Examples BLE AWS

With Amazon AWS Greengrass

This guide shows how to build application examples which involve Bluetooth Low Energy (BLE) devices implementing the "BlueST" protocol that connect to a Linux gateway, and to make them communicate to the Amazon AWS IoT Cloud through the AWS Greengrass edge computing service.

The Greengrass edge computing service allows to perform local computation of Lambda functions with the same logic available on the cloud even when the connection to the cloud is missing; moreover, as soon as the connection becomes available the shadow devices on the cloud get automatically synchronized to the local virtual devices.

The application examples here described are provided within the "EdgeSTSDK" Python SDK, and implements the following scenario:

- "example_ble_aws_1.py": Two BLE devices exporting the "Switch" feature as specified by the "BlueST" protocol; pressing the user button on a device makes the LED of the other device toggle its status;
- "example_ble_aws_2.py": The same as "example_ble_aws_1.py", plus the two BLE devices that export environmental and inertial features, so that data from Pressure, Humidity, Temperature, Accelerometer, Gyroscope, and Magnetometer sensors are sent to the IoT Cloud.

Davide Aliprandi – System Research and Applications - STMicroelectronics S.r.l.

Table of Contents

Requirements

- Amazon Developer account
- Raspberry PI3 with an 8GB SD card and:
    - "Stretch" Raspbian OS
- Two Bluetooth Low Energy IoT nodes made up of:
    - NUCLEO-F401RE MCU board
    - X-NUCLEO-IDB05A1 Bluetooth Low Energy expansion board
    - X-NUCLEO-IKS01A2 MEMS Environmental and Inertial expansion board

Amazon AWS Greengrass

Accessing Greengrass Edge-Computing Service

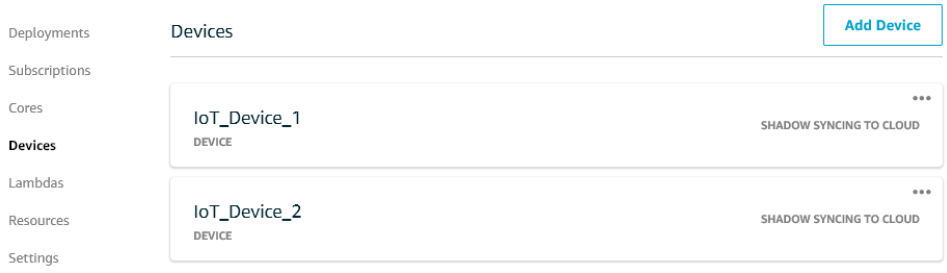
- Sign in to your Amazon AWS account at the following webpage:
  - https://aws.amazon.com/
- Click the "Sign In To the Console" button.
- From the "Services" page, look for the "AWS Greengrass" service.
- Select from the top-right drop-down menu' a region where Greengrass is available, for example "Frankfurt" / "eu-central-1" or "Oregon" / "us-west-2".
- Select the "Greengrass" item from the left menu.
- Enter into the "Groups" section and click the "Get Started" button.

Creating a Group

- Click on "Create Group", then "Use easy creation".
- Name your group, for example as "GG_Group_1".
- Click "Next", then "Create Group and Core".
- Download the security certificate and key by clicking on "Download these resources as a tar.gz". They will have to be copied on your core device later.
- Download the Greengrass SDK by selecting the desired target platform (e.g. "ARMv7l" if you are using, for example, a Raspberry PI3) and clicking on "Download Greengrass version X.Y.Z"; it will be installed later.
- Click "Finish".

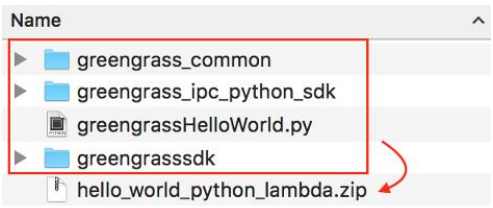
Creating Devices

- Enter the "GG_Group_1" group, select "Devices", "Add Device", "Create New Device", and create the devices "IoT_Device_1" and "IoT_Device_2"; select "Use Defaults".
- Download certificates and keys of the devices by clicking on "Download these resources as a tar.gz", and the Certification Authority as "root.ca.pem" by clicking "Download a root CA". They will have to be copied on your core device later.
- Enter the "GG_Group_1" group, select "Devices", and for each device switch from "Local Shadow Only" to "Sync to the Cloud" capability.
- You should see a list of devices like the following:

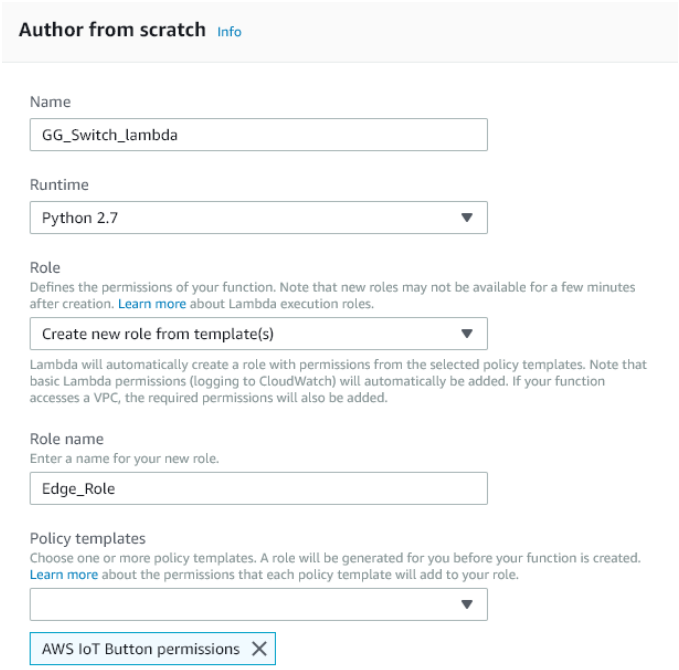


Creating a Greengrass Lambda Function

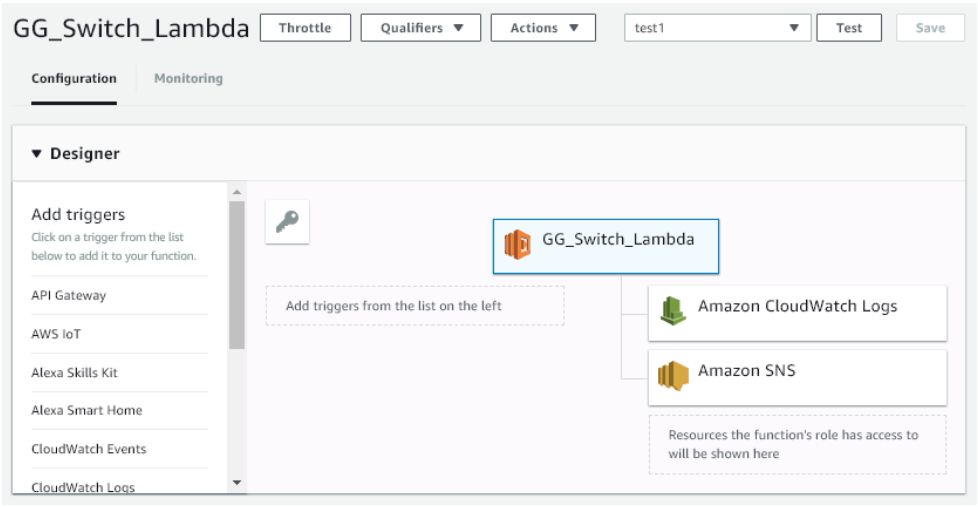
- In the AWS IoT console, choose "Software", download the Python AWS Greengrass core SDK to your computer ("greengrass-core-python-sdk-1.1.0.tar"), and decompress it.
- Enter the "aws_greengrass_core_sdk\examples\HelloWorld" folder and decompress the "greengrassHelloWorld.zip" file into a "greengrassHelloWorld" folder.
- Rename the "GG_Switch_Lambda.py" file to "greengrassHelloWorld.py" and copy it into the "greengrassHelloWorld" folder.
- Compress the following content into a "hello_world_python_lambda.zip" file:



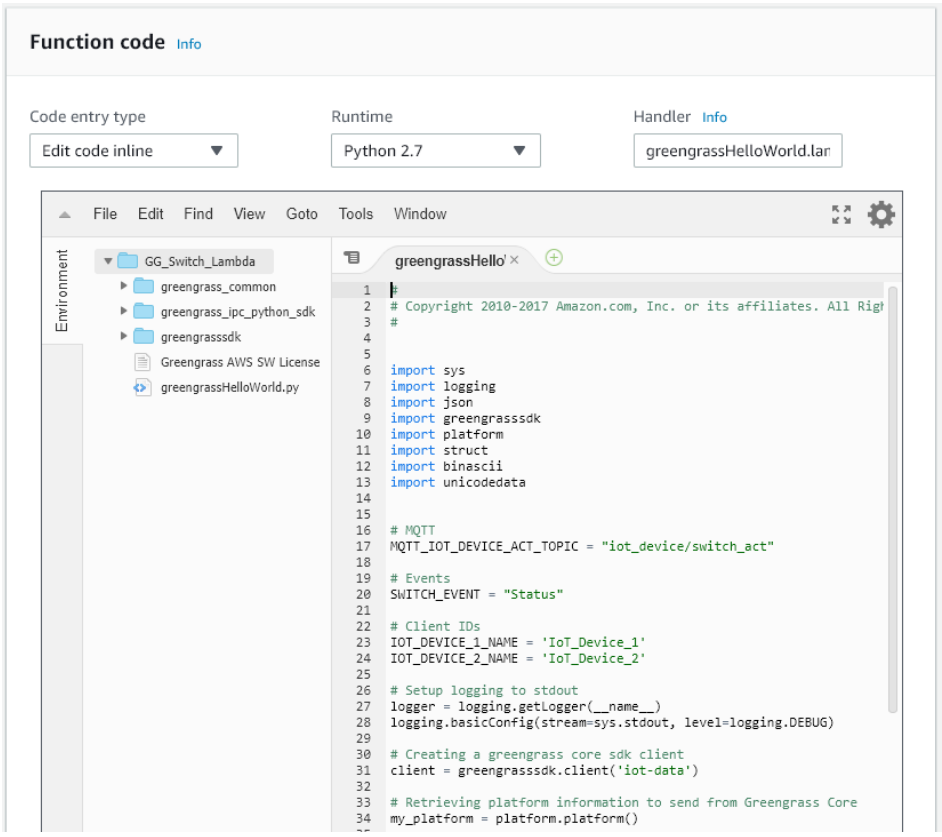
- From the "GG_Group_1" management page, select "Lambdas" from the left menu, "Add lambda", and "Create new lambda".
- Within the "Author from scratch" panel create a lambda function named "GG_Switch_Lambda" that use the "Python 2.7" runtime.
- Under "Role" select "Create a role from template", put "Edge_Role" under "Role name", then select "AWS IoT Button permissions" under "Policy templates", and "Create function" to confirm.
- You should see a control page like the following:



- You will end up with an editing page like the following:



- On the "Configuration" tab, in the "Function code" region, under the "Code entry type" drop-down menu, choose "Upload a .ZIP file". For Runtime, choose Python 2.7. Type "greengrassHelloWorld.function_handler" into the "Handler" field. Upload the "hello_world_python_lambda.zip" file you created earlier by clicking on "Upload", then on "Save".
- You should see an editing page like the following:



- Ensure that the name of devices within the lambda function is the same as the things created previously.
- Then under “Actions” click on “Publish new version”, with a new version, e.g. “V1”.
- Return on the AWS IoT console and enter the “GG_Group_1” Greengrass group, select “Lambdas”, “Add Lambda”, then “Use existing Lambda”. Select the “GG_Switch_Lambda” lambda just created, click “Next”, select the desired version, and click on “Finish”.
- This is the context where the Lambda function just created works:
    1. Whenever the user button is pressed on a device, the sending device publishes a JSON message on a “sense” topic with its device identifier and the status of the button;
    2. The lambda function swaps the device identifier and publishes the new message on an “act” topic;
    3. The recipient device toggles the status of its LED.

Creating Subscriptions

- Enter the "GG_Group_1" Greengrass group, select "Subscriptions" from the left menu, click on "Add Subscription", and create the following custom subscriptions to manage the "Switch" feature:

FROM	TO	TOPIC
IoT_Device_1	GG_Switch_Lambda:1	iot_device/switch_sense
IoT_Device_2	GG_Switch_Lambda:1	iot_device/switch_sense
GG_Switch_Lambda:1	IoT_Device_1	iot_device/switch_act
GG_Switch_Lambda:1	IoT_Device_2	iot_device/switch_act

- Just for the example "example ble aws 2.py", create the following custom subscriptions to manage the other features, that allow the devices to send environmental and inertial data to the IoT Cloud:

FROM	TO	TOPIC
IoT_Device_1	IoT Cloud	iot_device/env_ine_sense
IoT_Device_2	IoT Cloud	iot_device/env_ine_sense

- Then, for each device, create the following system subscriptions, that are needed to synchronize the status of the shadow devices on the Cloud to the status of the devices on the Edge (when using the Edge capabilities):

FROM	TO	TOPIC
<Device>	Local Shadow Service	$aws/things/<Device>/shadow/update
<Device>	Local Shadow Service	$aws/things/<Device>/shadow/get
<Device>	Local Shadow Service	$aws/things/<Device>/shadow/delete
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/update/accepted
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/update/rejected
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/update/delta
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/get/accepted
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/get/rejected
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/delete/accepted
Local Shadow Service	<Device>	$aws/things/<Device>/shadow/delete/rejected

Substitute <Device> with the proper device name, for each device of the system. To save some time, you can insert, for example, "$aws/things/<Device>/shadow/update/#" with the "#" wildcard, to cover all the "update"'s related subscriptions, and so on for the "get" and "delete" topics.

Raspberry PI3

Setting up the Raspberry PI3

- Please see the official documentation to set up your Raspberry PI3.
- From a PC/MAC flash the latest "Raspbian OS" release on a fresh SD card. At the time of writing the following release has been used:
  - "Stretch" Raspbian OS
  - "Linux raspberrypi 4.9.59-v7+ #1047" kernel
- Enable the SSH service temporarily:
  - `touch /<volume>/boot/ssh`
- Insert the SD card into your Raspberry PI3.
- Login:
  - `ssh pi@raspberrypi.local`
- Run "raspi-config" and:
  - Enable the SSH service permanently.
  - Set the correct timezone.
- Update the system by running the following instructions:
  - `sudo rpi-update`
  - `sudo apt-get install --reinstall raspberrypi-bootloader raspberrypi-kernel sqlite3`
  - `sudo apt-get update && sudo apt-get upgrade`
- Install the required packages:
  - `sudo apt-get install python-pip libglib2.0-dev vim git`

Setting up Amazon AWS Greengrass SDK

- Please see the official documentation to install the Amazon AWS Greengrass SDK. At the time of writing the instructions are as follows.
  - https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html
- Install the Amazon AWS IoT Python SDK, required by the EdgeST SDK:
  - `sudo pip install AWSIoTPythonSDK`
- Add user and group to the system:
  - `sudo adduser --system ggc_user`
  - `sudo addgroup --system ggc_group`
- Enable hardlink and softlink protection at operating system start-up by adding the following two lines to the end of the "/etc/sysctl.d/98-rpi.conf" file:
  - `fs.protected_hardlinks = 1`
  - `fs.protected_symlinks = 1`

- Reboot the PI, connect to it through SSH, and then run the following commands to confirm the hardlink/symlink change:
  - `sudo sysctl -a 2> /dev/null | grep fs.protected`
- You should see "fs.protected_hardlinks = 1" and "fs.protected_symlinks = 1".
- Append the following configuration parameter to the command line within the "/boot/cmdline.txt" file, and then reboot the PI3:
  - `cgroup_memory=1`
- Copy the "Greengrass-linux-armv7l-1.3.0.tar.gz" file downloaded when creating a group into the root of the PI3.
- Install Amazon AWS IoT Greengrass SDK:
  - `sudo tar -xzvf greengrass-platform-version.tar.gz -C /`
- Install the Symantec VeriSign root Certification Authority "root.ca.pem" downloaded when creating the devices. This certificate enables your device to communicate with AWS IoT using the MQTT messaging protocol over TLS. Copy it onto the PI3 at the folder:
  - `/greengrass/certs`
- Configure Amazon AWS IoT Greengrass SDK by editing the following file and filling in with the correct parameters:
  - `/greengrass/config/config.json`

  Please note that you are required to put your <iot_host_prefix> and <region> parameters within the file, that you can find in the AWS IoT console under "Settings". Ensure also the core's name is correct, e.g.: "GG_Group_1_Core", as well as the core's certificate and private key filenames.

  Example "config.json" file:

```
{
    "coreThing": {
        "caPath": "root.ca.pem",
        "certPath" : "GG_Group_1_Core.pem",
        "keyPath" : "GG_Group_1_Core.prv",
        "thingArn": "arn:aws:iot:<region>:<Account-ID/alias>:thing/
                     GG_Group_1_Core",
        "iotHost": "<iot_host_prefix>.iot.<region>.amazonaws.com",
        "ggHost": "greengrass.iot.<region>.amazonaws.com"
    },
    "runtime": {
        "cgroup": {
            "useSystemd": "yes"
        }
    }
}
```

- Further information can be found here:
  - https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-device-start.html#config.json-params

- Copy and rename the core's certificate and keys downloaded when creating the group into the "/greengrass/certs/" folder:
    - `GG_Group_1_Core.pem`
    - `GG_Group_1_Core.prv`
    - `root.ca.pem`

  Ensure the core's certificate and private key filenames are the same as specified within the "config.json" file.
- Create a "devices_ble_aws" folder into the PI3 "/home/pi/" folder, and copy and rename the certificates and the private keys downloaded when creating the devices into it:
    - `IoT_Device_1.pem`
    - `IoT_Device_1.prv`
    - `IoT_Device_2.pem`
    - `IoT_Device_2.prv`

Setting up BlueST and EdgeST SDKs

- Please see the official documentation of the SDKs:
    - https://github.com/STMicroelectronics-CentralLabs/BlueSTSDK_Python
    - https://github.com/STMicroelectronics-CentralLabs/EdgeSTSDK_Python
- Clone the Git repository of the SDKs onto the PI3 "/home/pi" home folder:
    - `git clone https://github.com/STMicroelectronics-CentralLabs/BlueSTSDK_Python.git`
    - `git clone https://github.com/STMicroelectronics-CentralLabs/EdgeSTSDK_Python.git`
- Install the bluepy Python library, required by the BlueST SDK:
    - `sudo pip install bluepy`
- Install the concurrent.futures Python module (and any other missing module), required by the BlueST SDK:
    - `sudo pip install futures`

Setting up the application example(s)

- Edit the selected application example(s) ("example_ble_aws_1.py", "example_ble_aws_2.py", etc…) you can find within the "/home/pi/EdgeSTSDK_Python/edge_st_examples/aws/" folder, and set the "IOT_DEVICE_X_NAME" and "IOT_DEVICE_X_MAC" global variables properly:
    - Name of the devices created on the Cloud;
    - You can retrieve MAC addresses through a smartphone application, provided that the firmware is already installed on the devices (see Chapter "Bluetooth Low Energy Devices").

Bluetooth Low Energy Devices

Source code

- The example applications require different firmware on the BLE devices:
    - "example_ble_aws_1.py":
        - https://os.mbed.com/teams/ST/code/Node_BLE_Switch_Device/
    - "example_ble_aws_2.py":
        - https://os.mbed.com/teams/ST/code/Node_BLE_Sensors_Device/

  In fact, the second firmware works for both applications, as it exports all the features.
- Import the selected mbed OS application to your ARM mbed account, compile, and flash it onto the two BLE devices.

Deploying Greengrass on the core device

Starting the Greengrass daemon

- Login to the Raspberry PI.
- Enter the PI3 "/home/pi" home folder:
  - `cd /home/pi`
- Restart the Greengrass daemon:
  - `sudo /greengrass/ggc/core/greengrassd restart`

Performing a Greengrass deployment

- Sign in to your Amazon AWS account at the following webpage:
  - https://aws.amazon.com/
- Deploy the system by clicking on "Deployments" from the left panel, then "Deploy".
- Select "Automatic", "Grant permission", and wait for the deployment's green status.

Running the application example(s)

Running the main Python script

- Login to the Raspberry PI.
- Enter the PI3 "/home/pi" home folder:
  - `cd /home/pi`
- Become super user (to use the Bluetooth peripheral), set the Python path environmental variable, and restart the Greengrass daemon:
  - `sudo su`
  - `export PYTHONPATH=/home/pi/BlueSTSDK_Python/:/home/pi/EdgeSTSDK_Python/`
  - `/greengrass/ggc/core/greengrassd restart`
- Run the application example by passing the endpoint and the path to the root Certification Authority certificate:
  - `python ./EdgeSTSDK_Python/edge_st_examples/aws/example_ble_aws_X.py -e <iot_host_prefix>.iot.<region>.amazonaws.com -r /greengrass/certs/root.ca.pem`
- For both examples:
  - Press a BLE device's user button to toggle the other device's LED status.
- For "example_ble_aws_2.py":
  - You can see sensors' data on the Amazon's "Test" page ("Amazon AWS IoT Console", "Services", "IoT Core", "Test"), by subscribing to the "iot_device/env_ine_sense" topic.
- Press "CTRL+C" to stop the application.