

Green Grass Demo: Vineyard / Agricultural IIoT

Setup Guide

Table of Contents

Table of Contents	2
Revision History:	4
Introduction	5
Scope:	5
Hardware and Software Requirements	6
Hardware components:	6
Software Components:	6
Installation of Linaro OS on Dragon-Board 410c	7
Download SD Card image:	7
Choose Host Machine(Linux):	7
Installation of Greengrass on Dragon-Board 410c	11
Run AWS Greengrass core software:	11
Enable symlink and hardlink protection:	11
Reboot the system:	11
Install AWS CLI on Linux (Dragonboard 410c):	11
Create AWS Greengrass Group and AWS Greengrass Core:	12
Create a Greengrass Group:	14
Copy and Extract the Greengrass Software Package onto your core device(Dragon-board-410c):	17
Install Certificates on your AWS Greengrass Core Device (Dragon-board 410c):	17
Configure your Greengrass core:	18
Install Library to use GPIO of the 96Boards	22
Update the installed image:	22
Install Package dependencies:	22
Install 96Boards configuration files:	22
Install and build libsoc:	22
Install and build 96BoardsGPIO:	23
Install extra tool packages:	24
Attach Sensor Mezzanine Board to Dragon-Board 410c	25
Connect Grove Moisture Sensor:	25
Connect Grove RGB LED Sensor:	26
Connect Grove 10DoF IMU Sensor:	27

Connect Grove Mini Fan Sensor:	28
Connect Grove Digital Light Sensor:	29
Connect All Sensor together:	30
Code Dump in Mezzanine Board	31
Register a Device in the Thing Registry	35
To register your device in the thing registry:	35
Create and Activate a Device Certificate:	37
Attach a Certificate to a Thing:	39
Add Your device to a Greengrass Group:	40
Deploy all the device in Greengrass:	42
Create Lambda Function	43
Create a “Hello World” Lambda Function:	43
Add the Lambda Function to Your Group Definition:	47
Add Subscription to Your Group Definition:	49
Deploy your Group:	51
Installation of necessary packages	54
Installation of node.js6x:	54
Installation of aws-iot-device-sdk for python:	54
Installation of smbus:	54

Revision History:

Revision	Date	Changes
0.1	08-Nov-2017	Initial Version

Introduction

Scope:

Smart Vineyard System built on top of DragonBoard™ 410c from Arrow Electronics integrated with sensors to Monitor and Optimize growing conditions by capturing data and processing it on the edge gateway, and making real time decisions about irrigation, grape temperature for frost prevention.

The data collected from the sensors shall be stored on the cloud and analytics performed on the data on hourly, daily and long term including the quality and Quantity of wine grapes.

Hardware and Software Requirements

Hardware components:

- DragonBoard™ 410c from Arrow Electronics with power supply.
- Host machine(Linux).
- MicroSD card with 8GB or more of storage.
- USB Mouse and keyboard.
- HDMI Monitor with HDMI cable and HDMI connector.
- Sensor Mezzanine board.
- Grove Moisture sensor v1.4, Grove Digital light sensor v1.1, Grove 10DoF IMU v2.0, Grove Mini fan v1.1, Grove Chainable RGB LED v2.0.

Software Components:

- Linaro aarch64 - Please follow the below instruction to install Linaro OS on dragon board 410c.

Installation of Linaro OS on Dragon-Board 410c

Download SD Card image:

To download “SD Card Image–Install and boot from eMMC” [click here](#).

Choose Host Machine(Linux):

1. Prepare MicroSD Card,
 - a. Ensure data from microSD card is backed up.
 - b. Everything on microSD card will be lost by the end of this procedure.
2. Find SD Card Device name,
 - a. Use host computer.
 - b. Open “Terminal” application from your host machine or press **Ctrl+Alt+t**
 - c. Remove SD card from host computer and run the following command from your terminal,

```
$ lsblk
```

- d. Note all recognized disk names.
 - e. Insert SD card and run the following command (again),
- ```
$ lsblk
```
- f. Note the newly recognized disk. This will be your SD card.
    - g. Remember your SD card device name, it will be needed in Step 7.
  3. Recall Download Location,
    - a. Locate SD card install file from Downloads page.
    - b. This file will be needed for the next step.

4. Unzip SD Card Install Image,
  - a. Command for Unzip file:

```
$ unzip <filename>
```

- b. When unzipped, you will have a folder with the following contents:
    - i. Install Image(.img)
    - ii. License.txt
5. Go to directory with SD Card Install Image folder using Terminal,
  - a. Open “Terminal” application from your host computer.
  - b. Change to the directory where you unzipped SD Card Install Image,

```
$ cd /home/username/Downloads
```

6. Locate SD Card Install Image,
  - a. Make sure you are in the extraction directory.
  - b. Inside this folder you will find install image “**db410c\_sd\_install\_debian.img**”.
  - c. This .img file is what will be flashed to your SD Card.
7. Install Image onto SD Card. Checklist,
  - a. SD card inserted into host computer.
  - b. Recall SD Card device name from Step 2
  - c. Within that downloaded folder you will find “**db410c\_sd\_install\_debian.img**” and from terminal run the below command.

```
$ sudo dd if=db410c_sd_install_debian.img of=/dev/XXX bs=4M
oflag=sync status=noxfer
```

**NOTE** : Syntax for the command above,

```
$ sudo dd if=<name of image downloaded> of=/dev/<SD Card
deviceName without partition> bs=4M oflag=sync status=noxfer
```

- d. This command will take some time to execute. Be patient and avoid tampering with the terminal until process has ended.
  - e. Once SD card is done flashing, remove from host computer.
8. Prepare Dragonboard 410c with SD card,
  - a. Make sure Dragonboard™410c is unplugged from power.
  - b. Set S6 switch on Dragonboard™410c to **0-1-0-0**, “SD Boot switch” should be set to “**ON**”. See S6 switch on the below diagram:



- c. Connect an HDMI monitor to the Dragonboard™410c with an HDMI cable, and power on the monitor.
  - d. Plug a USB keyboard and/or mouse into either of the two USB connectors on the Dragonboard™410c.
  - e. Insert the microSD card into the Dragonboard™ 410c.
  - f. Plug power adaptor into Dragonboard™ 410c, wait for board to boot up.



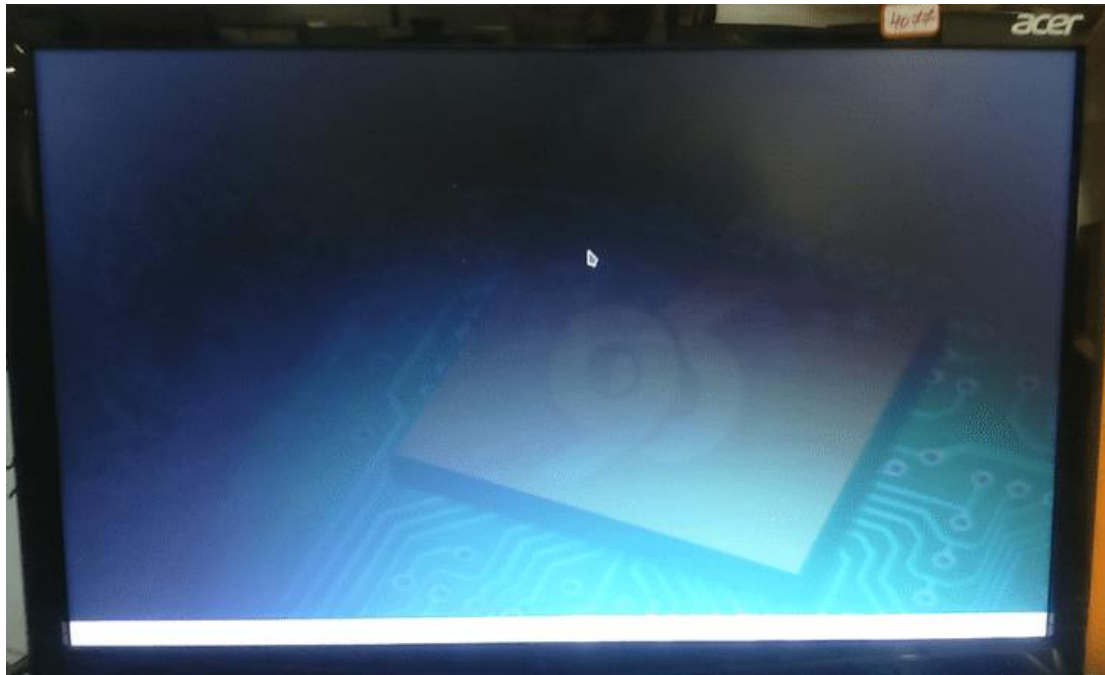


9. Install image onto Dragonboard 410c. If Steps 1 - 8 were followed correctly, the below screen should be visible from your Dragonboard™410c,



- Select the image to install and click "Install icon". OS will be installed into the eMMC memory.
- This process can take a few minutes to complete.
- Upon completion, "Flashing has completed and OS has installed successfully...." message will appear.
- Before clicking "OK"**, Remove the SD Card.
- Set S6 switch on Dragonboard™ 410c to **0-0-0-0**, all switches should be set to **"OFF"**.
- Now click **"OK"** button and allow Dragonboard™410c to reboot.

- g. It takes some time to reboot, After Some time, if screen is not appeared, then plug it off and again replugin it of Dragonboard, After that Below Screen will appear.



## Installation of Greengrass on Dragon-Board 410c

### Run AWS Greengrass core software:

#### *Enable symlink and hardlink protection:*

The Debian release for the DragonBoard 410c has symlink and hardlink protection disabled by default. The following creates **sysctl** configuration file with the options to enable the protection. Open a terminal application from your dragonboard 410c or press **Ctrl+Alt+t** and type the below command,

```
$ sudo sh -c 'echo "fs.protected_hardlinks = 1" >> /etc/sysctl.d/local-ggc.conf'
$ sudo sh -c 'echo "fs.protected_symlinks = 1" >> /etc/sysctl.d/local-ggc.conf'
```

#### *Reboot the system:*

```
$ sudo reboot
```

After rebooting your Dragonboard 410c, you can verify the protection is working by checking:

```
$ sudo cat /proc/sys/fs/protected_{hardlinks,symlinks}
```

**NOTE** : You should see two 1s.

Use the following command to add a user for greengrass. Open a terminal from your dragonboard 410c or press **Ctrl+Alt+t** and type the below command,

```
$ sudo adduser --system ggc_user
$ sudo addgroup --system ggc_group
$ sudo apt-get update
```

Use the following command to install sqlite3,

```
$ sudo apt-get install sqlite3
```

Use the following command to install cmake,

```
$ sudo apt-get install cmake
```

### Install AWS CLI on Linux (Dragonboard 410c):

Open your terminal application:-

1. Check your Python installation,

```
$ python --version
```

2. Download the AWS CLI Bundled installer from below command,

```
$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o
"awscli-bundle.zip"
```

3. Unzip the package,

```
$ unzip awscli-bundle.zip
```

4. Run the install executable,

```
$ sudo ./awscli-bundle/install -i /usr/local/aws -b
/usr/local/bin/aws
```

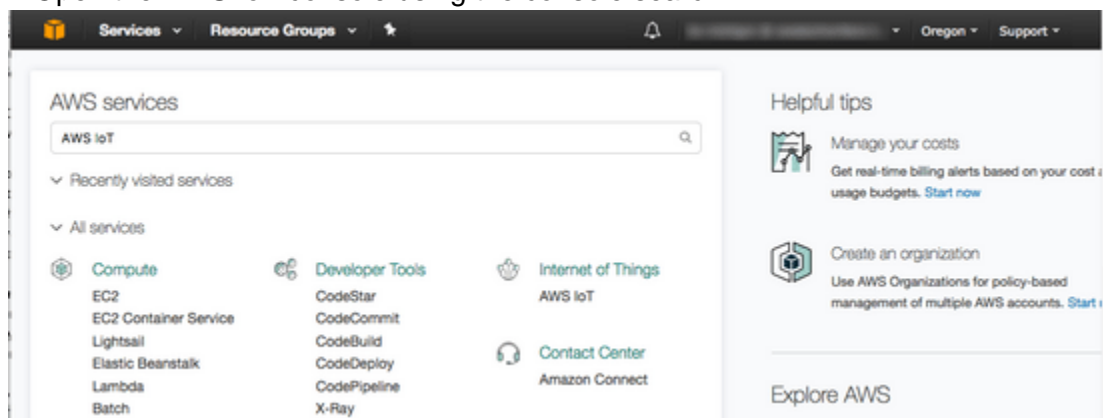
5. Reboot the system, open your terminal in dragon-board 410c and type,

```
$ sudo reboot
```

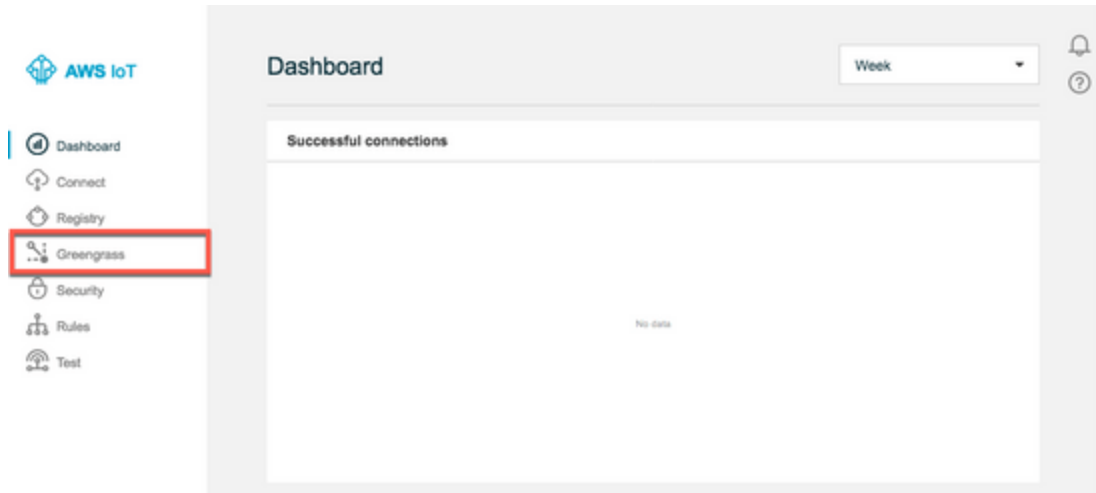
## Create AWS Greengrass Group and AWS Greengrass Core:

Assuming you have AWS account.

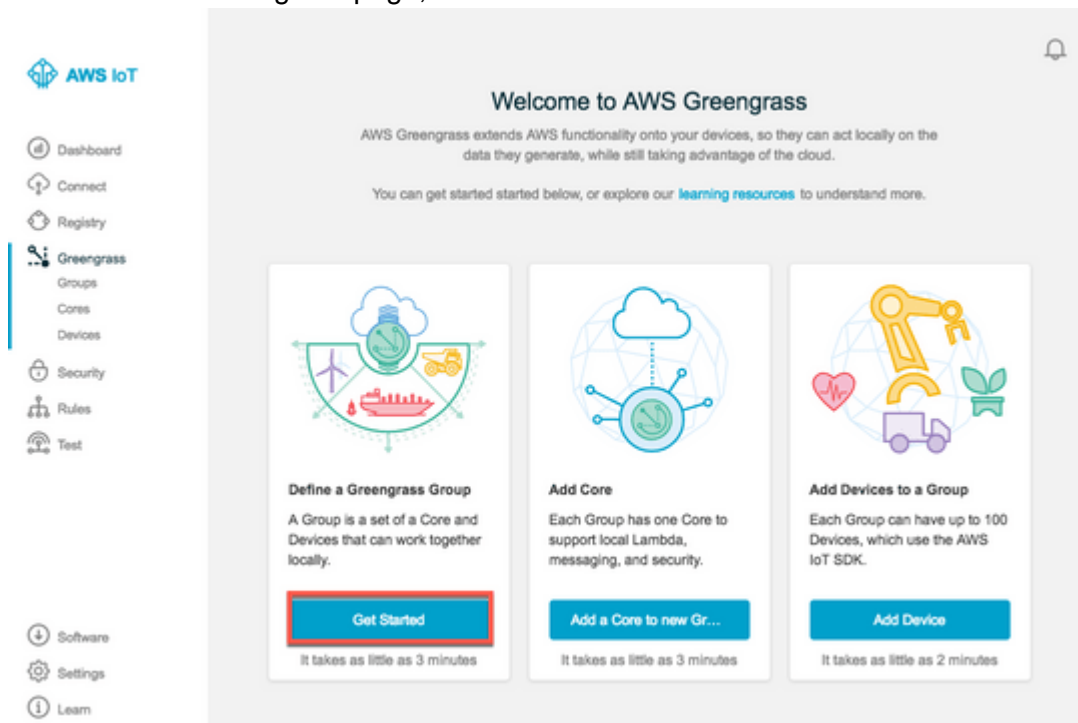
1. Sign in to the AWS Management console.
2. Open the AWS IoT console using the console search.



3. From the navigation pane, choose **Greengrass**.



- On the AWS Greengrass page, choose **Get Started**.



## Create a Greengrass Group:

On the Set up your Greengrass group page, choose **Use easy creation**

**Set up your Greengrass Group**

Setting up your Group requires you to provision a Core device in the IoT Registry, acquire a certificate for your Core, and assign an IAM role to your Group. If you're unfamiliar with any of these steps we recommend the easy Group creation. Finally, you'll need to install Greengrass software on your Core device.

**Easy Group creation (recommended)**  
This process will automatically provision a Core in the registry, use default settings to generate a new Group, and provide your Core with a new certificate and a key pair.

[Use easy creation](#)

**Advanced Group creation**  
This customizable process will take you step-by-step through the Core provisioning and will allow you to customize the IAM Role for your Group and the certificate for your Core, and provide a key pair.

[Customize](#)

[Back](#) [Use easy creation](#)

1. Type a name for your group, and then choose **Next**.

**SET UP YOUR GREENGRASS GROUP**  
**Name your Group**

The Greengrass Group is a cloud-configured managed collection of local devices and Lambda functions that can be programmed to communicate with each other through a Core device. Groups can contain up to 100 local devices.

Group Name

[Back](#) [Next](#)

2. Use the default name for your AWS Greengrass core, and then choose **Next**.

SET UP YOUR GREENGRASS GROUP

Every Group needs a Core to function

Every Greengrass Group requires a device running Core software. It enables communication between Devices, local Lambda functions, and AWS cloud computing services. Adding information to the Registry is the first step in provisioning a device as your Greengrass Core.

Name

MyFirstGroup\_Core

Show optional configuration (this can be done later) ▼

Back

Next

3. Choose **Create Group and Core**.

SET UP YOUR GREENGRASS GROUP

Run a scripted easy Group creation

In order to speed up and simplify Group creation AWS Greengrass will handle the following processes and use default settings. By proceeding to the next step, you are giving permission for us to complete the following steps.

AWS Greengrass will take these actions on your behalf using default settings:

|                                                                 |                            |
|-----------------------------------------------------------------|----------------------------|
| Create a new Greengrass Group in the cloud                      | <a href="#">Learn more</a> |
| Provision a new Core in the IoT Registry and add to the Group   | <a href="#">Learn more</a> |
| Generate public and private key set for your Core               | <a href="#">Learn more</a> |
| Generate a new security certificate for the Core using the keys | <a href="#">Learn more</a> |
| Attach a default security policy to the certificate             | <a href="#">Learn more</a> |

Back




Create Group and Core

- Click the links to download the private key, public key, and certificate for your AWS Greengrass core.

## Connect your Core device

The final steps are to load the Greengrass software and then connect your Core device to the cloud. You can defer connecting your device at this time, but **you must download your public and private keys now as these cannot be retrieved later.**

### Download and store your Core's security resources

|                             |                        |                                                                                    |
|-----------------------------|------------------------|------------------------------------------------------------------------------------|
| A certificate for this Core | f4be4f2917.cert.pem    |  |
| A public key                | f4be4f2917.public.key  |                                                                                    |
| A private key               | f4be4f2917.private.key |  |
| Core-specific config file   | config.json            |  |

[Download these resources as a tar.gz](#)

- Choose the AArch64(ARMv8)(for your dragonboard 410c)and then choose **Download Greengrass** to download the Greengrass software package. When the download is complete, choose **Finish**.





### Download the current Greengrass Core software

To install Greengrass on your Core download the package and follow [Getting Started Guide](#) .

### Software configurations

All

All

| Architecture ▾  | Distribution ▾       | OS ▾  |                                                                                                              |
|-----------------|----------------------|-------|--------------------------------------------------------------------------------------------------------------|
| x86_64          | Amazon Linux         | Linux |  <a href="#">Download</a> |
| ARMv8 (AArch64) | Ubuntu 14.04 - 16.04 | Linux |  <a href="#">Download</a> |
| ARMv7l          | Raspbian Jessie      | Linux |  <a href="#">Download</a> |
| x86_64          | Ubuntu 14.04 - 16.04 | Linux |  <a href="#">Download</a> |

By downloading this software you agree to the [Greengrass Core Software License Agreement](#) .



## Copy and Extract the Greengrass Software Package onto your core device(Dragon-board-410c):

1. Determine the IP address of the device where you will install the AWS Greengrass core software (Dragonboard 410c).
2. The name of Greengrass software package, “**greengrass-platform-version.tar.gz**”
3. Use the scp command to copy the Greengrass software package into the home directory of your AWS Greengrass core device. (Use the IP\_ADDRESS of your Greengrass core device).

```
$ scp greengrass-platfrom-version.tar.gz linaro@IP_ADDRESS:~
```

4. Similarly copy the all the certificates & key into the AWS GreenGrass core device (Dragonboard 410c).
5. On your AWS Greengrass core device (Dragonboard 410c), use the following command to expand the package,

```
$ sudo tar -zxvf greengrass-<os-name>-<platform-type>-<version>.tar.gz -C /
```

## Install Certificates on your AWS Greengrass Core Device (Dragon-board 410c):

1. Download the AWS IoT root CA certificate from Symantec/Version. You can download from command-line of host machine,

```
$ wget https://www.symantec.com/content/en/us/enterprise/verisign/roots/VeriSign-Class%203-Public-Primary-Certification-Authority-G5.pem
```

2. Move downloaded file into AWS GreenGrass core device (Dragonboard 410c) using scp command, (Use Dragonboard 410c ipaddress)

```
$ scp VeriSign-Class\ 3-Public-Primary-Certification-Authority-G5.pem linaro@IP_ADDRESS:~
```

3. Move the file in device to /greengrass/certs directory. Command as follows,

```
$ sudo mv VeriSign-Class\ 3-Public-Primary-Certification-Authority-G5.pem /greengrass/certs/root-ca.pem
```

4. Copy the AWS Greengrass core's private key and certificate and the AWS IoT root CA certificate into the /greengrass/configuration/certs directory of your AWS Greengrass core device. Use the following names,

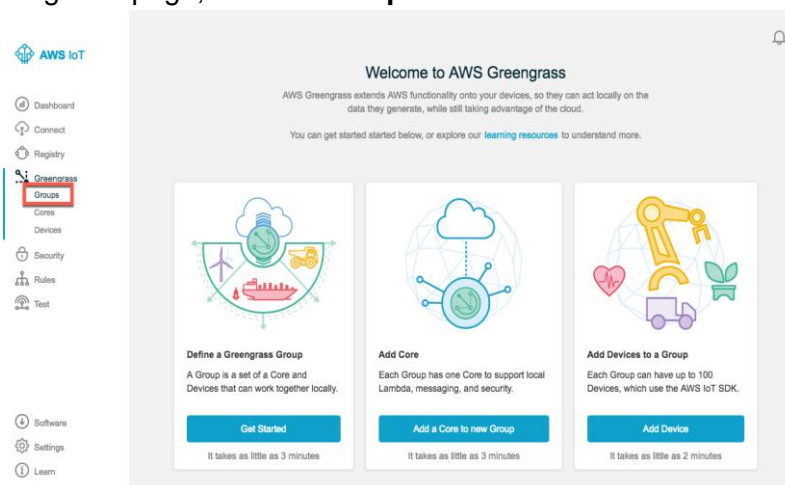
| File          | Description                                                                                                                           |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------|
| cloud.pem.crt | The AWS Greengrass core certificate downloaded from the console (for example, 61970a435c-certificate.pem.crt or b01d32b3f6.cert.pem). |
| cloud.pem.key | The private key downloaded from the console (for example, 61970a435c-private.pem.key or b01d32b3f6.private.key).                      |
| root-ca.pem   | The AWS IoT root CA certificate from Symantec.                                                                                        |

**NOTE:** You can use the scp command to copy the files to your AWS Greengrass core device. By default, you do not have permission to copy the files directly to greengrass directory. To work around this, copy the files to your user directory and then use move the files to the /greengrass/configuration/certs directory.

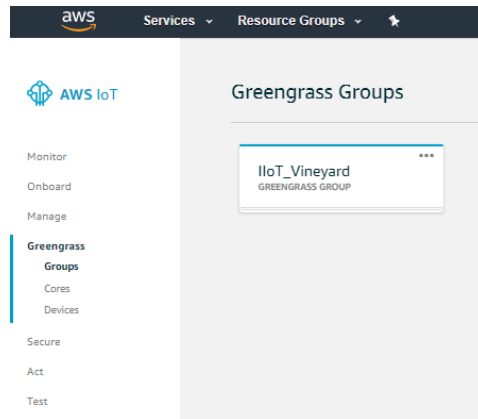
## Configure your Greengrass core:

You need to associate your AWS Greengrass core with the AWS IoT thing you created for it. You do that by using the thing's ARN, the unique identifier assigned in the cloud to your AWS Greengrass core device when you provisioned it.

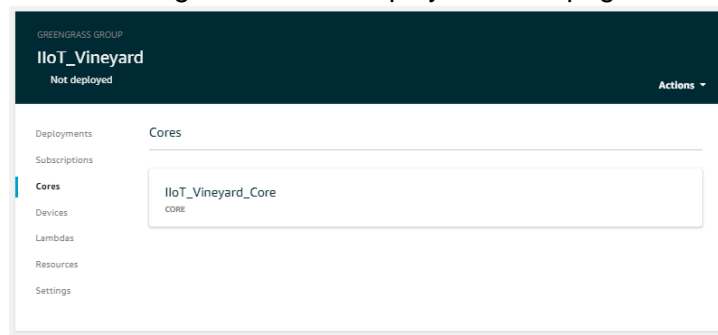
1. Find the thing ARN for your AWS Greengrass core. In the AWS Greengrass console, from the navigation page, choose **Groups**.



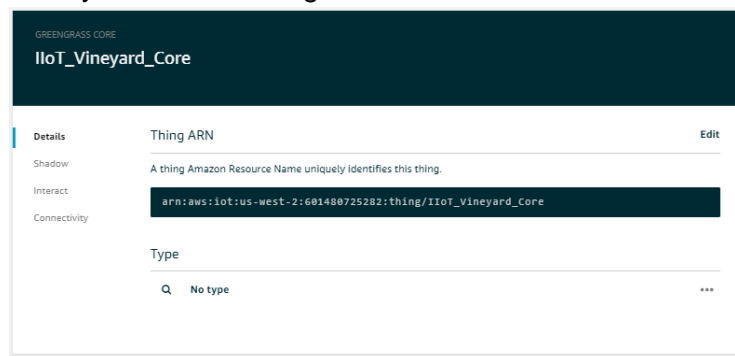
- Choose your group to display its detail page.



- In the navigation pane, choose **Cores**.
- Choose your AWS Greengrass core to display its detail page.



- Copy the ARN of your AWS Greengrass core.



- On your AWS Greengrass core device, open your terminal do following,

```
$ sudo vim /greengrass/configuration/config.json
```

Initially it will look like this,

```
{
 "coreThing": {
 "caPath": "<ROOT_CA_PEM_HERE>",
 "certPath": "<CLOUD_PEM_CRT_HERE>",
 "keyPath": "<CLOUD_PEM_KEY_HERE>",
 "thingArn": "<THING_ARN_HERE>",
 "iotHost":
"<HOST_PREFIX_HERE>.iot.<AWS_REGION_HERE>.amazonaws.com",
 "ggHost": "greengrass.iot.<AWS_REGION_HERE>.amazonaws.com",
 "keepAlive": 600
 },
 "runtime": {
 "cgroup": {
 "useSystemd": "[yes|no]"
 }
 }
}
```

Update the file with following content,

```
{
 "coreThing": {
 "caPath": "root-ca.pem",
 "certPath": "cloud.pem.crt",
 "keyPath": "cloud.pem.key",
 "thingArn": "arn:aws:iot:us-west-
2:601480725282:thing/IIoT_Vineyard_Core",
 "iotHost": "your-AWS-IoT-endpoint",
 "ggHost": "greengrass.iot.us-west-2.amazonaws.com",
 "keepAlive": 600
 },
 "runtime": {
 "cgroup": {
 "useSystemd": "yes"
 }
 }
}
```

| Attribute | Description                                                                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| caPath    | The path to the AWS IoT root CA certificate.                                                                                                                  |
| certPath  | The path to the AWS Greengrass core certificate.                                                                                                              |
| keyPath   | The path to the AWS Greengrass core private key.                                                                                                              |
| thingArn  | The ARN of the AWS IoT thing associated with your AWS Greengrass core.                                                                                        |
| iotHost   | Your AWS IoT endpoint. Can be obtained using the <code>aws iot describe-endpoint</code> CLI command or in the <b>Settings</b> section of the AWS IoT console. |
| ggHost    | A fully qualified AWS Greengrass host address. Replace <code>AWS_REGION</code> with the AWS region you are using.                                             |

7. Start Your AWS Greengrass Core to connect It to the Cloud.
8. Change to `/greengrass` directory and run the following command to start your core and enable a cloud connection to AWS IoT.

```
$ sudo ./greengrassd start
```

If the Greengrass daemon started successfully, you should see the following output:

```
Setting up greengrass daemon
Validating execution environment
ggc_group:x:119:
Found cgroup subsystem: cpu
Found cgroup subsystem: cpuacct
Found cgroup subsystem: blkio
Found cgroup subsystem: memory
Found cgroup subsystem: devices
Found cgroup subsystem: freezer
Found cgroup subsystem: net_cls
Starting greengrass daemon
PID: 1306
Greengrass daemon started
```

If you see the error like, **The cgroup subsystem is not mounted: cpuset**  
This means cgroup is not mount, to do you need to run a script. Click here for [script](#).

## Install Library to use GPIO of the 96Boards

### Update the installed image:

Open your terminal from 96Boards and type below command:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
```

### Install Package dependencies:

```
$ sudo apt-get install git build-essential autoconf automake libtool
swig3.0 python-dev nodejs-dev cmake pkg-config libpcrc3-dev
$ sudo apt-get clean
```

### Install 96Boards configuration files:

```
$ git clone https://github.com/96boards/96boards-tools.git
$ sudo cp 96boards-tools/70-96boards-common.rules /etc/udev/rules.d/
$ sudo vim /etc/profile.d/96Boards-sensor.sh
```

Append the following lines into the above opened file,

```
export JAVA_TOOL_OPTIONS="-Dgnu.io.rxtx.SerialPorts=/dev/tty96B0"
export MONITOR_PORT=/dev/tty96B0
export PYTHONPATH="$PYTHONPATH:/usr/local/lib/python2.7/site-packages"
```

Copy the file,

```
$ sudo cp /etc/profile.d/96boards-sensors.sh /etc/X11/Xsession.d/96boards-
sensors
```

### Install and build libsoc:

From your terminal type below command:

```
$ cd /home/linaro
$ git clone https://github.com/jackmitch/libsoc.git
$ cd libsoc
$ autoreconf -i
$./configure --enable-python2 --enable-board=dragonboard410c
$ make
```

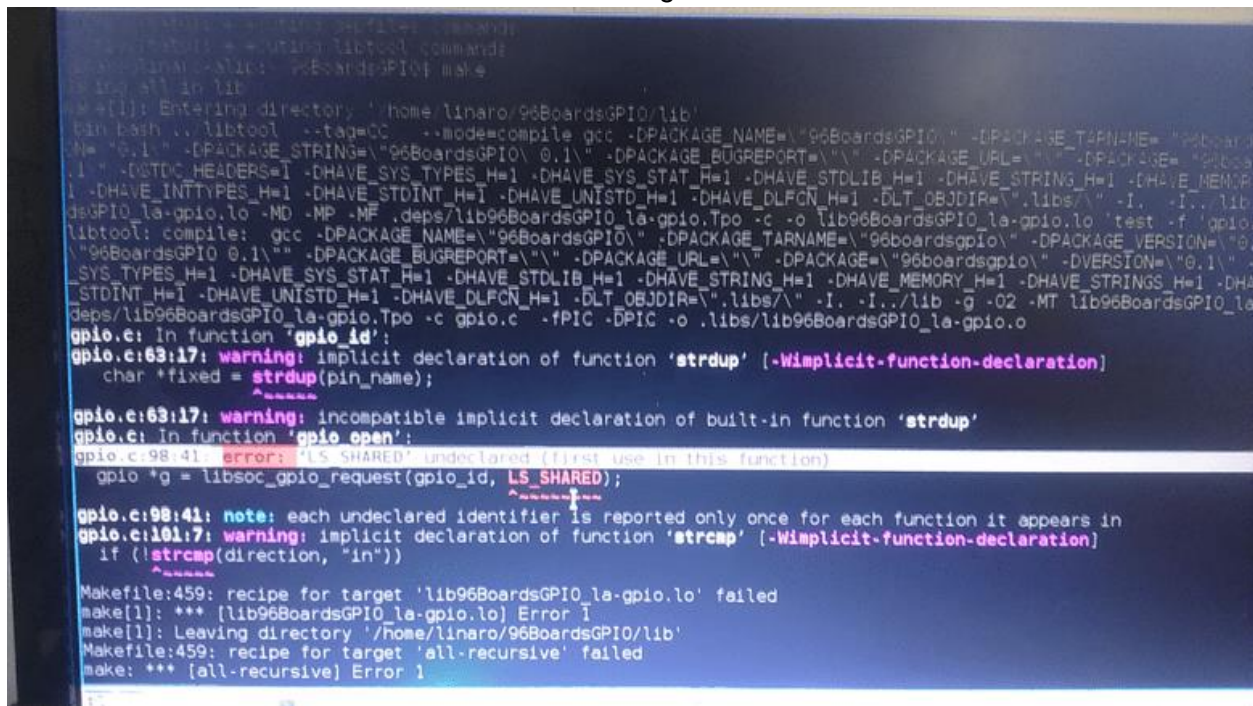
```
$ sudo make install
$ sudo ldconfig /usr/local/lib
```

## Install and build 96BoardsGPIO:

This library requires libsoc to be installed first. From your terminal type below command:

```
$ cd /home/linaro
$ git clone https://github.com/96boards/96BoardsGPIO.git
$ cd 96BoardsGPIO
$./autogen.sh
$./configure
$ make
```

In case if the make command it shows the following error:



```
make[1]: Entering directory '/home/linaro/96BoardsGPIO/lib'
/bin/bash ./libtool --tag=CC --mode=compile gcc -DPACKAGE_NAME='96BoardsGPIO' -DPACKAGE_TARNAME='96boards' -DPACKAGE_VERSION='0.1' -DPACKAGE_STRING='96BoardsGPIO 0.1' -DPACKAGE_BUGREPORT='' -DPACKAGE_URL='' -DPACKAGE='96boards' -DSTDC_HEADERS=1 -DHAVE_SYS_TYPES_H=1 -DHAVE_SYS_STAT_H=1 -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -DHAVE_MEMORY_H=1 -DHAVE_INTTYPES_H=1 -DHAVE_STDINT_H=1 -DHAVE_UNISTD_H=1 -DHAVE_DLFCN_H=1 -DLT_OBJDIR='libs/' -I. -I../lib -dsGPIO_la-gpio.lo -MD -MP -MF .deps/lib96BoardsGPIO_la-gpio.Tpo -c -o lib96BoardsGPIO_la-gpio.lo 'test -f 'gpio.c' || echo './gpio.c' -DPACKAGE_NAME='96BoardsGPIO' -DPACKAGE_TARNAME='96boardsgpio' -DPACKAGE_VERSION='0.1' -DPACKAGE_BUGREPORT='' -DPACKAGE_URL='' -DPACKAGE='96boardsgpio' -DVERSION='0.1' -DHAVE_SYS_TYPES_H=1 -DHAVE_SYS_STAT_H=1 -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -DHAVE_MEMORY_H=1 -DHAVE_STRINGS_H=1 -DHAVE_UNISTD_H=1 -DHAVE_DLFCN_H=1 -DLT_OBJDIR='libs/' -I. -I../lib -g -O2 -MT lib96BoardsGPIO_la-gpio.lo -fPIC -DPIC -o .libs/lib96BoardsGPIO_la-gpio.o
gpio.c: In function 'gpio_id':
gpio.c:63:17: warning: implicit declaration of function 'strdup' [-Wimplicit-function-declaration]
char *fixed = strdup(pin_name);
 ^~~~~~
gpio.c:63:17: warning: incompatible implicit declaration of built-in function 'strdup'
gpio.c: In function 'gpio_open':
gpio.c:98:41: error: 'LS_SHARED' undeclared (first use in this function)
gpio *g = libsoc_gpio_request(gpio_id, LS_SHARED);
 ^~~~~~
gpio.c:98:41: note: each undeclared identifier is reported only once for each function it appears in
gpio.c:101:7: warning: implicit declaration of function 'strcmp' [-Wimplicit-function-declaration]
if (!strcmp(direction, "in"))
 ^~~~~~
Makefile:459: recipe for target 'lib96BoardsGPIO_la-gpio.lo' failed
make[1]: *** [lib96BoardsGPIO_la-gpio.lo] Error 1
make[1]: Leaving directory '/home/linaro/96BoardsGPIO/lib'
Makefile:459: recipe for target 'all-recursive' failed
make: *** [all-recursive] Error 1
```

Please run the following procedure to fix,

```
$ cd /home/linaro/96BoardsGPIO/lib
```

In this Directory you will see **gpio.c** file. In that file go to line number **98** and modify as follow:

```
$ sudo vim gpio.c +98
```

before changing line inside file(gpio.c), It should looks like as below:

```
gpio *g = libsoc_gpio_request(gpio_id, LS_SHARED)
```

After changing line inside file(gpio.c), It should looks like as below:

```
gpio *g = libsoc_gpio_request(gpio_id, LS_GPIO_SHARED)
```

Add Header file in gpio.c for removing the warnings:

```
#include <string.h>
```

save the gpio.c file and proceed with following command,

```
$ cd /home/linaro/96BoardsGPIO
$ make
$ sudo make install
$ sudo ldconfig /usr/local/lib
```

Reset the system,

```
$ sudo reboot
```

## Install extra tool packages:

From your terminal application (Dragonboard 410c) type below command for **ARDUINO IDE**,

```
$ sudo apt-get install arduino-mk arduino git build-essential autoconf
libtool swig3.0 python-dev nodejs-dev cmake pkg-config libpcrc3-dev
$ sudo apt-get clean
```



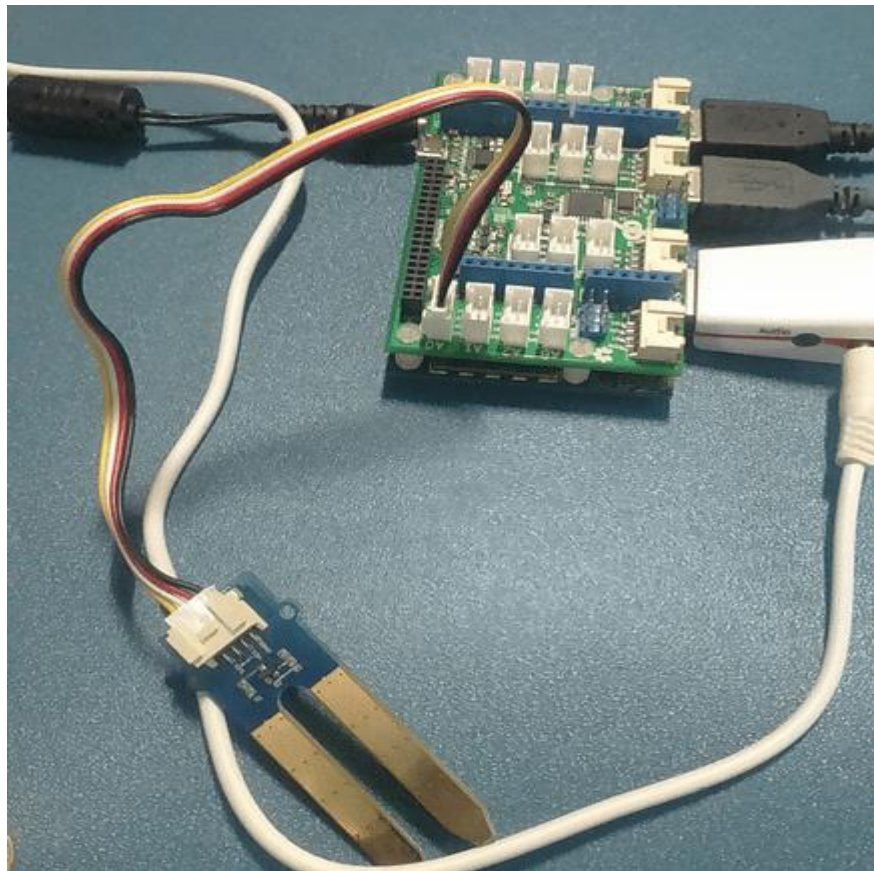
## **Attach Sensor Mezzanine Board to Dragon-Board 410c**

The following figure shows how to attach both board,



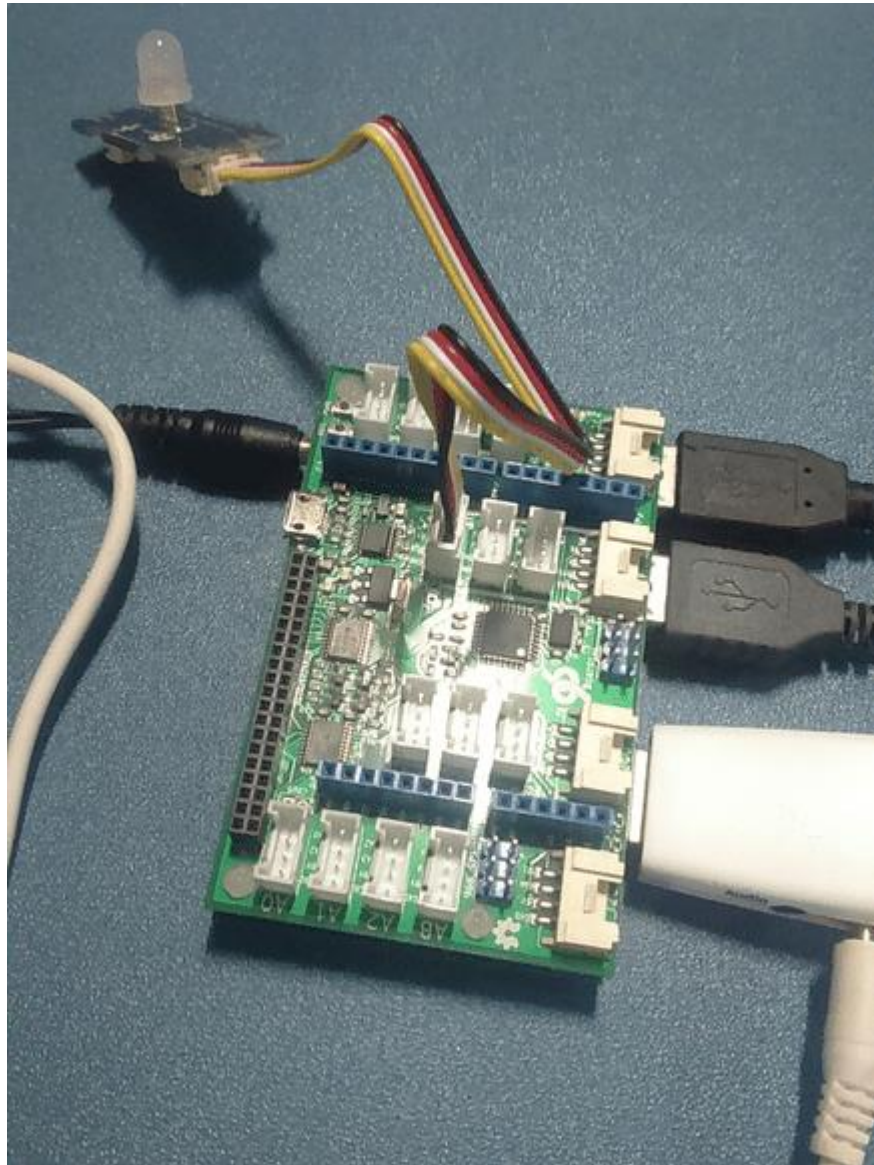
### **Connect Grove Moisture Sensor:**

The figure shows how to connect grove moisture-sensor to A0 pin of seed Sensor Mezzanine for 96Boards.



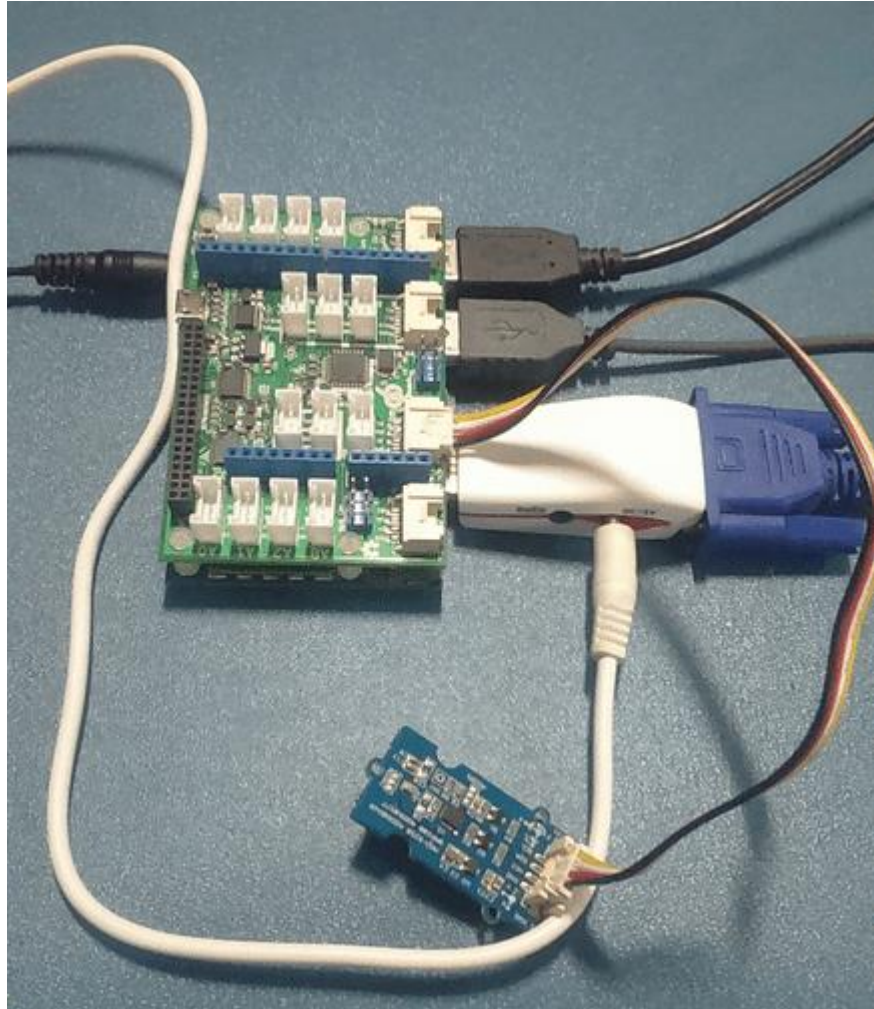
## Connect Grove RGB LED Sensor:

The figure shows how to connect grove RGB\_LED sensor to KL port pin of seeed Sensor Mezzanine for 96Boards.



## Connect Grove 10DoF IMU Sensor:

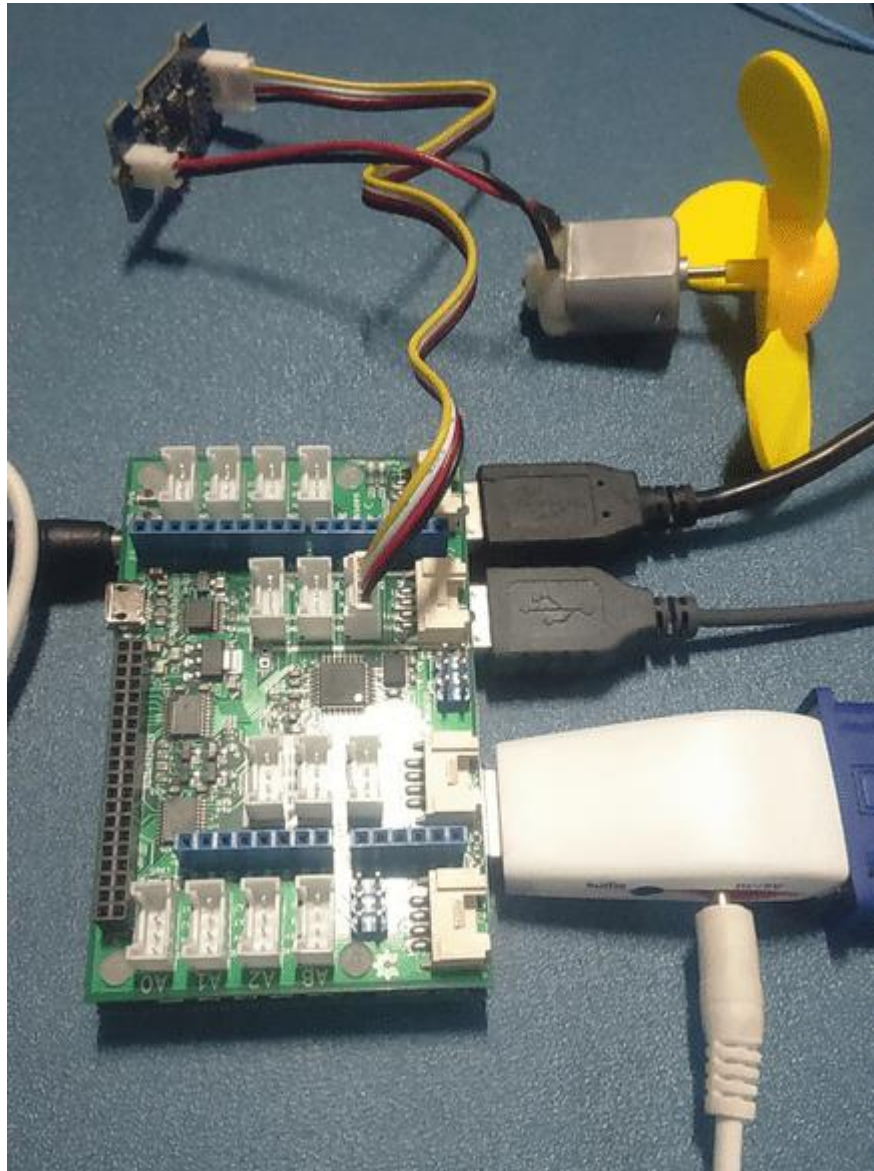
The figure shows how to connect grove 10DoF IMU sensor to i2c port(baseboard) of seeed Sensor Mezzanine for 96Boards.





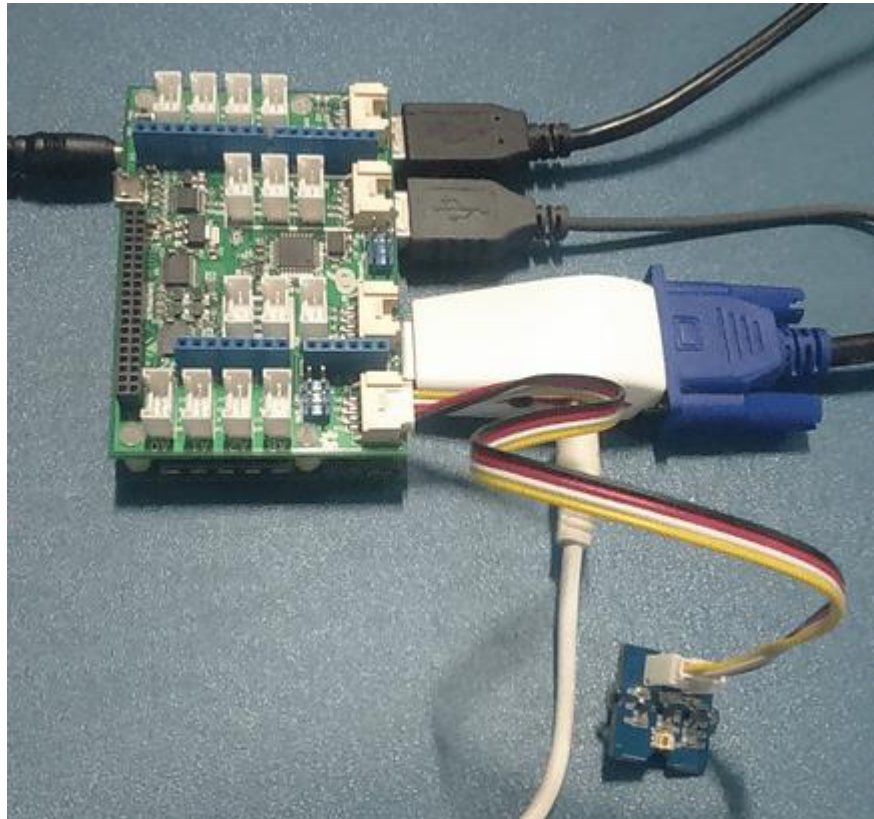
## Connect Grove Mini Fan Sensor:

The figure shows how to connect grove Connect grove mini fan sensor to GH port of seed Sensor Mezzanine for 96Boards.

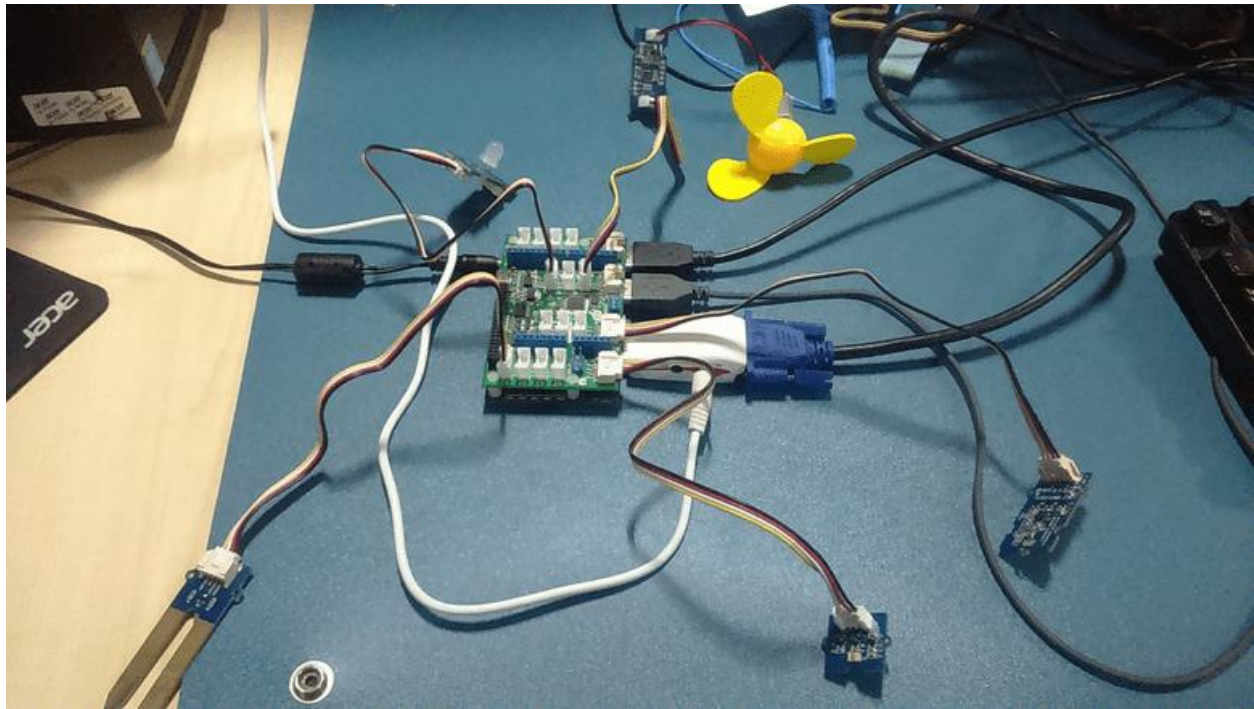


## Connect Grove Digital Light Sensor:

The figure shows how to connect grove Connect grove digital light sensor to i2c port of seeed Sensor Mezzanine for 96Boards.



## Connect All Sensor together:



## Code Dump in Mezzanine Board

1. To download library [click here](#).

2. Unzip the downloaded file,

```
$ cd /home/linaro/Downloads
$ unzip Grove_Digital_Light_Sensor-master.zip
```

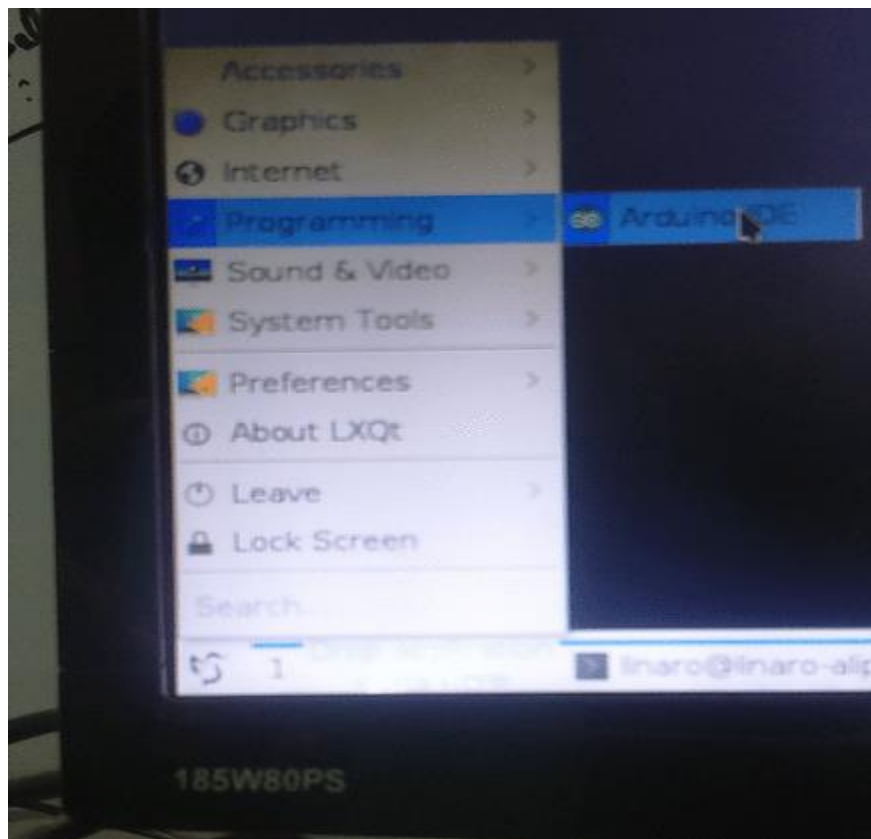
3. After unzip, rename the folder with Digital\_Light\_TSL2561(because In Arduino library its support file with name & number).

```
$ sudo mv Grove_Digital_Light_Sensor-master Digital_Light_TSL2561
```

4. Place Digital\_Light\_TSL2561 library in /usr/share/arduino/libraries from your terminal application.

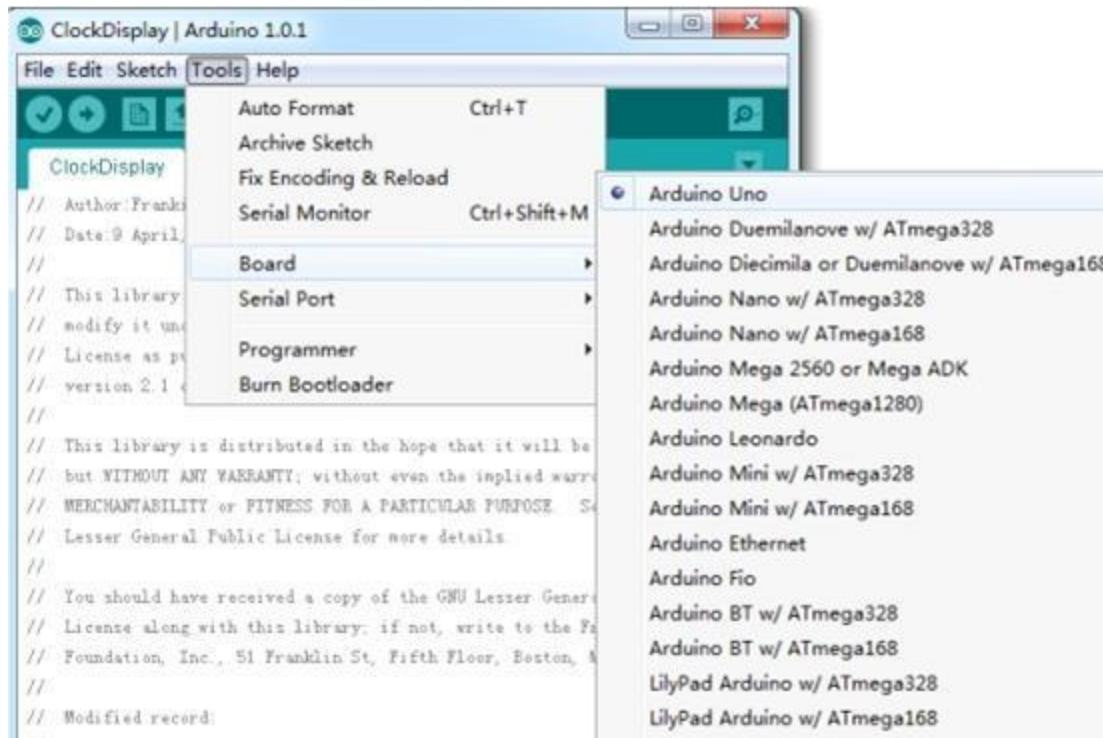
```
$ sudo cp -r Digital_Light_TSL2561 /usr/share/arduino/libraries
```

5. Open Arduino IDE.

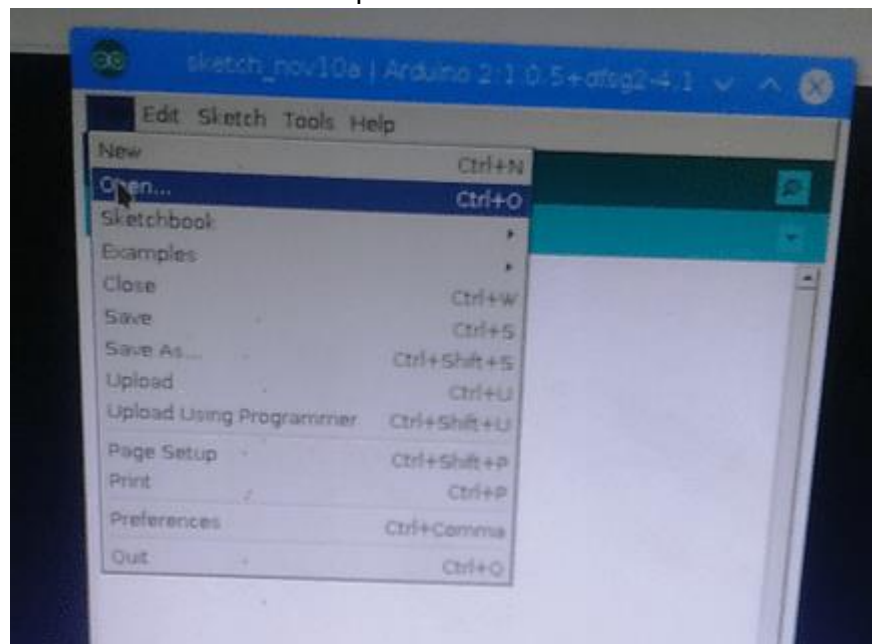


6. Select correct board from tools (in this case select Arduino Uno).



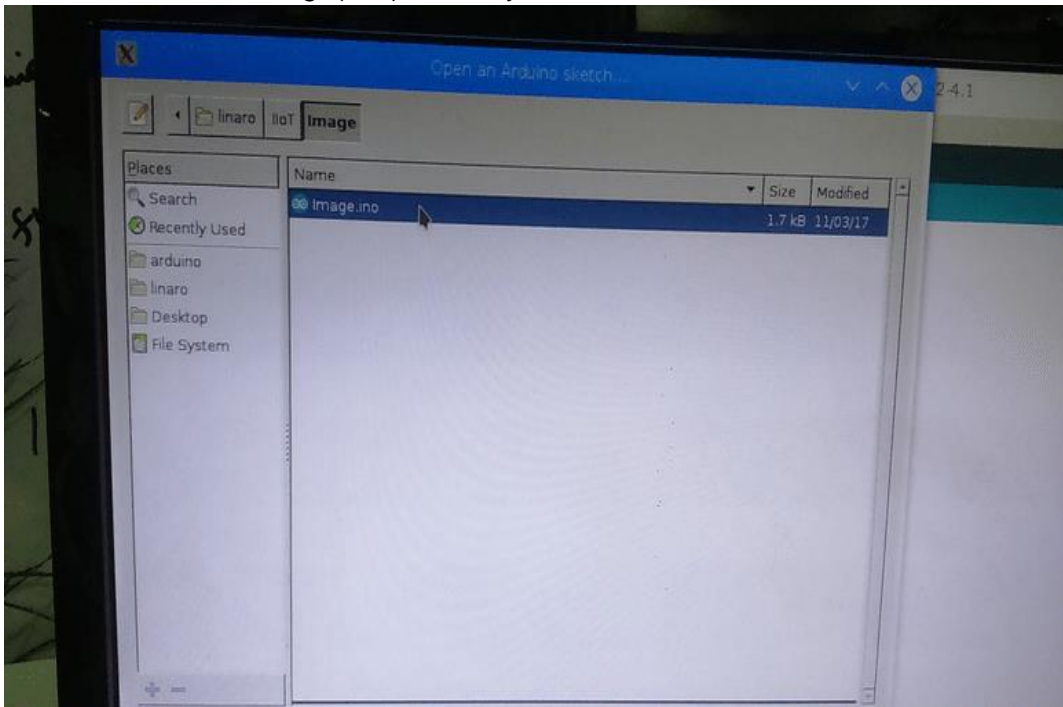


7. Click on 'files' and then click on 'open'.



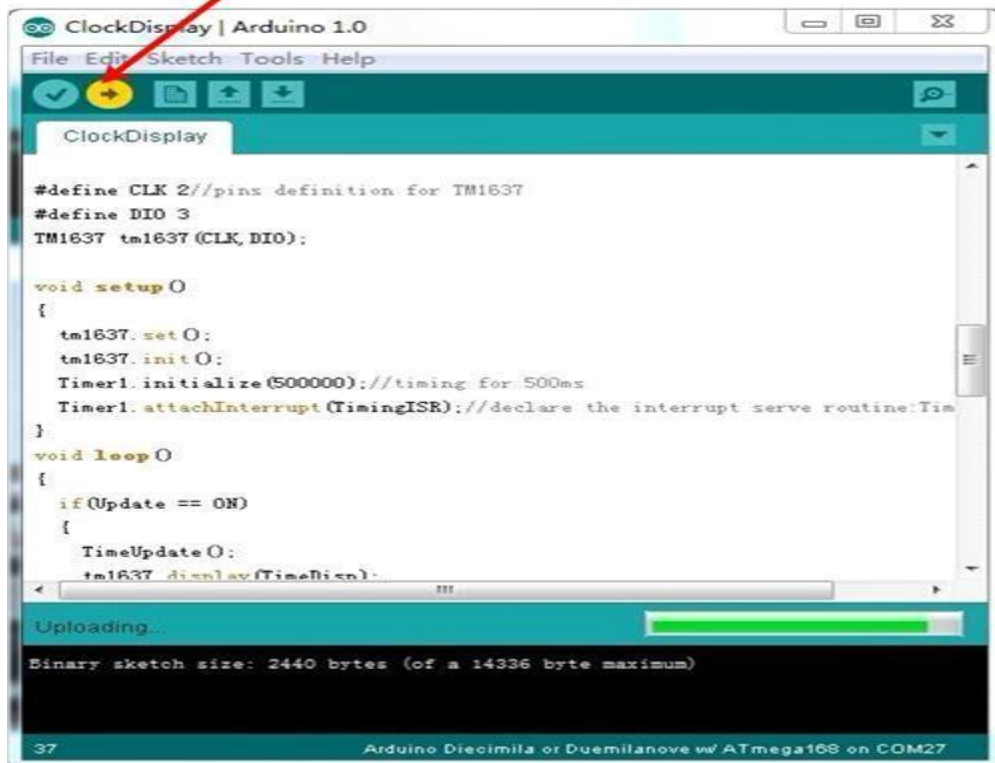


8. Select the arduino image(.ino), where your code located.

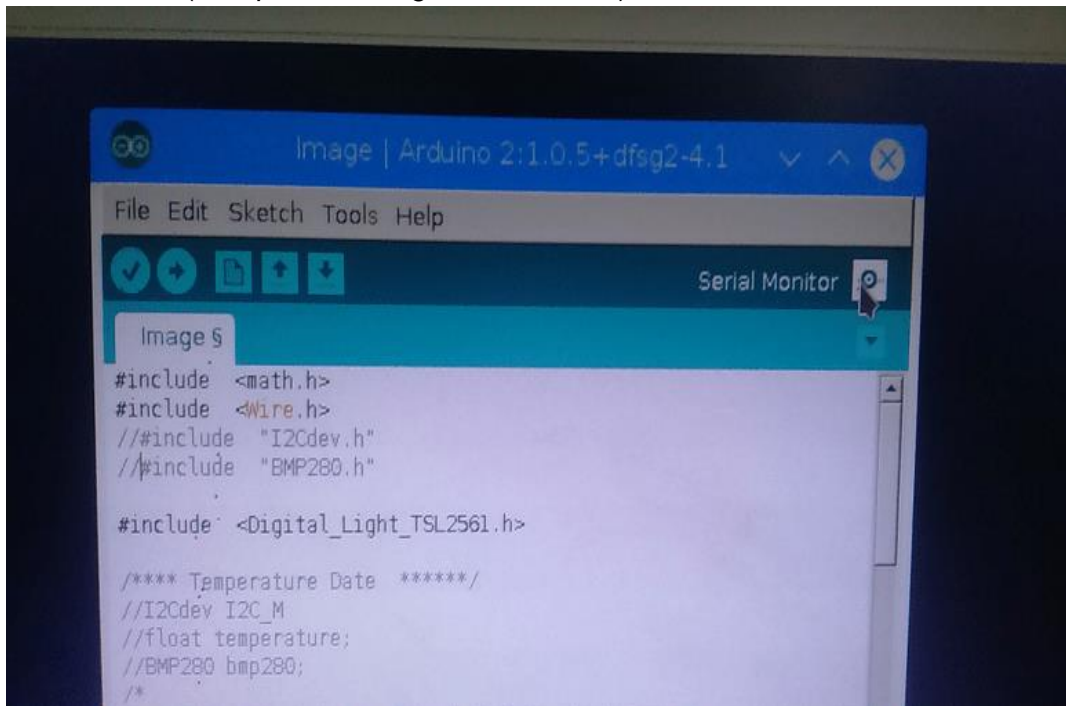


9. Upload the image,

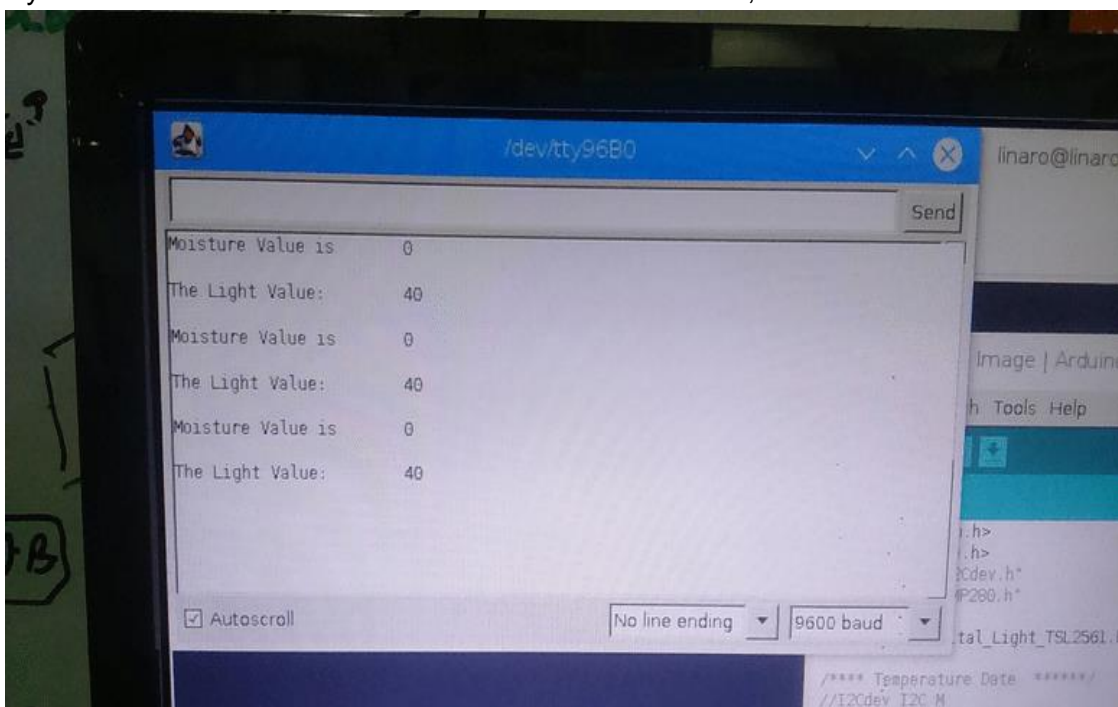
Click to upload



After dumping the code in Arduino check whether its Working Properly or not. You can verify using Serial Monitor (icon present at right-most-corner) and click on it.



When you Click on Serial Monitor Icon it will look like as below,

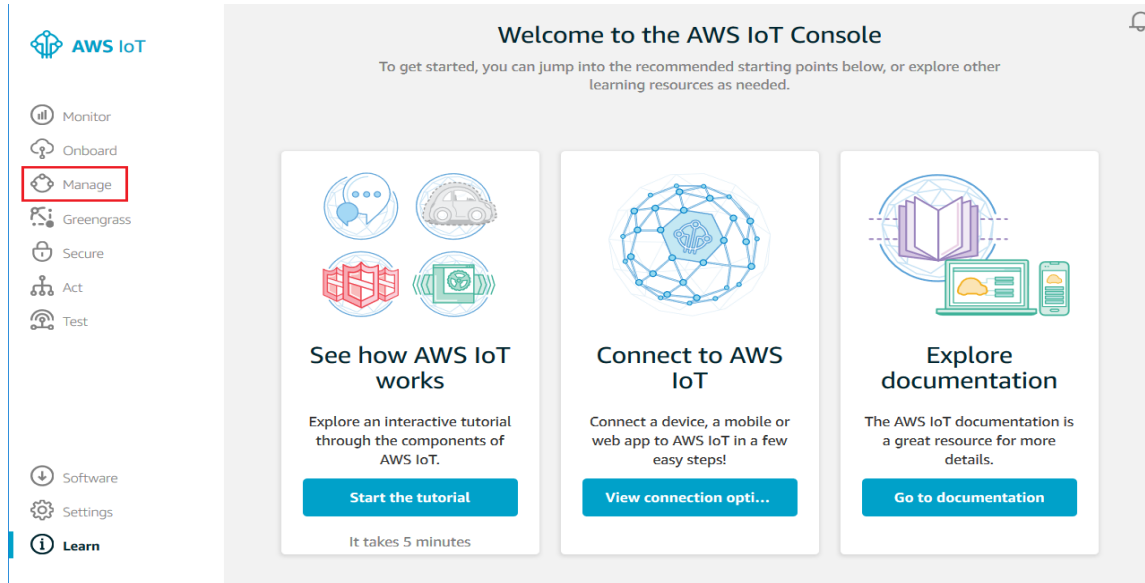


After that you close all arduino terminal.

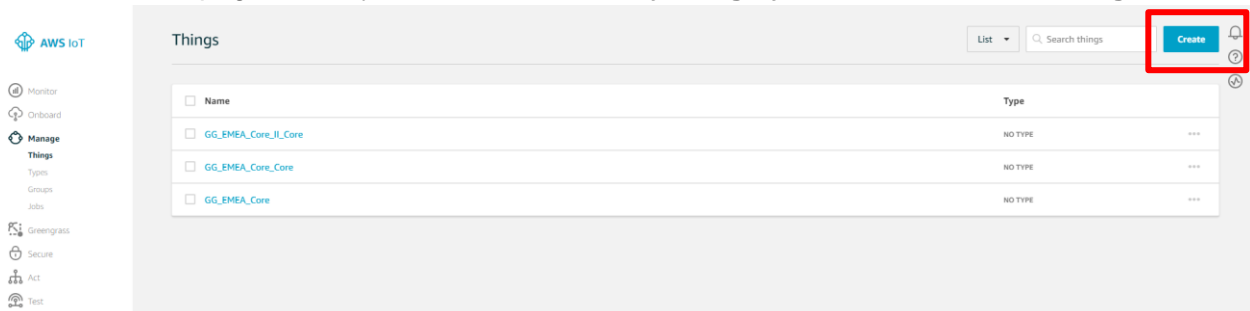
## Register a Device in the Thing Registry

To register your device in the thing registry:

1. On the **Welcome to the AWS IoT Console** page, in the left navigation pane, choose **Manage** to expand the choices, and then choose **Things** (as necessary).



2. On the page that says **You don't have any things yet**, choose **Create a thing**.



- On the **Register a thing** page, in the **Name** field, type a name for your device, such as **Moisture**. Choose **Next** to add your device to the thing registry.

CREATE A THING

STEP 1/3

Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name

Moisture

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected

Create a type

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

Groups /

Create group Change

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

| Attribute key                               | Value                                             |       |
|---------------------------------------------|---------------------------------------------------|-------|
| Provide an attribute key, e.g. Manufacturer | Provide an attribute value, e.g. Acme-Corporation | Clear |
| <div>Add another</div>                      |                                                   |       |

Show thing shadow ▼

Cancel

Back

Next

## Create and Activate a Device Certificate:

1. Next Screen allows you to add a certificate to your thing. Select “One-click certificate creation”

**CREATE A THING** STEP 2/5

### Add a certificate for your thing

A certificate is used to authenticate your device's connection to AWS IoT.

**One-click certificate creation (recommended)**  
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

[Create certificate](#)

**Create with CSR**  
Upload your own certificate signing request (CSR) based on a private key you own.

[Create with CSR](#)

**Use my certificate**  
Register your CA certificate and use your own certificates for one or many devices.

[Get started](#)

**Skip certificate and create thing**  
You will need to add a certificate to your thing later before your device can connect to AWS IoT.

[Create thing without certificate](#)

2. On the **Certificate created!** page, choose **Download** for the certificate, private key, and the root CA for AWS IoT (the public key need not be downloaded). Save each of them to your computer, and then choose **Activate** to continue.

**Certificate created!**

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

|                              |                        |                          |
|------------------------------|------------------------|--------------------------|
| A certificate for this thing | 2a540e2346.cert.pem    | <a href="#">Download</a> |
| A public key                 | 2a540e2346.public.key  | <a href="#">Download</a> |
| A private key                | 2a540e2346.private.key | <a href="#">Download</a> |

You also need to download a root CA for AWS IoT from Symantec:  
A root CA for AWS IoT [Download](#)

[Activate](#)

[Done](#) [Attach a policy](#)

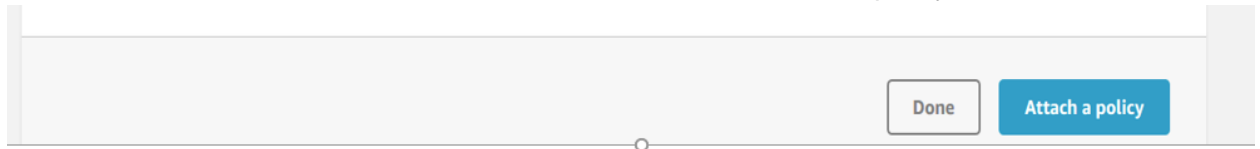
Be aware that the downloaded filenames may appear differently than those listed on the Certificate created! page. For example,

- 2a540e2346-certificate.pem.crt
- 2a540e2346-private.pem.key
- 2a540e2346-public.pem.key

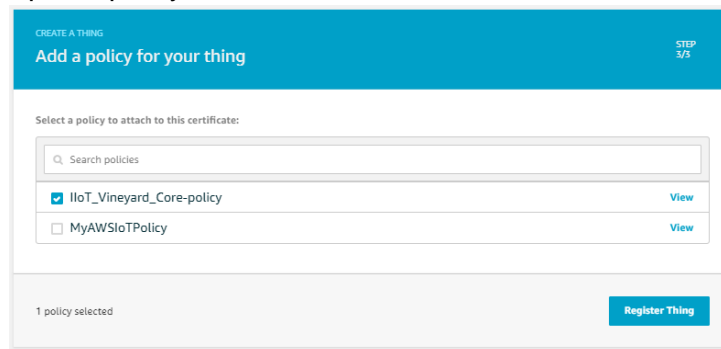
After Downloading all keys rename as,

- Moisture.cert.pem
- Moisture.private.key
- Moisture.public.key

And rename root certificate as **root-CA.crt**. Also click on “Attach a policy” button below:



Select your appropriate policy like i choosen below,

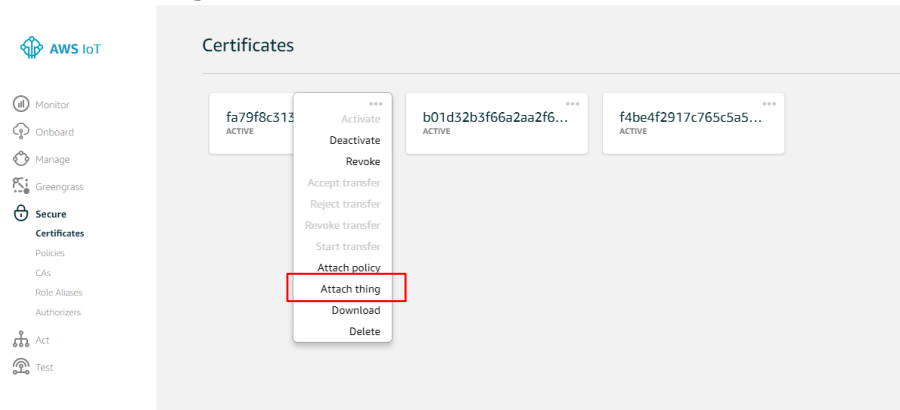


3. Choose “**Register Thing**”.

## Attach a Certificate to a Thing:

To attach a certificate to the thing representing your device in the thing registry,

1. In the box for the certificate you created, choose ... to open a drop-down menu, and then choose **Attach thing**.



2. In the **Attach things to certificate(s)** dialog box, select the check box next to the thing you registered, and then choose **Attach** (Your thing name will be **Moisture** instead of MyIoTButton).

### Attach things to certificate(s)

Things will be attached to the following certificate(s):  
fa79f8c3131887840cf9547d633f69a824136969766cbc2f3a7a3cbee1e2dcb1

Choose one or more things

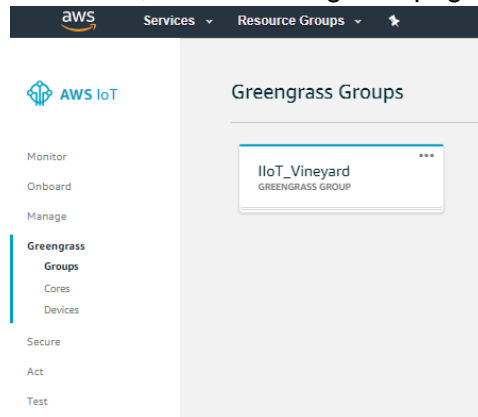
Search things

|                                     |                      |
|-------------------------------------|----------------------|
| <input type="checkbox"/>            | GG_EMEA_Core_II_Core |
| <input type="checkbox"/>            | GG_EMEA_Core_Core    |
| <input type="checkbox"/>            | GG_EMEA_Core         |
| <input checked="" type="checkbox"/> | Moisture             |

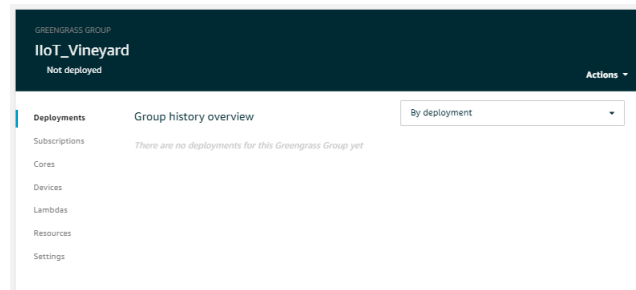
1 thing selected
Cancel
Attach

## Add Your device to a Greengrass Group:

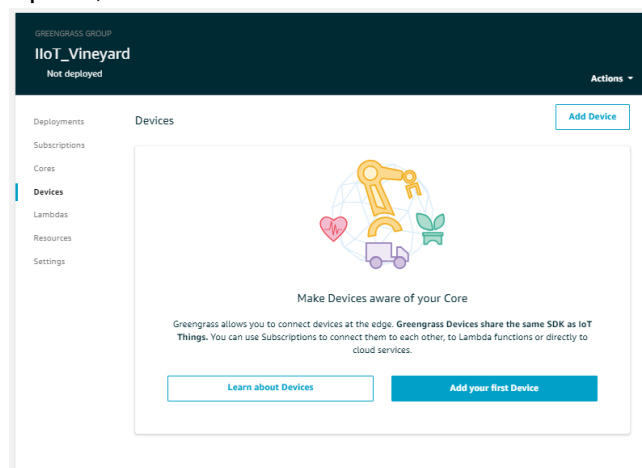
1. In the AWS Greengrass console, from the navigation page, choose **Groups**.



2. Choose your Greengrass Group.



3. In the navigation pane, choose **Device** and select **Add Device**.





4. Choose **Select an IoT Thing** and select your device name(**Moisture**) and choose **finish**.

ADD A DEVICE  
Add a Device

Greengrass Devices can be created by re-purposing an existing IoT Thing from your Registry or by creating new Registry items, and then adding them to a Greengrass Group.

Create a new Device

You will create a new Device and generate a certificate, a private key and a public key.

Create New Device

Use an existing IoT Thing as an Device

You can add an existing IoT Thing to your Group.

Select an IoT Thing

Cancel
Back
Create New Device

ADD A DEVICE  
Use an existing IoT Thing as a Device

You can select an existing AWS IoT Thing and import it into the Greengrass Group as a Device.

Select a Thing

Search Things

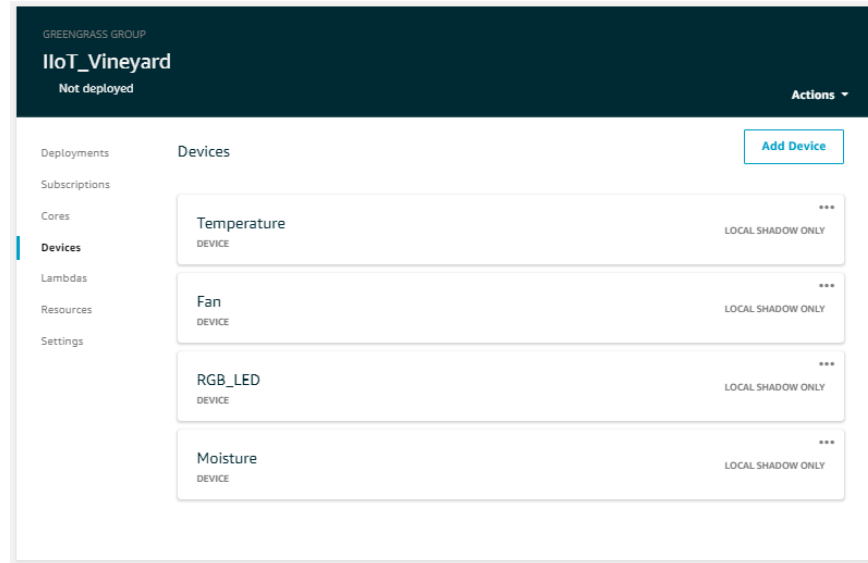
☐ IoT\_Vineyard\_Core
☒ Moisture

Cancel
Back
Finish

**NOTE:** Follow the above procedure (8.1 to 8.4) to create a thing (like Moisture) for Fan, Temperature, RGB\_LED

## Deploy all the device in Greengrass:

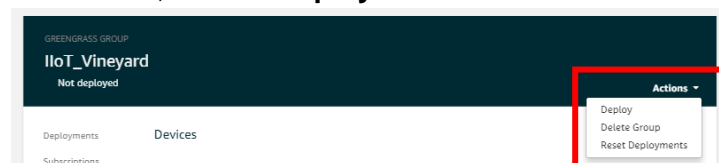
1. From AWS IoT console, go through **Greengrass** and choose **Groups** and select **GreenGrass\_Group**.
2. From GreenGrass\_Group, in navigation pane choose **Devices**.



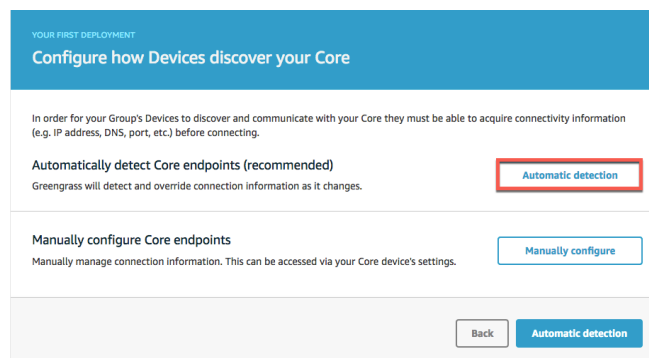
3. You will see all devices in GreenGrass\_Group.
4. Before Deploying, open a terminal application from your core device (Dragonboard 410c), Go to /greengrass directory and start your greengrass to run.

```
$ cd /greengrass
$ sudo ./greengrassd start
```

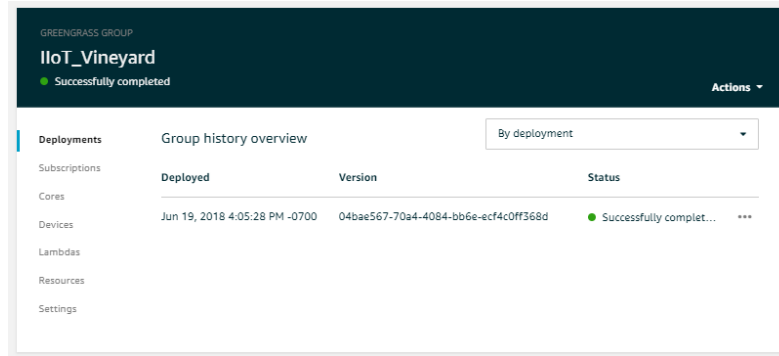
5. After running greengrass in core devices(DragonBoard410c).
6. From the **Actions** menu, choose **Deploy**.



7. On the **Configure how Devices discover your Core** page, choose **Automatic detection**.



- Your deployment might take a couple of minutes. You will know the deployment was successful when a Deployment successfully completed message is displayed in the group details page.

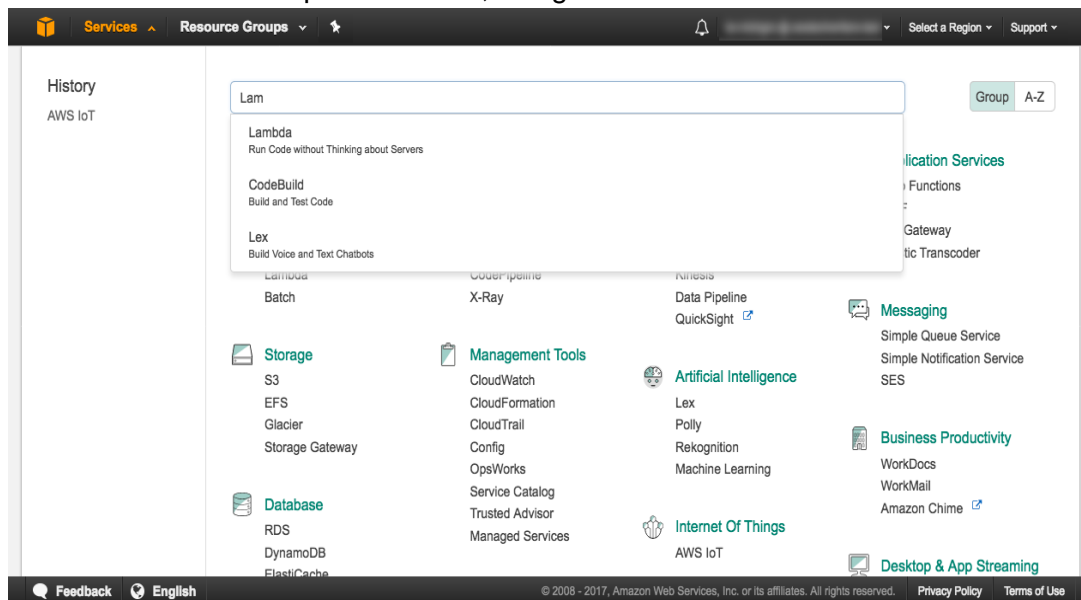


## Create Lambda Function

### Create a “Hello World” Lambda Function:

AWS Greengrass cores can run Lambda functions in response to messages sent by your devices (or other Lambda functions). You will now create a Lambda function that you will add to your AWS Greengrass group and then deploy it to your AWS Greengrass core.

- From the **Service** drop-down menu, navigate to the AWS Lambda console.



- In the Lambda console, choose **Create a Lambda function**.

COMPUTE

# AWS Lambda

lets you run code without thinking about servers.

You pay only for the compute time you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

**Get started**

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

**Create a function**



- In the **Filter** text box, type Greengrass, and then choose the **greengrass-hello-world** Lambda function blueprint for python.

The screenshot shows the AWS Lambda 'Create function' console. The 'Blueprints' tab is selected and highlighted with a red box. Below it, a search filter 'keyword: green' is applied. Two blueprints are listed: 'greengrass-hello-world' (python) and 'greengrass-hello-world-nodejs' (nodejs6.10). The 'greengrass-hello-world' blueprint is highlighted with a red box. The 'Serverless Application Repository' tab is also visible on the right.

- Choose **Configure**.

5. Scroll down until you see **Lambda function handler and role**. For **Role**, select **Choose an existing role**. For **Existing role**, select any role. If you don't have a role, select **Create a new role from template(s)**, and then choose any template from the **Policy templates** drop-down list. Choose **Next**.

Lambda > Functions > Create function > Using blueprint greengrass-hello-world

 This function contains external libraries. 


**Basic information** [Info](#)

Name


Role  
Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

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  
Enter a name for your new role.

 This new role will be scoped to the current function. To use it with other functions, you can modify it in the IAM console.

Policy templates  
Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.



6. On the **Review** page, choose **Create function**.

**Lambda function code**  
Code is pre-configured by the chosen blueprint. You can configure it after you create the function.

Runtime  
Python 2.7

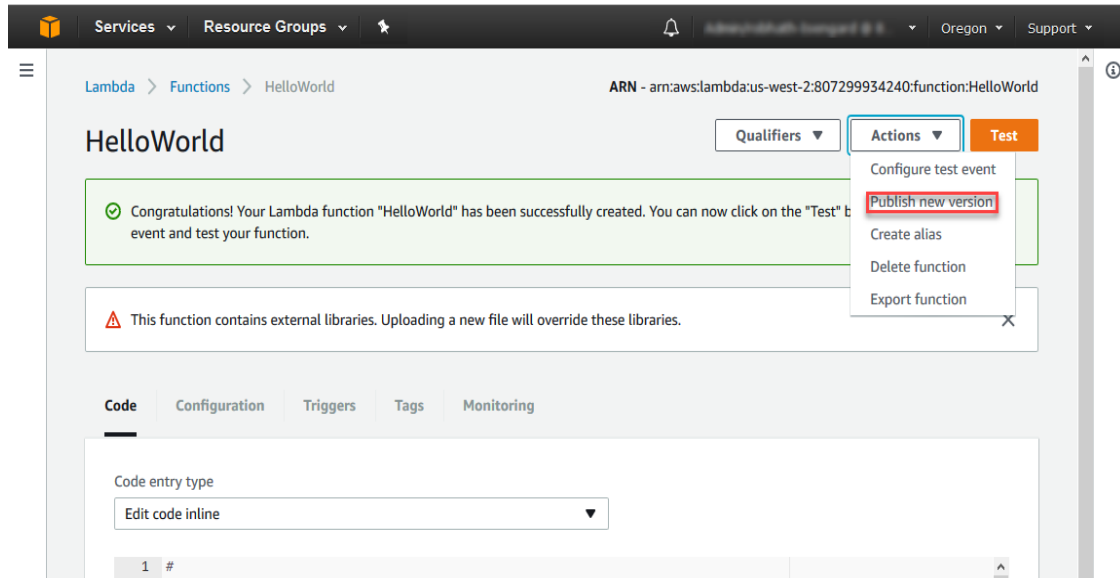
```

10 # Long-Lived it will run forever when deployed to a Greengrass core. The handler
11 # will NOT be invoked in our example since we are executing an infinite loop.
12
13 import greengrasssdk
14 import platform
15 from threading import Timer
16 import time
17
18 # Creating a greengrass core sdk client
19 client = greengrasssdk.client('iot-data')
20
21 # Retrieving platform information to send from Greengrass Core
22 my_platform = platform.platform()
23
24
25
26 # When deployed to a Greengrass core, this code will be executed immediately
27 # as a long-lived Lambda function. The code will enter the infinite while loop
28 # below.
29 # If you execute a 'test' on the Lambda Console, this test will fail by hitting the
30 # execution timeout of three seconds. This is expected as this function never returns
31 # a result.
32
33 def greengrass_hello_world_run():
34 if not my_platform:
35 client.publish(topic='hello/world', payload='Hello world! Sent from Greengrass Core.')
36 else:
37 client.publish(topic='hello/world', payload='Hello world! Sent from Greengrass Core running on platform')
38

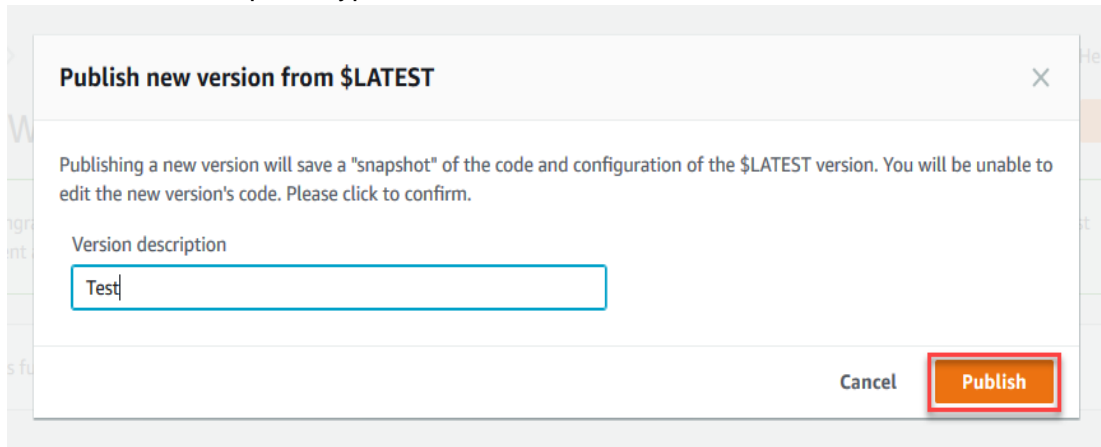
```

\* These fields are required.

- Now you copy the lambda function code from software package and need to publish a new version of the Hello World (or your function name) Lambda function and configure the handler (jiot\_event\_handler) function name in the handler text box. From the **Actions** menu, choose **Publish new version**.



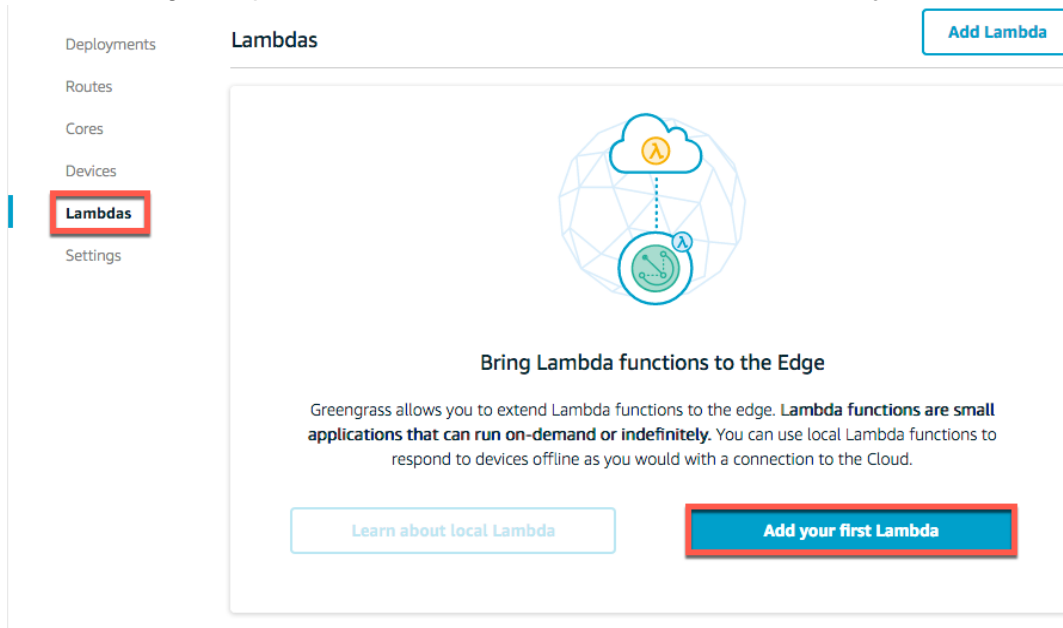
- In Version description, type Test, and then choose Publish.



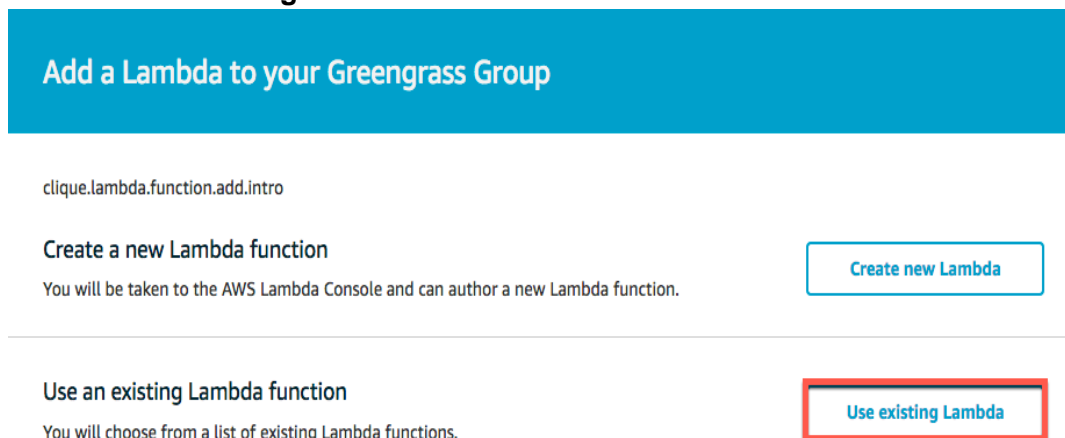
## Add the Lambda Function to Your Group Definition:

Next, you are going to add the Hello World Lambda function to your group definition. After it is deployed locally, that function sends data back to the AWS IoT platform and shows you have a deployed a functioning core.

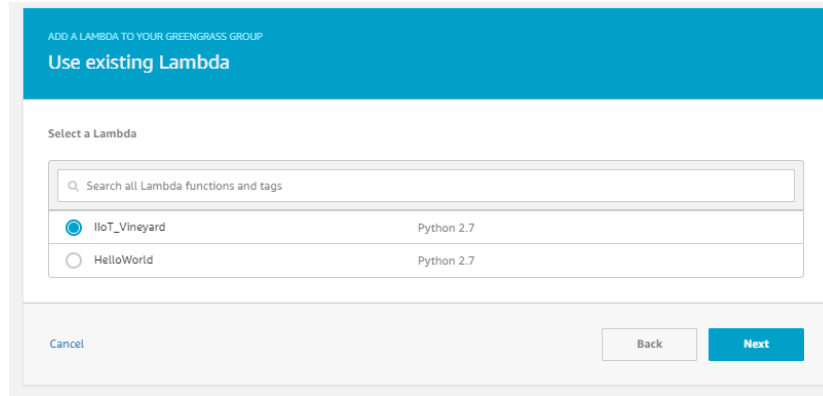
1. In the AWS IoT console, choose **Greengrass**, and then choose **Groups**.
2. Select the title for your group.
3. In the navigation pane, choose **Lambdas**, and then choose **Add your first Lambda**.



4. Choose **Use existing Lambda**.



5. Select the Lambda function you created earlier, and then choose **Next**. If you create many Lambda functions for Greengrass, you can organize them, and make searching for them easier, by using the Lambda function tagging feature. AWS Greengrass supports searching for Lambda functions by their tags in this console pane.



ADD A LAMBDA TO YOUR GREENGRASS GROUP

### Use existing Lambda

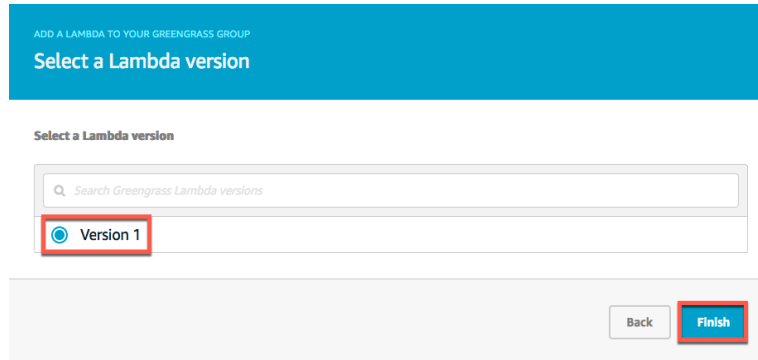
Select a Lambda

Search all Lambda functions and tags

|                                  |               |            |
|----------------------------------|---------------|------------|
| <input checked="" type="radio"/> | IloT_Vineyard | Python 2.7 |
| <input type="radio"/>            | HelloWorld    | Python 2.7 |

Cancel Back Next

6. Select the version of the Lambda function to use, and then choose **Finish**.



ADD A LAMBDA TO YOUR GREENGRASS GROUP

### Select a Lambda version

Select a Lambda version

Search Greengrass Lambda versions

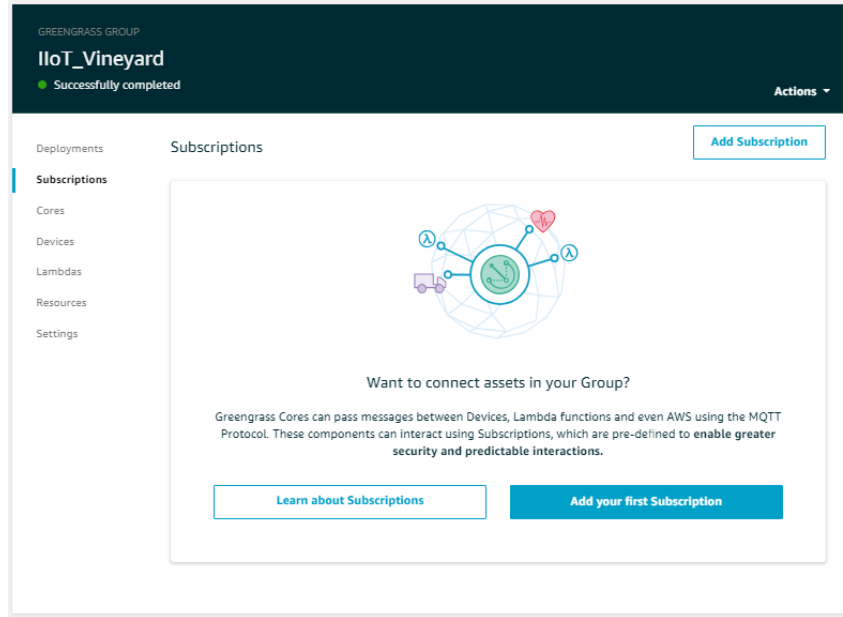
|                                  |           |
|----------------------------------|-----------|
| <input checked="" type="radio"/> | Version 1 |
|----------------------------------|-----------|

Back Finish

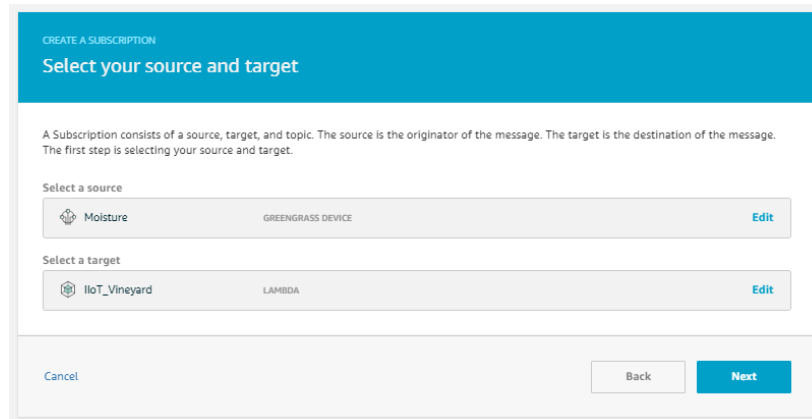


## Add Subscription to Your Group Definition:

1. In the AWS Greengrass console, find your group, and then select it.
2. On the group details page, choose **Subscriptions**.
3. Choose **Add your first Subscription**.



4. Under **Select a source**, choose your device name(**Moisture**).
5. Under **Select a target**, choose your created lambda function.
6. Choose **Next**.



7. In **Optional topic filter**, type **ggClient/moisture**, and choose **Next**.

The screenshot shows a web form titled "CREATE A SUBSCRIPTION" with the subtitle "Filter your data with a topic". Below the subtitle, there is explanatory text: "Your Source publishes data to your Target Asset. A topic filter can be used to constrain or control what data is sent to the Target. If you do not include a topic filter, all messages from the Source will be passed to the Target." The form has three main sections: "Source" with a dropdown menu showing "Moisture" (GREENGRASS DEVICE), "Optional topic filter" with a text input field containing "ggClient/moisture" and a link "How do I enter a topic filter?", and "Target" with a dropdown menu showing "IoT\_Vineyard" (LAMBDA). At the bottom right, there are "Back" and "Next" buttons.

8. Choose **Finish** to confirm and save your subscription.

The screenshot shows the same "CREATE A SUBSCRIPTION" form, but now it is titled "Confirm and save your Subscription". The explanatory text is: "Your Subscription is complete and your objects are connected in this Group. You can now save, and then deploy your new Group definition to have this change take effect." The form fields are the same as in the previous screenshot, but the "Optional topic filter" field is now a read-only display. At the bottom right, there are "Back" and "Finish" buttons.

9. Repeat steps from 3 to 8 to add subscriptions for Fan, RGB\_LED and Temperature, Based on the below configuration.

| Source               | Target               | topic-filter         |
|----------------------|----------------------|----------------------|
| Moisture             | your lambda-function | ggClient/moisture    |
| your lambda-function | RGB_LED              | ggClient/rgblisten   |
| Temperature          | your lambda-function | ggClient/temperature |
| your lambda-function | Fan                  | ggClient/fanlisten   |

If you add all, the following figure shows look as given below:

| Source          | Target          | Topic                |     |
|-----------------|-----------------|----------------------|-----|
| Moisture        | IIoT_Vineyard:2 | ggClient/moisture    | ... |
| Temperature     | IIoT_Vineyard:2 | ggClient/temperature | ... |
| IIoT_Vineyard:2 | Fan             | ggClient/fanlisten   | ... |
| IIoT_Vineyard:2 | RGB_LED         | ggClient/rgblisten   | ... |

## Deploy your Group:

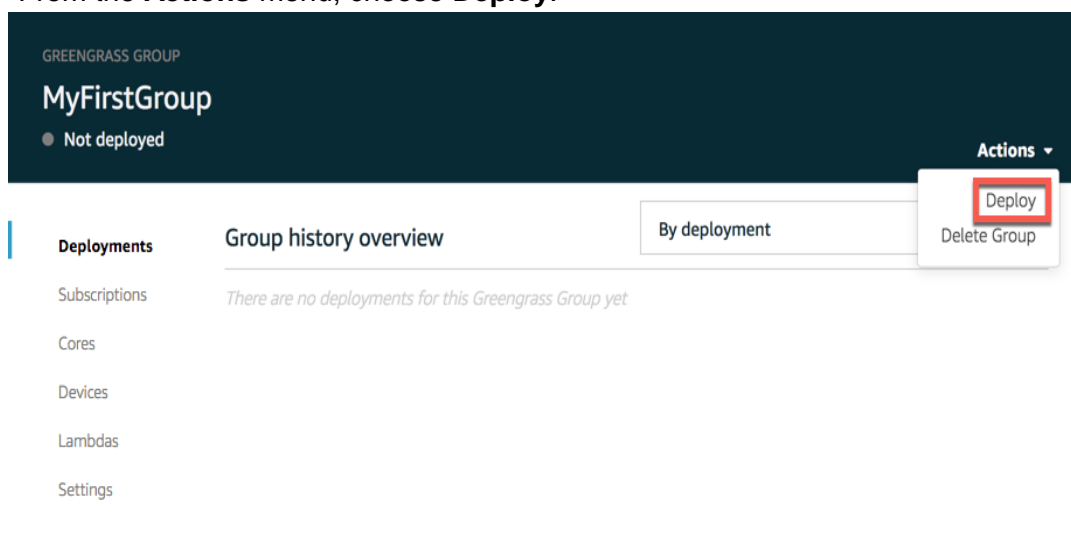
You have created a group and a core definition, but this information exists only in the cloud. Deploying a group takes this configuration information and copies it onto your AWS Greengrass core device.

1. In the AWS Greengrass console, choose **Groups**, choose your group, and then choose **Deployments**.

**NOTE:** Before Deployment, open a terminal application from your core device (Dragonboard 410c), Go to /greengrass directory and start your greengrass to run.

```
$ cd /greengrass
$ sudo ./greengrassd start
```

2. From the **Actions** menu, choose **Deploy**.



- On the **Configure how Devices discover your Core** page, choose **Automatic detection**.

YOUR FIRST DEPLOYMENT

## Configure how Devices discover your Core

In order for your Group's Devices to discover and communicate with your Core they must be able to acquire connectivity information (e.g. IP address, DNS, port, etc.) before connecting.

### Automatically detect Core endpoints (recommended)

Greengrass will detect and override connection information as it changes.

Automatic detection

---

### Manually configure Core endpoints

Manually manage connection information. This can be accessed via your Core device's settings.

Manually configure

Back

Automatic detection

- On the **Grant permission to access other services** page, choose **Grant permission**.

YOUR FIRST DEPLOYMENT

## Grant permission to access other services

**Why does Greengrass require a service-level role permission to function?**

AWS Greengrass service works with other services like AWS IoT and AWS Lambda. In order to function properly Greengrass needs your permission to access these other services and read and write data on your behalf. You can also see [the specifics of this managed policy](#)

You only need to complete this step once for all groups. You can modify the service-level role permission in the IoT settings.

Back

Grant permission

Your deployment might take a couple of minutes. You will know the deployment was successful when a Deployment successfully completed message is displayed in the group details page:

GREENGRASS GROUP

### IIoT\_Vineyard

● Successfully completed Actions ▾

**Deployments**

Subscriptions

Cores

Devices

Lambdas

Resources

Settings

Group history overview By deployment ▾

| Deployed                      | Version                              | Status                        |
|-------------------------------|--------------------------------------|-------------------------------|
| Jun 19, 2018 4:53:52 PM -0700 | c70d4917-03ca-4562-8c02-fd93db28c32d | ● Successfully complet... *** |
| Jun 19, 2018 4:05:28 PM -0700 | 04bae567-70a4-4084-bb6e-ecf4c0ff368d | ● Successfully complet... *** |

## Installation of necessary packages

### Installation of node.js6x:

From your terminal in DragonBoard 410c type below command to install in root user,

```
$ sudo su -
$ curl -sL https://deb.nodesource.com/setup_6.x | bash -
$ apt-get install -y nodejs
```

### Installation of aws-iot-device-sdk for python:

From your terminal in DragonBoard 410c type below command to install,

```
$ git clone https://github.com/aws/aws-iot-device-sdk-python.git
$ cd aws-iot-device-sdk-python
$ python setup.py install
```

### Installation of smbus:

From your terminal in DragonBoard 410c type below command to install,

```
$ sudo apt-get install python-smbus
```

**NOTE:** For configuring and starting the sensor and web application please, refer to the GitHub repository:

<https://github.com/globaledgesoft/smartvine-greengrass>