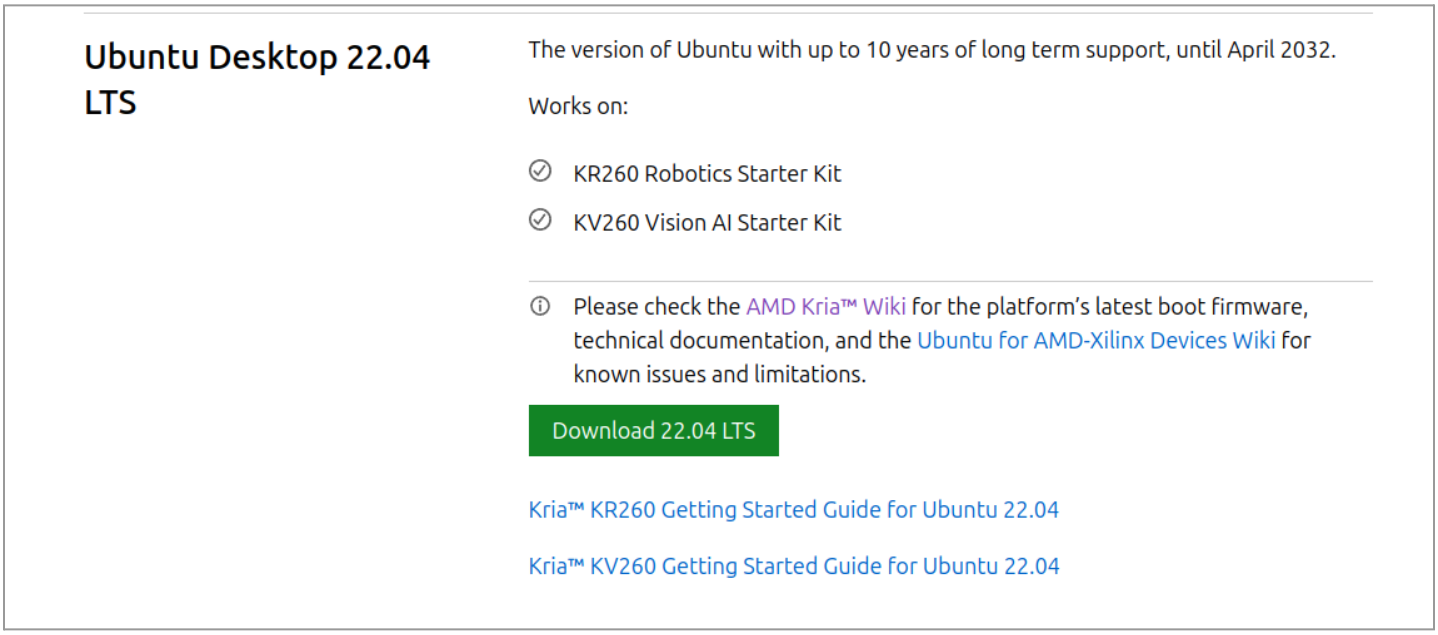
2023.12.21/v0.1 Sanam Shakya

Preparing Ubuntu 22.04 OS for KRIA KR260 board

Download the Ubuntu 22.04 image from the download link



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.

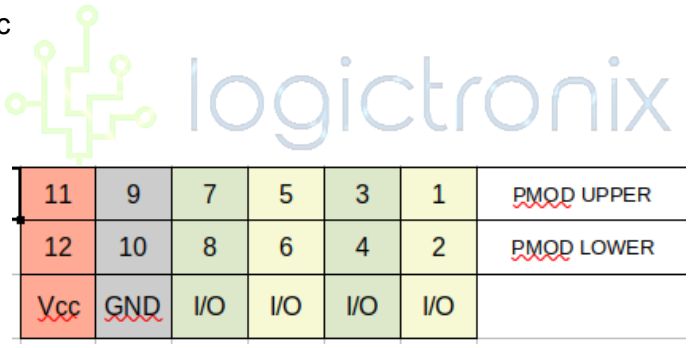
```
sudo i2cdetect -l
```

```
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

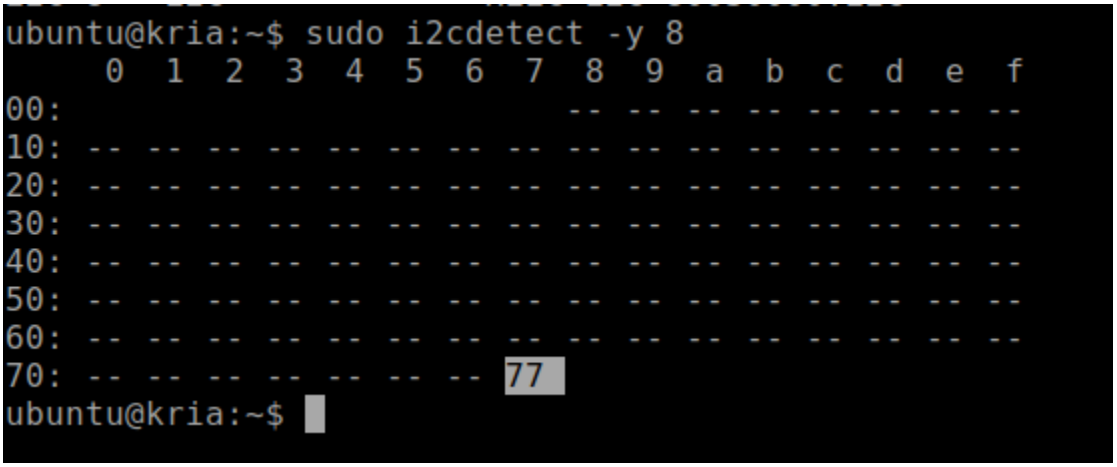
Connect BMP180 sensors, Vcc, GND, I2C SDA and I2C SCLK pins to PMOD as explained below:
PMOD1-> 6 - I2C SCLK
PMOD1-> 8 - I2C SDA
PMOD1-> GND - BMP180 GND
PMOD1->Vcc - BMP180 Vcc



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
```

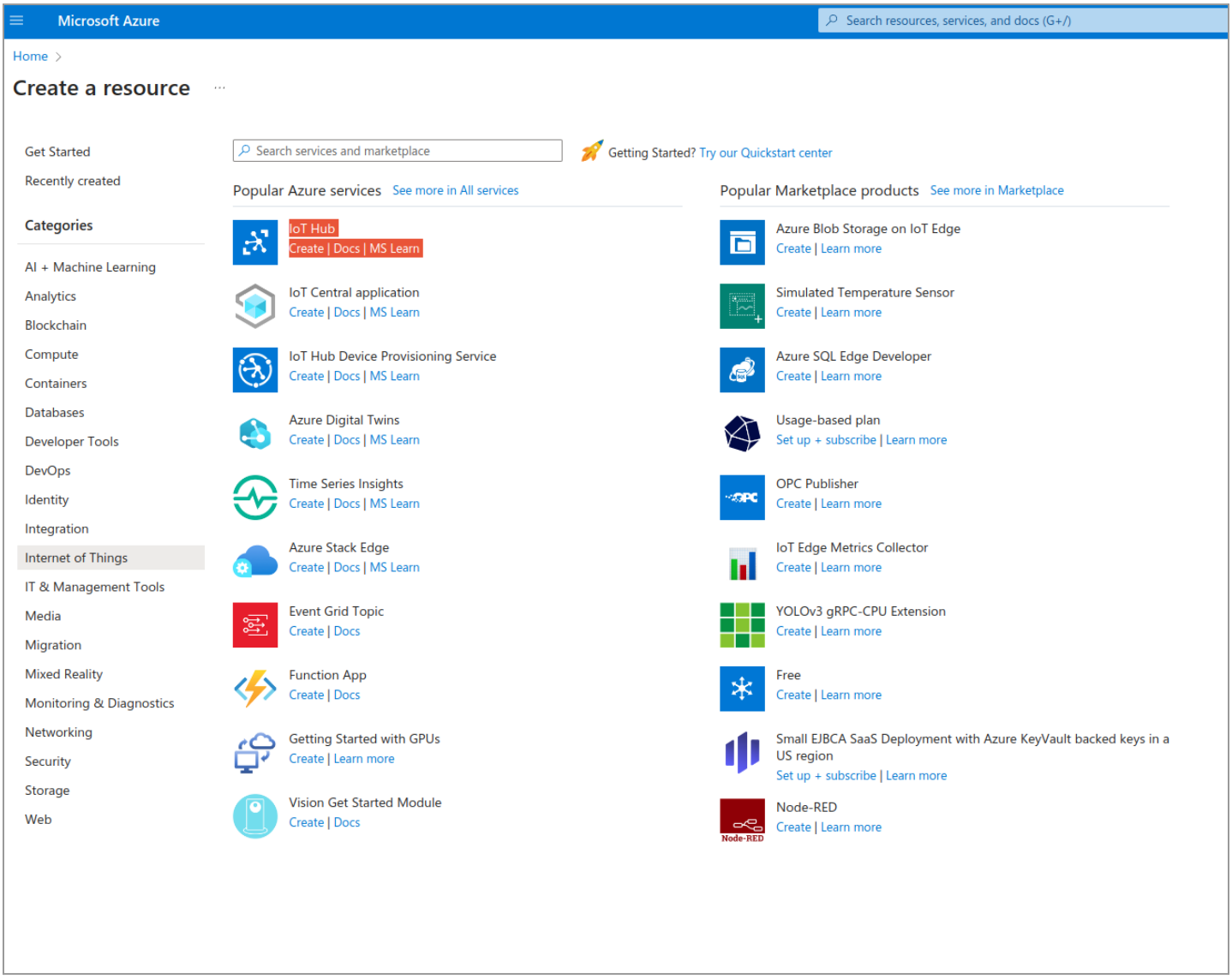


In i2c scan, we find a device is available at address '77', which corresponds to BMP180 i2c sensor.
Next we will add the component for publishing BMP180 sensor data to the AWS IoT cloud.

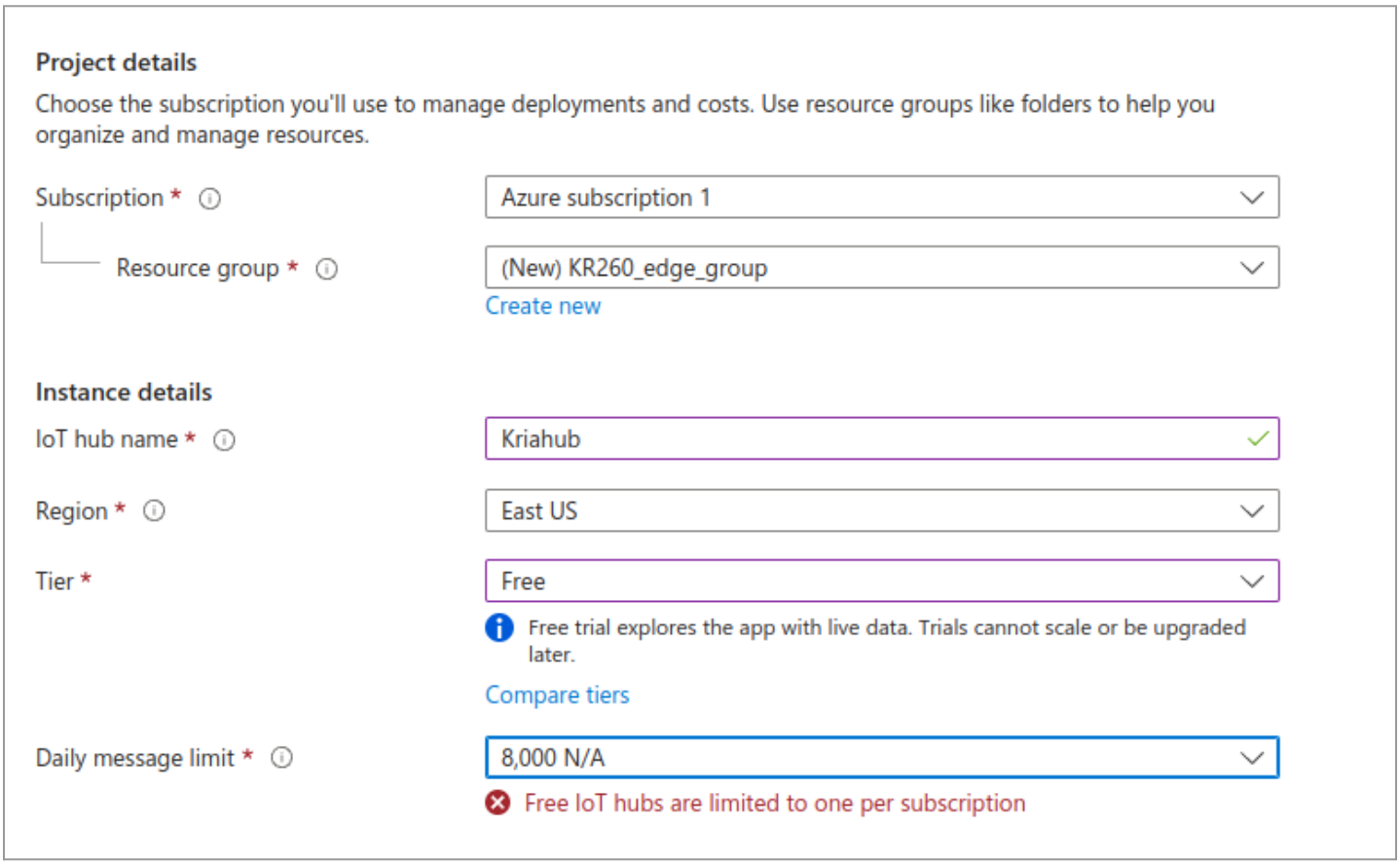
Follow these steps after installing AWS greengrass core device in the KR260 board as mentioned in Kria connect to AWS IoT - GPIO document.

Create IoT Hub in Azure Portal:

- Go to Azure portal " *https://portal.azure.com* ".
- Create a resource >> IoT Hub.



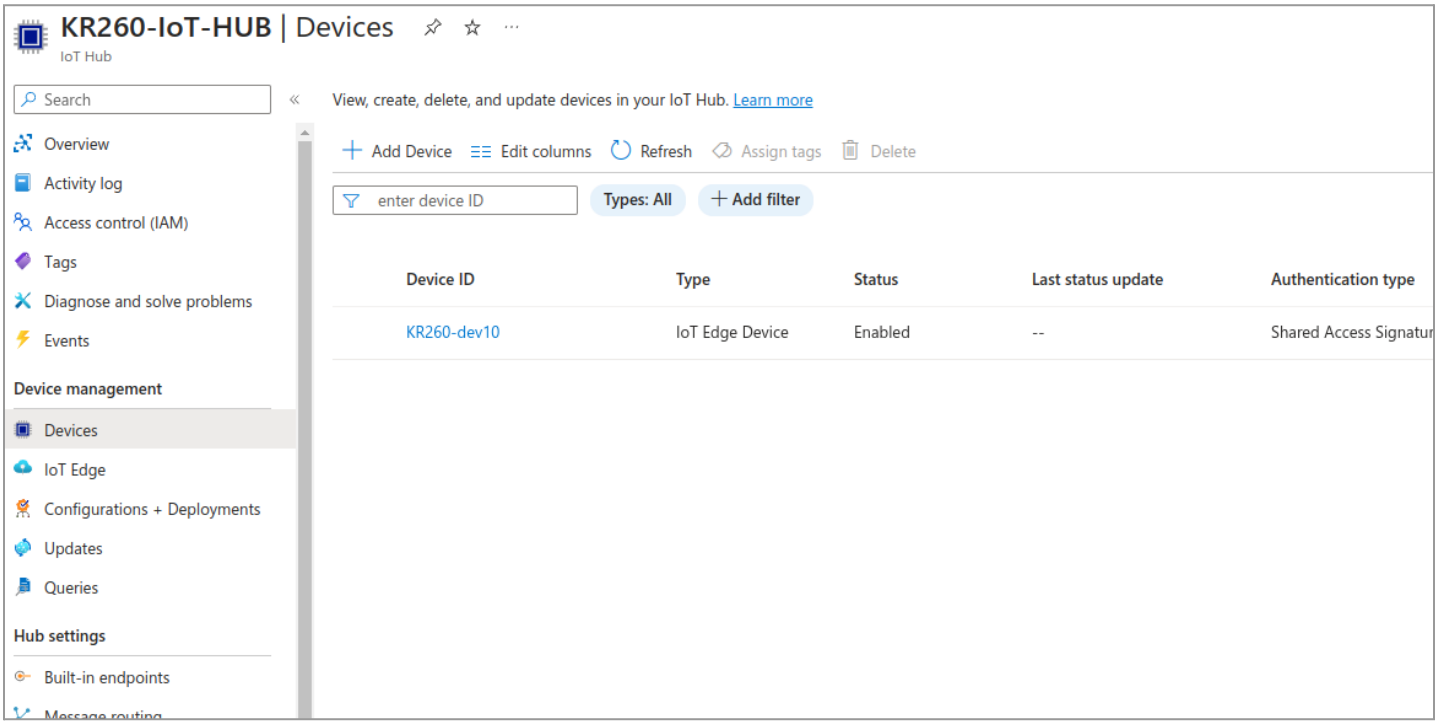
Next, create one IoT Hub Service and fill in the necessary details



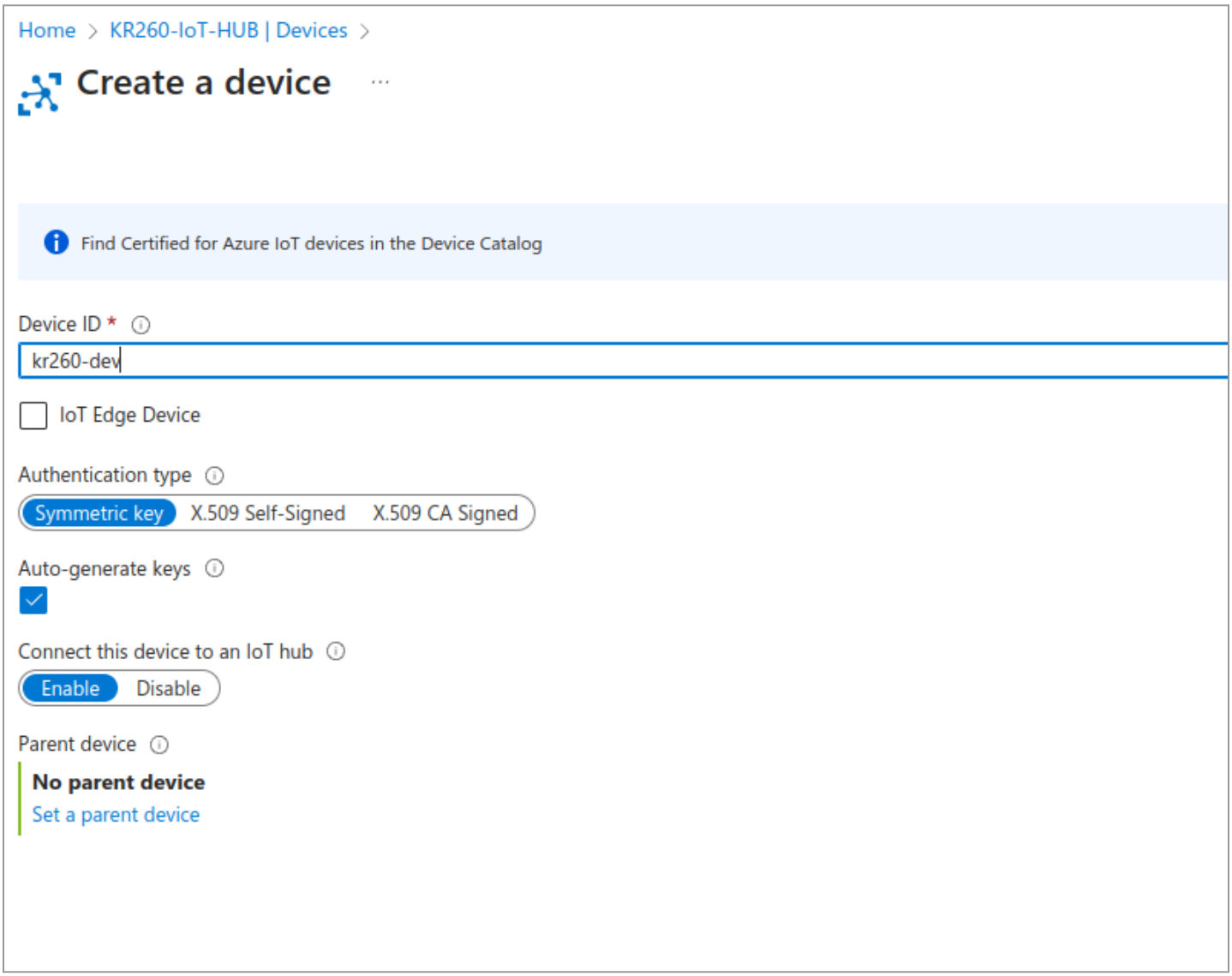
- Click on Review+ Create button to create the Azure IoT Hub.
- Next, create a device where you can actually receive some data from the hardware.

Create an IoT Device

Go to the IoT Device and click on new, and give the device ID

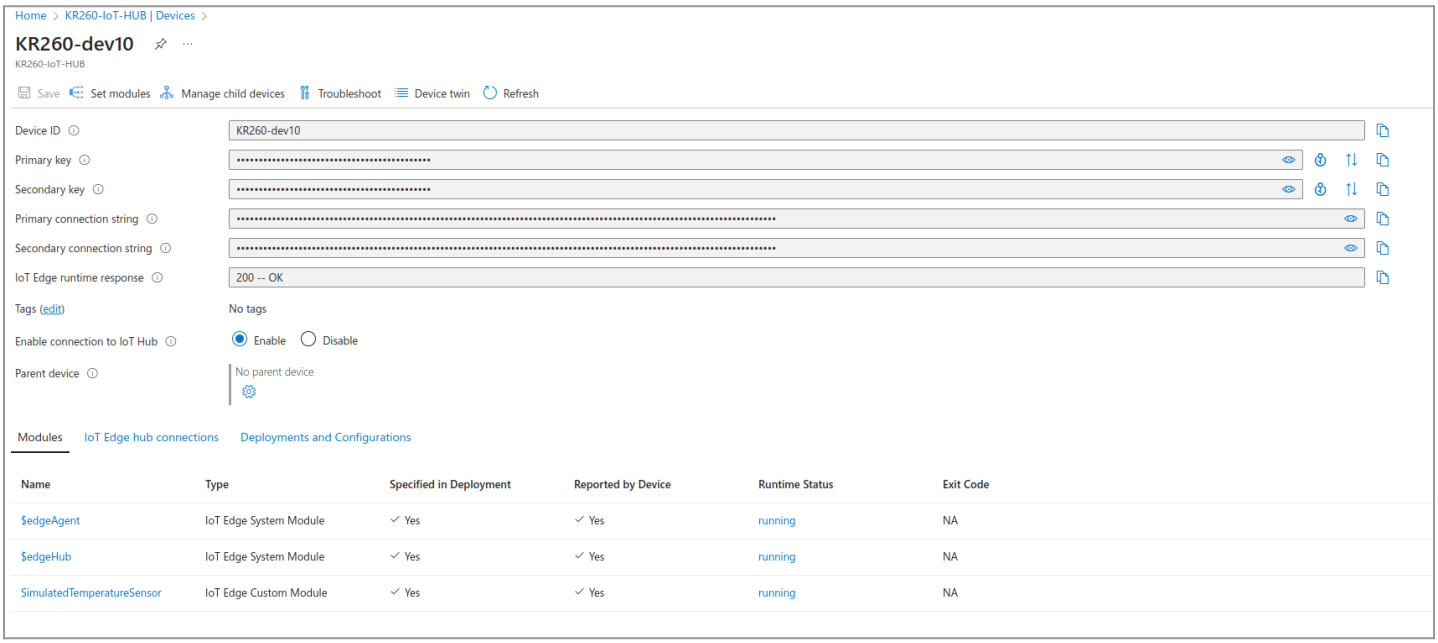


Next Click on +Add Device to add the device to the IoT Hub. This will open the form for creating the device.



After this device will be available in the IoT hub Device list.

Next, look into device information for getting the keys and connection string.



Copy the “Primary Connection String” which will be used in the python application for sending the sensor data to IoT hub.

Installing python packages

azure.iot.device python module is required to create a azure IoT device at the edge device. Install it using python pip3:

```
sudo pip3 install azure-iot-device
```

Further for getting 'bmp180' sensor data from i2c, install bmp180 python driver module from git. For installing, run following commands:

```
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
```

Adding python application in KRIA

Copy the azure_bmp180.py example code to the KR260 board.
Next update the “CONNECTION STRING” with the above Primary Connection string.

```python
import random
import time
from bmp180 import bmp180

bmp = bmp180(0x77)


from azure.iot.device import IoTHubDeviceClient, Message

CONNECTION_STRING = "<Connection String>"

TEMPERATURE = 20.0
HUMIDITY = 60
MSG_TXT = '{{"temperature": {temperature},"humidity": {humidity}}}'

def iothub_client_init():
    client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
    return client

def iothub_client_telemetry_sample_run():

    try:
        client = iothub_client_init()
        print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit" )
        while True:

            temperature = TEMPERATURE + (random.random() * 15)
            humidity = HUMIDITY + (random.random() * 20)
            msg_txt_formatted = MSG_TXT.format(temperature=bmp.get_temp(), humidity=humidity)
            message = Message(msg_txt_formatted)

```

Then run the application in console:

```
sudo python3 azure_bmp180.py
```

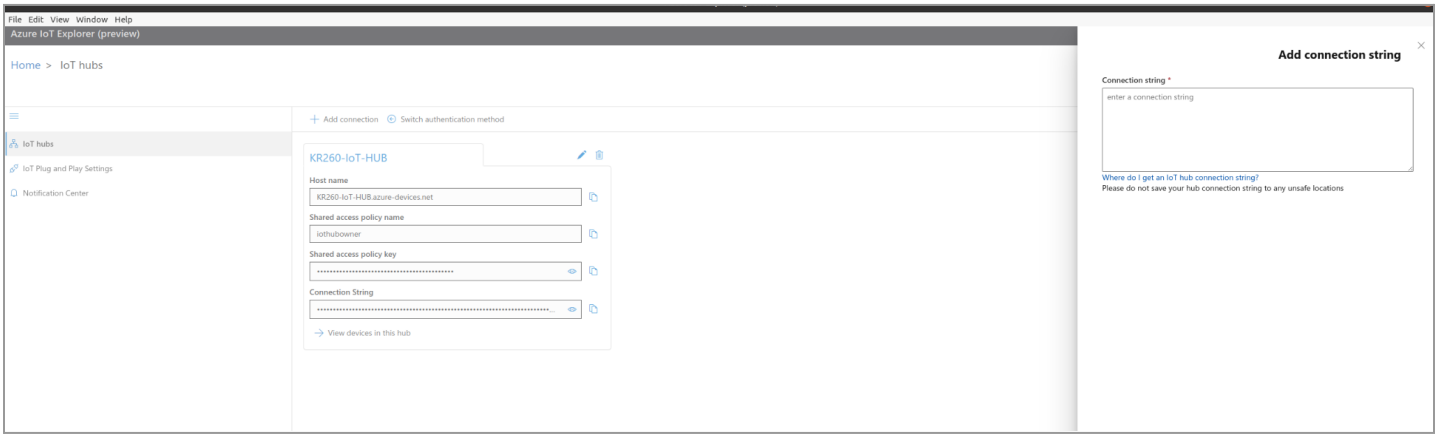
Here is the console log after a successful message send to Azure IoT hub.

```
ubuntu@kria:~$ sudo python3 azure_bmp180.py
IoT Hub Quickstart #1 - Simulated device
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 40.0641054832476,"pressure": 87702.42108554466}
Message successfully sent
Sending message: {"temperature": 40.0641054832476,"pressure": 87693.25119464338}
Message successfully sent
Sending message: {"temperature": 40.0523917363651,"pressure": 87700.52978966695}
Message successfully sent
Sending message: {"temperature": 40.04067743940643,"pressure": 87700.52978966695}
Message successfully sent
Sending message: {"temperature": 40.05824867855125,"pressure": 87702.13451080855}
Message successfully sent
Sending message: {"temperature": 40.0641054832476,"pressure": 87700.07092057214}
Message successfully sent
```

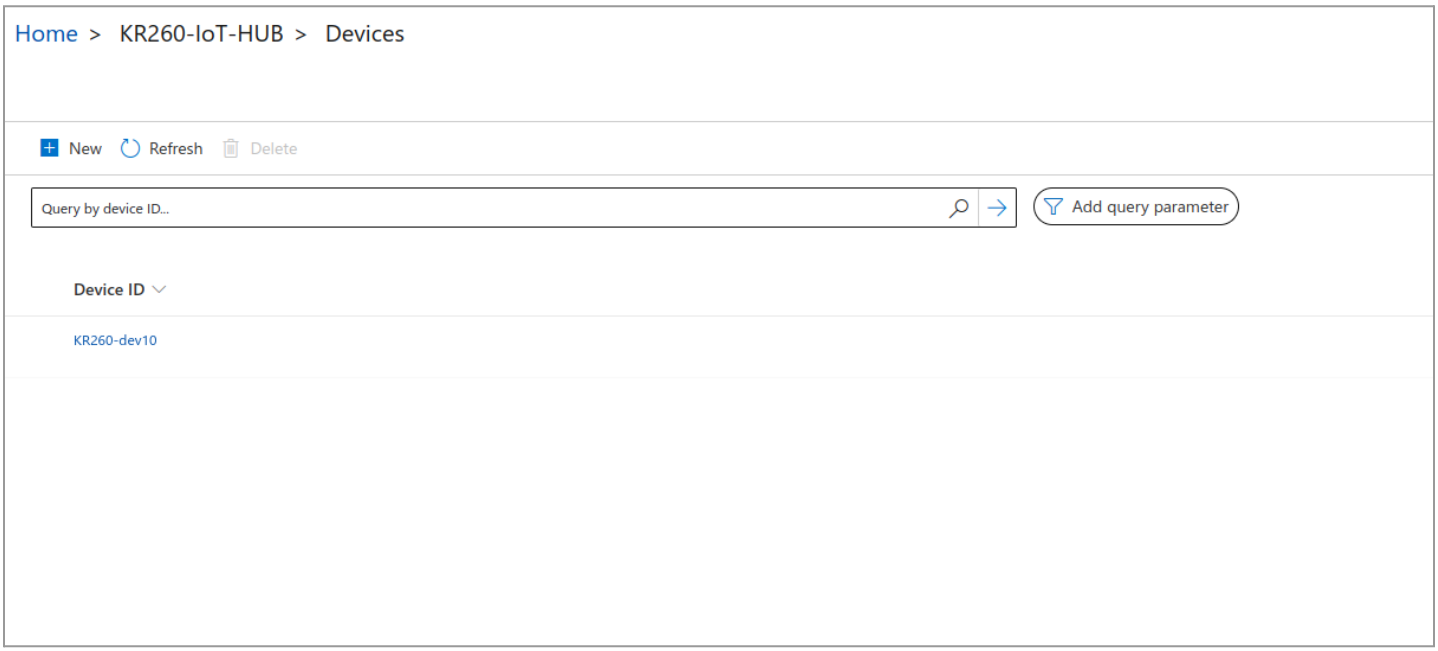
Viewing message in Host Machine

For viewing the message published by Azure IoT Device in KR260, one can use Azure IoT explorer available in following link:
https://github.com/Azure/azure-iot-explorer/releases

In IoT HUbs page of the application, in +Add connection copy the connection string for the IoT hub and save the configs:



One can find the corresponding device list in the IoT HuB page of Azure IoT explorer application.



Just click onto the device to view the device information and also the message send by python application running in the KR260 board.

For viewing the message send to device, go to Telemetry and click the >Start button.
After this one can view the message send to the device.