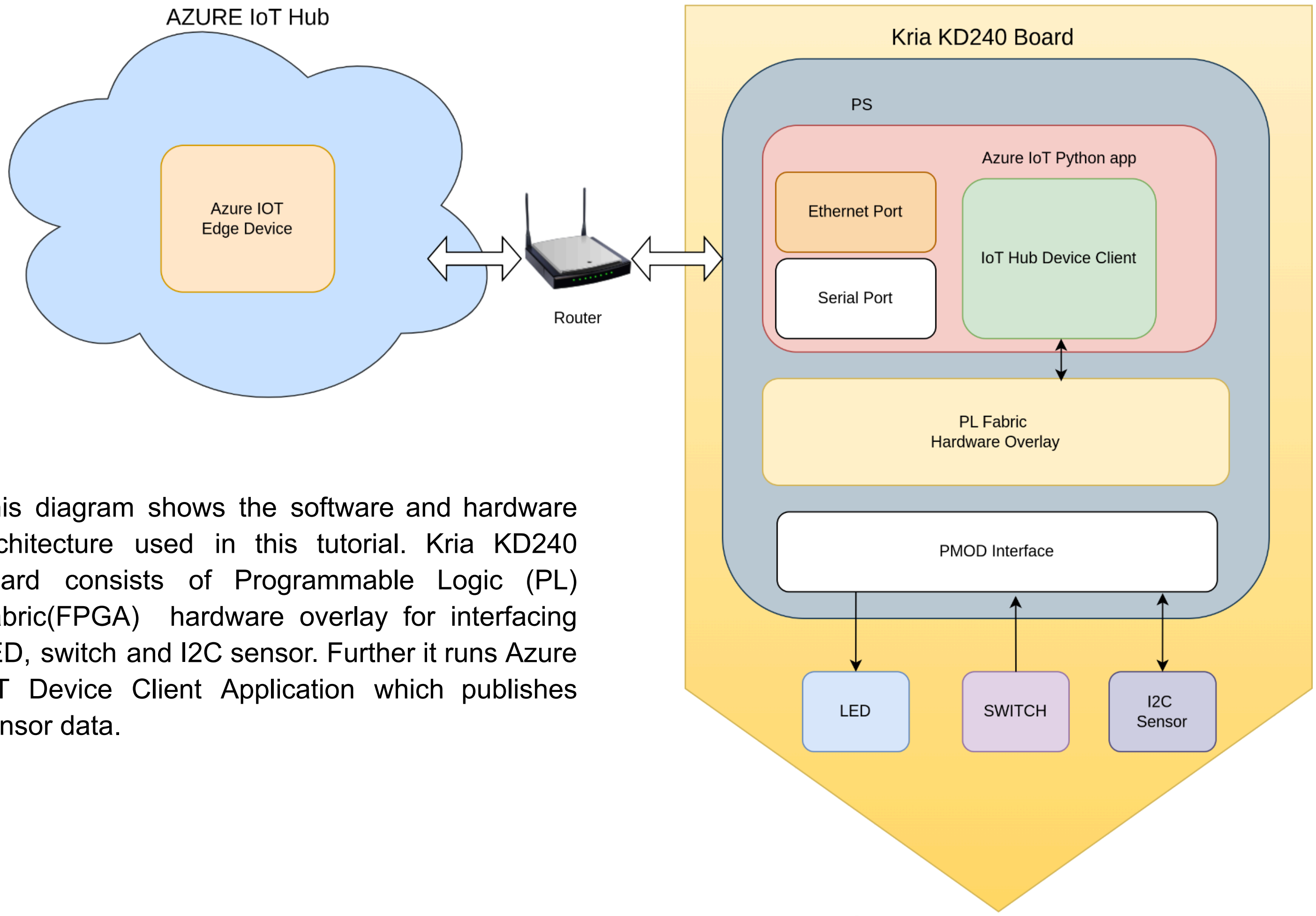
KD240 to Azure IoT Architecture



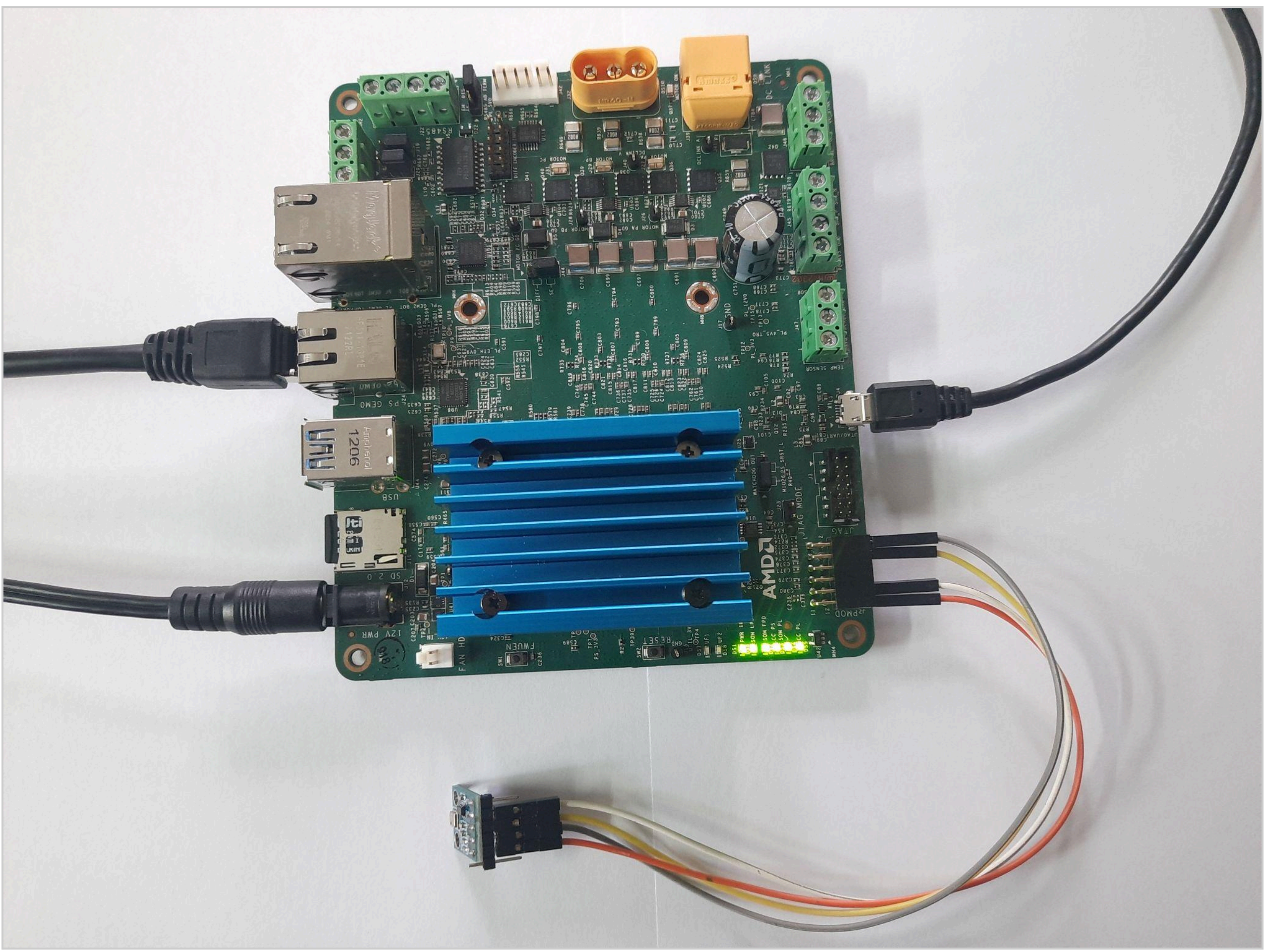
This diagram shows the software and hardware architecture used in this tutorial. Kria KD240 board consists of Programmable Logic (PL) Fabric(FPGA) hardware overlay for interfacing LED, switch and I2C sensor. Further it runs Azure IoT Device Client Application which publishes sensor data.

Required Hardware Components

1. Kria KD240 board
2. BMP180 Module (Available at [Amazon](#))
3. Connecting wires

Software requirements

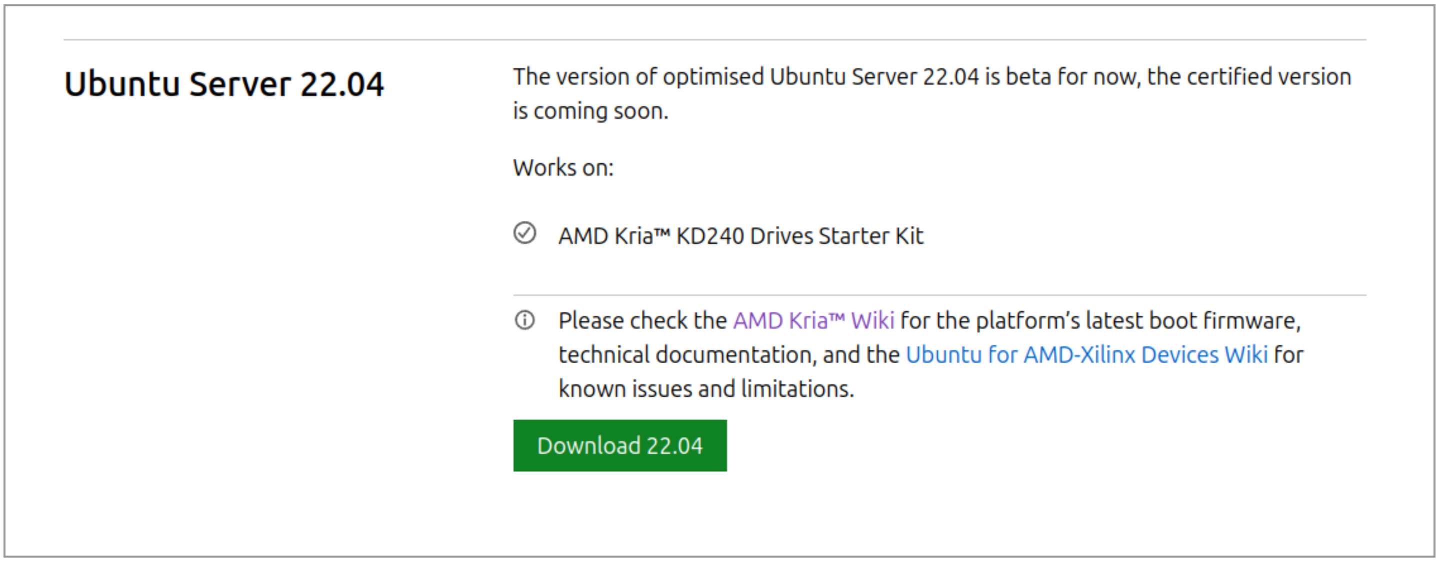
1. Ubuntu 22.04 for Kria KD240 board
2. Azure account



KD240 board connected to BMP180 through PMOD-I2C

Preparing Ubuntu 22.04 OS for KRIA KD240 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 KD240 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 KD240 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 KD240 device console.

Installing hardware overlay

Get the KD240 firmware folder. It contains:

- kd240-gpio-i2c.bit.bin
- kd240-gpio-i2c.dtbo
- shell.json

Copy these file to the KD240 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 "kd240-gpio-i2c" at "/lib/firmware/xilinx" and copy the files in "kd240-gpio-i2c" folder.

```
cd /lib/firmware/xilinx
sudo mkdir kd240-gpio-i2c
sudo cp <kd240-firmware directory>/krc260_i2c* ./
sudo cp <kd240-firmware directory>/shell.json ./
```

Next, check the available fpga firmware using `xmutil listapps` command. `kd240-i2c` will be available in the list.

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

       kd240-gpio-i2c            XRT_FLAT           kd240-gpio-i2c           XRT_FLAT               (0+0)                  -1
     k24-starter-kits            XRT_FLAT         k24-starter-kits           XRT_FLAT               (0+0)                  0,
```

Next load the `kd240-gpio-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 kd240-gpio-i2c
```

```
ubuntu@kria:~$ sudo xmutil loadapp kd240-gpio-i2c
[  141.337484] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/firmware-name
[  141.347670] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /fpga-full/resets
[  141.357614] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/afi0
[  141.367136] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/clocking0
[  141.377081] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_0
[  141.387107] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_intc_1
[  141.397141] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_gpio_0
[  141.407176] OF: overlay: WARNING: memory leak will occur if overlay removed, property: /__symbols__/axi_iic_0
kd240-gpio-i2c: loaded to slot 0
```

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          xiic-i2c 80010000.i2c             I2C adapter
ubuntu@kria:~$
```

`i2c-2` 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-> 3 - I2C SCLK
PMOD1-> 1 - 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 KD240 PMOD port, use i2c utility tools to scan for the available devices in i2c-8 channel.

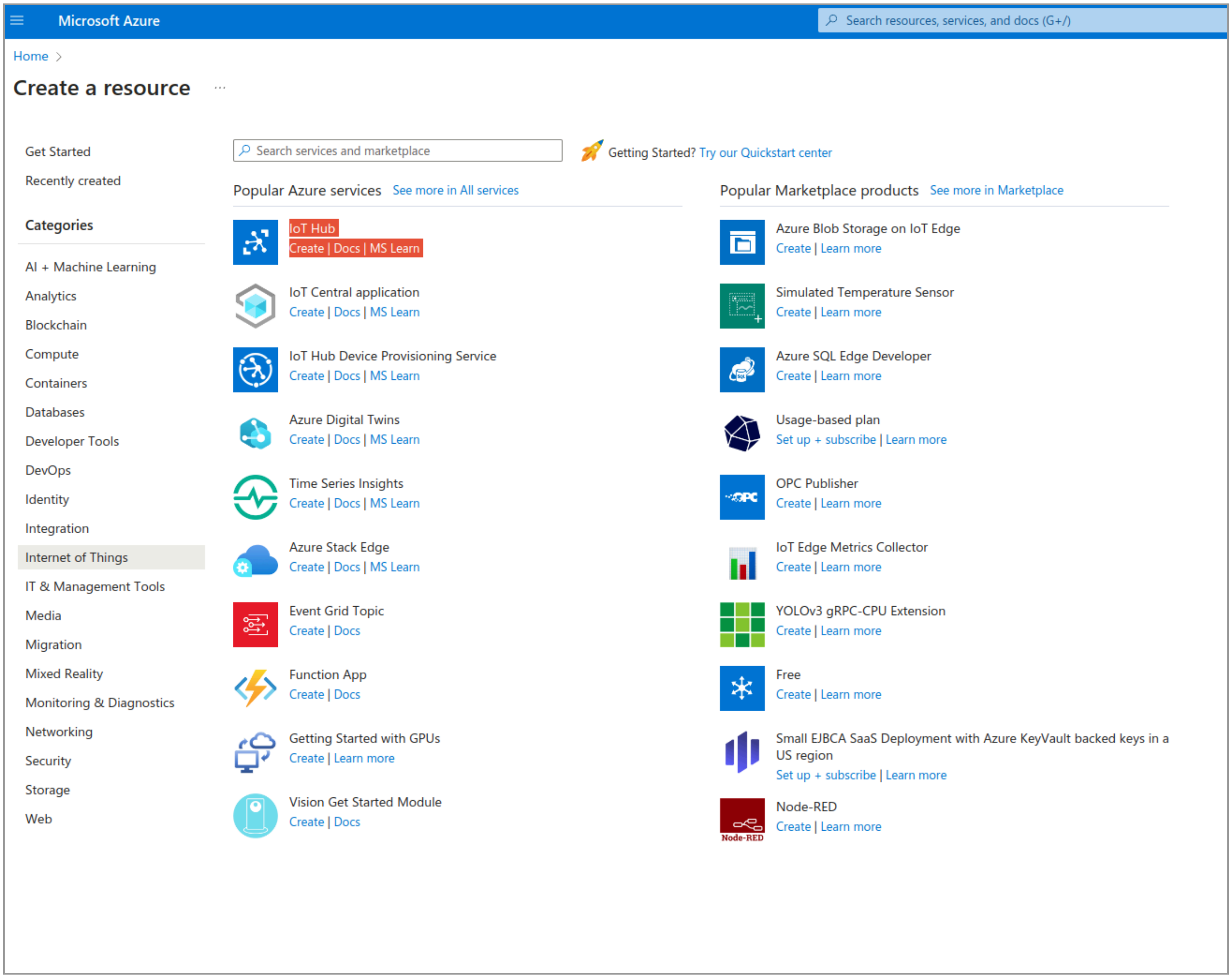
```
sudo i2cdetect -y 2
```

```
ubuntu@kria:~$ sudo i2cdetect -y 2
     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.

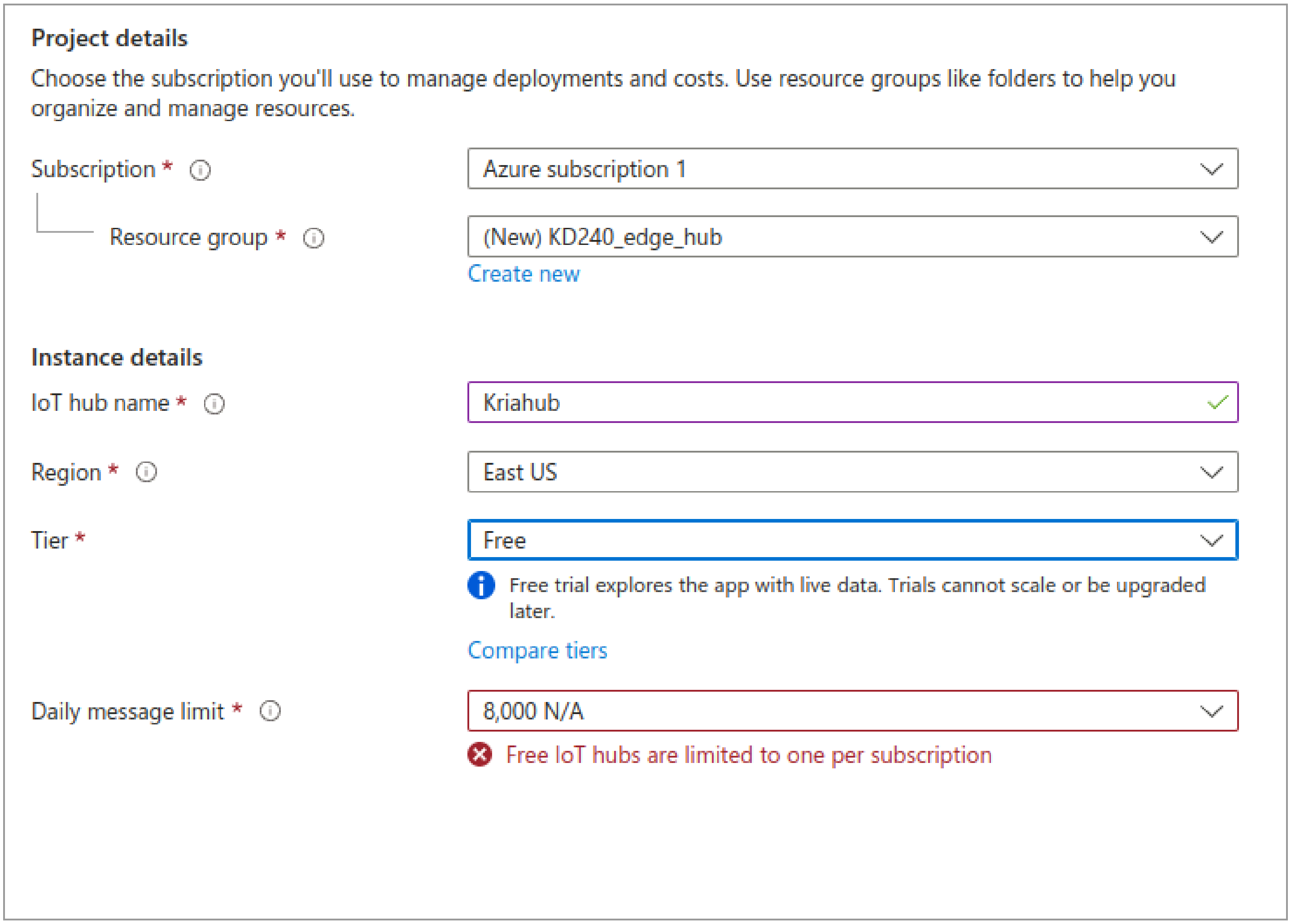
Create IoT Hub in Azure Portal:

- 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

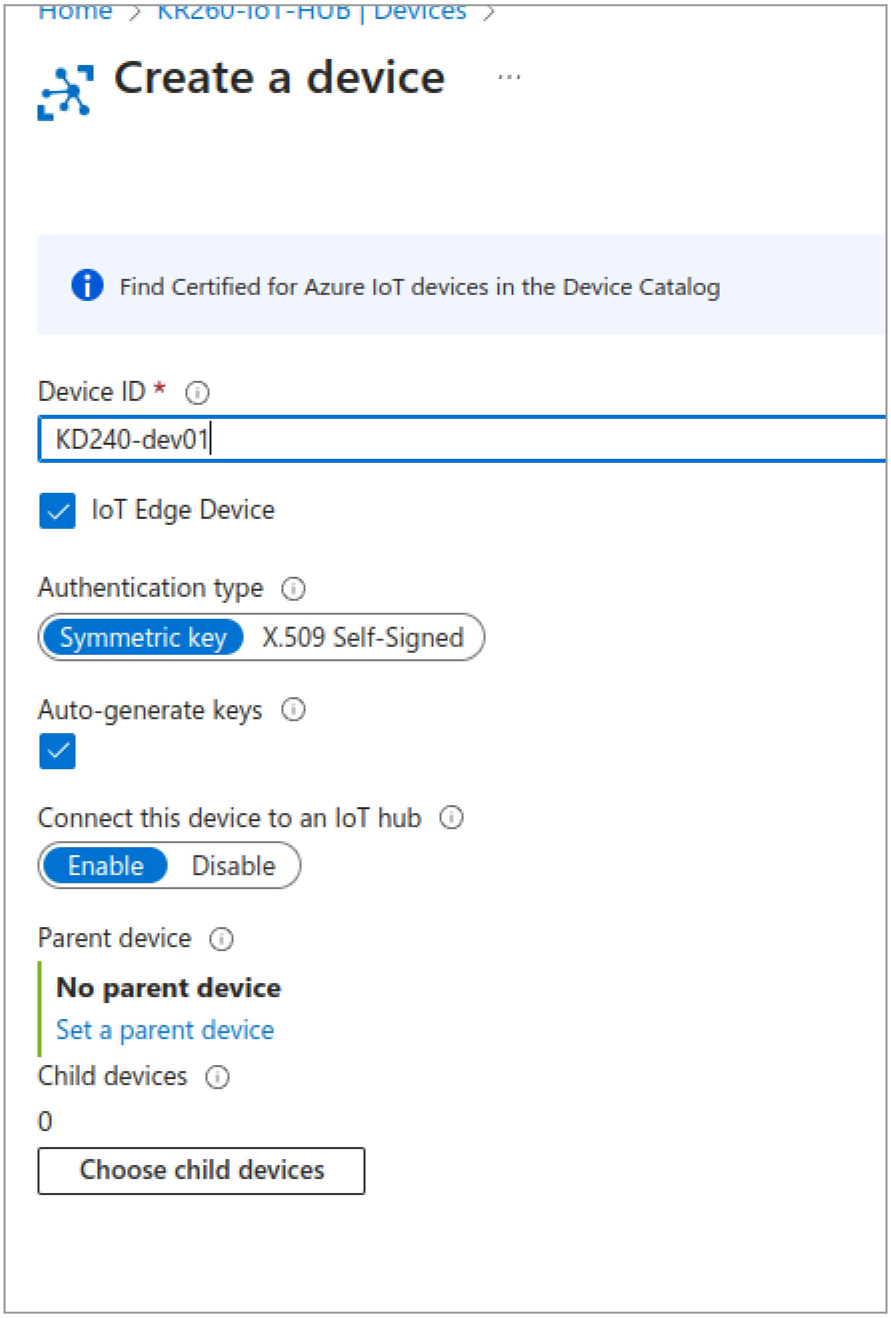
Next, create one IoT Hub Service and fill in the necessary details



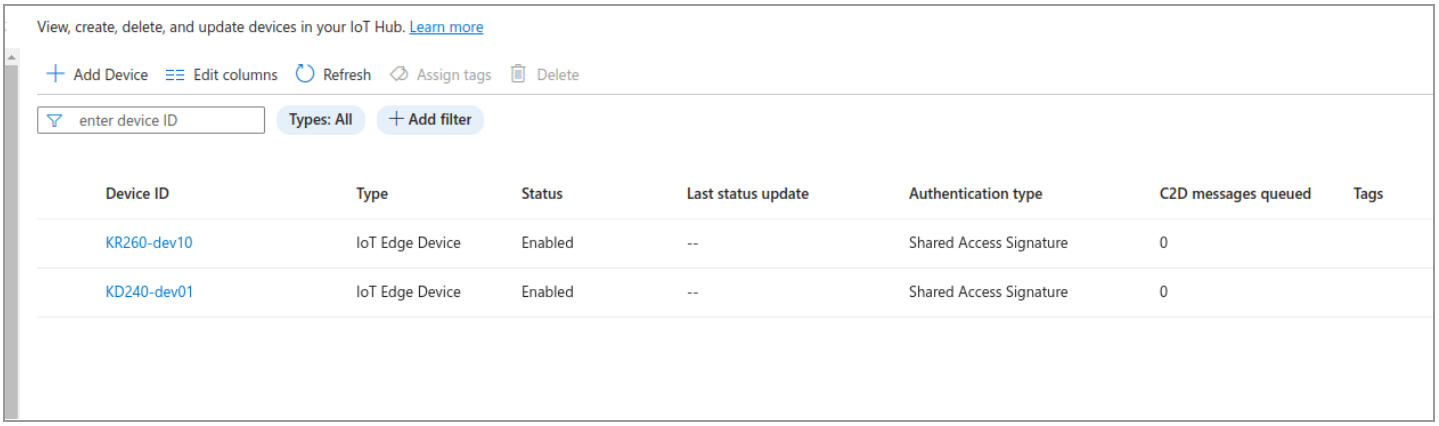
- Click on Review+ Create button to create the Azure IoT Hub.
- Next, create a device where you can actually receive some data from the hardware.

Create an IoT Device

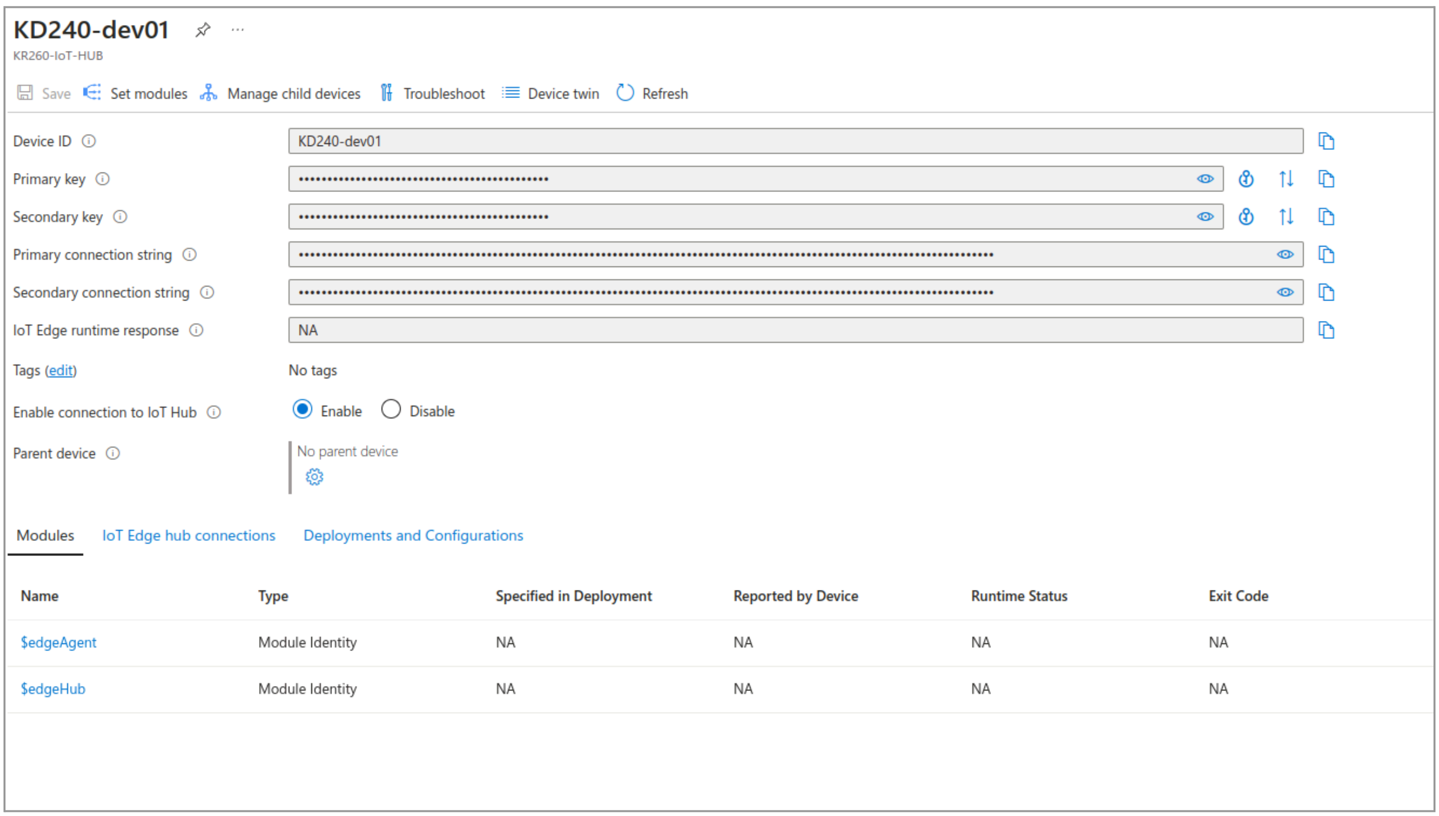
Go to the IoT Device and click on new, and give the device ID



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-2 channel by changing following lines:

```
import smbus
import math
from time import sleep

class bmp180:
    # Global variables
    address = None
    bus = smbus.SMBus(2)
    mode = 1 # TODO: Add a way to change the mode

    # BMP180 registers
    CONTROL_REG = 0xF4
    DATA_REG = 0xF6

    # Calibration data registers
"bmp180.py" 225L, 6914B written
ubuntu@kria:~/bmp180/bmp180$
```

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 KD240 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
```

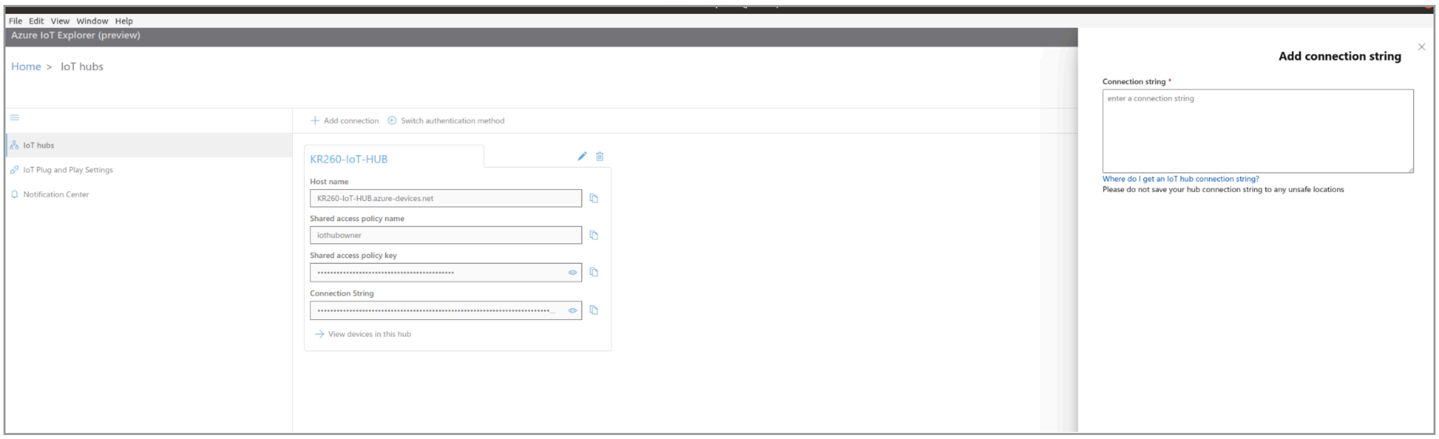
Here is the console log after a successful message send to Azure IoT hub.

```
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 39.75936867897661,"pressure": 87758.24057110936}
Message successfully sent
Sending message: {"temperature": 39.765232533114784,"pressure": 87737.39283234128}
Message successfully sent
Sending message: {"temperature": 39.75350468584645,"pressure": 87751.60383676378}
Message successfully sent
Sending message: {"temperature": 39.75936867897661,"pressure": 87751.60383676378}
Message successfully sent
```

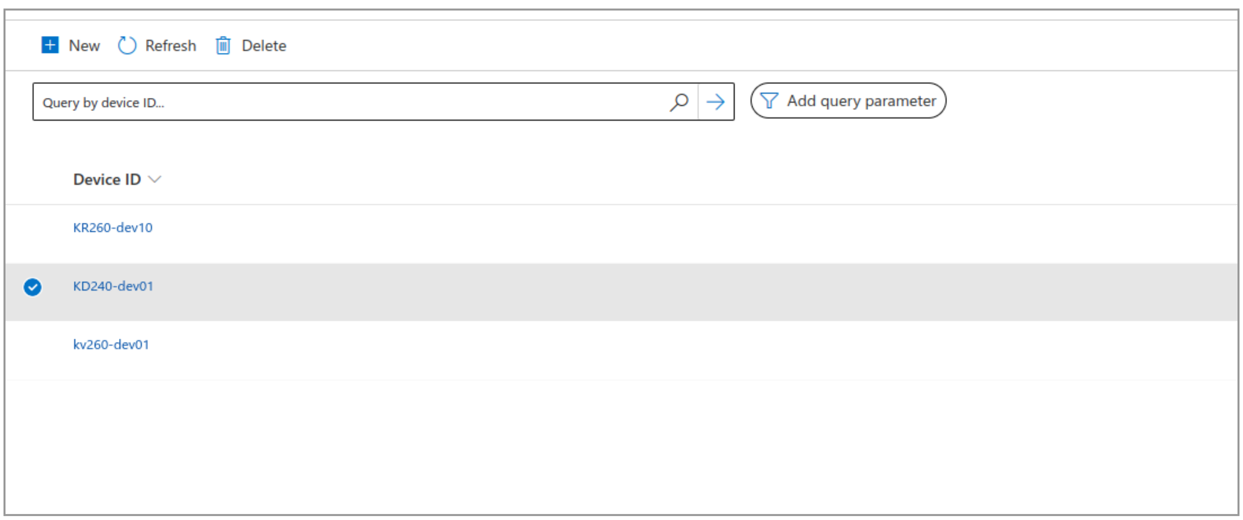
Viewing message in Host Machine

For viewing the message published by Azure IoT Device in KD240, 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:

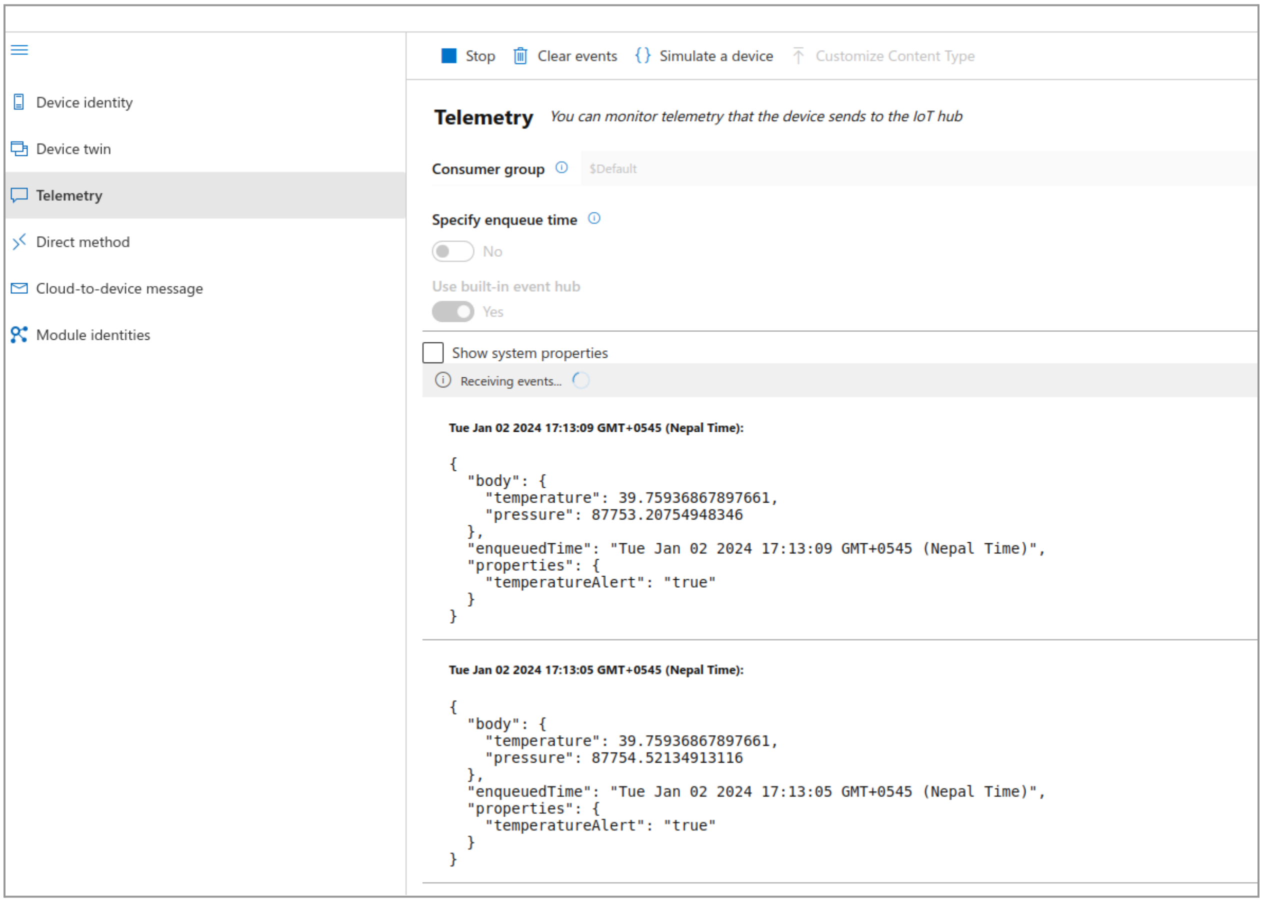


One can find the corresponding device list in the IoT HuB page of Azure IoT explorer application.



Just click onto the device to view the device information and also the message send by python application running in the KD240 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.



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

***