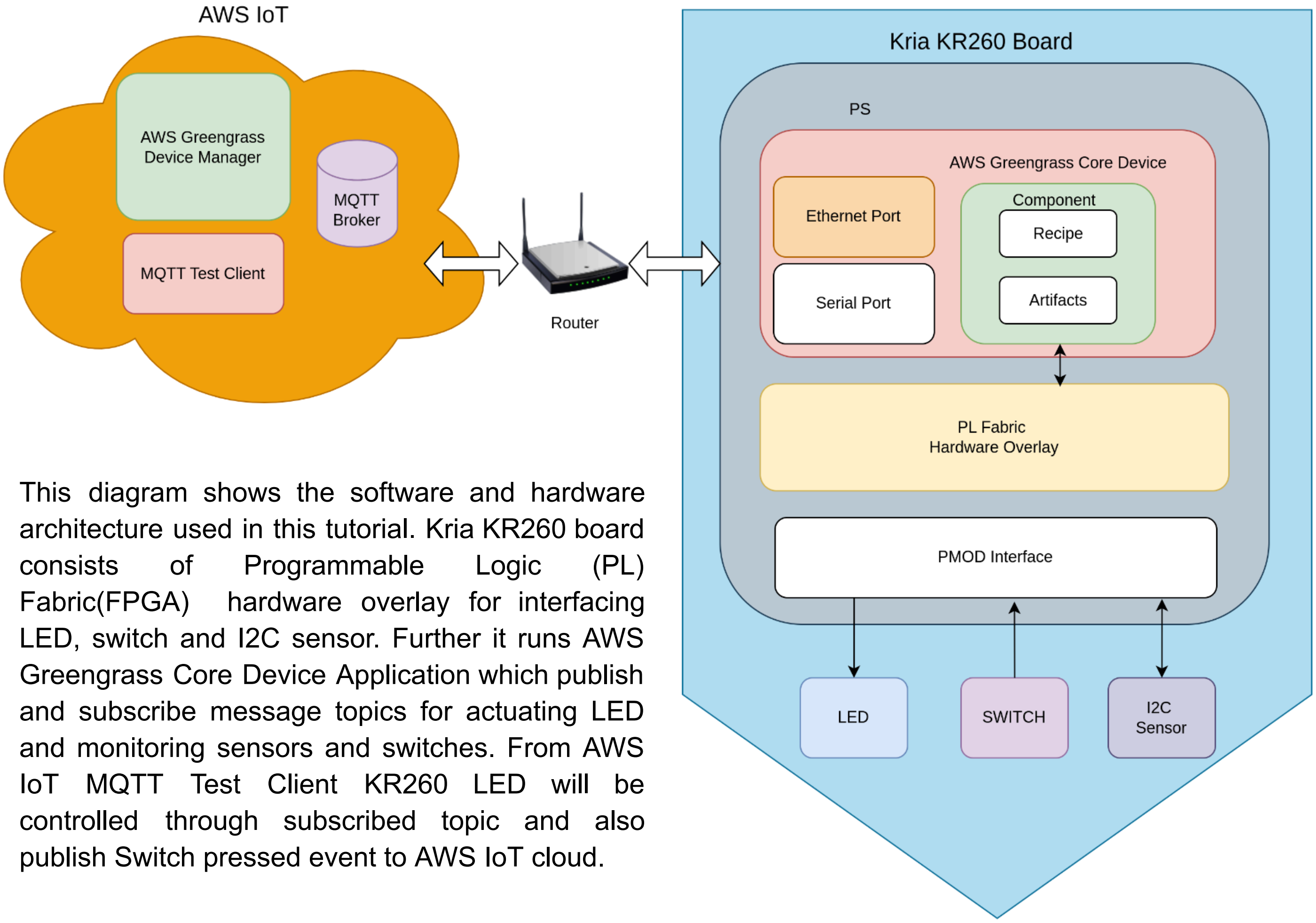
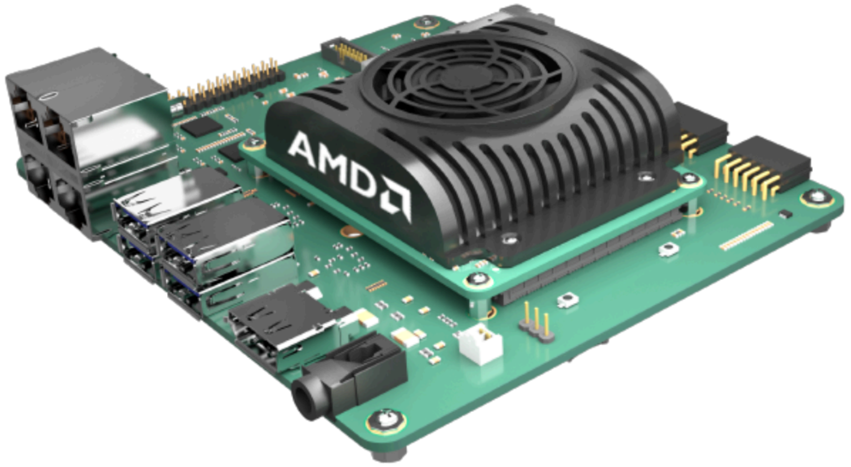
# KR260 to AWS IoT Greengrass Architecture



This diagram shows the software and hardware architecture used in this tutorial. Kria KR260 board consists of Programmable Logic (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.



## Required Hardware Components

1. Kria KR260 board
2. BMP180 Module (Available at Amazon)
3. Connecting wires

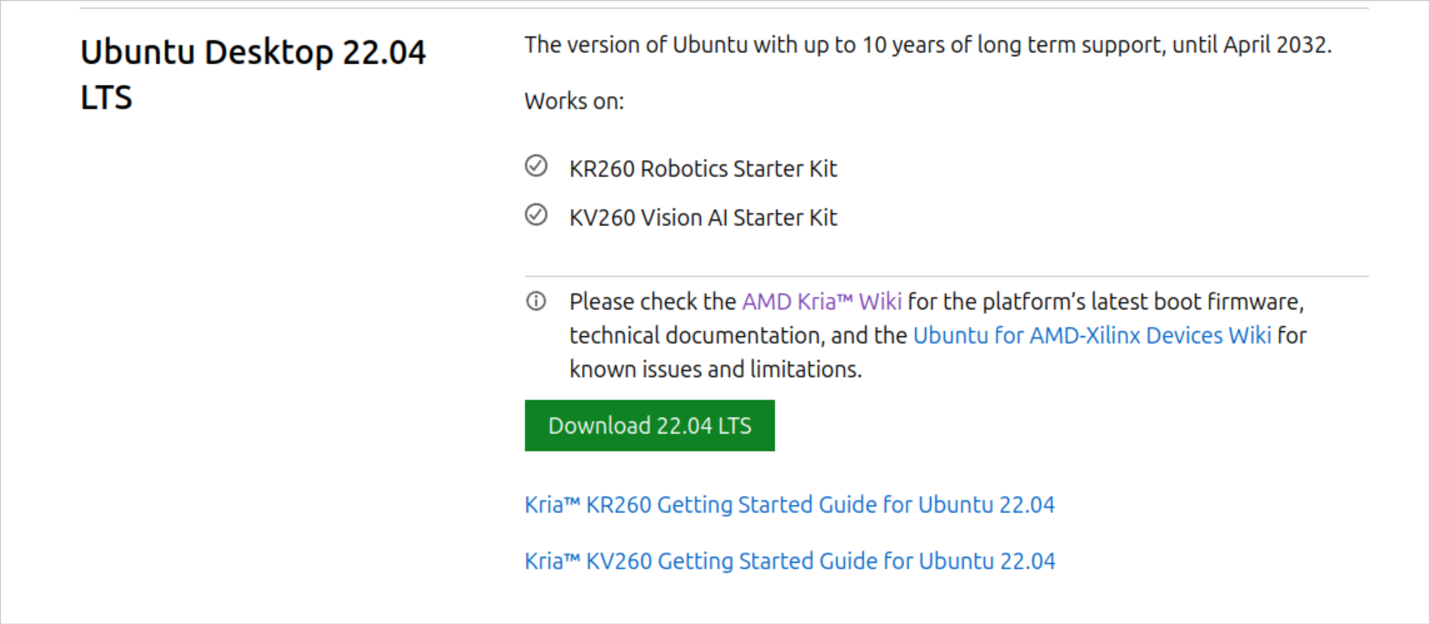
## Software requirements

1. Ubuntu 22.04 for Kria KR260 board
2. AWS account

   Further details and download links are available at /documents/KR260 to AWS Greengrass IoT - GPIO-(Ubuntu/Petalinux).pdf .

# Preparing Ubuntu 22.04 OS for KRIA KR260 board

Download the Ubuntu 22.04 image from the download link



## Ubuntu Desktop 22.04 LTS

The version of Ubuntu with up to 10 years of long term support, until April 2032.

Works on:

- ✓ KR260 Robotics Starter Kit
- ✓ KV260 Vision AI Starter Kit

ⓘ Please check the AMD Kria™ Wiki for the platform's latest boot firmware, technical documentation, and the Ubuntu for AMD-Xilinx Devices Wiki for known issues and limitations.

Kria™ KR260 Getting Started Guide for Ubuntu 22.04

Kria™ KV260 Getting Started Guide for Ubuntu 22.04

Next, prepare the SD card with the above downloaded Ubuntu image using burning tools like Balena Etcher.

Now boot the KR260 with the SD card with Ethernet and USB to Serial cable connected to board. We will be using Serial console for initial access and debugging and Ethernet network for accessing through SSH and KR260 connected to the internet.

For initial login here are the Login Details:
Username : ubuntu
Password: ubuntu
This will ask to change the password. So update the password and login the system.
After successful login, one can access the KR260 device console.

# Installing hardware overlay

Get the KR260 firmware folder. It contains:

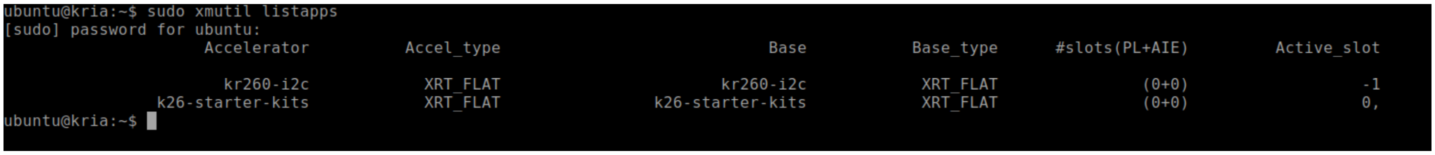
- kr260_i2c.bit.bin
- kr260_i2c.dtbo
- shell.json

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.

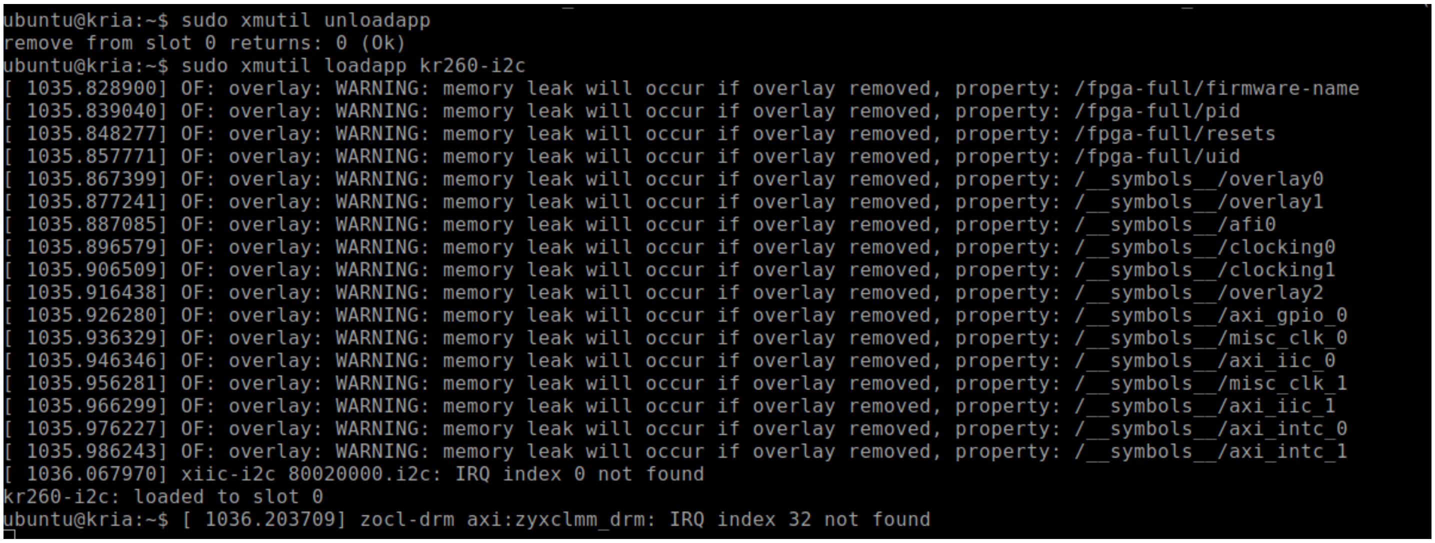


```
ubuntu@kria:~$ sudo xmutil listapps
[sudo] password for ubuntu:
                       Accelerator          Accel_type                      Base          Base_type       #slots(PL+AIE)          Active_slot

                         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.

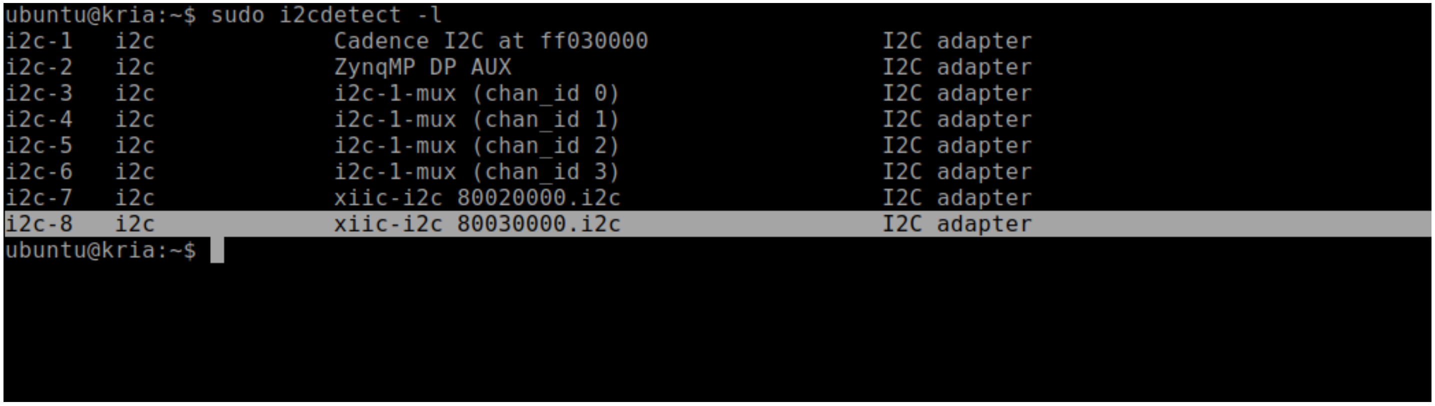
```
sudo xmutil unloadapp
sudo xmutil loadapp kr260-i2c
```



```
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, check the available i2c channels available in the system using `i2cdetect` i2c utility tool.

```
sudo i2cdetect -l
```

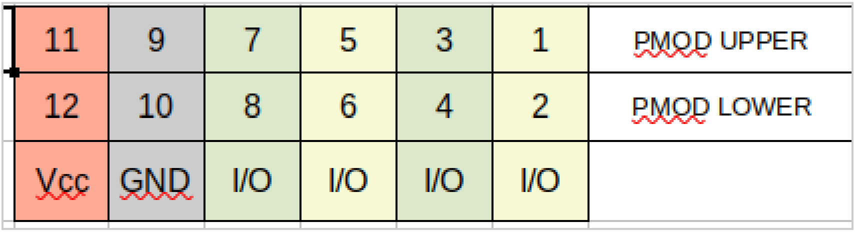


```
ubuntu@kria:~$ sudo i2cdetect -l
i2c-1   i2c             Cadence I2C at ff030000                 I2C adapter
i2c-2   i2c             ZynqMP DP AUX                           I2C adapter
i2c-3   i2c             i2c-1-mux (chan_id 0)                   I2C adapter
i2c-4   i2c             i2c-1-mux (chan_id 1)                   I2C adapter
i2c-5   i2c             i2c-1-mux (chan_id 2)                   I2C adapter
i2c-6   i2c             i2c-1-mux (chan_id 3)                   I2C adapter
i2c-7   i2c             xiic-i2c 80020000.i2c                   I2C adapter
i2c-8   i2c             xiic-i2c 80030000.i2c                   I2C adapter
ubuntu@kria:~$ █
```

`i2c-8` channel will be used to connect to BMP180 sensor.

# Connecting BMP180 to AXI I2C Bus

Connect BMP180 sensors, Vcc, GND, I2C SDA and I2C SCLK pins to PMOD as explained below:
PMOD1-> 6 - I2C SCLK
PMOD1-> 8 - I2C SDA
PMOD1-> GND - BMP180 GND
PMOD1->Vcc - BMP180 Vcc

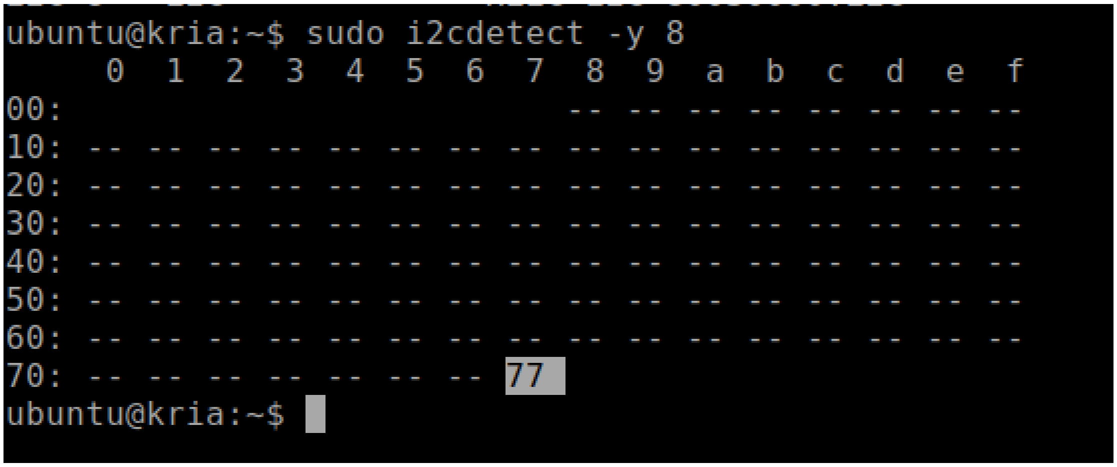


| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | 9 | 7 | 5 | 3 | 1 | PMOD UPPER |
| 12 | 10 | 8 | 6 | 4 | 2 | PMOD LOWER |
| Vcc | GND | I/O | I/O | I/O | I/O | |

PMOD port numbering

After connecting BMP180 sensor to KR260 PMOD port, use i2c utility tools to scan for the available devices in i2c-8 channel.

```
sudo i2cdetect -y 8
```



```
ubuntu@kria:~$ sudo i2cdetect -y 8
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
ubuntu@kria:~$
```

In i2c scan, we find a device is available at address ‘77’, which corresponds to BMP180 i2c sensor. Next we will add the component for publishing BMP180 sensor data to the AWS IoT cloud.

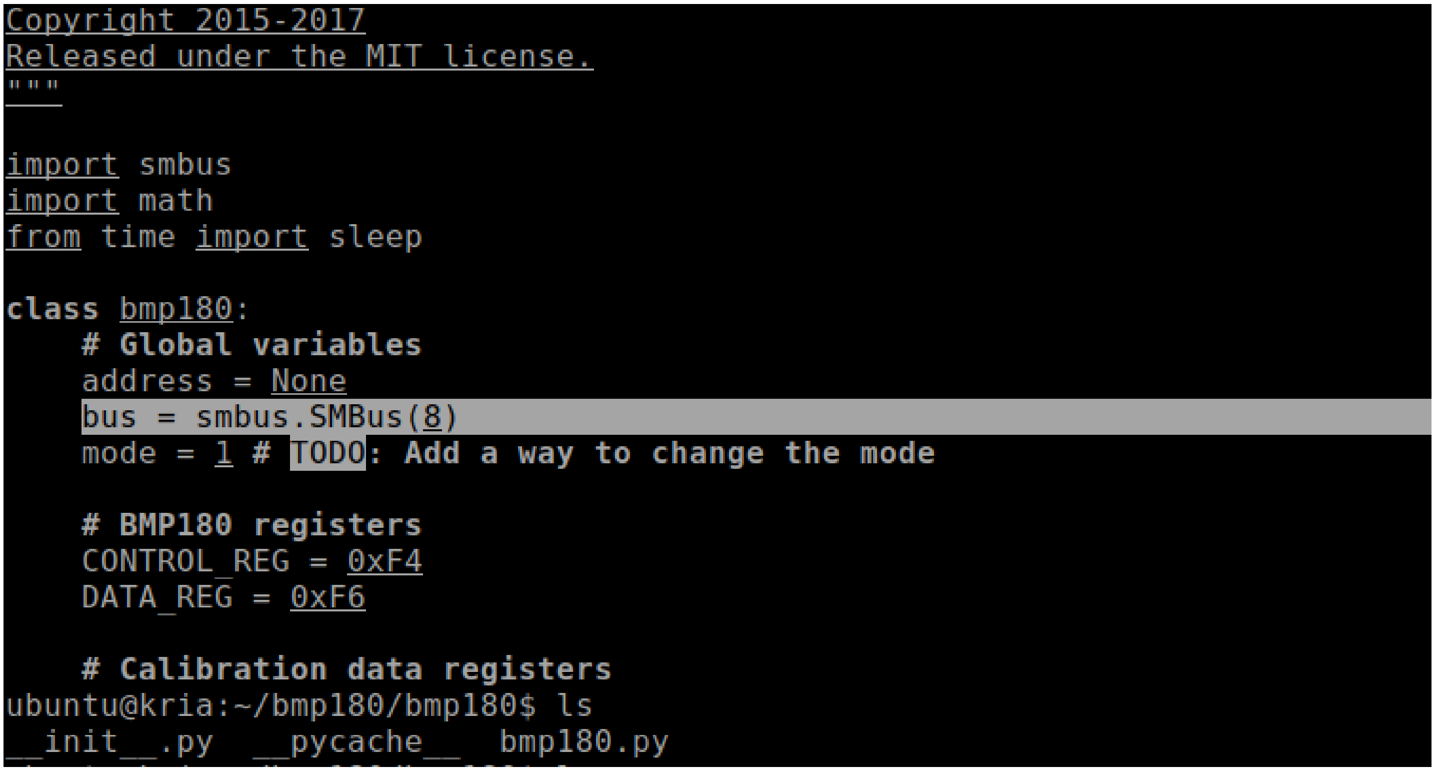
Follow these steps after installing AWS greengrass core device in the KR260 board as mentioned in Kria connect to AWS IoT - GPIO document.

# Installing the component

First install the BMP180 python library by running the following commands in KR260 terminal:

```
git clone https://github.com/m-rtijn/bmp180
cd bmp180
```

Update the ~/bmp180/bmp180/bmp180.py to use i2c-8 channel by changing following lines:



```
Copyright 2015-2017
Released under the MIT license.
"""

import smbus
import math
from time import sleep

class bmp180:
    # Global variables
    address = None
    bus = smbus.SMBus(8)
    mode = 1 # TODO: Add a way to change the mode

    # BMP180 registers
    CONTROL_REG = 0xF4
    DATA_REG = 0xF6

    # Calibration data registers
ubuntu@kria:~/bmp180/bmp180$ ls
__init__.py  __pycache__  bmp180.py
```

Install the bmp180 module by running:

```
sudo python3 setup.py install
```

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

**Artifacts**

- com.example.bmp180
    - 1.0.0
        - mqtt.py (This python code published temperature data)

**Recipe**

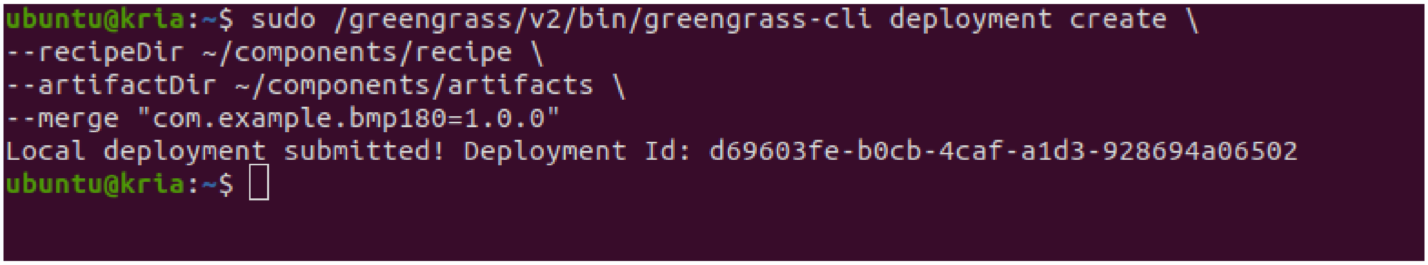
- Com.example.bmp180-1.0.0.json

For installing the greengrass-cli core device, first bring the AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY for the device user to the terminal environment.

```
export AWS_ACCESS_KEY_ID=<AWS ACCESS KEY ID>
export AWS_SECRET_ACCESS_KEY=<AWS SECRET ACCESS KEY>
```

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

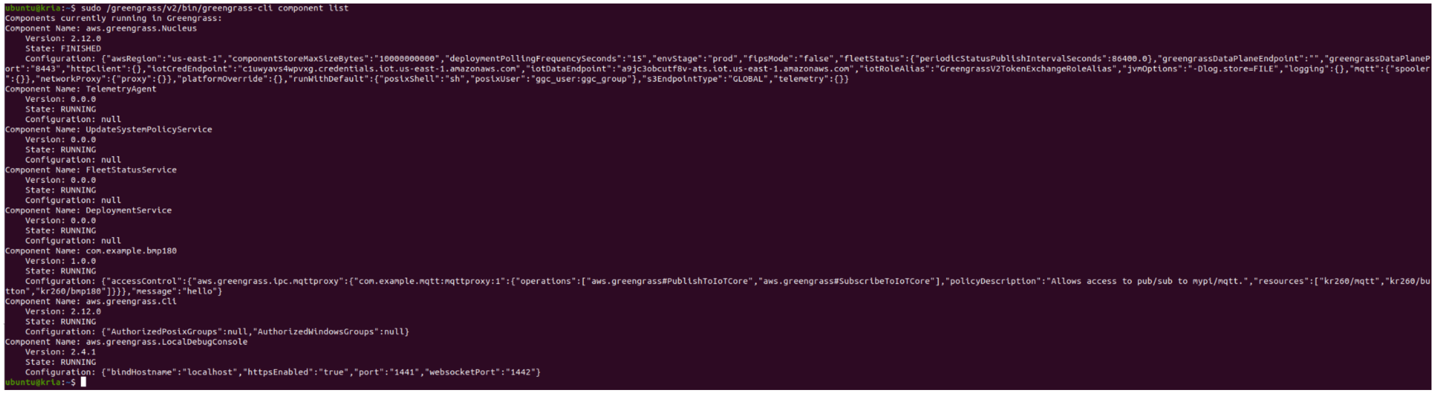
```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.bmp180=1.0.0"
```



```
ubuntu@kria:~$ sudo /greengrass/v2/bin/greengrass-cli deployment create \
--recipeDir ~/components/recipe \
--artifactDir ~/components/artifacts \
--merge "com.example.bmp180=1.0.0"
Local deployment submitted! Deployment Id: d69603fe-b0cb-4caf-a1d3-928694a06502
ubuntu@kria:~$
```

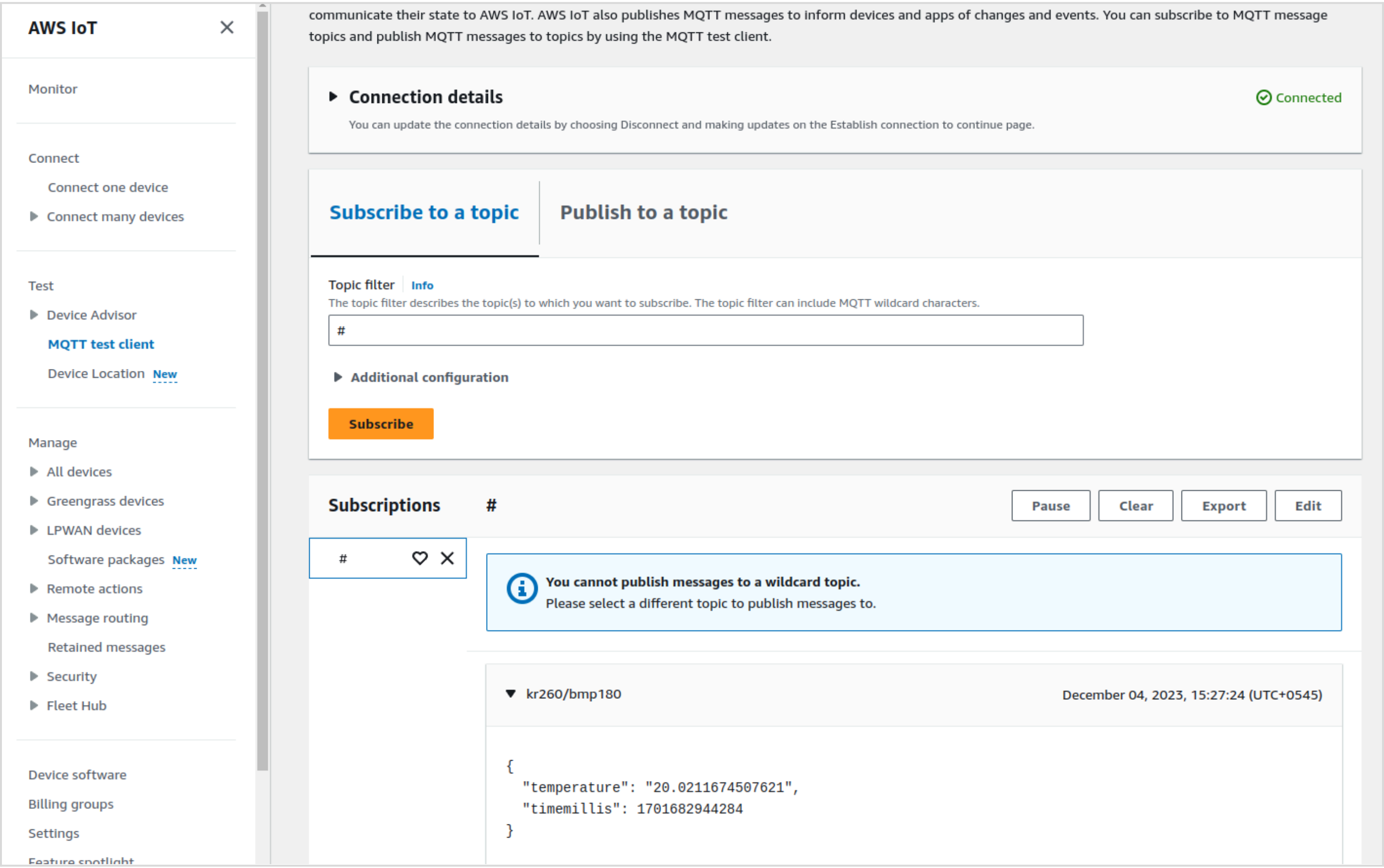
Now check the installed component is in "running state"

```
 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":{"periodicStatusPublishIntervalSeconds":86400.0},"greengrassDataPlaneEndpoint":"","greengrassDataPlaneP
ort":"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: TelemetryAgent
    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: DeploymentService
    Version: 0.0.0
    State: RUNNING
    Configuration: null
Component Name: com.example.bmp180
    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/bu
tton","kr260/bmp180"]}}}},"message":"hello"}
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"}
ubuntu@kria:~$
```

Now in aws IoT console, open "MQTT test client" and subscribe to "#".
You will find the published message in the `kr260/bmp180` topic, which is shown in the following picture.



communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

**Connection details**

You can update the connection details by choosing Disconnect and making updates on the Establish connection to continue page.

Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

#

Subscriptions #

**You cannot publish messages to a wildcard topic.**
Please select a different topic to publish messages to.

kr260/bmp180 December 04, 2023, 15:27:24 (UTC+0545)

```
{
  "temperature": "20.0211674507621",
  "timemillis": 1701682944284
}
```

Now we can collect the sensor data into the database and also create logic to trigger actions on the basis of sensor data.

***