CK-RX65N

Getting Started Guide with RYZ014A Azure Cellular Application

Introduction

This document describes a system that uses the CK-RX65N Cloud Kit board from Renesas. This system incorporates the CK-RX65N running Azure RTOS and via an Ethernet/Cellular connection visualizes HS3001, ZMOD4410, ZMOD4510, OB1203, ICP10101 and ICM20948 sensors information on Azure IoT Explorer, and controls LEDs on the board.

There are two methods of connectivity for CK-RX65N. One is the Ethernet, second is the Cellular Cat-M1.

This document shows both connectivity methods.

In addition, this document also describes the following:

- How to activate the SIM card contained in the CK-RX65N.
- How to operate and install the certification information for the cloud.
- How to view and run the sensor data on the Azure IoT viewer.

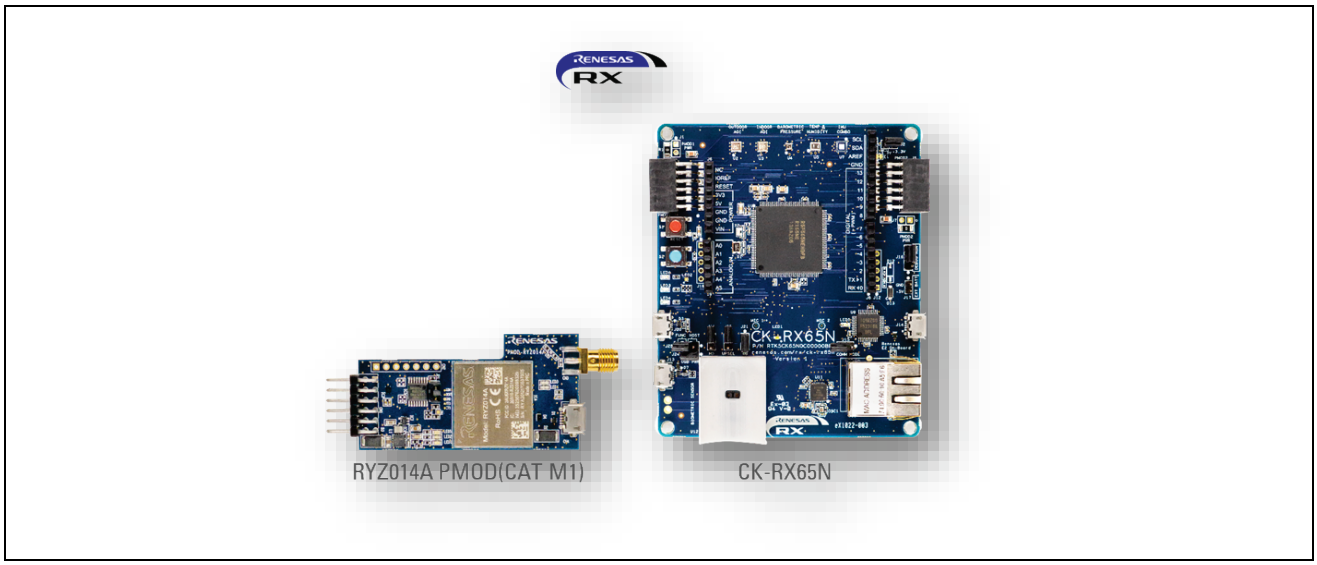

Figure 1. CK-RX65N (with RYZ014A Pmod)

Target Device

CK-RX65N

Program and binary data

- Program
- Binary data

Important Notice

On November 21, 2023, Microsoft announced that they have decided to contribute Azure RTOS to Open Source under the stewardship of the Eclipse foundation and Azure RTOS becomes Eclipse ThreadX.

For detailed information, please refer to the announcement titled at Microsoft Contributes Azure RTOS to Open Source - Microsoft Community Hub.

The support strategy scheme for Eclipse ThreadX will be determined and communicated at a later date.

Microsoft will discontinue the Azure RTOS and Azure RTOS Middleware under the existing agreement LICENSED-HARDWARE.txt.

It is important to note that updates for Azure RTOS on this hardware will no longer be provided.

Contents

1. Terms

Terms used in this document are explained below.

Table 1. Terms

Term	Meaning
IoT	Internet of Things

2. Preparation

2.1 Hardware Configuration

The hardware configuration of the demo project is shown in the following table.

Table 2. Hardware Configuration

Item	Content	Description
CK-RX65N Cloud Kit	Target board for CK-RX65N	Please see detail. https://www.renesas.com/rx/ck-rx65n
RYZ014A Cellular PMOD module	SIM card	This PMOD is contained with CK-RX65N of kit with SIM card
PC	Windows® 10 Azure IoT Explorer (0.14.5.0)	Recommended OS Azure cloud Data viewer

2.2 Software Configuration

The software configuration of the demo project is listed in the table below.

Table 3. Software Configuration

Item	Content	Version
Integrated development environment	e^2 studio	2023-10 or later
Complier	CC-RX	V3.04
Communication Software	Tera term	Version 4.106
Emulator	E2 emulator Lite (on-board)	-
OpenSSL		3.0.5
RTOS	Azure RTOS	V6.1.11

2.3 Tera term Setting

Table 4. Tera Term Setting

Item	Settings
Baud rate	115200
Data length	8
Parity	None
Stop bits	1
Flow Control	none

3. System Diagram

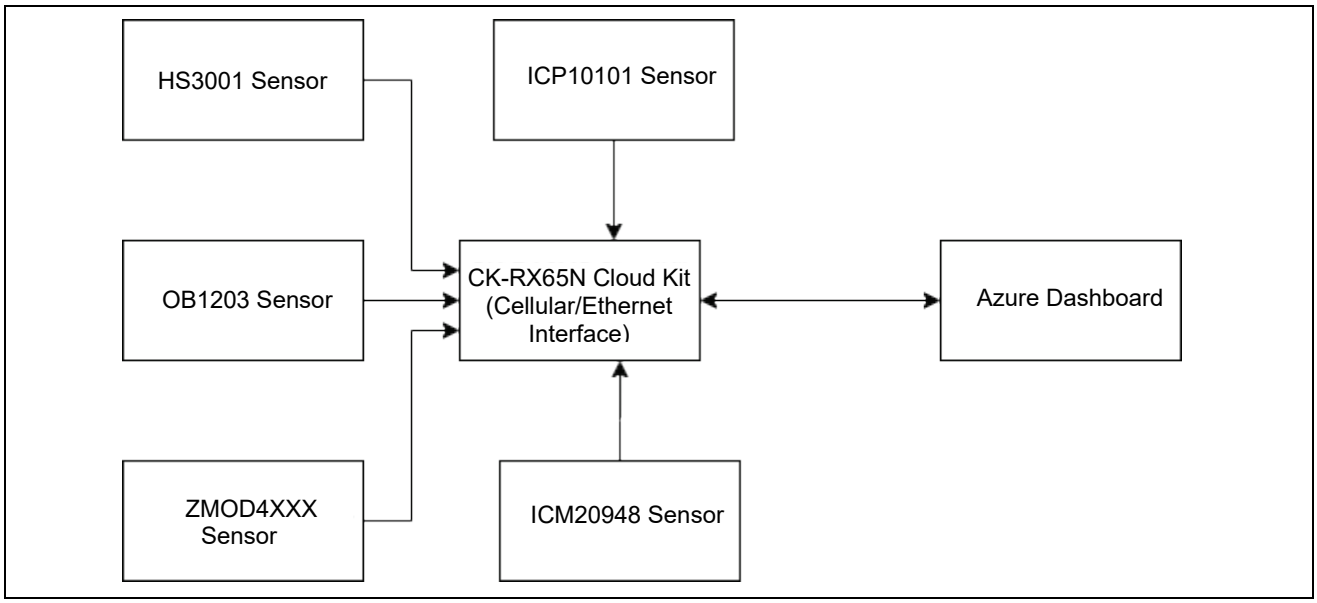

Figure 2. System Diagram

4. Activating a SIM card

One of two SIM cards – a Truphone or MicroAI Launchpad is included in this kit.

Note: The MicroAI SIM card support for CK-RX65N is discontinued. If a MicroAI SIM card was included in the kit, please contact Renesas support to request a replacement to a Truphone SIM card. To identify a MicroAI SIM card, the manufacturer's name is not printed on the SIM card.
Please activate the SIM card as following steps.

To activate the included Truphone SIM card, please visit the Truphone SIM Activation platform at truphone.com/connectit and use the following steps:

1. On the Business page, click **Start activation** button under **IoT SIM Activation.**

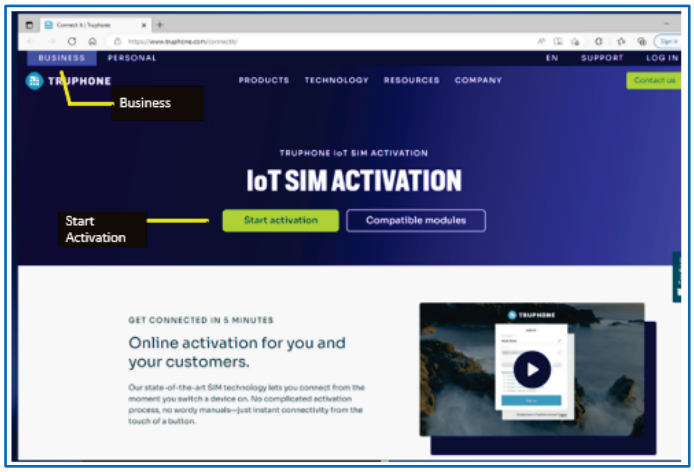

Figure 3. Activating the SIM Card on Truphone

2. Create a new Truphone Account by selecting **Sign up** (next to **Don't have an account yet?**) and fill-in your full name, Email, and a password. Then click **Sign up** to create a new account.

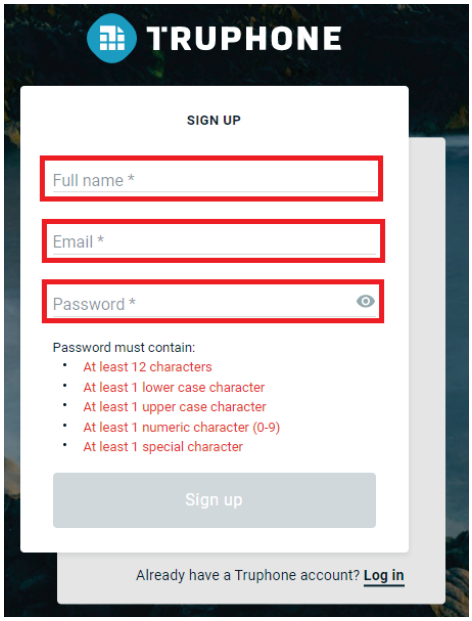

Figure 4. Signing in

3. Select **Personal** as the account type and press **Get Started**.

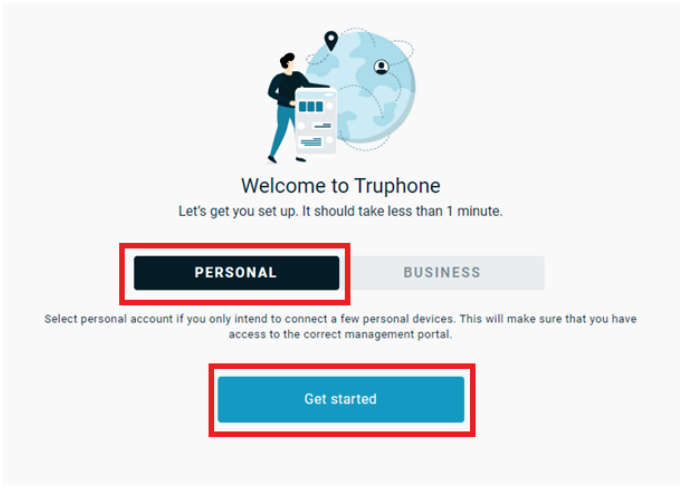

Figure 5. Selecting the Account Type

4. Verify your email by entering the activation code sent to your email account and click **Continue**.

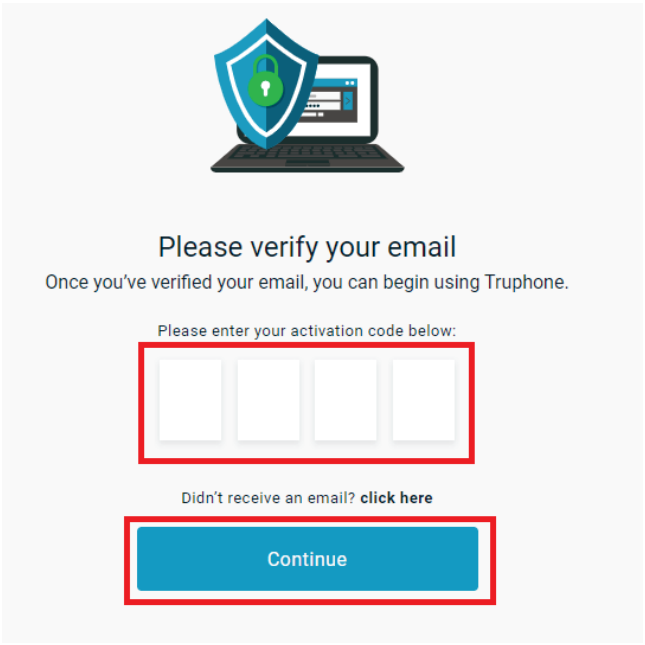

Figure 6. Verifying the email

5. Complete the **Profile information** form – then select **Create account**.

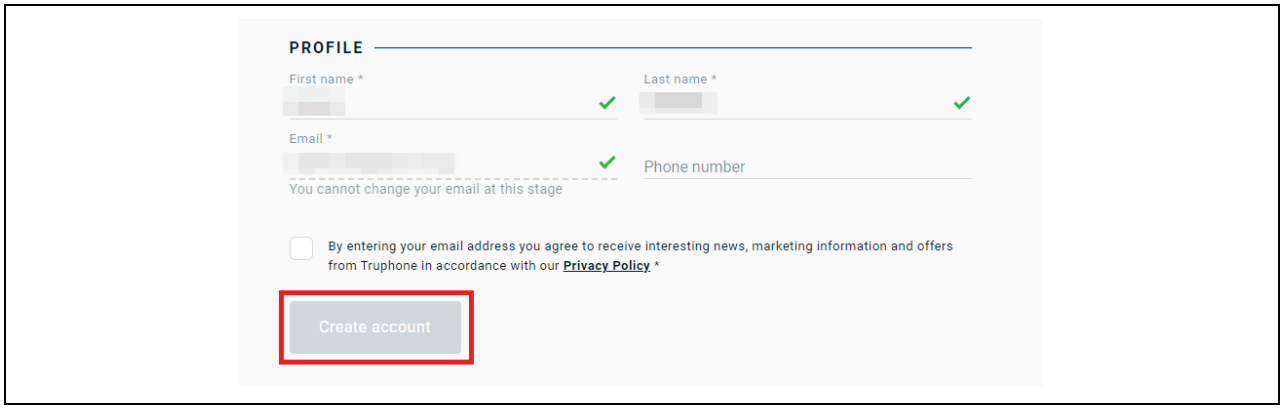

Figure 7. Completing the Profile Information

6. Select **Activate SIMS** to activate your individual SIM by **ICCID**.

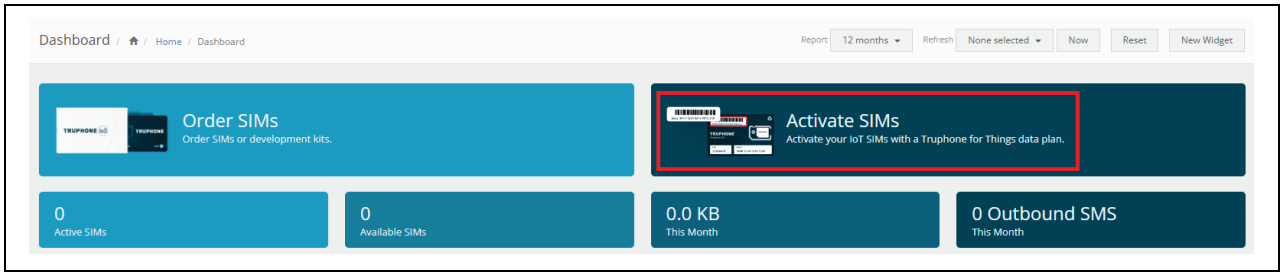

Figure 8. Activating SIM

7. Enter the **ICCID** value.

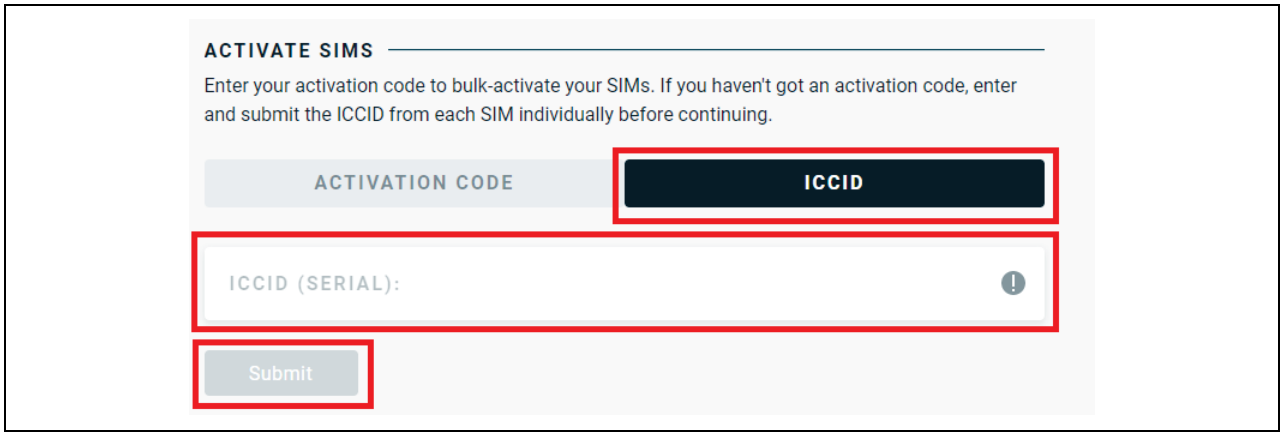

Figure 9. Entering the ICCID

8. You will receive email confirmation when the SIM card activation is complete.
   The **CK-RX65N** kit and SIM card should be activated and can be validated on the Tera Term terminal.
   Note: The SIM card includes free credit for the first 90 days/50MB.
   After the free data charge expires, communication charges will be incurred.

Disclaimer

The activation steps above are provided by SIM Provider Truphone. They are the most current at the time of publishing this application note. If you need help activating your SIM card, contact Truphone support iot.truphone.com or Contact Support | Truphone.

If you have a SIM card from any other provider, then contact the technical support for that provider.

For any other issue that cannot be resolved please contact Renesas Support at Technical Support.

Note: The SIM card provider for the Quick Start Guide example project is Truphone. If you use any other SIM card provider, you must change the Access Point Name required for the SIM card provider in your global region. Failure to do so could result in the RYZ014A not connecting to the cellular network.

5. Azure Account and Credentials Creation

5.1 Install Azure CLI

To prepare Azure cloud resources and connect a device to Azure, you can use Azure CLI. Azure CLI can be installed locally on your PC.

1. Azure CLI can be downloaded from the Microsoft site (https://docs.microsoft.com/en-us/cli/azure/install-azure-cli)
2. The installer name will be similar to azure-cli-2.44.x.msi. or a later version. Click on the installer and the install shield will guide you through the installation process.
3. Install the current release of the Azure CLI. After the installation is complete, you will need to close and reopen any active Windows Command Prompt or PowerShell windows to use the Azure CLI.
4. After the Azure CLI installation is successful, open and launch the Windows PowerShell to use the Azure CLI. A screenshot of the Windows PowerShell is shown below.

Figure 10. Windows Power Shell

5. If you already have Azure CLI installed locally, run *az - -version* to check the version. This application note requires Azure CLI 2.44.0 or later.

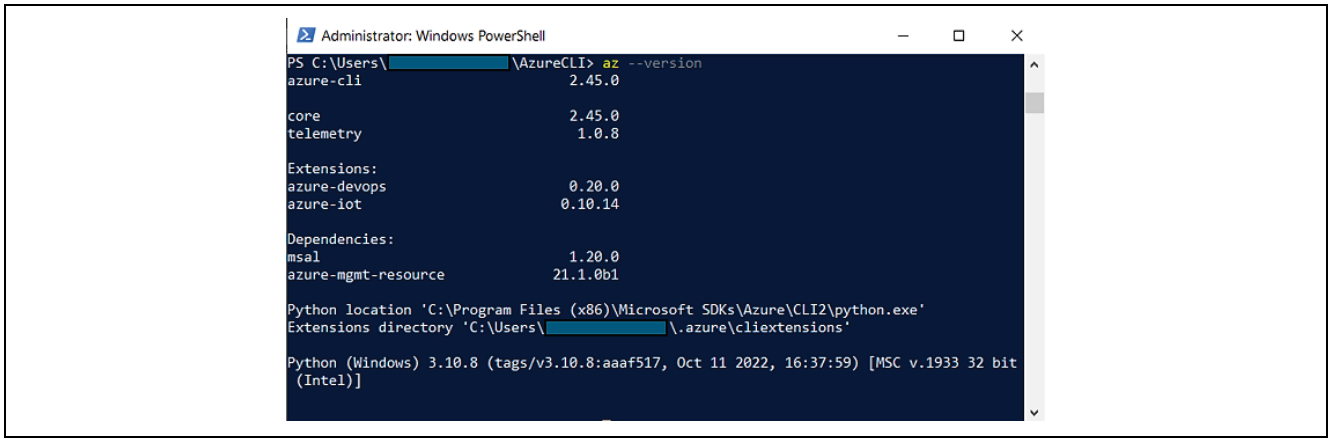

Figure 11. AZURE CLI Version

5.2 Create an IoT Hub

You can use Azure CLI to create an IoT Hub that handles events and messaging for your device.

Note 1: Before you start creating the IoT Hub you are required to have a login to your Azure Portal via web browser. Otherwise, you may notice an error that you are not logged in while creating the IoT Hub at https://portal.azure.com/.

Note 2: If you do not have the Azure Account, you can create one which is valid for 12 months with limited features from the following link https://azure.microsoft.com/en-us/free/

Note 3: Some of the user parameters while creating the IoT Hub needs to be unique. Users are required to take care of this while creating the IoT Hub credentials.

To create an IoT hub use the following steps:

1. In your CLI console, run the `az extension add` command to add the Microsoft Azure IoT Extension for Azure CLI to your CLI shell. The IoT Extension adds IoT Hub, IoT Edge, and IoT Device Provisioning Service (DPS) specific commands to Azure CLI.
   - `az extension add --name azure-iot`

   Note 4: When you run the command for the first time you may not notice output on the console as shownbelow. It just accepts the command.

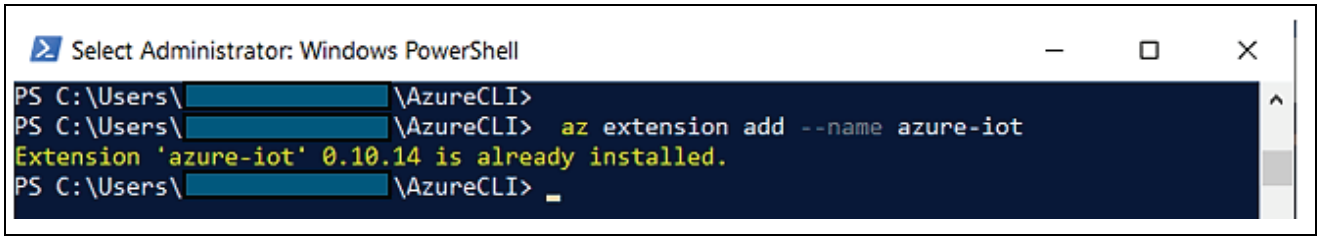

Figure 12. Add Extension for Azure CLI

2. Run the `az login` command to login to the Azure account. Running the `az login` command opens the browser for login. You can enter the login credentials to login to the Azure account. You will notice a similar message on the browser on successful login.

   Note: You can find more information on the Azure CLI at Overview of the Azure CLI | Microsoft Docs

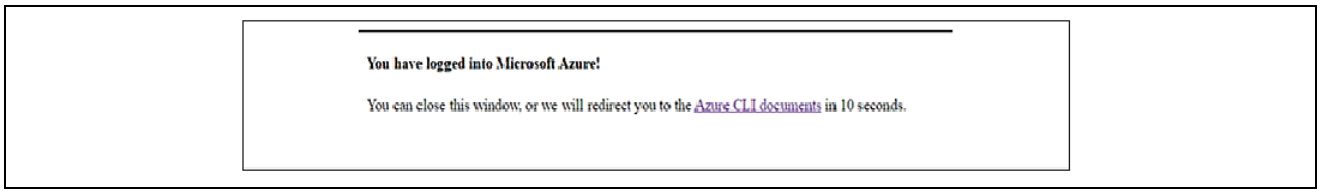

Figure 13. Successful Login to the Azure Account

3. Run the `az group create` command to create a resource group. The following command creates a resource group named `MyRXResourceGroup` in the `westus` region.
4. Note: Optionally, to set an alternate `location`, run `az account list-locations` to see available locations. Then specify the alternate location in the following command in place of `westus`.

   — `az group create --name MyRXResourceGroup --location westus`

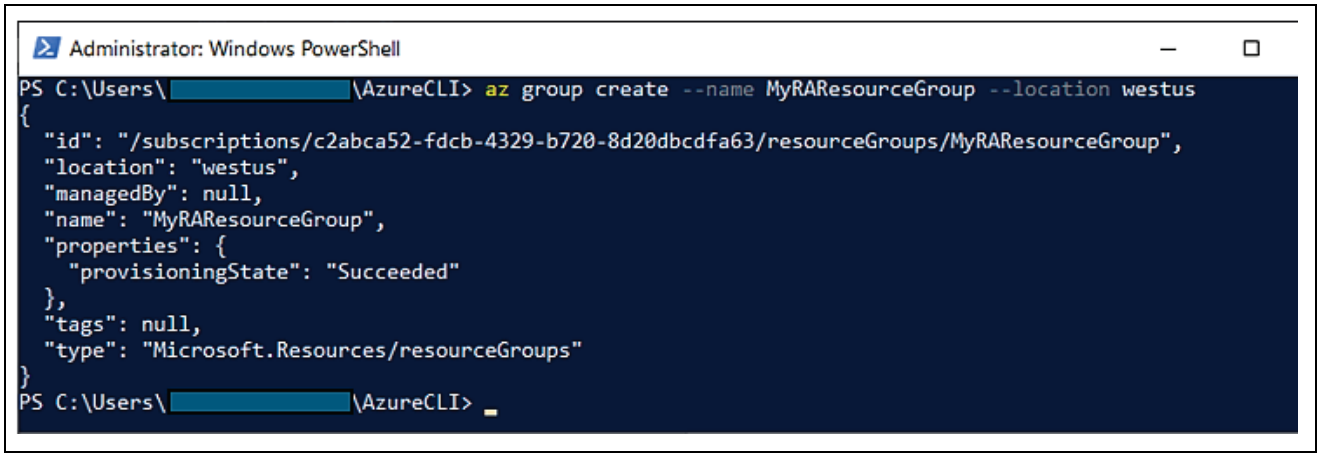

Figure 14. Create Resource Group

5. Run the `az iot hub create` command to create an IoT Hub. It might take a few minutes to create an IoT Hub.
   Replace the *`YourIotHubName`* placeholder below with the name you chose for your IoT Hub. An IoT Hub name must be globally unique in Azure. This placeholder is used in the rest of this tutorial to represent your unique IoT Hub name. Use any command given below.

   — `az iot hub create --resource-group MyRXResourceGroup --name {YourIoTHubName}`

   OR

   — `az iot hub create --resource-group MyRXResourceGroup --name {YourIoTHubName} --location {YourLocation}`

   Note: It may take few minutes to create the IoT Hub. In this case, the IoTHub name used is `RXCLOUDHUB`.

   Note: Microsoft® recommends creating new IoT Hub. If the IoT hub was created previously (2-3 year old) it may not work as desired. So we recommend to create new IoT Hub to run the application to yield the proper results

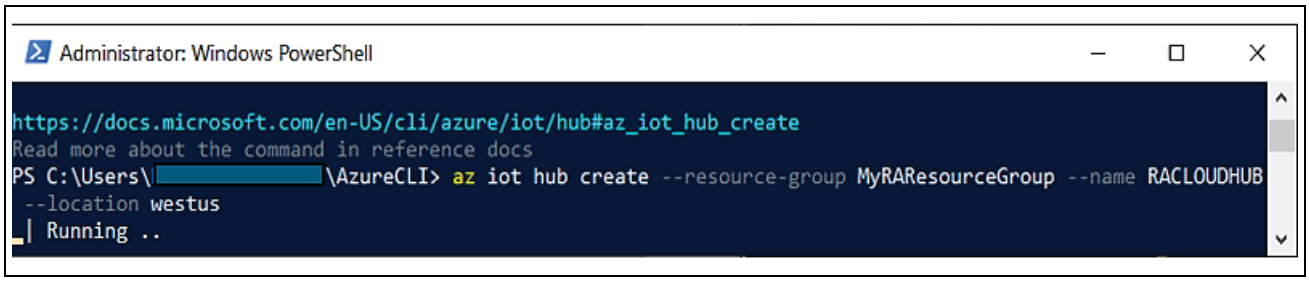

Figure 15. IoT Hub Creation in Progress

6. After the IoT hub is created, view the JSON output in the console, and copy the `hostName` value to a safe place. You use this value in a later step. The `hostName` value looks like the following example:
   — `{Your IoT hub name}.azure-devices.net`

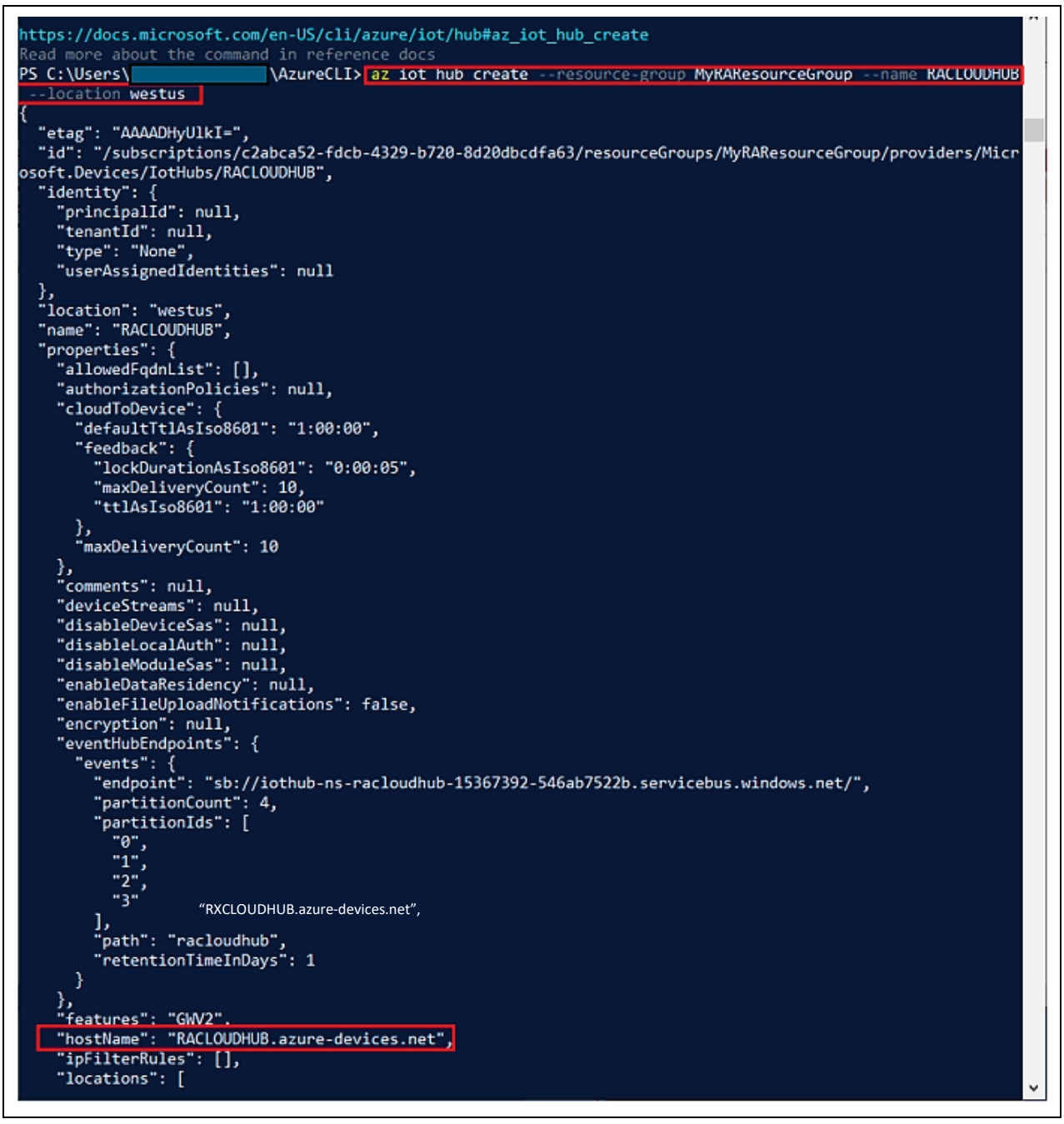

Figure 16. JSON Output after IoT Hub Creation

5.3 Certificate Creation Process

You can use GIT BASH utility for this process. All OpenSSL commands and Self Signed Certificate creation process is given at this link.

Steps areas are follows:

1. Set x509 configuration file for common name in cert.

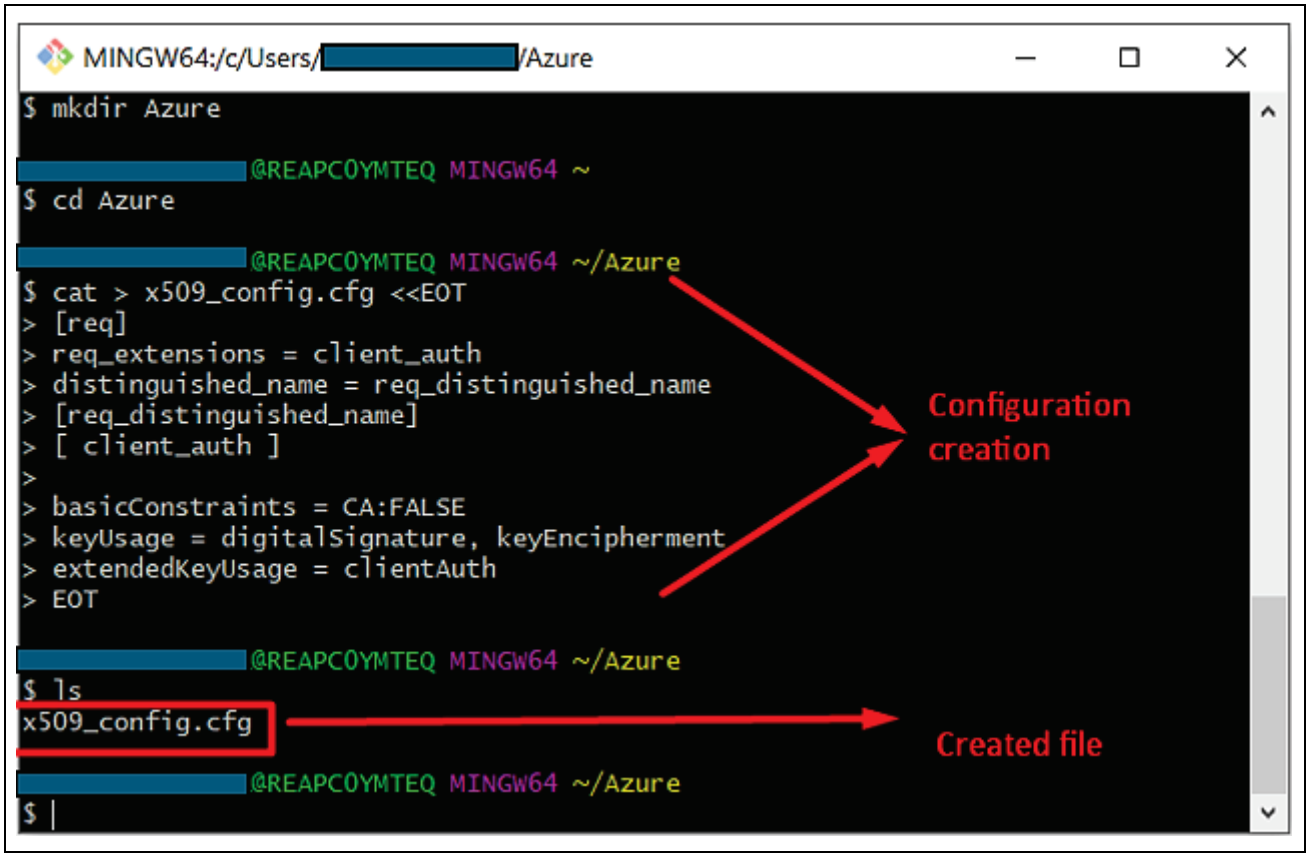

Figure 17. Set X509 Configuration File

2. Create RSA self-signed certificate

   Generate private key and certificate (public key) using the command as shown in the snapshot *"openssl genrsa -out privkey.pem 2048"*

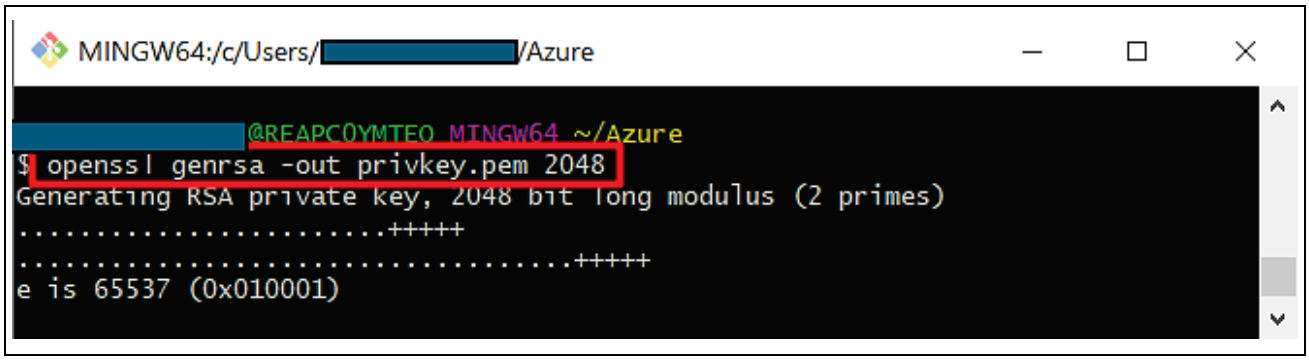

Figure 18. Generate private key and certificate (public key)

3. Embed Device ID in certificate.
   This command will not give you any response if successfully executed.
   openssl req -new -days 365 -nodes -x509 -key privkey.pem -out cert.pem -config x509_config.cfg -subj "//CN=<Same as device Id>"

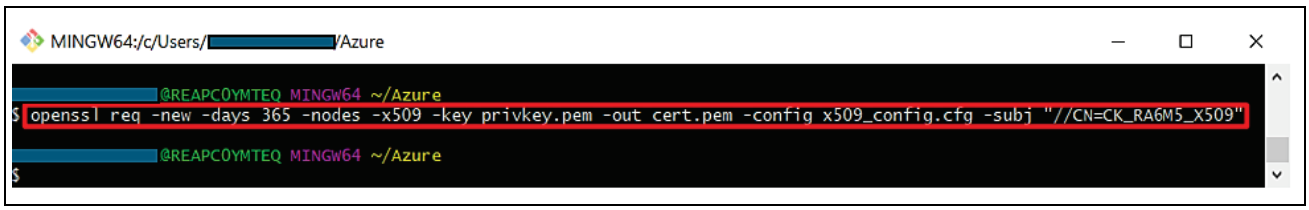

Figure 19. Embed Device ID in Certificate

4. Convert format of key from pem to der "`openssl rsa -outform der -in privkey.pem -out privkey.der`"
   Here you get response "writing RSA key"

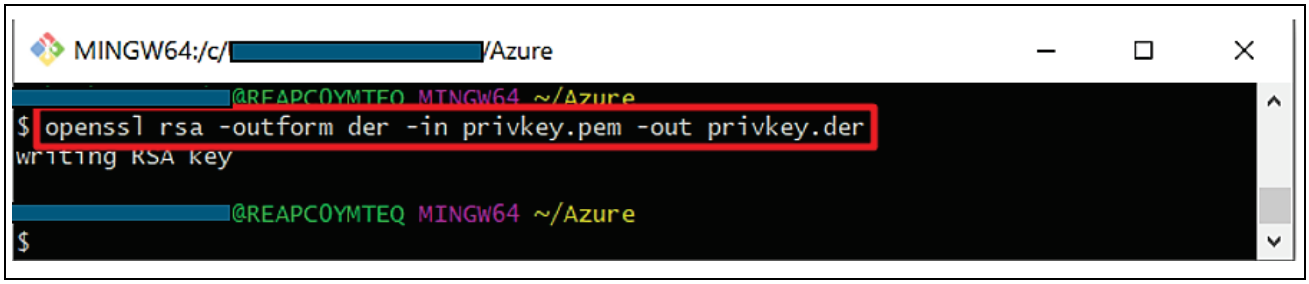

Figure 20. Convert format from key to der

5. Convert format of cert from pem to der "`openssl x509 -outform der -in cert.pem -out cert.der`"
   This command will not give you any response if successfully executed.

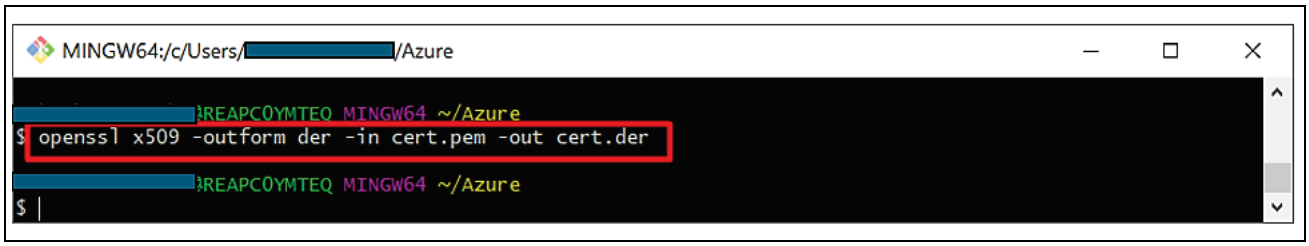

Figure 21. Convert format of cert from pem to der

6. Convert der to hex array (ubuntu) and set them in `sample_device_identity.c`

7. Check whether `sample_device_identity.c` is created.

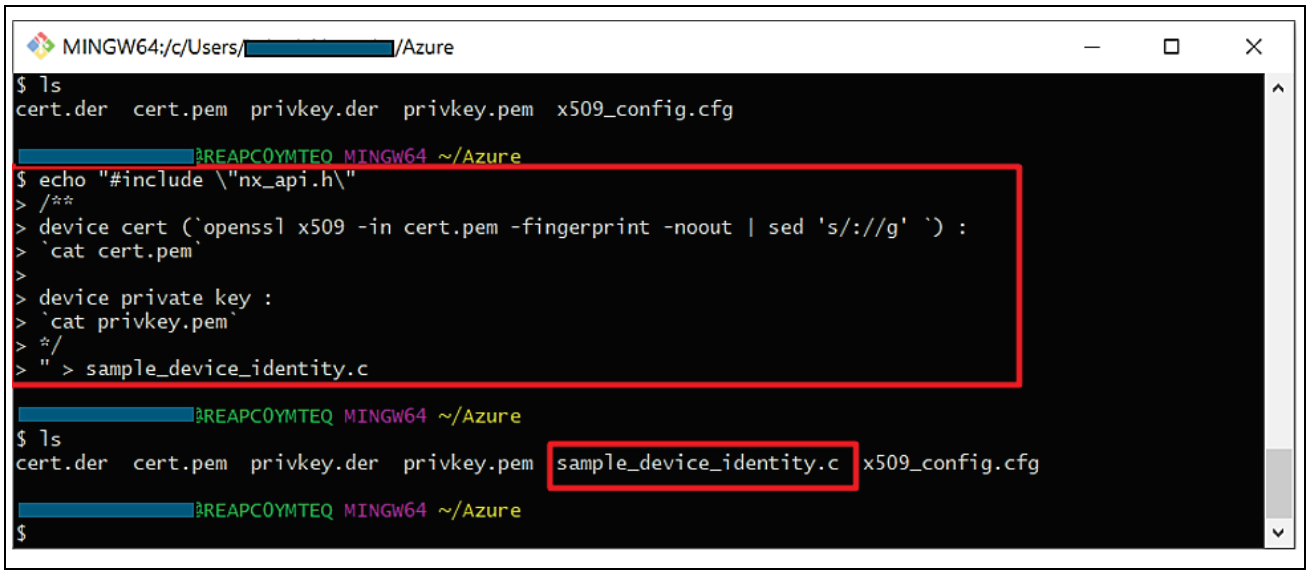

Figure 22. Convert der to hex array and set them in Sample_device_identity.c

8. Producing `sample_device_cert_ptr` and `sample_device_private_key_ptr` array containing device certificate and private key equivalent hex values along with length.

"xxd -i cert.der | sed -E "s/(unsigned char) (\w+)/\1 sample_device_cert_ptr/g; s/(unsigned int) (\w+)_len/\1 sample_device_cert_len/g" >> sample_device_identity.c"

"xxd -i privkey.der | sed -E "s/(unsigned char) (\w+)/\1 sample_device_private_key_ptr/g; s/(unsigned int) (\w+)_len/\1 sample_device_private_key_len/g" >> sample_device_identity.c"

These commands will not give you any response if successfully executed.

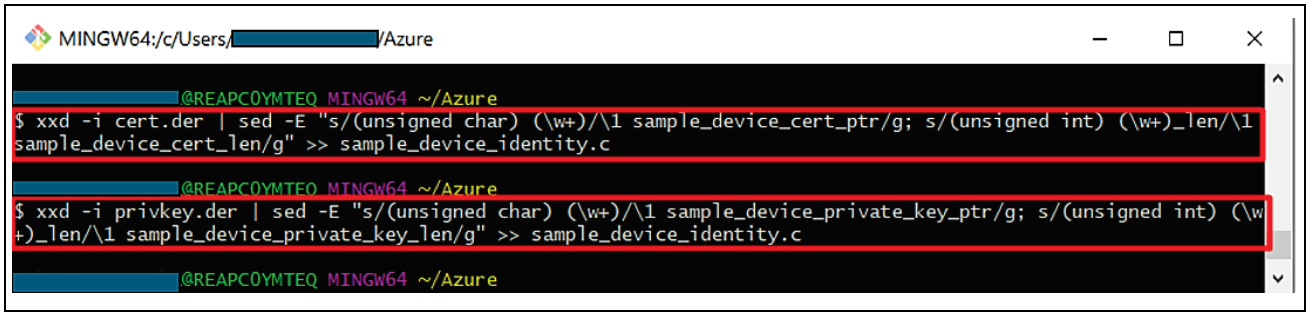

Figure 23. Producing arrays containing hex values

Check the content of `sample_device_identity.c` with cat command. In this file you will get Device certificate along with SHA1 fingerprint, Device Private Key, sample_device_cert_ptr and sample_device_private_key_ptr array along with their length. This fingerprint you have to use in device creation process.

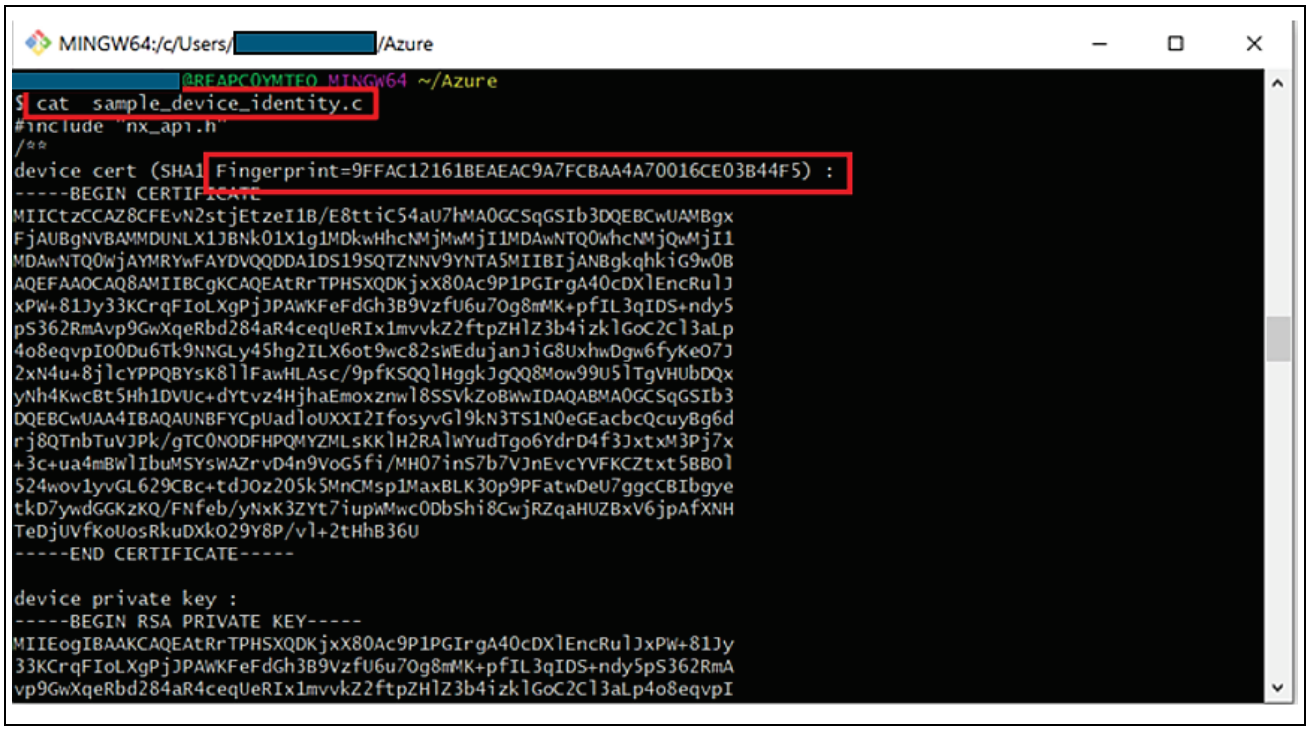

Figure 24. Check the content of sample_device_identity.c

5.4 View Device Properties

You can use the Azure IoT Explorer (https://docs.microsoft.com/en-us/azure/iot-pnp/howto-use-iot-explorer) to view and manage the properties of your devices. In the following steps, you'll add a connection to your IoT Hub in IoT Explorer. With the connection, you can view properties for devices associated with the IoT Hub.

Download and install latest (above v0.15.6.0) Azure IoT Explorer from: https://github.com/Azure/azure-iot-explorer/releases

Note: Click and install the downloaded msi file `Azure.IoT.Explorer.Preview.0.15.6.msi` or newer version of the downloaded file. The install shield guides you through the installation process.

5.5 Set IoT Hub

To add a connection to your IoT Hub:

1. In your Azure CLI console, run the az iot hub `show-connection-string` command to get the connection string for your IoT hub.
   - *"az iot hub connection-string show -n {YourIoTHubName}"*

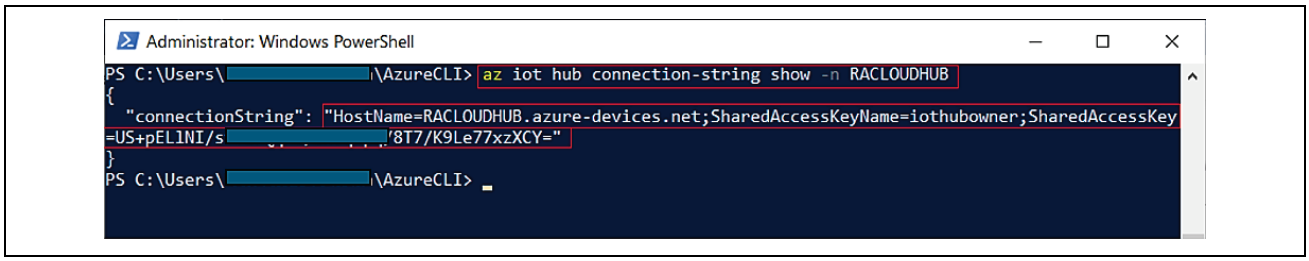

Figure 25. Connection String

2. Copy the connection string.
3. Open the Azure IoT Explorer and select **IoT hubs > Add connection**.
4. Paste the connection string into the **Connection string** box.
5. Select **Save**.

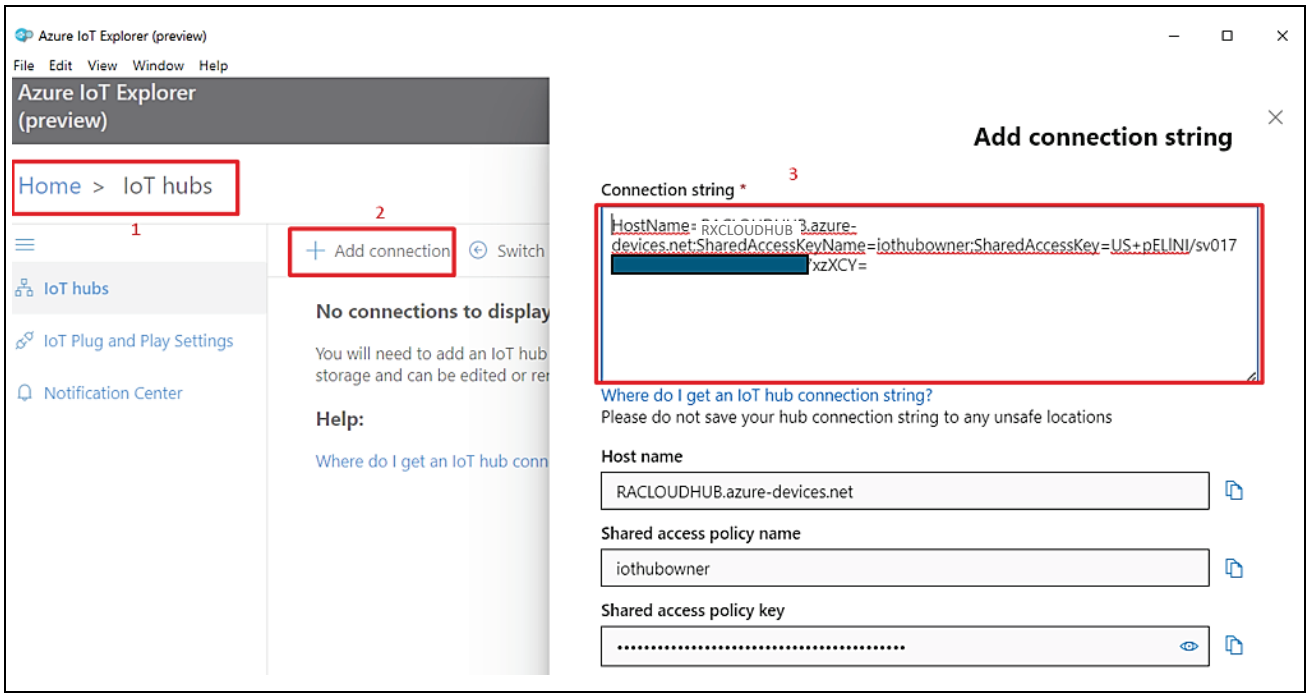

Figure 26. Adding Connection String

Note: In some cases, Azure IoT Explorer may report an error that the default port that IoT Explorer is trying to use is being used by another application. In order to overcome this error, you can add a different port number for the Azure IoT Explorer as shown below.

Go to your PC, edit the system environmental variables as shown in the screenshots shown below.

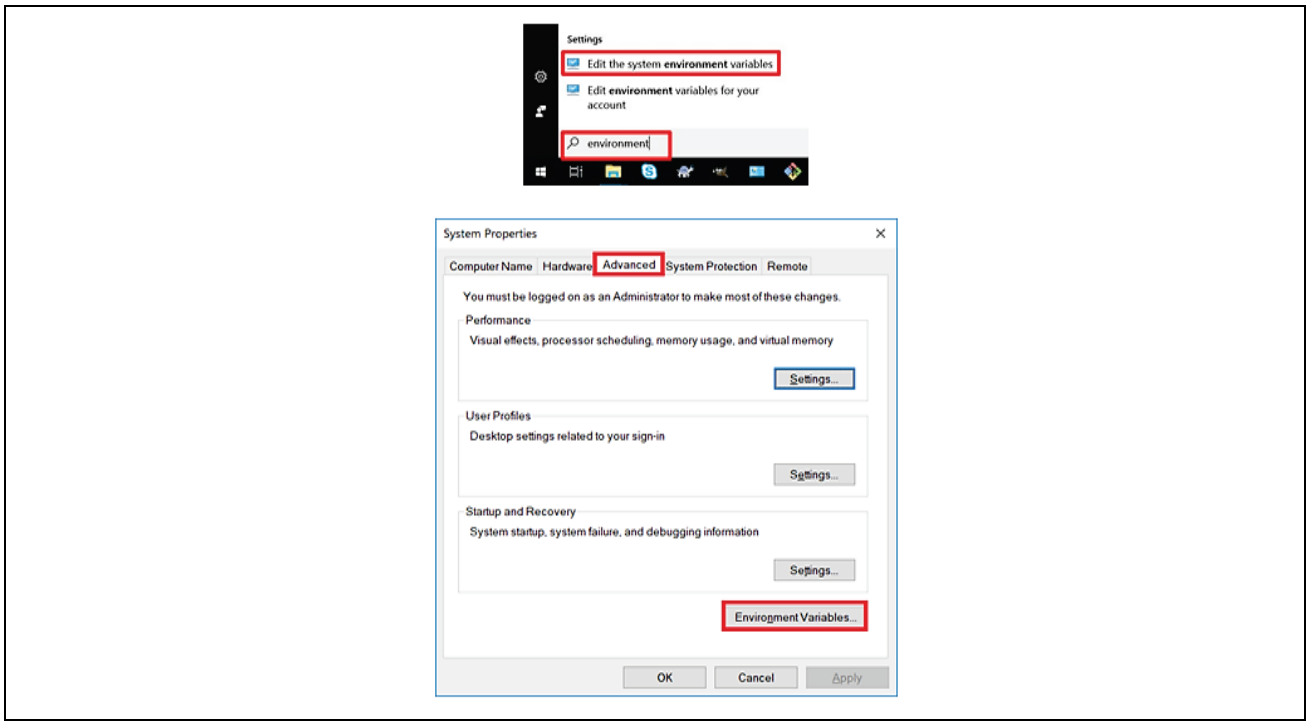

Figure 27. Editing System Environment Variable

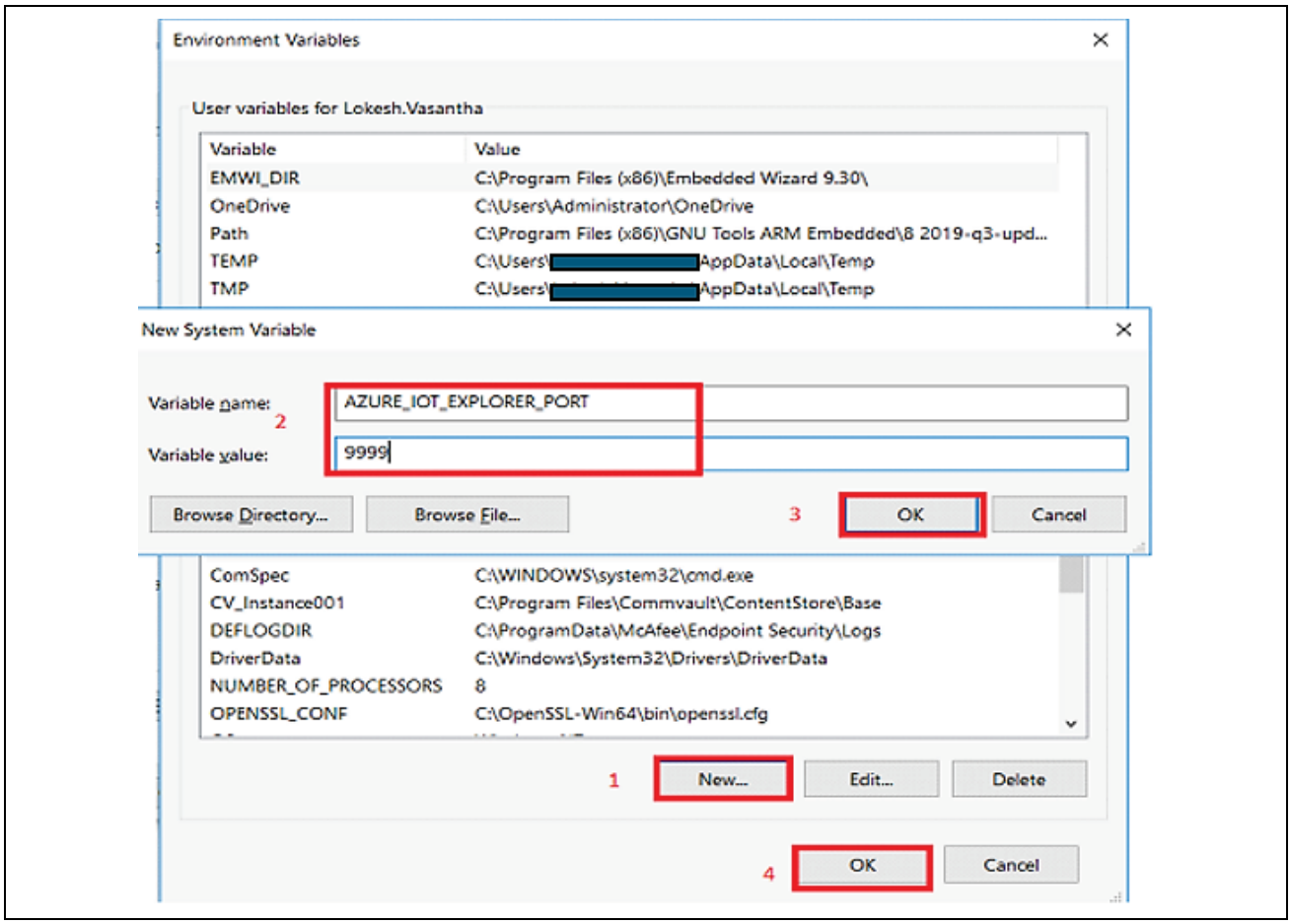

Figure 28. Adding System Environment Variable for Alternate Port - Azure IoT Explorer

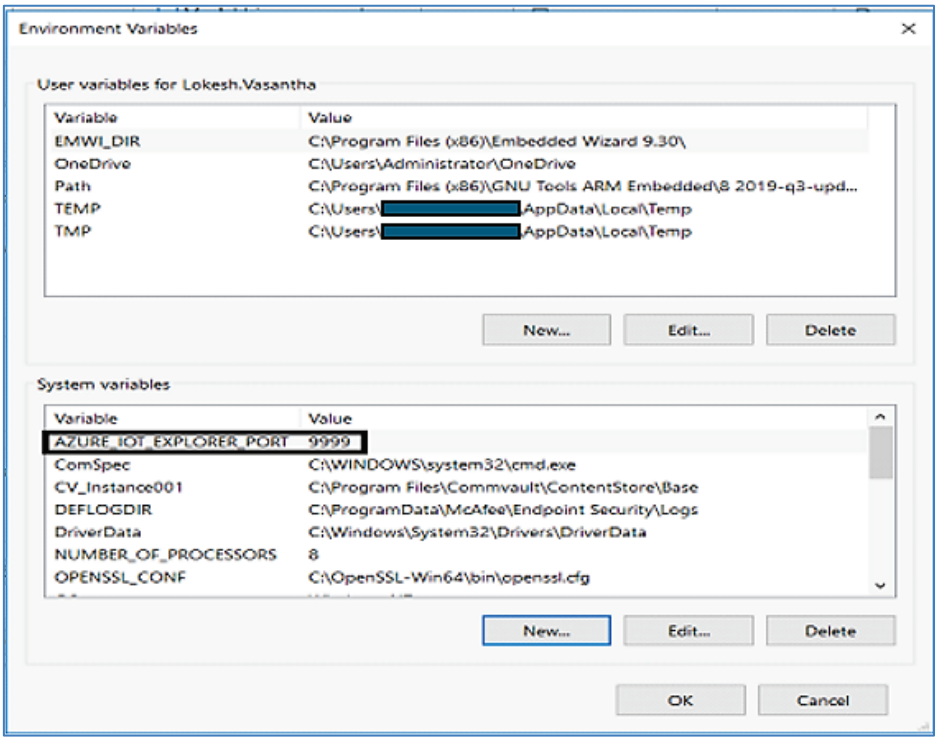

Figure 29. Added Alternate Port for Azure IoT Explorer

If the connection succeeds, the Azure IoT Explorer switches to a **Devices** view and lists your device.

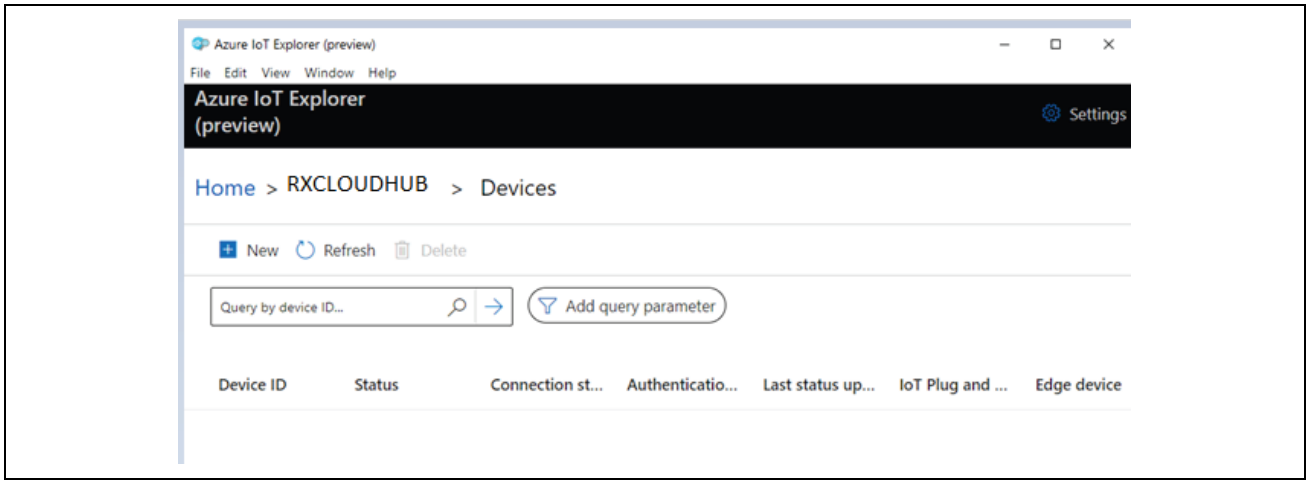

Figure 30. **Listed Devices**

5.6 Register an IoT Hub Device

In this section, you create a new device instance and register it with the IoT Hub you created. You will use the connection information for the newly registered device to securely connect your physical device in a later section.

To register a device:

1. You can Create Device with help of Azure IoT Explorer as below.
   Click on **New**.

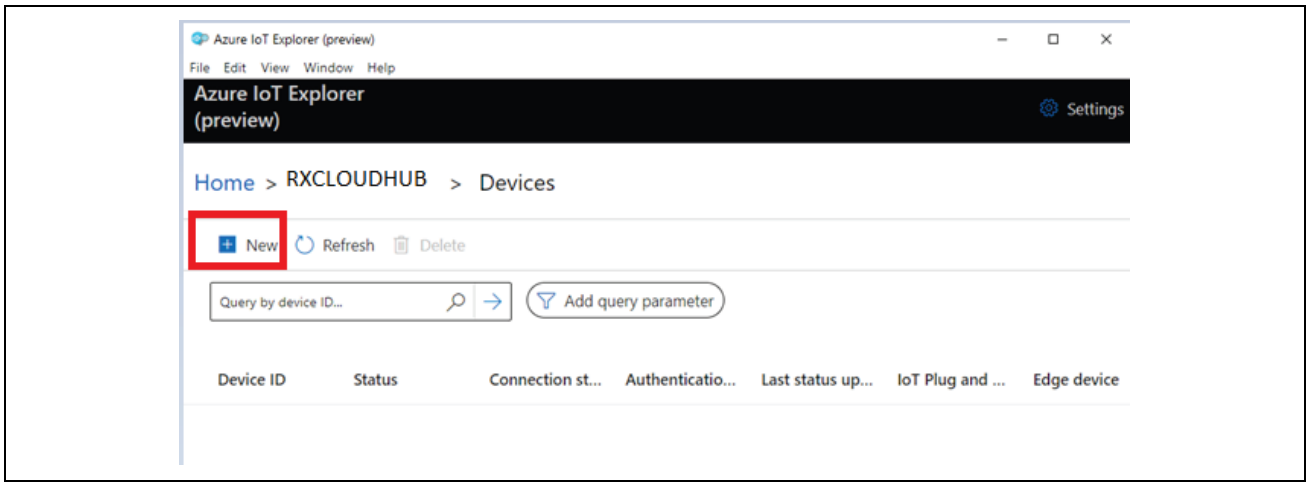

Figure 31. New Device Creation process with Azure IOT Explorer

2. In this stage, you have to give Device ID, Authentication type, Primary thumbprint, Secondary thumbprint then click on **Create**. Use fingerprint generated in previous section (**Figure 24. Check the content of sample_device_identity.c**) for the primary and secondary thumbprints.

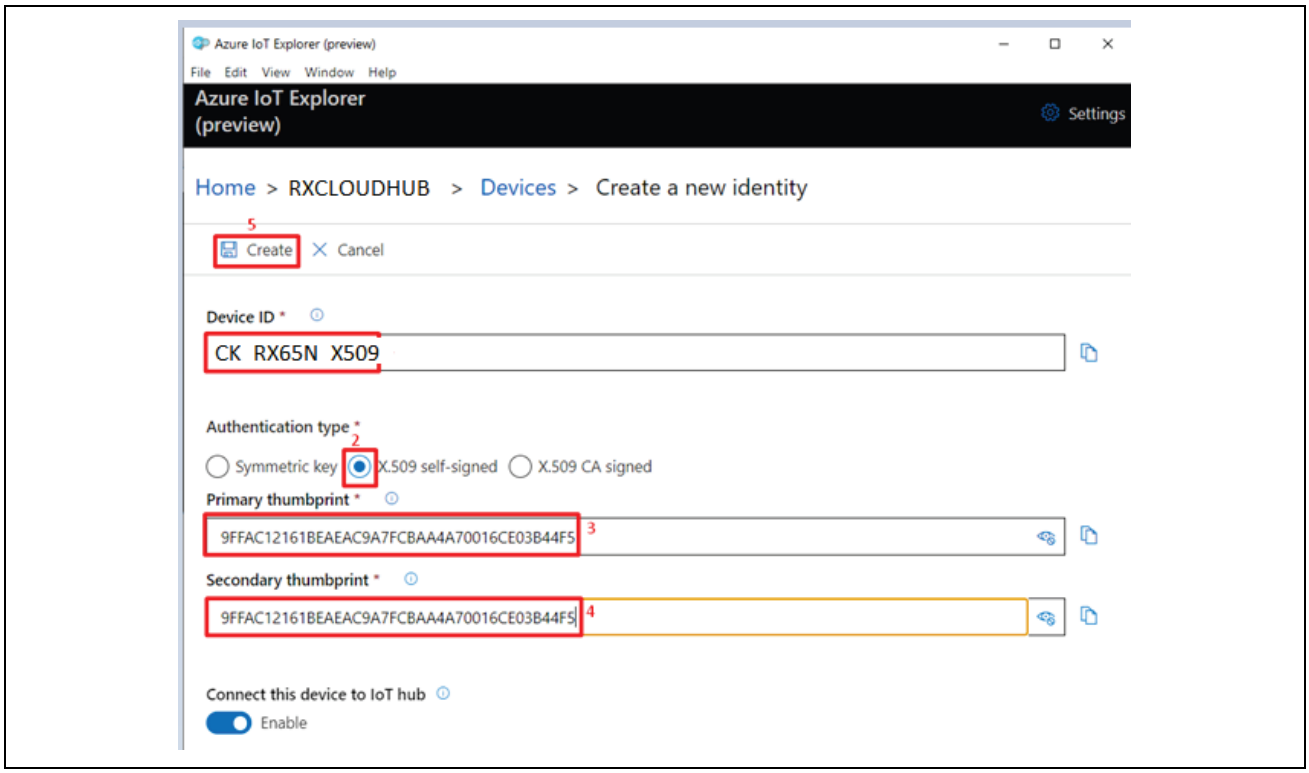

Figure 32. Naming, Authentication type and Thumbprints

3. You can see your created device in Devices section of Azure IoT Explorer.

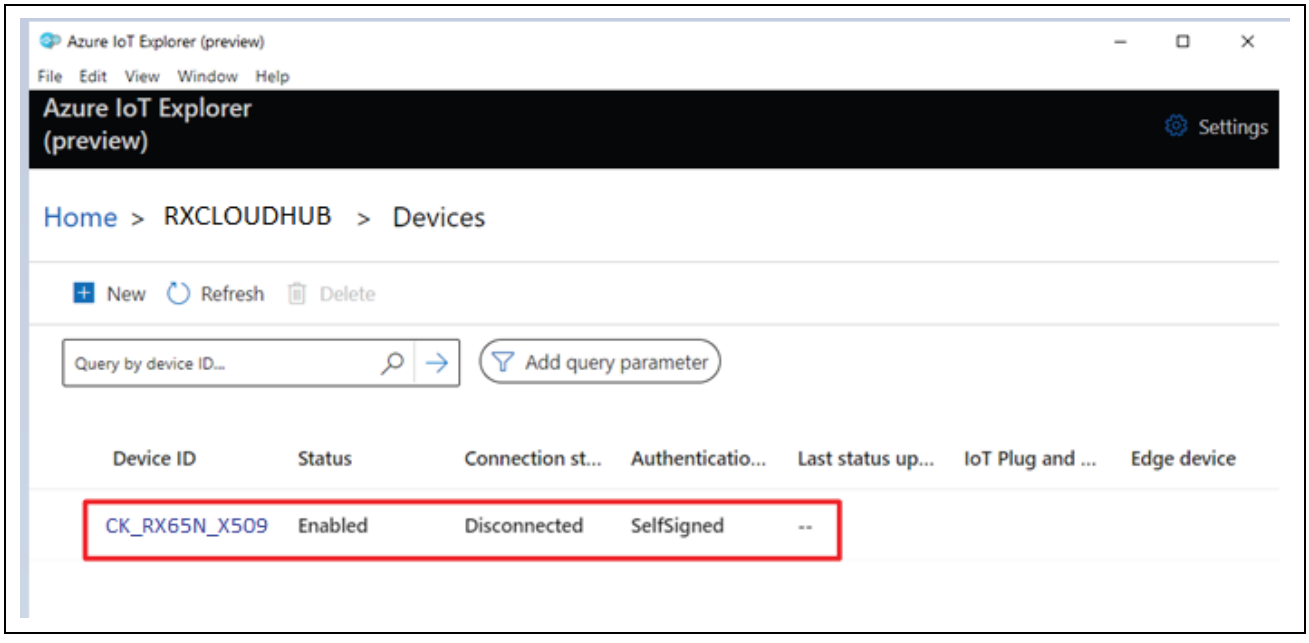

Figure 33. Newly Created Device

5.7 Prepare the Device

To connect the device to Azure, modify a configuration file for Azure IoT settings (of your Device ID and Hostname), build and flash the image to the device.

Add Configuration

1. Import the application project into an empty e² studio. Open `sample_config.h` and make the changes to the configuration as shown in the snapshot with your host name, device Id and USE_DEVICE_CERTIFICATE.

Constant name	Value
HOST_NAME	{Your IoT hub hostName value}
DEVICE_ID	{Your deviceID value}
USE_DEVICE_CERTIFICATE	1

6. Common Users: To Import the Project and Run

6.1 Import the Project

Use the following steps to prepare the software for the demo program:

1. Extract the project files from the archive and copy them to the C drive.
   Please unzip it the project file to **the short path of your PC**.
   If the path is deep, a build error may occur due to the file path length issue.
2. Launch e² studio and specify a workspace directory and click **Launch**.

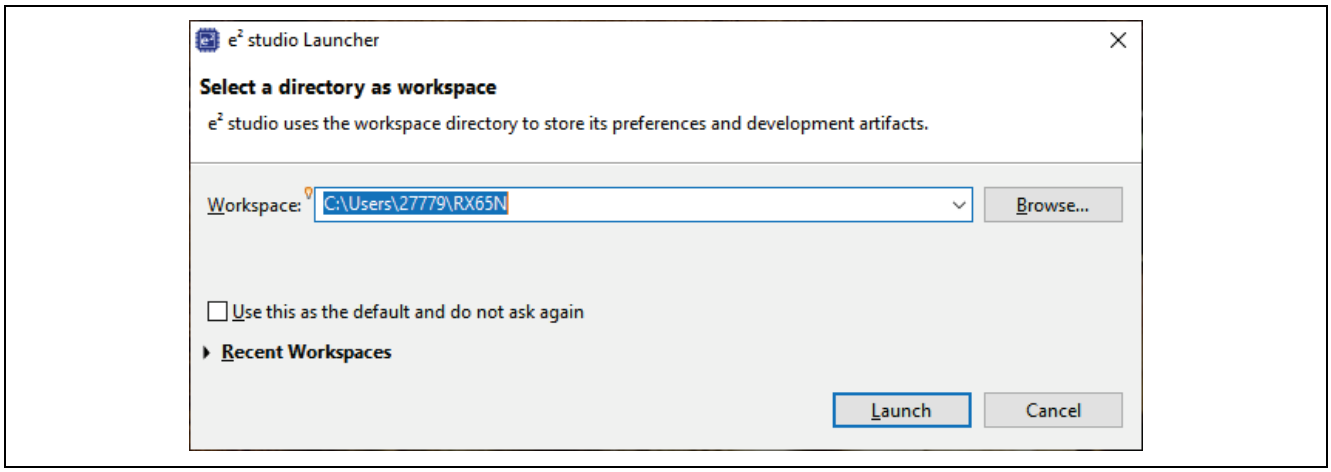

Figure 34. Launch e² studio

3. Select **File** > **Import**….

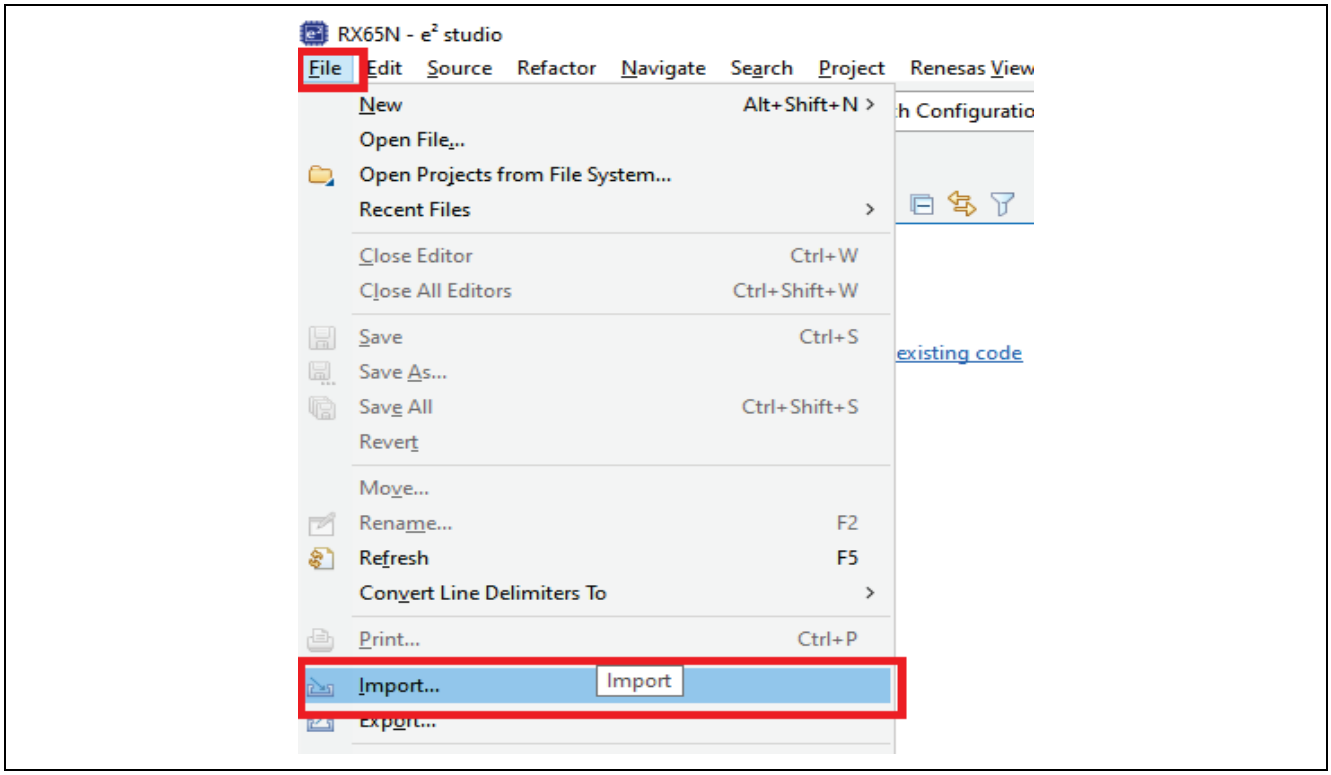

Figure 35. Select Import

4. Click **General** > **Existing Projects into Workspace** > **Next**.

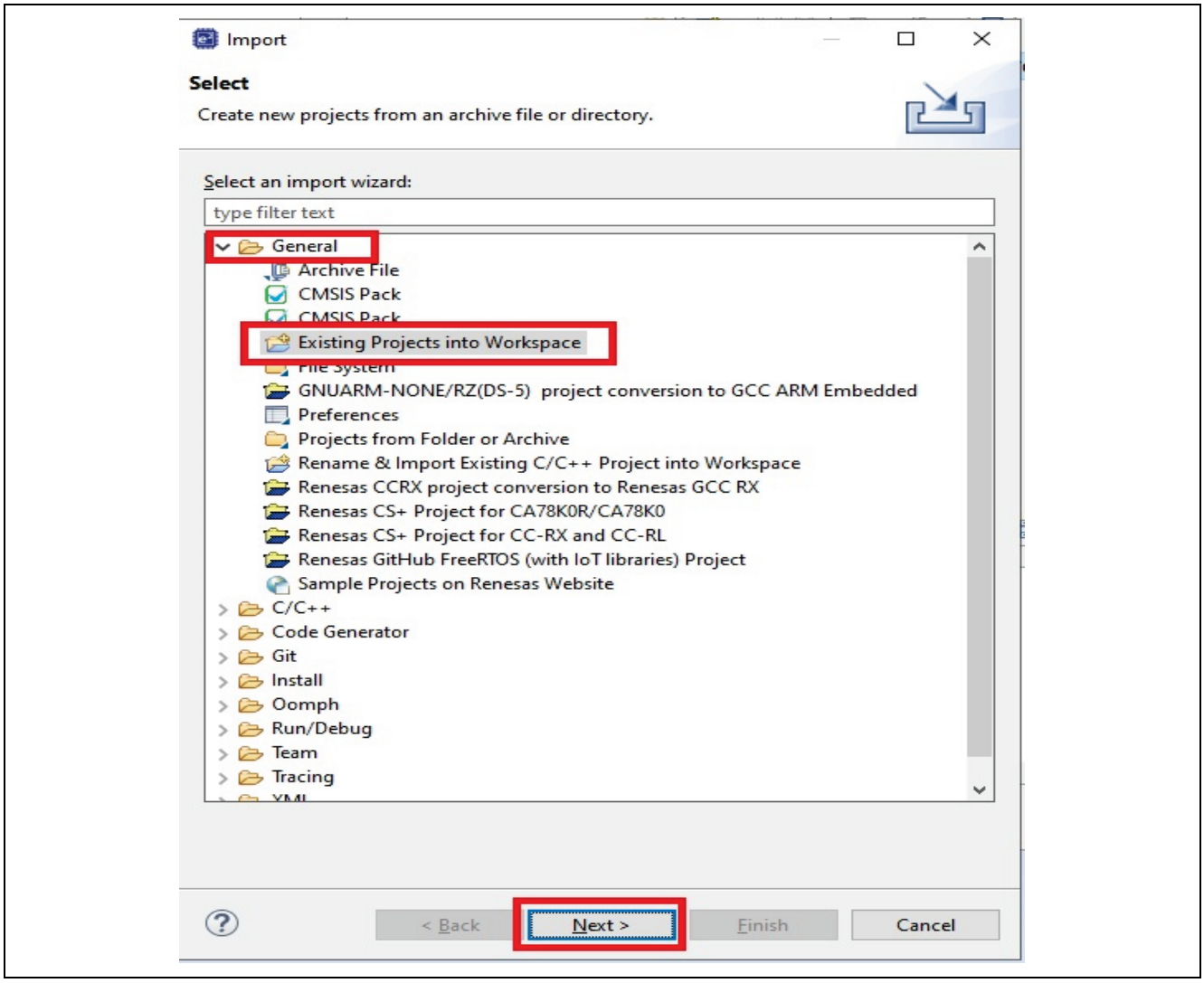

Figure 36. Select Existing Projects into Workspace

5. Click **Browse…**, then specify the root directory as follows.

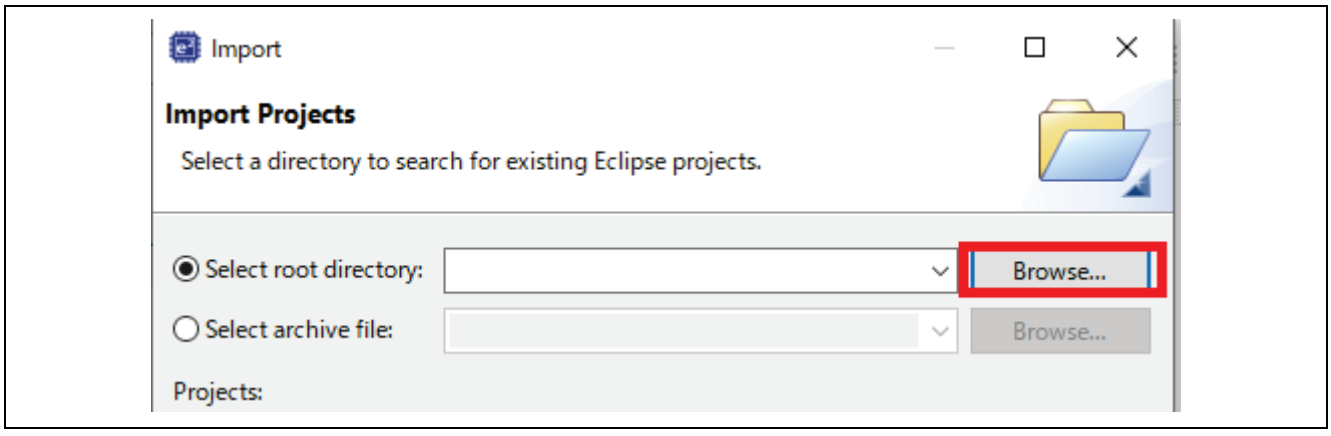

Figure 37. Find the Project

You can choose two types of connectivity and compilers when importing the project.
Please go to **"[Project Root folder]\projects\renesas\"** folder.

Table 5. Project Details

Project Name	Compiler	Connectivity
rx65n-new-ck in the "ck_rx65n_azure_iot_pnp" zip	CC-RX	Ether
rx65n-new-ck-cellular in the "ewf_fork" zip		Cellular
rx65n-new-ck-gcc (**Planning**)	GCC	Ether
rx65n-new-ck-cellular-gcc (**Planning**)		Cellular

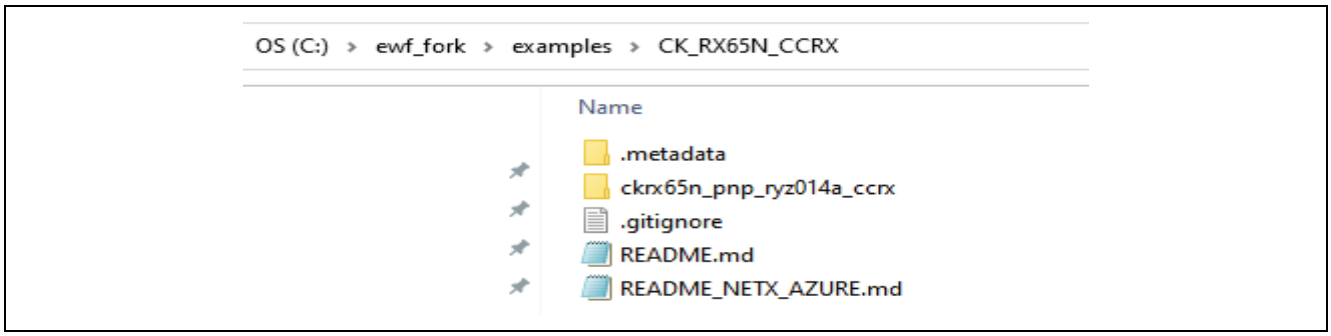

Figure 38. Project Files

This example explains the procedure when selecting "**ckrx65n_pnp_ryz014a_ccrx**" as the cellular project.
Open the "**C:\ewf_fork\examples\CK_RX65N_CCRX\ckrx65n_pnp_ryz014a_ccrx**" folder.
If you use other project, please open "**\ckrx65n_pnp_ryz014a_ccrx**" folder of your selected project.

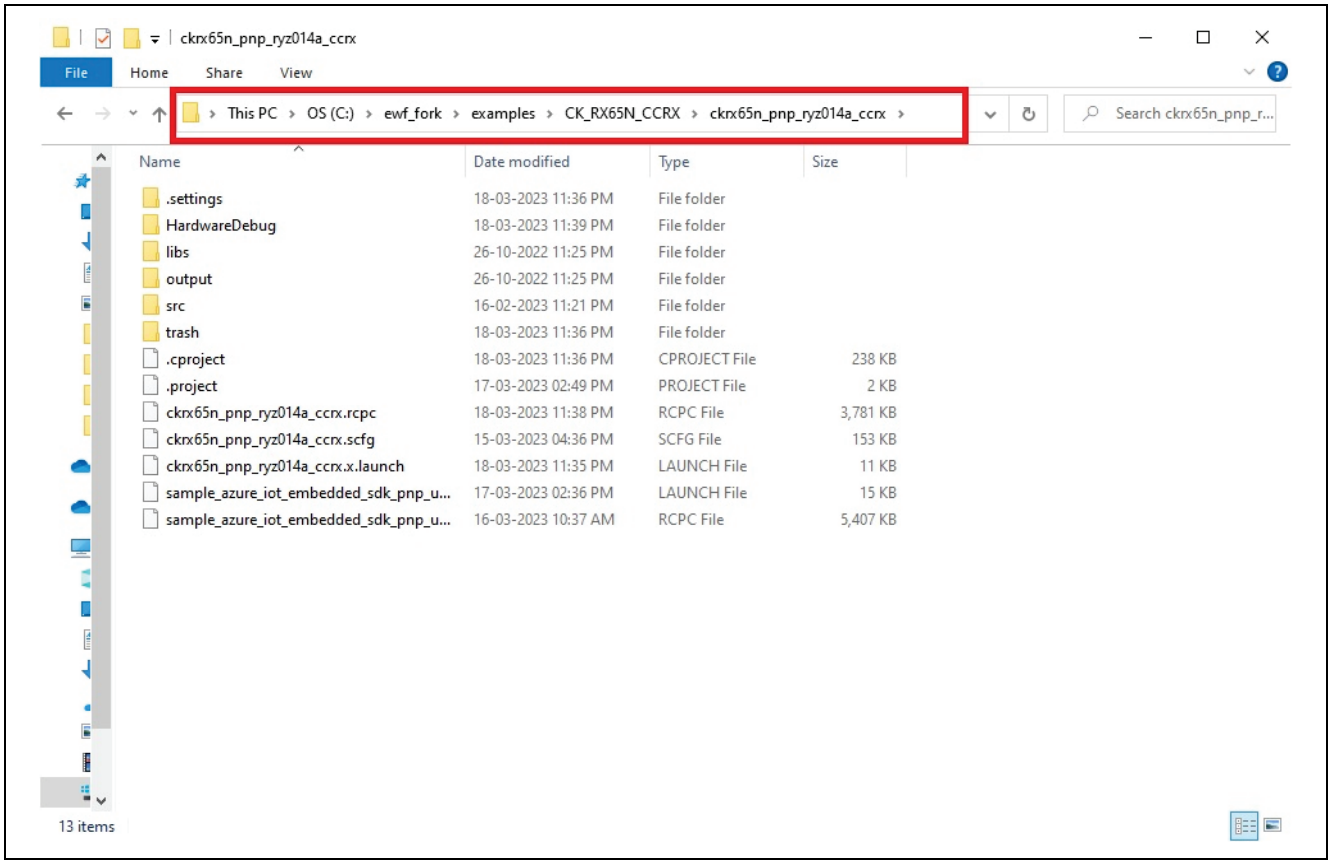

Figure 39. Select the Project Folder

6. Finally, click **Finish**.
   Note: Make sure that the **Copy projects into workspace** option is unchecked.

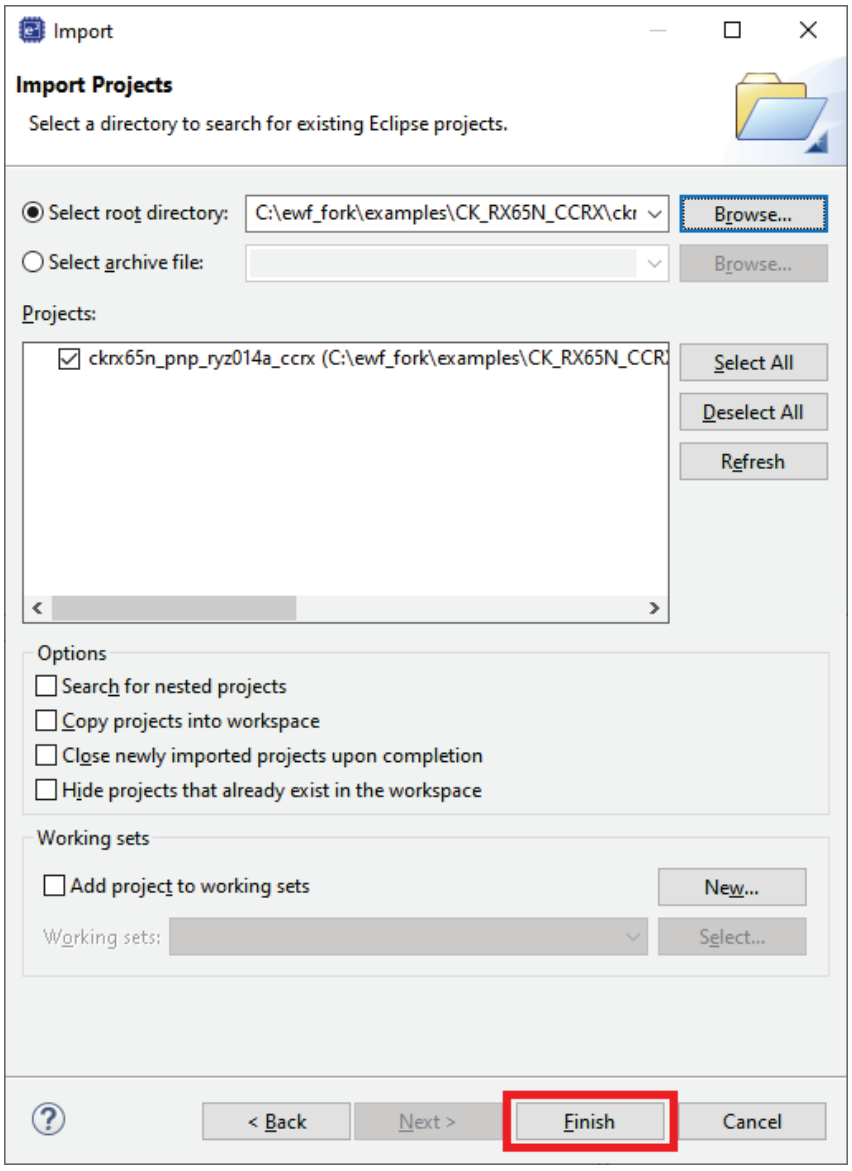

Figure 40. Finish the import the project

7. Check and set the SIM card information.

   Doble click the “ckrx65n_pnp_ryz014a_ccrx.sfg” to open the smart configurator.

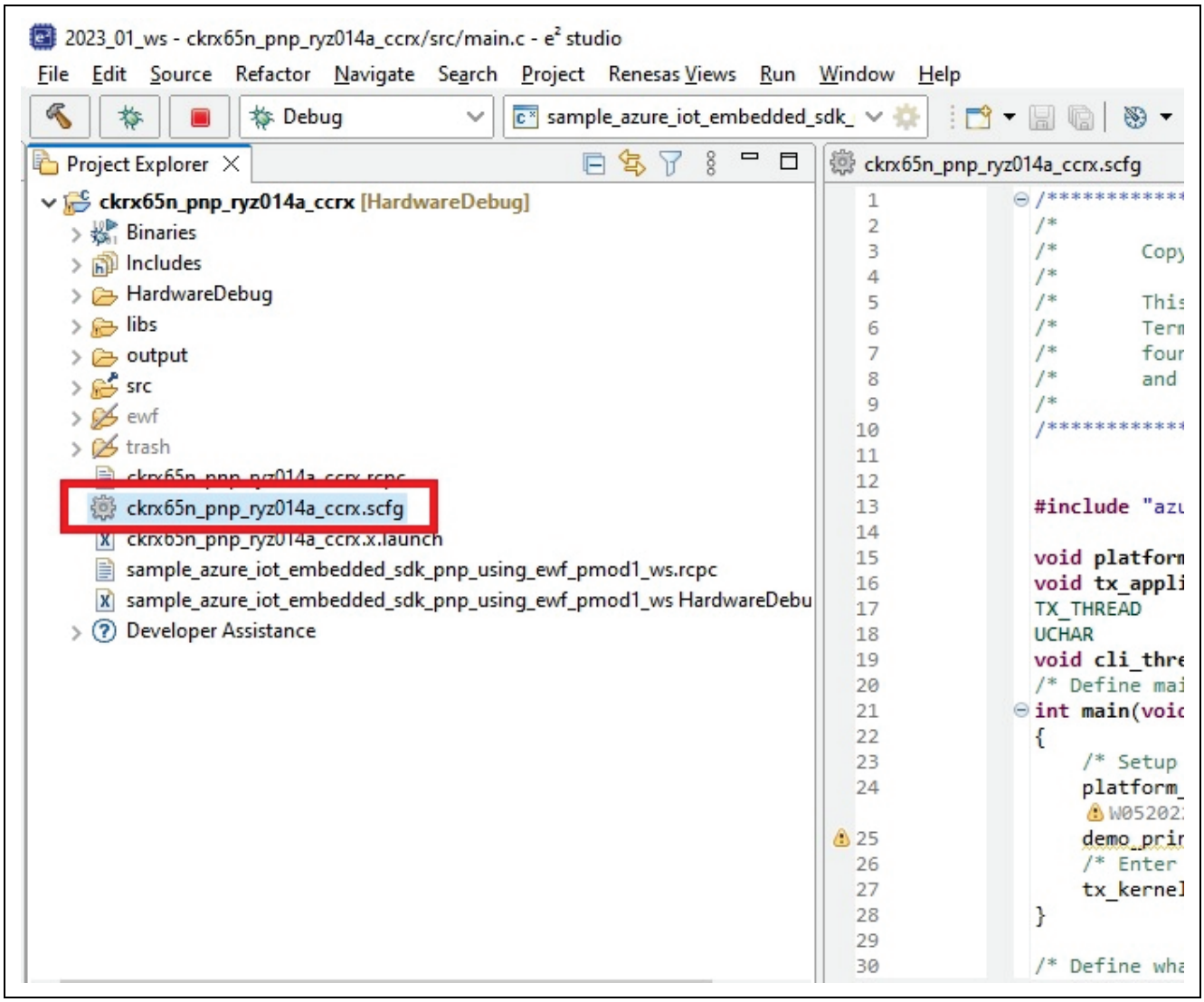

Figure 41. Open the Smart Configurator

8. Execute code generation. If you have changed the Smart Configurator settings, click **Generate Code**.

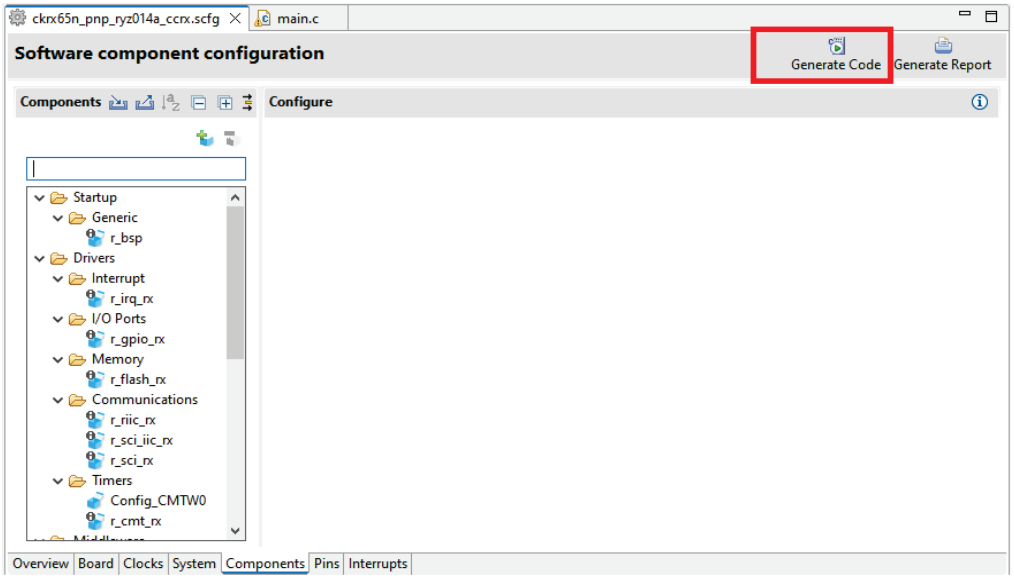

Figure 42. Generate Code

9. Select **Project** > **Build Project** and confirm that 0 errors are reported.

Note: Make sure to clean the project before building it for the first time. If a demo build error occurs after the initial build, clean the project again and then build it.

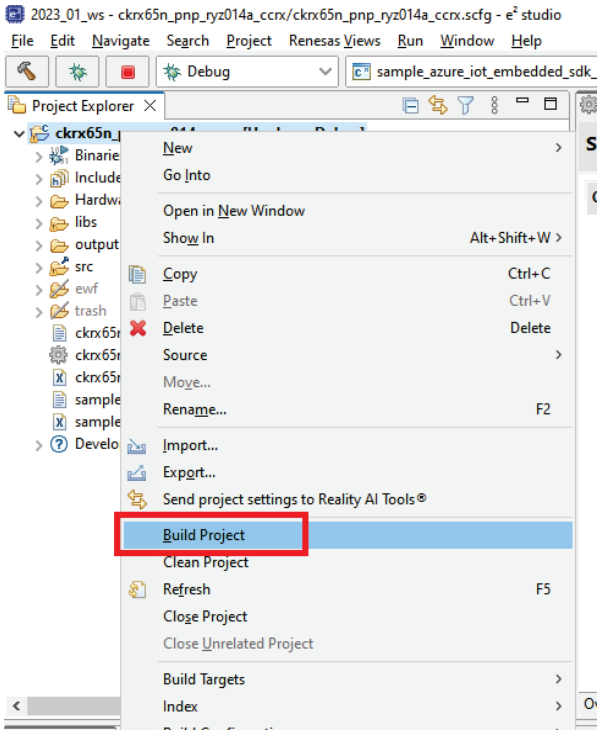

Figure 43. Build the Project

2. Debug Configuration.
   Follow the graphics shown in this step.

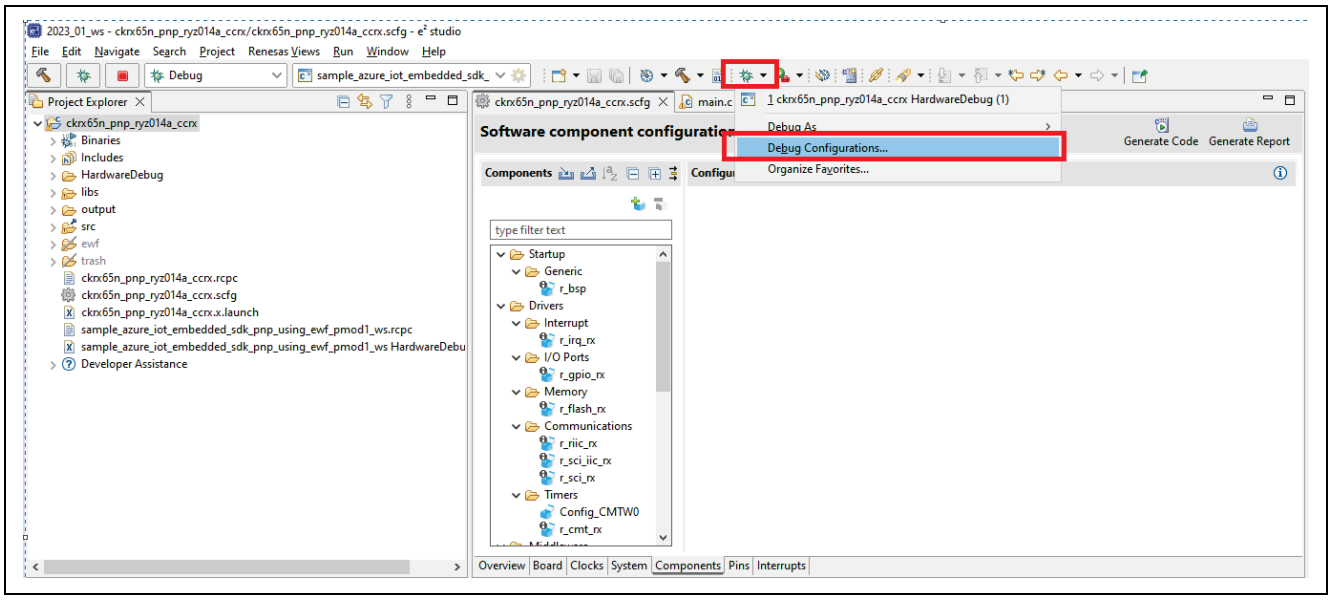

Figure 44. Configuration debugger 1/2

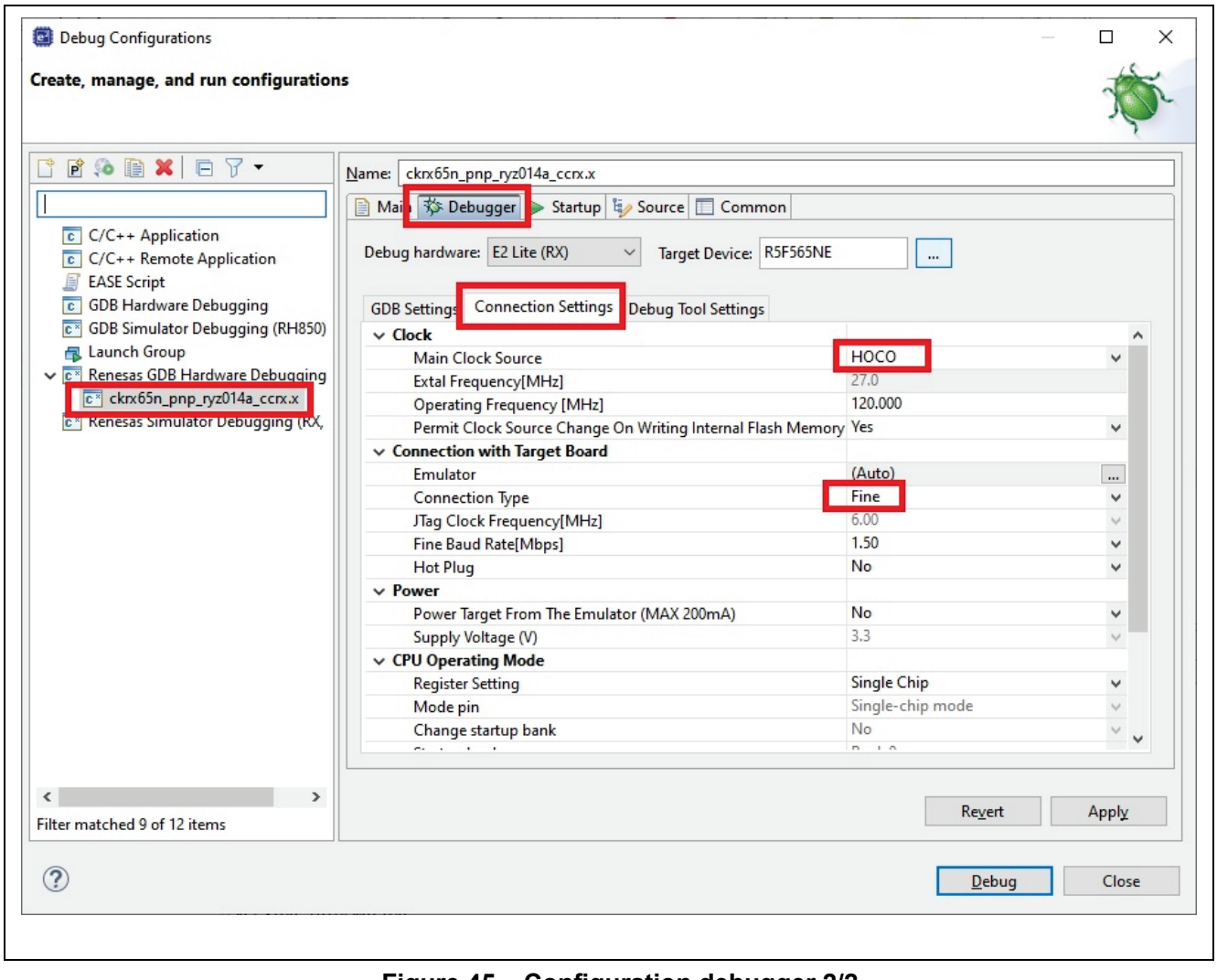

Figure 45. Configuration debugger 2/2

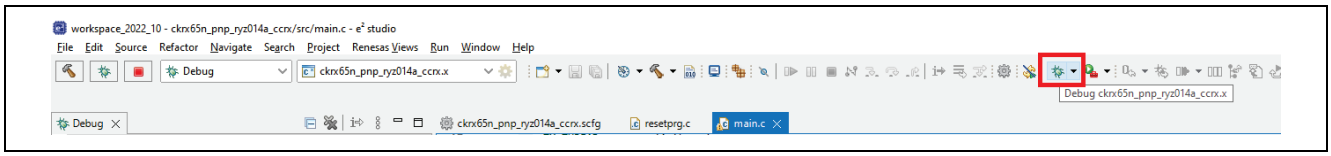

Figure 46. Start Debug

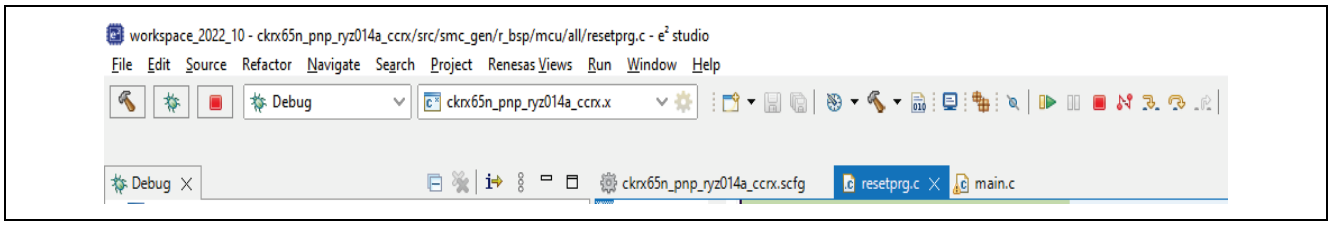

Figure 47. Run

6.2 Serial Terminal Settings

1. Open Device Manager.

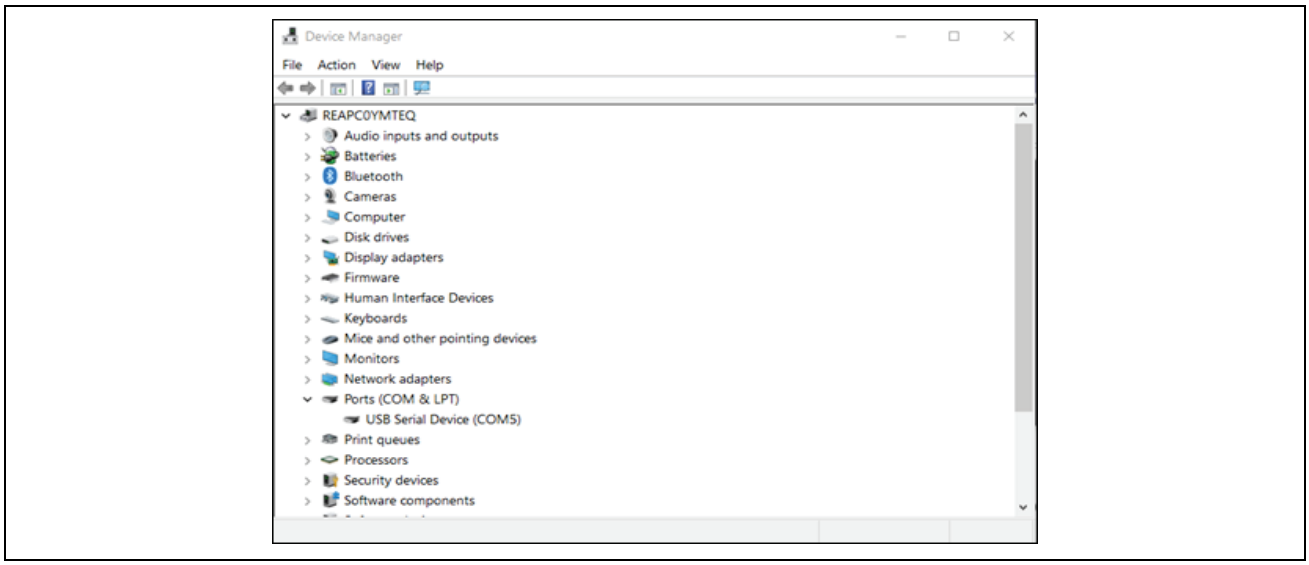

Figure 48. USB Serial Device in Windows Device Manager

2. Open Tera Term, select **New connection** and select **Serial** and **COMxx: USB Serial Device (COMxx)** and click **OK**.

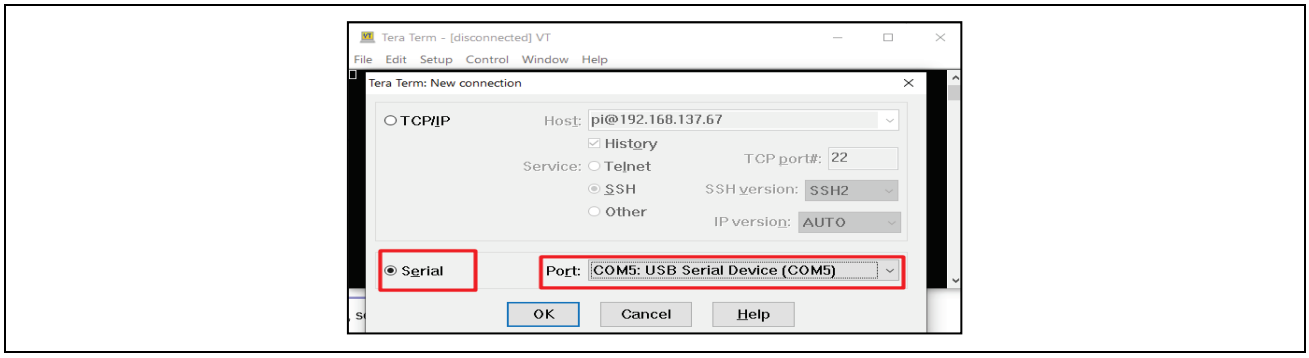

Figure 49. Selecting the Serial Port on Tera Term

3. Using the **Setup** menu pull-down, select **Serial port…** and ensure that the speed is set to **115200**.

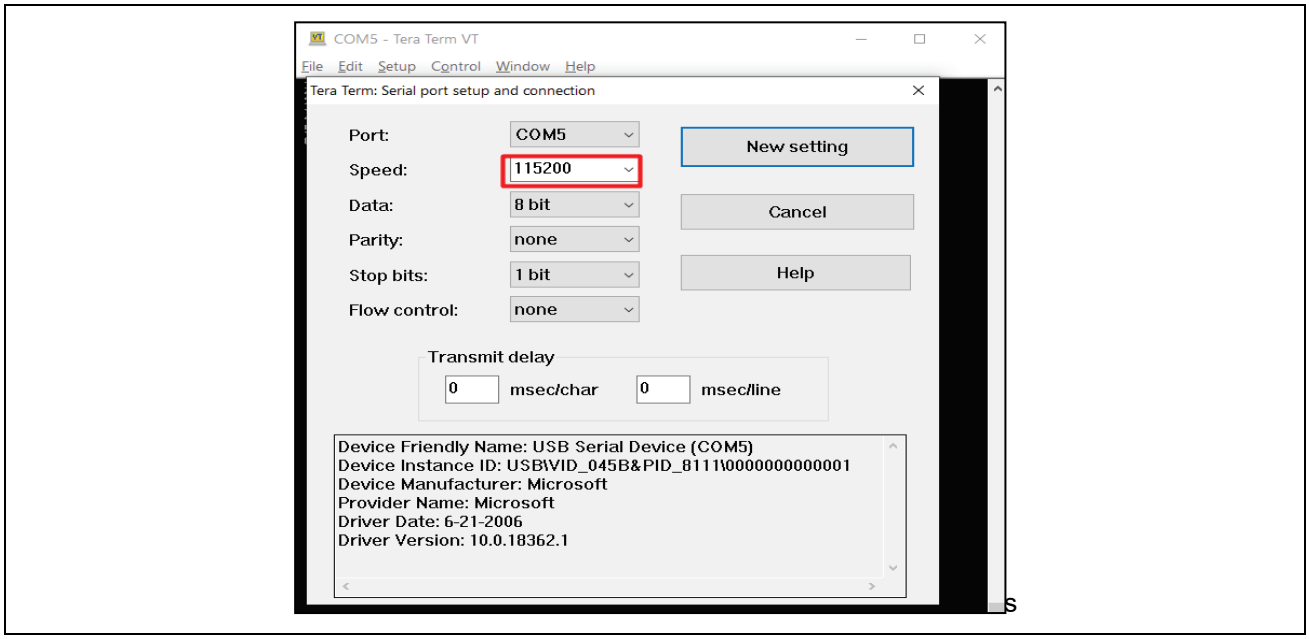

Figure 50. Select 115200 on the Speed Pulldown

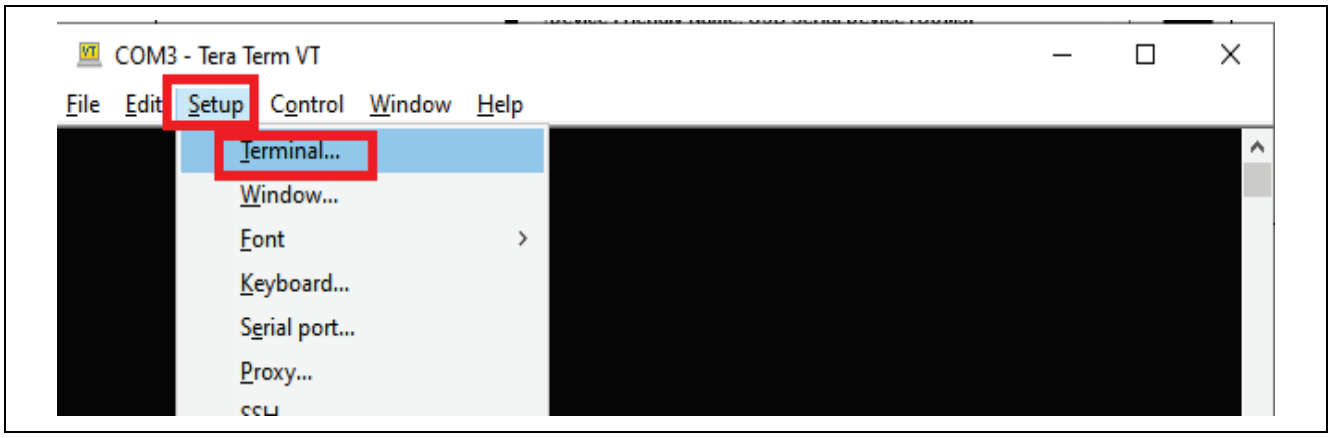

Figure 51. Tera term Settings

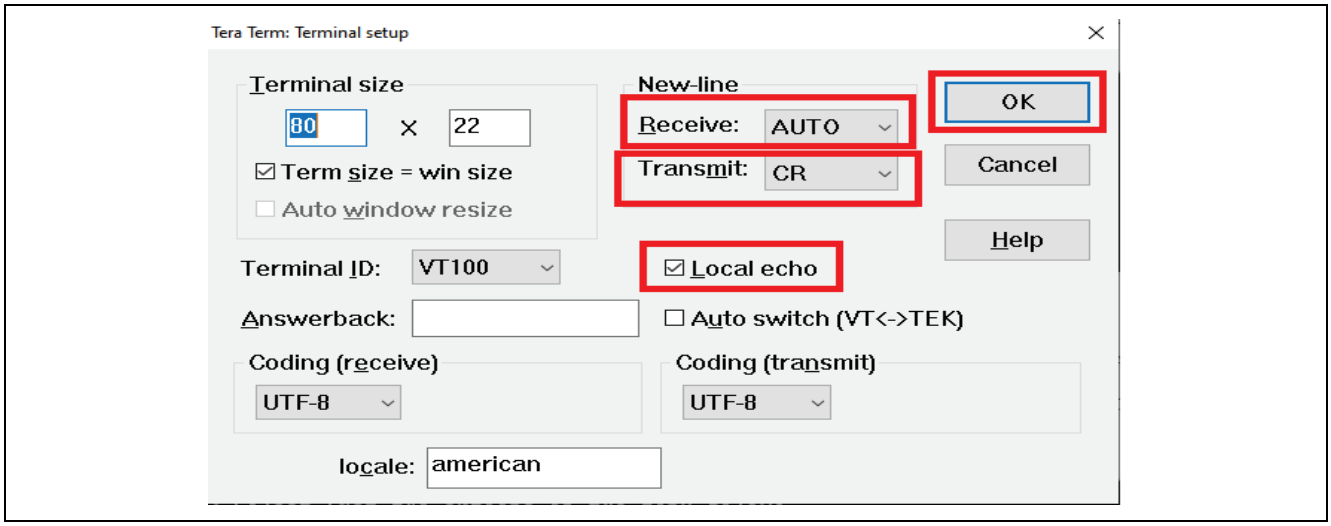

Figure 52. Tera Term Settings

4. Complete the connection. The Configuration CLI Menu will be displayed on the console as shown below.
   Note: Please reset the board by pressing the S1 user switch if the menu is not displayed.

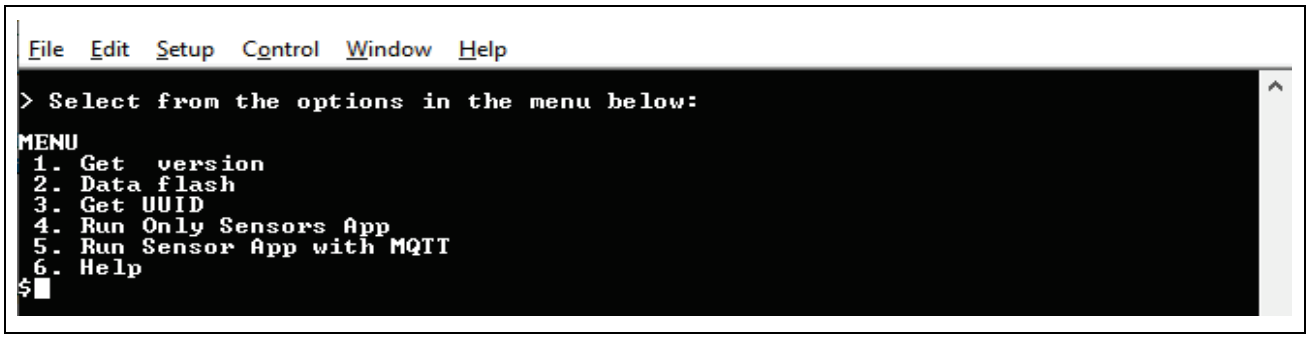

Figure 53. Main Menu

5. You can select options from the MENU by pressing key **1 to 5**. Press spacebar to go to previous menu FSP Version and UUID details as follows.

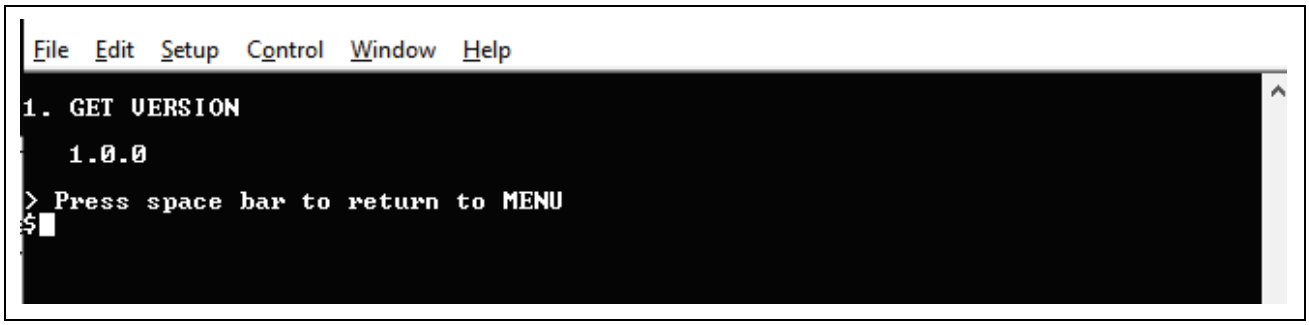

Figure 54. Version Information

Figure 55. Getting Board UUID Information

6.3 Storing Device Certificate, Host Name, Device ID

Please reset the board by pressing the S1 user switch if the menu is not displayed.

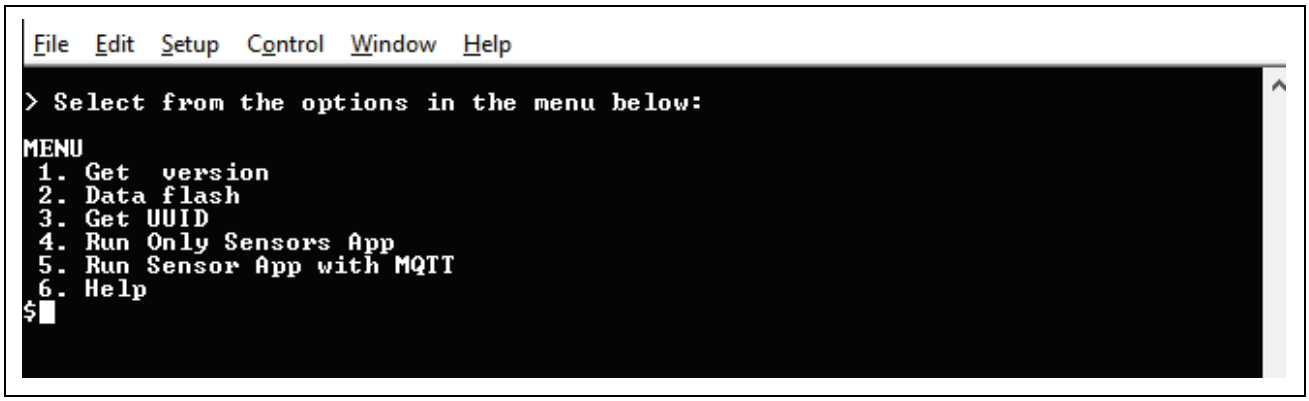

Figure 56. Main Menu

1. Press 2 on the Main Menu to display Data Flash related commands as shown in the following screenshots. This sub menu has commands to store, read, and validate the data.

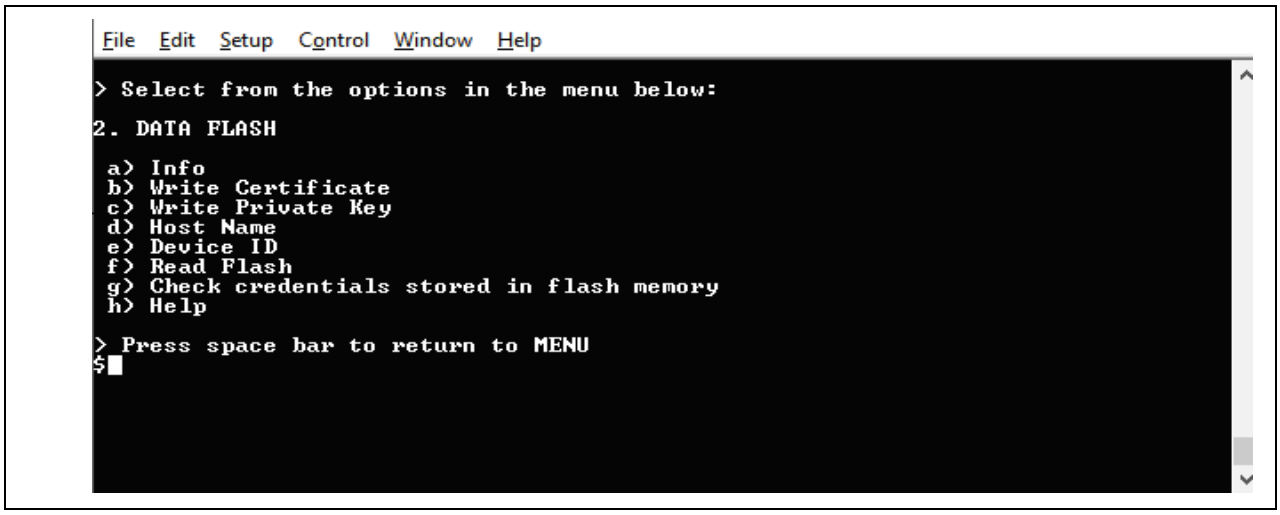

Figure 57. DATA Flash Menu

2. Press 2 on the Main Menu to display Data Flash related commands as shown in the following screenshots. This sub menu has commands to store, read, and validate the data.
3. Press **b** for **Write Certificate.**

Figure 58. Select file to write data in data flash.

4. Go to **Tera Term > File > Send file.**

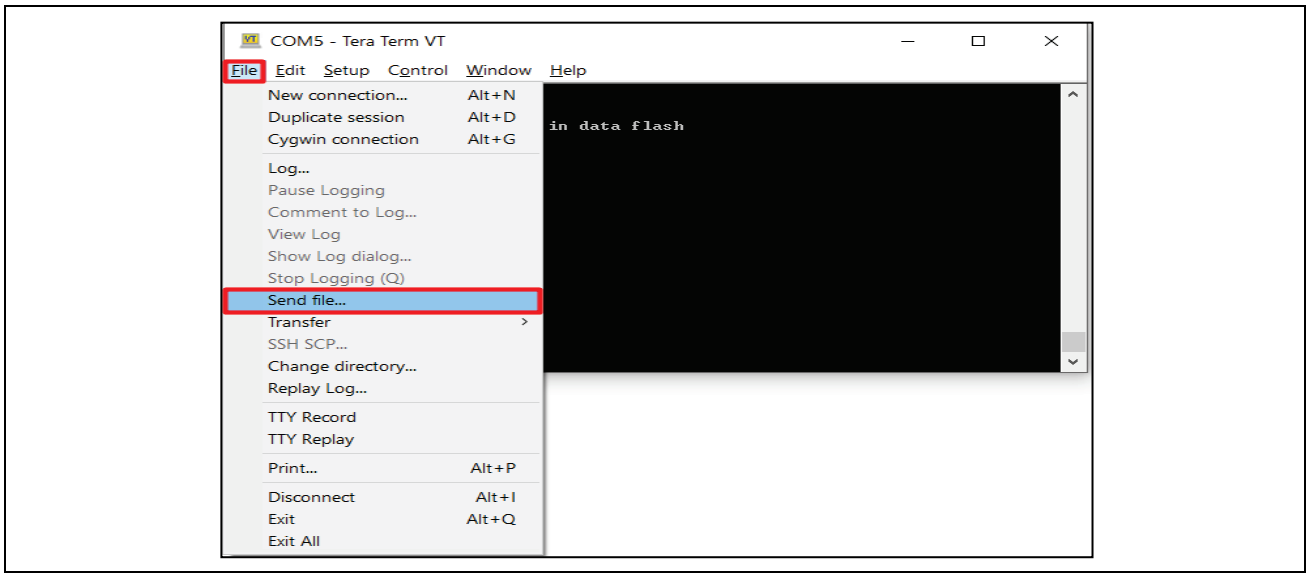

Figure 59. Send File Option in File Menu

5. Browse to the folder where X509 certificates are generated.
Select **cert.pem**. Press **Open**.

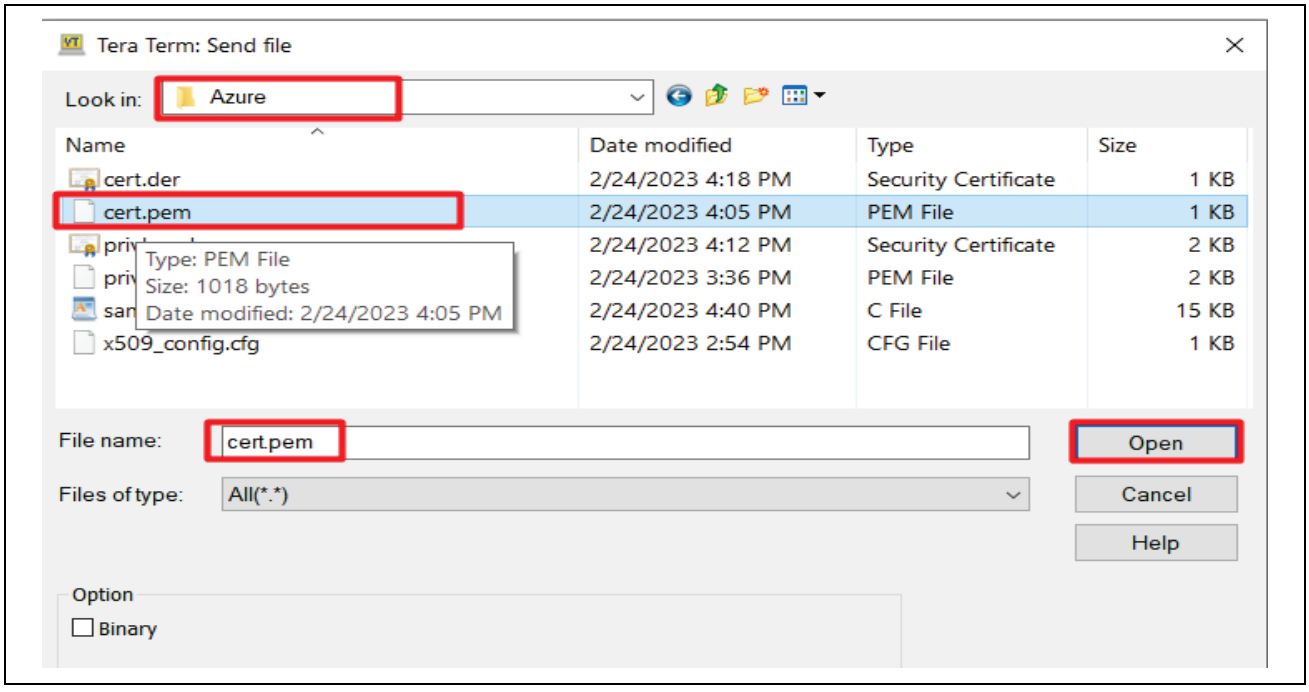

Figure 60. Browse, Select and Open the File to be Written

6. Status of Device Certificate Downloading as below

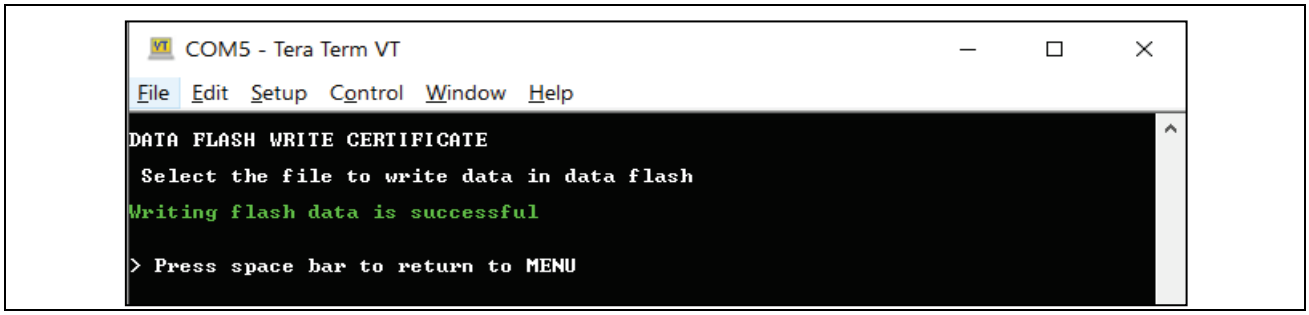

Figure 61. Status of File Writing Process

7. Similarly, to store the device private key, press **c** in **Data Flash** menu, go to **Tera Term > File > Send file,** select file **privkey.pem** from the folder where you have generated certificates.
8. To store MQTT Broker End point aka **Host Name,** first copy Host Name without double quotes then press **d** in **Data Flash** menu, go to **Tera Term >Edit > Paste<CR>;** you will get the copied Host Name in the clipboard, please verify and confirm it and press **OK.**

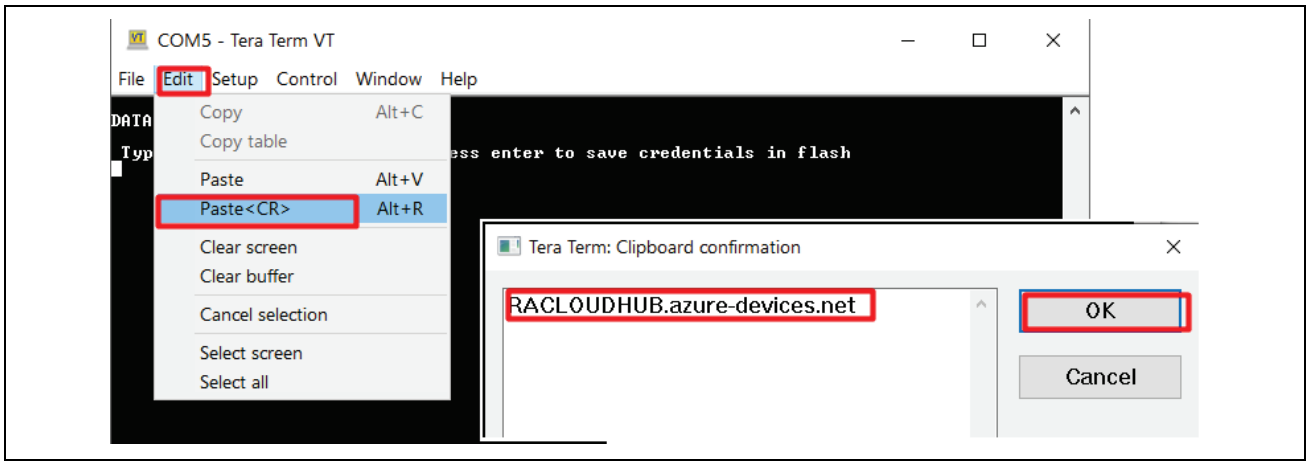

Figure 62. Input MQTT Broker End point aka Host Name

9. To store IoT Thing Name aka **DEVICE ID**, first copy DEVICE ID created without double quotes, press **e** in **Data Flash** menu and follow the procedure in step 5.

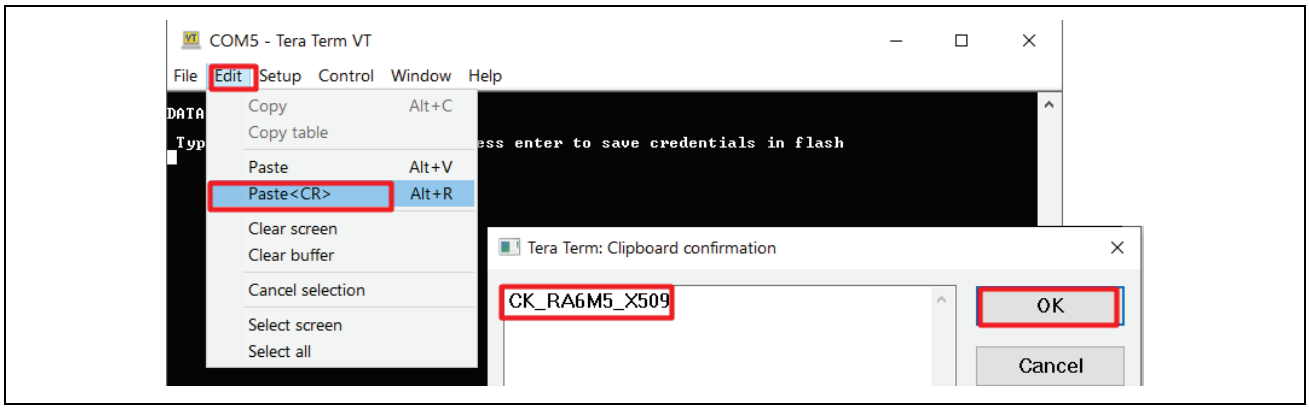

Figure 63. Input Device ID aka IoT Thing Name

10. To verify the data stored in DATA flash press **f** in **Data Flash** menu, scroll down to see data**.**

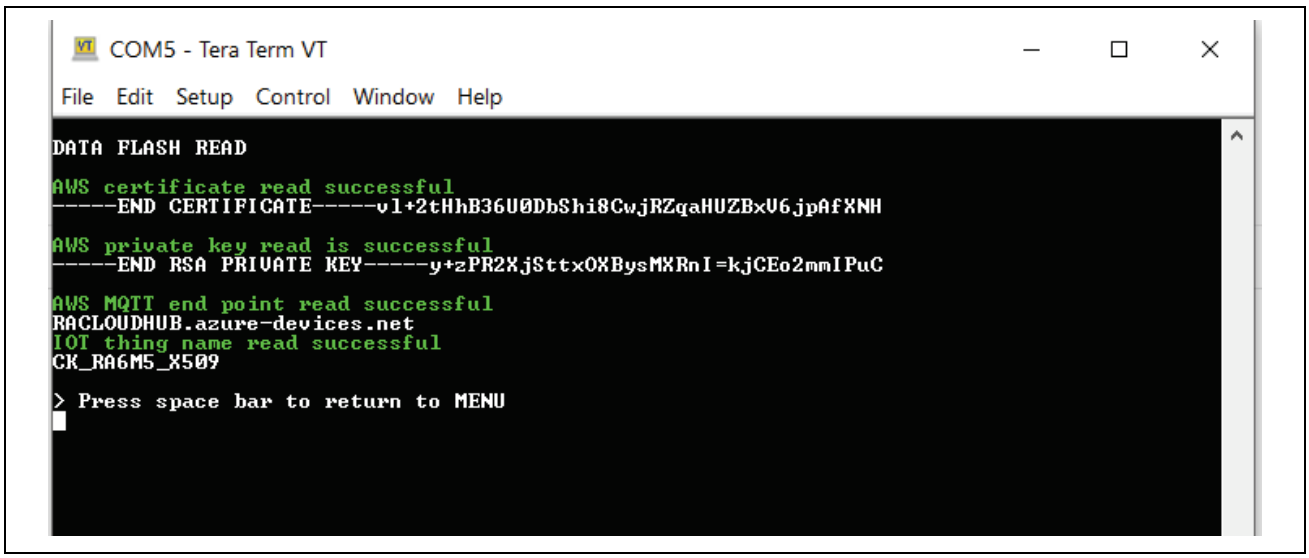

Figure 64. Scroll Down and Verify the Data Stored in Data Flash

11. Sensor Data Output on Serial Terminal.

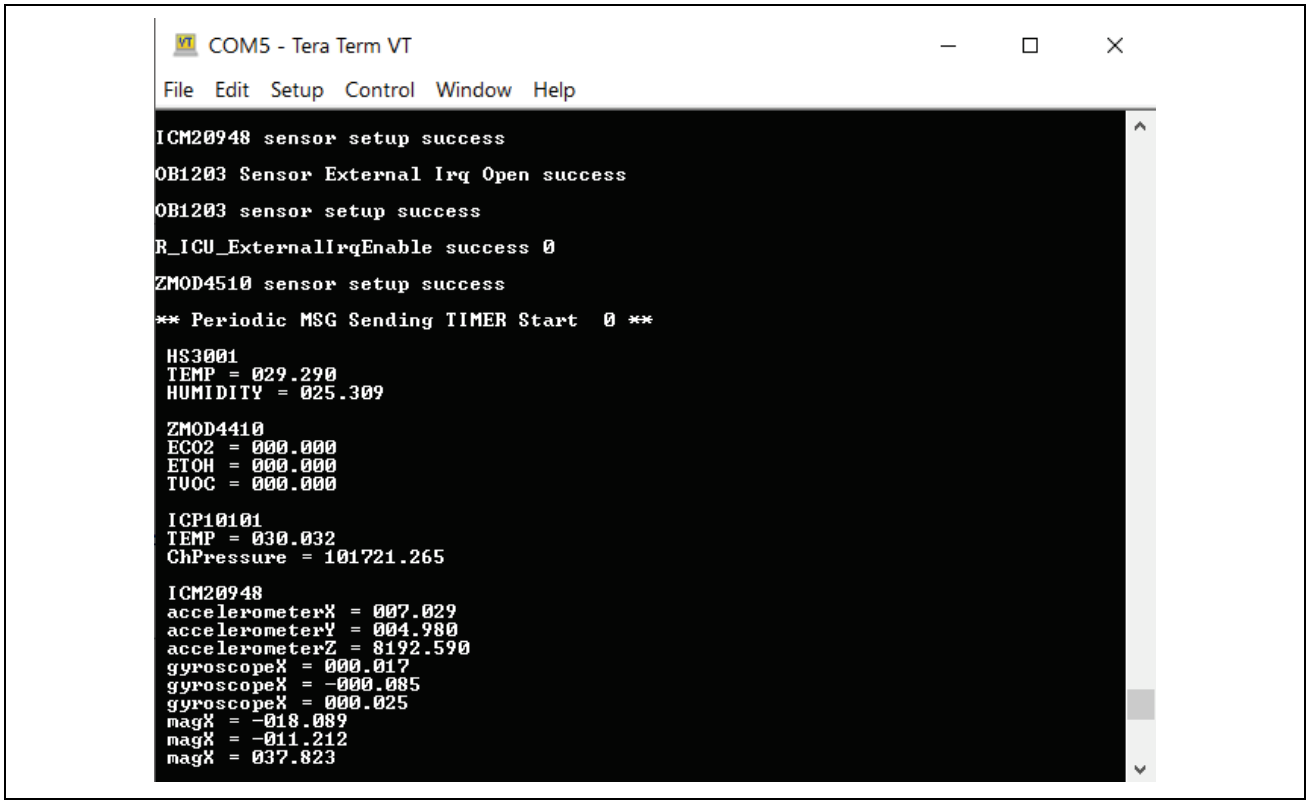

Figure 65. Sensor Data on Serial Terminal

6.4 Send Device to Cloud Message

With Azure IoT Explorer, you can view the flow of telemetry from your device to the cloud. To view telemetry in Azure IoT Explorer:

1. In IoT Explorer select **Telemetry**. Confirm that **use built-in event hub** is set to **Yes**.
2. Select **Start**.
3. View the telemetry as the device sends messages to the cloud.

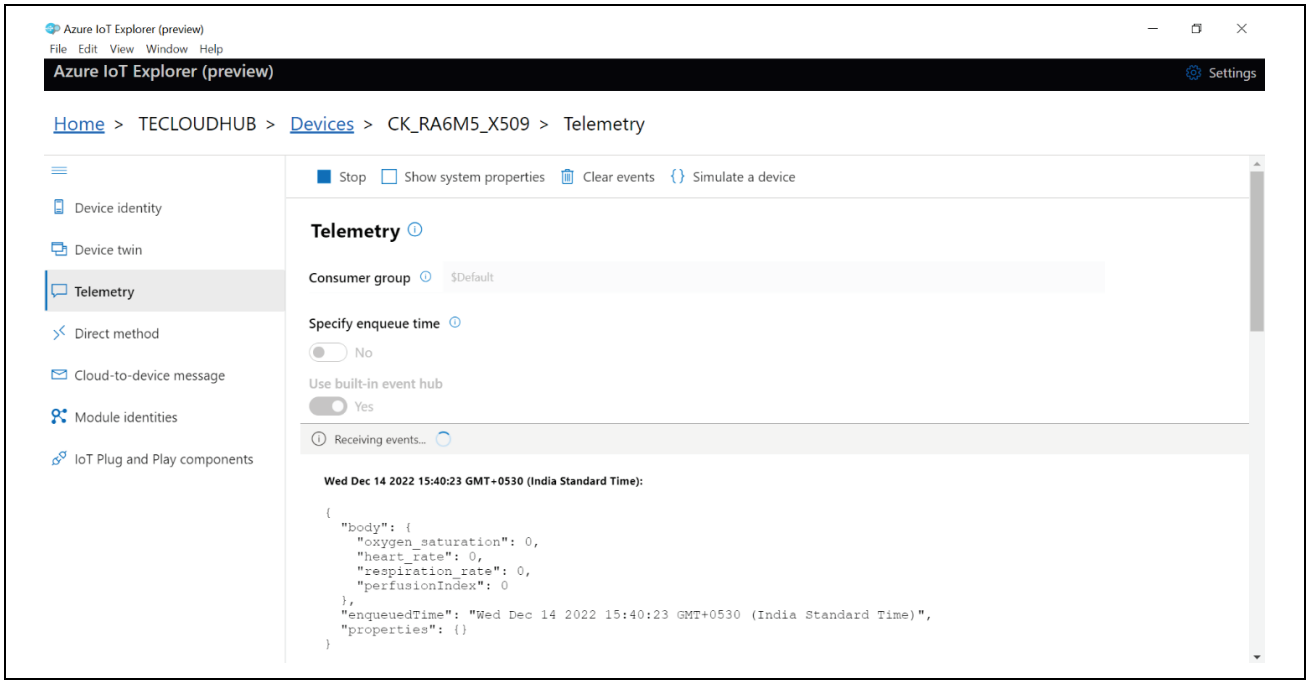

Figure 66. Device Telemetry Details

7. IoT Plug and Play Certification Requirements

This section outlines the device specific capabilities that will be represented in the Azure IoT Device catalog. A capability is a singular device attribute that may be software implementation or combination of software and hardware implementations.

7.1 Program Purpose

IoT Plug and Play enables solution builders to integrate smart devices with their solutions without any manual configuration. At the core of IoT Plug and Play, is a device model that advertises the device capabilities to an IoT Plug and Play-enabled application.

Promise of IoT Plug and Play certification are:

1. Defined device models and interfaces are compliant with the Digital Twin Definition Language.
2. Easy integration with Azure IoT based solutions using the Digital Twin APIs : Azure IoT Hub and Azure IoT Central.
3. Product truth validated through testing telemetry from end point to cloud using DTDL.

Note: Upon completed testing and validation, we may request that the product is evaluated by Microsoft®.

7.2 Requirements

[Required] Device to cloud: The purpose of this test is to make sure that devices that send telemetry work with the IoT Hub

Name	IoTPnP.D2C
Target Availability	Available now
Applies to	Leaf device/Edge device
OS	Agnostic
Validation Type	Automated
Validation	Device must send any telemetry schemas to IoT Hub. Microsoft provides the portal workflow to execute the tests. Device to cloud (required): **1.** Validates that the device can send message to AICS managed IoT Hub **2.** User must specify the number and frequency of messages. **3.** AICS validates the telemetry is received by the Hub instance
Resources	**Certification steps (has all the additional resources)**

[Required] DPS: The purpose of test is to check the device implements and supports IoT Hub Device Provisioning Service with one of the three attestation methods

Name	**AzureCertified.DPS**
Target Availability	New
Applies To	Any device
OS	Agnostic
Validation Type	Automated

Name	**AzureCertified.DPS**
Validation	Device supports easy input of target DPS ID scope ownership. Microsoft provides the portal workflow to execute the tests to validate that the device supports DPS **1.** User must select one of the attestation methods (X.509, TPM and SAS key) **2.** Depending on the attestation method, user needs to take corresponding action such as **a)** Upload X.509 cert to AICS managed DPS scope **b)** Implement SAS key or endorsement key into the device
Resources	Device provisioning service overview

[Required] DTDL v2: The purpose of test to ensure defined device models and interfaces are compliant with the Digital Twins Definition Language v2.

Name	**IoTPnP.DTDL**
Target Availability	Available now
Applies To	Any device
OS	Agnostic
Validation Type	Automated
Validation	The portal workflow validates: **1.** Model ID announcement and ensure the device is connected using either the MQTT or MQTT over WebSockets protocol **2.** Models are compliant with the DTDL v2 **3.** Telemetry, properties, and commands are properly implemented and interact between IoT Hub Digital Twin and Device Twin on the device
Resources	Public Preview Refresh updates

[Required] Device models are published in public model repository

Name	**IoTPnP.ModelRepo**
Target Availability	Available now
Applies To	Any device
OS	Agnostic
Validation Type	Automated
Validation	All device models are required to be published in public repository. Device models are resolved via models available in public repository **1.** User must manually publish the models to the public repository before submitting for the certification. **2.** Note that once the models are published, it is immutable. We strongly recommend publishing only when the models and embedded device code are finalized.*1 *1 User must contact Microsoft support to revoke the models once published to the model repository **3.** Portal workflow checks the existence of the models in the public repository when the device is connected to the certification service
Resources	Model repository

[If implemented] Device info Interface: The purpose of test is to validate device info interface is implemented properly in the device code

Name	**IoTPnP.DeviceInfoInterface**
Target Availability	Available now
Applies To	Any device
OS	Agnostic
Validation Type	Automated
Validation	Portal workflow validates the device code implements device info interface **1.** Checks the values are emitted by the device code to IoT Hub **2.** Checks the interface is implemented in the DCM (this implementation will change in DTDL v2) **3.** Checks properties are not write-able (read only) **4.** Checks the schema type is string and/or long and not null
Resources	Microsoft defined interface
Azure Recommended	N/A

[If implemented] Cloud to device: The purpose of test is to make sure messages can be sent from cloud to devices

Name	**IoTPnP.C2D**
Target Availability	Available now
Applies To	Leaf device/Edge device
OS	Agnostic
Validation Type	Automated
Validation	Device must be able to Cloud to Device messages from IoT Hub. Microsoft provides the portal workflow to execute these tests. Cloud to device (if implemented): **1.** Validates that the device can receive message from IoT Hub **2.** AICS sends random message and validates via message ACK from the device
Resources	**1.** Certification steps (has all the additional resources), **2.** Send cloud to device messages from an IoT Hub

[If implemented] Direct methods: The purpose of test is to make sure devices works with IoT Hub and supports direct methods

Name	**IoTPnP.DirectMethods**
Target Availability	Available now
Applies To	Leaf device/Edge device
OS	Agnostic

Name	**IoTPnP.DirectMethods**
Validation Type	Automated
Validation	Device must be able to receive and reply commands requests from IoT Hub. Microsoft provides the portal workflow to execute the tests. Direct methods (if implemented): **1.** User has to specify the method payload of direct method. **2.** AICS validates the specified payload request is sent from Hub and ACK message received by the device
Resources	**1.** Certification steps (has all the additional resources), **2.** Understand direct methods from IoT Hub

[If implemented] Device twin property: The purpose of test is to make sure devices that send telemetry works with IoT Hub and supports some of the IoT Hub capabilities such as direct methods, and device twin property

Name	**IoTPnP.DeviceTwin**
Target Availability	Available now
Applies To	Leaf device/Edge device
OS	Agnostic
Validation Type	Automated
Validation	Device must send any telemetry schemas to IoT Hub. Microsoft provides the portal workflow to execute the tests. Device twin property (if implemented): **1.** AICS validates the read/write-able property in device twin JSON **2.** User has to specify the JSON payload to be changed **3.** AICS validates the specified desired properties sent from IoT Hub and ACK message received by the device
Resources	**1.** Certification steps (has all the additional resources), **2.** Use device twins with IoT Hub

8. Azure IoT Central Architecture

- Devices
- Gateways
- Export data

IoT Central is a ready-made environment that lets you quickly evaluate your IoT scenario. It is an application platform as a service (aPaaS) IoT solution and its primary interface is a web UI. There's also a REST API that lets you interact with your application programmatically.

This article provides an overview of the key elements in an IoT Central solution architecture.

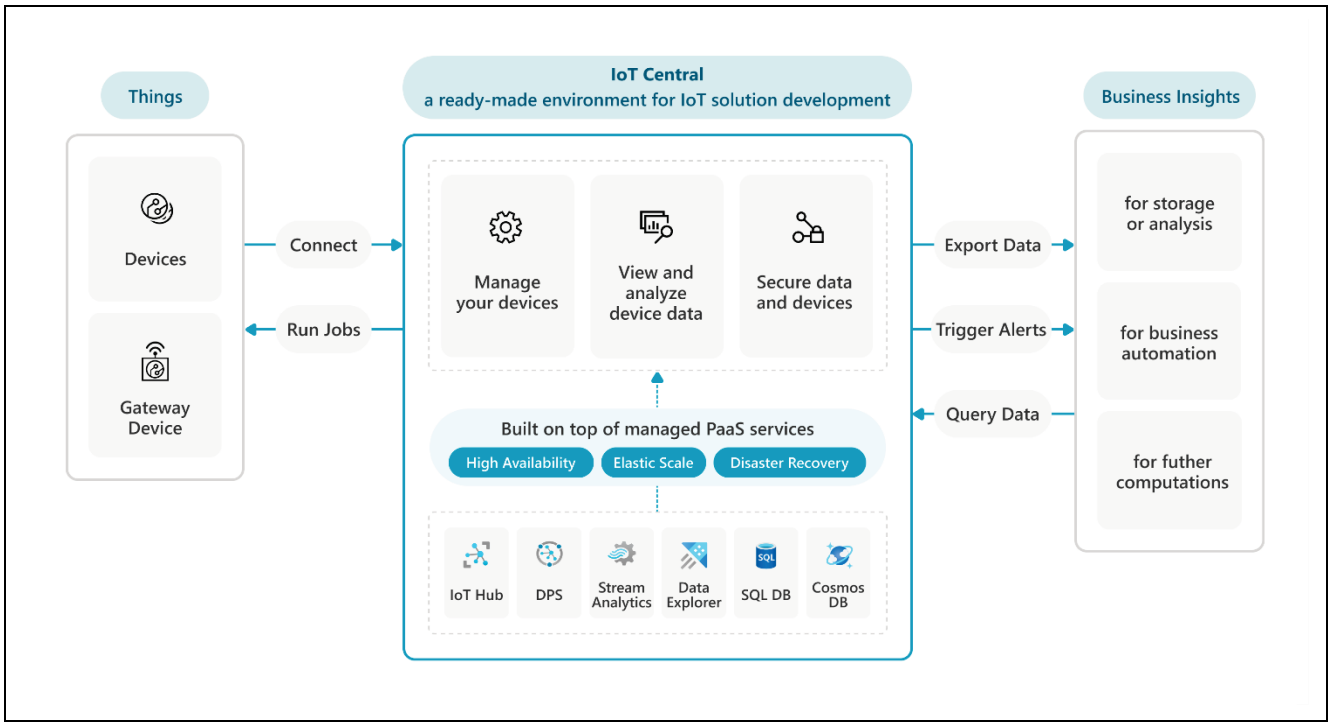

Figure 67. IoT Central Architecture

Key capabilities in an IoT Central application are described in the following sections.

8.1 Manage Devices

IoT Central lets you manage the fleet of IoT devices that are sending data to your solution. For example, you can:

- Control which devices can connect to your application and how they authenticate.
- Use device templates to define the types of device that can connect to your application.
- Manage devices by setting properties or calling commands on connected devices. For example, set a target temperature property for a thermostat device or call a command to trigger a device to update its firmware. You can set properties and call commands on:
  - Individual devices through a customizable web UI.
  - Multiple devices with scheduled or on-demand jobs.
  - Maintain device metadata such as customer address or last service date.

8.1.1 View and Analyze Data

In an IoT Central application, you can view and analyze data for individual devices or for aggregated data from multiple devices:

- Use mapping to transform complex device telemetry into structured data inside IoT Central.
- Use device templates to define custom views for individual devices of specific types. For example, you can plot temperature over time for an individual thermostat or show the live location of a delivery truck.
- Use the built-in analytics to view aggregate data for multiple devices. For example, you can see the total occupancy across multiple retail stores or identifying the stores with the highest or lowest occupancy rates.
- Create custom dashboards to help you manage your devices. For example, you can add maps, tiles, and charts to show device telemetry.

8.1.2 Secure your Solution

In IoT Central, you can configure and manage security in the following areas:

- User access to your application.
- Device access to your application.
- Programmatic access to your application.
- Authentication to other services from your application.
- Audit logs track activity in your application.

To learn more, see the IoT Central security guide.

8.2 Devices

Devices collect data from sensors to send as a stream of telemetry to an IoT Central application. For example, a refrigeration unit sends a stream of temperature values or a delivery truck streams its location.

A device can use properties to report its state, such as whether a valve is open or closed. An IoT Central application can also use properties to set device state, for example setting a target temperature for a thermostat.

IoT Central can also control devices by calling commands on the device. For example, instructing a device to download and install a firmware update.

The telemetry, properties, and commands that a device implements are collectively known as the device capabilities. You define these capabilities in a model that's shared between the device and the IoT Central application. In IoT Central, this model is part of the device template that defines a specific type of device. To learn more, see Assign a device to a device template.

The device implementation should follow the IoT Plug and Play conventions to ensure that it can communicate with IoT Central. For more information, see the various language SDKs and samples.

Devices connect to IoT Central using one the supported protocols: MQTT, AMQP, or HTTP.

8.3 Gateways

Local gateway devices are useful in several scenarios, such as:

- Devices cannot connect directly to IoT Central because they cannot connect to the internet. For example, you may have a collection of Bluetooth enabled occupancy sensors that need to connect through a gateway device.
- The quantity of data generated by your devices is high. To reduce costs, combine or aggregate the data in a local gateway before you send it to your IoT Central application.
- Your solution requires fast responses to anomalies in the data. You can run rules on a gateway device that identify anomalies and take an action locally without the need to send data to your IoT Central application.

Gateway devices typically require more processing power than a standalone device. One option to implement a gateway device is to use Azure IoT Edge and apply one of the standard IoT Edge gateway patterns. You can also run your own custom gateway code on a suitable device.

8.4 Export Data

Although IoT Central has built-in analytics features, you can export data to other services and applications.

Transformations in an IoT Central data export definition let you manipulate the format and structure of the device data before it's exported to a destination.

9. Note and Troubleshooting

9.1 About Stabilization Time for Sensor

There is stabilization time for each sensor. It cannot read correct values during the time.

The following table shows the detail of stabilization time of each sensor of table.

Table 6. Sensor Stabilization Time

Sensor Name	When power up first time	After soft or hard reset
ZMOD4410 IAQ	Up to 1 min	Up to 1 min
ZMOD4510 OAQ	Up to 1.5 hours	Up to 1 hour
OB1203	Up to 20 min (After putting finger on sensor, it may take up to 60 secs to sense data)	Up to 20 sec (After putting finger on sensor, it may take up to 60 secs to sense data)
HS3001	Up to 30 seconds	Up to 10 seconds
ICP	Up to 30 seconds	Up to 10 seconds
ICM	Up to 30 seconds	Up to 10 seconds

9.2 Connection Issue When using Ethernet (Wired Cable)

The Ether PYH only supports full duplex communication. If your router or Ethernet hub only supports half duplex, it cannot connect to the internet. Please use full duplex devices.

9.3 Current Supply Short Issue When using RYZ014A

If the CK-RX65N board is not powered through the Debug port (J14) the current available to the board may be limited to 100 mA. When using the supplied RYZ014A Pmod module with other code (found here: RYZ014A - LTE Cat-M1 Cellular IoT Module | Renesas), be aware that this Pmod has a maximum operating current of **480 mA** dependent upon the LTE band, Tx/Rx settings, and network coverage. Please ensure that the host board can supply sufficient power or provide supplemental USB power via CN4 on the Pmod to avoid RF instability.

9.4 When Build Errors Occur

If a 'No such file or directory' error occurs, the project path may be too long. When the path is longer than 256 characters, e^2 studio outputs errors at build time.
When this error occurs, move the project to a shorter path location (for example, under C:\).

Website and Support

Visit the following vanity URLs to learn about key elements of the RX family, download components and related documentation, and get support.

CK-RX65N Kit Information	renesas.com/rx/ck-rx65n
RX&RA Cloud Solutions	renesas.com/cloudsolutions
RX Cloud solution web	renesas.com/rx-cloud
RX Product Information	renesas.com/rx
RX Product Support Forum	renesas.com/rx/forum
RX Driver Package	renesas.com/RDP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar.20.23	—	First version
1.10	Jun.02.23	5	Added 4.1 about the activation procedure for Truphone SIM
1.11	Jan.19.24	2	Add “Important notice” for Azure RTOS
		4	Correct sensor’s name (ZMOD4xxx) in Figure 2
		5	Add notice about SIM Card’s information which is included in kit
		—	Remove section “4.2 Activating a SIM card on MicroAI Launchpad” due to the MicroAI SIM card is discontinued to support CK-RX65N.
		—	Hidden personal information: Figure 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 28, 29.
		—	Remove section “6.5 Send Cloud-to-Device Message” which haven’t full supported yet.
		—	Fix typo mistake
1.12	Mar.12.24	—	Fixed issues in the table of contents and headings.

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on
   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins
   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.
(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.