Table of Contents

1. Document information

Note that all instructions in this document have been written assuming a KR260 Ubuntu Edge device.

1.1 Document revision history

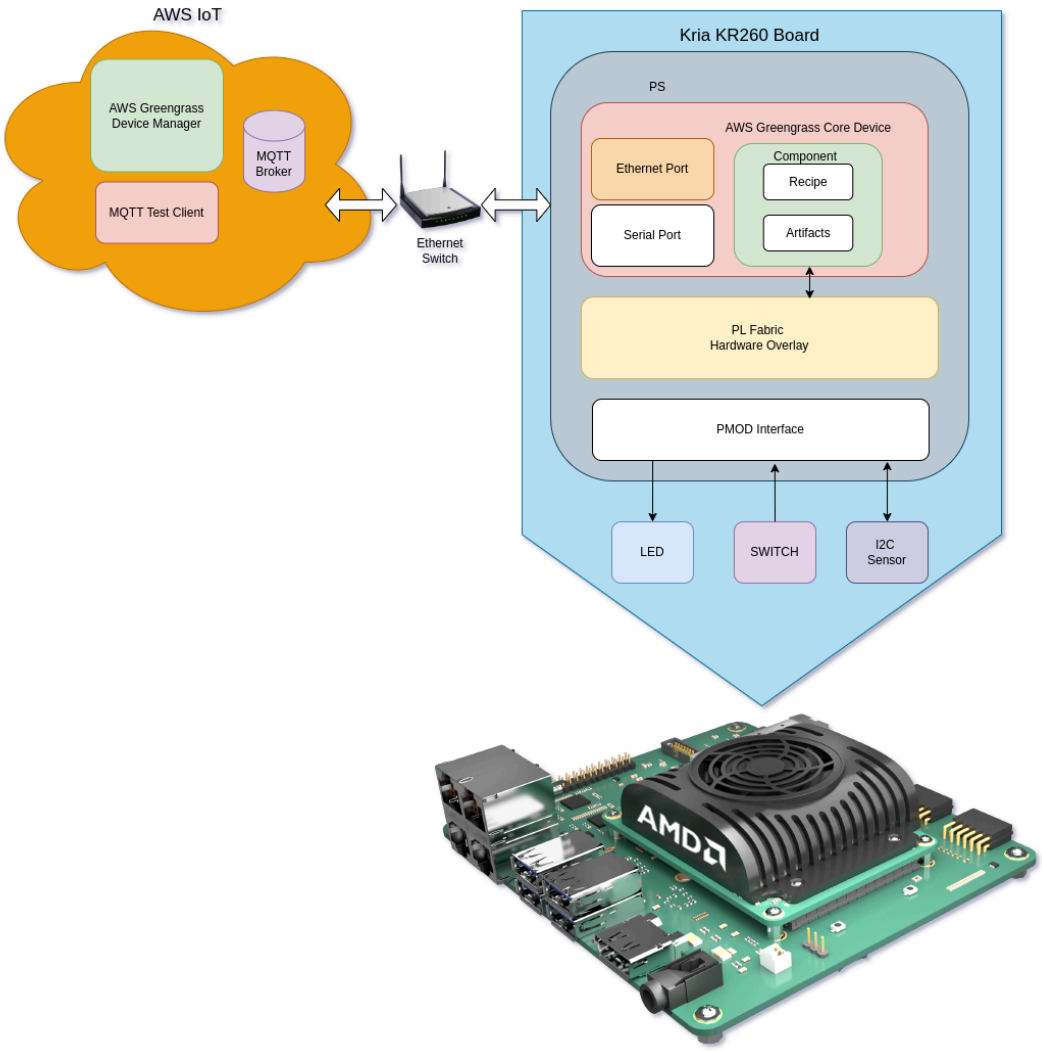
30-Dec-2023	Initial Draft
27-Feb-2024	Updated Document Format

2. Overview

AMD Kria SoM is Zynq® UltraScale+™ MPSoC architecture based FPGA platform, which is suitable for vision AI solutions for edge applications.

AMD KR260 is Kria SoM based Robotics Starter Kit, which is suitable for Robotics, Industrial Control and Machine Learning inference based applications. By using PMOD and RPi based header, we can interface various sensors and publish data to the cloud platforms.

Following diagram shows the software and hardware architecture used in this getting started guide. Kria KR260 board consists of PL Fabric(FPGA) hardware overlay for interfacing LED, switch and I2C sensor. Further it runs AWS Greengrass Core Device Application which publish and subscribe message topics for actuating LED and monitoring sensors and switches. From AWS IoT MQTT Test Client KR260 LED will be controlled through subscribed topic and also publish Switch pressed event to AWS IoT cloud.



3. Hardware description

3.1 Datasheet

Refer to the Kria K26 SOM Product Brief and SOM Data Sheet at product documentation at the K26 Documentation page.
For a view of the hardware portfolio, refer to this Portfolio Comparison.

3.2 Standard kit contents

Details regarding the KR260 kit contents are provided here.
Refer to the section What's Inside the Box in the online documentation here.

3.3 User provided items

Refer to the section What You'll Need to Provide in the online documentation here.

Also see the recommended accessories the user can purchase.

3.4 3rd party purchasable items

Not applicable.

3.5 Additional References

Getting started guide for KR260 board is available here.
Refer to the all the resources related to the Kria™ KR260 Robotics Starter Kit on the GitHub page
Read the Kria KR260 Robotics Starter Kit User Guide and Data sheet

4. Set up your development environment

4.1 Tools installation (IDEs, Toolchains, SDKs)

Not Applicable

5. Set up device hardware

The KR260 board boots off an SD card. To create this SD card, refer to the instructions at Setting up the SD card image.
The instructions to set up and connect the board are available here.
Follow the instructions here to boot and monitor your board.

5.1 Installing hardware overlay

Get the KR260 firmware folder. It contains:

- kr260_i2c.bit.bin
- kr260_i2c.dtbo
- shell.json

Copy these file to the KR260 board. For firmware to be loaded using xmutil (FPGA manager), one has to copy these file at “/lib/firmware/xilinx”.
For this create the folder at “kr260-i2c” at “/lib/firmware/xilinx” and copy the files in “kr260-i2c” folder.

```
cd /lib/firmware/xilinx
sudo mkdir kr260-i2c
sudo cp <kr260-firmware directory>/krc260_i2c* ./
sudo cp <kr260-firmware directory>/shell.json ./
```

Next, check the available fpga firmware using `xmutil listapps` command. `kr260-i2c` will be available in the list.

```
ubuntu@kria:~$ sudo xmutil listapps
[sudo] password for ubuntu:
                  Accelerator         Accel_type                          Base          Base_type      #slots(PL+AIE)        Active_slot

                    kr260-i2c           XRT_FLAT                     kr260-i2c           XRT_FLAT             (0+0)                 -1
               k26-starter-kits           XRT_FLAT               k26-starter-kits           XRT_FLAT             (0+0)                 0,
ubuntu@kria:~$
```

Next load the `kr260-i2c` firmware, which contains necessary hardwares (gpio) and interfaces. In our Greengrass Demo we will be using these gpio to trigger the publishing data to AWS Greengrass IoT cloud server and also actuate GPIO on the message received from AWS cloud.

```
sudo xmutil unloadapp
sudo xmutil loadapp kr260-i2c
```

```
ubuntu@kria:~$ sudo xmutil unloadapp
remove from slot 0 returns: 0 (Ok)
ubuntu@kria:~$ sudo xmutil loadapp kr260-i2c
[ 1035.828900] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
[ 1035.839040] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/pid
[ 1035.848277] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
[ 1035.857771] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/uid
[ 1035.867399] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay0
[ 1035.877241] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay1
[ 1035.887085] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
[ 1035.896579] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
[ 1035.906509] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking1
[ 1035.916438] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/overlay2
[ 1035.926280] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
[ 1035.936329] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/misc_clk_0
[ 1035.946346] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
[ 1035.956281] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/misc_clk_1
[ 1035.966299] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_1
[ 1035.976227] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
[ 1035.986243] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
[ 1036.067970] xiic-i2c 80020000.i2c: IRQ index 0 not found
kr260-i2c: loaded to slot 0
ubuntu@kria:~$ [ 1036.203709] zocl-drm axi:zyxclmm_drm: IRQ index 32 not found
```

Now to access GPIO in user application, we will be using `gpiod` library.

5.2 Installing gpiod packages

GPIOD packages are required to access the GPIO channels. It also provides python binding for accessing GPIO in python programming. Install the package using apt-get:

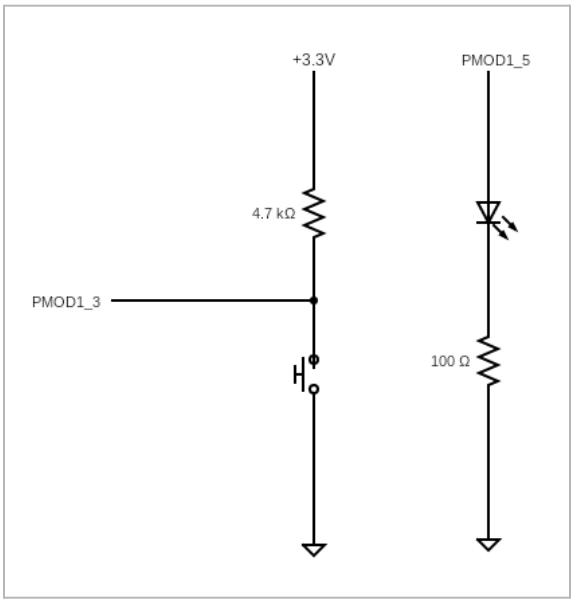
```
sudo apt-get install gpiod python3-libgpiod
```

Now we can check the available gpio using gpiod applications:

Using `gpiodetect` to get availabe gpio:

```
ubuntu@kria:~$ sudo gpiodetect
gpiochip0 [firmware:zynqmp-firmware:gpio] (4 lines)
gpiochip1 [zynqmp_gpio] (174 lines)
gpiochip2 [slg7xl45106] (8 lines)
gpiochip3 [80010000.gpio] (6 lines)
ubuntu@kria:~$
```

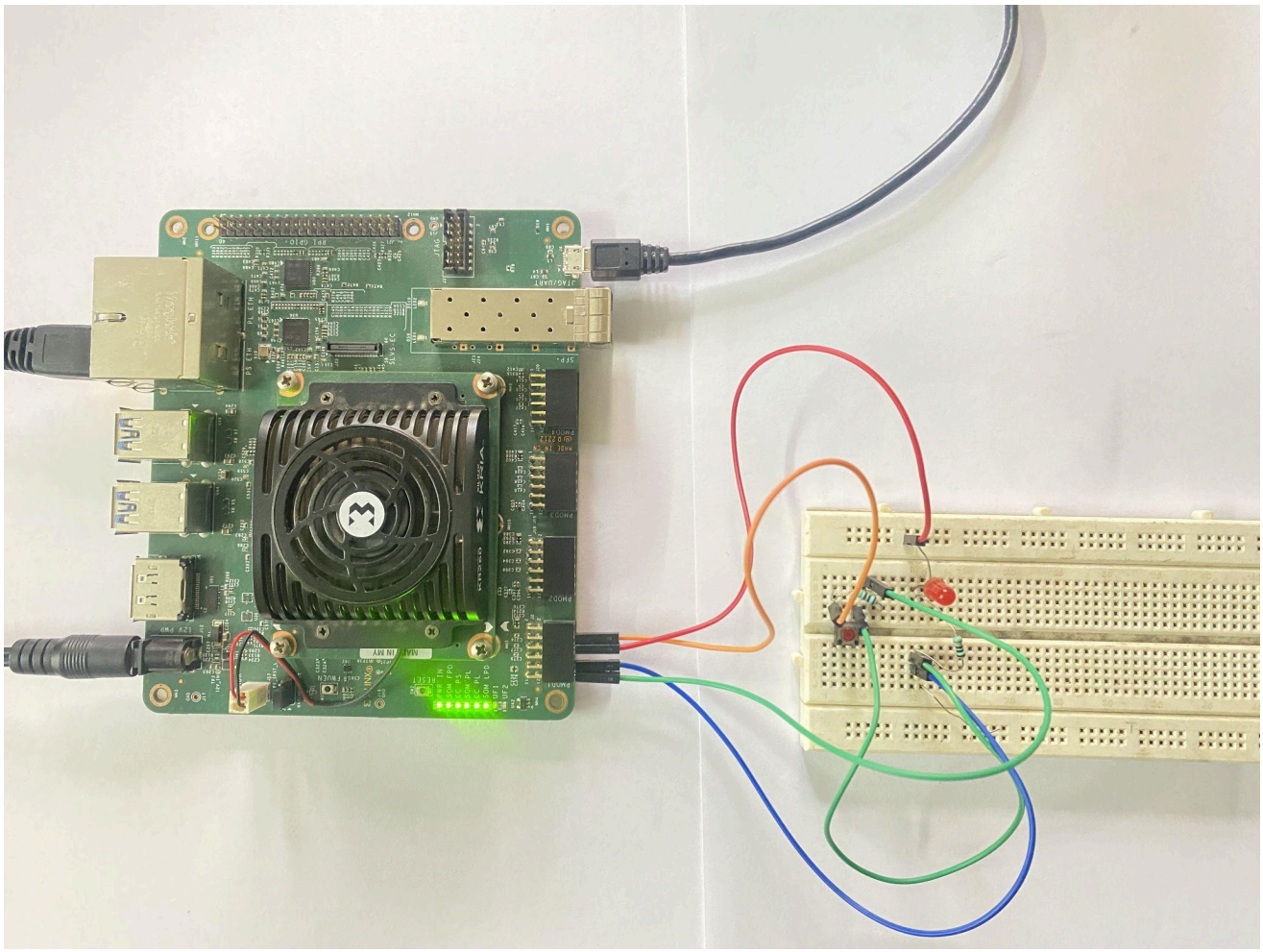
Here `gpiochip3` is the device corresponding to gpio in FPGA and it consists of 6 lines. Further these gpio lines are connected to PMOD 1 such that:
PMOD1-> 1 - gpiochip3 line 0 (Not in use)
PMOD1-> 3 - gpiochip3 line 1
PMOD1-> 5 - gpiochip3 line 2



Schematic for LED and Switch Connection

11	9	7	5	3	1	PMOD UPPER
12	10	8	6	4	2	PMOD LOWER
Vcc	GND	I/O	I/O	I/O	I/O	

PMOD port numbering



KR260 Board Connected to switch and LED through PMOD1

6. About AWS IoT Greengrass

To learn more about AWS IoT GreengrassV2, see how it works and what's new.

7. Greengrass prerequisites

Refer to the online documentation detailing the prerequisites needed for AWS IoT Greengrass.
Follow the instructions in the following sections:

- Step 1: Set up an AWS account
- Step 2: Set up your environment

8. Install AWS IoT Greengrass

Refer to the instructions in the following steps:

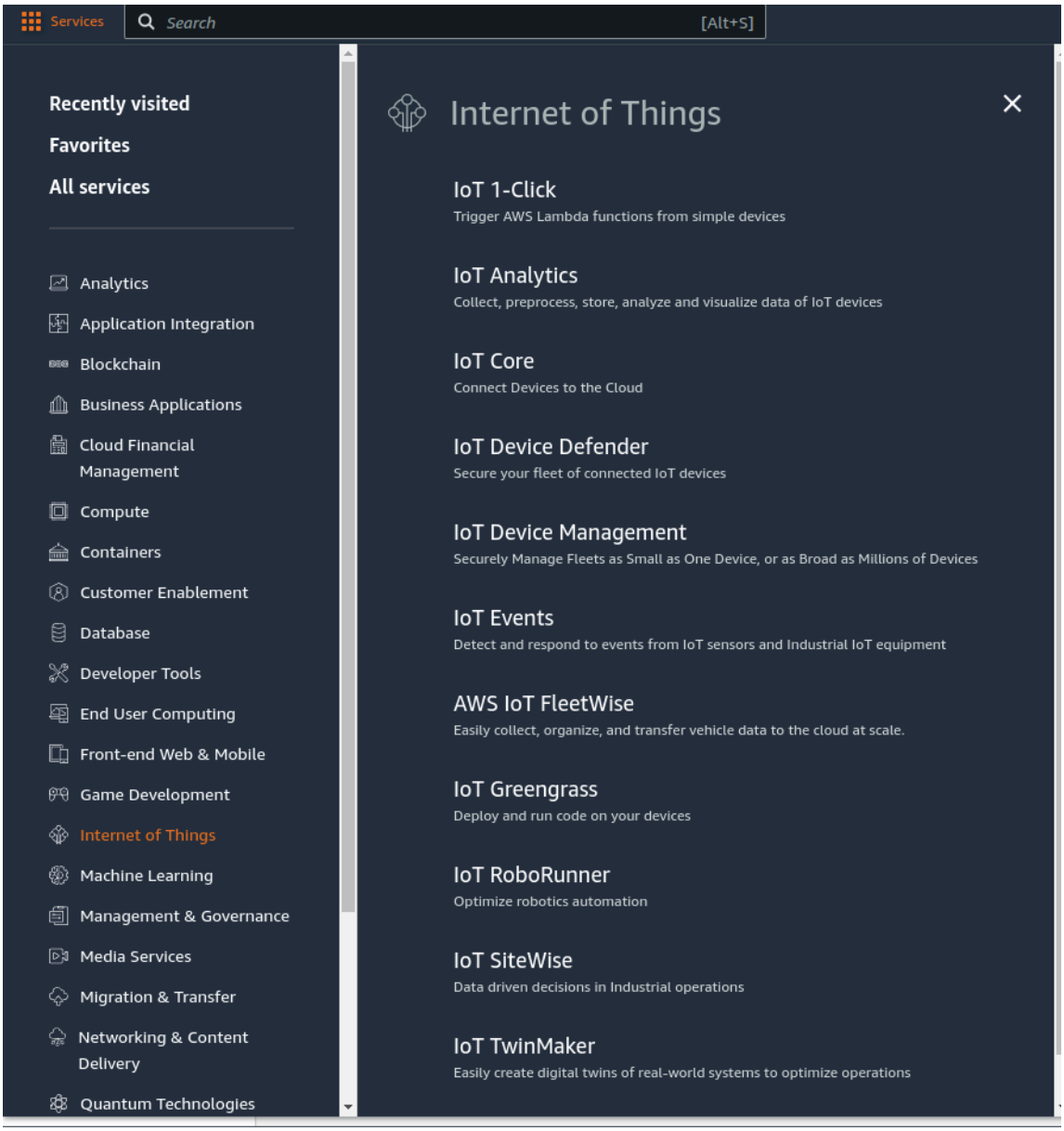
- Set up the device environment
- Provide AWS credentials to the device. For development environments, you can use the option “Use long-term credentials from an IAM User”. An example of how to do this is shown below:

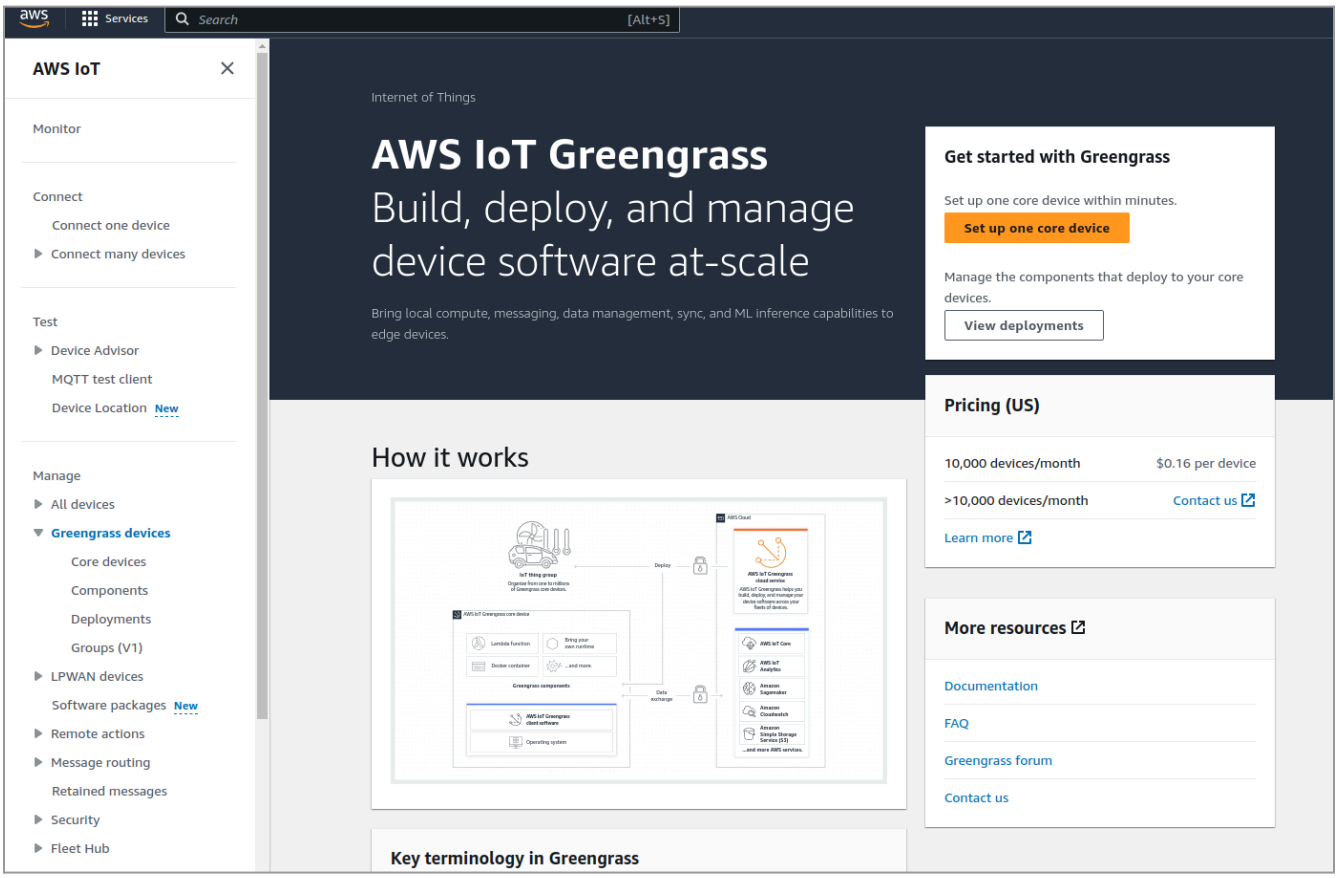
```
export AWS_ACCESS_KEY_ID=<the access key id for your user>
export AWS_SECRET_ACCESS_KEY=<the secret access key for your user>
```

- Download the AWS IoT Greengrass Core software
- Install the AWS IoT Greengrass Core software (console)
- Install the AWS IoT Greengrass Core software

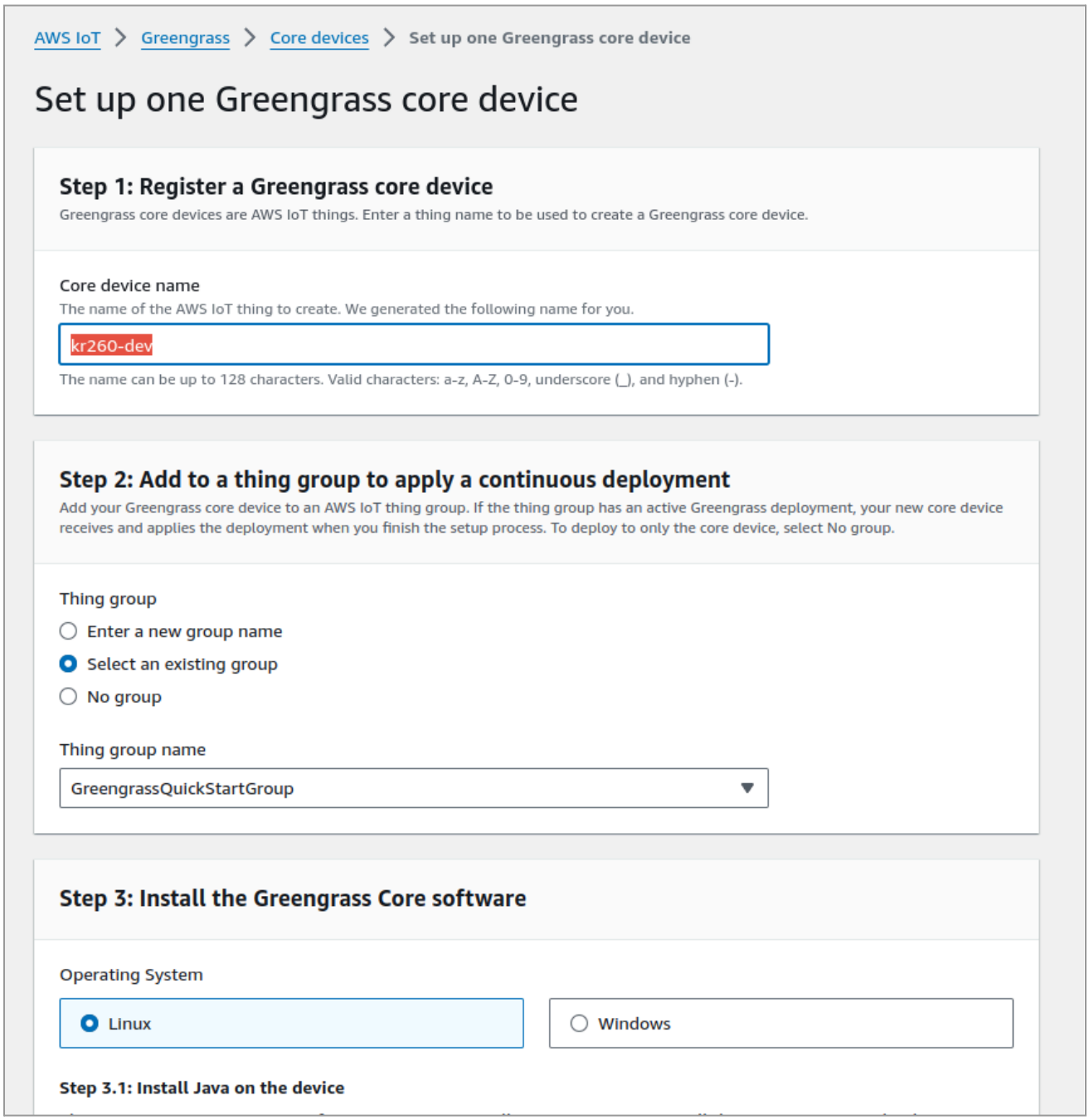
8.1 Steps detail for Installing AWS IoT Greengrass on KR260

Steps and scripts for installing greengrass device is provided by AWS Greengrass dashboard in AWS web console. So first access the AWS Greengrass IoT page, go to AWS Services -> Internet of Things -> IoT Greengrass link





Now click on “Set up one core device” button
This will open the Greengrass core device setup page:
Here you change the Core device name like `kr260-dev”



Now in KR260 terminal console run following commands and scripts:

```
export AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
export AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
```

Greengrass CLI depends on Java. So to install the dependency run the following:

```
sudo apt install default-jre
sudo apt install default-jdk
```

Download and install Greengrass core software.

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip > greengrass-nucleus-latest.zip && unzip greengrass-nucleus-latest.zip -d GreengrassInstaller
```

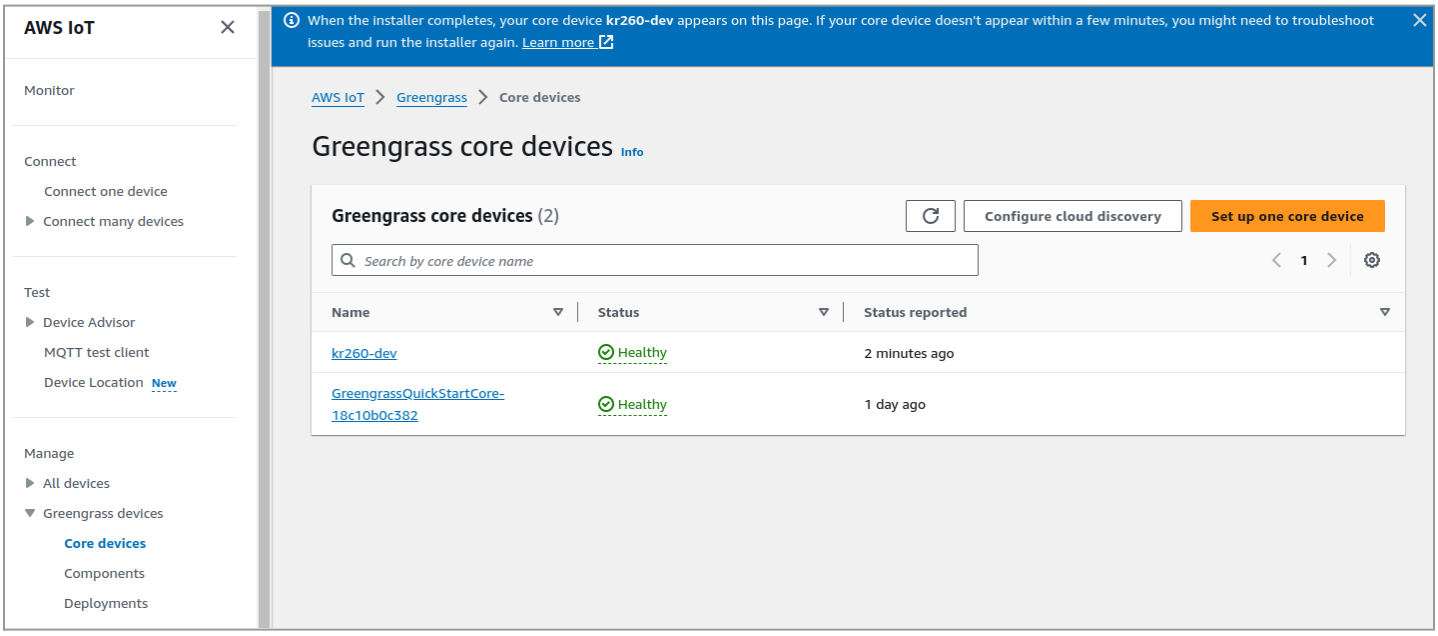
Next install the Greengrass core device:

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE -jar ./GreengrassInstaller/lib/Greengrass.jar --aws-region us-east-1 --thing-name kr260-dev --thing-group-name GreengrassQuickStartGroup --component-default-user ggc_user:ggc_group --provision true --setup-system-service true --deploy-dev-tools true
```

Here is the console log after running above command:

```
Provisioning AWS IoT resources for the device with IoT Thing Name: [kr260-dev]...
Found IoT policy "GreengrassV2IoTThingPolicy", reusing it
Creating keys and certificate...
Attaching policy to certificate...
Creating IoT Thing "kr260-dev"...
Attaching certificate to IoT thing...
Successfully provisioned AWS IoT resources for the device with IoT Thing Name: [kr260-dev]!
Adding IoT Thing [kr260-dev] into Thing Group: [GreengrassQuickStartGroup]...
IoT Thing Group "GreengrassQuickStartGroup" already existed, reusing it
Successfully added Thing into Thing Group: [GreengrassQuickStartGroup]
Setting up resources for aws.greengrass.TokenExchangeService ...
Attaching TES role policy to IoT thing...
No managed IAM policy found, looking for user defined policy...
IAM policy named "GreengrassV2TokenExchangeRoleAccess" already exists. Please attach it to the IAM role if not already
Configuring Nucleus with provisioned resource details...
Root CA file found at "/greengrass/v2/rootCA.pem". Contents will be preserved.
Downloading Root CA from "https://www.amazontrust.com/repository/AmazonRootCA1.pem"
Created device configuration
Successfully configured Nucleus with provisioned resource details!
Thing group exists, it could have existing deployment and devices, hence NOT creating deployment for Greengrass first party dev tools, please manually create a deployment if you wish to
Successfully set up Nucleus as a system service
ubuntu@kria:~$
```

Now in Greengrass set up page, one can view the Greengrass core devices and find above `kr260-dev` in the list.



In KR260 terminal one can get the device components by using `greengrass-cli`:

```
sudo /greengrass/v2/bin/greengrass-cli component list
```

```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: aws.greengrass.Nucleus
    Version: 2.12.0
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fipsMode":"false","fleetStatus":{"periodicSt
atusPublishIntervalSeconds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"cluwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.co
m","iotDataEndpoint":"a9jc3obcutf8v-ats.iot.us-east-1.amazonaws.com","iotRoleAlias":"GreengrassV2TokenExchangeRoleAlias","jvmOptions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"ne
tworkProxy":{"proxy":{}},"platformOverride":{},"runWithDefault":{"posixShell":"sh","posixUser":"ggc_user:ggc_group"},"s3EndpointType":"GLOBAL","telemetry":{}}
Component Name: FleetStatusService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Cli
    Version: 2.12.0
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
```

We will be adding component to publish and subscribe the topic to the AWS cloud Broker.

9. Installing the “mqtt-gpio” component

9.1 Get the component

Get the `components` folder from the downloaded resources and copy in the KR260 home directory.
It contains:

Artifacts

- com.example.mqtt
  - 1.0.0
    - mqtt.py (This python code published the data on button press and actuates gpio on receiving the data in subscribed topic)

Recipe

- com.example.mqtt-1.0.0.json

9.2 Deploy the component

To install the above component run the following in the KR260 terminal:

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.mqtt=1.0.0"
```

```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.mqtt=1.0.0"
Local deployment submitted! Deployment Id: 9e8f1be6-63b2-4189-aecc-607197755d22
ubuntu@kria:~$
```

9.3 Upload the component

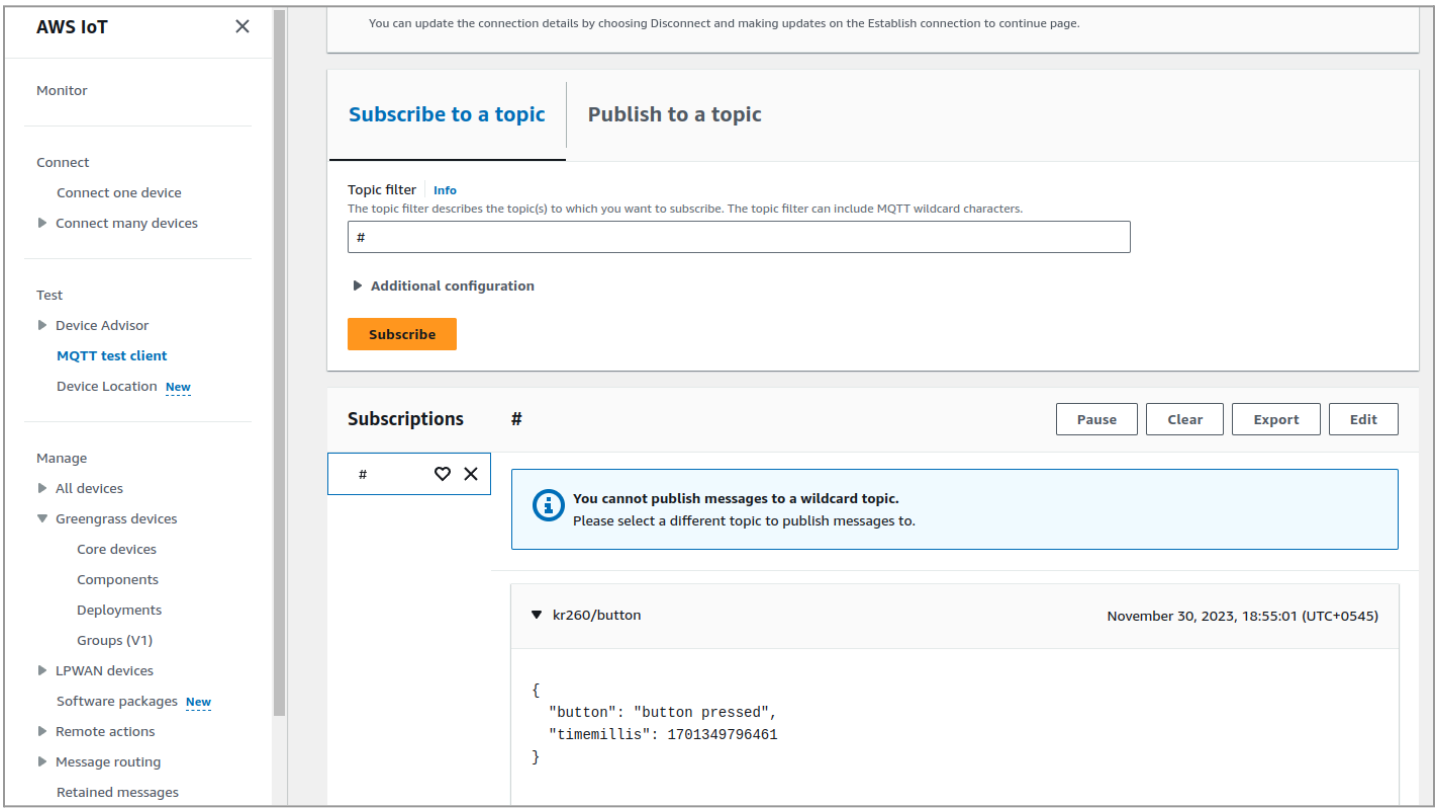
Follow the instructions online at Create your component in the AWS IoT Greengrass service to upload your component to the cloud, where it can be deployed to other devices as needed.

10. Debugging

Now check the installed component is in “running state”

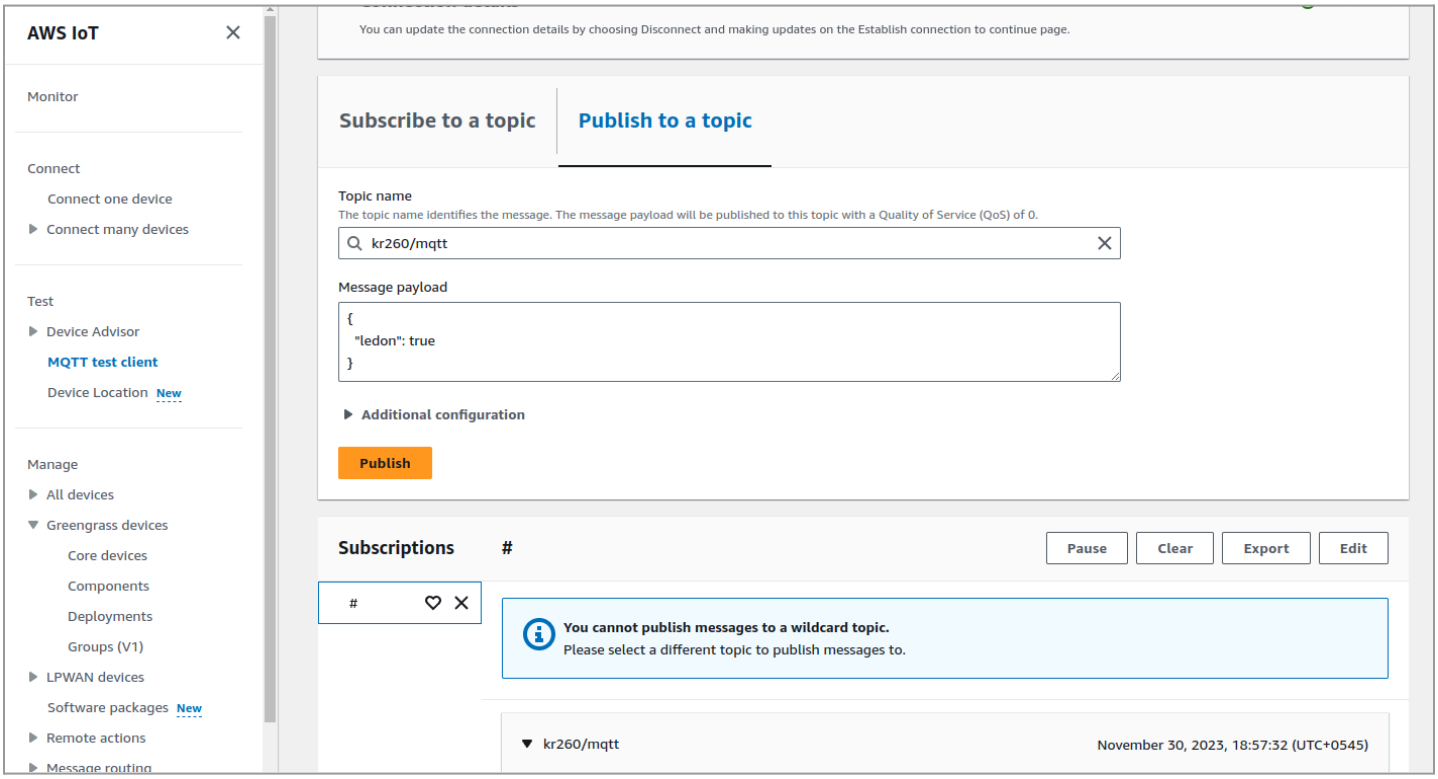
```
ubuntu@krla:~$ sudo /greengrass/v2/bin/greengrass-cli component list
Components currently running in Greengrass:
Component Name: aws.greengrass.Nucleus
    Version: 2.12.0
    State: FINISHED
    Configuration: {"awsRegion":"us-east-1","componentStoreMaxSizeBytes":"10000000000","deploymentPollingFrequencySeconds":"15","envStage":"prod","fipsMode":"false","fleetStatus":{"periodicStatusPublishIntervalSeconds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlanePort":"8443","httpClient":{},"iotCredEndpoint":"c1uwyavs4wpvxg.credentials.iot.us-east-1.amazonaws.com","iotDataEndpoint":"a9jc3obcutf8v-ats.iot.us-east-1.amazonaws.com","iotRoleAlias":"GreengrassV2TokenExchangeRoleAlias","jvmOptions":"-Dlog.store=FILE","logging":{},"mqtt":{"spooler":{}},"networkProxy":{"proxy":{}},"platformOverride":{},"runWithDefault":{"posixShell":"sh","posixUser":"ggc_user:ggc_group"},"s3EndpointType":"GLOBAL","telemetry":{}}
Component Name: com.example.mqtt
    Version: 1.0.0
    State: RUNNING
    Configuration: {"accessControl":{"aws.greengrass.ipc.mqttproxy":{"com.example.mqtt:mqttproxy:1":{"operations":["aws.greengrass#PublishToIoTCore","aws.greengrass#SubscribeToIoTCore"],"policyDescription":"Allows access to pub/sub to mypi/mqtt.","resources":["kr260/mqtt","kr260/button"]}}},"message":"hello"}
Component Name: TelemetryAgent
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: UpdateSystemPolicyService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: FleetStatusService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: aws.greengrass.Cli
    Version: 2.12.0
    State: RUNNING
    Configuration: {"AuthorizedPosixGroups":null,"AuthorizedWindowsGroups":null}
Component Name: aws.greengrass.LocalDebugConsole
    Version: 2.4.1
    State: RUNNING
    Configuration: {"bindHostname":"localhost","httpsEnabled":"true","port":"1441","websocketPort":"1442"}
```

Now in aws IoT console, open “MQTT test client” and subscribe to “#”

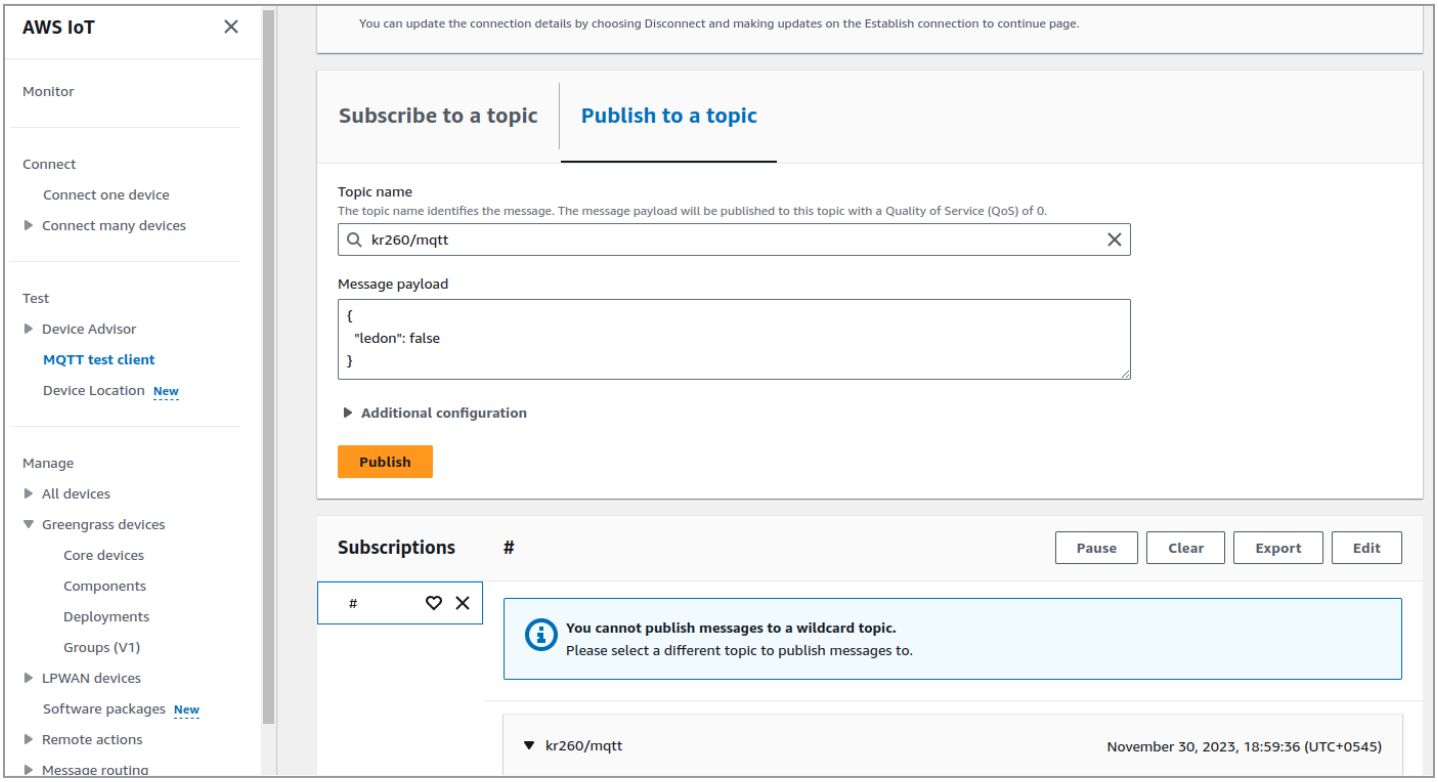


You can see the “button pressed” message once the button is pressed.

Now to control the LED, publish the message to “kr260/mqtt” topic. Here is the screenshot of the message which switch on the LED.



Now to switch off the LED send “false” message in the “kr260/mqtt” topic.



11. Troubleshooting

For more information, refer to the online documentation *Troubleshooting Greengrass v2*.

You can also refer to Logging and Monitoring to learn how to log API calls, gather system health telemetry data, and check core device status.