Example BLE AWS 1

With Amazon AWS Greengrass

This application example shows how to connect Bluetooth Low Energy (BLE) devices implementing the "BlueST" protocol 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.

This application example involves 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. In particular, whenever the user button is pressed on a device, the sending device publishes a JSON message on a "*switch_device/sense*" topic with its device identifier and the status of the button, a simple lambda function swaps the device identifier and publishes the new message on a "*switch_device/act*" topic, and the recipient device toggles the status of its LED.

Davide Aliprandi - SW Platforms and Cloud - Central Labs - STMicroelectronics S.r.l.

Table of Contents

Requirements

- Amazon Developer account.
- Raspberry PI3 with an 8GB SD card and:
    - "Stretch" Raspbian OS
    - "Linux RaspberryPi 4.9.59-v7+ #1047" kernel
- Two Bluetooth Low Energy IoT nodes made up of:
    - NUCLEO-F401RE MCU board
    - X-NUCLEO-IDB05A1 Bluetooth Low Energy expansion board

Amazon AWS Greengrass (Edge Computing)

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 "Oregon" / "West-2" region (top right page corner).
- 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 your security certificates and keys 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 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:
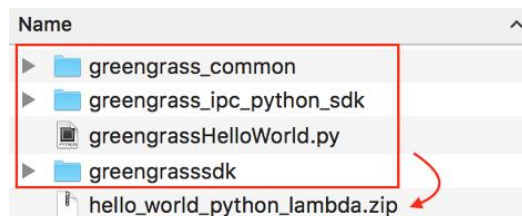
- From the "GG_Group_1" management page, select "Lambdas" from the left menu, "Add lambda", and "Create new lambda".

- 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:
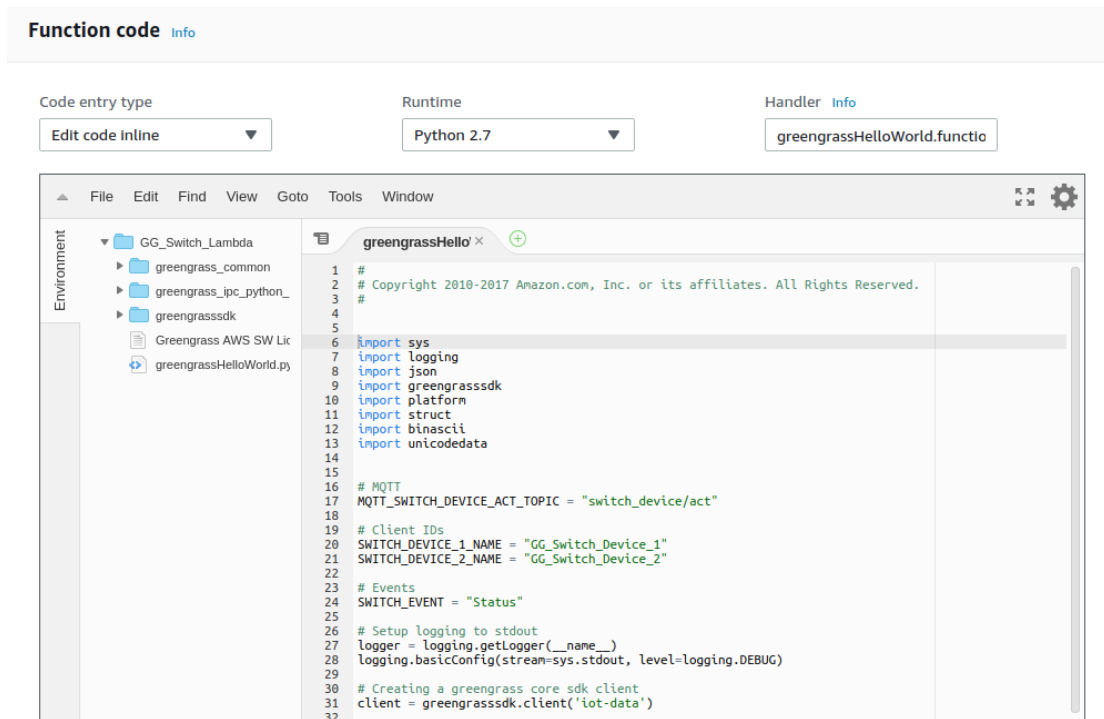
- Then under "Actions" click on "Publish new version", with a new version, e.g. "V1".
- Return on the AWS IoT console and enter the "GG_Group_1" Greengrass group, select "Lambdas", "Add Lambda", then "Use existing Lambda". Select the "GG_Switch_Lambda" lambda previously created, click "Next", select the desired version, and click on "Finish".

Creating Devices

- Enter the "GG_Group_1" group, select "Devices", then "Add Device", "Create New Device", and create the two devices "GG_Switch_Device_1" and "GG_Switch_Device_2"; then 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 Subscriptions

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

FROM	TO	TOPIC
GG_Switch_Device_1	GG_Switch_Lambda:1	switch_device/sense
GG_Switch_Device_2	GG_Switch_Lambda:1	switch_device/sense
GG_Switch_Lambda:1	GG_Switch_Device_1	switch_device/act
GG_Switch_Lambda:1	GG_Switch_Device_2	switch_device/act

- Then, for each device, create the following system subscriptions, related to shadow-devices:

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

- At the end you should have 24 subscriptions in your "GG_Group_1" Greengrass group.

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 mosquitto vim`

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 bluepy Python library, the Eclipse Paho MQTT Python client library (required to exchange messages via MQTT protocol), and the Amazon AWS IoT Python SDK:
  - `sudo pip install bluepy paho-mqtt 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 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`
- Further information can be found here:
  - https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-device-start.html#config.json-params
- Example "config.json" file:

```
{
    "coreThing": {
        "caPath": "root.ca.pem",
        "certPath": "core-certificate.pem.crt",
        "keyPath": "core-private.pem.key",
        "thingArn": "arn:aws:iot:us-west-2:<Account-ID/alias>:thing/
                    GG_Group_1_Core",
        "iotHost": "xxxxxxxxxxxxx.iot.us-west-2.amazonaws.com",
        "ggHost": "greengrass.iot.us-west-2.amazonaws.com"
    },
    "runtime": {
        "cgroup": {
            "useSystemd": "yes"
        }
    }
}
```

- Copy the core's certificate and keys downloaded when creating the group into the "/greengrass/certs/" folder. Please rename the files as shown here below:
    - root.ca.pem
    - core.cert.pem
    - core.private.key
    - core.public.key
- Create a "devices_ble_aws_1" folder into the PI3 "/home/pi/" folder, and copy all certificates and keys downloaded when creating the devices into this folder. Note that the "devices_ble_aws_1" is defined within the code through the "DEVICES_PATH" global variable. Please rename the files as shown here below:
    - root.ca.pem
    - switch_device_1.cert.pem
    - switch_device_1.private.key
    - switch_device_1.public.key
    - switch_device_2.cert.pem
    - switch_device_2.private.key
    - switch_device_2.public.key
- Copy the content of the BlueST SDK and the EdgeST SDK repositories into the following folders of the PI3:
    - /home/pi/BlueSTSDK_Python
    - /home/pi/EdgeSTSDK_Python

Bluetooth Low Energy Nodes

- Import the "Node_BLE_Switch_Device" mbed OS application to your ARM mbed account, compile, and flash it onto the two switch enabled BLE devices:
    - https://os.mbed.com/teams/ST/code/Node_BLE_Switch_Device/
- Edit the "example_ble_aws_1.py" example application in the "/home/pi/EdgeSTSDK_Python/edge_st_example/aws/" folder and set the "SWITCH_DEVICE_1_MAC" and "SWITCH_DEVICE_2_MAC" global variables to the proper MAC address of your switch enabled BLE devices (which you can retrieve for example through a smartphone application).

Deploying Greengrass on the core device

Starting the Greengrass daemon

- Login to the Raspberry PI.
- Start 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", then "Grant permission".
- Wait for the green status of the deployment.

Running the demo

Setup

- Connect and power on all the devices.

Running the main Python script

- Login to the Raspberry PI.
- Start the Greengrass daemon:
  - `sudo /greengrass/ggc/core/greengrassd restart`
- Enter the following folder:
  - `cd /home/pi/`
- Become super user (required to use the Bluetooth peripheral):
  - `sudo su`
- Set the Python path environmental variable:
  - `export PYTHONPATH="/home/pi/BlueSTSDK/:/home/pi/EdgeSTSDK/"`
- Run the application example by passing the endpoint and the path to the root certification authority certificate:
  - `python ./EdgeSTSDK_Python/edge_st_example/aws/example_ble_aws_1.py -e xxxxxxxxxxxxxx.iot.us-west-2.amazonaws.com -r /greengrass/certs/root.ca.pem`
- Press the user button on a switch enabled BLE device to toggle the status of the LED on the other device.
- To stop the application press CTRL+C.