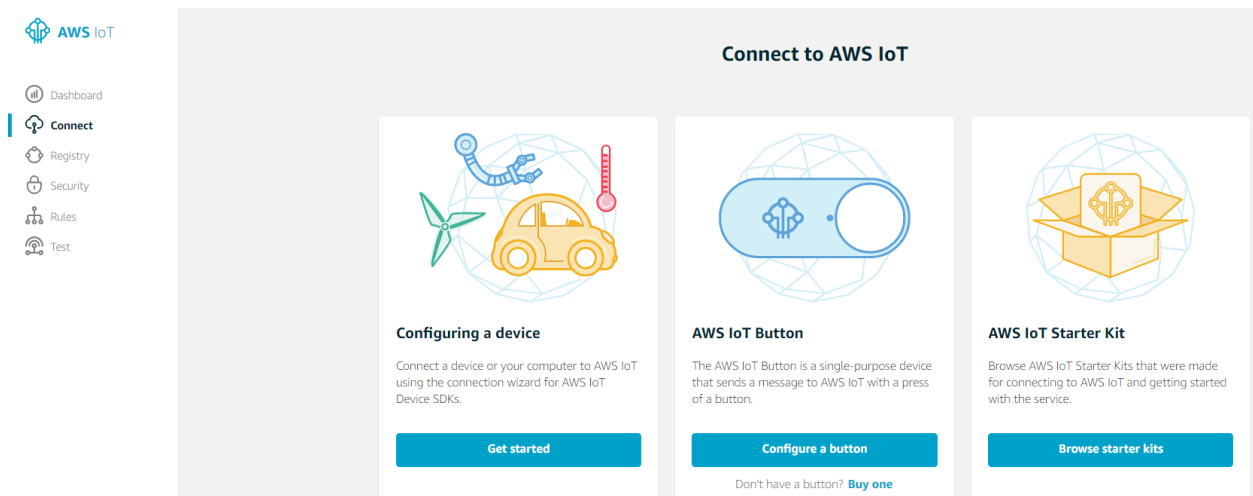


Figure 4: AWS IoT Connect screenshot for configuring connection to a Thing with a Get started Wizard

Select “Get Started” in the “Configuring a device” selection and select the SDK as shown in the screenshot below in Figure 5.

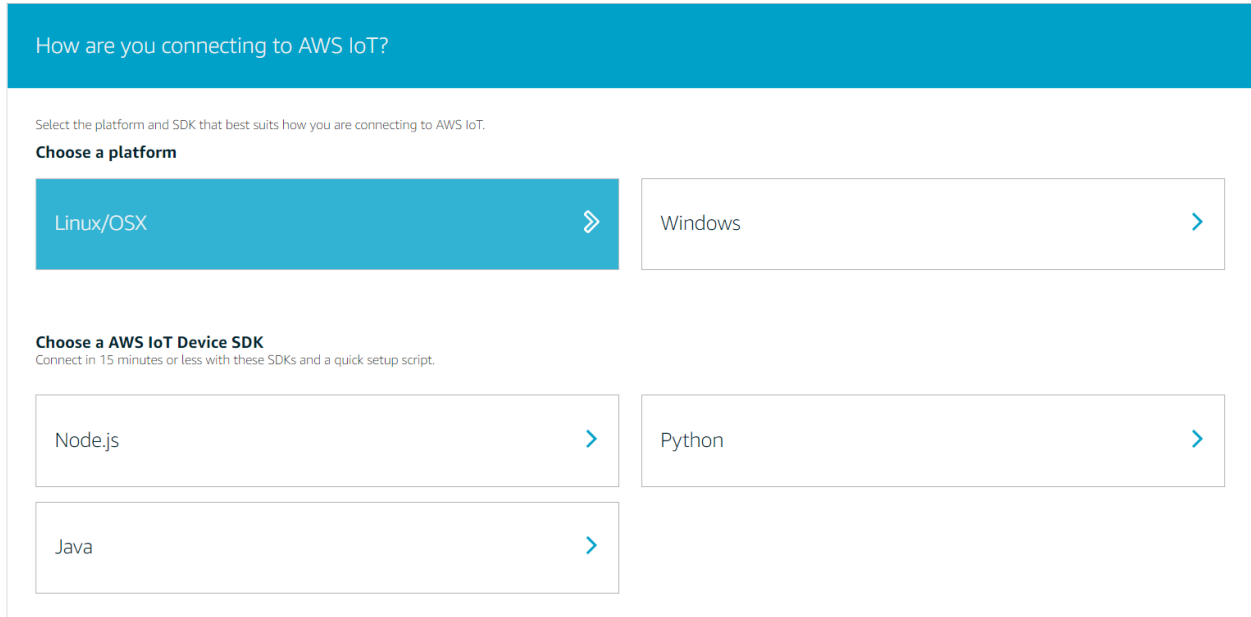


Figure 5: AWS IoT SDK selection wizard for finding the right library to use for connecting the AWS IoT cloud service

Select the Linux operating system and the language - Node.js and then a wizard screen like the one shown below in Figure 6 will be presented.

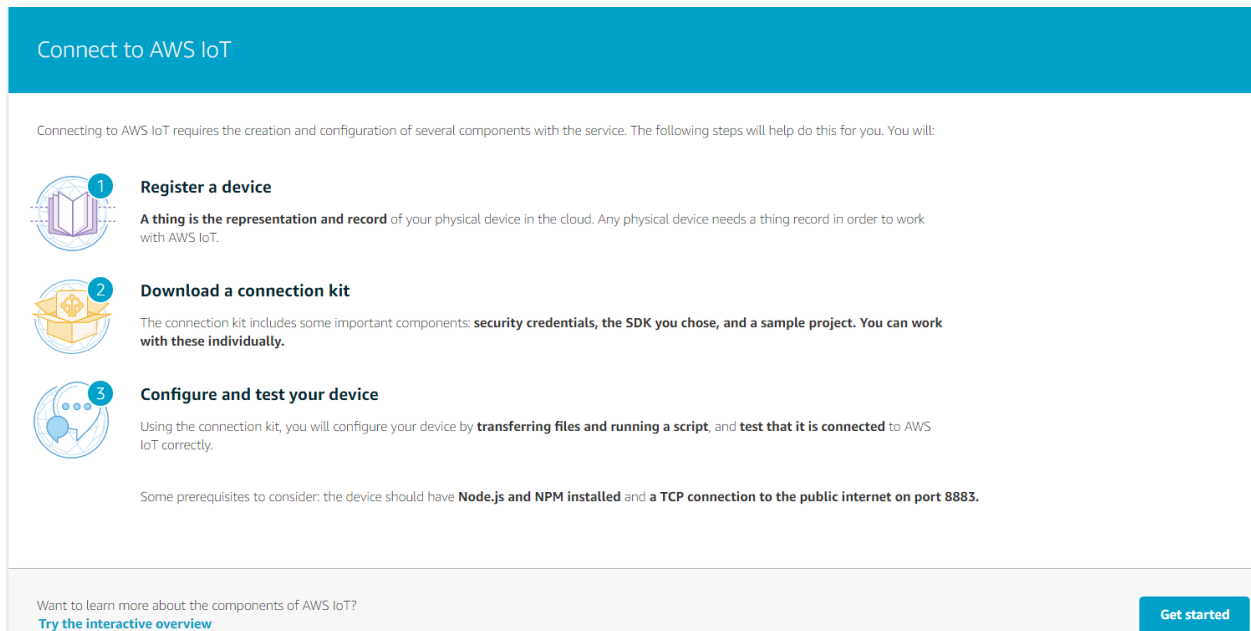


Figure 6: AWS IoT connection wizard for registering and connecting the cloud service

Follow through on executing the three step wizard by clicking on the **Get started** button, which will present a screen similar to the one below in Figure 7.

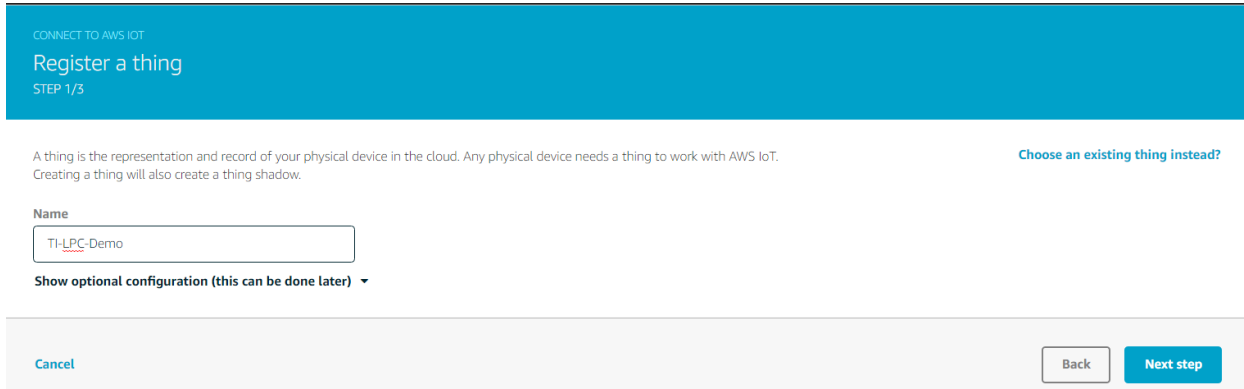


Figure 7: AWS IoT connection wizard Step 1 for registering and connecting the cloud service

Enter the name of the Thing you would like to connect to the AWS IoT gateway and then click on the **Next step** button. Once the name has been entered you must download the security certificate and access credentials to connect to the AWS IoT Gateway.

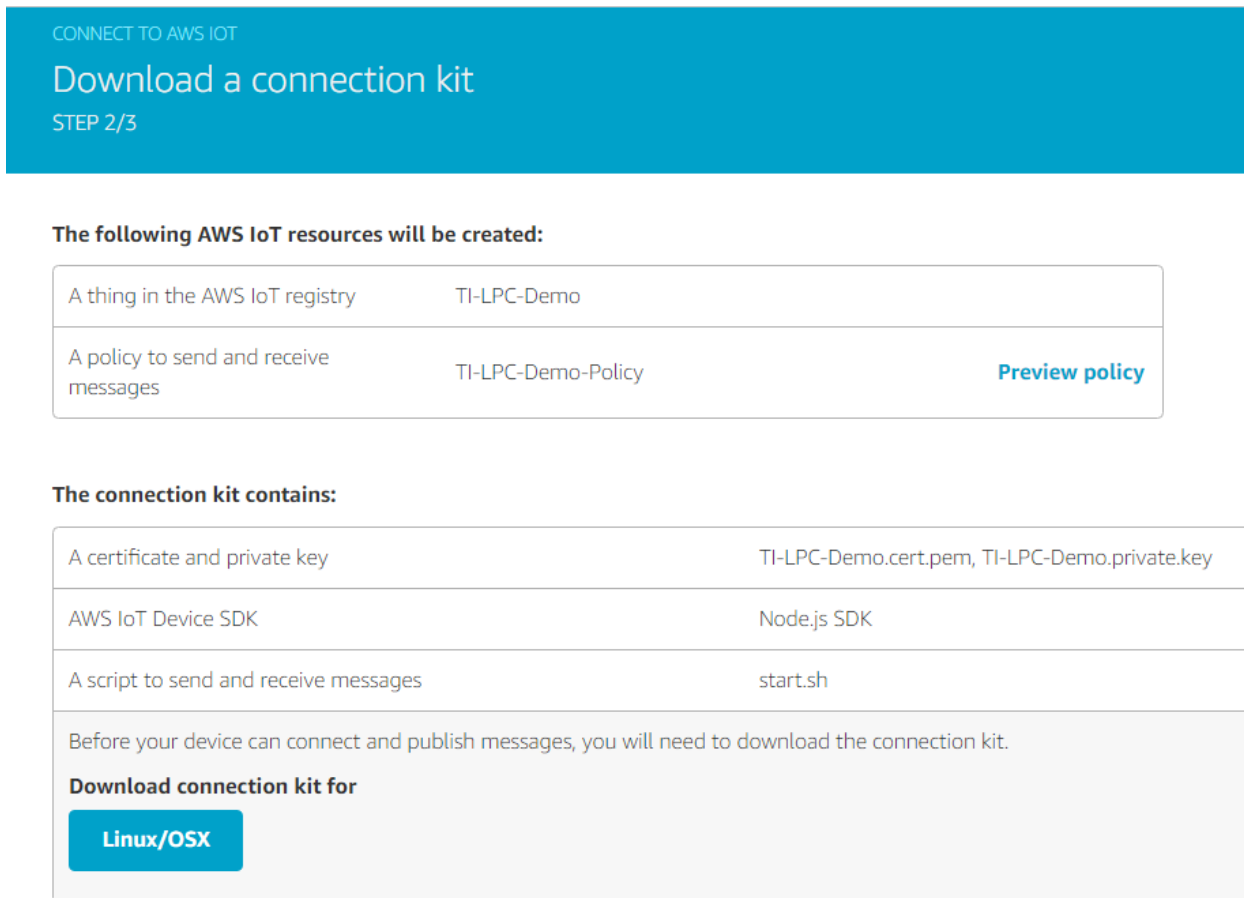


Figure 8: AWS IoT connection wizard Step 2 for registering and connecting the cloud service

A key part to establishing the connection between the Device and the AWS IoT Gateway is to load the certificate and ensure that the Device can be authenticated. The files are available for download by clicking

on the Linux/OSX button. The policy associated with the Device on the AWS IoT can be viewed by clicking on the Preview policy link.

Once you are ready to download and install the certificate files on your Device then follow the instructions as stated in the final step of the wizard as shown in the screenshot below in Figure 9.

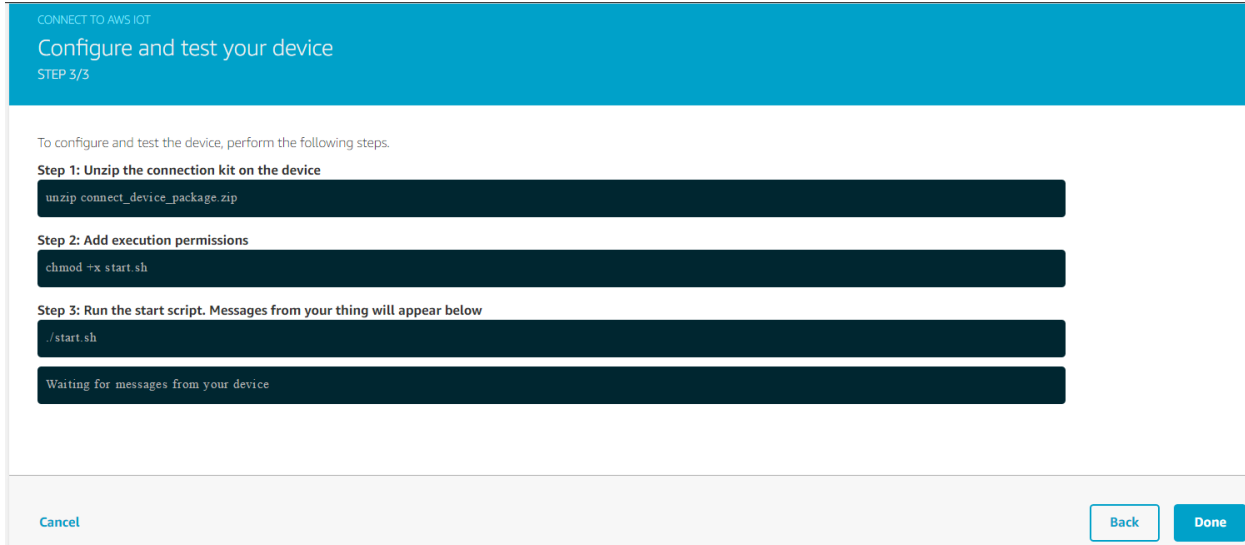


Figure 9: AWS IoT connection wizard Step 3 for connecting and testing the device and communicating with the cloud service

Once you are done with the basic copying and configuration of the Device you have completed all of the basic steps to establish connectivity with the AWS IoT service and will be presented with a screen as shown in Figure 10 below

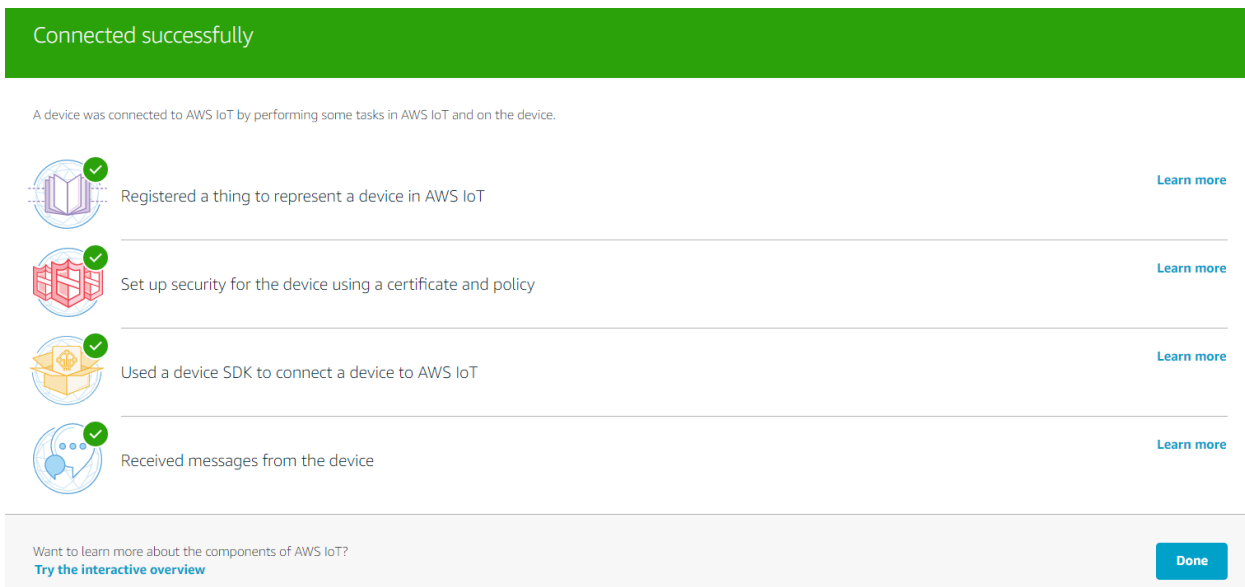


Figure 10: AWS IoT connection wizard showing completion of all steps for connecting the Device with the cloud

The AWS IoT service will then navigate you to the Device registry that keeps track of your registered devices as shown in Figure 11 below.

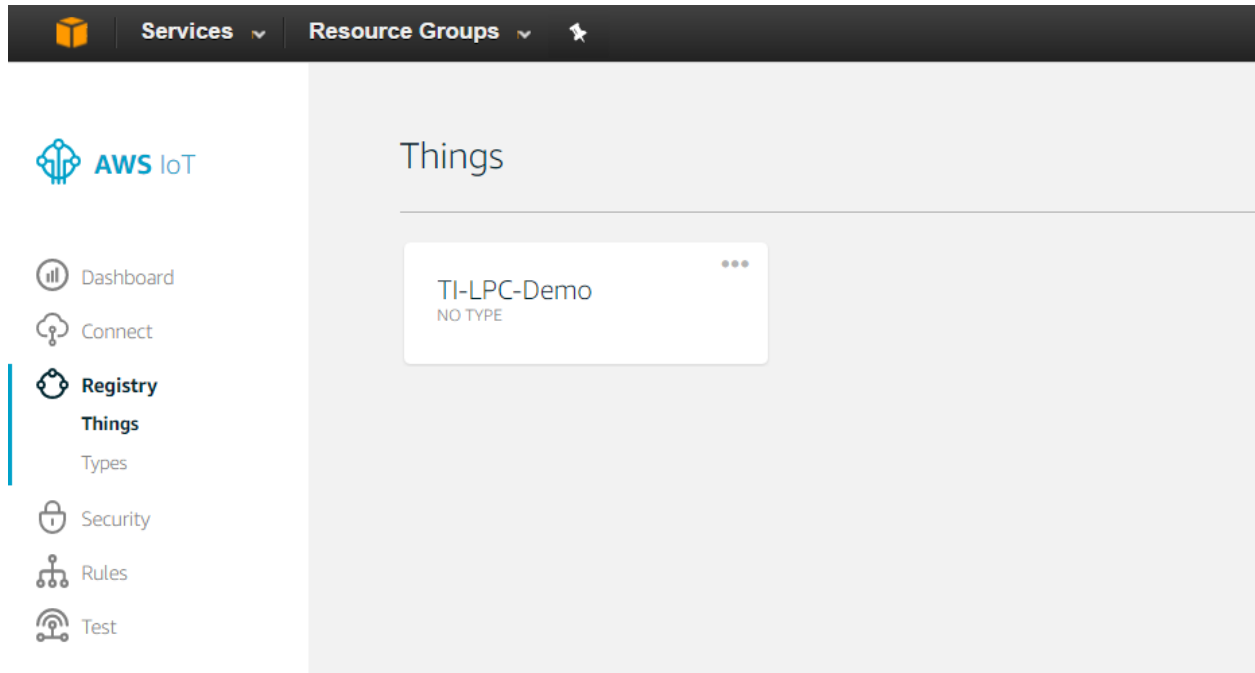


Figure 11: AWS IoT Device Registry with all of the Things in use

2.1 Software Development Kit (SDK) for Devices

The first step towards connecting to the AWS IoT Gateway is to ensure the selection of the appropriate Software Development Kit (SDK) to interact with the Board/Device. For the purposes of the initial demonstration application developed as part of this Application Note, the Node.js SDK was used. The SDK along with the certificate and key files must be copied on to the Board/Device to allow for connection with the AWS IoT Gateway.

2.2 Thing

Once the basic connectivity between the Device and AWS IoT Gateway is established. One can learn more about the Thing settings by clicking on the **Registry** menu item and the **Things** sub-menu item. And then select the specific **Thing** of interest. A screen like the one shown in Figure 12 below will be displayed. Every Thing is identified using a unique Amazon Resource Number (ARN).

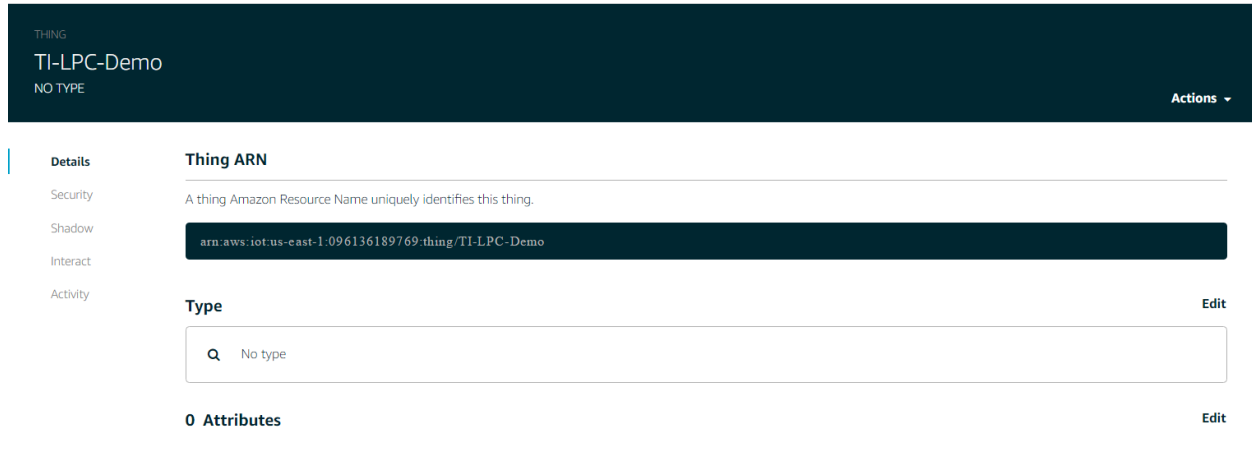


Figure 12: AWS IoT Thing screenshot showing the various parameters and attributes

The various selections on the sub-menu include Security, Shadow, Interact and Activity. The Security sub-menu allows you to review or create certificates associated with the Thing. The Shadow is the virtual representation of the Thing and allows for interacting with the physical Thing even when it is offline. The AWS IoT service will transmit the messages to the Device once it is available from the Shadow. The Shadow has a unique ARN. The Interact sub-menu provides all of the connectivity and interaction information for exchanging messages between the physical Device and the Thing representation on the cloud platform. Each AWS IoT Gateway has a unique URL which is shown on the screen below in Figure 13. One has the ability to utilize multiple connection protocols e.g. HTTPS or MQTT. The default port for HTTPS is 443 whereas the port for MQTT is 8883. The reference design used the MQTT protocol and hence the 8883 port was utilized.

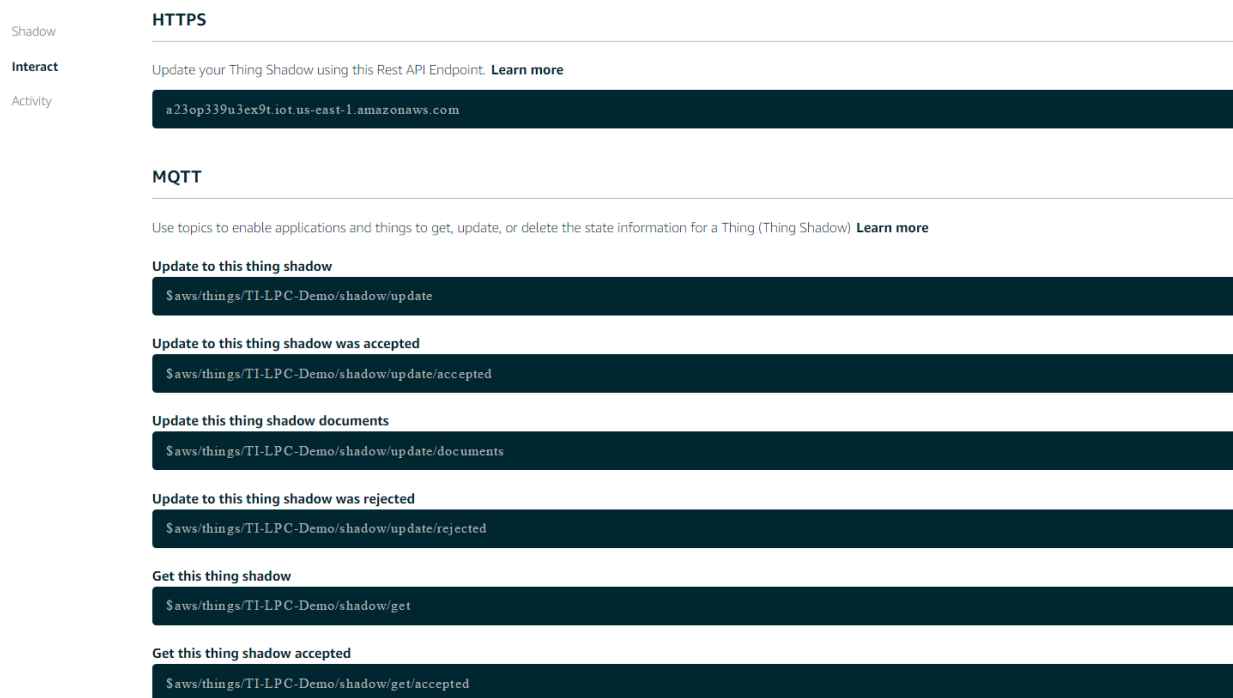


Figure 13: AWS IoT Screenshot of Thing Interact sub-menu with information on how to connect and exchange messages

The interaction between the Device and the Thing (Device representation in the cloud) is an event based queuing service that uses topics to help drive actions. The terminology associated with queuing systems such as topics, subscriptions and publishing applies to interacting with the MQTT protocol and the Device. The Thing Shadows service uses reserved MQTT topics to enable applications and things to get, update, or delete the state information for a thing (thing shadow). The names of these topics start with `$aws/things/thingName/shadow`. Each Shadow has JSON documents associated with it often referred to as the payload.

2.3 Topic

The Thing Shadows service uses reserved topics to enable applications and things to get, update, or delete the state information for a thing (thing shadow). The names of these topics start with `$aws/things/thingName/shadow`. Publishing and subscribing on thing shadow topics requires topic-based authorization. It is a recommended best practice to avoid wild card subscriptions to shadow topics. The following are the MQTT topics used for interacting with thing shadows. The Thing Shadows services acts as an intermediary, allowing devices and applications to retrieve and update thing shadows.

The Thing Shadows service uses a number of MQTT topics to facilitate communication between applications and devices. To see how this works, use the AWS IoT MQTT client to subscribe to the following MQTT topics:

`$aws/things/TI-LPC-Demo/shadow/update/accepted`

The Thing Shadows service sends messages to this topic when an update is successfully made to the thing shadow.

`$aws/things/TI-LPC-Demo/shadow/update/rejected`

The Thing Shadows service sends messages to this topic when an update to the thing shadow is rejected.

`$aws/things/TI-LPC-Demo/shadow/update/delta`

The Thing Shadows service sends messages to this topic when a difference is detected between the reported and desired sections of the thing shadow.

`$aws/things/TI-LPC-Demo/shadow/get/accepted`

The Thing Shadows service sends messages to this topic when a request for the thing shadow is made successfully.

`$aws/things/TI-LPC-Demo/shadow/get/rejected`

The Thing Shadows service sends messages to this topic when a request for the thing shadow is rejected.

`$aws/things/TI-LPC-Demo/shadow/delete/accepted`

The Thing Shadows service sends messages to this topic when the thing shadow is deleted.

`$aws/things/TI-LPC-Demo/shadow/delete/rejected`

The Thing Shadows service sends messages to this topic when a request to delete the thing shadow is rejected.

`$aws/things/TI-LPC-Demo/shadow/update/documents`

The Thing Shadows service publishes a state document to this topic whenever an update to the thing shadow is successfully performed.

When a device generates an event e.g. a sensor comes online, it sends its current state to the Thing Shadows service by sending an MQTT message to the `$aws/things/TI-LPC-Demo/shadow/update` topic.

If the thing shadow doesn't exist, it is created. Otherwise, the thing shadow is updated with the data in the message. If you don't see a message published to `$aws/things/TI-LPC-Demo/shadow/update/accepted`, check the subscription to `$aws/things/TI-LPC-Demo/shadow/update/rejected` to see any error messages.

Messages are published to the `/update/documents` topic whenever an update to the thing shadow is successfully performed.

3 TI Industrial IoT Gateway to Cloud Interface

The Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Reference Design application was developed using the Amazon Web Services (AWS) IoT cloud service with all components necessary for pushing data from the BeagleBone Black board to the AWS hosted application and vice versa. The diagram in Figure 14 below provides an overview of the AWS based application developed by stackArmor.

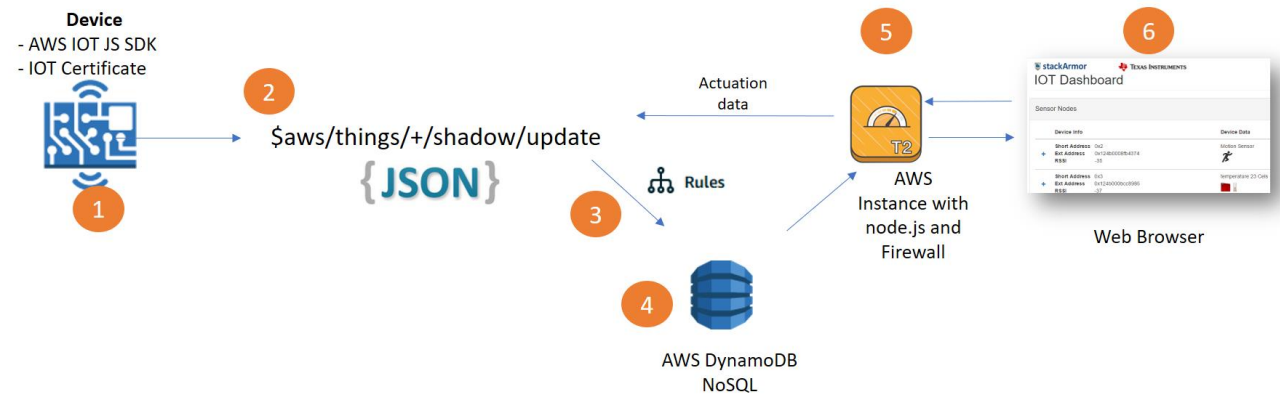


Figure 14: Reference Application Design High-level Cloud Architecture Diagram for the Sub 1 GHz IIOT Gateway with a network of nodes using the BeagleBone Black

The communication link between the Network/Device and the AWS Cloud is established using the AWS IoT service. Key components and flows are described at a high level below.

1. As described earlier, the AWS IoT service provides a Software Development Kit (SDK) that must be installed on the BeagleBone Black Device along with the security certificate to allow the AWS IoT cloud service to receive and accept messages. The Node.js SDK was used for the reference design.
2. A message is transmitted between the Network/Device and the AWS IoT Gateway through the concept of a Topic, which has a JSON object with the data values from the Device that need to be processed and displayed. The developed solution is bi-directional in that it transmits sensor data from the device to the

cloud as well as transmits actuation signals to change the state of the device or sensors from the cloud web application.

3. The AWS IoT Rules Engine detects the incoming message published to the Topic and executes an instruction to store the JSON object into a database. The screenshot in Figure 15 below shows the configuration of the logic that directs the topic to execute an action to insert a record into the AWS DynamoDB database service.

The screenshot displays the configuration of an AWS IoT Gateway Rule. It is divided into three main sections: Overview, Description, and Actions.

- Overview:** Shows the rule name 'sensor_update2'.
- Description:** States 'This is similar to the sensor_update rule but it adds an additional timestamp column to allow for serversite reporting.'
- Rule query statement:** Contains the SQL query: `SELECT * FROM '$aws/things/+shadow/update where endswith(topic(3),'network') = false'`. Below the query, it specifies 'Using SQL version 2016-03-23'.
- Actions:** Shows a single action: 'Insert a message into a DynamoDB table' with the table name 'sensor_update2'. An 'Add action' button is visible below the action list.

Figure 15: Screenshot of AWS IoT Gateway Rules Engine with logic to store incoming data objects into a AWS DynamoDB database record

There are a number of actions that can be configured and triggered to deliver specific outcomes.

4. The DynamoDB service stores the incoming messages in the format transmitted by the device for further processing and historical storage. The AWS DynamoDB service is a fully managed NoSQL database that stores un-structured data such as JSON objects. The screenshot in Figure 16 below shows the 4 tables designed for the reference application.

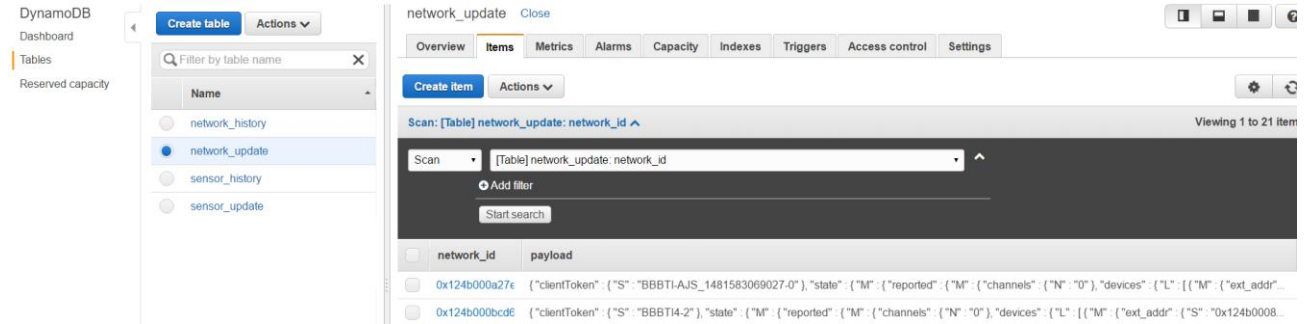


Figure 16: Screenshot of AWS DynamoDB with 4 tables and actual data messages from networks and devices

5. A web server hosted on a AWS server instance includes a web server and a NGINX firewall/proxy to respond to requests for data from the client browser. The web server responds to requests for data and the dashboard application.

6. The user types the URL of the dashboard with the unique assigned network_ID. The dashboard application service is displayed in the web browser. The screenshot in Figure 17 below shows the access URL for the development dashboard web application.

Network Information

| | |
|--------------|--|
| PanID | 0xacdc |
| Coord Addr | 0x1234 |
| Network Mode | Non Beacon |
| Security | 1 |
| Network | <input checked="" type="checkbox"/> On |

Network Chart

Sensor Nodes

| Device Info | | Device Data | |
|-------------|---------------|------------------|---------------------|
| + | Short Address | 0x1 | temperature 25 Cels |
| | Ext Address | 0x124b000bcd3b86 | |
| | RSSI | -54 | |
| + | Short Address | 0x2 | temperature 29 Cels |
| | Ext Address | 0x124b0008fb14c9 | |
| | RSSI | -23 | |

Figure 17: Screenshot of web application dashboard that displays the collected data and allows for sending actuation messages using the toggle buttons.

The web application has been developed using various open source component libraries such as Angular.JS and Bootstrap. That query the underlying AWS DynamoDB database to collect and display the data in the dashboard.

3.1 Message Types

The reference application displays primarily two types of messages 1) network information and 2) device data. The actual sensor data with recorded values for are embedded in the device data. The screenshot in Figure 18 below shows the key data elements.

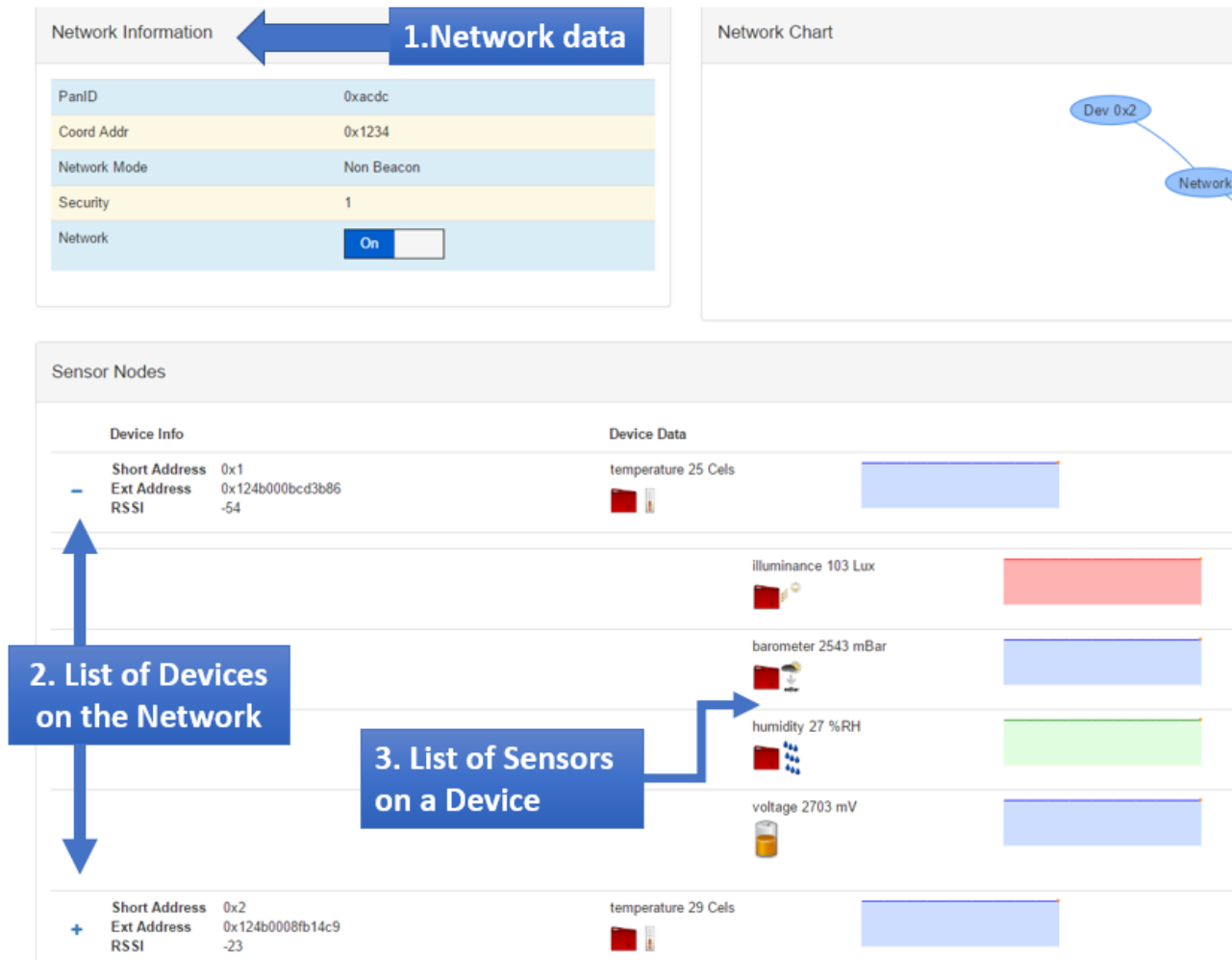


Figure 18: Screenshot of sample application developed for displaying Network data, Devices and Sensor data from each Device.

The data in the dashboard is displayed by retrieving the data from the AWS DynamoDB database. Four tables, or data stores, were created to store incoming messages from the Network and Devices respectively. All of the messages are stored in the AWS DynamoDB service as JSON documents.

network_update: This table stores the latest incoming message from all Network's attached to the AWS IoT Gateway. At any given point in time, only the latest incoming message for a given Network is stored.

network_history: This table stores a historical record of all Network related messages along with the timestamp.

device_update: This table stores the latest incoming message from all Device's attached to the AWS IoT Gateway. At any given point in time, only the latest incoming message for a given Device is stored.

device_history: This table stores a historical record of all Device related messages along with the timestamp.

The database stores the messages as transmitted by the network and devices. The actual format of the messages and attributes are documented in Chapter 5 and Chapter 6 of the Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Reference Design document available at <http://www.ti.com/lit/pdf/tiduci9>.

3.1.1 Network Information Message Type (From TI IoT Gateway to the Cloud)

The Network message is a JSON document that is identified by a unique Network ID and all incoming messages are stored in the AWS cloud platform using the AWS DynamoDB service which is a NoSQL data store.

The specific attributes contained in the message are shown in the image extracted from the TI Design Guide Chapter 5.1.1. The list of devices provides the nodes within the network. The web application displays a graphic of a network node with multiple device nodes based on data contained in this message.

- **name**: begins as the short address of the network but allows for the cloud to provide a more specific name
- **channels**: list of channels that the wireless network is operating on
- **pan_id**: the 16-bit PAN identifier of the network
- **short_addr**: the 16-bit short address of the pan-coordinator
- **ext_addr**: the 64-bit IEEE extended address of the pan-coordinator device
- **security_enabled**: 'yes' if security enabled, 'no' otherwise
- **mode**: network operation mode (beacon/non-beacon/frequency hopping)
- **state**: pan-coordinator state values (waiting/starting/restoring/started/open/closed)
- **devices**: list of wireless nodes in the network
 - **name**: begins as the short address of the device but allow cloud to update
 - **short_addr**: the 16-bit short address of the pan-coordinator
 - **ext_addr**: the 64-bit IEEE extended address of the pan-coordinator device
 - **topic**: the topic that the device will send its sensor data updates to
 - **object_list**: list of IPSO Alliance Smart Objects (sensors) attached to this device
 - **oid**: object ID which specifies the sensor type in the IPSO standard
 - **iid**: list of instance IDs for the current object (can be multiple same type sensors)

3.1.2 Device Information Message Type (From TI IoT Gateway to the Cloud)

The Device is identified by a unique Device ID and all incoming messages are stored in the AWS cloud platform using AWS DynamoDB. The specific attributes within a device are described in the TI Design Guide Chapter 5.1.2. This message type provides information about the wireless device as well as the latest data for all of the sensors connected to the device. This message type will be sent when a device reports sensor data or switches between an active/inactive state.

- **active**: whether or not the wireless node is active
- **ext_addr**: the 64-bit IEEE extended address of the pan-coordinator device
- **rssi**: received signal strength indicator of the last message received
- **smart_objects**: list of the IPSO Alliance Smart Objects connected to this wireless device
 - **object ID description**: type of sensor (as defined in the IPSO standard). Can be multiple types of sensors connected to each device
 - **instance ID**: the instance ID for the parent object type. Can be multiple sensors of the same type
 - **resource ID description list**: sensor data name value pairs (e.g. "sensorValue": 32.5, "units": "Celsius", etc.). These resources match what is specified for the given object ID in the IPSO standard

3.1.3 Update Network State Message Type (From Cloud to TI IoT Gateway)

The update network message is created in response to the user's command to open or close the wireless network to new devices joining. The web application provides a button to allow the opening or closing of the network that triggers a message with the state attribute

- **state**: should be set to either 'open' or 'closed'

3.1.4 Device Actuation Message Type (From Cloud to TI IoT Gateway)

The reference implementation is bi-directional that allows users to trigger changes on the network and devices as well as display data. The Device actuation message currently supports toggling an LED on the wireless device's board. The Device Actuation Message is sent from the web application to the AWS IoT topic associated with the specific network_ID and device_ID which then is transmitted to the device. The following field is the only requirement for this message: toggleLED: should be set to 'true'

3.2 Data Flows and Rules Engine

This section provides an overview of the actual logic used to collect and process data coming from the network and device as well as from the web application to the network and device respectively.

3.2.1 Network Information Sent to the Cloud

The data transmitted from the network to the AWS IoT Gateway is published on a topic. The actual data transmitted from the Network is described in TI Design Guide Chapter 6.1.1. The current implementation accepts all incoming messages from the various networks and devices. The actual topic logic used to pull and store the data is provided below.

```
SELECT *, state.reported.ext_addr as state.reported.ext_addr FROM '$aws/things/+ /shadow/update' where endswith(topic(3), "network") = true
```

The logic uses a wild-card "+" after the /things/ to pull all incoming data. The actual network identifier is stored in a separate column in the database for later retrieval. The data is stored as an item within the AWS DynamoDB service. The screenshot below in Figure 19 shows how the data is captured and stored.

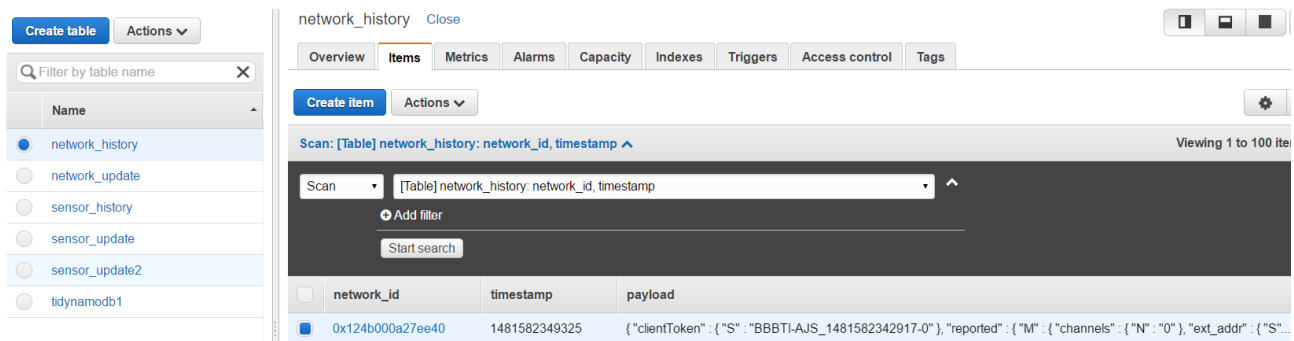


Figure 19: Screenshot of AWS DynamoDB item with data received from a Network connected to the AWS IoT Gateway

3.2.2 Device Information Sent to the Cloud

The data transmitted from the device to the AWS IoT Gateway is published on a topic. The current implementation accepts all incoming messages from the various devices and by using a logic filter the network and device messages are stored in different database tables as described earlier. The actual topic logic used to pull and store the data is provided below.

```
SELECT * FROM '$aws/things/+ /shadow/update' where endswith(topic(3), "network") = false
```

The logic uses a wild-card "+" after the /things/ to pull all incoming device data. The actual device identifier is stored in a separate column in the database for later retrieval. The logic value in the where clause

`endswith(topic(3), "network") = false` helps identify and differentiate device messages from network messages. The screenshot below in Figure 20 shows an item in the AWS DynamoDB table for sensor data and the associated columns for sensor identifier, timestamp and the actual message data received.

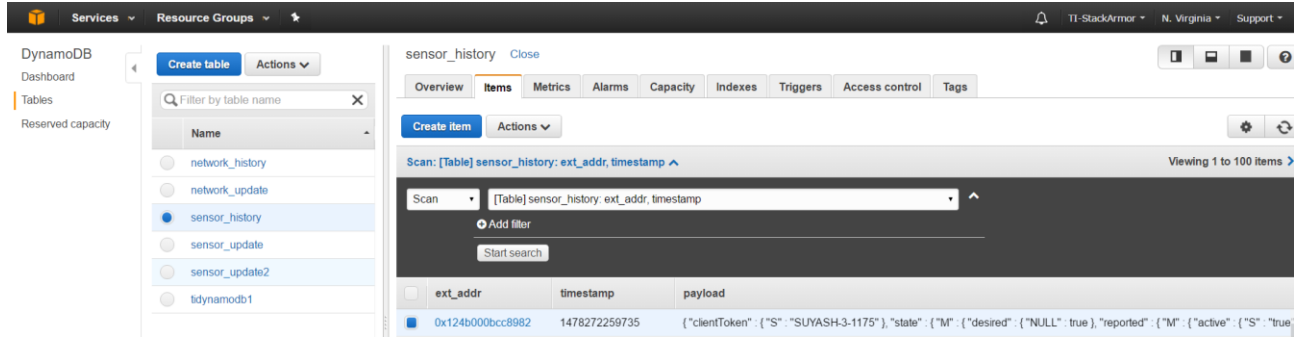


Figure 20: Screenshot of AWS DynamoDB item with data received from a Sensor device connected to the AWS IoT Gateway

3.2.3 Update Network State Message Sent to the TI IoT Gateway

The reference design web application provides bi-directional data flows and allows for messages to be transmitted from the cloud to the network or sensor device. The screenshot below in Figure 21 shows the toggle/on-off buttons in the application. When the user clicks on the buttons, a message is transmitted to the network.

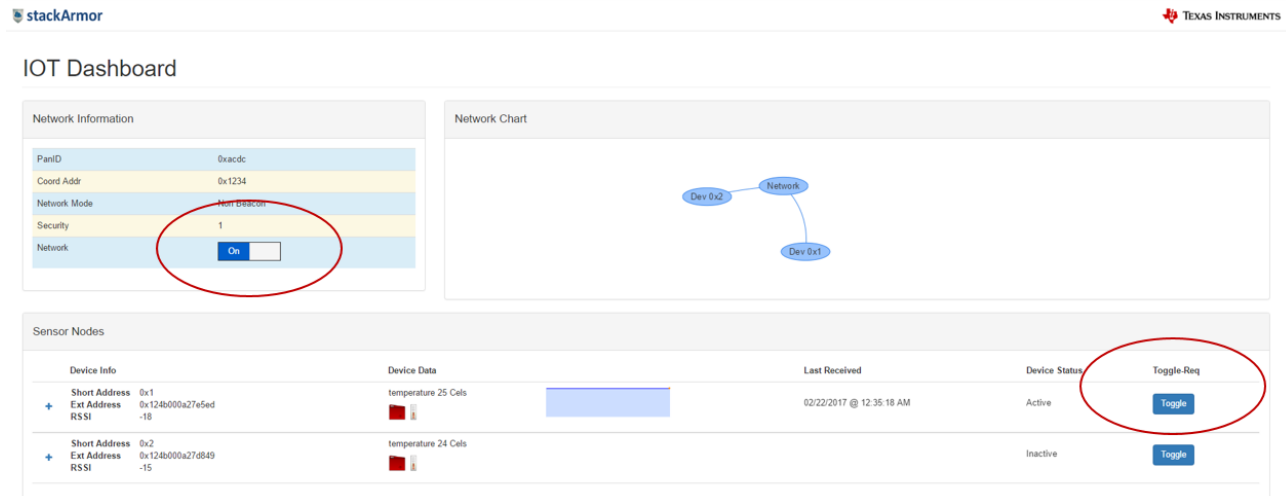


Figure 21: Screenshot of web application with the ability to transmit actuation signals from the cloud platform to the IoT device and network.

The web application directly communicates with the AWS IoT Gateway using the following parameters:

```
"host": "xxxxxxxxxxxxx.iot.us-east-1.amazonaws.com",
"port": 8883,
"clientId": "BBBTI2",
```

The actual message is published as \$aws/things/ti_iot_' + net_addr + "_network/shadow/update topic with the new state value.

3.2.4 Device Actuation Message Sent to the TI IoT Gateway

Similar to the Network actuation message, the web application directly communicates with the AWS IoT Gateway to manage the sensor device by publishing a message to the update topic as shown below.

\$aws/things/ti_iot_' + net_addr + '_' + dev_addr + '/shadow/update with the new state value. This message in turn is transmitted and read by the Sensor device that triggers a change in state.

4 Build a TI IOT Solution on AWS

In this section we summarize and integrate all of the steps necessary to implement the Sub-1 GHz Sensor to Cloud Industrial IoT Gateway Reference Design. For all Board and Device related information please refer to the associated design note <http://www.ti.com/tool/TIDEP0084>

In order to use the AWS IoT Gateway, you must setup an AWS Account. You can open an account by going to the AWS website at <https://aws.amazon.com> and clicking on the button in the top right corner “Sign in to the Console”. The screenshot as shown below in Figure 22 provides a view of the page.

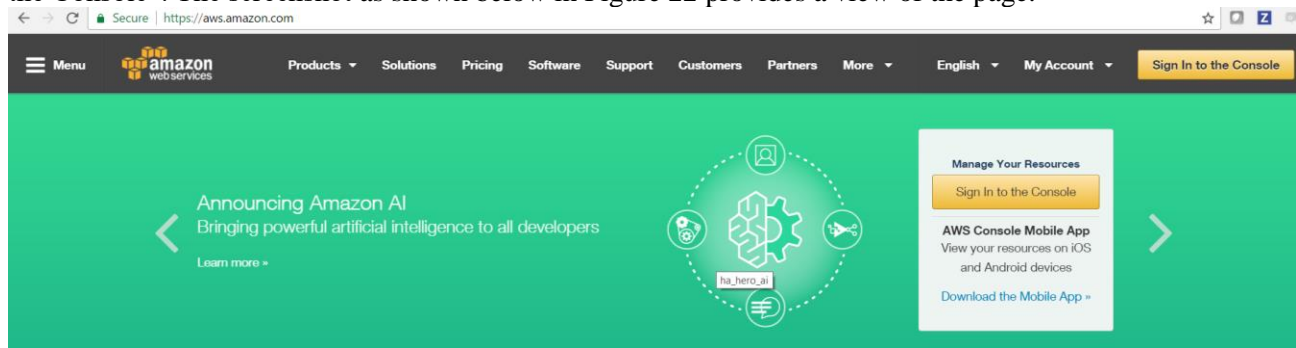


Figure 22: Screenshot of AWS Home page to begin opening an account.

Once you navigate to the console page. Select the “I am a new user” and follow the prompts to begin the account setup process. Alternatively, you can contact AWS Partners like stackArmor (<https://www.stackArmor.com>) who can help orchestrate this process for you.




Sign In or Create an AWS Account

What is your email (phone for mobile accounts)?

E-mail or mobile number:

- I am a new user.
- I am a returning user and my password is:

Sign in using our secure server 

[Forgot your password?](#)



Learn more about [AWS Identity and Access Management](#) and [AWS Multi-Factor Authentication](#), features that provide additional security for your AWS Account. View full [AWS Free Usage Tier](#) offer terms.

4.1 Configure AWS IoT Gateway

Once the AWS Account is setup. You are now ready to configure your AWS IoT Gateway as described in the earlier sections of this application note. The key steps are the creation of the AWS IoT Gateway end-point for the Thing with the URL and the port. Also, the certificates to be downloaded and installed on the actual device.

4.2 Setup and Configure BeagleBone Black (BBB)

Follow the instructions as described in the Sub 1-GHz IIOT Reference Design Application Note on how to download and install the SDK, the certificates and the URL for the AWS IoT Gateway.

5 References and Resources

| Resource | Description | Link |
|----------|--|---|
| 1 | TIDN Sub 1-GHz IIOT Gateway Reference Design | http://www.ti.com/tool/TIDEP0084 |
| 2 | AWS IoT Documentation | https://aws.amazon.com/documentation/iot/ |
| 3 | AWS IoT Developer Guide | http://docs.aws.amazon.com/iot/latest/developerguide/iot-dg.pdf |

Note:

This document has been prepared by stackArmor, an AWS Partner and TI Designs Network partner. stackArmor with competency in IoT solutions design and development and you can learn more about stackArmor by visiting the [TIDN page](#) at the following link:

<http://www.ti.com/devnet/docs/catalog/companyfolder.tsp?actionPerformed=companyFolder&companyId=14898> or visiting <https://www.stackArmor.com/IOT>

The reference design and architecture described in this document was developed for a demonstration application and not for production deployments. The selected components, security architecture with respect to certificates and web application design may change based on specific deployment needs and message processing volumes. The information in this document is provided on an as-is basis and no warranty of any kind is expressed or implied. All copyrights and trademarks are acknowledged.