AN12982

Cloud-based condition monitoring for industrial motors

Rev. 1.1 — 10 February 2021 Application note

Document information

Information	Content
Keywords	NXP quad motor-control development platform, Azure IoT Hub, EdgeLock SE050, secure cloud onboarding, secure authentication, secure communication channel, trust provisioning, key management, root of trust.
Abstract	This document details the steps for bringing up the Azure IoT Hub demo application, a software example provided as part of the NXP quad motor-control development platform enablement kit. This demo implements a sample application that allows users to collect telemetry data and control multiple motors remotely, leveraging Azure IoT Hub cloud service platform.

Revision history

Revision history

Revision number	Date	Description
1.0	2020-10-28	First version.
1.1	2021-02-10	• Updated figures to the latest version • Specified software version

1 About the NXP quad motor-control development platform

The NXP quad motor-control development platform is a cost-effective platform to allow rapid development of multi-motor control applications. It is based on NXP's i.MX RT1050 crossover processor, able to operate up to four motors and able to address the increased need for cost-constrained, centralized motor-control systems.

The NXP quad motor-control development platform uses a modular architecture. It is supported by dedicated motor-control software libraries and NXP FreeMaster real-time debugger. It exhibits a flexible design consisting of a *daughter board*, a *digital board* and a *power stage board* as illustrated in Figure 1.

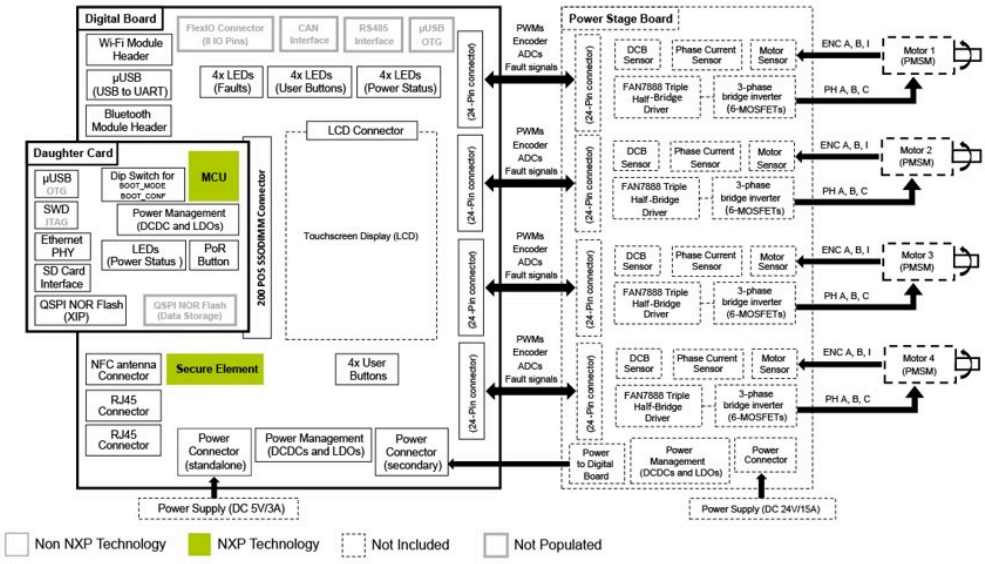

Figure 1. NXP quad motor-control development platform architecture

The *daughter board* embeds an i.MX RT1050 crossover processor and is the core of this motor-control reference design. It is based on a 528 MHz ARM Cortex-M7 core and comes with high-speed communication and peripheral interfaces, advanced graphics support for industrial HMI and sensor interfaces. As a result, the i.MX RT1050 crossover processor provides a high level of integration in sophisticated automation and multi-motor applications.

The *digital board* works as an external platform to prototype your next multi-motor control application. It includes the headers and the footprints required to easily plug-in widely used industrial communication and peripheral interfaces supported by the i.MX RT1050 crossover processor. It also provides the connectors that expose the control signals for the four motor devices.

The *power stage board* supports the control and connection of up to four PMSM or BLDC motors. It includes the power management unit, provides the motor-control capabilities, such as rotational or linear motion, and comes with motor connectors built in. The PCB of the *power stage board* is not delivered, but its design files are made available as part of the NXP quad motor-control development platform.

2 About cloud-based condition monitoring

Condition monitoring is the process of collecting and analyzing certain equipment parameters and key operation indicators. With it, we aim to predict whether an asset will break, how it will break, and how much time you have to fix or replace the asset before it functionally fails. As such, the use of condition monitoring allows maintenance to be scheduled, or other actions to be taken to prevent damages and avoid their consequences.

In many cases, machines, systems, devices and objects are geographically distributed. This results in critical challenges for maintenance teams, such as lack of existing access to data showing the current condition and health of the company's machinery. In this context, cloud-based condition monitoring solutions allow technicians and managers to access data from any equipment at any time using a computer or their smart devices.

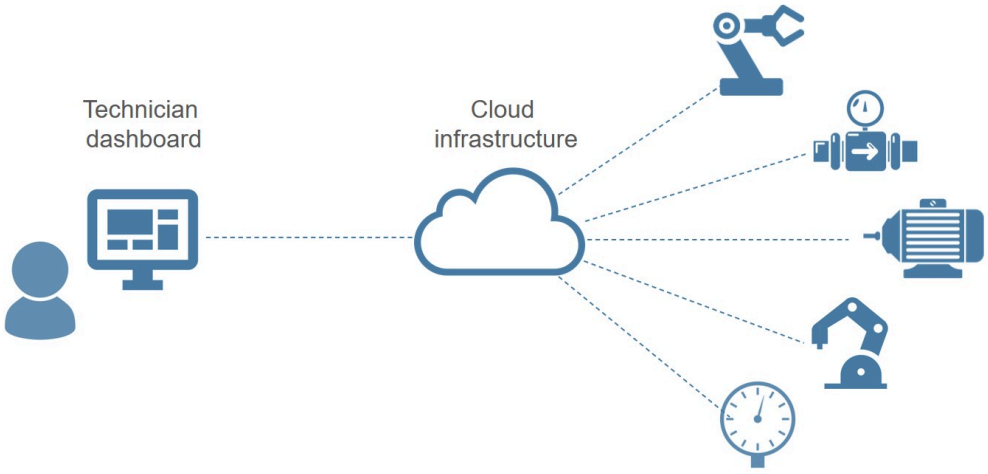

Figure 2. Cloud-based condition monitoring

In addition, cloud-based condition monitoring solutions have several advantages compared to on-premise solutions, including access from anywhere, advanced analytics, configurable dashboards, and scalability, etc. For instance, timely alerts can be configured in case there is a violation in the limit values of the parameters under monitoring. This way, immediate action can be taken to avoid unwanted downtime, saving time and money.

The rapid development of Industry 4.0 revolves significantly around Internet of Things. Numbers of things are efficiently interconnected, especially in industrial automation, which leads to condition and controlled monitoring to increase productivity. In this respect, many of the public cloud service providers offer IoT services using standard protocols for real-time storage and data aggregation. This makes it possible to access real-time data online, reduce operational risks and lower service costs, among others.

3 About Azure IoT Hub demo application

The Azure IoT Hub demo application is a software example provided as part of the NXP quad motor-control development platform enablement package. This demo implements a sample application that allows users to collect telemetry data and control multiple motors remotely, leveraging Azure IoT Hub cloud service platform.

Figure 3 illustrates the Azure IoT Hub demo application setup. This setup consists of:

- An NXP quad motor-control development platform, which runs a dedicated software to securely connect to Azure IoT Hub and upload telemetry data from the motors.
- A QMCJsonExchanger Windows application instance, which downloads and displays telemetry data from Azure IoT Hub, and allows users to remotely send control-messages to the NXP quad motor-control development platform from any laptop.
- An Azure IoT Hub account, which is configured to authenticate the NXP quad motor-control development platform and used to connect the QMCJsonExchanger application with the NXP quad motor-control development platform over the Internet.

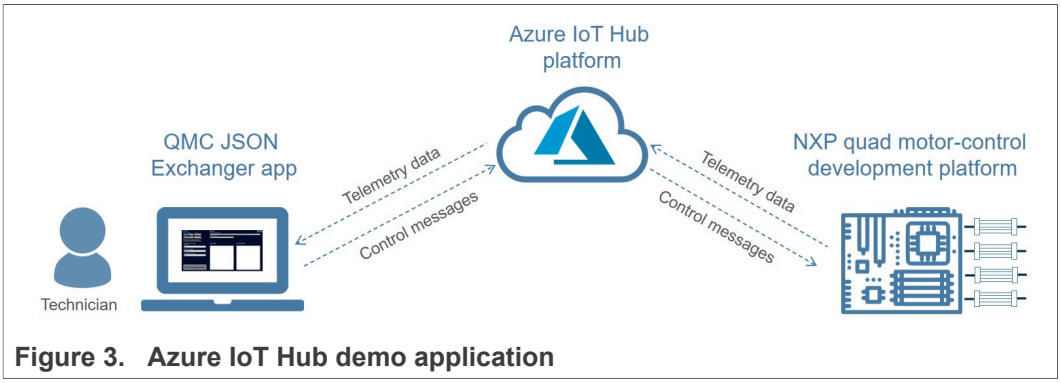

Figure 3. Azure IoT Hub demo application

To enable secure connection with the Azure IoT Hub cloud, the NXP quad motor-control development platform on-boards an EdgeLock SE050 security IC. This IC acts as a security enclave, designed to provide a tamper-resistant memory to securely store keys and credentials needed for device authentication and registration in the cloud. Therefore, with the EdgeLock SE050, the NXP quad motor-control development platform can securely connect to Azure IoT Hub without writing security code or exposing credentials or keys.

This document provides comprehensive instructions for bringing up this Azure IoT Hub demo application. It consists of the following steps:

- Trust provisioning.
- Azure IoT Hub account setup.
- Build and run QMC JSON Exchanger application.
- Run the Azure IoT Hub demo application.

4 Trust provisioning

The device identity should be unique, verifiable and trustworthy so that device registration attempts and any data uploaded to Azure IoT Hub can be trusted by the platform. Azure IoT Hub uses X.509 certificate-based attestation mechanisms for confirming the device authenticity during a registration attempt. This authentication scheme requires a certificate chain of trust, from the CA certificate to the device certificate as well as their associated private key.

This section explains how to generate and provision credentials in EdgeLock SE050, required for the Azure IoT Hub cloud authentication used in this demo. The steps for this procedure are:

- Prepare hardware and plug-in boards to the computer.
- Download and install i.MX RT1050 crossover processor SDK.
- Import Azure MCUXpresso example .
- Flash VCOM binary.
- Download EdgeLock SE050 Plug and Trust middleware.
- Provision credentials using EdgeLock SE050 pycli tool.

Note: *The key generation and injection procedure described in this section is only applicable for evaluation or testing purposes. In a commercial deployment, key provisioning must take place in a trusted environment, in a facility with security features like tightly controlled access, careful personnel screening, and secure IT systems that protect against cyberattacks and theft of credentials.*

4.1 Prepare hardware and plug-in boards to the computer

After importing and building the project, we need to prepare the hardware setup to program the NXP quad motor-control application using the LPC-Link 2 debug probe. Follow these steps:

1. The daughter board allows the selection of the internal boot or serial downloader boot modes of the i.MX RT1050 crossover processor using the SW300 DIP switch. Select the internal boot mode configuring the SW300 DIP switch as described in Table 1:

Table 1. SW300 switch configuration

Switch	State
1	OFF
2	ON
3	ON

Table 1. SW300 switch configuration...*continued*

Switch	State
4	OFF

Figure 4 shows the SW300 configuration in the daughter board PCB:

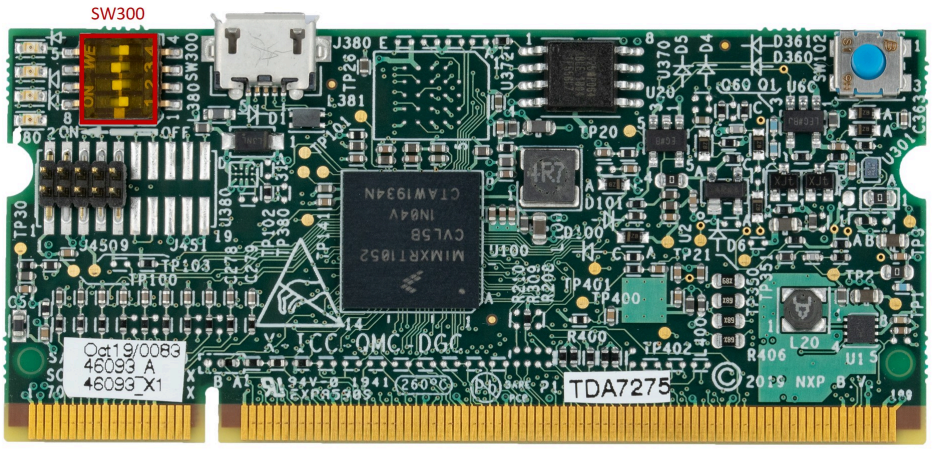

Figure 4. Daughter board SW300 switch configuration

2. Configure LPC-Link 2 board jumpers as shown in Figure 5:
   JP1 open: boot USB DFU.
   JP2 closed: buffers power by LPC-Link 2.
   Note: *For new debug probe MCU-Link, it is available from MCU-Link webpage.*

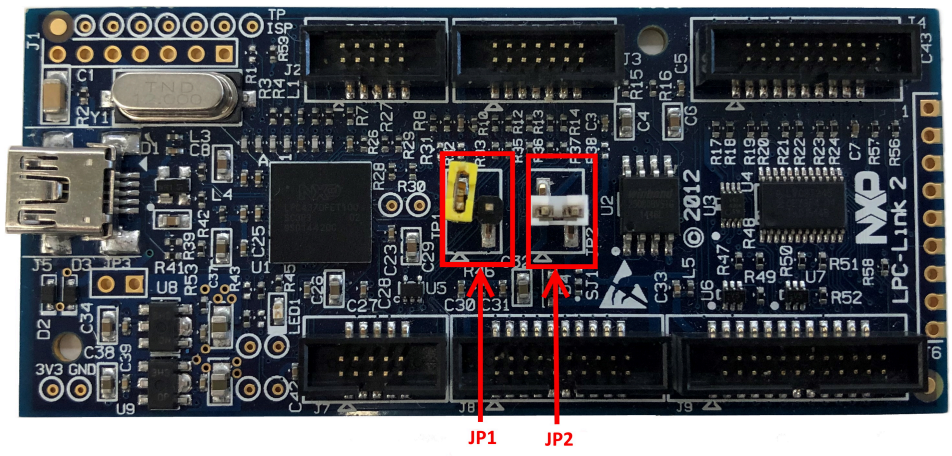

Figure 5. LPC-Link 2 jumper configuration

3. Attach the daughter board to the digital board using the EDGE connector as shown in Figure 6.

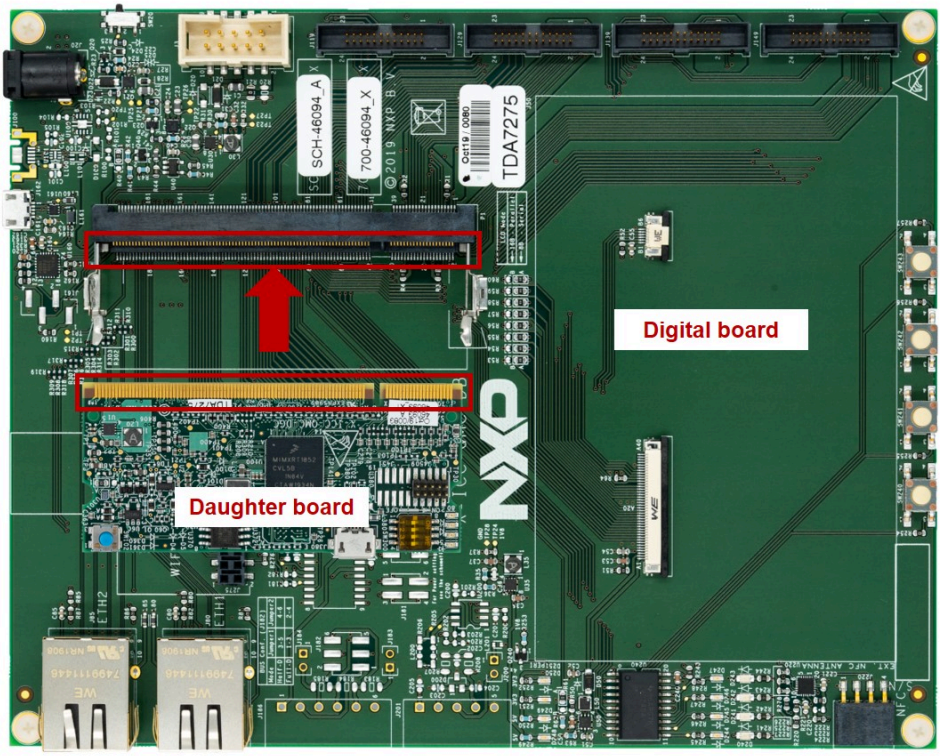

Figure 6. Connect daughter card to digital card

4. Plug-in the boards to the computer as depicted in Figure 7:
   a. Attach a USB cable from the computer to the digital board.
   b. Connect the LPC-Link 2 board (J7 jumper) to the daughter board using the SWD connector.
   c. Attach a USB cable from the computer to the LPC-Link 2 board.
   d. Power on the board using SW20

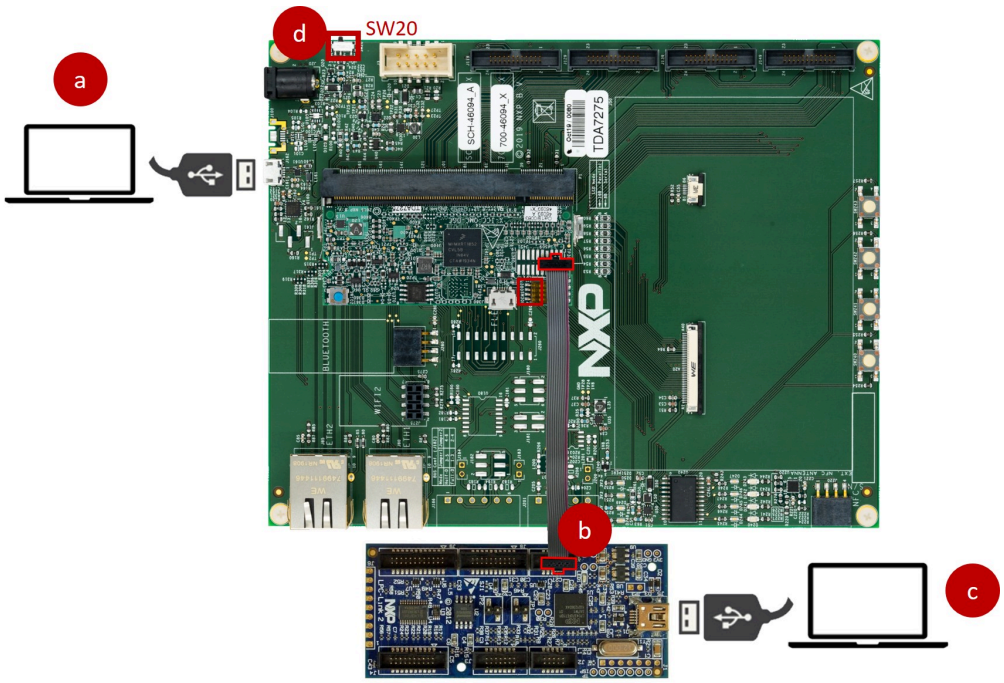

Figure 7. Plug-in the boards to the computer

4.2 Download and install i.MX RT1050 crossover processor SDK

The i.MX RT1050 crossover processor SDK brings open source drivers, middleware, and reference example applications for i.MX RT1050 crossover processor based applications. You can personalize and download your SDK from the MCU SDK Builder website. Follow these instructions to download and install the SDK needed for the NXP quad motor-control application:

1. Go to MCU SDK Builder and click *Select Development Board* as shown in Figure 8:

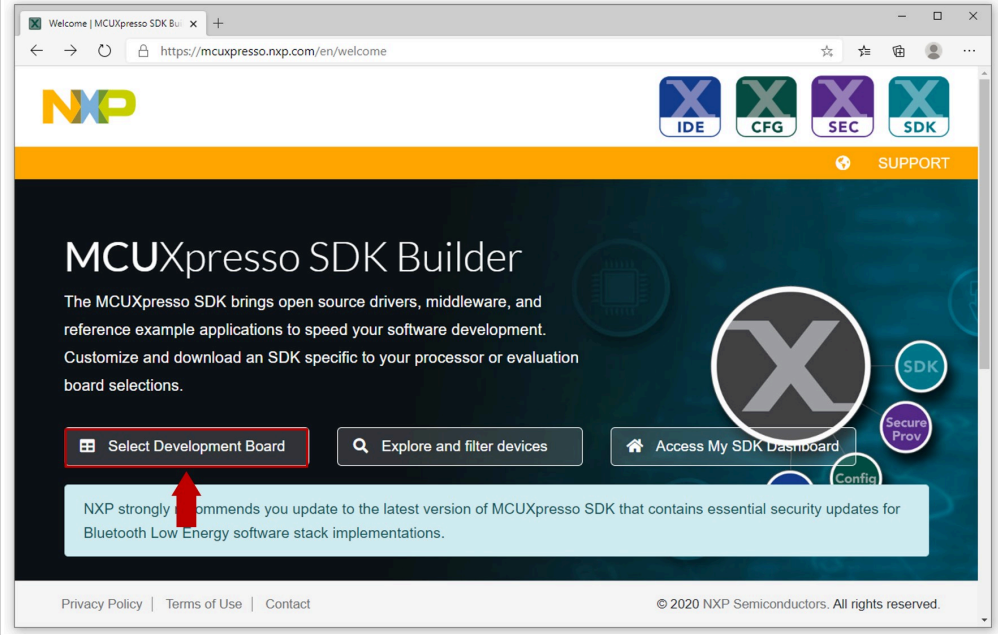

Figure 8. Select development board

2. You will be asked to sign-in with your NXP account (if you don't have one yet, click on **Register Now**), type your credentials in the fields and then click on **Sign in** as indicated in Figure 9.

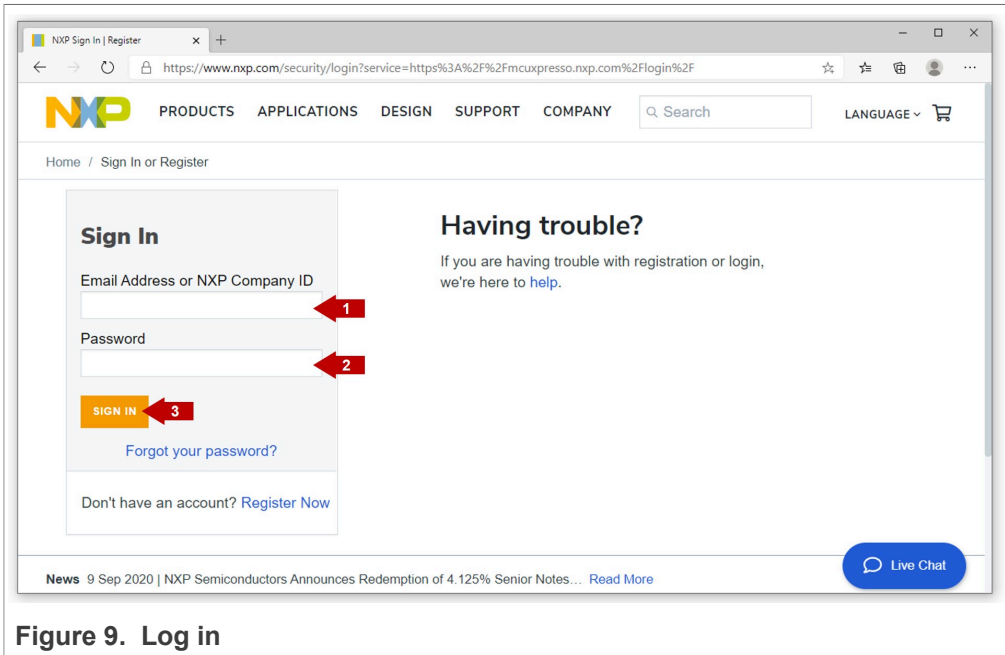

Figure 9. Log in

3. Type MIMXRT1052xxxxB on the text box (1), select it (2) in the *Processors* drop-down list and click (3) ***Build MCUXpresso SDK*** button as shown in Figure 10:

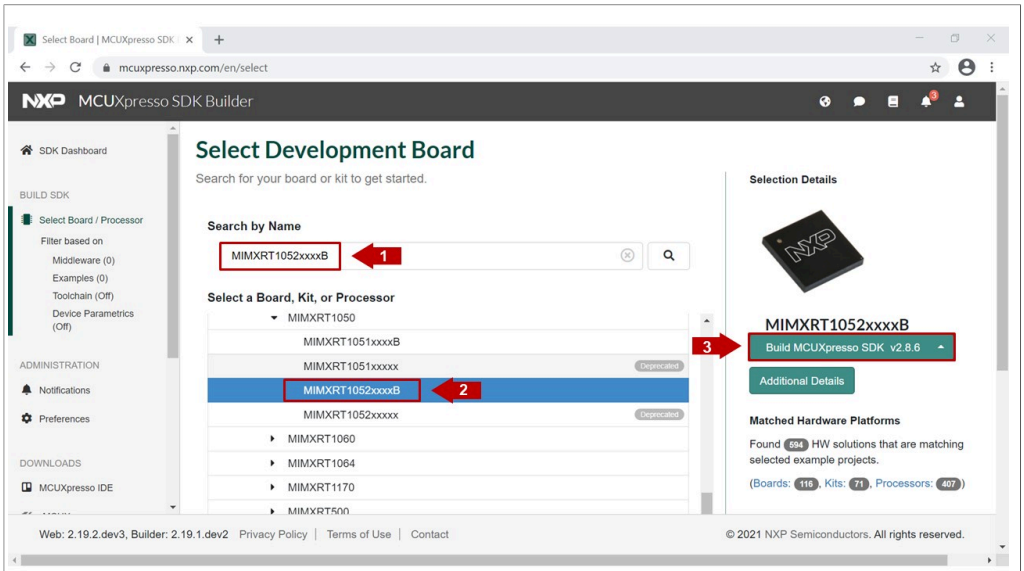

Figure 10. Select MIMXRT1052xxxxB development board

4. Choose *FreeRTOS*, click *Select All* and then click *Download SDK* button as seen in Figure 11:

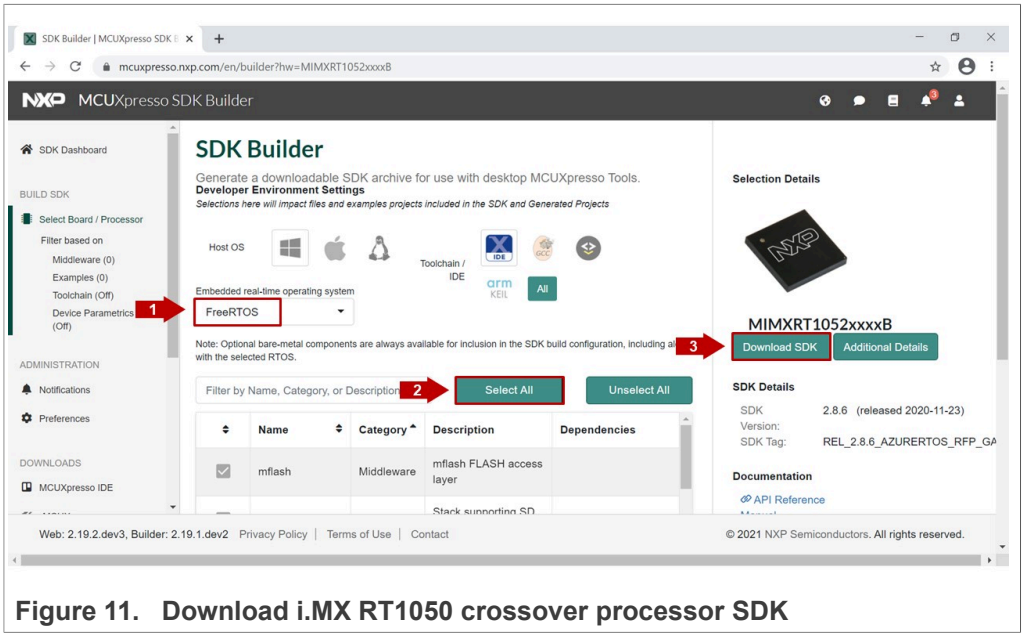

Figure 11. Download i.MX RT1050 crossover processor SDK

5. Accept the software terms and conditions as shown in Figure 12 and the download will start:

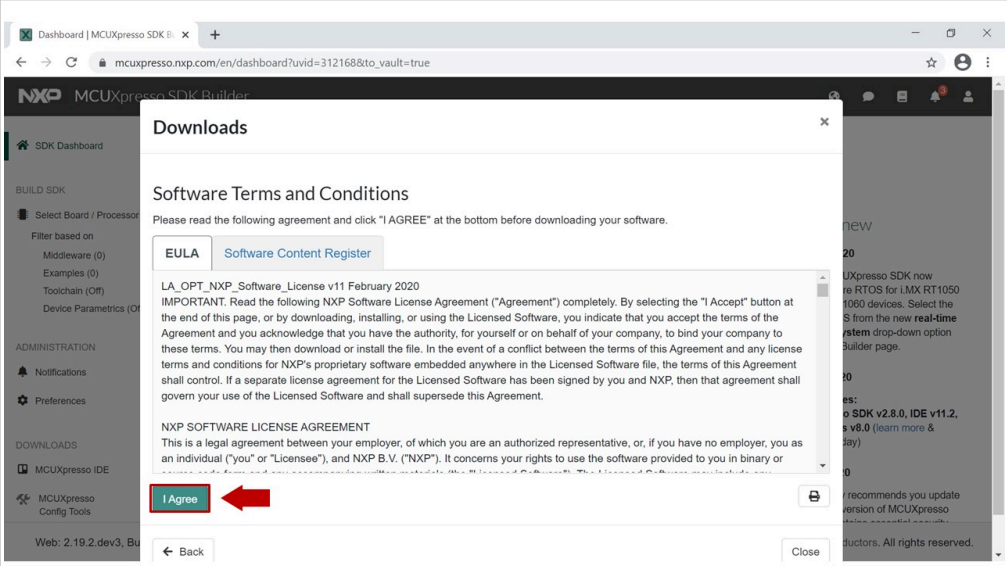

Figure 12. Accept i.MX RT1050 crossover processor SDK software terms and conditions

6. Open MCUXpresso, (1) select the ***Installed SDKs*** tab and (2) drag and drop the compressed SDK folder to its area. A warning window should pop up, (3) accept it as shown in Figure 13:

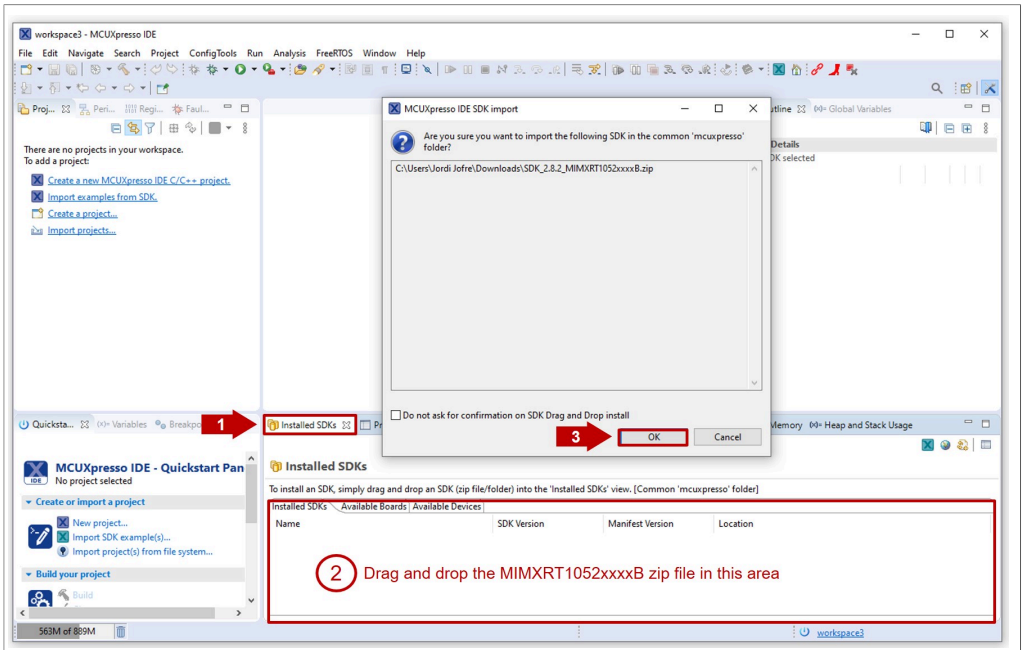

Figure 13. Drag and drop i.MX RT1050 crossover processor SDK

7. The i.MX RT1050 crossover processor SDK should now be listed in the ***Installed SDK*** tab like Figure 14.

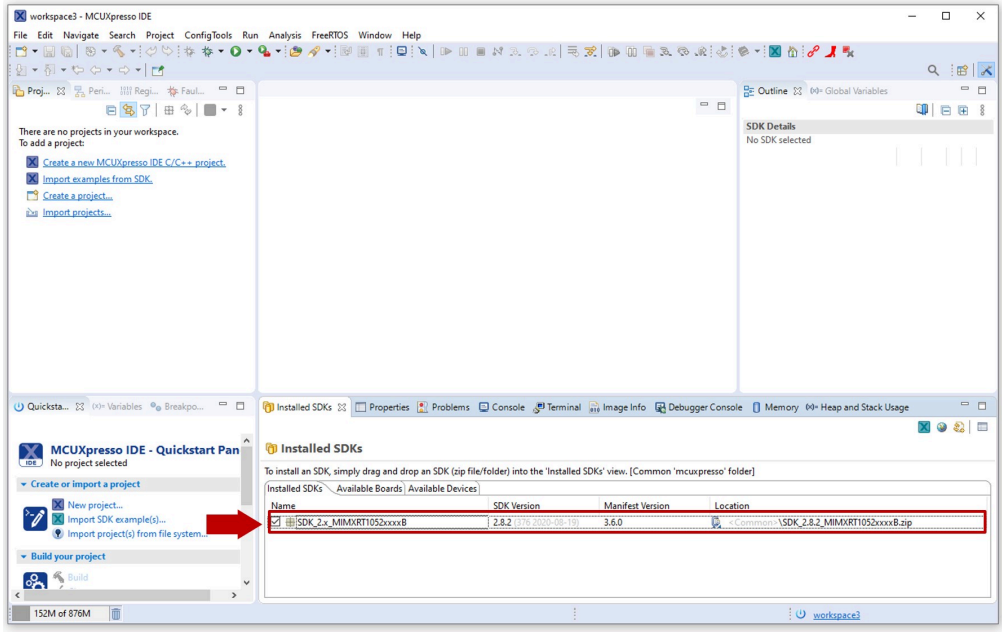

Figure 14. Installed i.MX RT1050 crossover processor SDK

4.3 Import Azure MCUXpresso example

The Azure IoT Hub demo application includes an MCUXpresso project example called `ICC_QMC_Motor_Control_App_Azure`, available from: www.nxp.com/quadmotorcontrol website in the section of **Tools and Software** with the name of ***MCUXpresso Project sample code with Azure***. Download it, and follow these steps to import it into your MCUXpresso workspace:

1. Make sure you have MCUXpresso installed in your laptop. If not, install it following the instructions in Appendix A: MCUXpresso installation
2. Open an MCUXpresso workspace.

3. Click ***Import project(s) from file system*** in the MCUXpresso IDE quick start panel as shown in Figure 15:

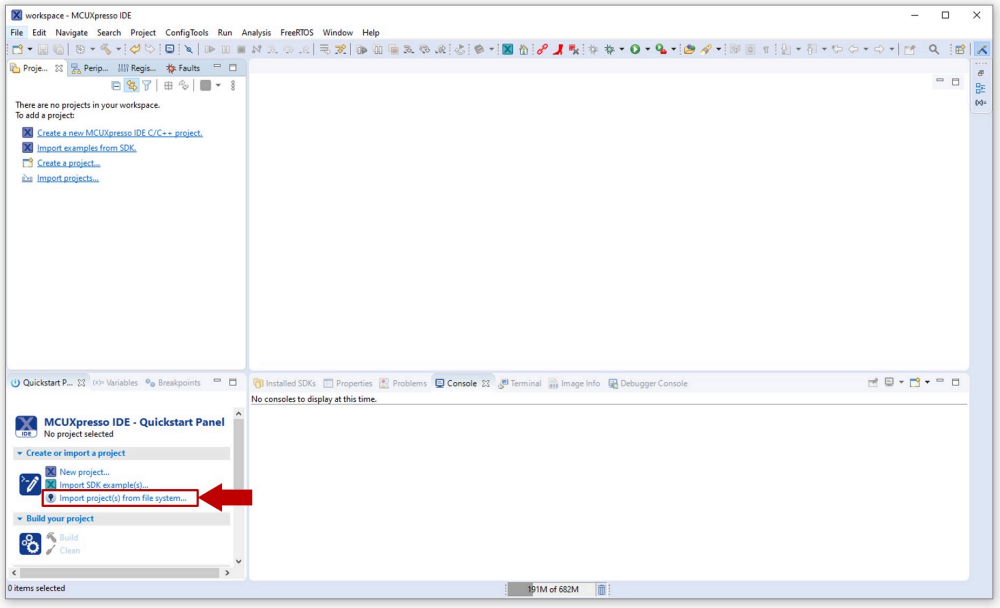

Figure 15. Import projects from file system

4. (1) Click the ***Browse*** button in project archive (zip) section, (2) navigate to the directory location where you previously downloaded the zip file from www.nxp.com/quadmotorcontrol website in the section of **Tools and Software** with the name of ***MCUXpresso Project sample code with Azure*** and select `ICC_QMC_Motor_Control_App_Azure` project, and (3) click ***Open***, as shown in Figure 16:

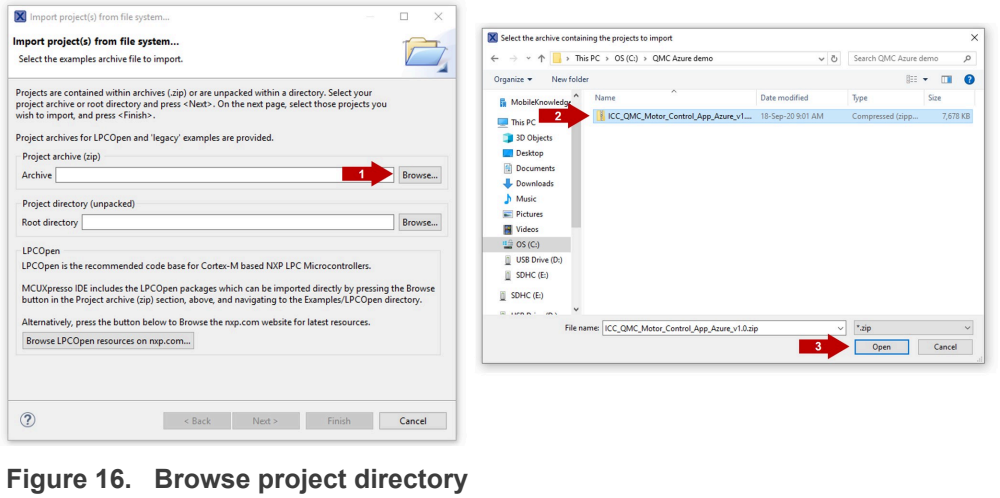

Figure 16. Browse project directory

5. Click ***Finish*** as shown in Figure 17:

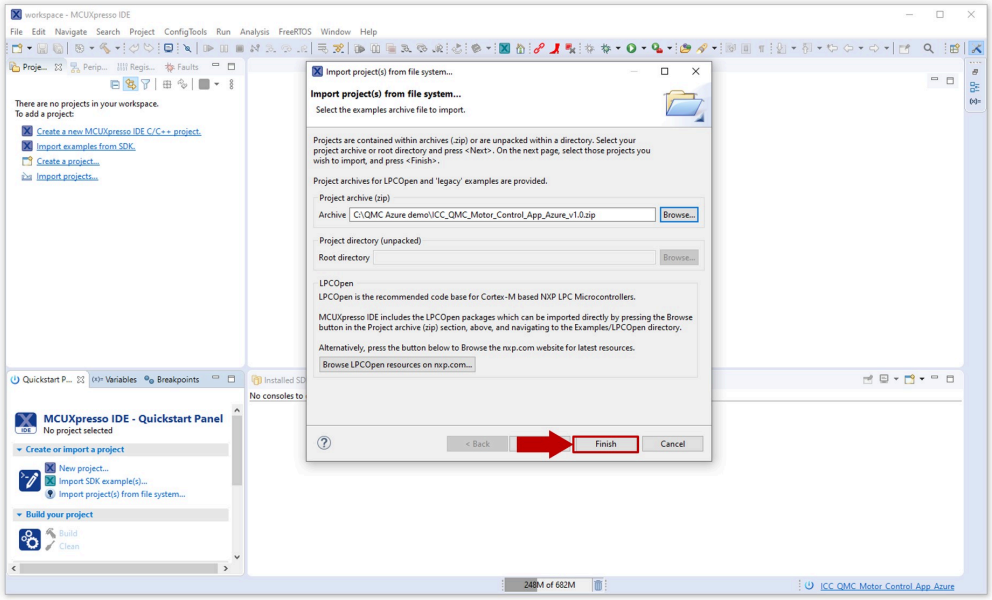

Figure 17. Import projects into workspace

6. The project should now be visible into your MCUXpresso workspace as shown in Figure 18:

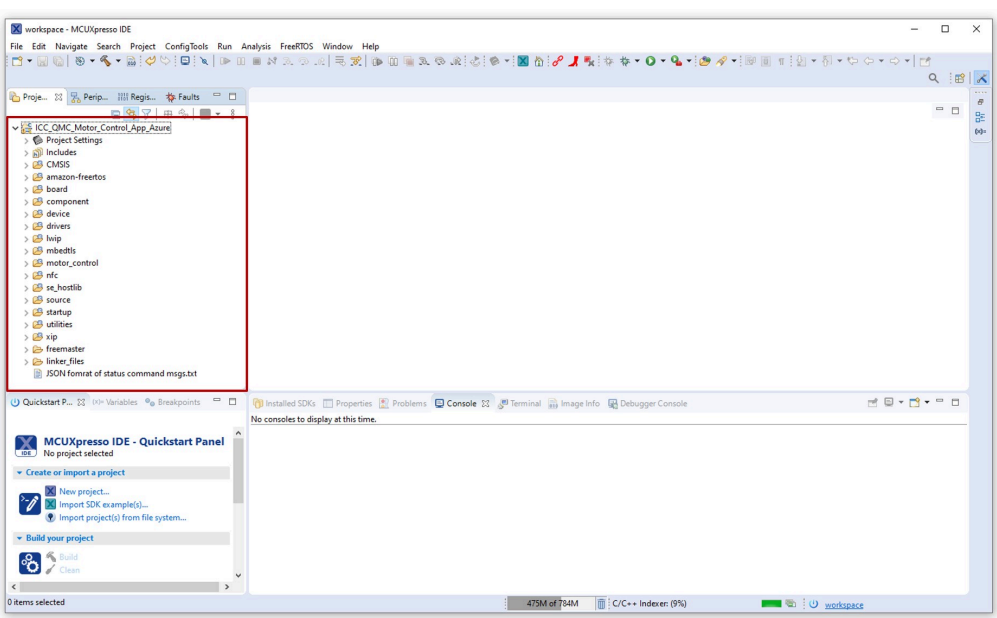

Figure 18. Project in the workspace

4.4 Flash VCOM binary

The VCOM software allows the NXP quad motor-control development platform to be used as a bridge between the laptop and the EdgeLock SE050. As such, it allows us

to inject credentials in the EdgeLock SE050 security IC by using the tools and scripts included in EdgeLock SE050 Plug and Trust middleware. To flash the VCOM software, follow these steps:

1. Select the project (1) and open the GUI Flash Tool (2) by clicking the icon in the top bar menu and selecting ***Connected***, as shown in Figure 19:

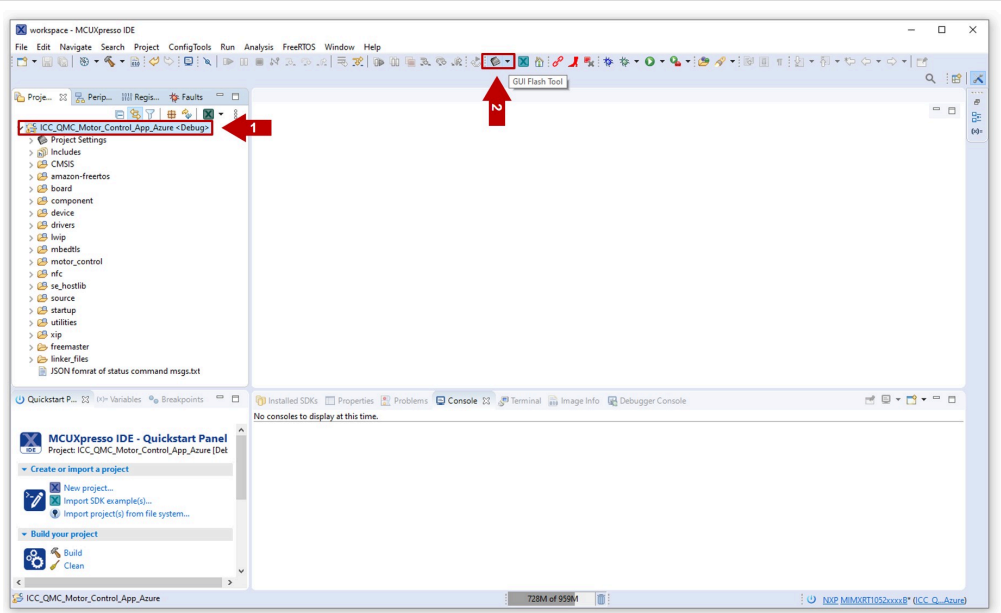

Figure 19. Open GUI Flash Tool

2. Make sure the LPC-Link2 debug probe is recognized by MCUXpresso (1) and click ***OK*** (2), as shown in Figure 20:

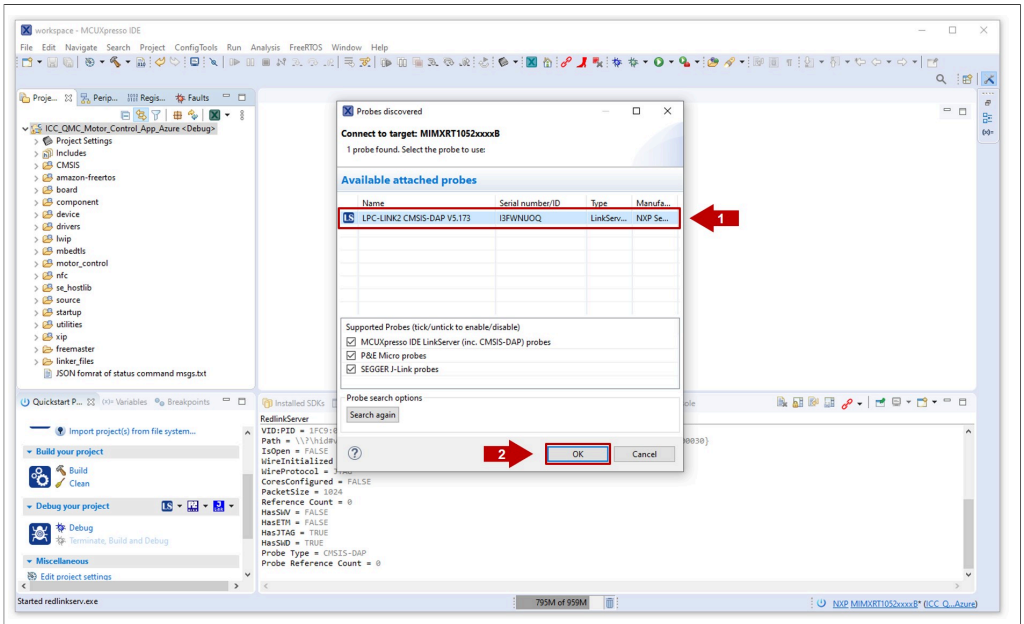

Figure 20. LPC-Link2 debug probe detection

3. From the GUI Flash Tool user interface, select the `se050_vcom_qmc.bin` binary from your file directory, as shown in Figure 21. This binary is also available for download from www.nxp.com/quadmotorcontrol.com webpage in the section of **Tools and Software** with the name of ***MCUXpresso Project sample code with Azure***.

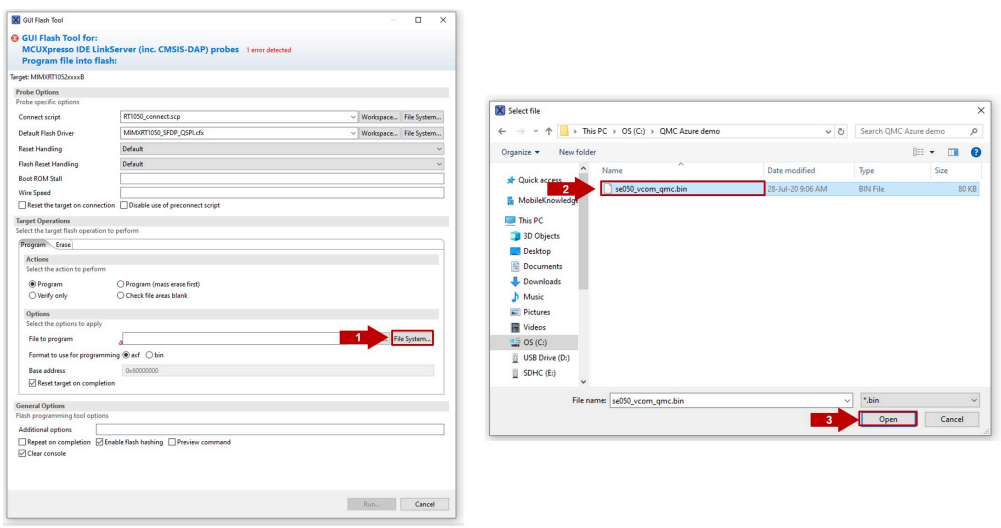

Figure 21. Select VCOM binary file

4. Now, flash the `se050_vcom_qmc.bin` binary by clicking the "Run" button, as shown in Figure 22:

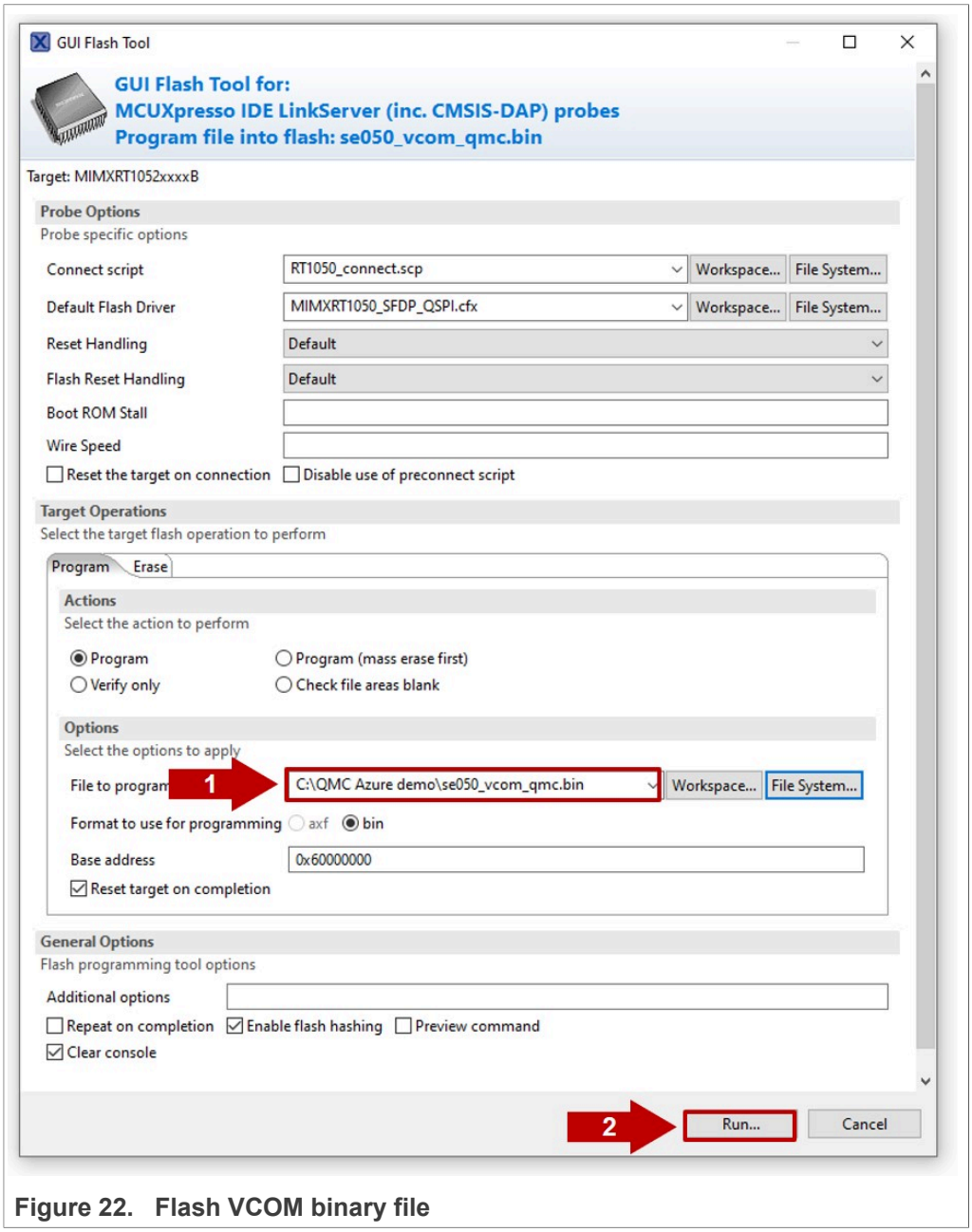

Figure 22. Flash VCOM binary file

5. The binary flashing program operation will be started. If it is completed successfully, it should return a success message like the one shown in Figure 23:

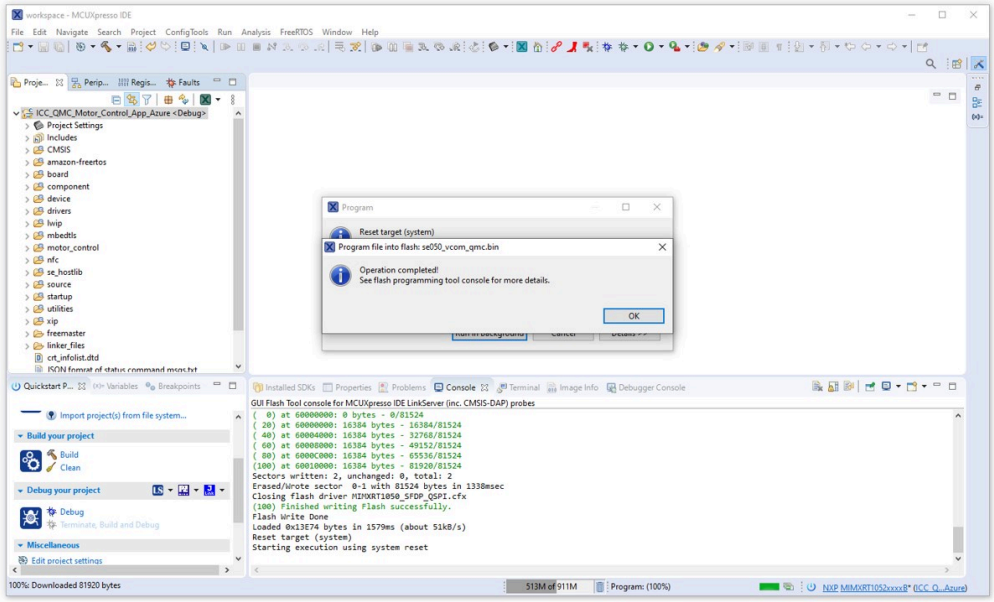

Figure 23. Flashing VCOM binary file completed

6. Disconnect the LPC-Link2 debugger and connect a USB cable from the computer to the USB connector on the smaller daughter board, as shown in Figure 24:

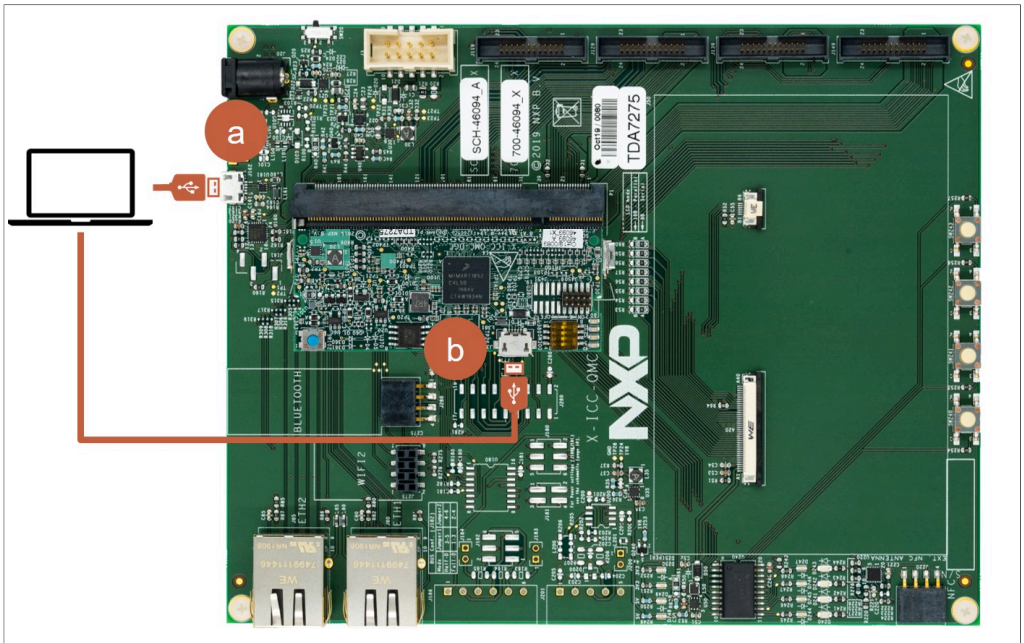

Figure 24. Connect daughter board to the laptop

7. The serial VCOM port should be recognized by your Device Manager, as shown in Figure 25.

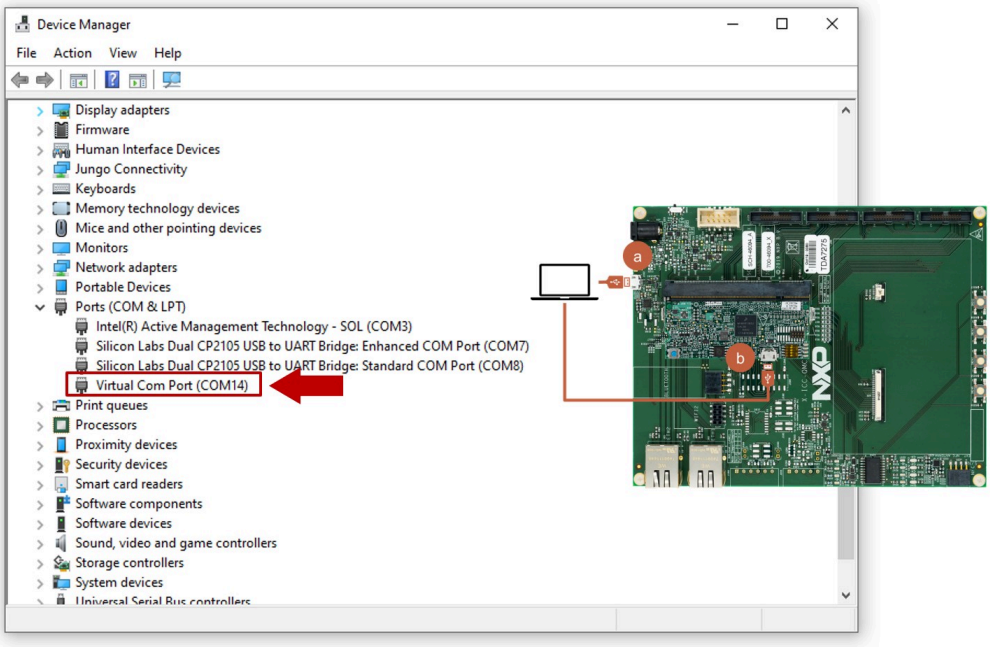

Figure 25. Connect daughter board to the laptop

Write down the VCOM port number as it will be used later (e.g. VCOM14 in Figure 24)

4.5 Download EdgeLock SE050 Plug and Trust middleware

The EdgeLock SE050 Plug and Trust middleware stack includes several project examples and support tools such as the `pycli` tool, which will be used in this document for key injection. To download EdgeLock SE050 Plug and Trust middleware:

1. Download "EdgeLock SE050 Plug and Trust middleware" from EdgeLock SE050 website under **Tools and Software** as shown in Figure 26.

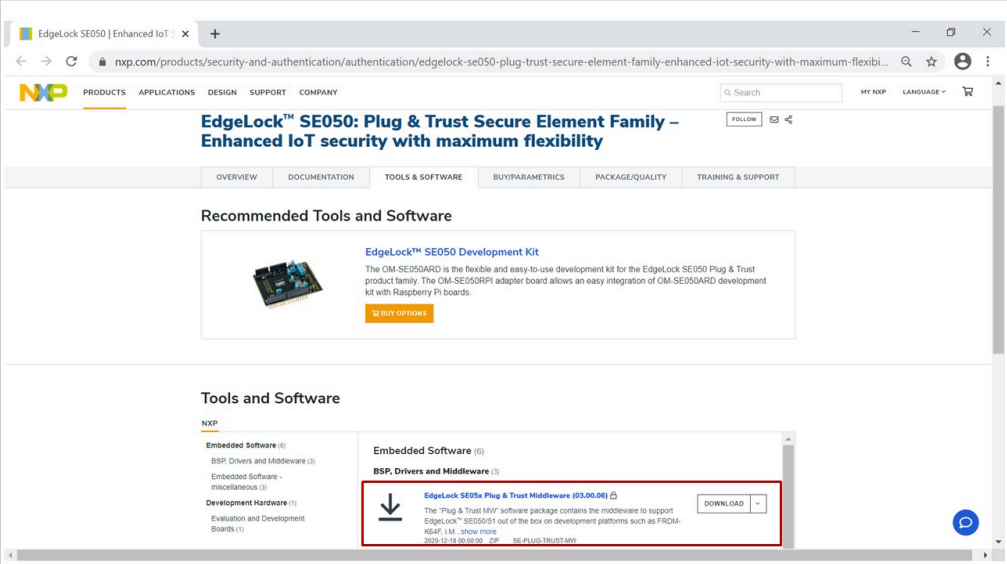

Figure 26. se050_middleware_download

2. Create a folder called ***se050_middleware*** in C: directory as shown in Figure 27:

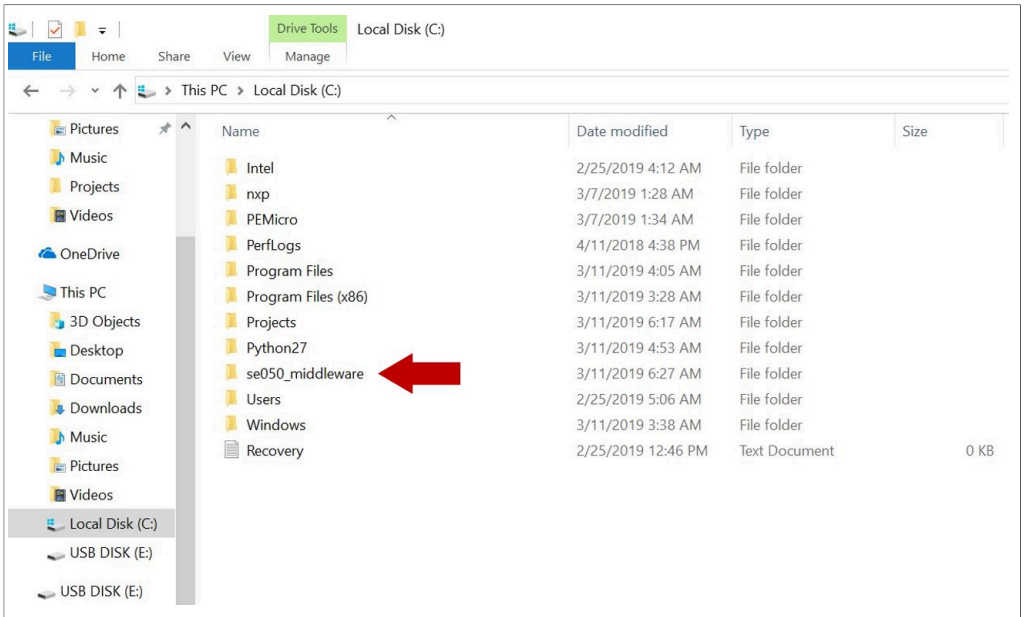

Figure 27. Create se050_middleware folder

3. Unzip SE050 middleware inside the ***se050_middleware*** folder. The unzipped package ***simw-top*** should look like as shown in Figure 28:

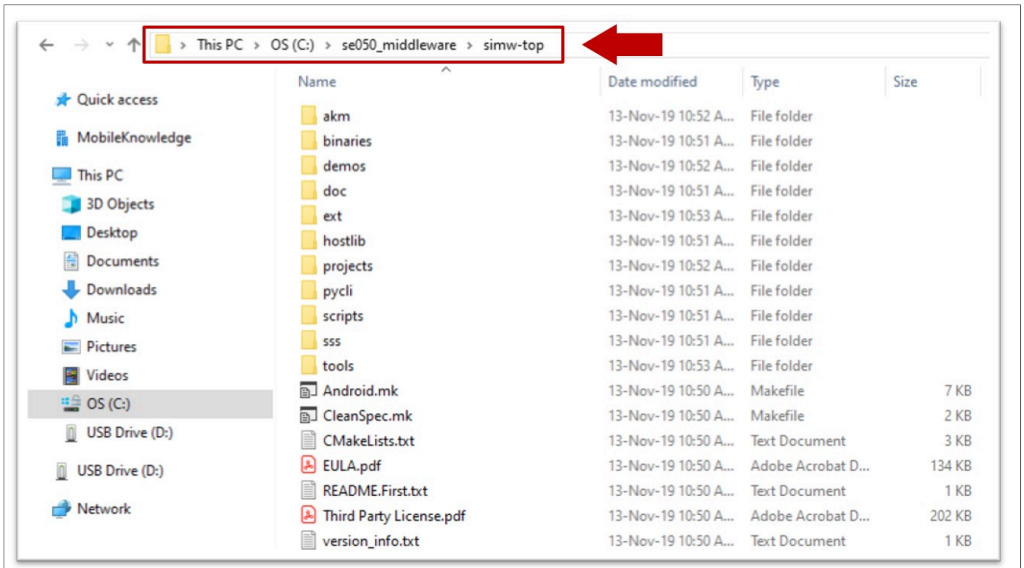

Figure 28. Unzip se050 middleware

Note: *It is recommended to keep* `simw-top` *with the* ***shortest*** *path possible and* ***without spaces*** *in it. This avoids some issues that could appear when building the middleware if the path contains spaces.*

4.6 Provision credentials using EdgeLock SE050 pycli tool

This section explains how to generate and inject your own credentials in EdgeLock SE050 using a provisioning script included as part of EdgeLock SE050 Plug and Trust middleware. This provisioning script is executed in a Python virtual environment. Creating a virtual environment venv (for Python 3) and virtualenv (for Python 2) allow you to manage separate package installations for different projects. They essentially allow you to create a "virtual" isolated Python installation and install packages into that virtual installation. When you switch projects, you can simply create a new virtual environment and not have to worry about breaking the packages installed in the other environments. It is always recommended to use a virtual environment while developing Python applications.

To create a new Python virtual environment and install EdgeLock SE050 pycli tool on it, follow these steps:

1. Install **Python 3.7 32-bit** version in your laptop, which can be downloaded from https://www.python.org/downloads/. For reference, Appendix B illustrates how Python 3.7.x 32-bit version can be installed, but the same procedure can be applied for more recent versions.
2. Open a command prompt

3. Go to `C:\se050_middleware\simw-top\pycli` folder:
(Figure 29) >> `cd C:\se050_middleware\simw-top\pycli`

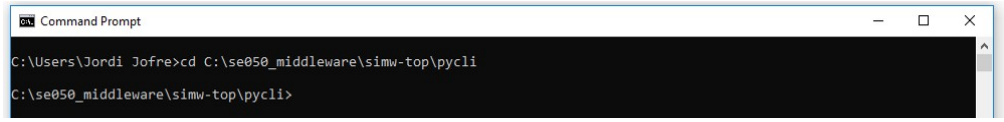

Figure 29. Go to pycli folder

4. Install Python virtual environment required package:
(Figure 30) >> `pip3 install virtualenv`

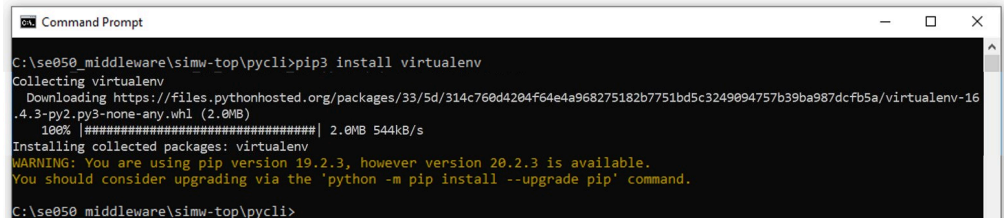

Figure 30. Install Python virtual environment required package

5. Create a new virtual environment:
>> `python -m virtualenv venv`
A new `venv` folder should have been created, as shown in Figure 31

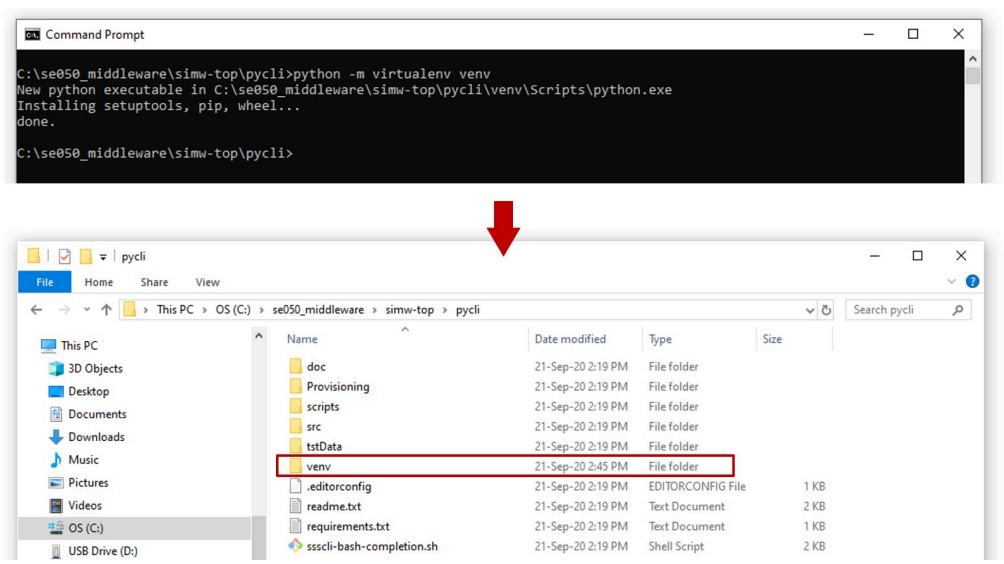

Figure 31. Create virtual environment venv

6. Activate the newly created virtual environment (Figure 32) >> `call venv\Scripts\activate.bat`

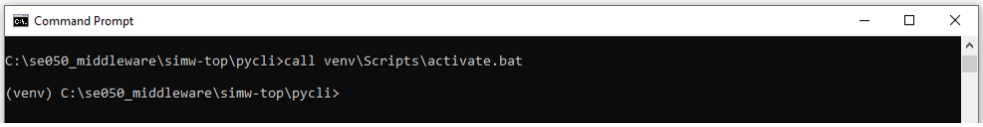

Figure 32. Activate venv

7. In the newly installed virtual environment, install required packages: (Figure 33) >> `pip install -r requirements.txt`

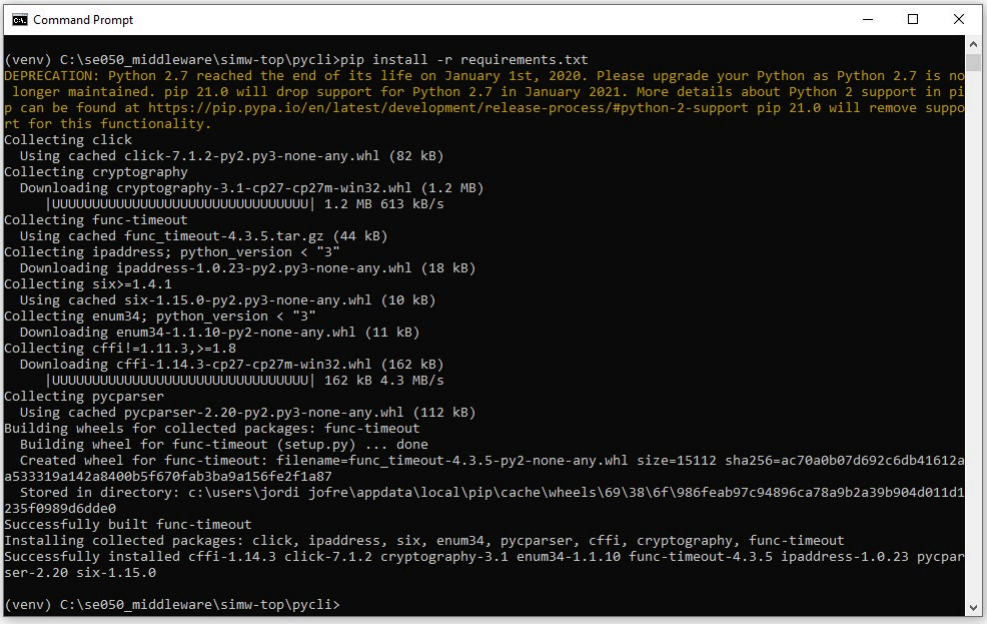

Figure 33. Install venv requirements

8. Go to `simw-top\pycli\src` folder and install the pycli tool, as shown in Figure 34:
   `>> cd src`
   `>> pip install --editable .`

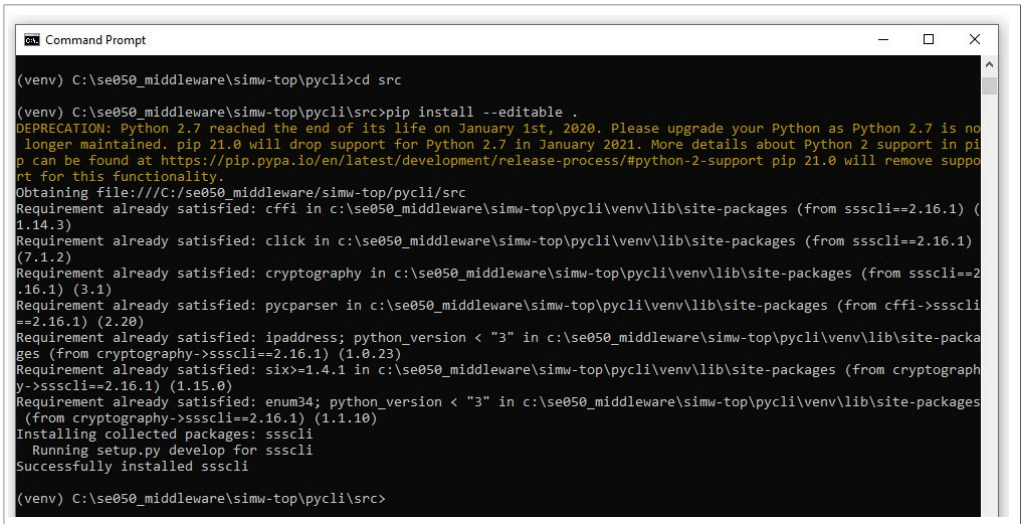

Figure 34. Install pycli

9. Go to `simw-top\pycli\Provisioning` folder, where you can find a set of scripts to generate and inject credentials in EdgeLock SE050, as shown in Figure 35:

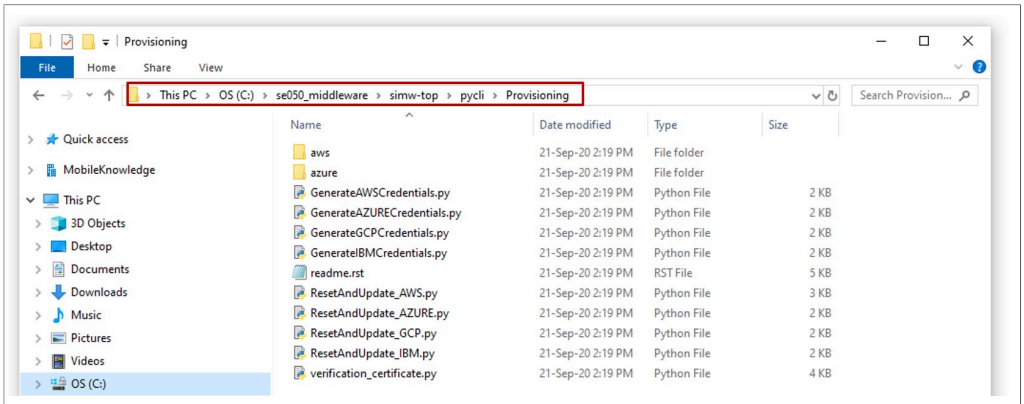

Figure 35. Provisioning scripts folder

10. Execute the `GenerateAZURECredentials.py` script. This script generates the necessary keys and certificates in your local machine.
(Figure 36) >> `python GenerateAZURECredentials.py <COM_NUMBER>`, where `<COM_NUMBER>` is the number assigned in your device manager:

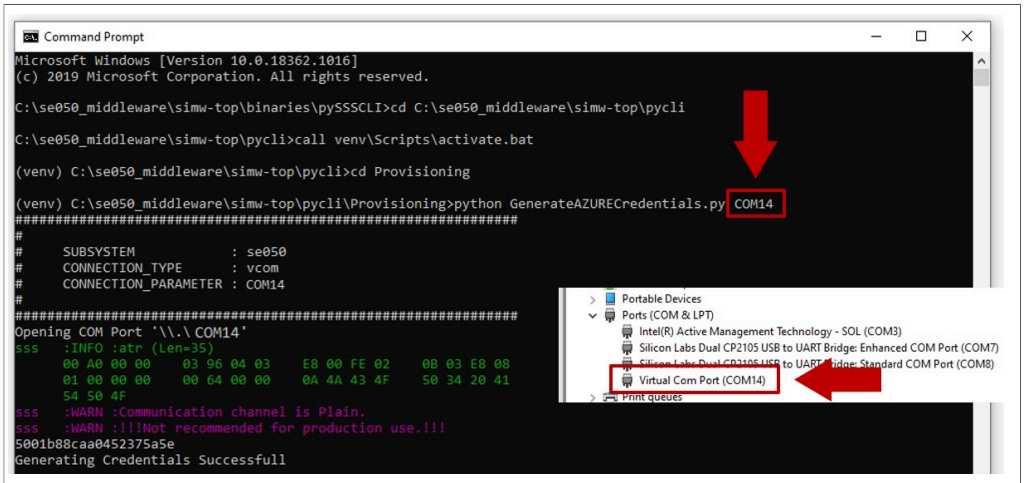

Figure 36. Execute GenerateAZURECredentials.py script

11. Make sure that a set of sample credentials have been generated inside `simw-top\pycli\Provisioning\azure` folder, as shown in Figure 37:

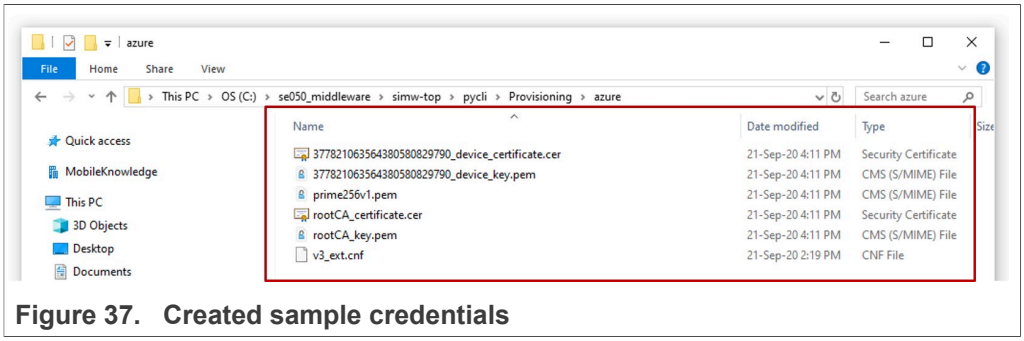

Figure 37. Created sample credentials

12. Execute the `ResetAndUpdate_AZURE.py` script. This script will inject the keys inside the EdgeLock SE050.
(Figure 38) >> `python ResetAndUpdate_AZURE.py <COM_NUMBER>`, where `<COM_NUMBER>` is the number assigned in your device manager:

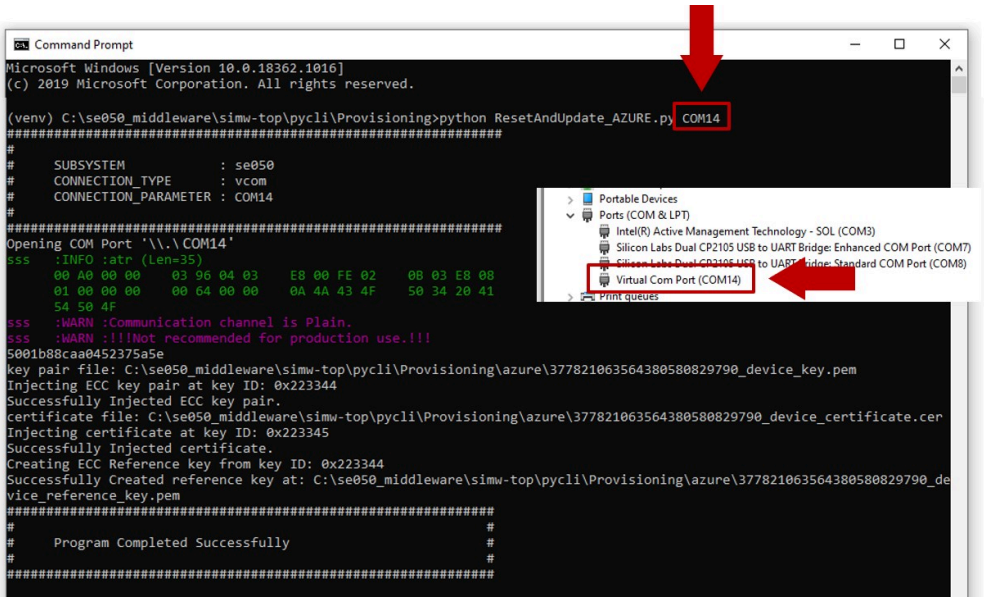

Figure 38. Execute ResetAndUpdate_AZURE.py script

5 Azure IoT Hub account setup

This section details the steps related to the Azure IoT Hub configuration.

5.1 Create Azure subscription

Microsoft Azure offers a free 30-day trial period to all new account holders. If you already have an account, you camp jump to Section 5.2. Otherwise, follow these steps to create a free account in Azure:

1. Go to https://www.azure.com and click the green ***Free acount*** button as shown in Figure 39:

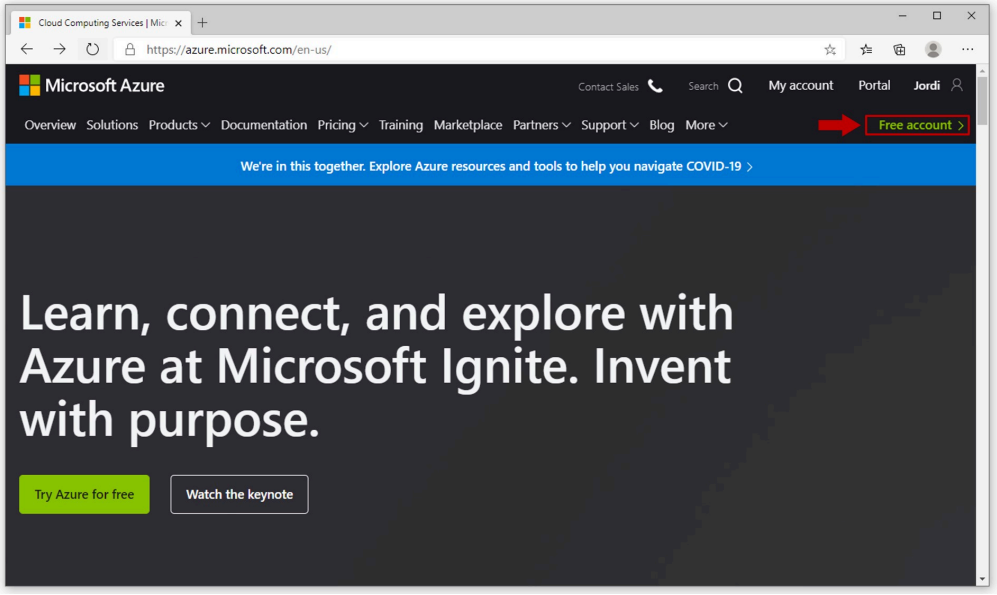

Figure 39. Azure subscription creation

2. If you already have an account with MicrosoftOffice 365, you will be prompted to log in. When you log in, you will be asked to perform an identity validation using your mobile phone as shown in Figure 40:

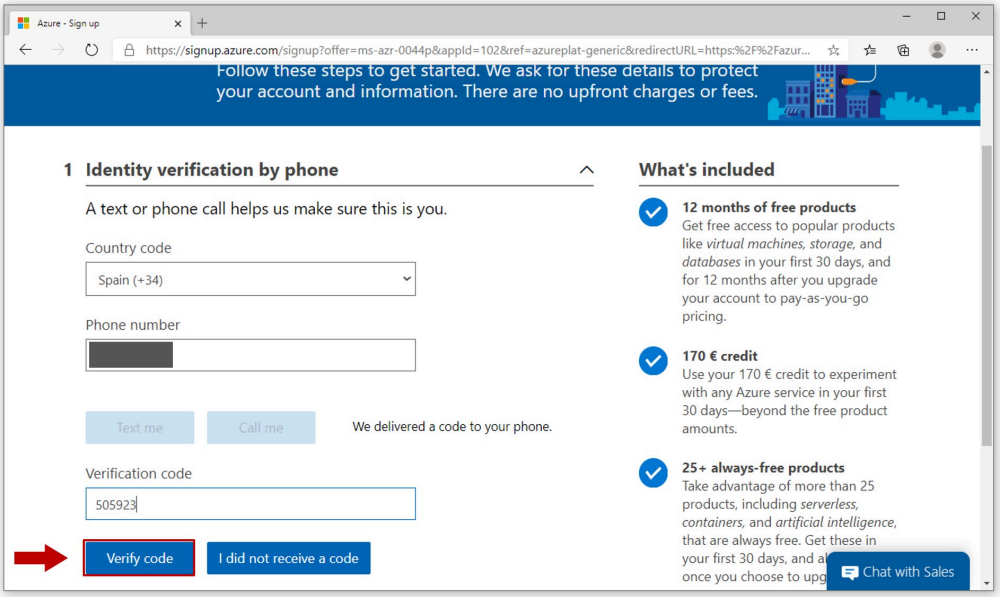

Figure 40. Azure free account sign up

3. Then, introduce your credit card details in the form shown in Figure 41. Microsoft will use it to verify your identity. There is no charge involved with the setting up of a trial account. However, there might be a record for a $0 transaction on your bank statement.

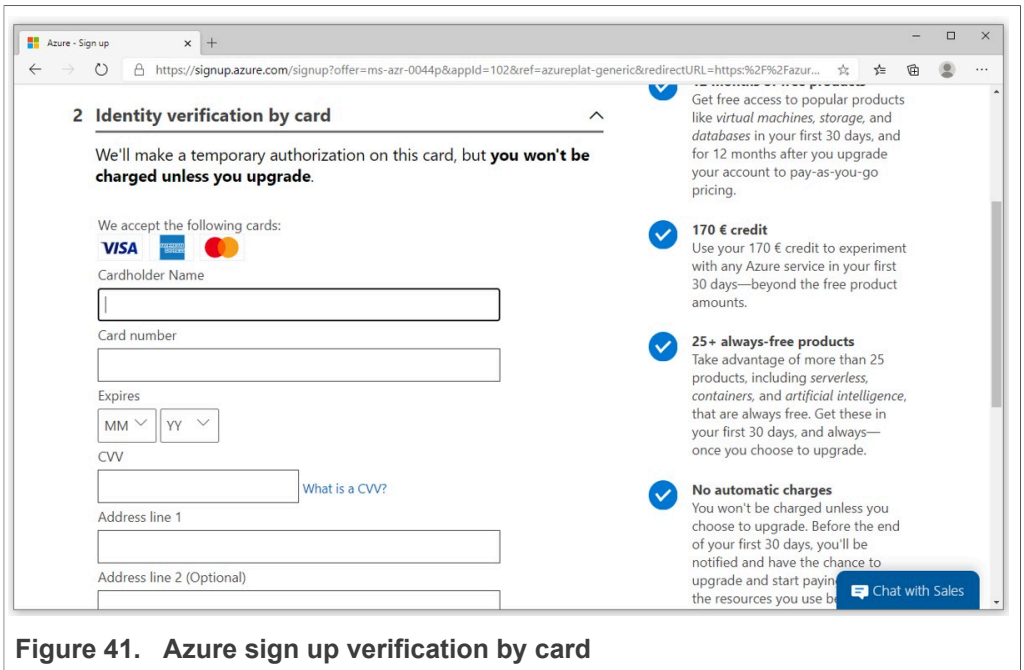

Figure 41. Azure sign up verification by card

4. Check the ***I agree*** checkbox and click ***Sign Up*** as shown in Figure 42. In a few seconds, your account will be ready.

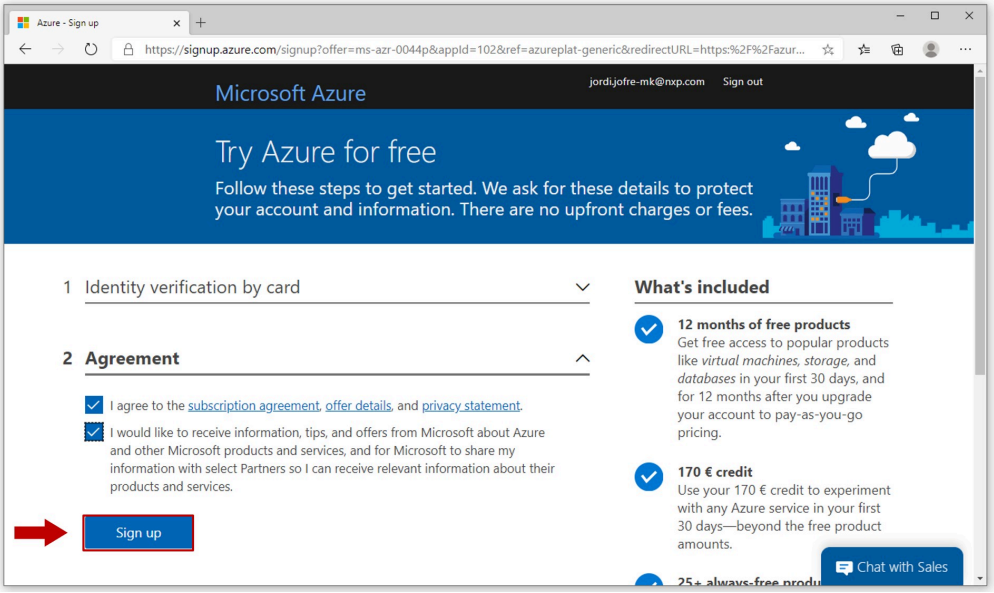

Figure 42. Azure sign up agreement acceptance

5. Your Microsoft Azure account is now created.

5.2 Create an Azure IoT Hub

This section describes how to create an Azure IoT Hub instance using the Azure portal. The Azure IoT Hub is a central message hub for secure bidirectional communication between the cloud-hosted application and the IoT devices.

To use the steps described in this section, you need an Azure subscription. If you do not have an Azure subscription yet, check Section 5.1 to create a free account. If you already have an Azure subscription:

1. Log in the Azure portal. In the Azure dashboard and click on ***Create a resource*** as shown in Figure 43:

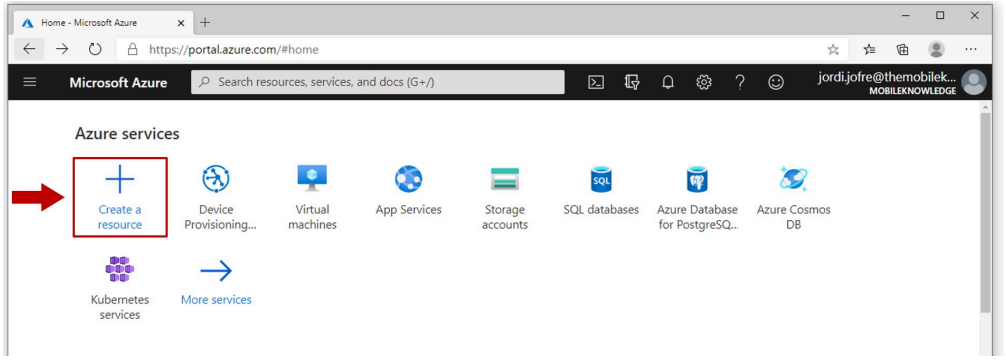

Figure 43. Create a resource

In the search bar, type *IoT Hub* (1), then click on the *IoT Hub* (2) resource, and finally click on *Create* button as shown in Figure 44:

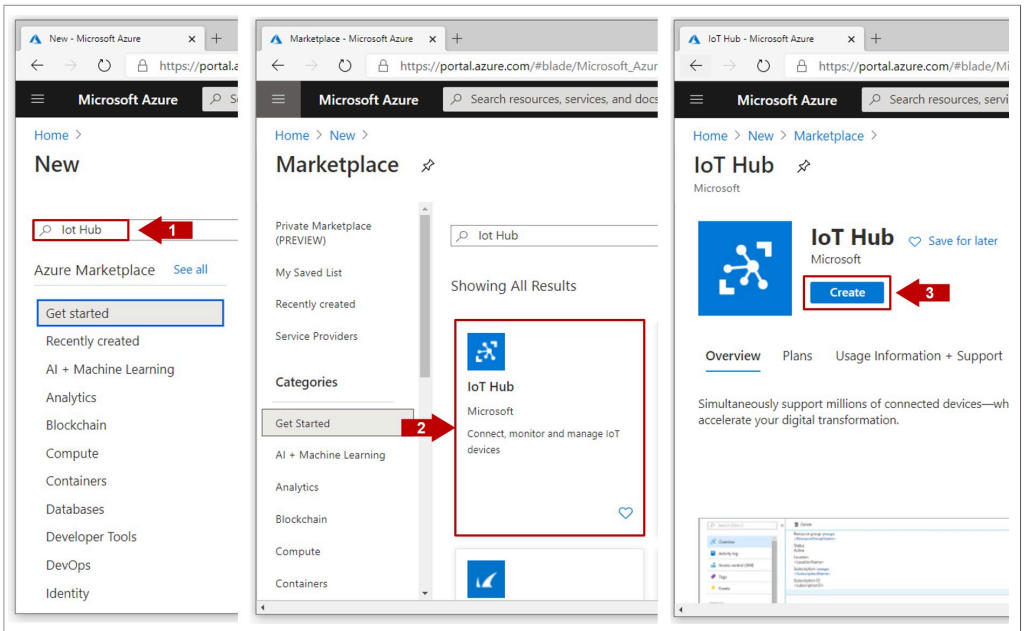

Figure 44. Create an Azure IoT Hub

2. In the ***Basics*** tab, fill in the fields as shown in Figure 45:
   a. **Subscription**: The subscription to use for your Azure IoT Hub. Select Free trial to use the free acount created in Section 5.1
   b. **Resource group**: A resource group is a container that holds related resources for an Azure solution. To create a new one, click ***Create new*** and fill in the name for your group.
   c. **Region**: Select the region in which you want your hub to be located.
   d. **IoT Hub name**: Write the name for your Azure IoT Hub. Note that this name must be globally unique.
   e. Click ***Next: Networking***

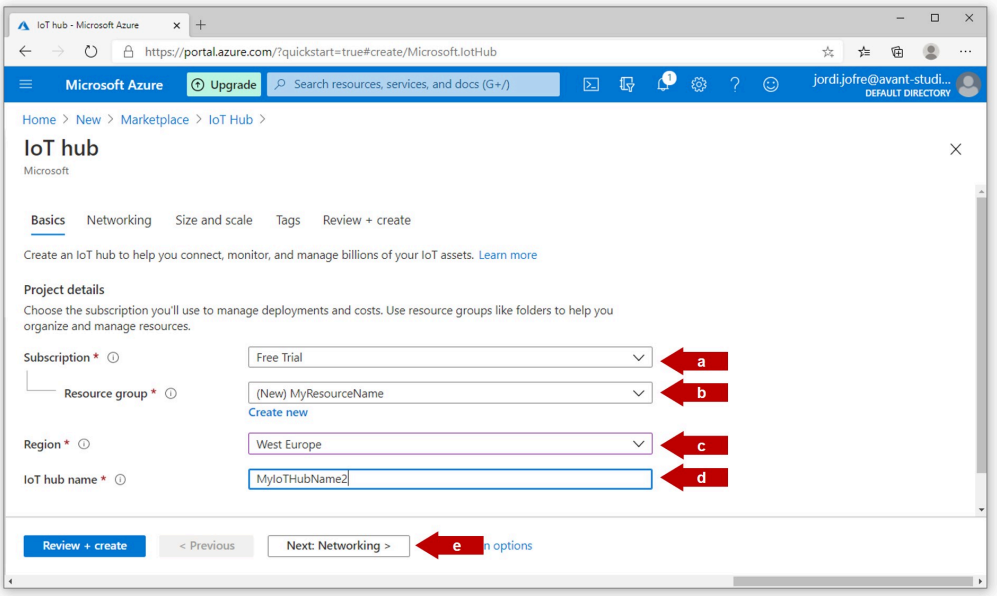

Figure 45. Create an Azure IoT Hub - Basics form

3. On the ***Networking*** tab, leave the Public endpoint (all networks) option selected and click Next:Size and scale, as shown in Figure 46:

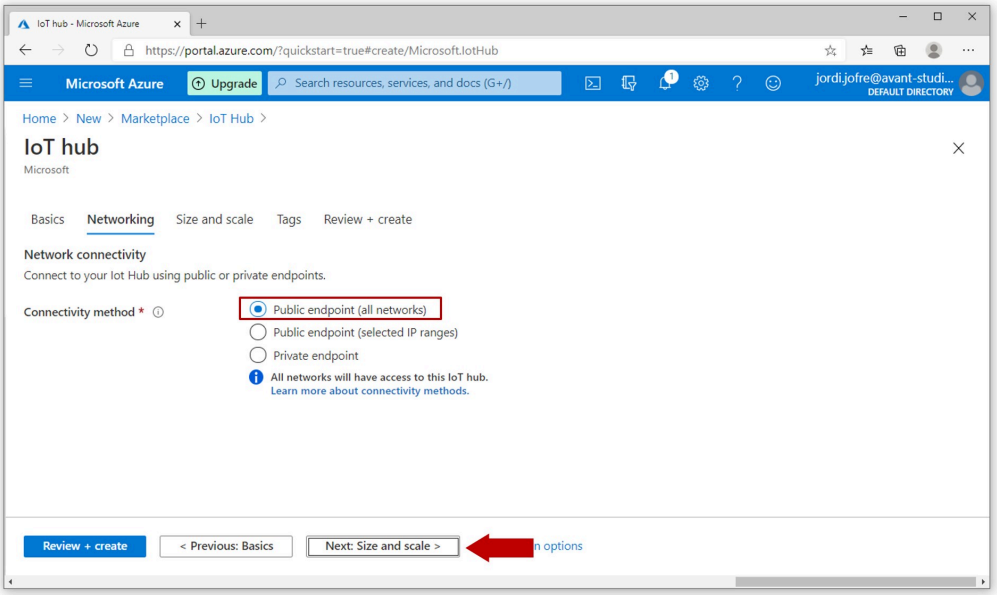

Figure 46. Create an Azure IoT Hub - Networking form

4. On the ***Size and scale*** tab, do as shown in Figure 47:
   a. Select **F1: Free tier**
   b. Click ***Review+create*** at the bottom

   The **F1: Free tier** is meant for testing and evaluation. It has all the capabilities of the standard tier, but limited messaging allowances.

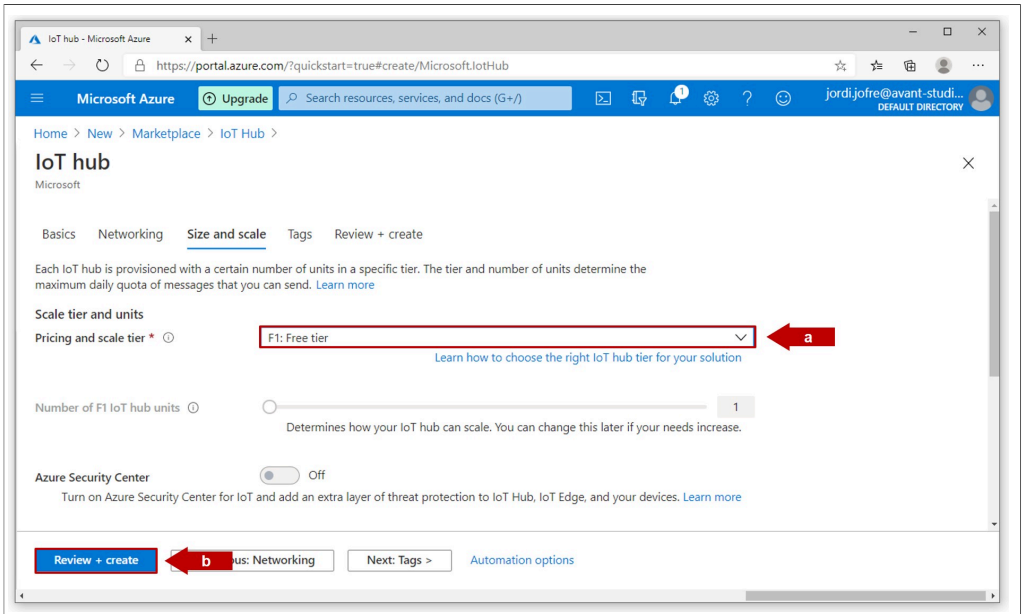

Figure 47. Create an Azure IoT Hub - Size and scale form

5. On the ***Review+create*** tab, click **Create** to confirm the Azure IoT Hub creation as shown in Figure 48. You might need to wait a few minutes until your deployment is ready:

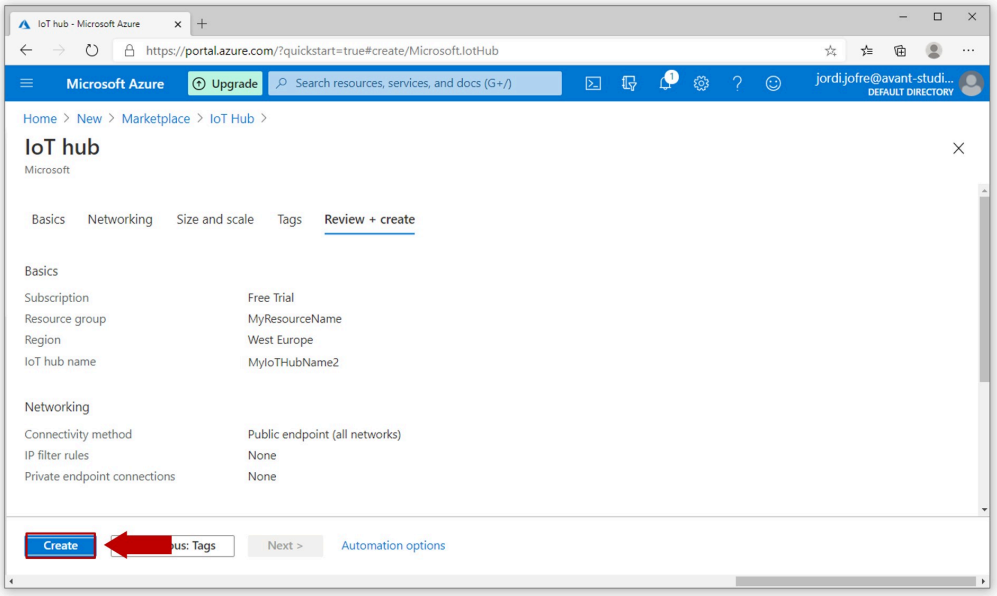

Figure 48. Create an Azure IoT Hub - Review and create form

6. When the Azure IoT Hub instance is deployed, you can see the deployment details and go to the recently created Azure IoT Hub resource as shown in Figure 49:

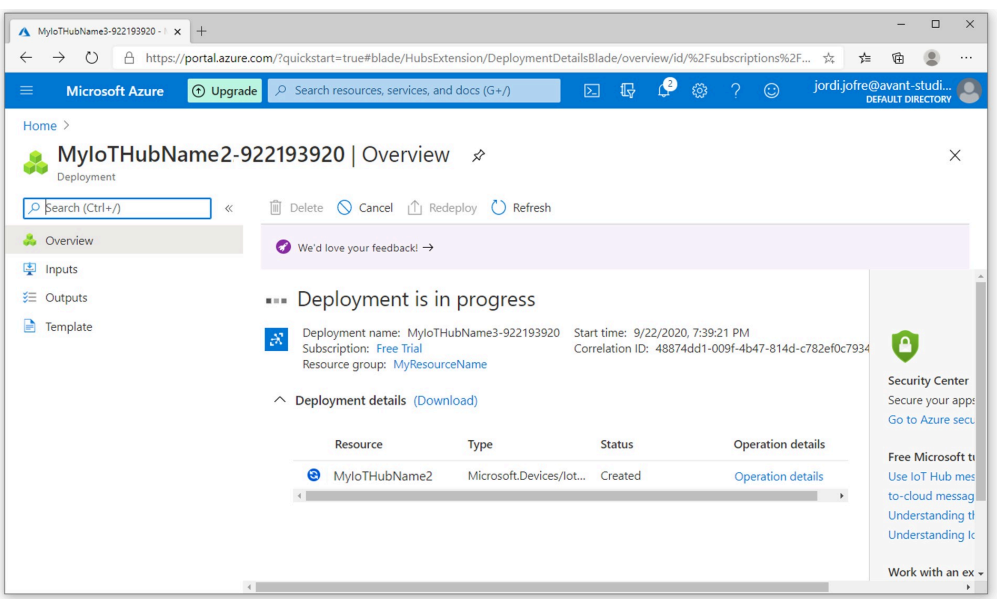

Figure 49. Create an Azure IoT Hub completition

5.3 Create a device

This section describes how to create a device instance using the Azure IoT Hub portal. Follow these steps:

1. In your Azure IoT Hub resource, do as shown in Figure 50:
   a. Go to ***IoT devices*** on the left hand side menu.
   b. Click ***New*** to create a new device.

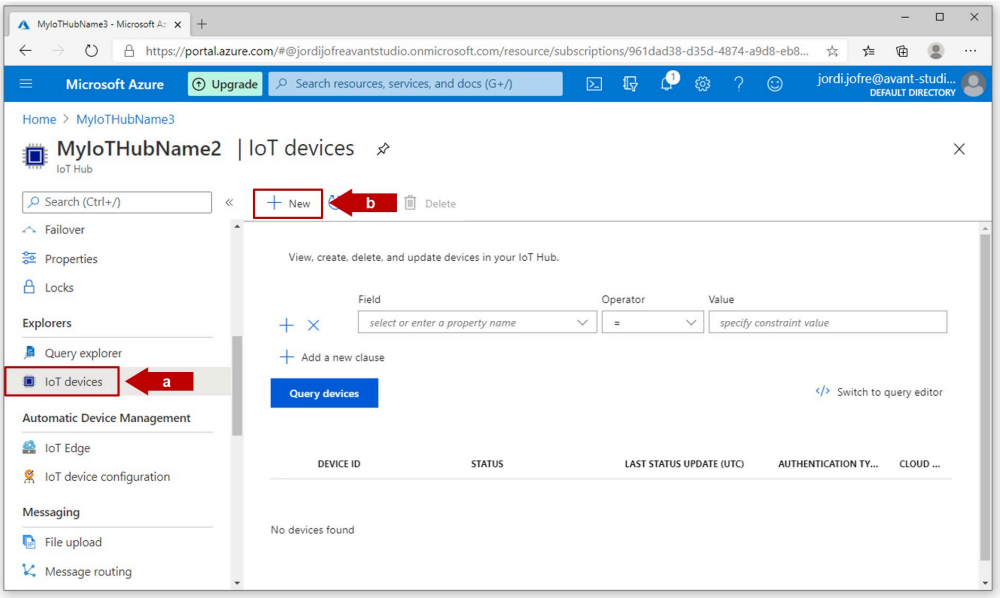

Figure 50. Create a device

2. To create a device, follow the indications in
   a. As device ID, copy the 24 digits generated in the certificate credential in Section 4.6
   b. Select **X.509 CA Signed** as authentication type.
   c. Click ***Save***.

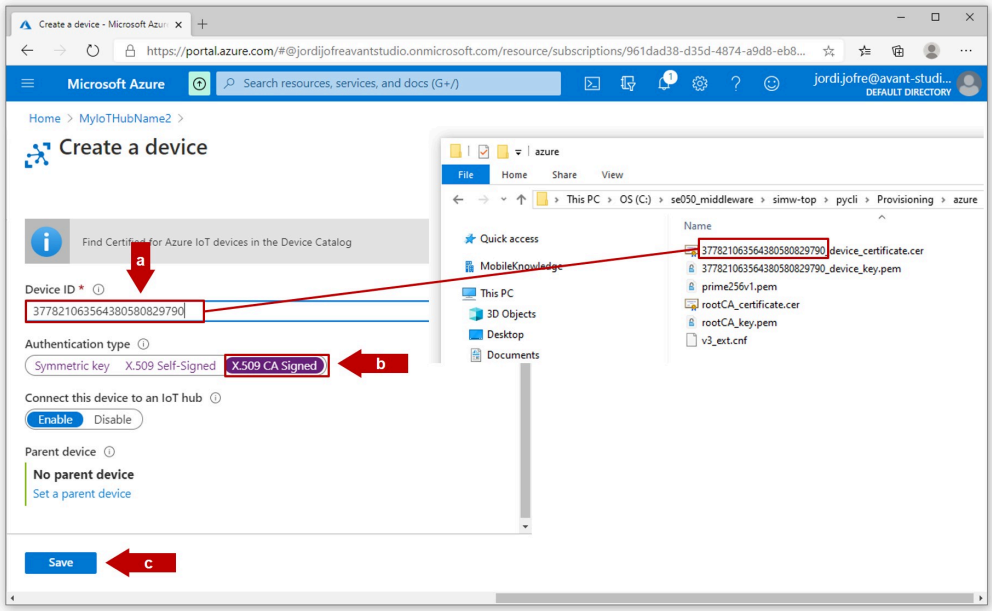

Figure 51. Create a device II

5.4 Upload root CA certificate to your Azure IoT Hub

Azure IoT Hub supports three attestation mechanisms for confirming the device authenticity during registration based on:

- **X.509 certificates**
- Trusted Platform Module (TPM)
- Symmetric key

This document describes how to use X.509 CA certificates to authenticate devices connecting to the Azure IoT Hub. The other two attestation mechanisms are beyond the scope of this application note. To upload a new root CA certificate:

1. In your Azure IoT Hub resource, click in ***Certificates*** menu (1) and then click ***Add*** (2) as shown in Figure 52:

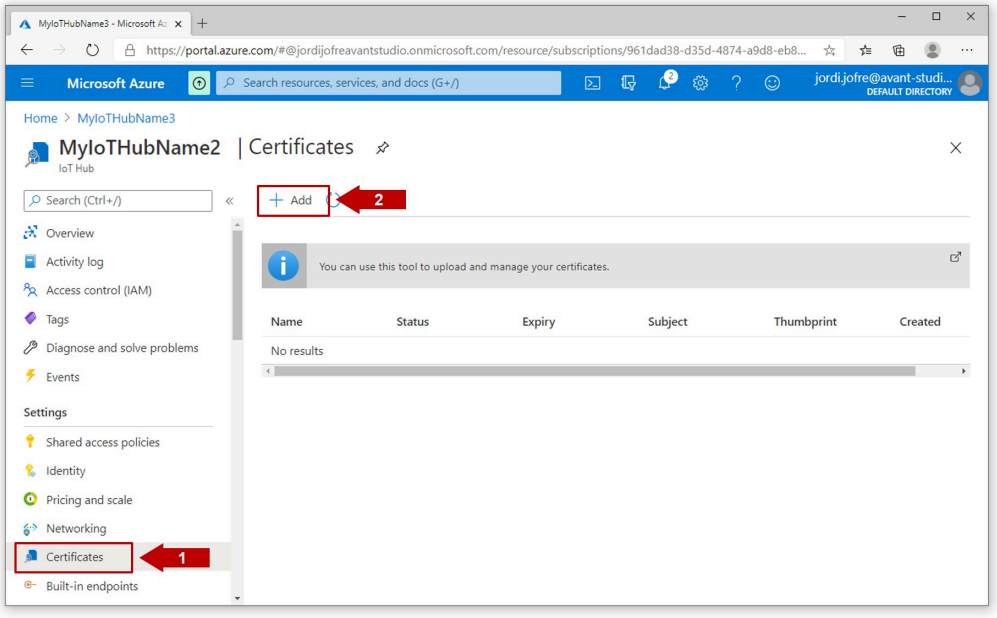

Figure 52. Upload a root CA certificate

Then, type the certificate name and select the certificate to upload, as shown in:

a. As ***Certificate Name***, for instance, use *rootCA_QMC*.

b. Upload the rootCA certificate that we generated in Section 4.6

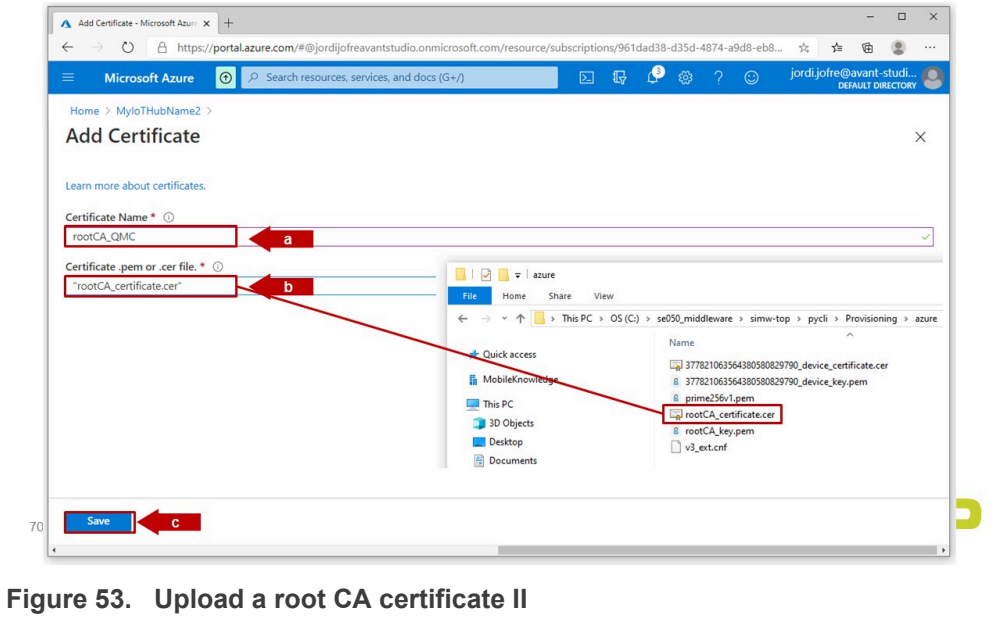

Figure 53. Upload a root CA certificate II

2. The certificate we have just uploaded will appear in the list as ***Unverified*** as shown in Figure 54.
   The certificate status remains in **Unverified** until we complete the *proof of possession* validation. The *proof of posession* mechanim verifies that the uploader is in possession of the certificate private key. For this verification, the Azure IoT Hub generates a *verification code* that needs to be included in a *verification certificate* signed by the CA certificate private key.

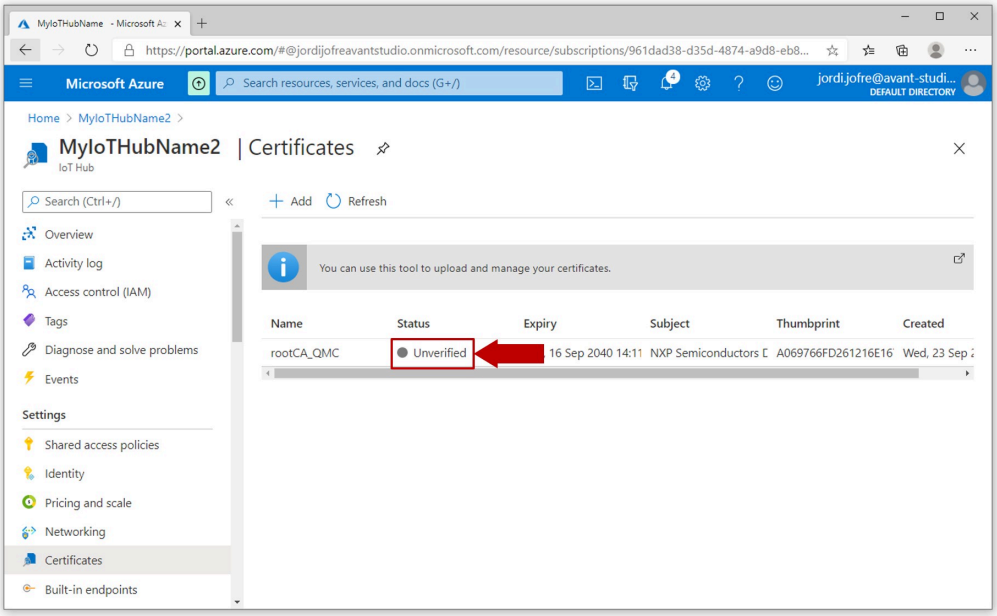

Figure 54. Root CA certificate unverified status

3. To obtain the verification code, do as shown in Figure 55
   a. Click on the certificate uploaded in the previous step
   b. Click ***Generate Verification code***
   c. Copy and save the generated ***verification code***. This code will be used to conduct the proof of verification and certificate ownership.

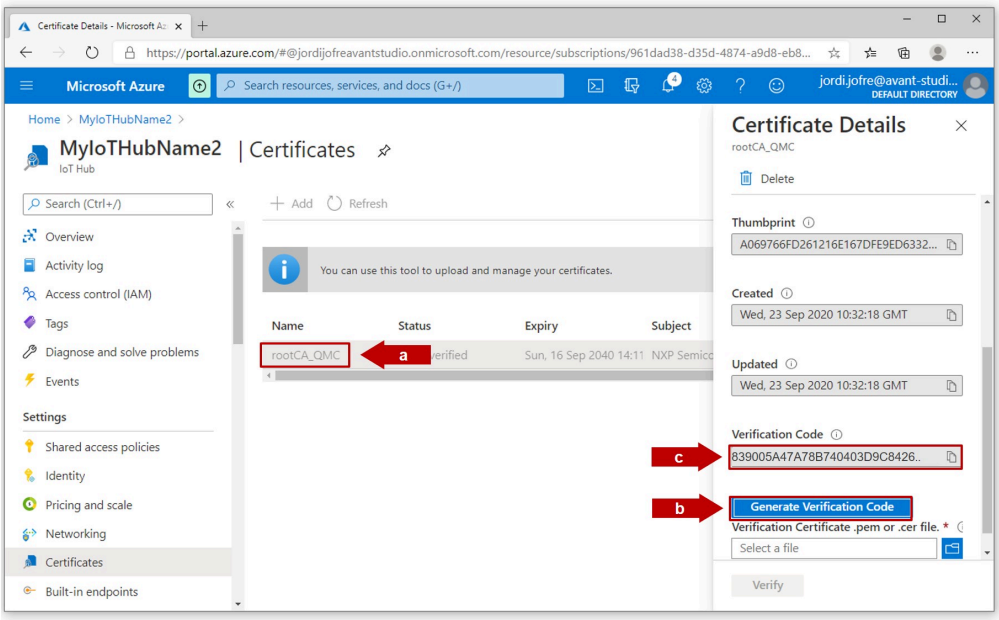

Figure 55. Obtain root CA certificate verification code

4. We use the pycli tool again to generate the verification certificate. If you have the command prompt still open, go to the next step. If you have already closed it, open one, go to `simw-top\pycli\` folder as shown in Figure 56, and type the following commands:

```
>> call venv\Scripts\activate.bat
>> cd Provisioning
```

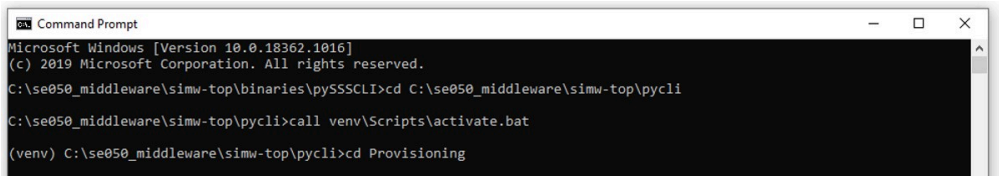

Figure 56. Open virtual environment

5. Execute the `verification_certificate.py` script, which require three arguments:
<RootCA_certificate>
<RootCA_private key>
<Azure_verification_code>
The first two correspond to credentials injected in Section 4.6 and the third, is the verification code we have just obtained from Azure. Figure 57 is an example of the command to be sent:

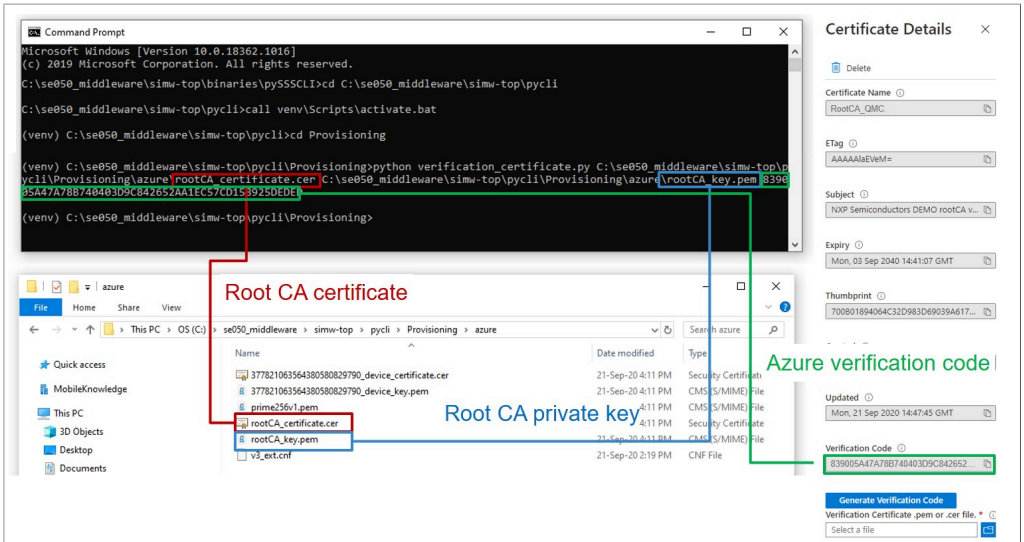

Figure 57. Generate verification certificate

6. Make sure a file called `verifyCert.cer` has been generated in `simw-top\pycli\Provisioning` folder, as shown in Figure 58:

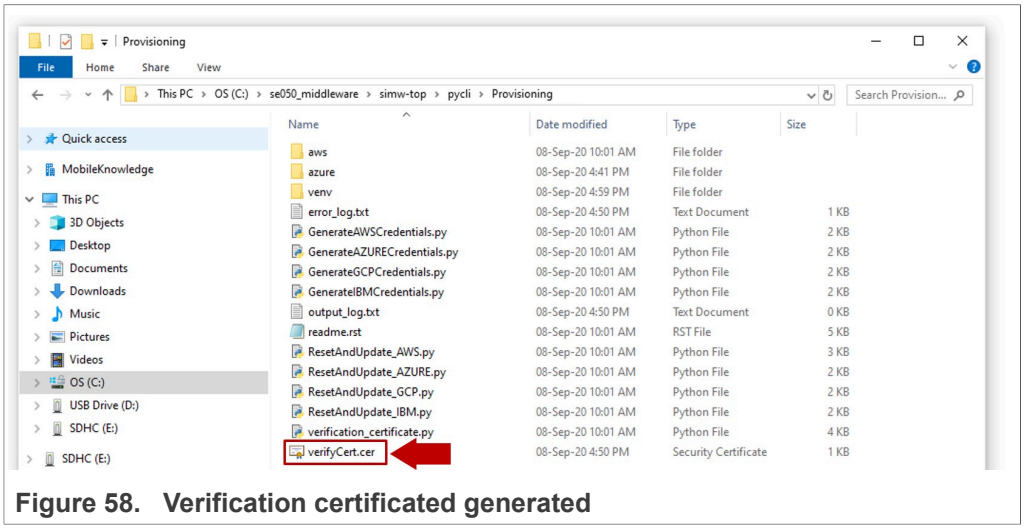

Figure 58. Verification certificated generated

5.5 Verify root CA certificate

To change the root CA certificate status from ***Unverified*** to ***Verified***:

1. In the Azure IoT Hub portal, do as shown in Figure 59:
   a. Click on ***verification certificate .pem or .cer file*** button
   b. Upload ***verifyCert.cer***, which is the signed verification certificate generated in Section 5.4
   c. Click ***Open***
   d. Click ***Verify***

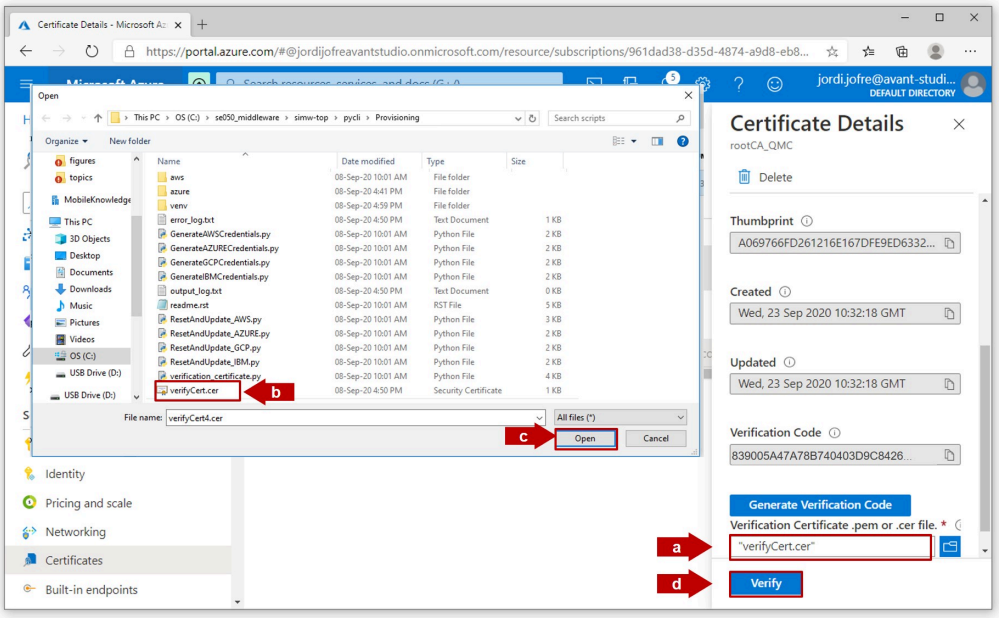

Figure 59. Root CA certificate verification

2. Check that the proof of posession verification was successul. Click on the ***Refresh*** icon. The certificate should have changed its status to ***Verified*** as shown in Figure 60. The registration of the OEM's root CA certificate is a one-time process.

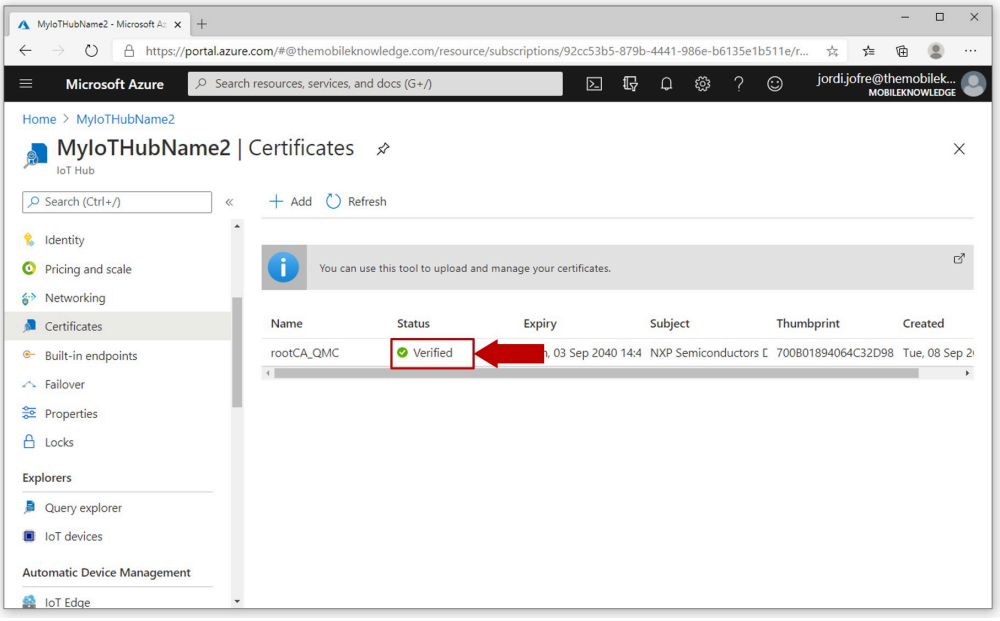

Figure 60. Root CA certificate verified

5.6 Collect Azure IoT Hub connection string

An Azure IoT Hub connection string allows secure access to Azure IoT Hub various functionality using a shared key. The connection string has the following format: `HostName=<Host Name>;SharedAccessKeyName=<Key Name>;SharedAccessKey=<SAS Key>`.

To find the connection string in your Azure IoT Hub account, follow these steps indicated in:

1. Select *Shared access policies* from the left-hand side menu.
2. Select *iothubowner* policy.

3. Copy the value in *Connection string-primary key* text box.

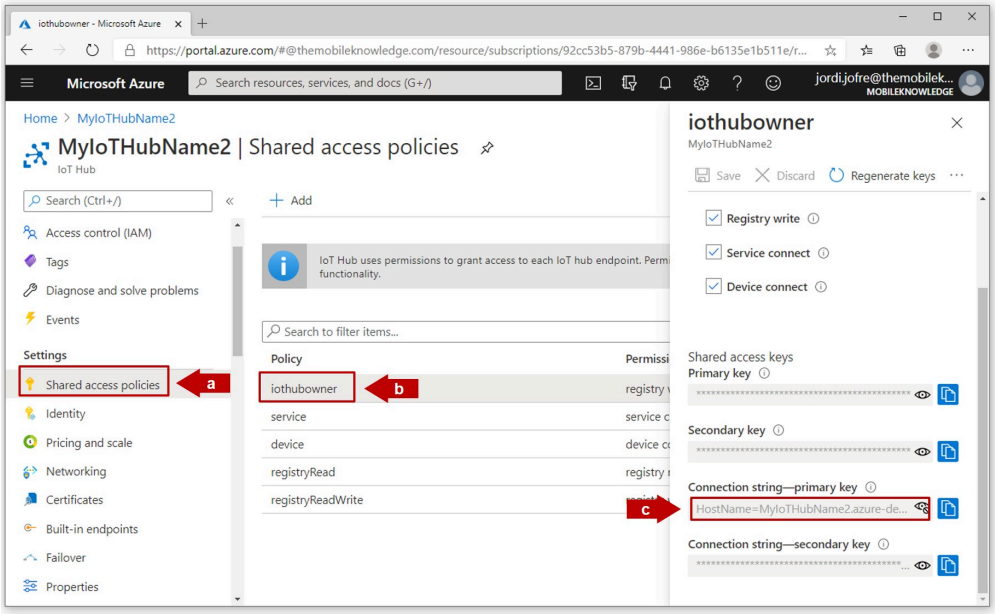

Figure 61. Connection string-primary key

The connection string will be used by the QMC JSON Exchanger application to send / retrieve data from Azure IoT Hub.

6 Build and run QMC JSON Exchanger application

The QMC JSON Exchanger application is a Windows application especially written to get the status and control the NXP quad motor-control development platform by exchanging JSON messages, either locally through UDP or remotely through the Azure IoT Hub.

In this document, we use the QMC JSON Exchanger application to send / retrieve data from Azure IoT Hub, which allows us to control and monitor the NXP quad motor-control development platform from any computer with an internet connection. To communicate with the Azure IoT Hub, the QMC JSON Exchanger application uses the Azure IoT Hub connection string retrieved in Section 5.6, which acts as a shared key.

Note: *The QMC JSON Exchanger application is provided only as a reference application for demonstration purposes, it is not intended to be used in a commercial deployment.*

The QMC JSON Exchanger application source code is delivered as a Visual Studio project, available from: www.nxp.com/quadmotorcontrol webpage in the section of **Tools and Software** with the name of ***MCUXpresso Project sample code with Azure***. Download it, and follow these steps to build it and run it in your laptop:

1. Install **Visual Studio 2017** version, or higher, with **.NET 4.7.2** framework, or higher, in your laptop, which can be downloaded free from https://visualstudio.microsoft.com/vs/. For reference, Appendix C illustrates how Visual Studio 2019 version can be installed, but the same procedure can be applied for other versions.

2. Unzip the project ***QmcJsonExchanger*** and double click in the Visual Studio solution file, called *QmcJsonExchanger.sln*, as shown in Figure 62:

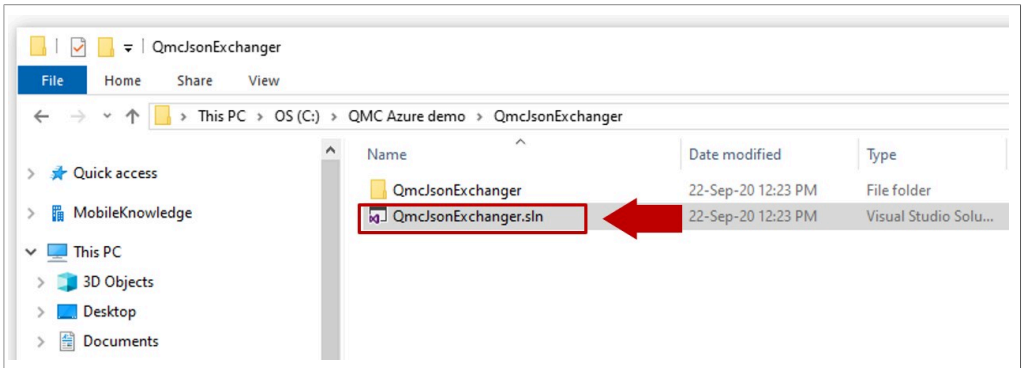

Figure 62. Open QMC JSON Exchanger application Visual Studio solution

If you see a security warning message as shown in Figure 63, click ***OK*** to continue opening the QMC JSON Exchanger application.

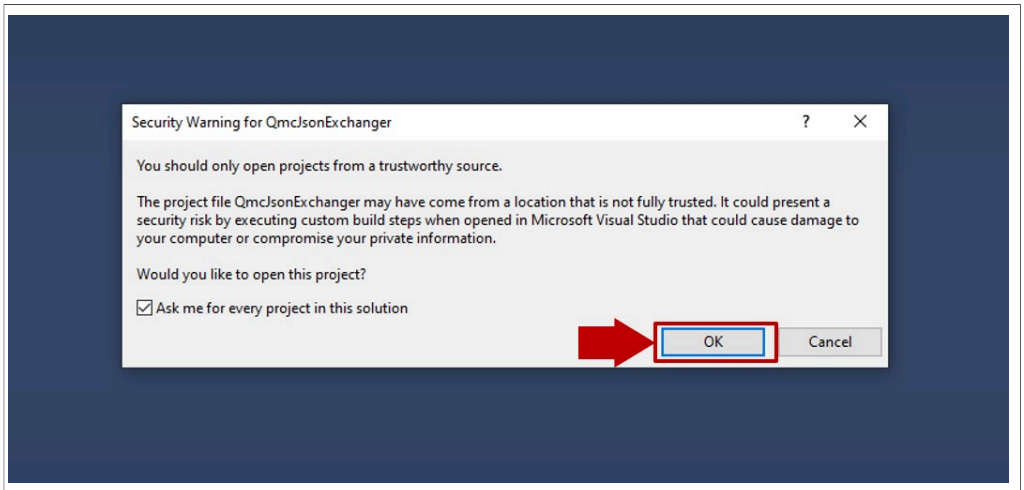

Figure 63. Accept security warning Visual Studio solution

The QMC JSON Exchanger application targets .NET Framework version 4.7.2 or newer. In case you do not have the .NET Framework version 4.7.2 installed, you may see the warning shown in Figure 64. Select *Download the targeting pack for*

"NETFramework, Version=v4.7.2.. The project will not change option, follow the instructions to install .NET Framework version 4.7.2 and restart your computer.

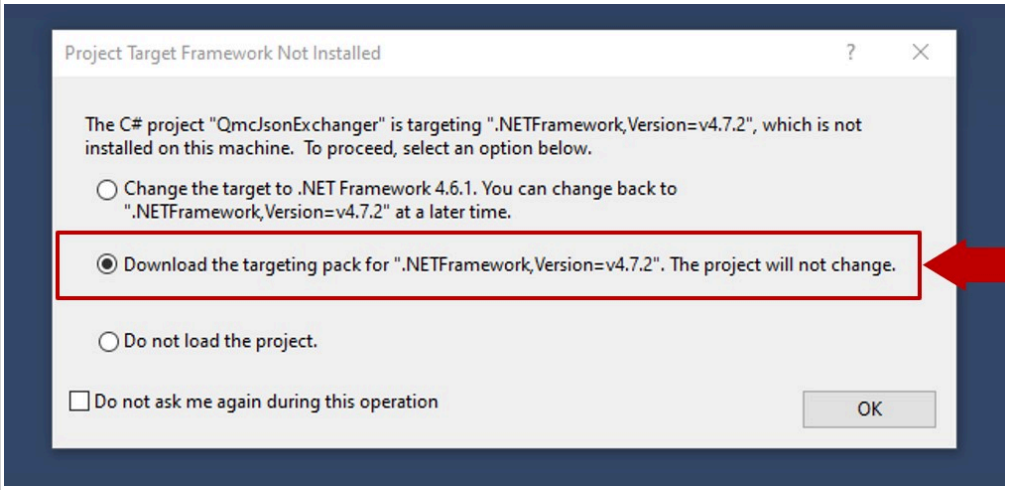

Figure 64. Install .NET Framework version 4.7.2 warning message

3. The Visual Studio project solution will be opened. To build and run the QMC JSON Exchanger application, open the file ***Program.cs*** in the Solution Explorer panel and click on the Start button in the top menu, as shown in Figure 65:

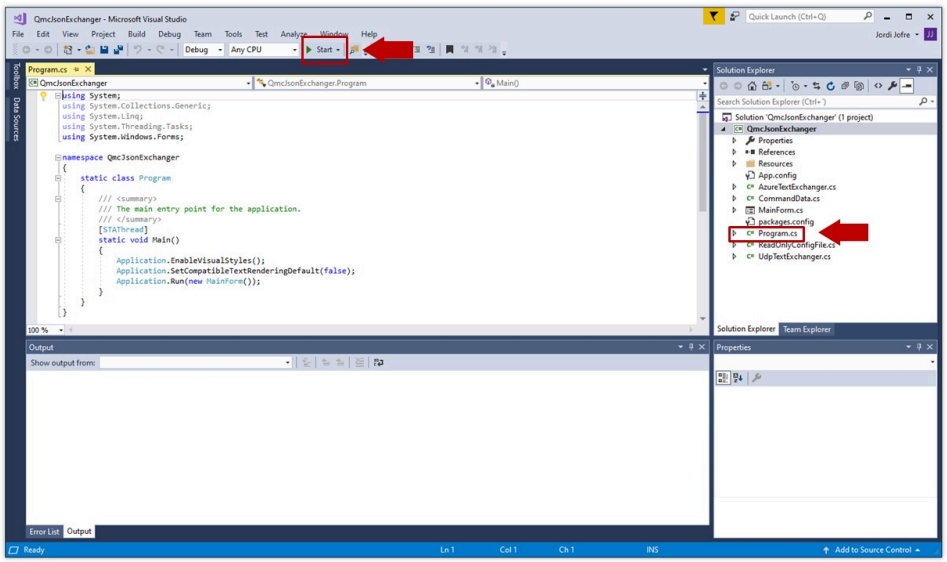

Figure 65. Build QMC JSON Exchanger application

4. Once the source code is compiled, the QMC JSON Exchanger application will be opened, as shown in Figure 66:

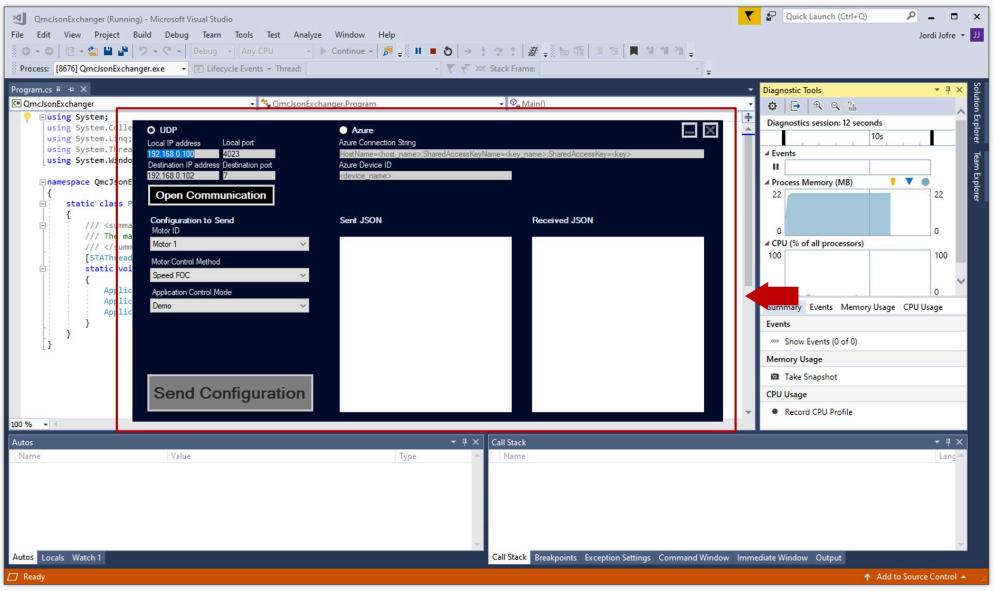

Figure 66. QMC JSON Exchanger application user interface

5. In case you experience the compilation error shown in Figure 67, you can solve it by granting security rights to the *Resources.resx* file.

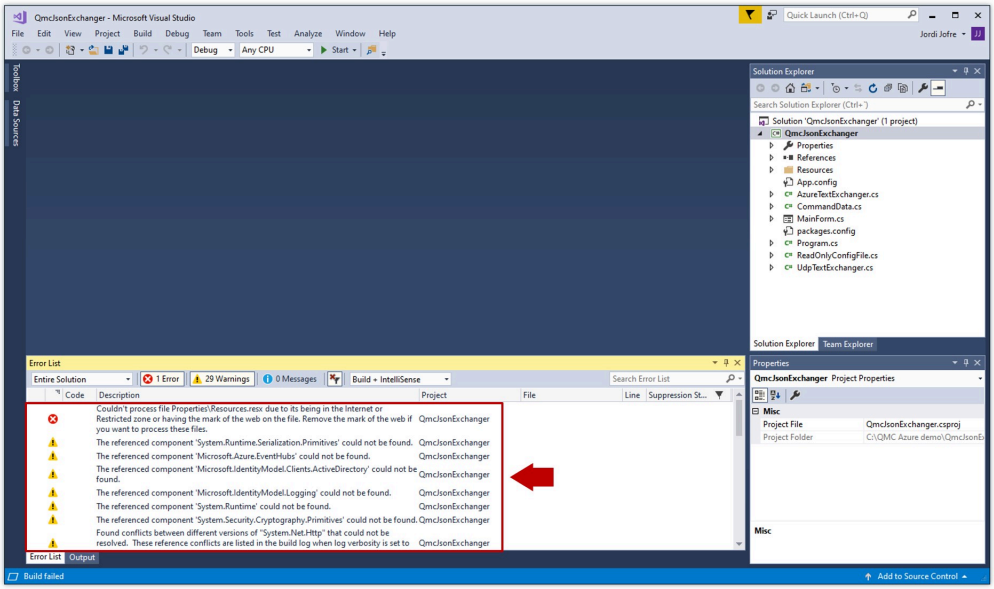

Figure 67. QMC JSON Exchanger application building error

To grant security rights to the *Resources.resx* file, use a Windows explorer and navigate to the *Resources.resx* file, which is stored in `<root_directory>\QmcJsonExchanger\QmcJsonExchanger\Properties` folder directory. Then:

a. Right click on the file.
b. Click on *Properties*.
c. Tick the *Unblock* option.
d. Click in *OK* button, as shown in Figure 68

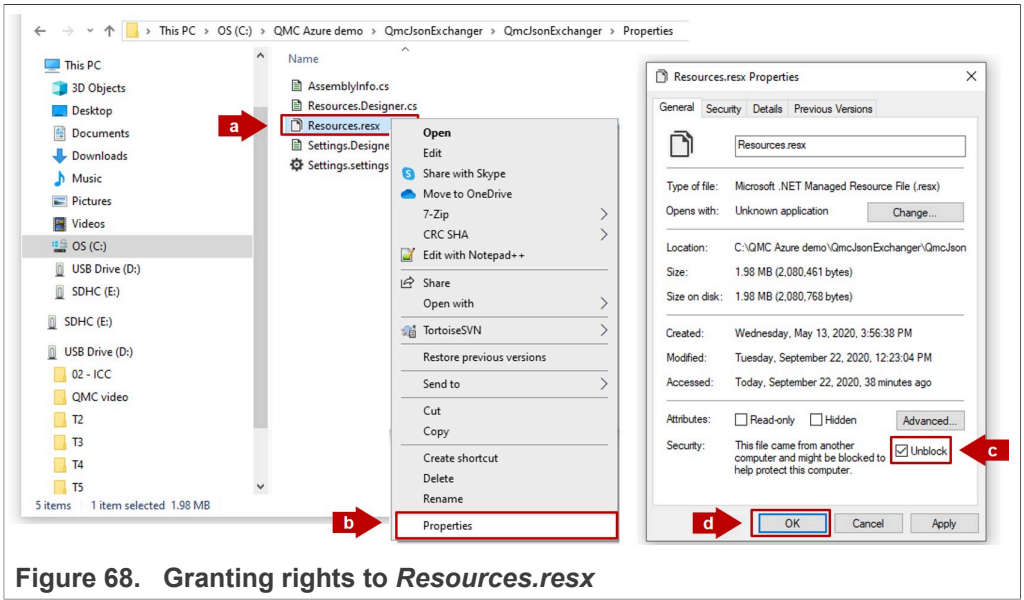

Figure 68. Granting rights to *Resources.resx*

After that, re-build the Visual Studio project once again.

7 Run the Azure IoT Hub demo application

This section explains how to put all the pieces together and run the Azure IoT Hub demo application.

7.1 Change MCUXpresso project settings

We need to change two variables in the MCUXpresso project related with your Azure IoT Hub account settings. Changing these settings is needed in order to successfully run the Azure IoT Hub demo application. In the MCUXpresso workspace:

1. Go to the project explorer and open the `azure_iot_config.h`, which can be found in `source\azure_task\` folder as shown in Figure 69:

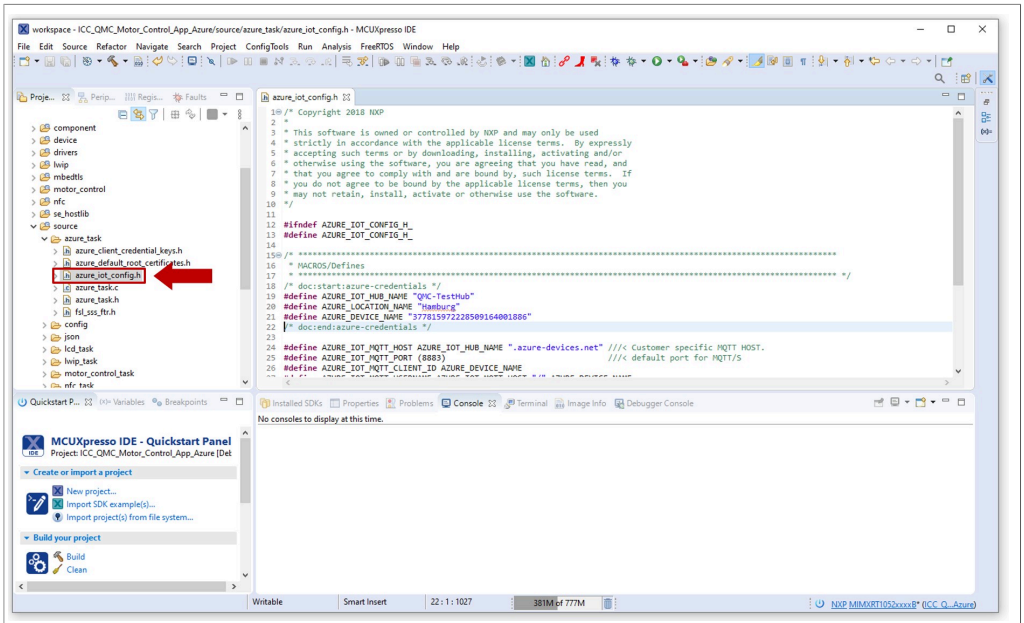

Figure 69. Find `azure_iot_config.h` file

2. Modify the `#define AZURE_IOT_HUB_NAME` macro definition to add your Azure IoT Hub name created in Section 5.2 as shown Figure 70:

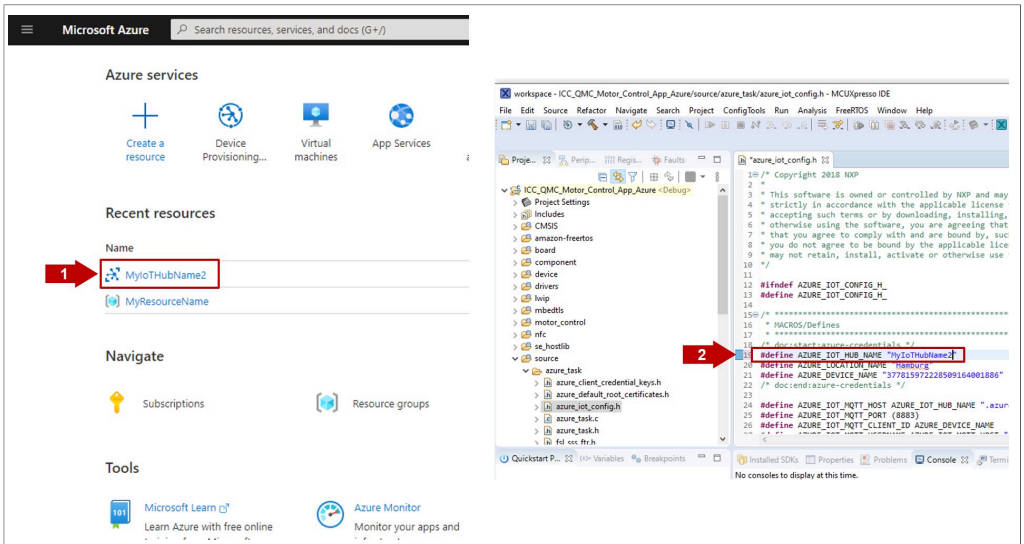

Figure 70. Change `#define AZURE_IOT_HUB_NAME`

3. Modify the `#define AZURE_DEVICE_NAME` macro definition to add your device ID created in Section 5.3 as shown in Figure 71:

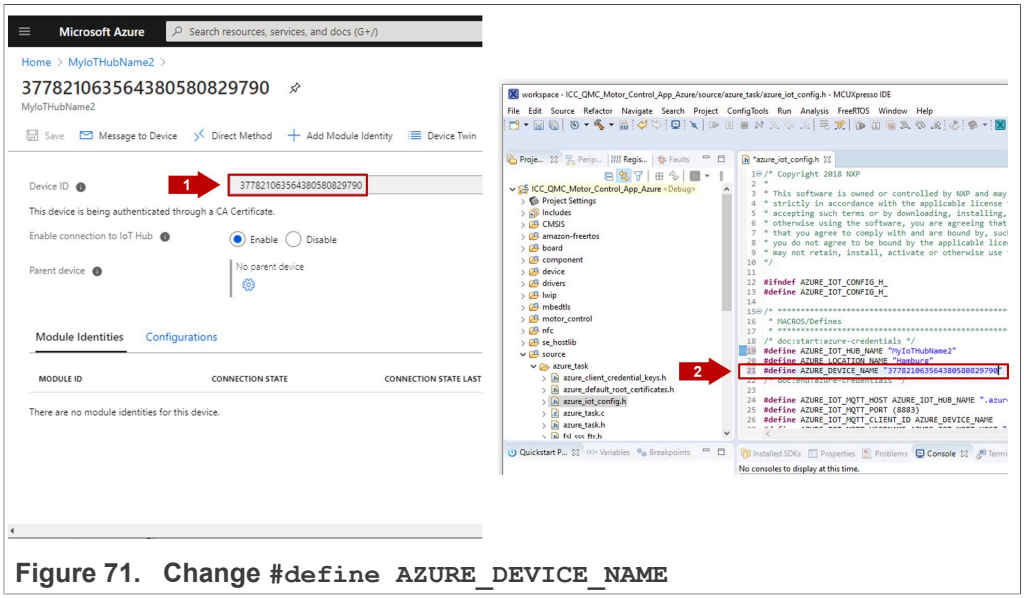

Figure 71. Change `#define AZURE_DEVICE_NAME`

7.2 Run MCUXpresso project example

After changing the `#define AZURE_IOT_HUB_NAME` and `#define AZURE_DEVICE_NAME` project macros, we can run the MCUXpresso project. Follow these steps:

1. Make sure the NXP quad motor-control application hardware, LPC-Link2 debug probe are connected as illustrated in Figure 72:

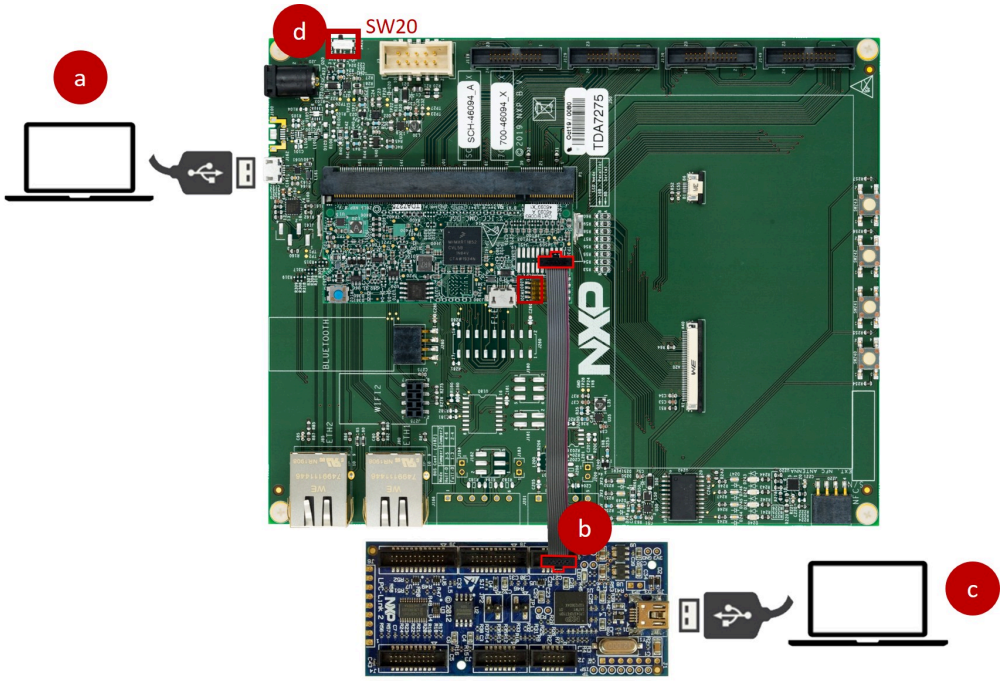

Figure 72. Plug-in the boards to the computer

2. Connect the digital board to a DHCP-capable network using an Ethernet cable so that the NXP quad motor-control application can connect to internet as shown in Section 7.2:

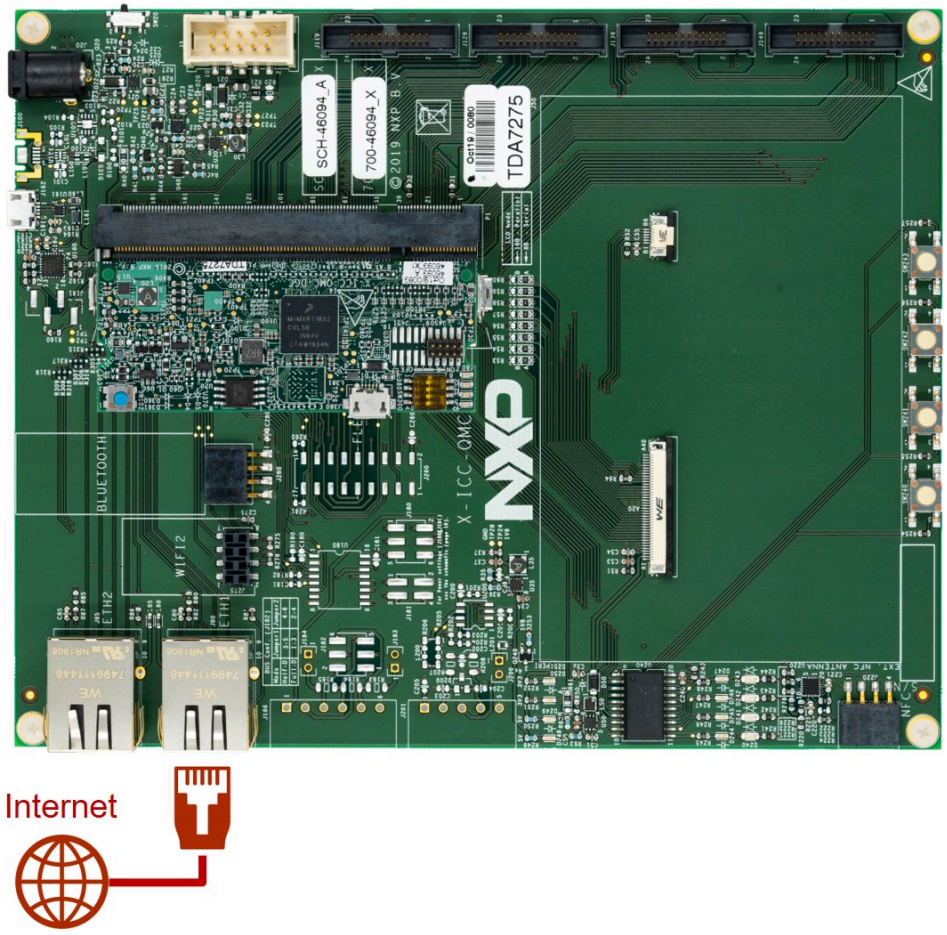

Figure 73. Connect NXP quad motor-control application to Internet

3. Connect the power stage board and the four Teknic M-2310P-LN-04K motors as shown in Figure 74:

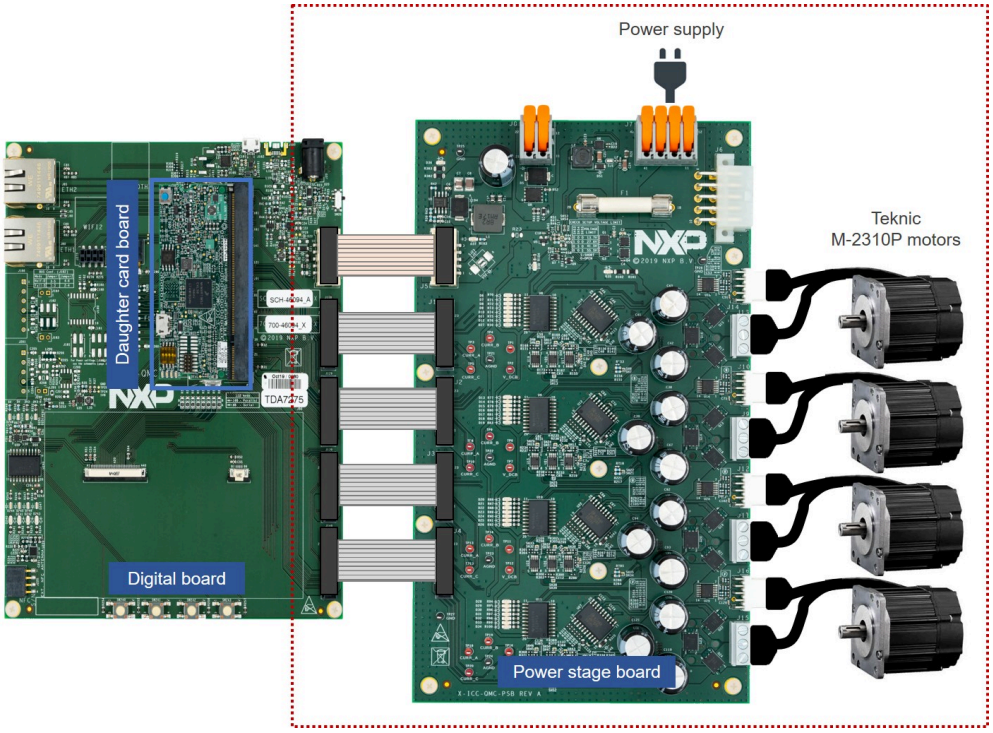

Figure 74. Connect power stage board and motors

Note: The four headers named "`mX_pmsm_appconfig.h`" in `<project_directory>\ source\ motor_control_task` can be swapped with

the ones in either the “*teknic*” or “*tg drives*” subfolders to specify the type of motor used in the physical demo (Figure 75)

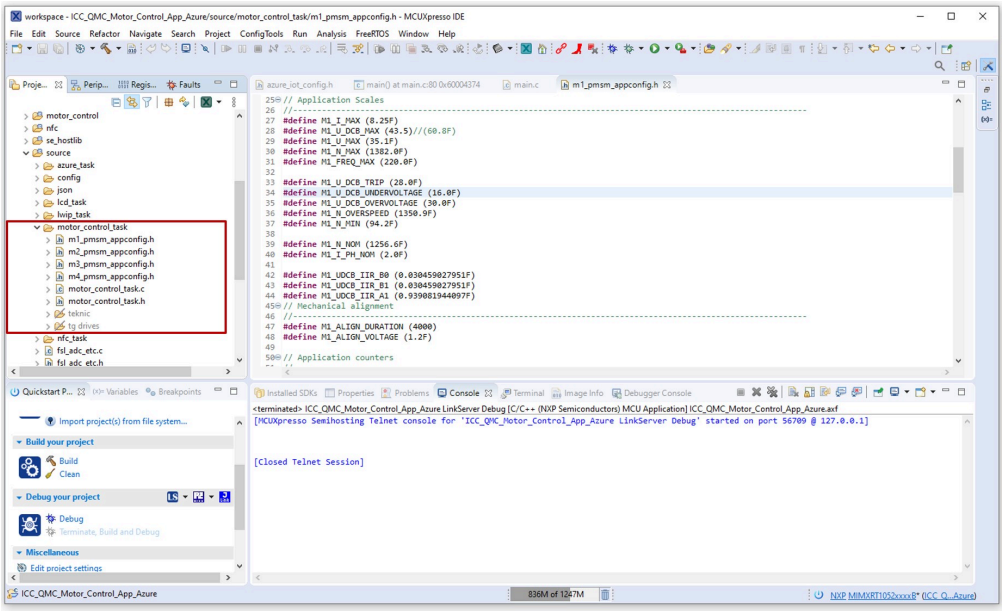

Figure 75. Motor parameter header files

4. After the setup is ready, go to the MCUXpresso Quickstart Panel and click ***Debug*** button as shown in Figure 76. Wait a few seconds until the project builds and executes:

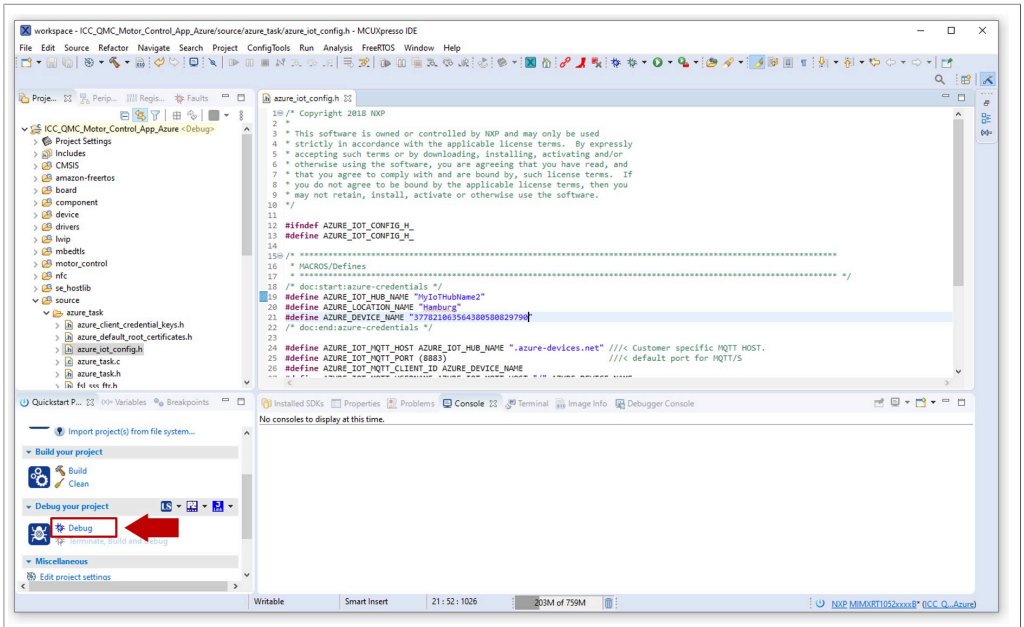

Figure 76. Debug MCUXpresso project

5. The compilation and execution will take a few seconds. After that, it will automatically stop in a breakpoint. Click on Resume to allow the software to continue its execution as shown in Figure 77:

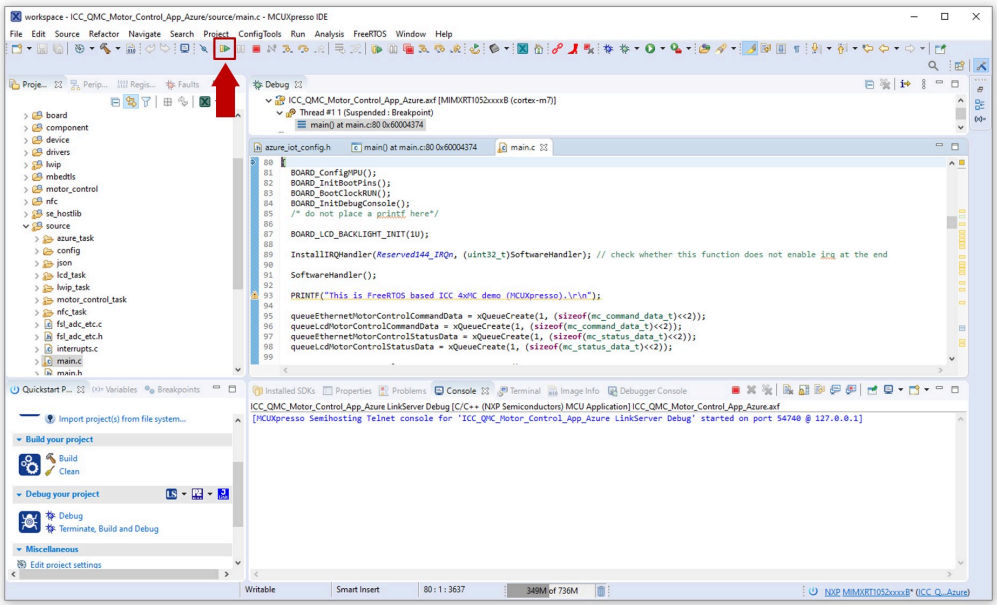

Figure 77. Resume MCUXpresso project

7.3 Run QMC JSON Exchanger application

Once the NXP quad motor-control application is already running with the MCUXpresso project, we can execute the QMC JSON Exchanger application to monitor data and send control commands. Follow these steps:

1. Make sure you can run the QMC JSON Exchanger application. If not, follow the instructions provided in Section 7.3.
2. Make sure your laptop is connected to Internet.

3. Open the Visual studio project solution and run the QMC JSON Exchanger application by clicking the ***Start*** button, as shown in Figure 78:

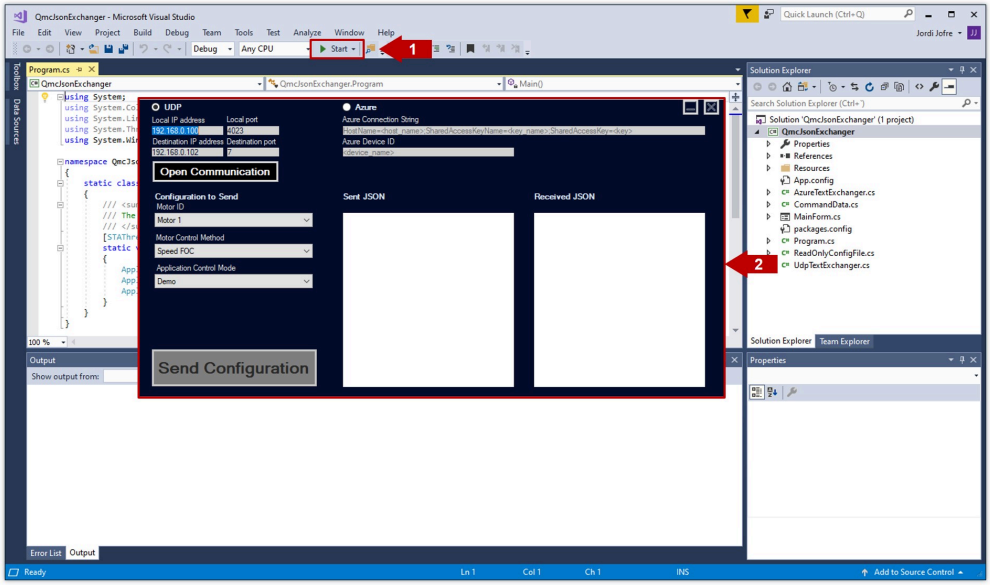

Figure 78. Run QMC JSON Exchanger application

4. Fill in the fields in QMC JSON Exchanger application as shown in Figure 79:
   a. Select the *Azure* option
   b. Type the Azure *connection string*, obtained in Section 5.6
   c. Type the *Device ID*, created in Section 5.3
   d. Click *Open Communication* button.

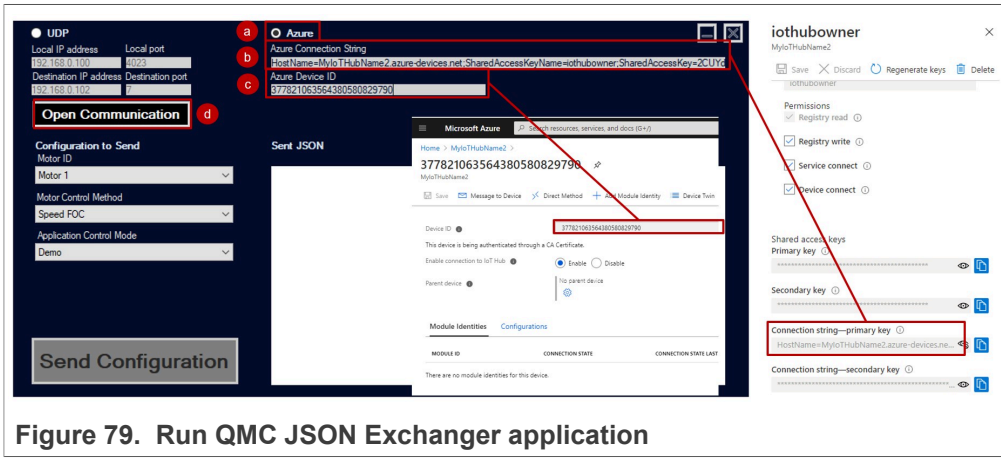

Figure 79. Run QMC JSON Exchanger application

5. Now, you can use the QMC JSON Exchanger application to monitor and send control commands to the NXP quad motor-control application as shown in Figure 80. For that:
   a. Select the *motor* (motor 1-4), the *motor-control method* (position control, speed FOC) and the *application control mode* (demo, manual mode) using the *Configuration to Send* area.
   b. Click on *Send Configuration* button.
   c. Check the control command sent in the *Sent JSON* text box.
   d. Check the telemetry data received in real-time from the NXP quad motor-control application in the *Received JSON* text box

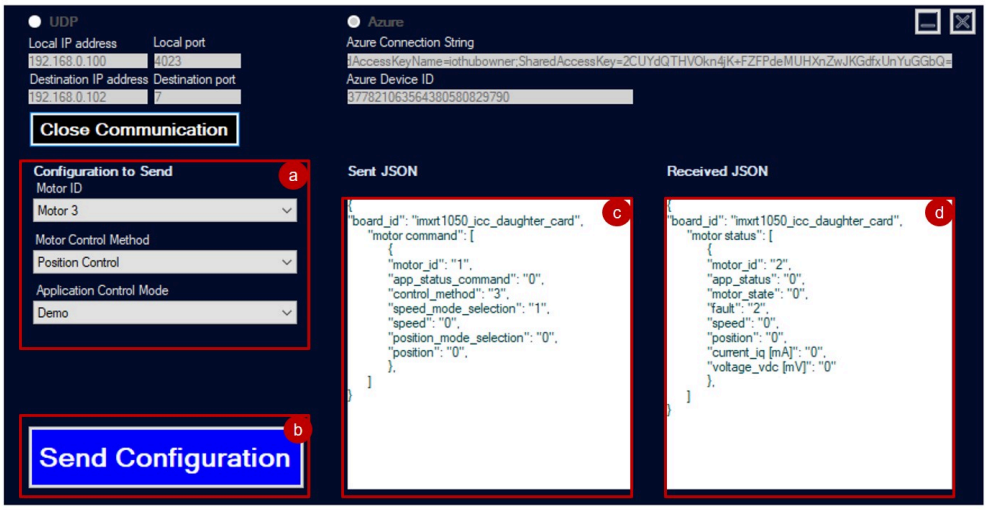

Figure 80. Use of the QMC JSON Exchanger application

You should also see how the motors behavior change according to the commands sent via the QMC JSON Exchanger application UI.

6. In addition, you can also check that some activity is traced in your Azure IoT Hub account dashboard, as shown in Figure 81:

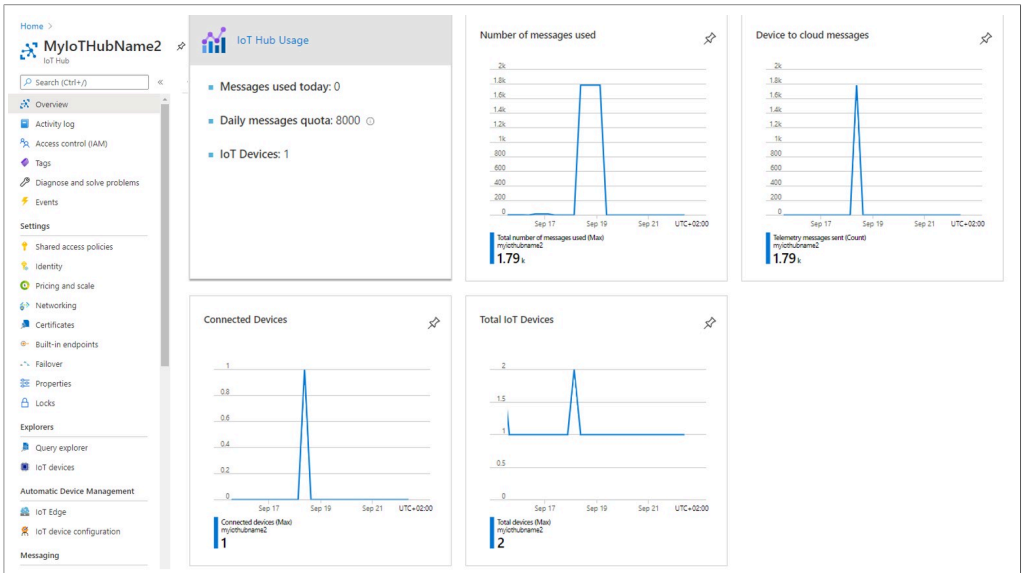

Figure 81. Azure IoT Hub account detected activity

If you have reached this point, the bringing up is completed. The Azure IoT Hub demo application is up and running.

8 Appendix A: MCUXpresso installation

MCUXpresso is a free-of-charge, code-size-unlimited, easy-to-use IDE for Kinetis and LPC MCUs, and i.MX RT crossover processors. In this demo it is used to build and run the example. In order to install it, follow the steps below:

Note: *MCUXpresso version 11.2.0 or later is recommended, as it offers all the functionalities for the required SDK version (2.8.2).*

1. Go to MCUXpresso and click the download button as indicated in Figure 82:

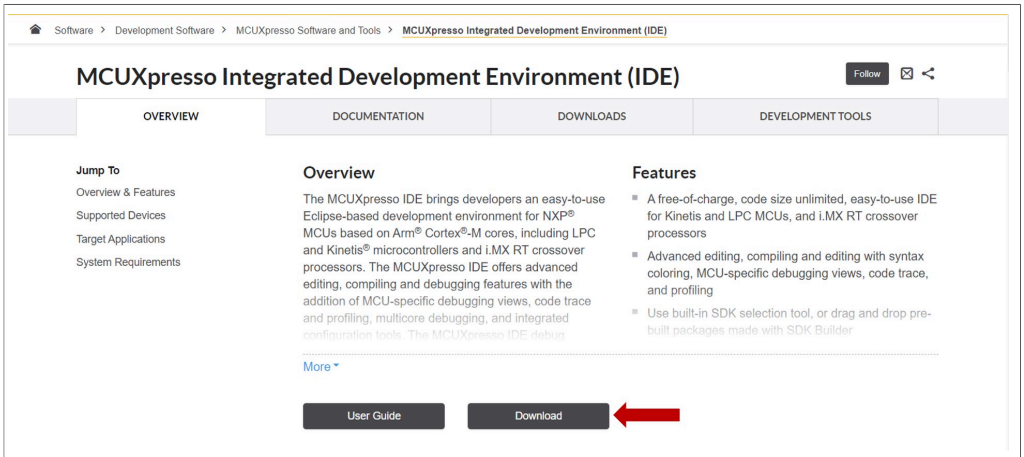

Figure 82. MCUXpresso web page

2. Select the MCUXpresso link with the latest version as indicated in Figure 83:

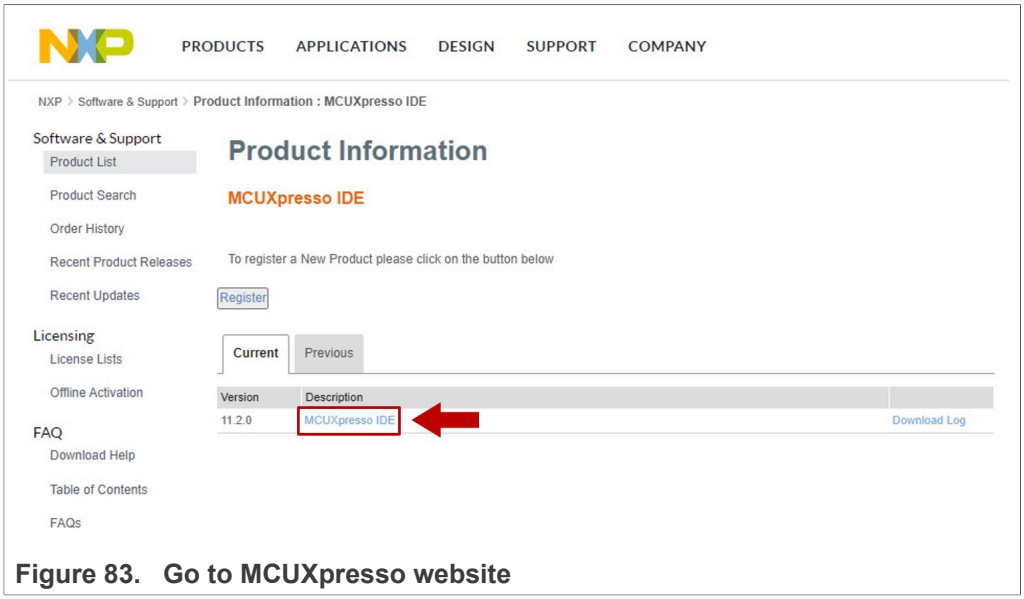

Figure 83. Go to MCUXpresso website

3. Accept software terms and conditions as shown in Figure 84:

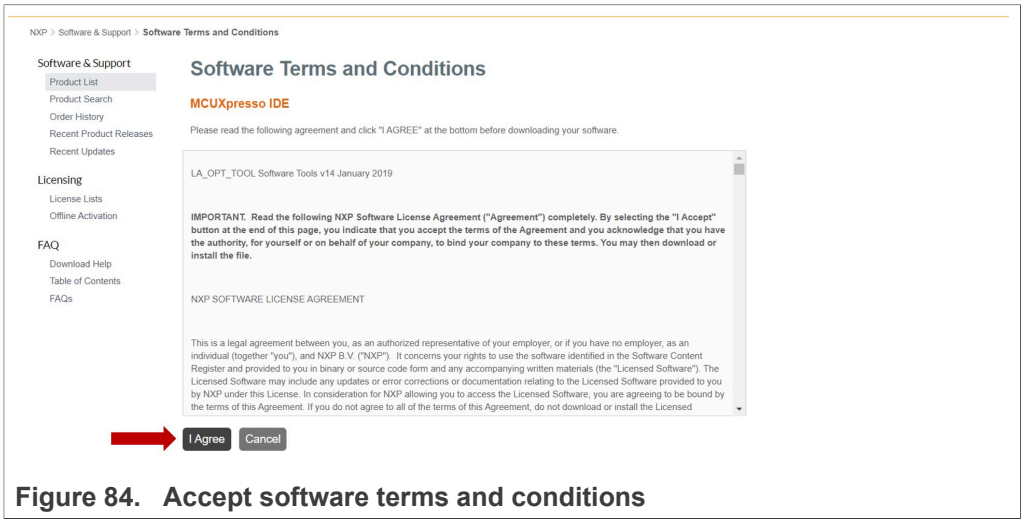

Figure 84. Accept software terms and conditions

4. Select your MCUXpresso product version and click on the corresponding operative system to start the download as shown in Figure 85:

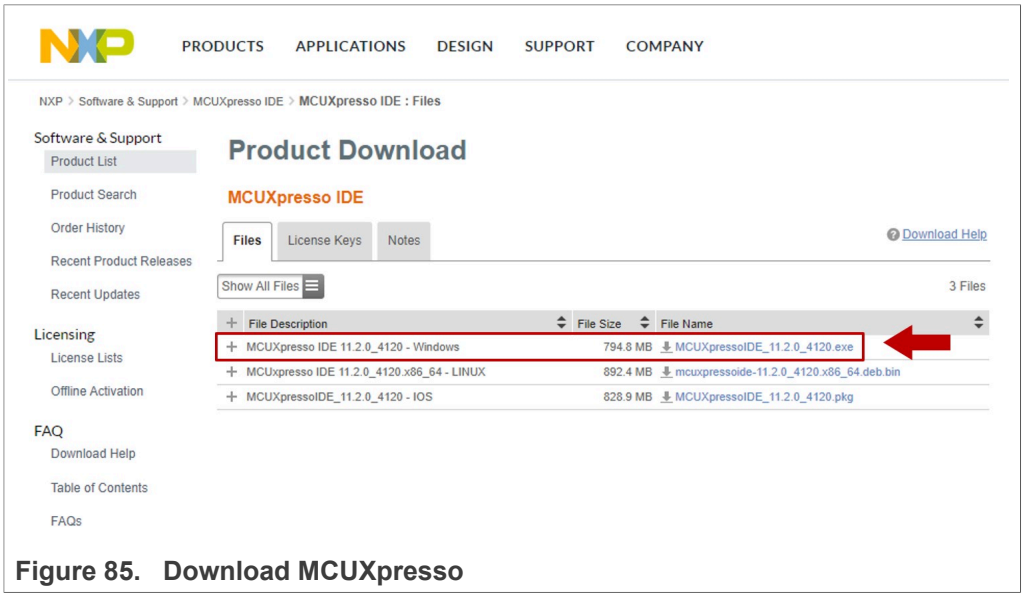

Figure 85. Download MCUXpresso

5. Double click on the installer file and follow the setup wizard until MCUXpresso installation is completed. Please, make sure you allow the installation of the additional

drivers required by MCUXpresso during the installation process as shown in Figure 86, Figure 87, Figure 88 and Figure 89:

Figure 86. Install MCUXpresso required drivers I

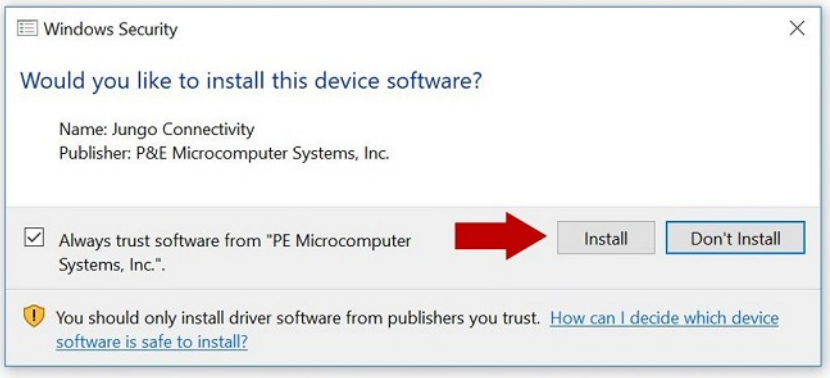

Figure 87. Install MCUXpresso required drivers II

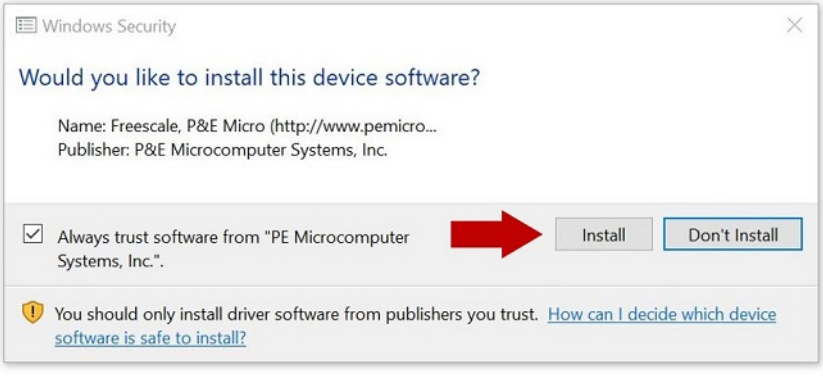

Figure 88. Install MCUXpresso required drivers III

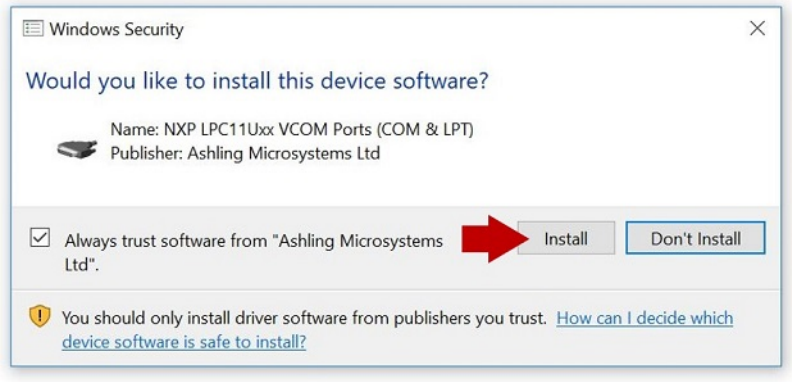

Figure 89. Install MCUXpresso required drivers IV

9 Appendix B: Install Python 3.7.x 32-bit version

Use these screenshots to install Python 3.7.x in your Windows machine:

1. Go to https://www.python.org/downloads/release/python-375/ and download **Python v.3.7.x 32 bit version**. Make sure you download Python v3.7.x 32 bit version.

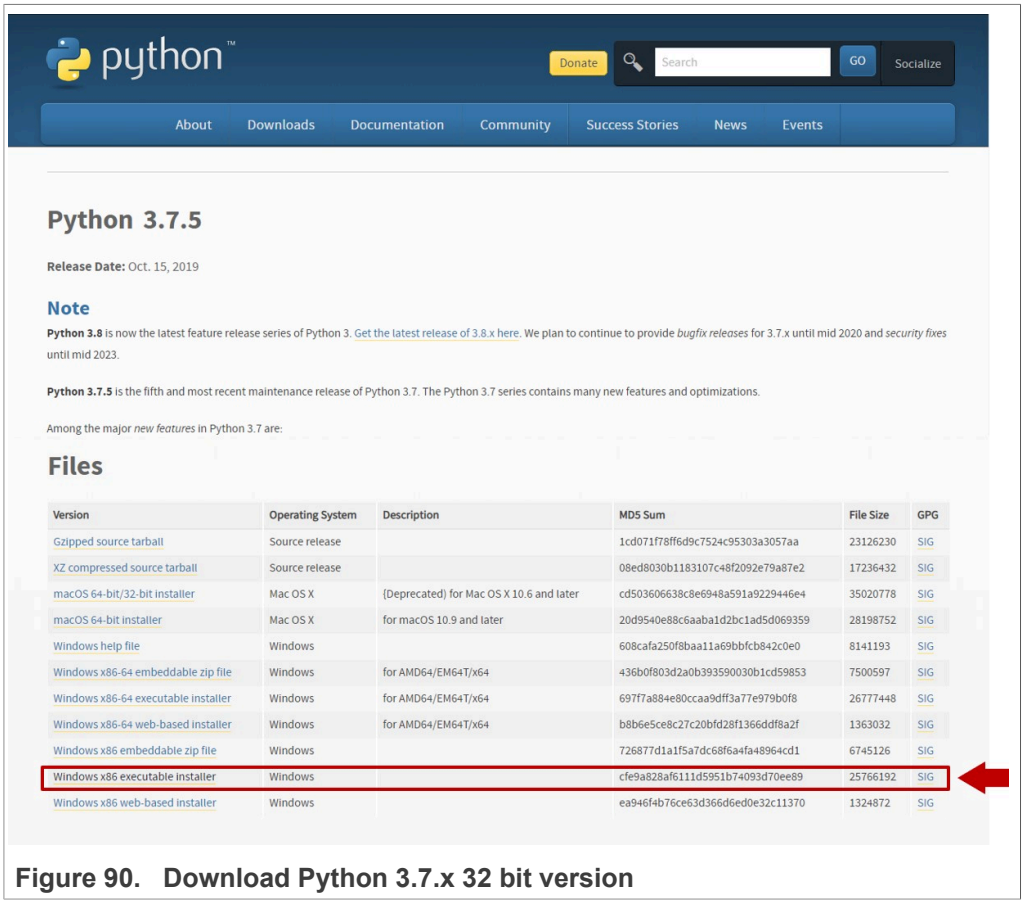

Figure 90. Download Python 3.7.x 32 bit version

2. Double click on the downloaded installer file. Select the "*Install launcher for all users*" and "*Add Python 3.7 to Path*" options and click *Install Now* as indicated in Figure 91:

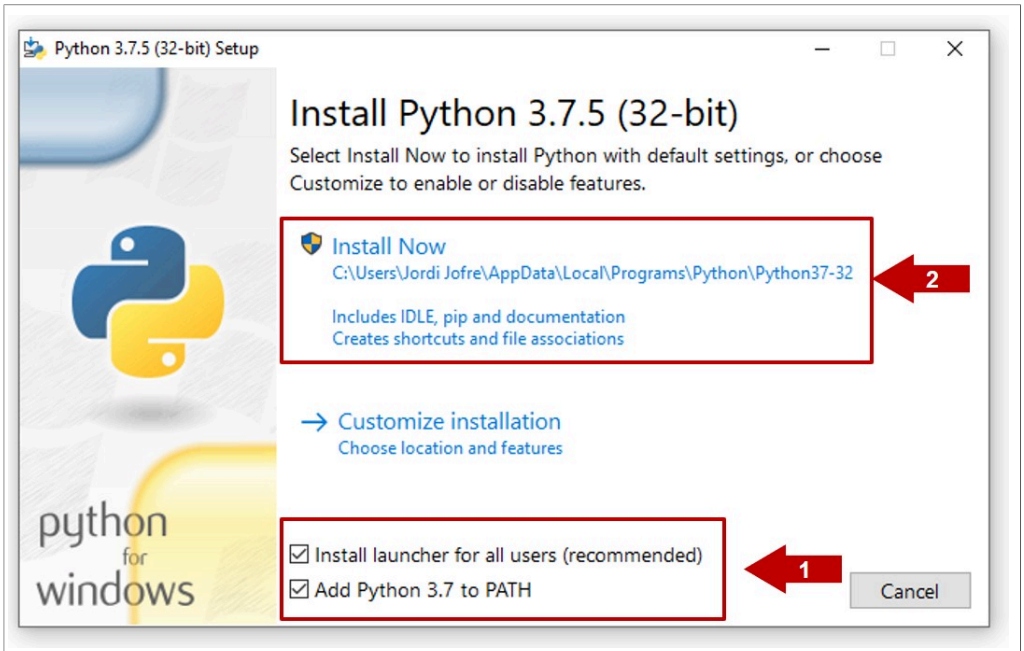

Figure 91. Install Python 3.5.x 32 bit for Windows

3. Wait a few seconds until the installation is completed as indicated in Figure 92

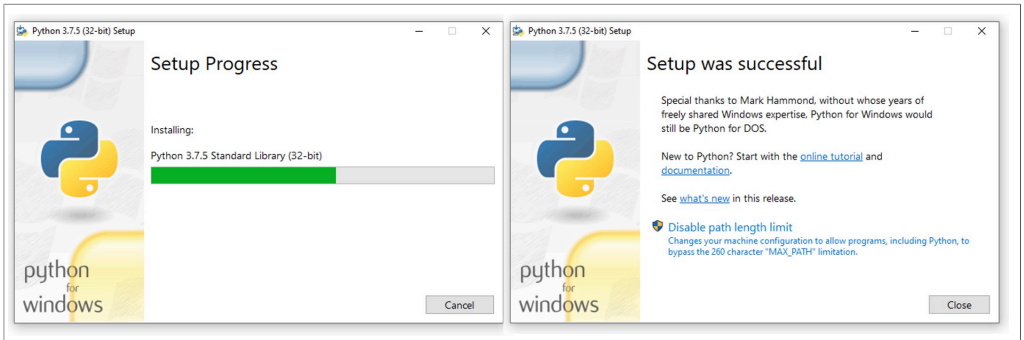

Figure 92. Python 3.5.x 32 bit installation completed

10 Appendix C: Install Visual Studio Community 2019

Visual Studio is Microsoft's fully-featured IDE for Android, iOS, Windows, web, and cloud. To install Visual Studio 2019:

1. Go to Visual Studio site.
2. Select (1) ***Windows*** and click on ***Community 2019*** in the *Download Visual Studio* button as shown in Figure 93:

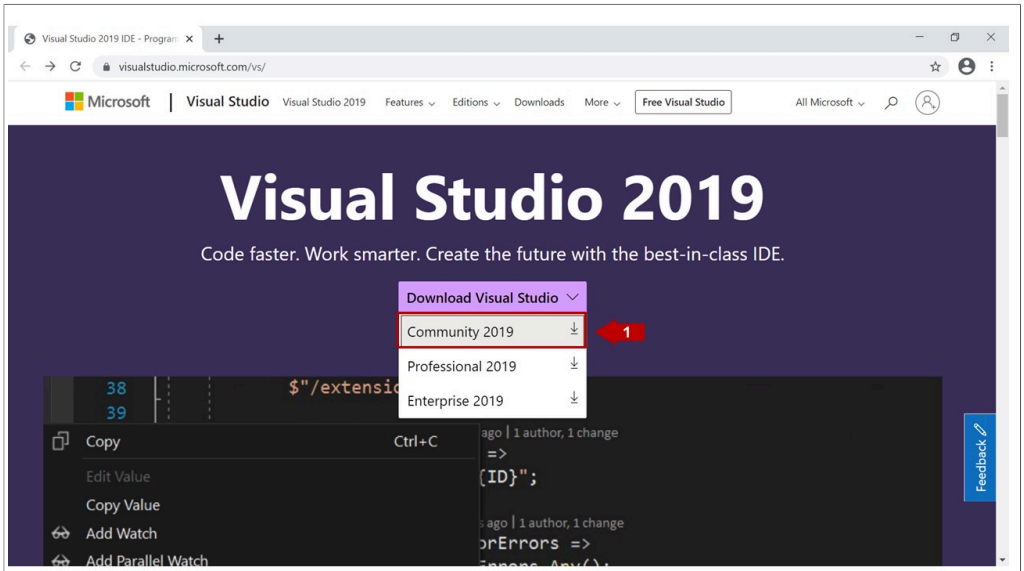

Figure 93. Download Visual Studio IDE

3. An *.exe installer will download to your laptop. Double click on the installer file and follow the setup wizard until the installation is completed. This process might take a few minutes. Figure 94 shows Visual Studio installation wizard as an example:

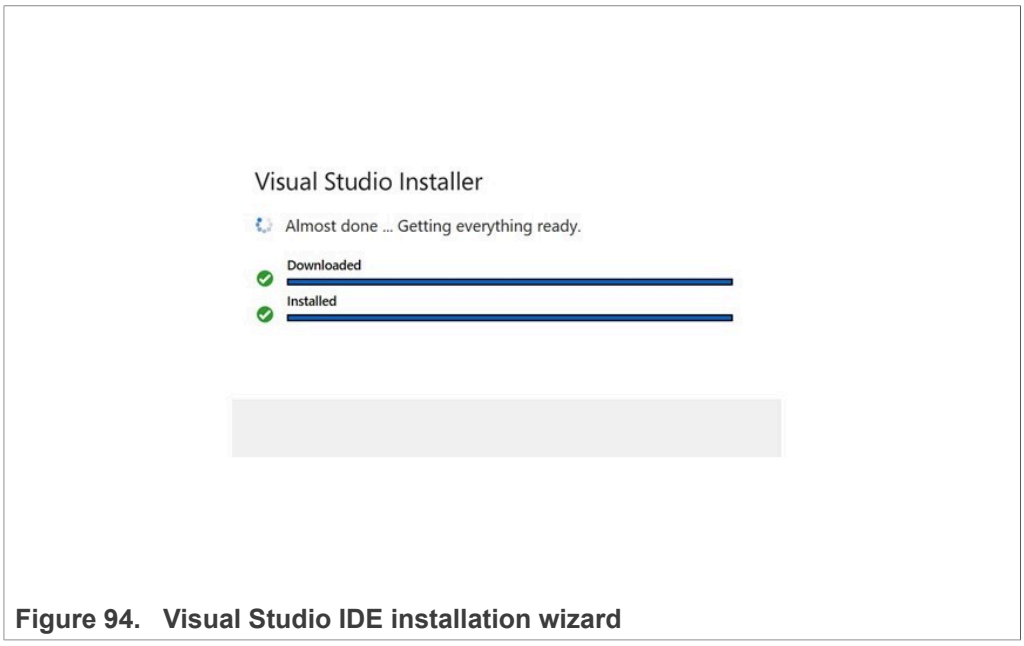

Figure 94. Visual Studio IDE installation wizard

4. As part of the Visual Studio setup, it is mandatory that you enable the installation of ***Desktop development with C++***. Select (1) ***Desktop development with C++*** and (2) click install as shown in Figure 95:

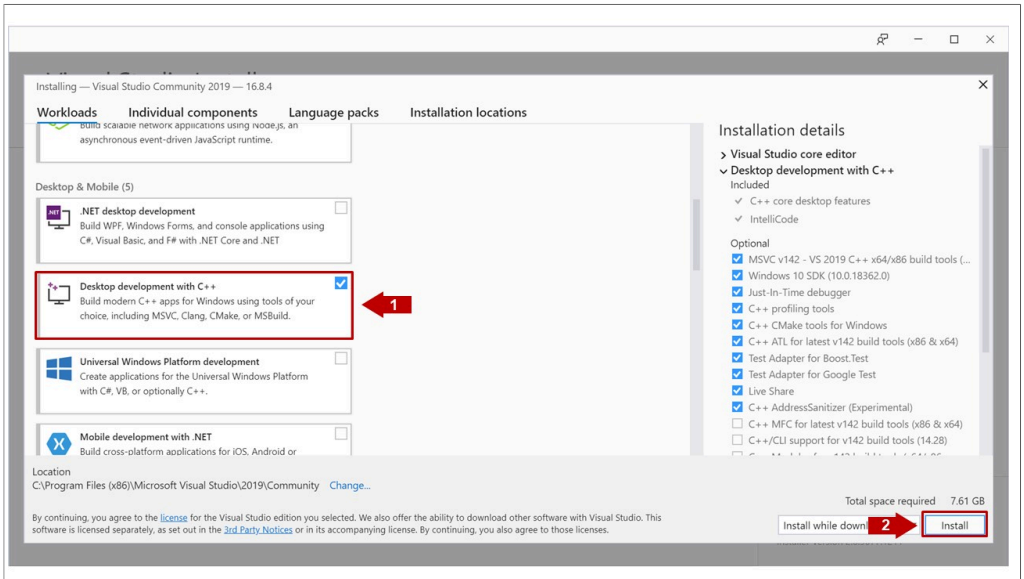

Figure 95. Select desktop development with C++

5. Visual C++ Tools for CMake is installed by default as part of the Desktop development with C++ workload. This process might take several minutes. Figure 96 shows Visual Studio installation wizard as an example:

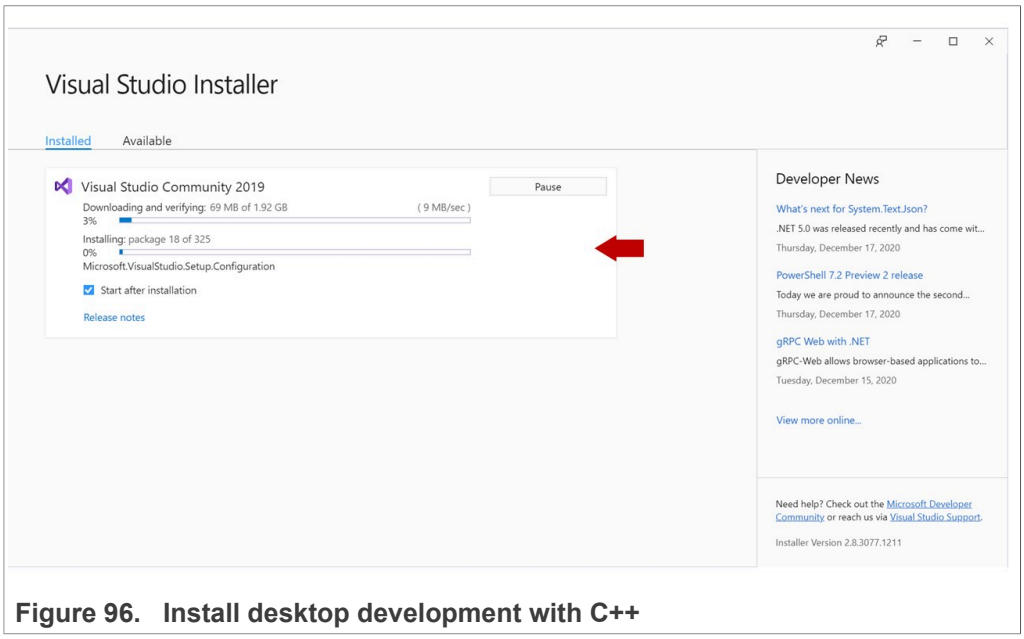

Figure 96. Install desktop development with C++

6. After the installation is completed, you might be asked to reboot your system as shown in Figure 97:

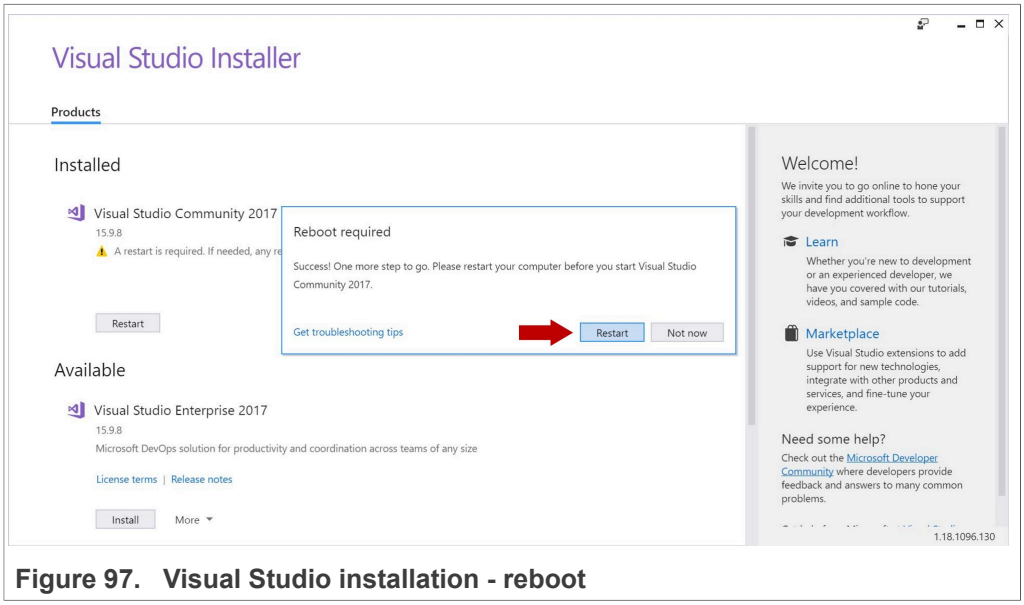

Figure 97. Visual Studio installation - reboot

11 Legal information

11.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

11.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

11.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Tables

Figures

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10 February 2021