

## Creating Petalinux Project

### Create KV260 Petalinux Project from BSP

In host machine, create the petalinux project using BSP available at following download link: Run following commands to create petalinux project:

<https://www.xilinx.com/member/forms/download/xef.html?filename=xilinx-kv260-starterkit-v2023.1-05080224.bsp>

```
petalinux-create -t project -s xilinx-kv260-starterkit-v2023.1-05080224.bsp -n
kv260-2023_1-default
cd kv260-2023_1-default
```

Here “kv260-2023\_1-default” is the directory created by petalinux-create command, you can change the name according to your need. This directory is the petalinux-project base directory, which we will be using in further steps.

Next build the project after sourcing the Petalinux 2023.1 environment:

```
source <Path to Petalinux 2023.1>/settings.sh
petalinux-config --silentconfig
petalinux-build
```

Here is the console log after running the petalinux-build.

```
sanam@sanam-ubuntu:~/kv260-2023_1-default$ petalinux-build
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Sourcing build environment
[INFO] Generating workspace directory
INFO: Bitbake petalinux-image-minimal
NOTE: Started PRServer with DBFile: /media/sanam/workspace2/sanam/kv260-p2022-2-default/build/cache/prserv.sqlite3, Address: 127.0.0.1:46633, PID: 558748
loading cache: 100% |
loaded 0 entries from dependency cache.
Parsing recipes: 100% |#####| Time: 0:02:02
Parsing of 4532 bb files complete (0 cached, 4532 parsed), 6025 targets, 620 skipped, 1 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
Initialising tasks: 100% |#####| Time: 0:00:18
Checking sstate mirror object availability: 100% |#####| Time: 0:00:39
State Summary: Wanted 1086 Local 29 Network 653 Missed 404 Current 2726 (62% match, 89% complete)
Downloading 2 state sstate objects for arch xilinx_kv260_kr: 100% |#####| Time: 0:00:00
NOTE: Executing tasks
NOTE: Tasks Summary: Attempted 9498 tasks of which 9483 didn't need to be rerun and all succeeded.
INFO: Failed to copy built images to tftp dir: /tftpboot
[INFO] Successfully built project
sanam@sanam-ubuntu:~/kv260-p2022-2-default$
```

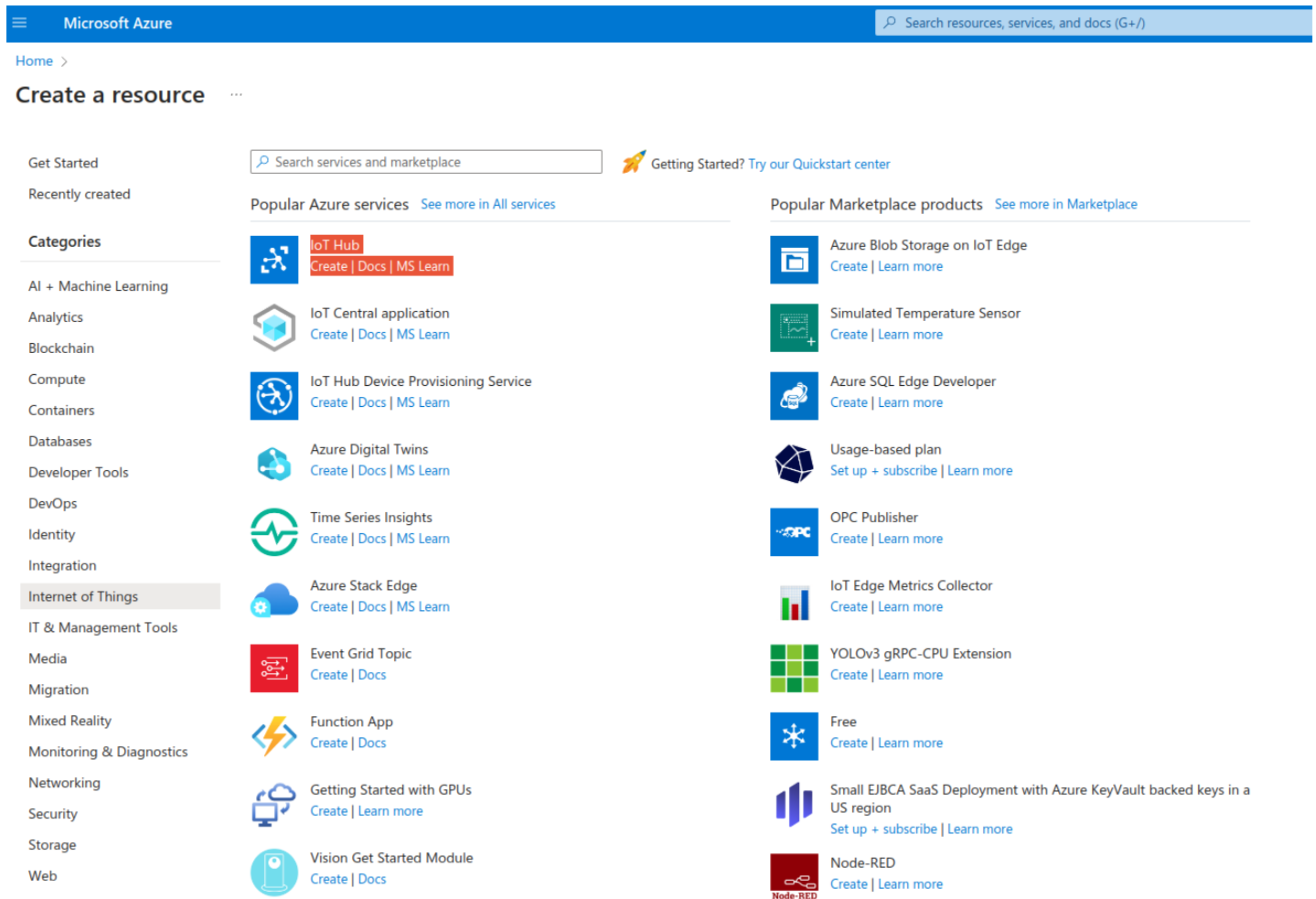
Next create the SD card image with the following commands:

```
package --wic --images-dir images/linux/ --bootfiles
"ramdisk.cpio.gz.u-boot,boot.scr,Image,system.dtb,system-zynqmp-sck-kv-g-revB.dtb"
--disk-name "mmcblk1"
```

This will create the petalinux-sdimage.wic image at <petalinux project directory>/image/linux folder. Copy the created wic image to SD card using tools like Balena Etcher.

## 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

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 \* ⓘ

Resource group \* ⓘ

[Create new](#)

## Instance details

IoT hub name \* ⓘ

Region \* ⓘ

Tier \*

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

[Compare tiers](#)

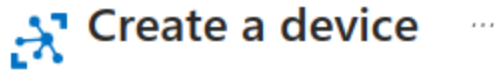
Daily message limit \* ⓘ


**x** 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




 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](#)

<input type="text" value="enter device ID"/>	Types: All	<a href="#">+ Add filter</a>				
Device ID	Type	Status	Last status update	Authentication type	C2D messages queued	Tags
<a href="#">KR260-dev10</a>	IoT Edge Device	Enabled	--	Shared Access Signature	0	
<a href="#">KD240-dev01</a>	IoT Edge Device	Enabled	--	Shared Access Signature	0	
<a href="#">kv260-dev01</a>	IoT Edge Device	Enabled	--	Shared Access Signature	0	

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

kv260-dev01

KR260-IOT-HUB

[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
<a href="#">\$edgeAgent</a>	Module Identity	NA	NA	NA	NA
<a href="#">\$edgeHub</a>	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.

## Adding python application in KRIA

Copy the simulated\_sensor.py example code in KV260 board.

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

```
1 import random
2 import time
3
4 from azure.iot.device import IoTHubDeviceClient, Message
5
6 CONNECTION_STRING = "<Connection String>"
7
8 TEMPERATURE = 20.0
9 HUMIDITY = 60
10 MSG_TXT = '{"temperature": {temperature},"humidity": {humidity}}'
11
12 def iot_hub_client_init():
13     client = IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
14     return client
15
16 def iot_hub_client_telemetry_sample_run():
17
18     try:
19         client = iot_hub_client_init()
```

Then run the simulated application in console:

```
python3 simulated_sensor.py
```

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

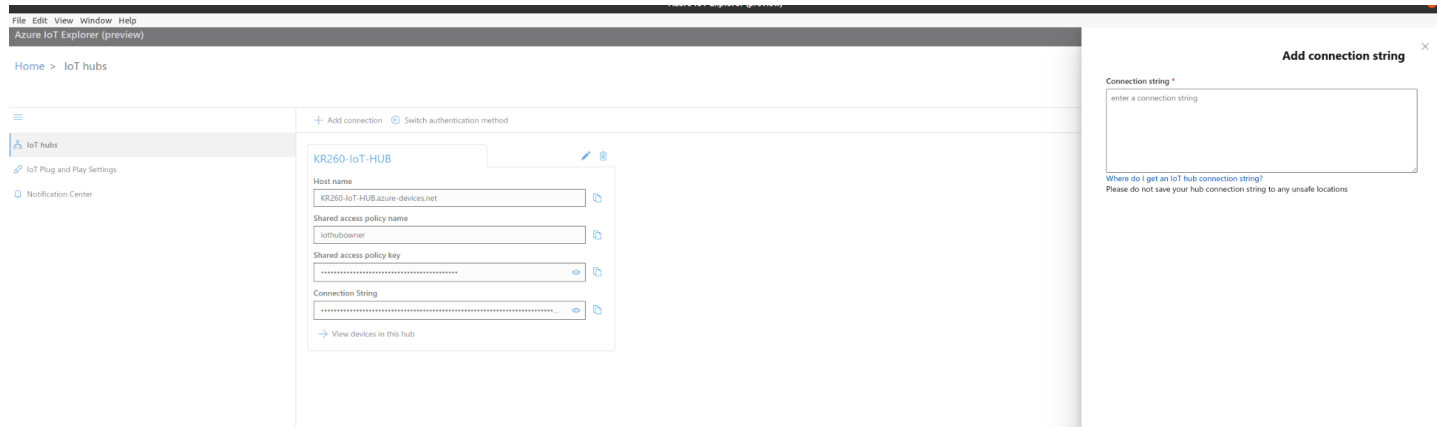
```
IoT Hub Quickstart #1 - Simulated device
Press Ctrl-C to exit
IoT Hub device sending periodic messages, press Ctrl-C to exit
Sending message: {"temperature": 21.869834376404423,"humidity": 74.29759396046798}
Message successfully sent
Sending message: {"temperature": 32.86165169899766,"humidity": 76.24063097582776}
Message successfully sent
Sending message: {"temperature": 26.783131268254383,"humidity": 64.12216333418469}
Message successfully sent
█
```

## 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:

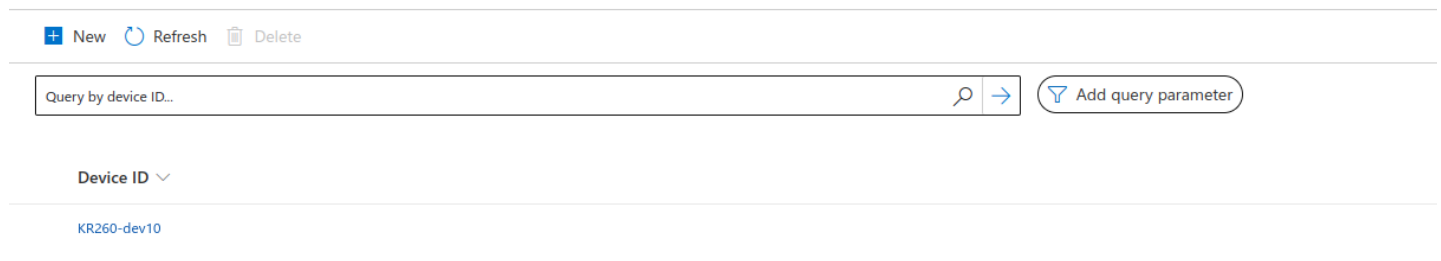
<https://github.com/Azure/azure-iot-explorer/releases>

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.

[Home](#) > [KR260-IoT-HUB](#) > [Devices](#)



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.

Device identity

Device twin

Telemetry

Direct method

Cloud-to-device message

Module identities

Stop

Clear events

Simulate a device

Customize Content Type

Telemetry

You can monitor telemetry that the device sends to the IoT hub

Consumer group

\$Default

Specify enqueue time

No

Use built-in event hub

Yes

Show system properties

Receiving events...

Fri Dec 15 2023 17:06:03 GMT+0545 (Nepal Time):

```
{
  "body": {
    "temperature": 25.25847962061951,
    "humidity": 62.77776214518302
  },
  "enqueuedTime": "Fri Dec 15 2023 17:06:03 GMT+0545 (Nepal Time)",
  "properties": {
    "temperatureAlert": "false"
  }
}
```

Fri Dec 15 2023 17:05:59 GMT+0545 (Nepal Time):

```
{
  "body": {
    "temperature": 20.3589672917612,
    "humidity": 73.20813395493155
  },
  "enqueuedTime": "Fri Dec 15 2023 17:05:59 GMT+0545 (Nepal Time)",
  "properties": {
    "temperatureAlert": "false"
  }
}
```