

Preparing Ubuntu 22.04 OS for KRIA KV260 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.

[Download 22.04 LTS](#)

[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 KV260 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 KV260 connected to the internet.

For initial login here are the Login Details:

Username : ubuntu

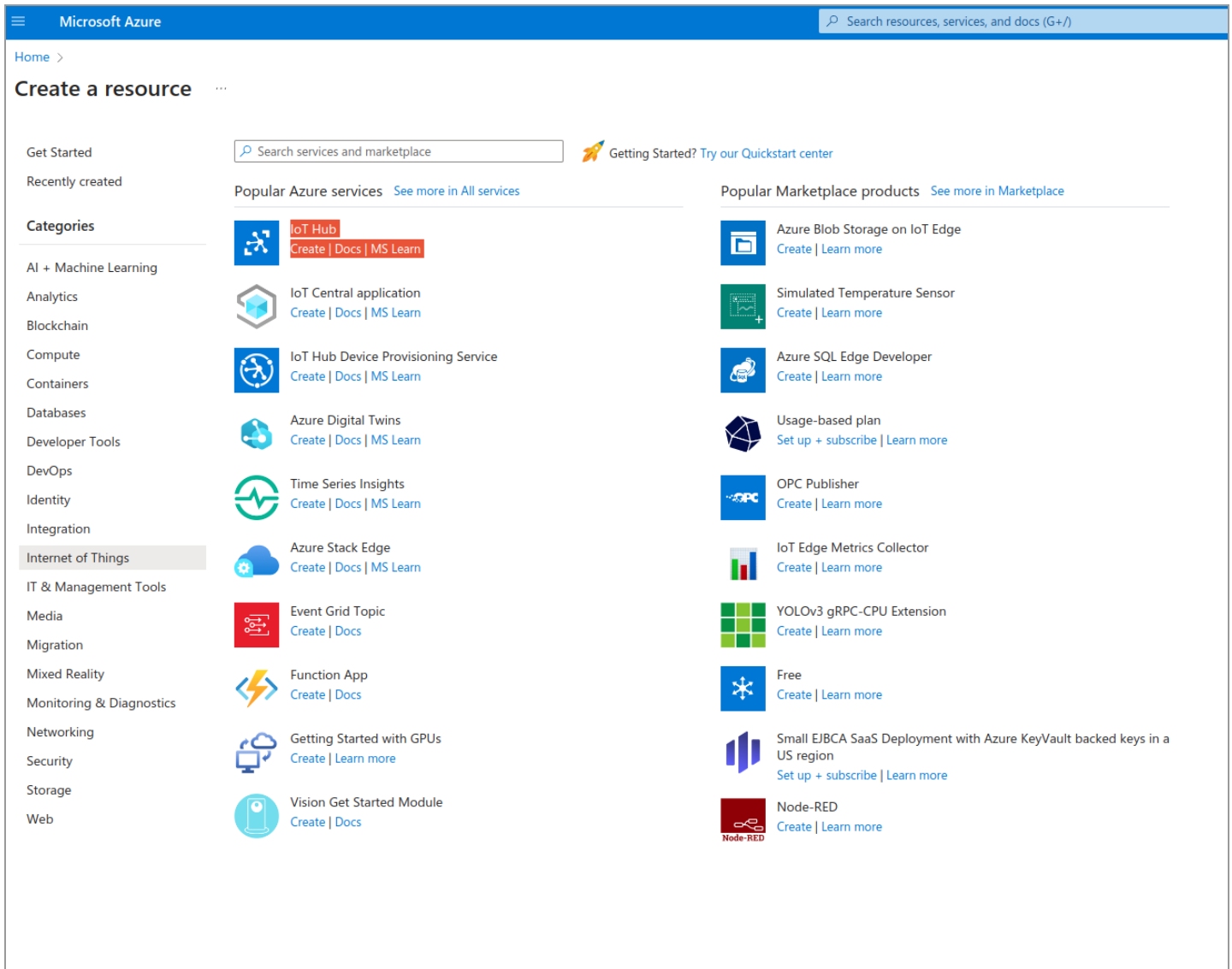
Password: ubuntu

This will ask to change the password. So update the password and login the system.

After successful login, one can access the KV260 device console.

Create IoT Hub in Azure Portal:

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



The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes the 'Microsoft Azure' logo and a search bar. The left sidebar lists various categories, with 'Internet of Things' highlighted. The main content area is titled 'Create a resource' and features a search bar for services and marketplace. Below this, there are two columns of service recommendations. The 'Popular Azure services' column lists several IoT-related services, with 'IoT Hub' at the top, followed by 'IoT Central application', 'IoT Hub Device Provisioning Service', 'Azure Digital Twins', 'Time Series Insights', 'Azure Stack Edge', 'Event Grid Topic', 'Function App', 'Getting Started with GPUs', and 'Vision Get Started Module'. The 'Popular Marketplace products' column lists products like 'Azure Blob Storage on IoT Edge', 'Simulated Temperature Sensor', 'Azure SQL Edge Developer', 'Usage-based plan', 'OPC Publisher', 'IoT Edge Metrics Collector', 'YOLOv3 gRPC-CPU Extension', 'Free', 'Small EJBCA SaaS Deployment with Azure KeyVault backed keys in a US region', and 'Node-RED'. Each service or product entry includes an icon, a name, and links to 'Create', 'Docs', or 'MS Learn'.

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

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) KV260_edge_group ▼

[Create new](#)

Instance details

IoT hub name * ⓘ

Kriahub ✓

Region * ⓘ

East US ▼

Tier *

Standard (most popular) ▼

[Compare tiers](#)

Daily message limit * ⓘ

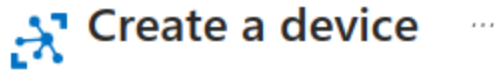
400,000 N/A ▼


[See all options](#)


- 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




 Find Certified for Azure IoT devices in the Device Catalog


Device ID * 

kv260-dev01


☒ IoT Edge Device

Authentication type 


☒ Symmetric key ☐ X.509 Self-Signed

Auto-generate keys 


☒

Connect this device to an IoT hub 

☒ Enable ☐ Disable

Parent device 

No parent device
[Set a parent device](#)

Child devices 

0

rix

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

View, create, delete, and update devices in your IoT Hub. [Learn more](#)

[+ Add Device](#) [≡ Edit columns](#) [↺ Refresh](#) [↻ Assign tags](#) [🗑 Delete](#)

enter device ID

Device ID	Type	Status	Last status update	Authentication type	C2D messages queued	Tags
KR260-dev10	IoT Edge Device	Enabled	--	Shared Access Signature	0	
KD240-dev01	IoT Edge Device	Enabled	--	Shared Access Signature	0	
kv260-dev01	IoT Edge Device	Enabled	--	Shared Access Signature	0	

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

kv260-dev01

Save

Set modules

Manage child devices

Troubleshoot

Device twin

Refresh

Device ID

kv260-dev01

Primary key

Secondary key

Primary connection string

Secondary connection string

IoT Edge runtime response

NA

Tags

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	Module Identity	NA	NA	NA	NA
\$edgeHub	Module Identity	NA	NA	NA	NA

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

Installing hardware overlay

Get the KV260 firmware folder. It contains:

- kv260-gpio-i2c.bit.bin
- kv260-gpio-i2c.dtbo
- shell.json

Copy these file to the KV260 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 "kv260-gpio-i2c" at "/lib/firmware/xilinx" and copy the files in "kv260-gpio-i2c" folder.

```
cd /lib/firmware/xilinx
sudo mkdir kv260-gpio-i2c
sudo cp <kv260-firmware directory>/kv260-gpio-i2c* ./
sudo cp <kv260-firmware directory>/shell.json ./
```

Next, check the available fpga firmware using `xmutil listapps` command. `kv260-gpio-i2c` will be available in the list.

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 `kv260-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 kv260-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
```

i2c-1	i2c	Cadence I2C at ff030000	I2C adapter
i2c-2	i2c	xiic-i2c 80010000.i2c	I2C adapter

`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 KV260 PMOD port, use i2c utility tools to scan for the available devices in i2c-8 channel.

```
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

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.

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 KV260 board.

Next update the "CONNECTION STRING" with the above Primary Connection string.

```
1 import random
2 import time
3 from bmp180 import bmp180
4
5 bmp = bmp180(0x77)
6
7
8 from azure.iot.device import IoTHubDeviceClient, Message
9
10 CONNECTION_STRING = "<Connection String>"
11
12 TEMPERATURE = 20.0
13 HUMIDITY = 60
14 MSG_TXT = '{"temperature": {temperature},"humidity": {humidity}}'
15
16 def iot_hub_client_init():
17     client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
18     return client
19
20 def iot_hub_client_telemetry_sample_run():
21
22     try:
23         client = iot_hub_client_init()
24         print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit" )
25         while True:
26
27             temperature = TEMPERATURE + (random.random() * 15)
28             humidity = HUMIDITY + (random.random() * 20)
29             msg_txt_formatted = MSG_TXT.format(temperature=bmp.get_temp(), humidity=humidity)
30             message = Message(msg_txt_formatted)
31
```

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.

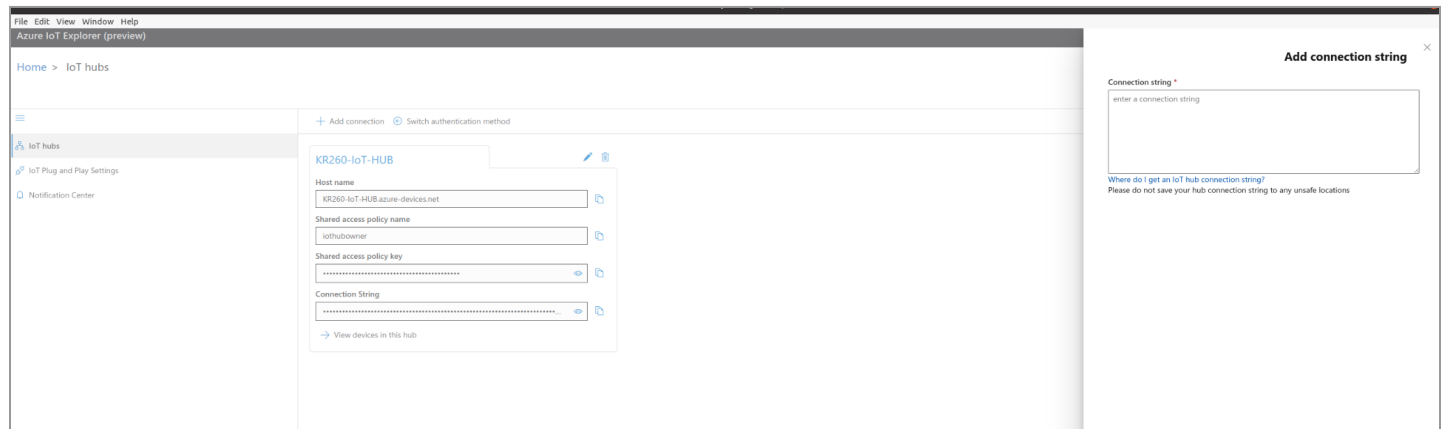
```
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 39.75936867897661,"pressure": 87758.24057110936}
Message successfully sent
Sending message: {"temperature": 39.765232533114784,"pressure": 87737.39283234128}
Message successfully sent
Sending message: {"temperature": 39.75350468584645,"pressure": 87751.60383676378}
Message successfully sent
Sending message: {"temperature": 39.75936867897661,"pressure": 87751.60383676378}
Message successfully sent
Sending message: {"temperature": 39.765232533114784,"pressure": 87747.88433116772}
Message successfully sent
Sending message: {"temperature": 39.77695982453383,"pressure": 87754.5890678738}
Message successfully sent
Sending message: {"temperature": 39.765232533114784,"pressure": 87752.98531510356}
Message successfully sent
Sending message: {"temperature": 39.75936867897661,"pressure": 87754.52134913116}
Message successfully sent
Sending message: {"temperature": 39.75936867897661,"pressure": 87753.20754948346}
```

Viewing message in Host Machine

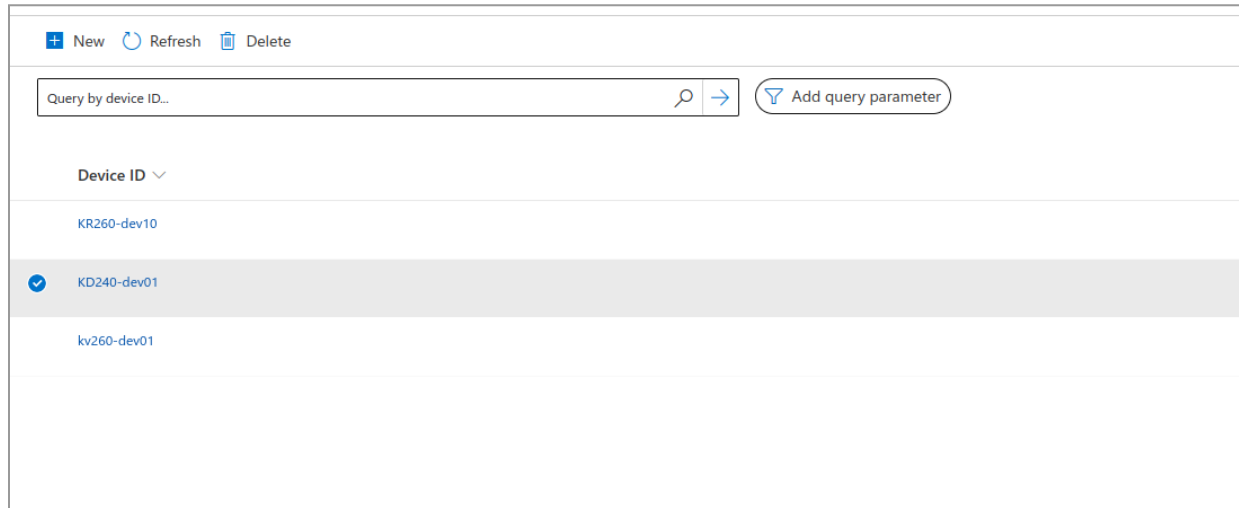
For viewing the message published by Azure IoT Device in KV260, one can use Azure IoT explorer available in following link:

<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 KV260 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.

