Creating Petalinux Project

Create KV260 Petalinux Project from BSP

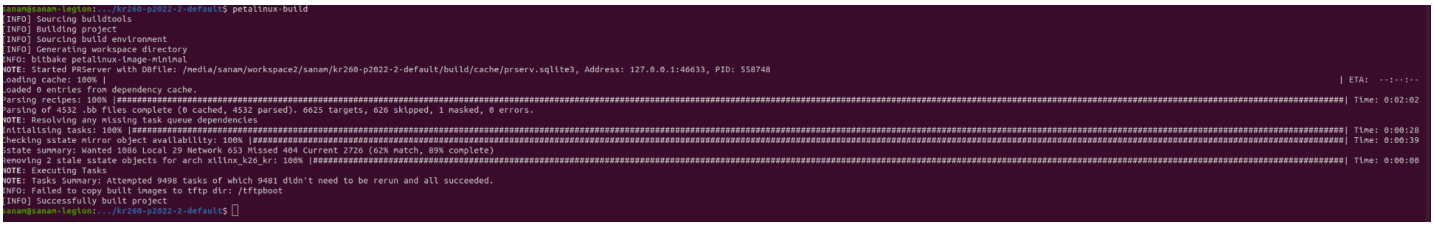
In host machine, create the petalinux project using BSP available at following download link: Run following commands to create petalinux project:
https://www.xilinx.com/member/forms/download/xef.html?filename=xilinx-kv260-starterkit-v2023.1-05080224.bsp

```
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.



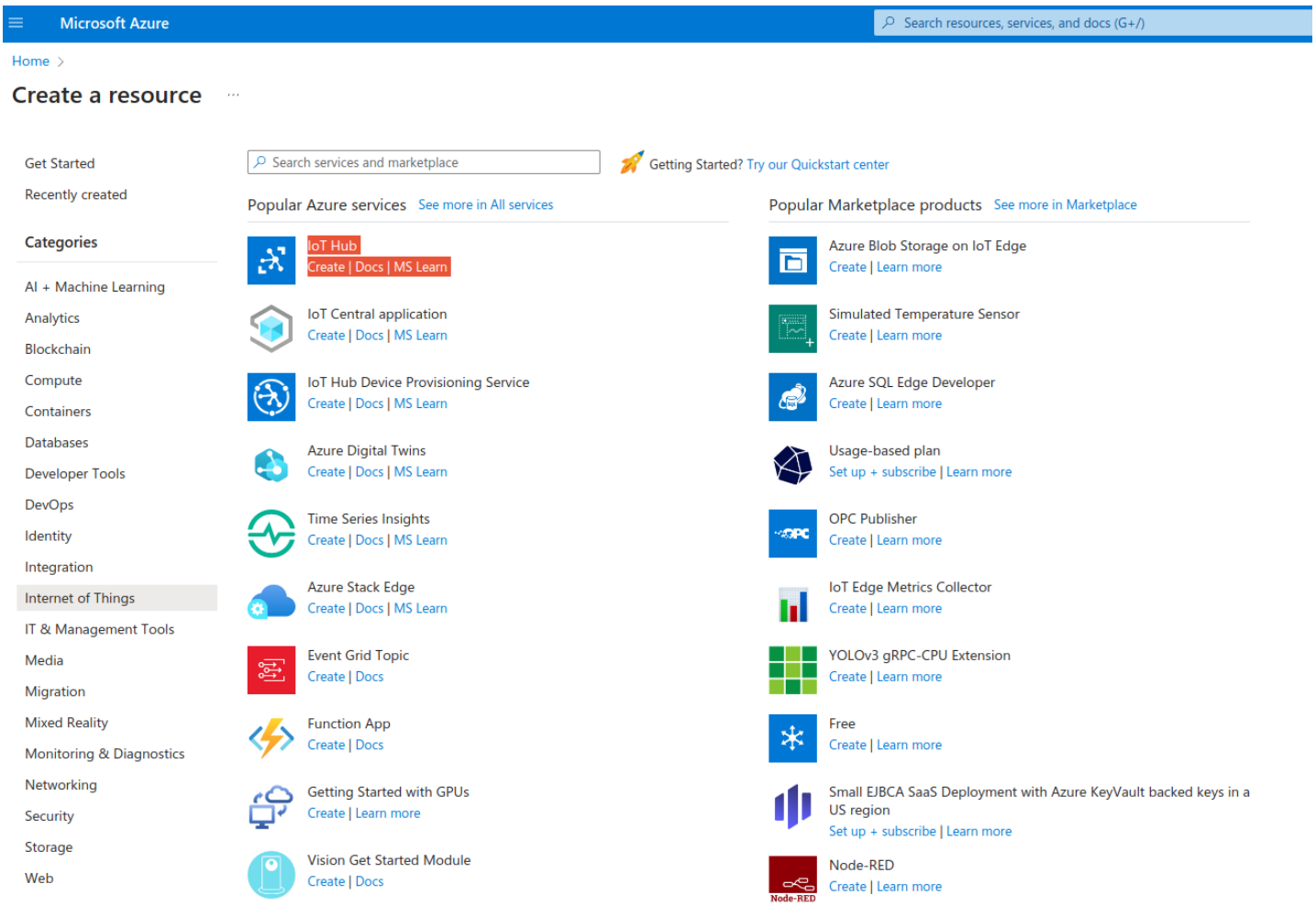
Next create the SD card image with the following commands:

```
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.



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) KR260_edge_group

Create new

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.

Compare tiers

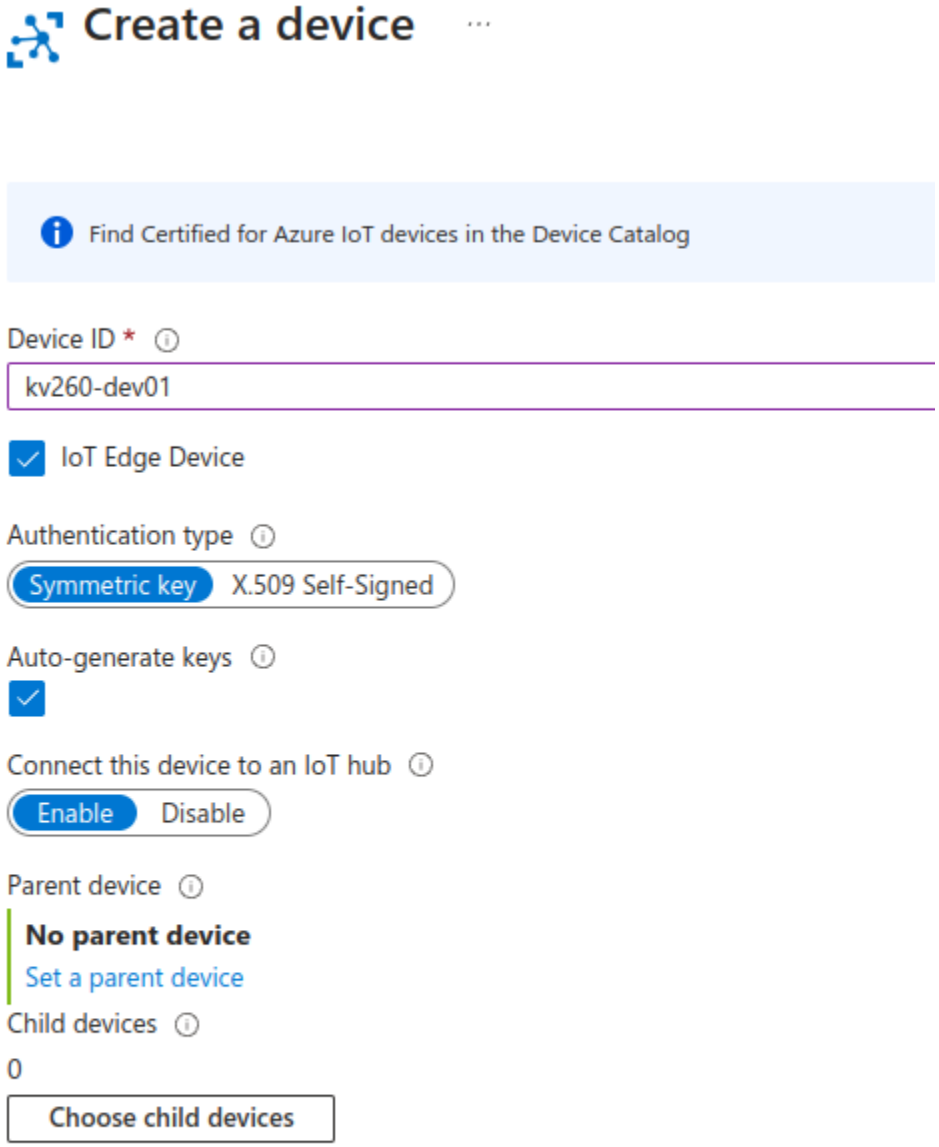
Daily message limit * — 8,000 N/A

Free IoT hubs are limited to one per subscription

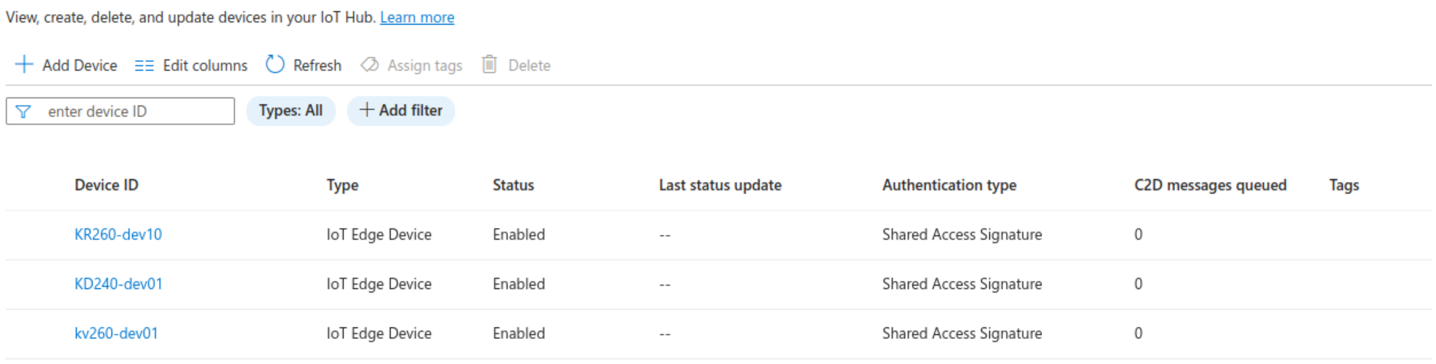
- Click on Review+ Create button to create the Azure IoT Hub.
- Next, create a device where you can actually receive some data from the hardware.

Create an IoT Device

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

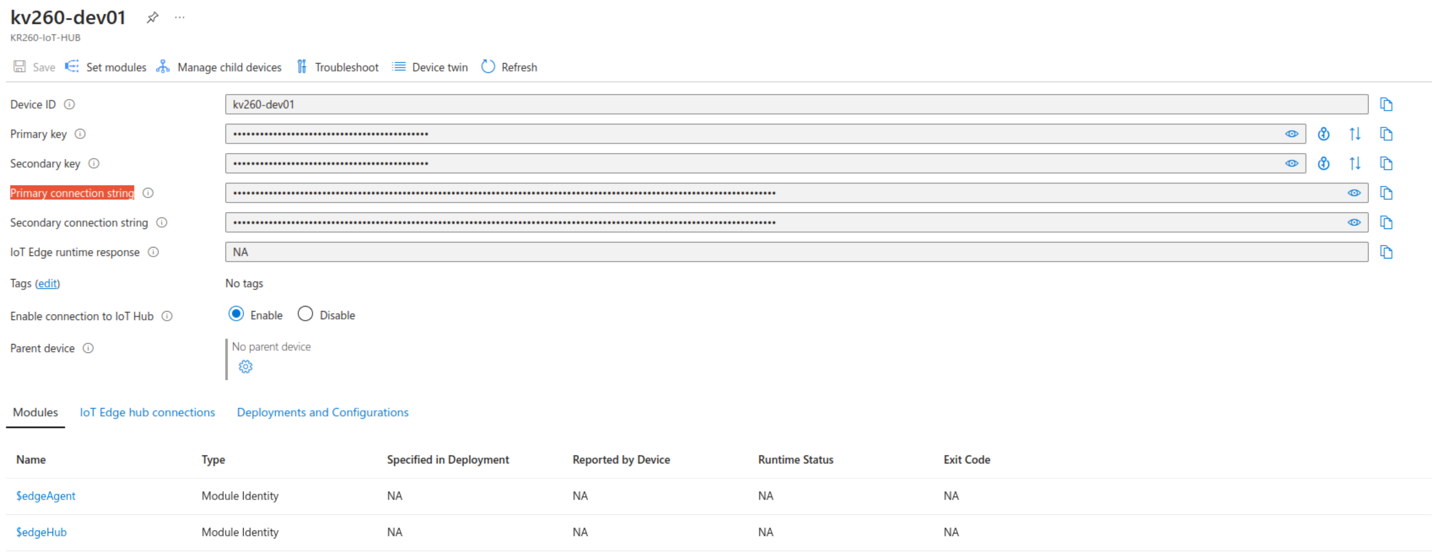


After this device will be available in the IoT hub Device list.



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.



Device ID: kv260-dev01
Primary key: ••
Secondary key: ••
Primary connection string: ••
Secondary connection string: ••
IoT Edge runtime response: NA
Tags (edit): No tags
Enable connection to IoT Hub: ● Enable ○ Disable
Parent device: No parent device

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.

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.

```
import random
import time

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:
```

Then run the simulated application in console:

```
python3 simulated_sensor.py
```

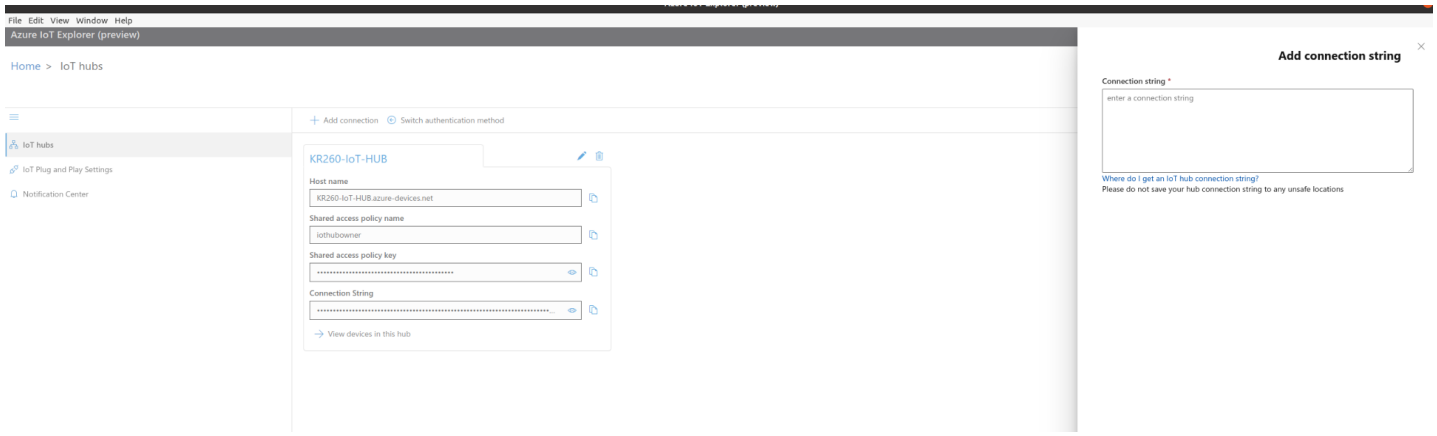
Here is the console log after successful message send to Azure IoT hub.

```
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.

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"
  }
}
```