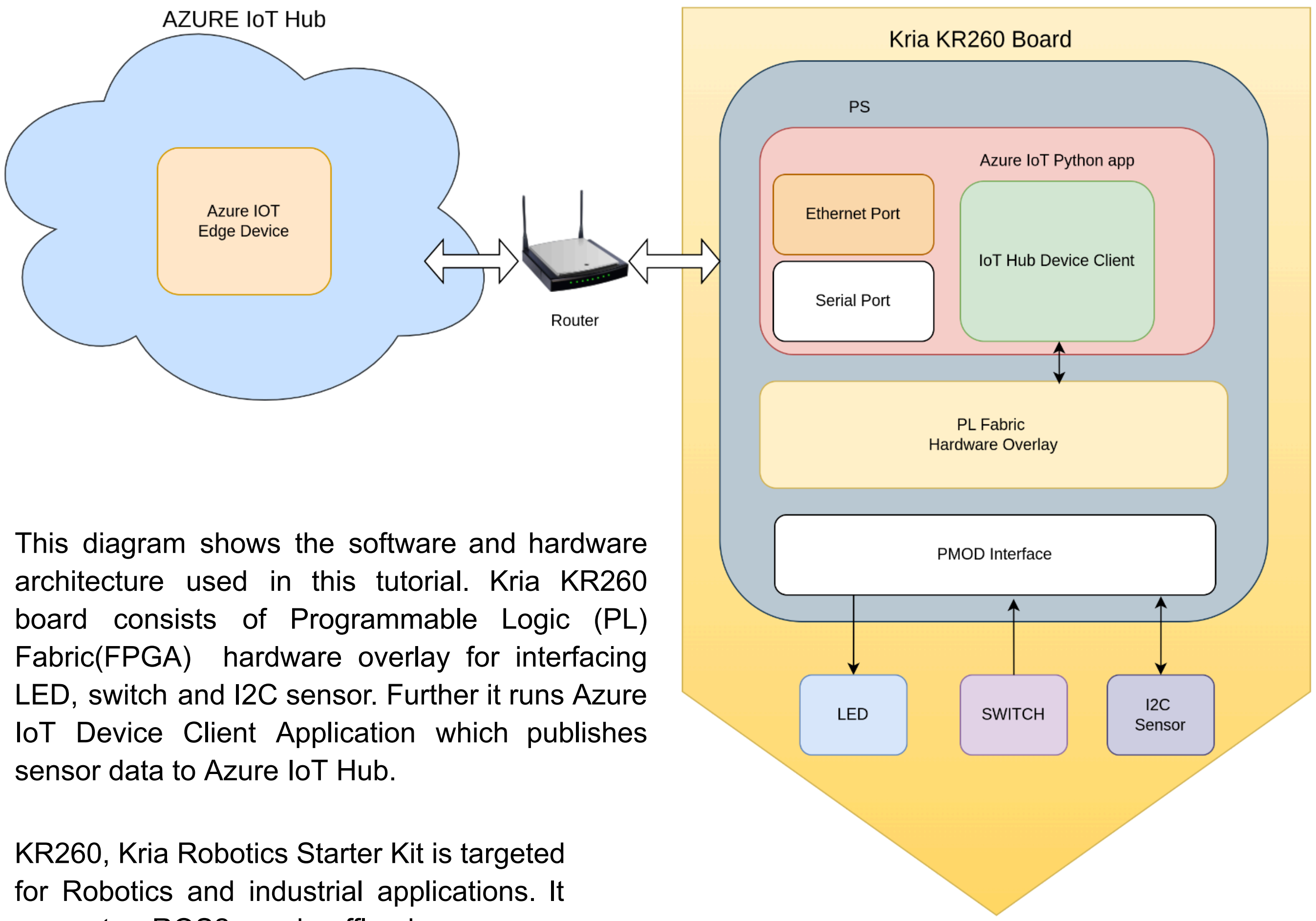
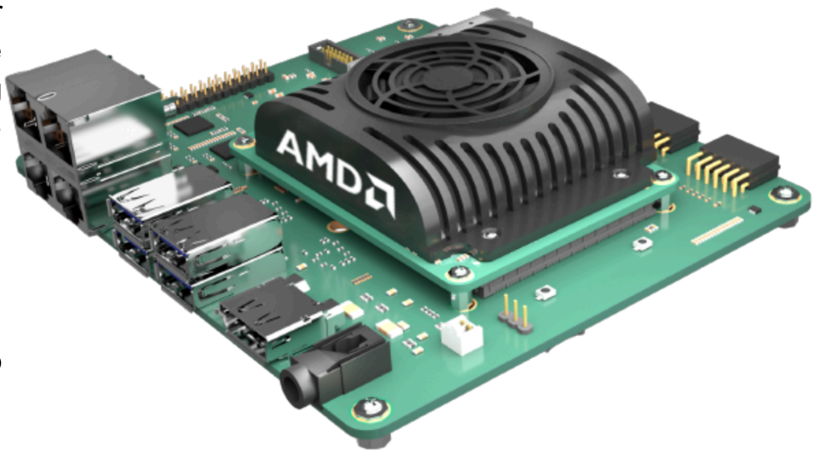
Connecting I2C sensor BMP180 in KR260 (Ubuntu) to AZURE IoT

KR260 to Azure IoT 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 Azure IoT Device Client Application which publishes sensor data to Azure IoT Hub.

KR260, Kria Robotics Starter Kit is targeted for Robotics and industrial applications. It supports ROS2 and offload processor intensive computation into Programmable Logic (PL) based accelerator. By using Deep Learning Processing Unit (DPU) we can run edge based inference on KR260, also by using different sensors, we can control, monitor robotics and industrial system. By harnessing cloud based platform, we can log, manage and plan predictive maintenance cycles with the help of machine learning and numerical modeling.

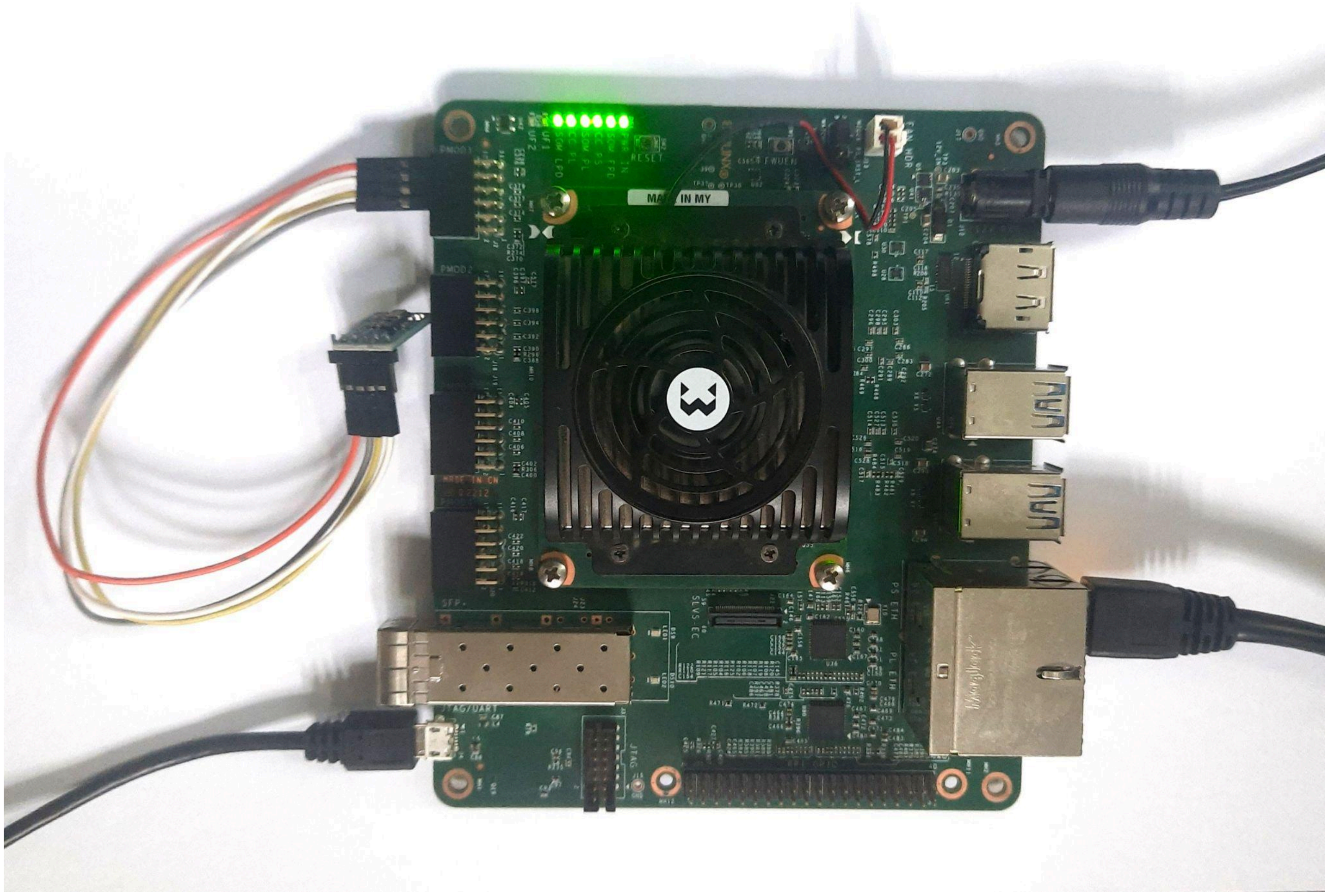


Required Hardware Components

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

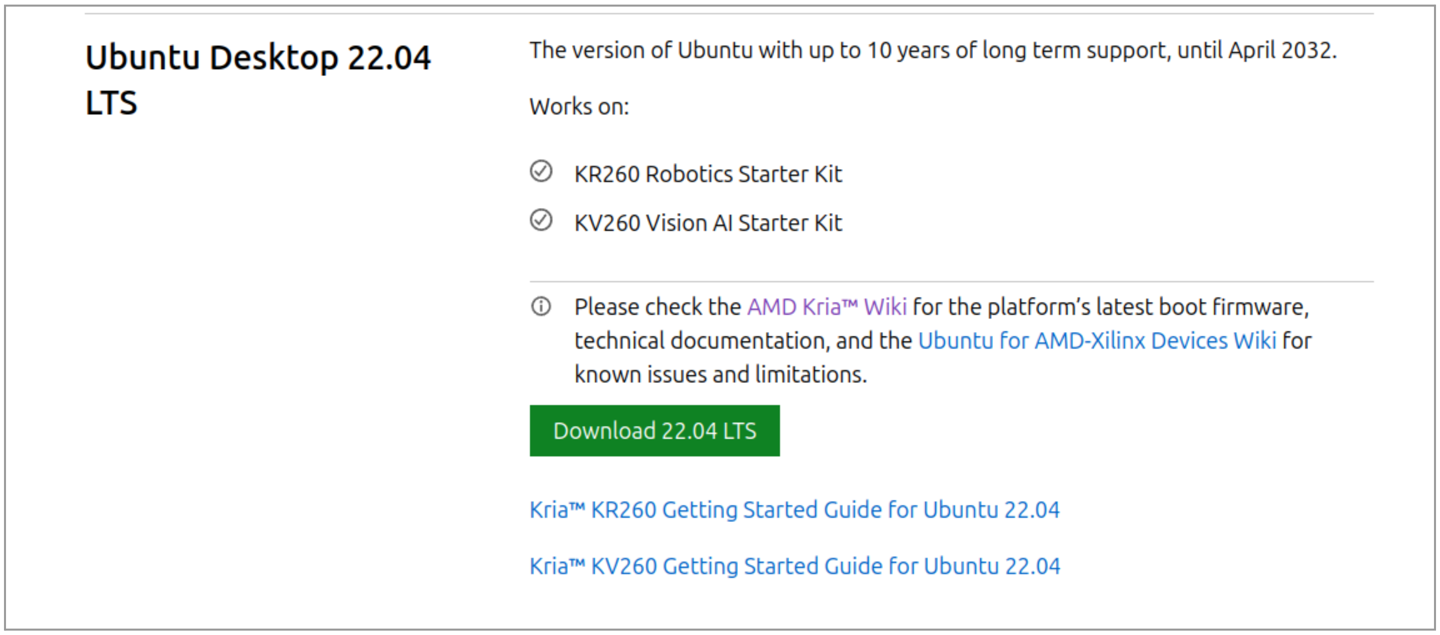
Software requirements

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


KR260 board connected to BMP180 through PMOD-I2C

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

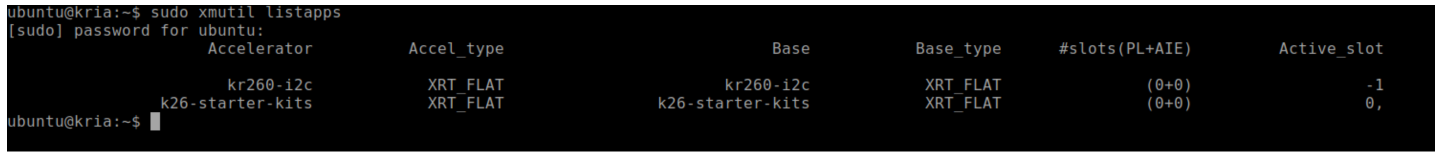
Get the KR260 firmware folder. It contains:

- 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>/kr260_i2c* ./
sudo cp <kr260-firmware directory>/shell.json ./
```

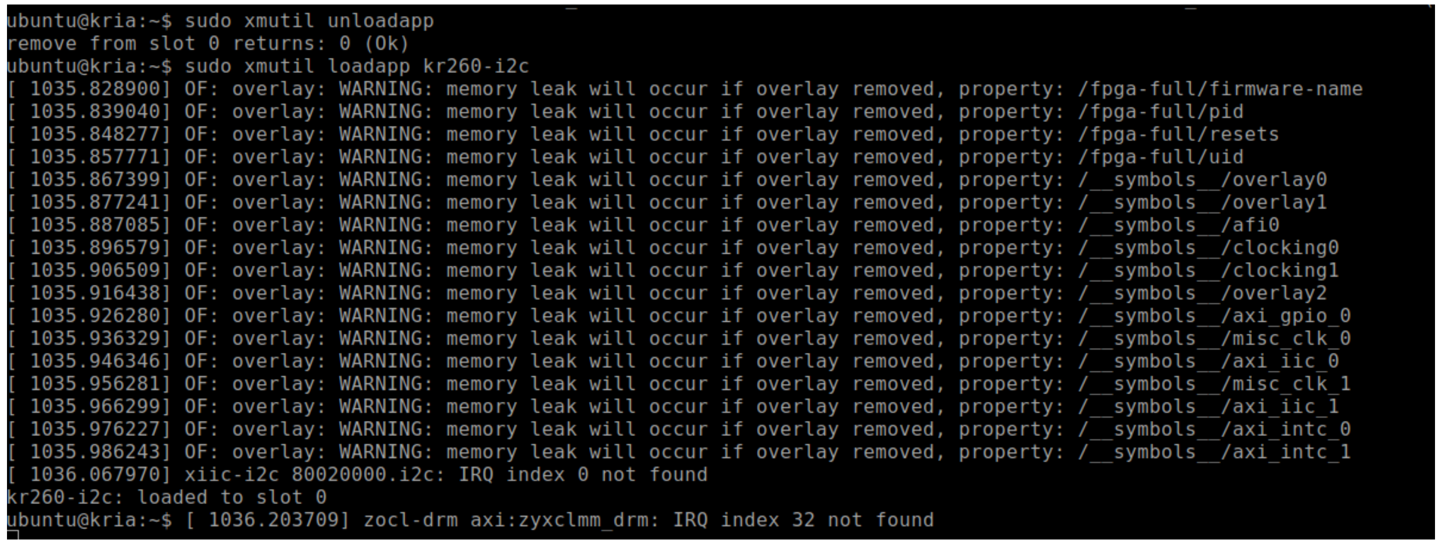
Next, check the available fpga firmware using `xmutil listapps` command. `kr260-i2c` will be available in the list.



```
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.

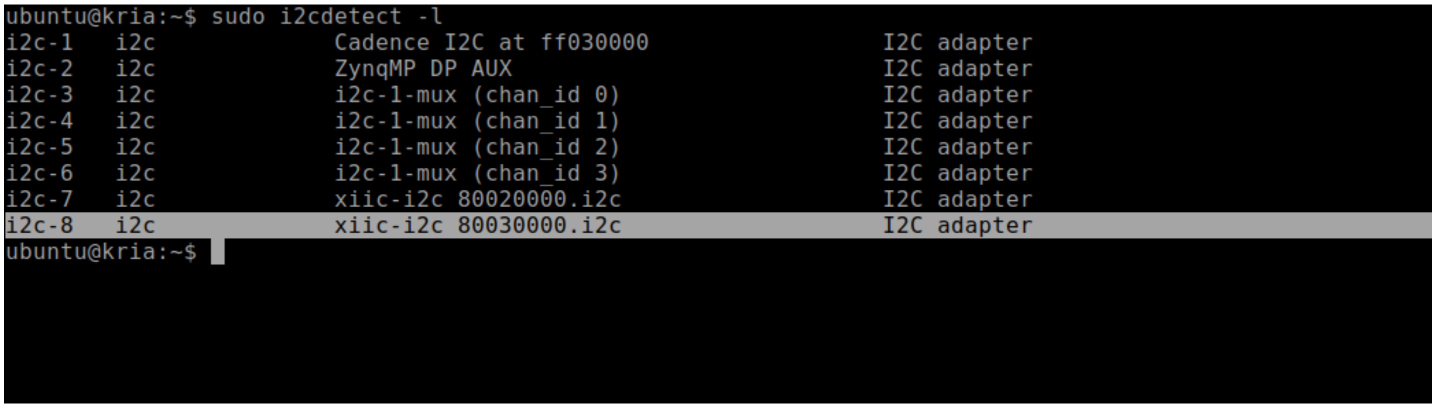
```
sudo xmutil unloadapp
sudo xmutil loadapp kr260-i2c
```



```
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
```

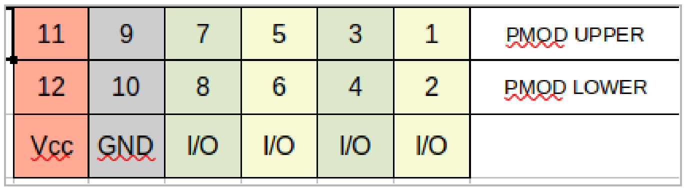


```
ubuntu@kria:~$ sudo i2cdetect -l
i2c-1   i2c             Cadence I2C at ff030000                 I2C adapter
i2c-2   i2c             ZynqMP DP AUX                           I2C adapter
i2c-3   i2c             i2c-1-mux (chan_id 0)                   I2C adapter
i2c-4   i2c             i2c-1-mux (chan_id 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

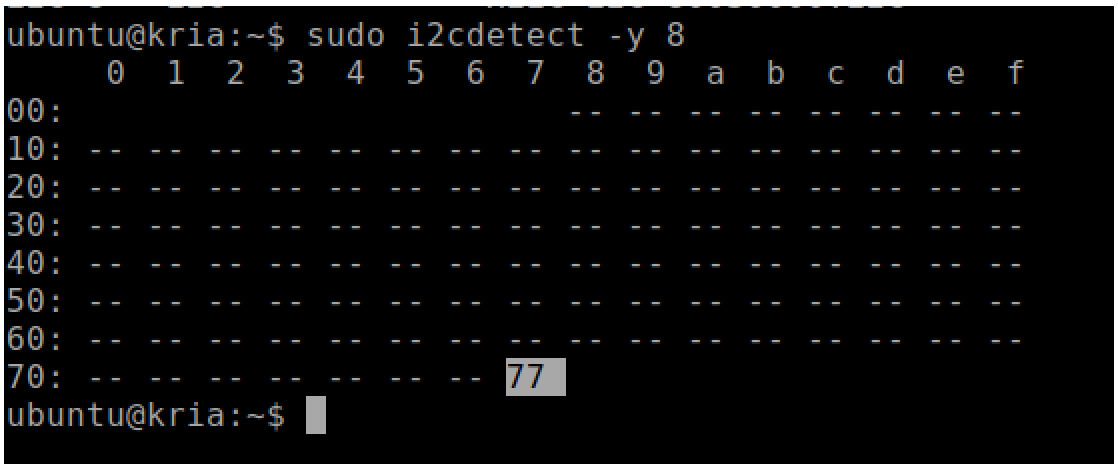


11	9	7	5	3	1	PMOD UPPER
12	10	8	6	4	2	PMOD LOWER
Vcc	GND	I/O	I/O	I/O	I/O	

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
```

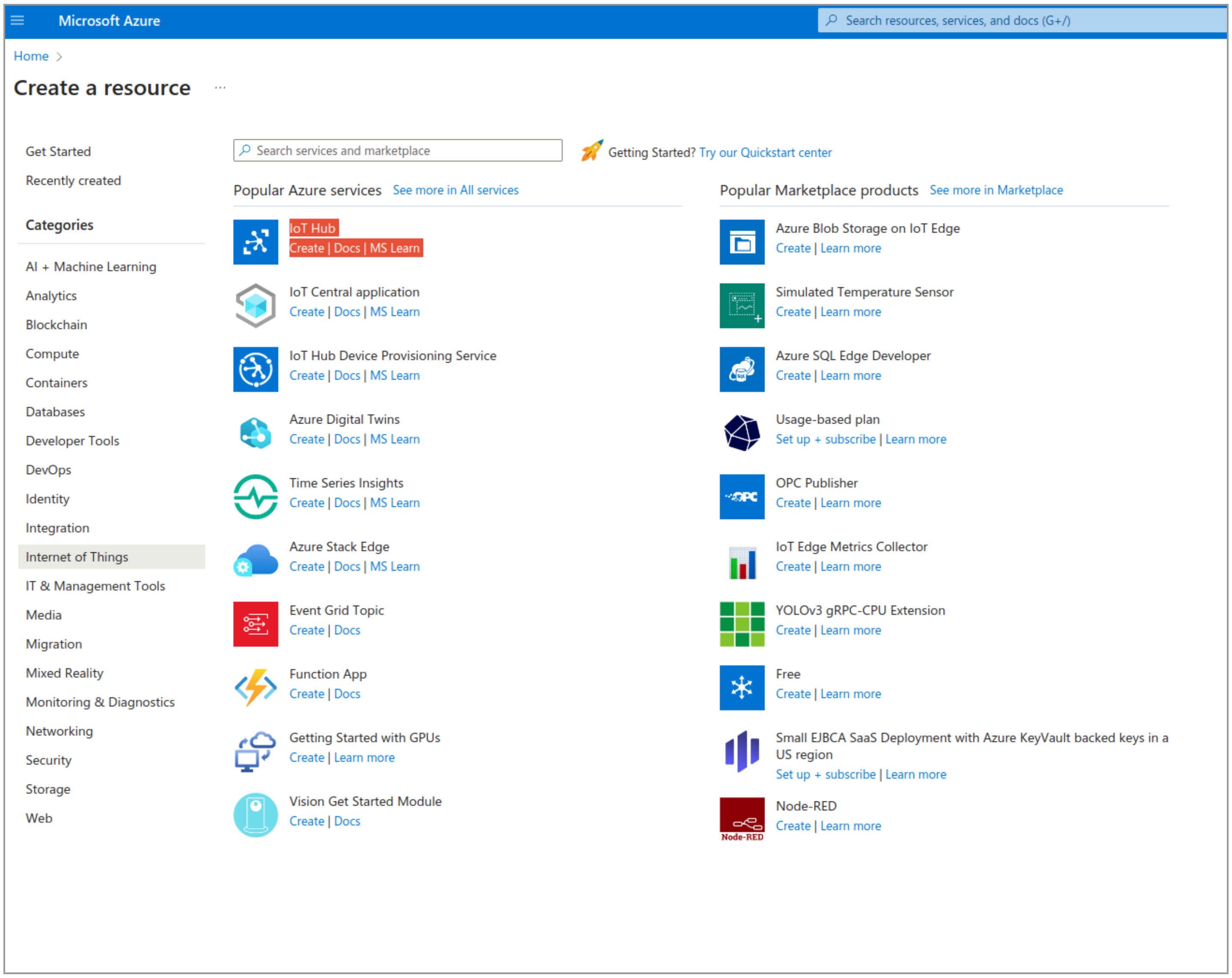


```
ubuntu@kria:~$ sudo i2cdetect -y 8
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: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

In i2c scan, we find a device is available at address ‘77’, which corresponds to BMP180 i2c sensor.

Create IoT Hub in Azure Portal:

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


Microsoft Azure
Search resources, services, and docs (G+/)
Home >
Create a resource

Get Started
Recently created

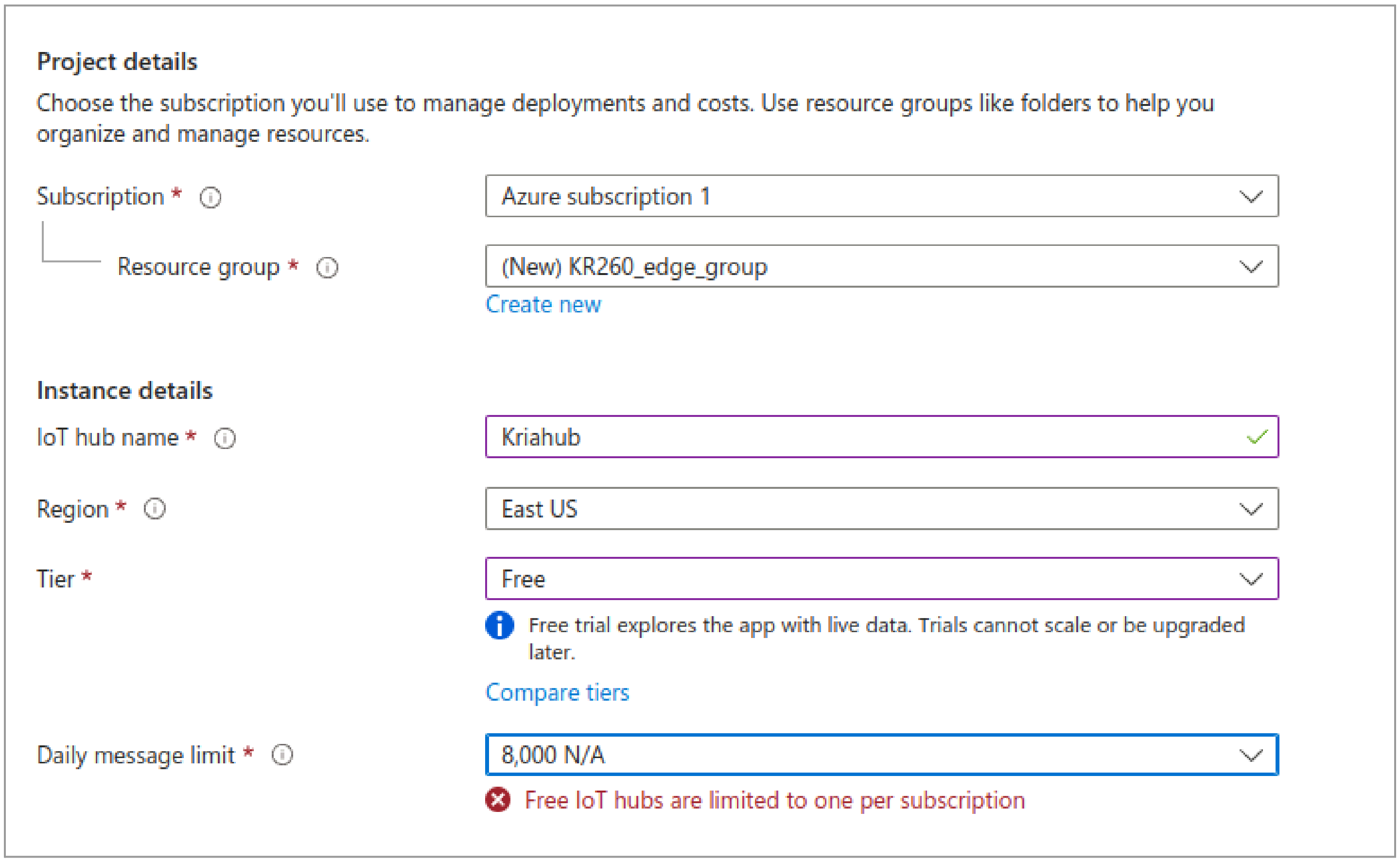
Categories
AI + Machine Learning
Analytics
Blockchain
Compute
Containers
Databases
Developer Tools
DevOps
Identity
Integration
Internet of Things
IT & Management Tools
Media
Migration
Mixed Reality
Monitoring & Diagnostics
Networking
Security
Storage
Web

Search services and marketplace
Getting Started? Try our Quickstart center

Popular Azure services — See more in All services
- IoT Hub — Create | Docs | MS Learn
- IoT Central application — Create | Docs | MS Learn
- IoT Hub Device Provisioning Service — Create | Docs | MS Learn
- Azure Digital Twins — Create | Docs | MS Learn
- Time Series Insights — Create | Docs | MS Learn
- Azure Stack Edge — Create | Docs | MS Learn
- Event Grid Topic — Create | Docs
- Function App — Create | Docs
- Getting Started with GPUs — Create | Learn more
- Vision Get Started Module — Create | Docs

Popular Marketplace products — See more in Marketplace
- Azure Blob Storage on IoT Edge — Create | Learn more
- Simulated Temperature Sensor — Create | Learn more
- Azure SQL Edge Developer — Create | Learn more
- Usage-based plan — Set up + subscribe | Learn more
- OPC Publisher — Create | Learn more
- IoT Edge Metrics Collector — Create | Learn more
- YOLOv3 gRPC-CPU Extension — Create | Learn more
- Free — Create | Learn more
- Small EJBCA SaaS Deployment with Azure KeyVault backed keys in a US region — Set up + subscribe | Learn more
- Node-RED — Create | Learn more

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



Project details

Choose the subscription you'll use to manage deployments and costs. Use resource groups like folders to help you organize and manage resources.

Subscription *	Azure subscription 1
Resource group *	(New) KR260_edge_group

Instance details

IoT hub name *	Kriahub
Region *	East US
Tier *	Free

Free trial explores the app with live data. Trials cannot scale or be upgraded later.

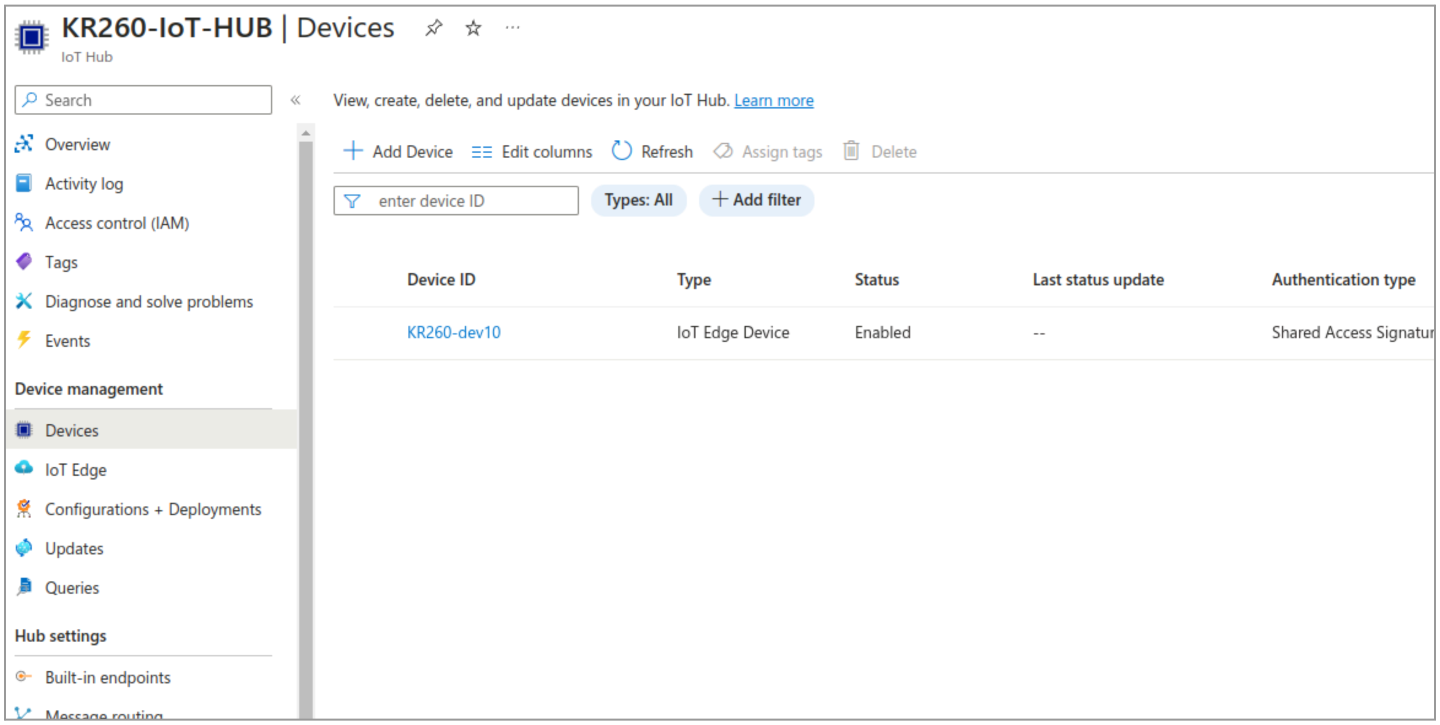
Daily message limit *	8,000 N/A

Free IoT hubs are limited to one per subscription

- 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

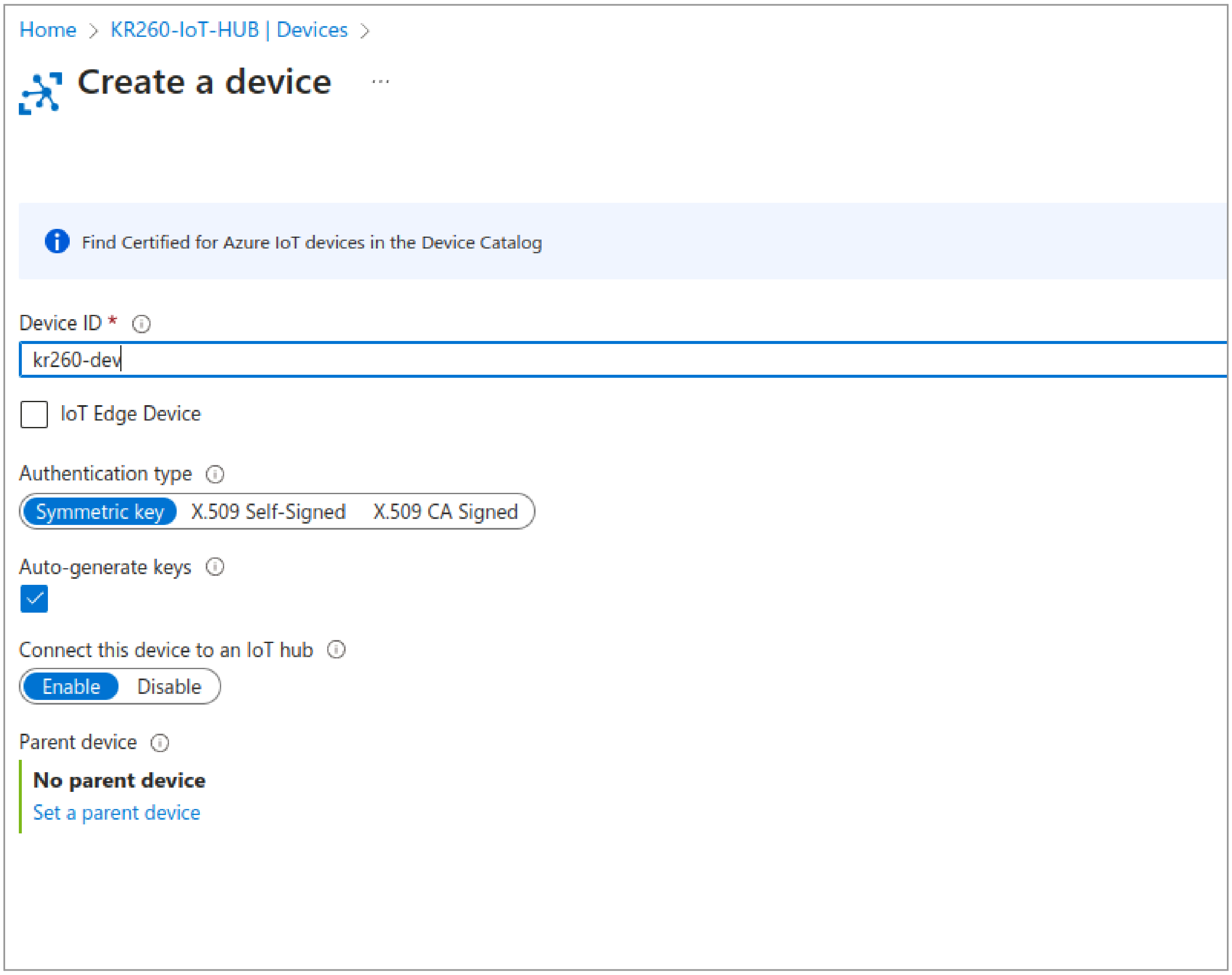
Go to the IoT Device and click on new, and give the device ID



View, create, delete, and update devices in your IoT Hub. Learn more

Device ID	Type	Status	Last status update	Authentication type
KR260-dev10	IoT Edge Device	Enabled	--	Shared Access Signatur

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


Home > KR260-IoT-HUB | Devices >

Create a device ...

Find Certified for Azure IoT devices in the Device Catalog

Device ID *
kr260-dev

[ ] IoT Edge Device

Authentication type
[✓ Symmetric key] X.509 Self-Signed X.509 CA Signed

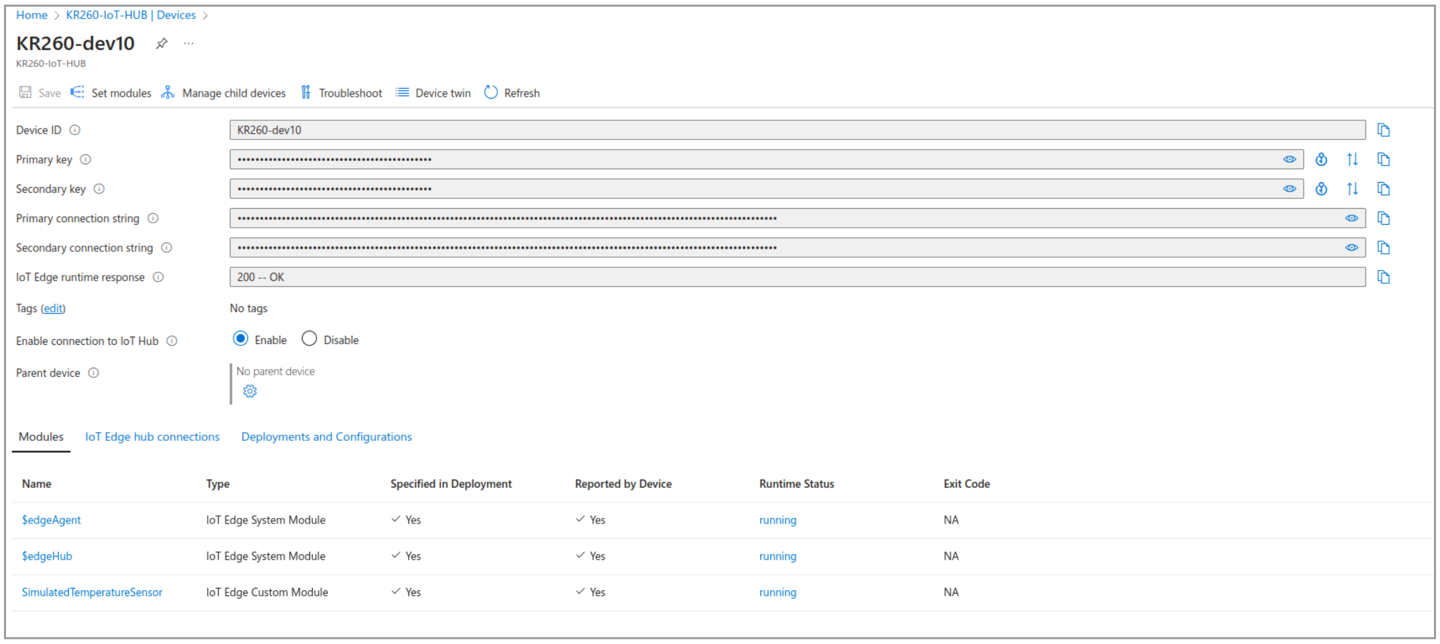
Auto-generate keys
[✓]

Connect this device to an IoT hub
[✓ Enable] Disable

Parent device
No parent device
Set a parent 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.



Home > KR260-IoT-HUB | Devices >

KR260-dev10

KR260-IoT-HUB

Save | Set modules | Manage child devices | Troubleshoot | Device twin | Refresh

Device ID	KR260-dev10
Primary key	••
Secondary key	••
Primary connection string	••
Secondary connection string	••
IoT Edge runtime response	200 -- OK
Tags (edit)	No tags
Enable connection to IoT Hub	[✓ Enable] Disable
Parent device	No parent device

Modules | IoT Edge hub connections | Deployments and Configurations

Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit Code
$edgeAgent	IoT Edge System Module	✓ Yes	✓ Yes	running	NA
$edgeHub	IoT Edge System Module	✓ Yes	✓ Yes	running	NA
SimulatedTemperatureSensor	IoT Edge Custom Module	✓ Yes	✓ Yes	running	NA

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

Installing python packages

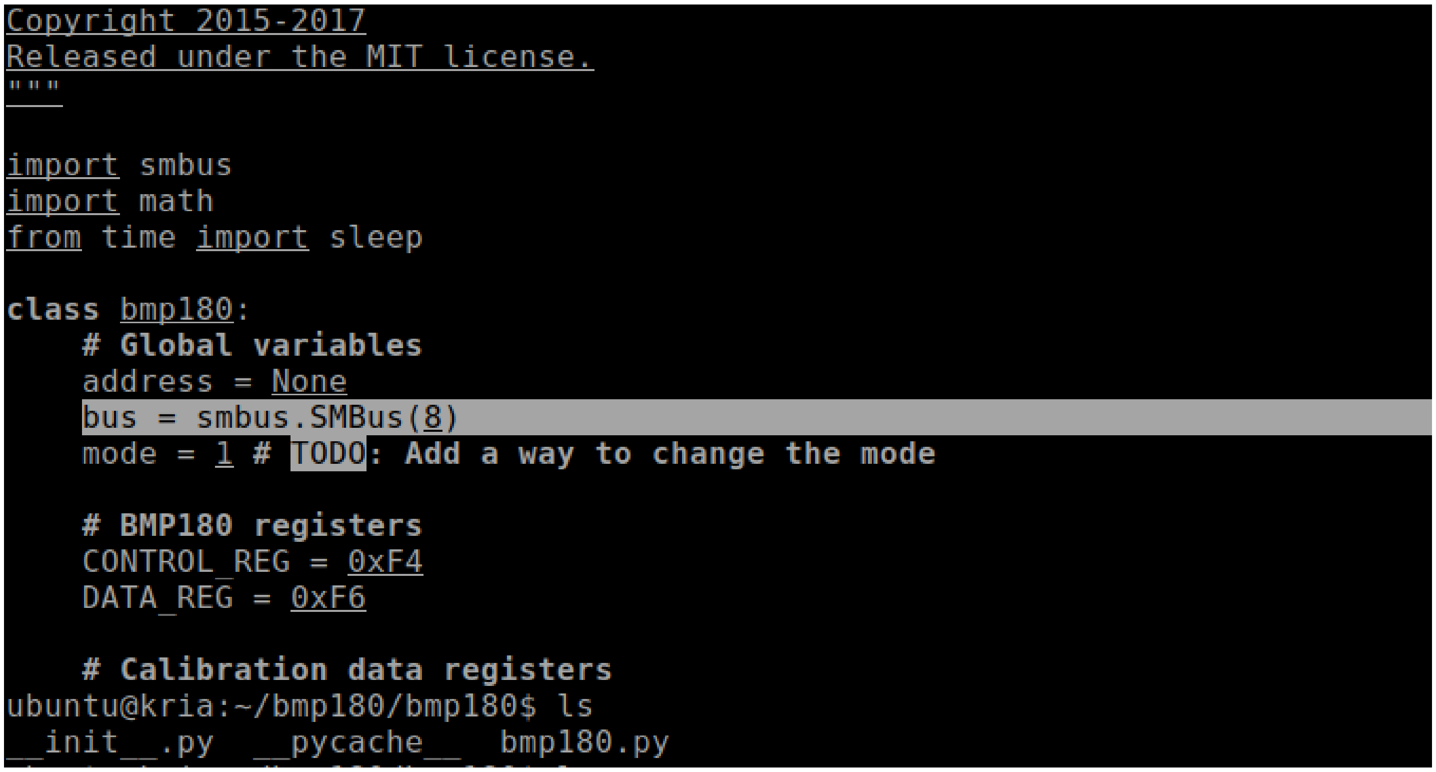
azure.iot.device python module is required to create a azure IoT device at the edge device. Install it using python pip3:

```
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
```

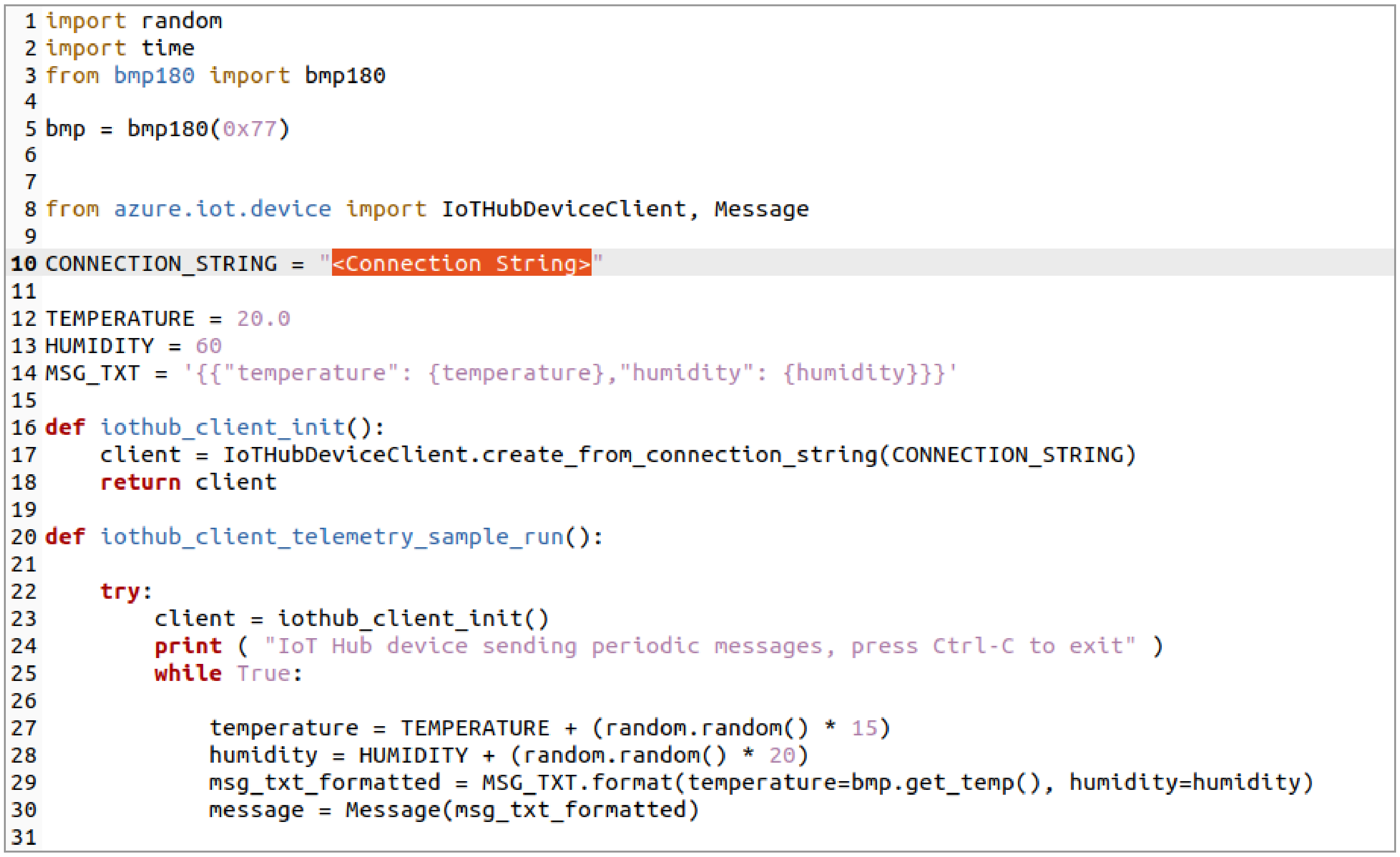
Install the bmp180 module by running:

```
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.

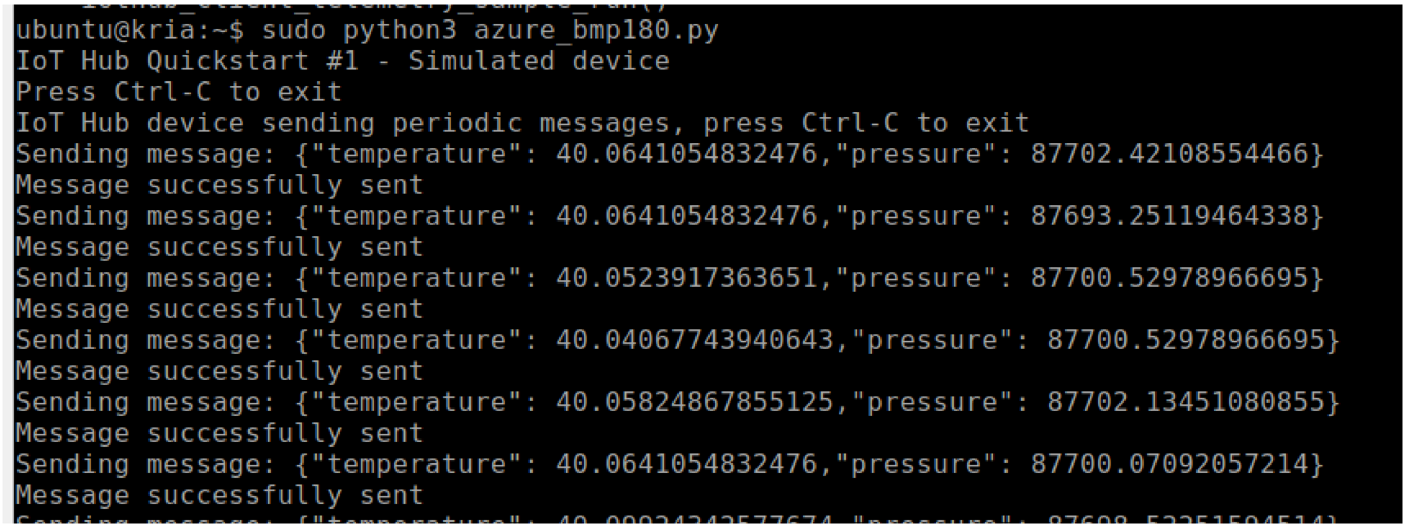


```
import random
import time
from bmp180 import bmp180

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.

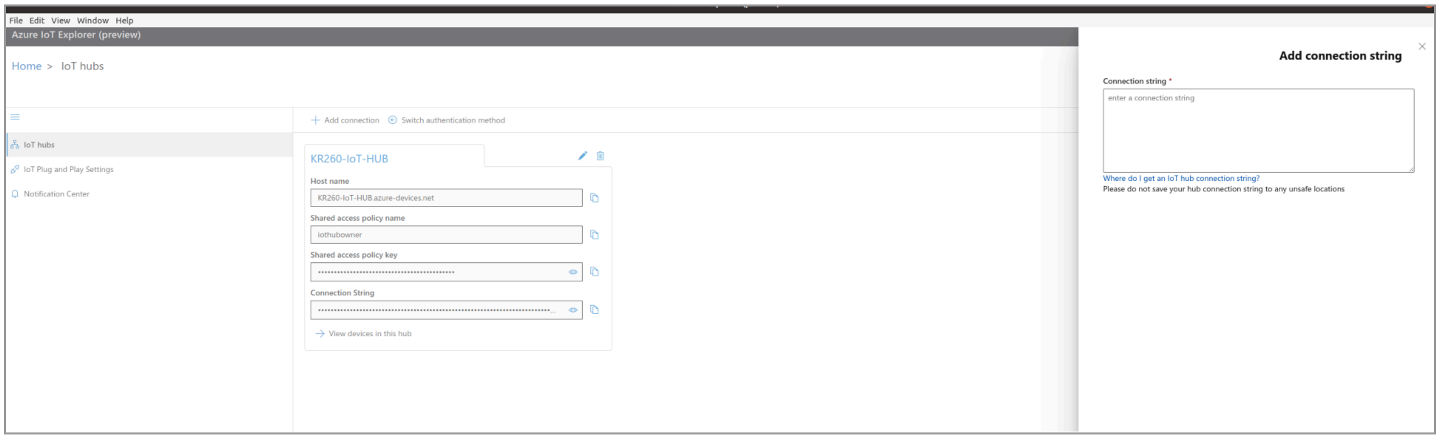


```
ubuntu@kria:~$ sudo python3 azure_bmp180.py
IoT Hub Quickstart #1 - Simulated device
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 40.0641054832476,"pressure": 87702.42108554466}
Message successfully sent
Sending message: {"temperature":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


File Edit View Window Help
Azure IoT Explorer (preview)
Home > IoT hubs
IoT hubs
IoT Plug and Play Settings
Notification Center
+ Add connection
Switch authentication method
View devices in this hub

KR260-IoT-HUB

Host name: KR260-IoT-HUB.azure-devices.net

Shared access policy name: iothubowner

Shared access policy key: ••

Connection String: ••…

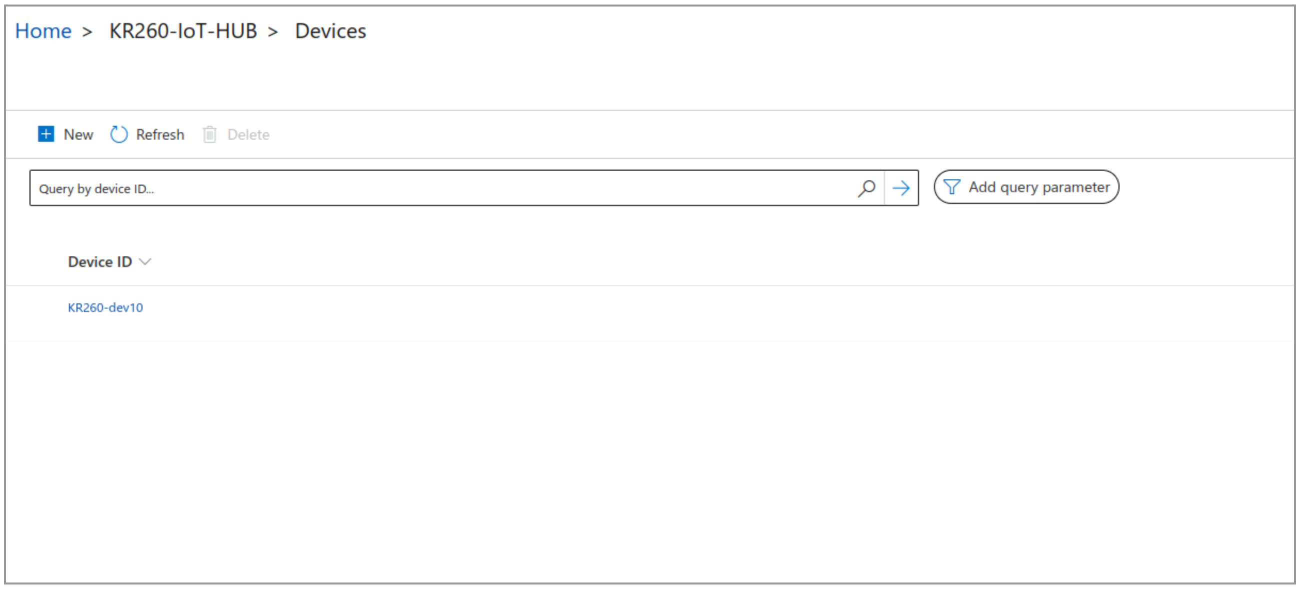
Add connection string

Connection string *

enter a connection string

Where do I get an IoT hub connection string?
Please do not save your hub connection string to any unsafe locations

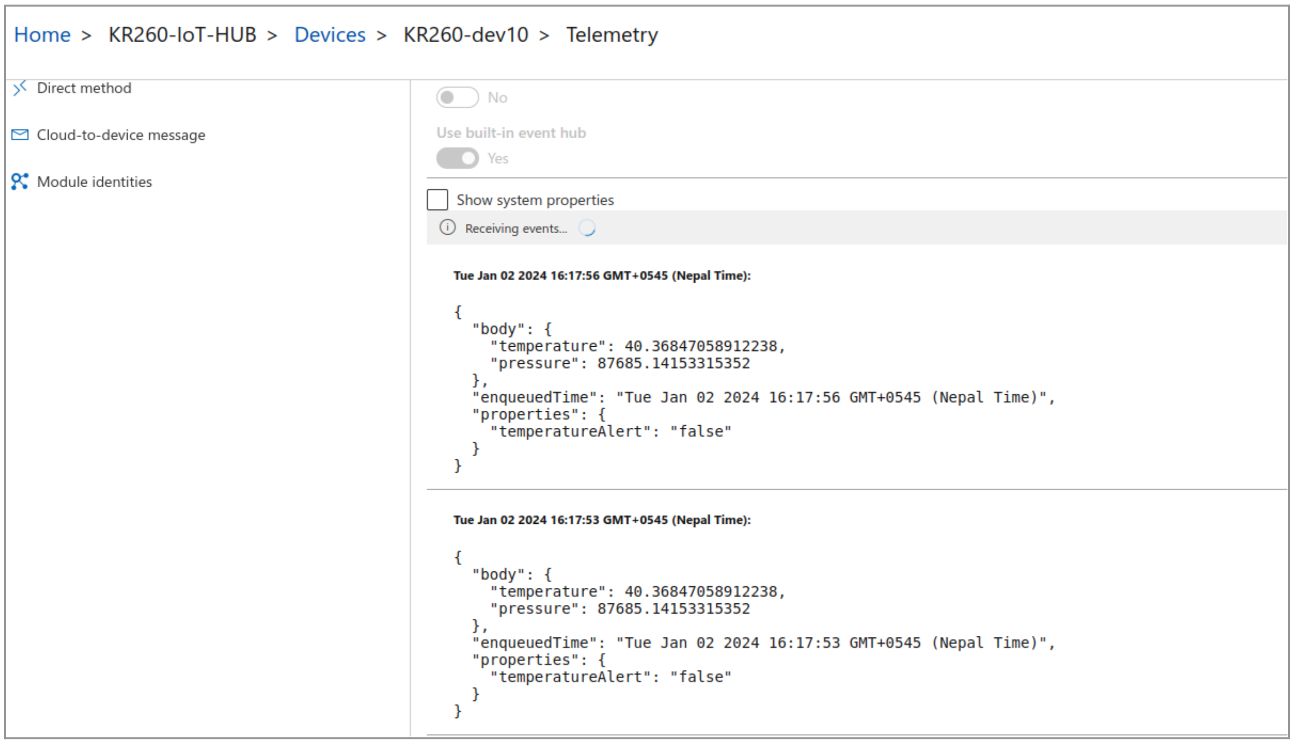
One can find the corresponding device list in the IoT HuB page of Azure IoT explorer application.



KR260-dev10

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.



Tue Jan 02 2024 16:17:56 GMT+0545 (Nepal Time):

```
{
  "body": {
    "temperature": 40.36847058912238,
    "pressure": 87685.14153315352
  },
  "enqueuedTime": "Tue Jan 02 2024 16:17:56 GMT+0545 (Nepal Time)",
  "properties": {
    "temperatureAlert": "false"
  }
}
```

Tue Jan 02 2024 16:17:53 GMT+0545 (Nepal Time):

```
{
  "body": {
    "temperature": 40.36847058912238,
    "pressure": 87685.14153315352
  },
  "enqueuedTime": "Tue Jan 02 2024 16:17:53 GMT+0545 (Nepal Time)",
  "properties": {
    "temperatureAlert": "false"
  }
}
```

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

***