W5100S-EVB-Pico

Getting Started Guide

for Azure IoT

Version 1.0.0

Table of Contents

1 Document information

1.1 Revision history

Version	Date	Description of change
V1.0.0	2022-01-19	Initial Release

2 Introduction

This document describes how to connect **W5100S-EVB-Pico** running **Windows 10** with Azure IoT SDK. This multi-step process includes:

- Configuring Azure IoT Hub
- Registering your IoT device
- Provisioning your devices on Device Provisioning service
- Build and deploy Azure IoT SDK on device

3   Prerequisites

3.1   Prepare your development environment

3.1.1   Tools installation

Windows 10 was used during preparation of this guide document. Linux and MacOS user should use compatible software, hardware-wise there is no difference. Please refer to the guide in *section 3.1.3* to find instructions for installing toolchain on Linux and MacOS.

1) Install the Toolchain

To build you will need to install extra tools below.

- · [ARM GCC compiler](#)
- · [CMake](#)
- · [Build Tools for Visual Studio](#)
- · [Python 3.9](#)
- · [Git](#)
- · [Visual Studio Code](#)

Download the executable installer for each of these from the links above, and then carefully follow the instructions in the following sections to install all six packages on to your Windows computer.
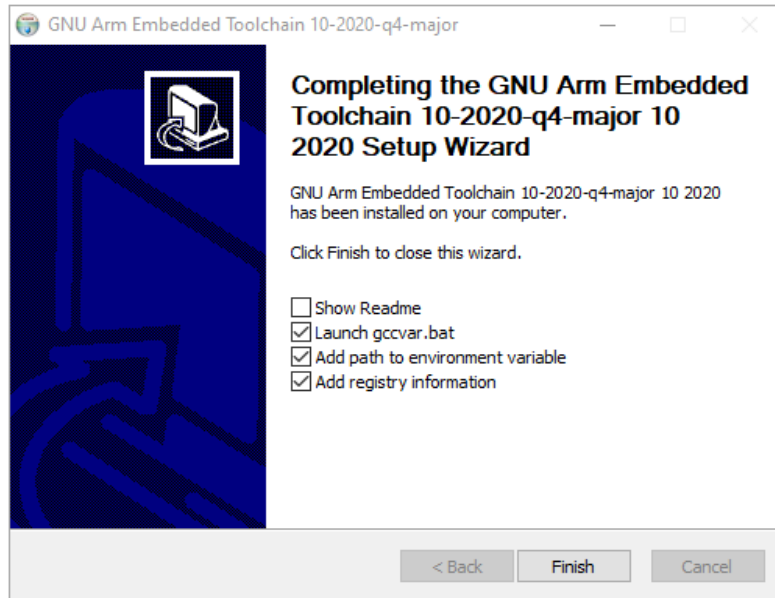
① Install ARM GCC compiler

Figure 1. Install ARM GCC compiler

During installation you should check the box to register the path to the ARM compiler as an environment variable in the Windows shell when prompted to do so.
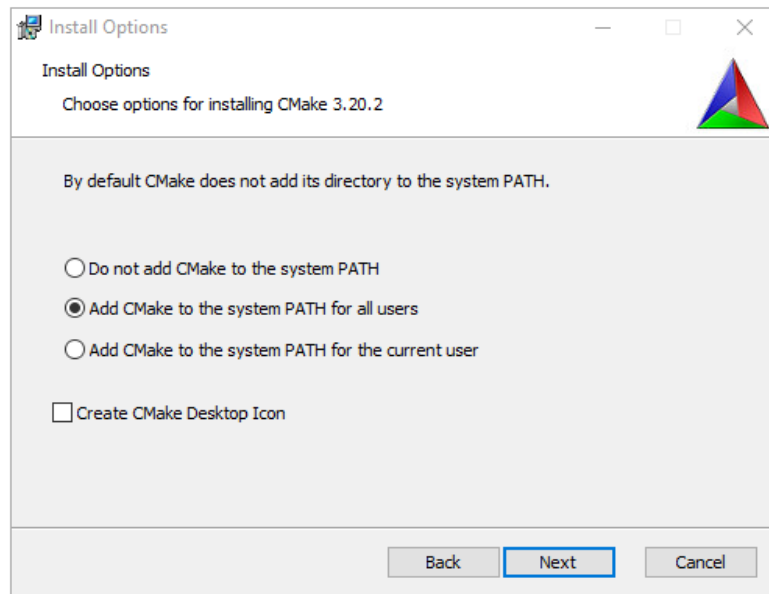
② Install CMake

Figure 2. Install CMake

During the installation add CMake to the system **PATH** for all users when prompted by the installer.
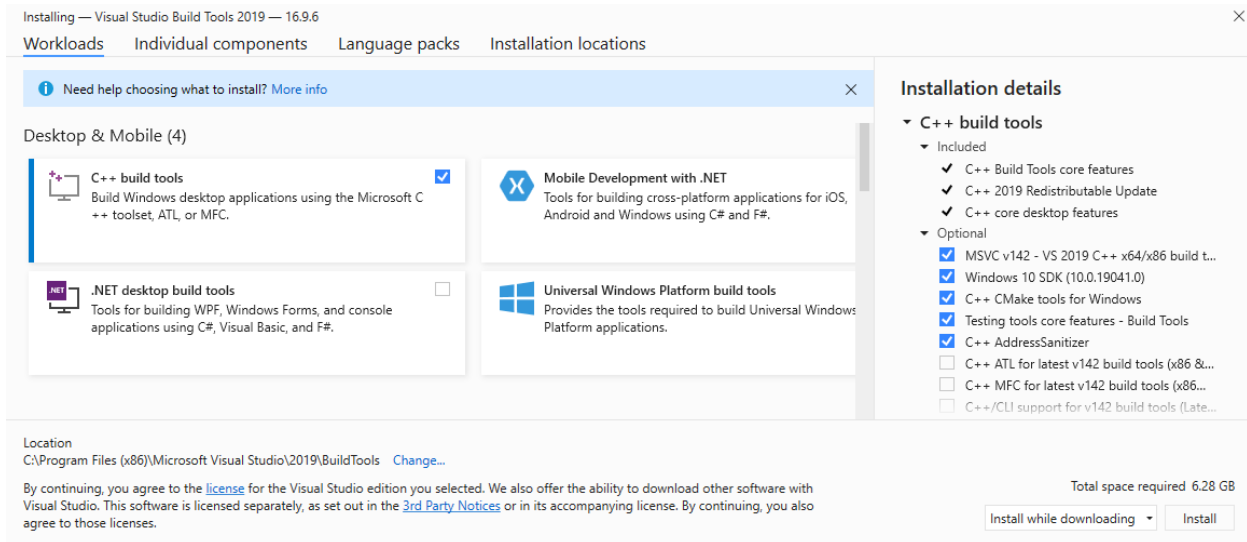
③ Install Build Tools for Visual Studio

Figure 3. Install Build Tools for Visual Studio

When prompted by the Build Tools for Visual Studio installer you need to install the C++ build tools only.

④ Install Python 3.9

During the installation, ensure that it's installed 'for all users' and add Python 3.9 to the system **PATH** when prompted by the installer. You should additionally disable the **MAX_PATH** length limit when prompted at the end of the Python installation.

Figure 4. Install Python

⑤ Install Git

When installing Git you should ensure that you change the default editor away from vim.
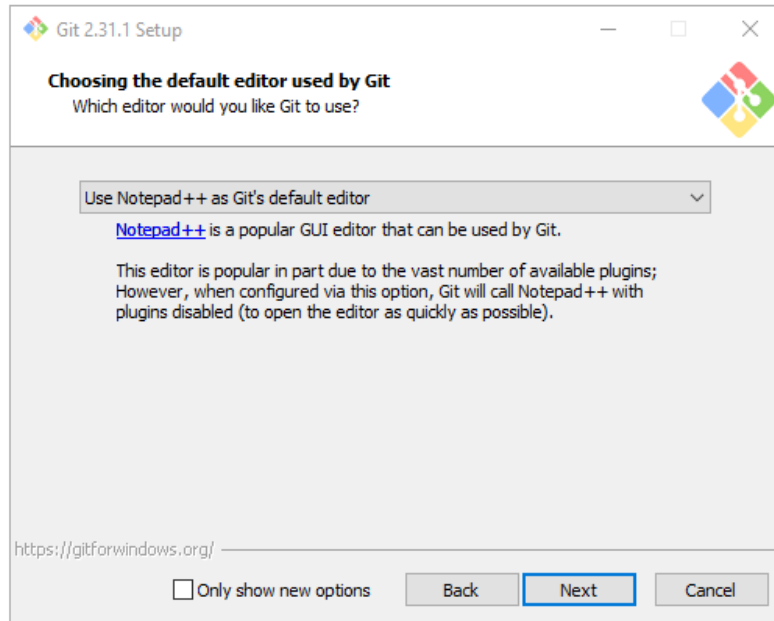
Figure 5. Install Git

⑥ Install Visual Studio Code

During the installation add Visual Studio Code to the system **PATH**.
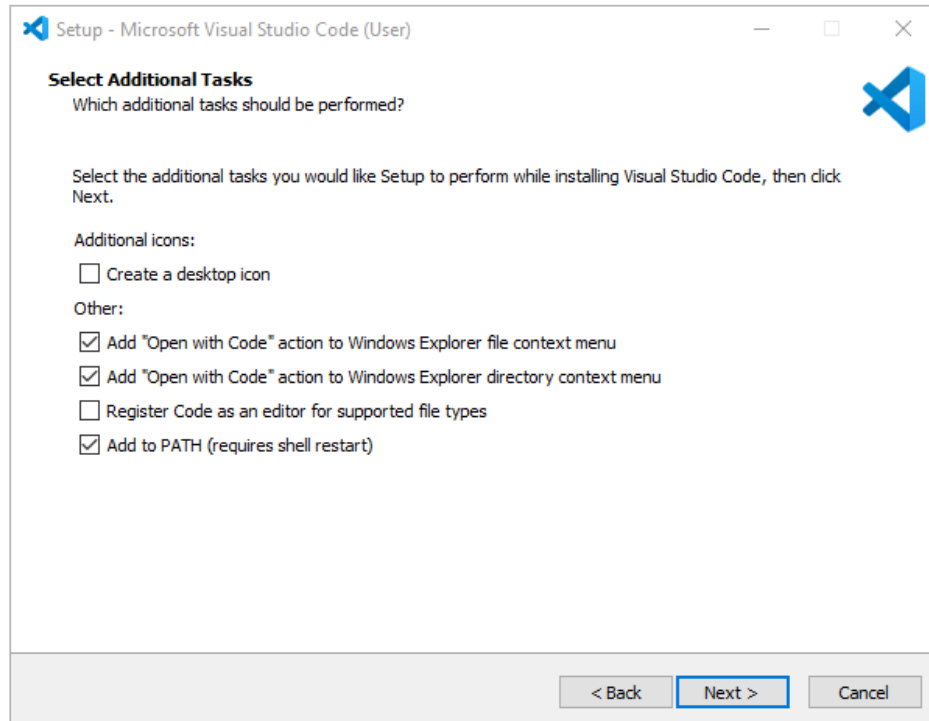
Figure 6. Install Visual Studio Code

2) Clone the **Raspberry Pi Pico SDK** and WIZnet's **Azure IoT SDK example** using below commands
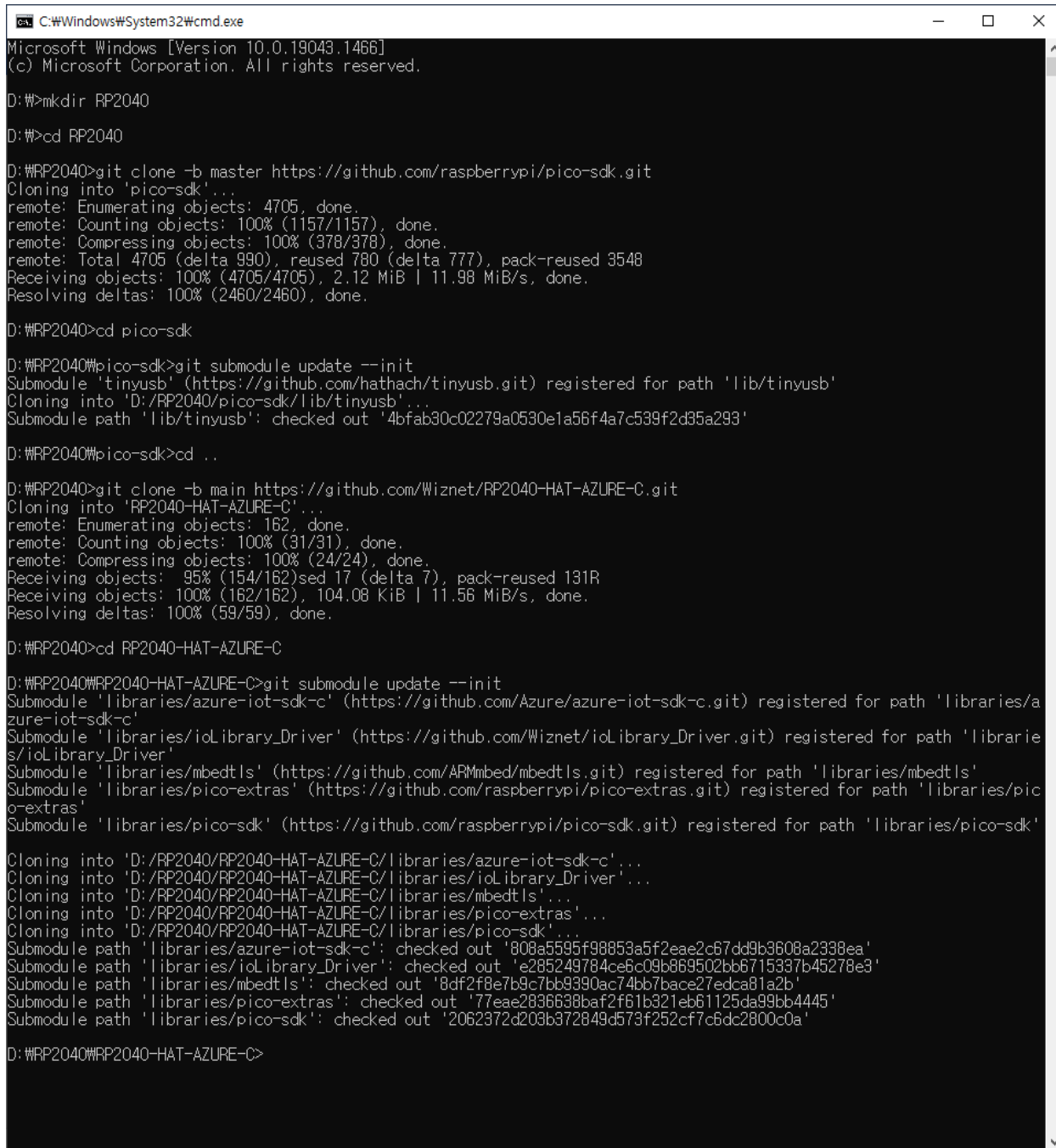
- Raspberry Pi Pico SDK : https://github.com/raspberrypi/pico-sdk
- Azure IoT SDK example : https://github.com/Wiznet/RP2040-HAT-AZURE-C

```
// create a project directory
D:\>mkdir RP2040
D:\>cd RP2040

// get the SDK
D:\RP2040> git clone -b master https://github.com/raspberrypi/pico-sdk.git
D:\RP2040> cd pico-sdk
D:\RP2040\pico-sdk> git submodule update --init

// get the example
D:\RP2040\pico-sdk> cd ..
D:\RP2040> git clone -b main https://github.com/Wiznet/RP2040-HAT-AZURE-
C.git
D:\RP2040> cd RP2040-HAT-AZURE-C
```

```
D:\RP2040\RP2040-HAT-AZURE-C> git submodule update --init
```


Figure 7. Get the SDK and example

3) Set up Visual Studio Code

    ① Open a new Visual Studio 2019 Developer Command Prompt

    ② Run the below command to open Visual Studio Code

```
D:> code -n
```

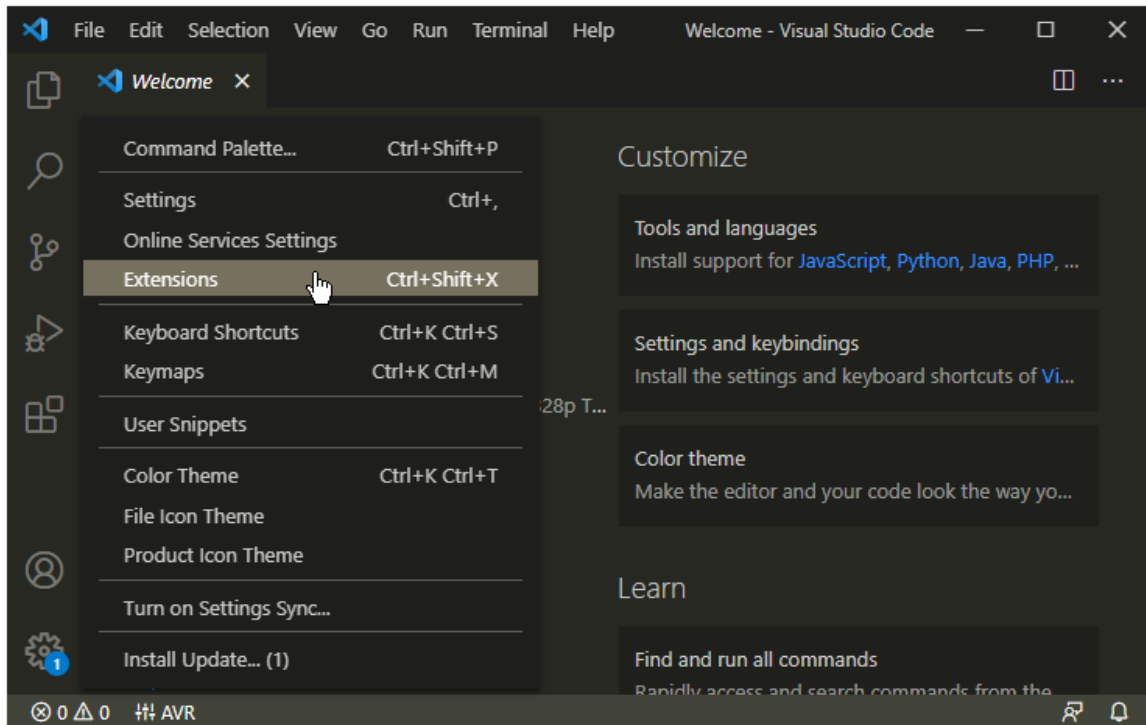③ Opening Visual Studio Code from Developer Command Prompt

④ Open Extensions

Figure 8. Install Extensions in Visual Studio Code

⑤ Install some tools
- CMake Tools
- C/C++

⑥ Open CMake Tools Extension Settings

⑦ Scroll down and set up some items
- Add Cmake: Configure Environment Item as **PICO_SDK_PATH**
- Add Cmake: Configure Environment Value as **D:\RP2040\pico-sdk**
- Add Cmake: Generator as **NMake Makefiles**
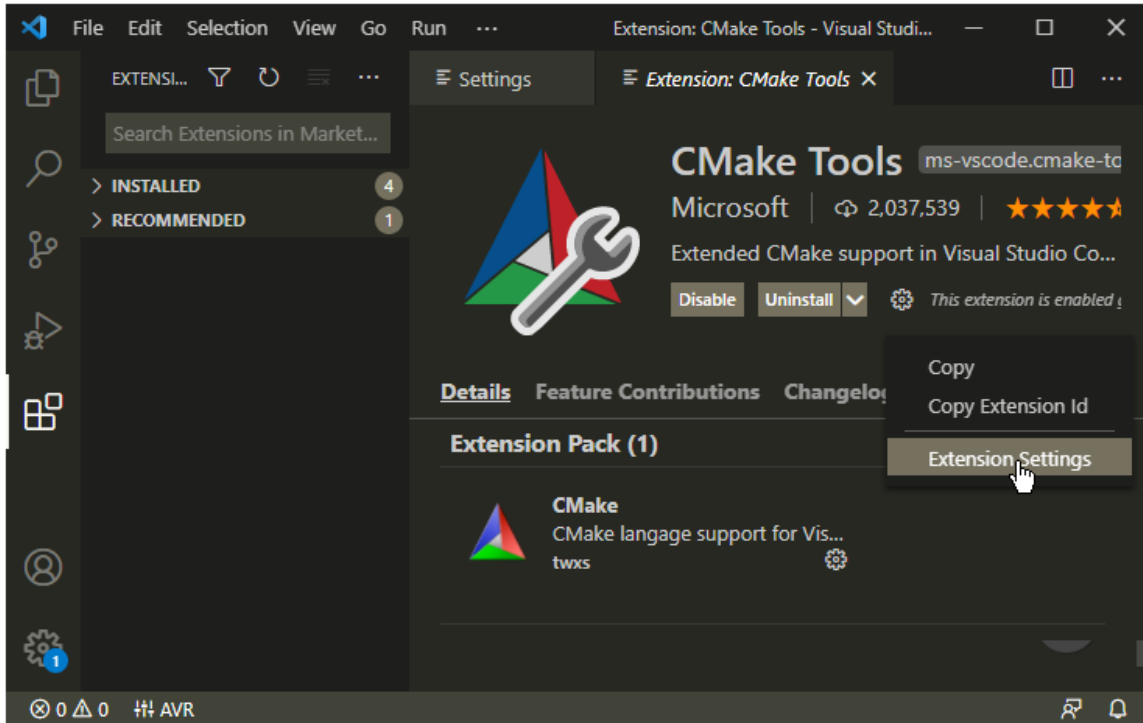
Figure 9. CMake Tools Extension Settings in Visual Studio Code
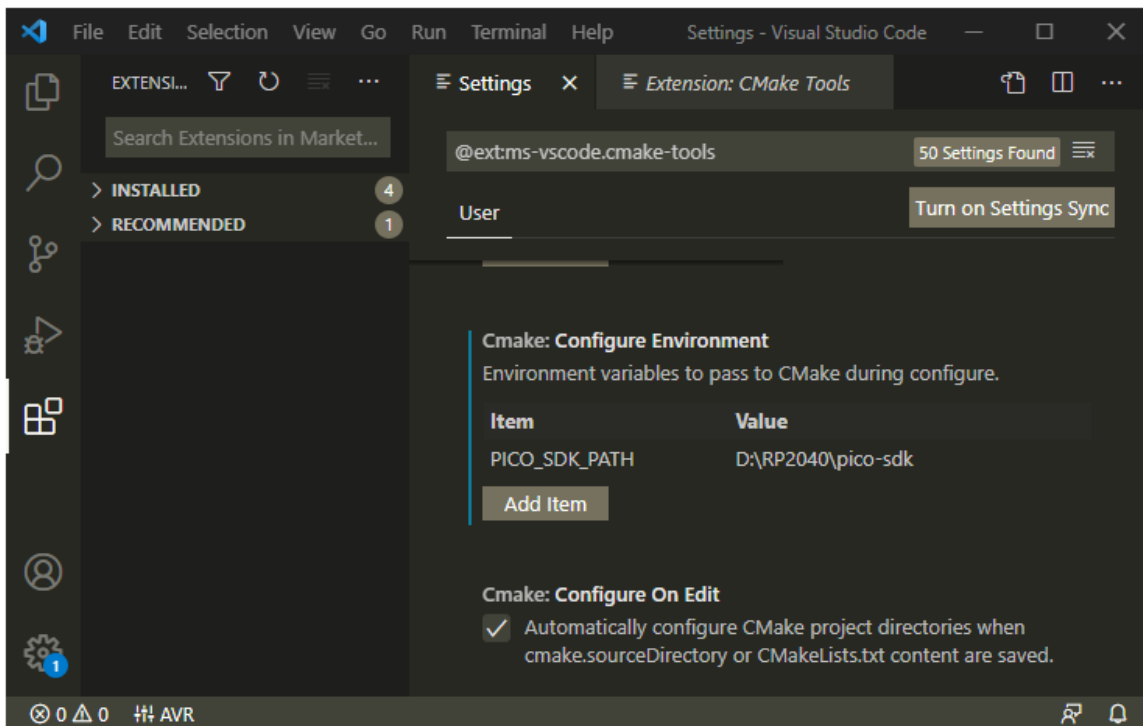
Figure 10. Add CMake Configure Environment path
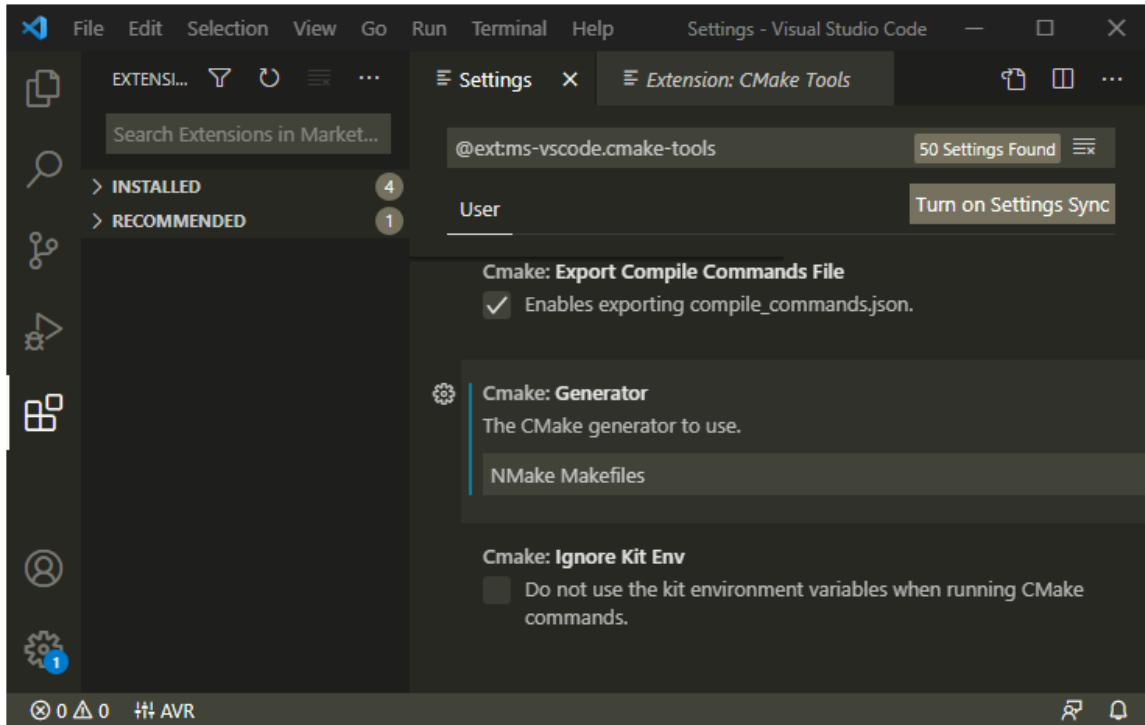
Figure 11. Add CMake Generator name

⑧ Add folder RP2040-HAT-AZURE-C to Visual Studio Code

⑨ Visual Studio Code will scan for kits

⑩ Select 'Yes' when asked: Would you like to configure project RP2040-HAT-AZURE-C?

⑪ Select 'Yes' if you like to configure the project upon opening

⑫ Click CMake in the bottom bar to select the kit RP2040-HAT-AZURE-C

⑬ Select Debug / Release based on your preference

⑭ Visual Studio Code will save all file and configure the project

⑮ Click on Build to build all examples, if no error Build will finish with exit code 0

3.1.2   Other software required to develop and debug applications for the device

Serial terminal program is required for operation check and debugging.

·   [Tera Term](#)

You may use your preferred serial terminal program.

3.1.3   Additional software references

Refer to the '**9.2. Building on MS Windows**' section of '**Getting started with Raspberry Pi Pico**' guide document below to set up the development environment.

- ·   [Getting started with Raspberry Pi Pico](#)

3.2   Setup your IoT Hub

Refer to the '**Create an IoT hub**' section of '**Quickstart: Set up the IoT Hub Device Provisioning Service with the Azure portal**' guide below to set up the IoT Hub.

- ·   [Quickstart: Set up the IoT Hub Device Provisioning Service with the Azure portal](#)

3.3   Provision your device over DPS

Refer to the '**Create a new IoT Hub Device Provisioning Service**' section of '**Quickstart: Set up the IoT Hub Device Provisioning Service with the Azure portal**' guide below to set up the DPS.

- ·   [Quickstart: Set up the IoT Hub Device Provisioning Service with the Azure portal](#)

After completing the DPS settings according to the guide document above, some additional settings are required.

1) Generate the certificate in X.509 format required when provisioning the device
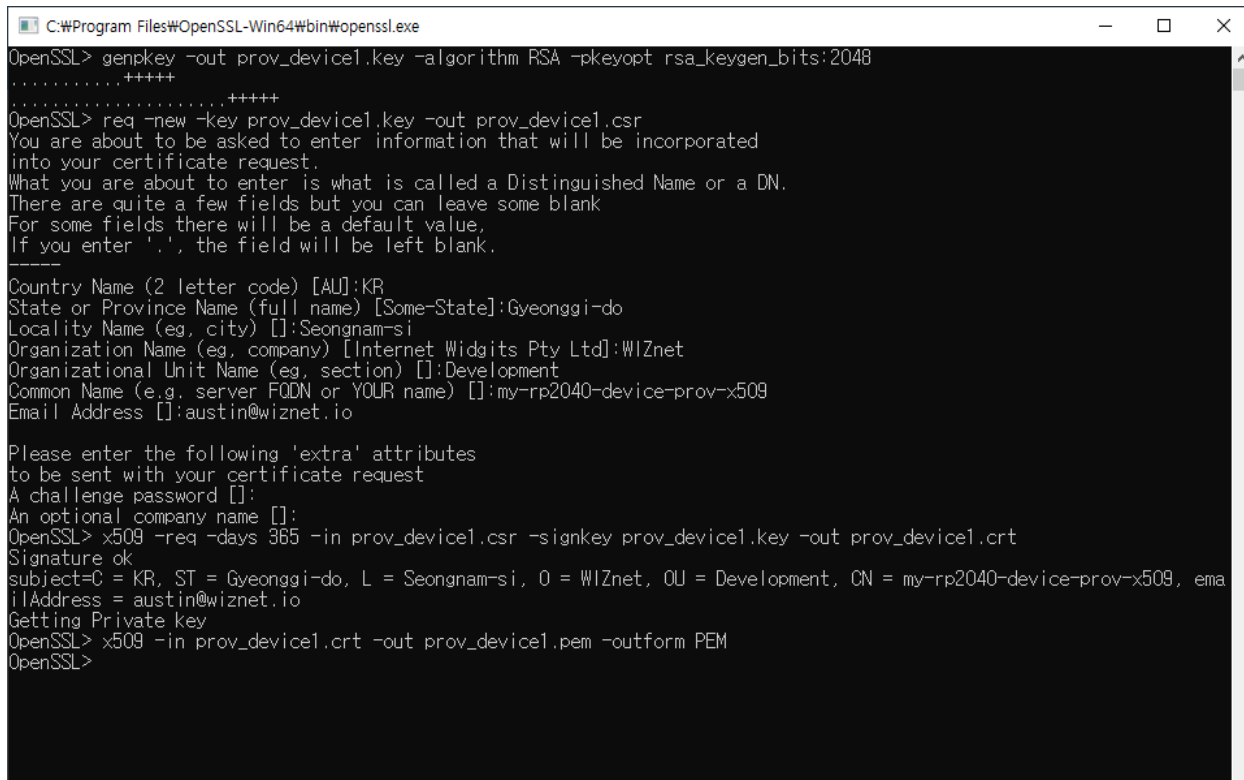
Generate the certificate in X.509 format that is required when provisioning the device using OpenSSL.

For the OpenSSL commands to generate the certificate in X.509 format, refer to the following.

```
/* Generate the certificate */
OpenSSL> genpkey -out [key name].key -algorithm RSA -pkeyopt
rsa_keygen_bits:2048
OpenSSL> req -new -key [key name].key -out [csr name].csr
OpenSSL> x509 -req -days 365 -in [csr name].csr -signkey [key name].key -
out [crt name].crt
```

```
OpenSSL> x509 -in [crt name].crt -out [pem name].pem -outform PEM

// e.g.
OpenSSL> genpkey -out prov_device1.key -algorithm RSA -pkeyopt
rsa_keygen_bits:2048
OpenSSL> req -new -key prov_device1.key -out prov_device1.csr
OpenSSL> x509 -req -days 365 -in prov_device1.csr -signkey prov_device1.key
-out prov_device1.crt
OpenSSL> x509 -in prov_device1.crt -out prov_device1.pem -outform PEM
```


Figure 12. generate certificate in X.509

If OpenSSL is not installed, you can download it from the link below and install it, or you can use another preferred program to generate the certificate in X.509 format.

- OpenSSL

2) Set up Enrollment and register the generated the certificate in X.509 format
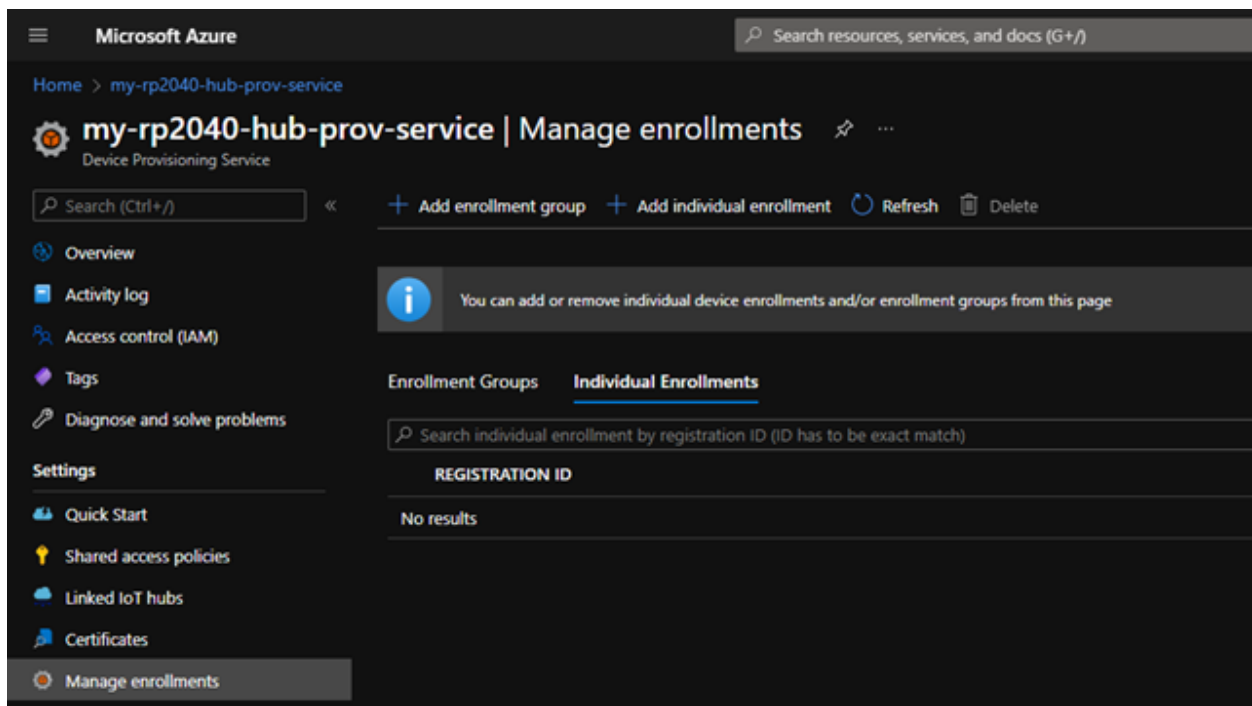
Figure 13. Add Individual Enrollment

Figure 14. Set up enrollment items
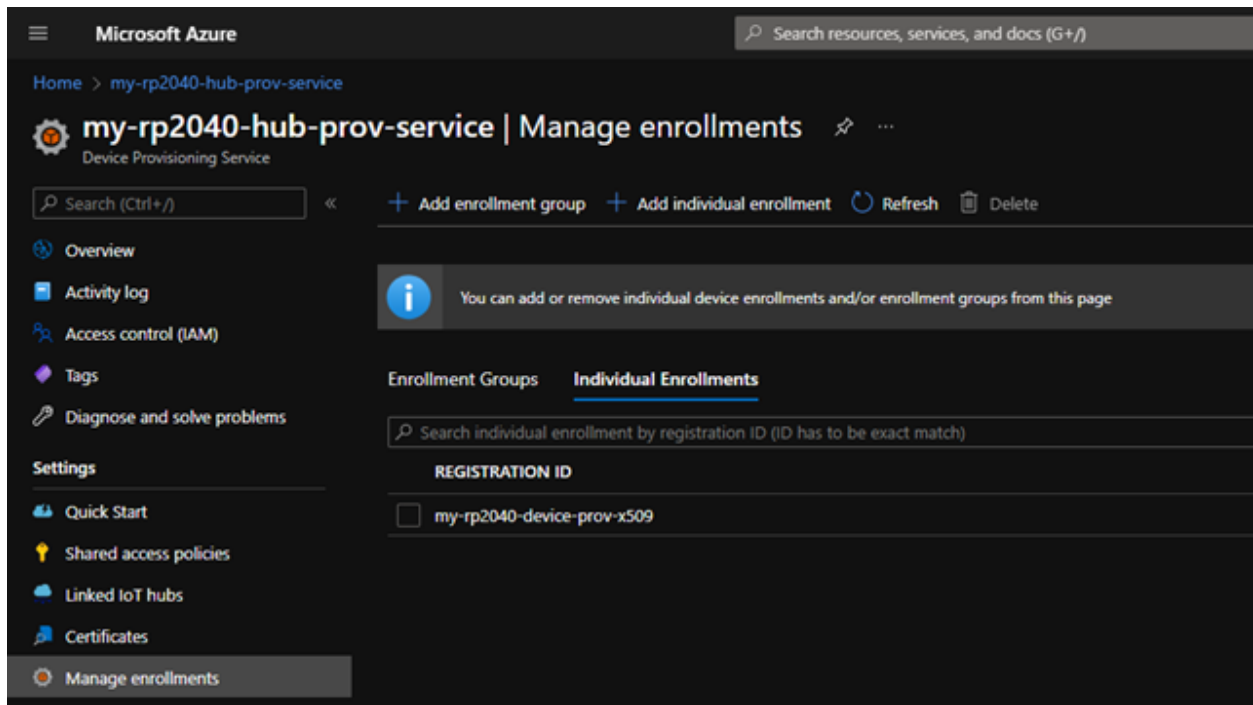
Figure 15. Registered Individual Enrollment

4   Prepare your Device

4.1   Connect with ethernet cable

Connect ethernet cable to device ethernet port.

4.2   Connect with 5 pin micro USB cable.

Connect device to desktop or laptop using 5 pin micro USB cable.

5   Build SDK and Run Samples

5.1   Select sample application

Since you need to use the **prov_dev_client_ll_sample** sample application, uncomment the
following in '**main.c**' in the '**RP2040-HAT-AZURE-C\examples**' directory to configure the
sample application.

```
/**
  * -------------------------------------------------------------------------
  * Macros
  * -------------------------------------------------------------------------
  */
// The application you wish to use should be uncommented
//
//#define APP_TELEMETRY
//#define APP_C2D
//#define APP_CLI_X509
#define APP_PROV_X509
```
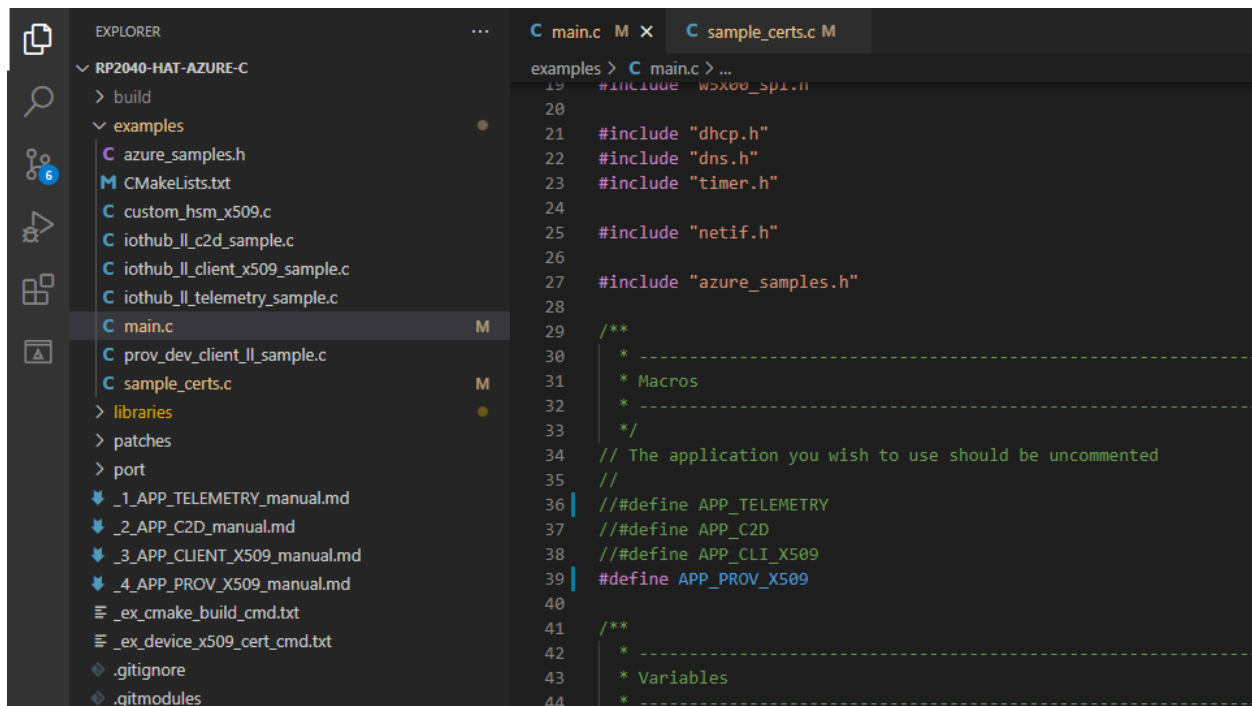

Figure 16. Set up sample application

5.2   Enter certificate and DPS related information

For provisioning the device to the IoT Hub through DPS and connecting to IoT Hub, enter **ID scope**, **common name**, and **certificate in X.509 format** generated and registered in *section 3.3*.

The ID scope can be checked in the Device Provisioning Service set in the Azure Portal.
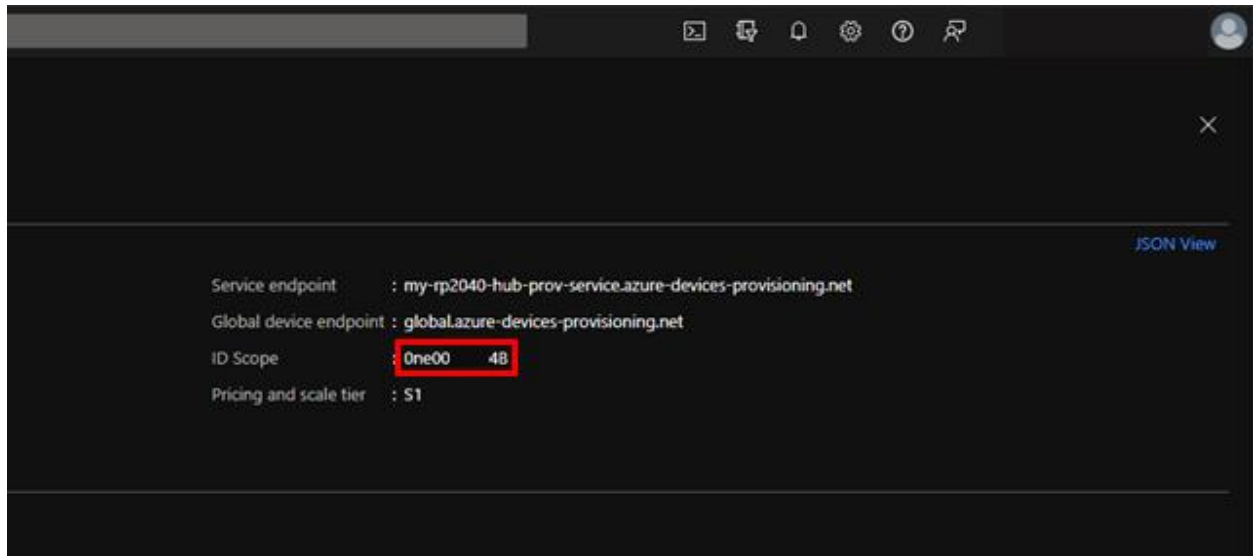
Figure 17. ID scope

The common name is one of the information entered when generating the certificate in X.509 format in OpenSSL.

Figure 18. common name

Set up the above information in '**sample_certs.c**' in the '**RP2040-HAT-AZURE-C\examples**' directory.

```c
const char pico_az_id_scope[] = "[ID Scope]";

const char pico_az_COMMON_NAME[] = "[custom-hsm-device]";

const char pico_az_CERTIFICATE[] =
"-----BEGIN CERTIFICATE-----""\n"
"-----END CERTIFICATE-----";

const char pico_az_PRIVATE_KEY[] =
"-----BEGIN PRIVATE KEY-----""\n"
"-----END PRIVATE KEY-----";
```
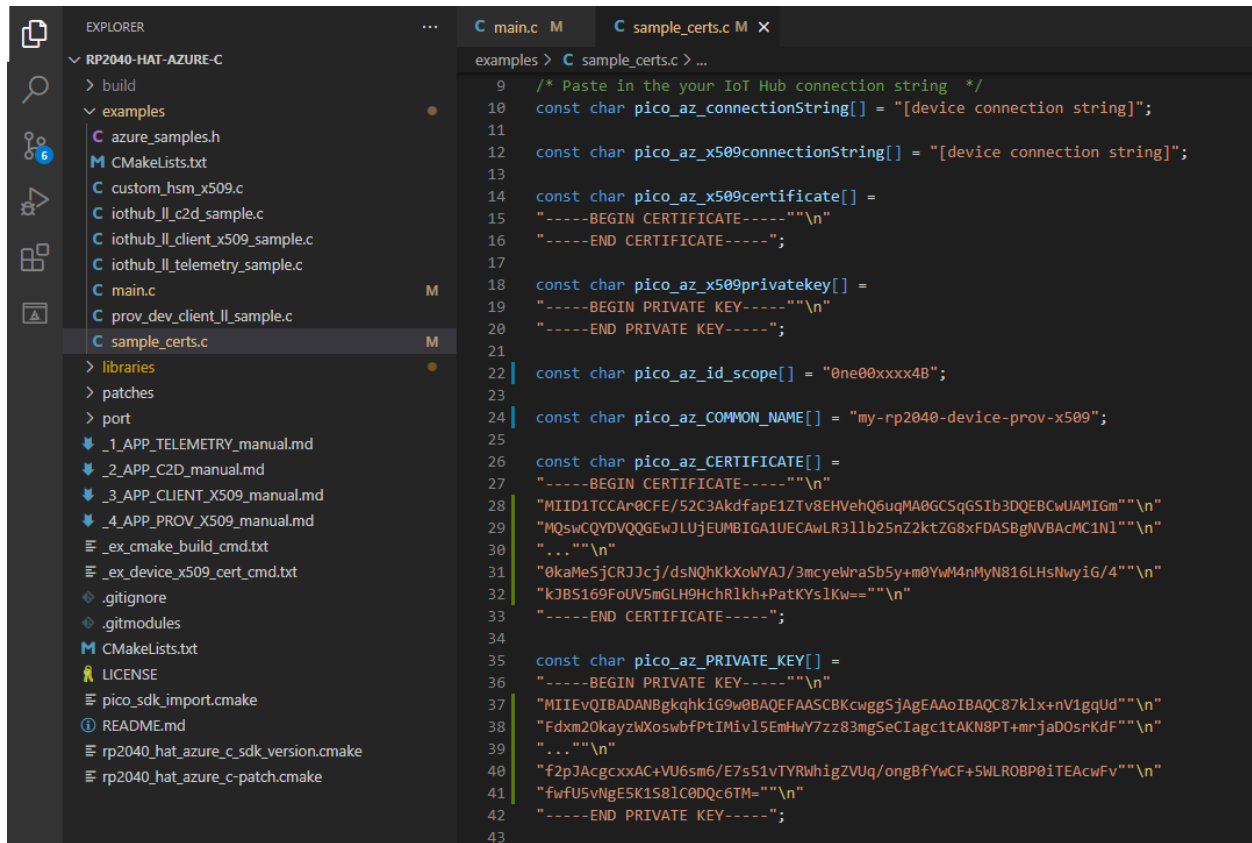
Figure 19. Set up 'sample_certs.c'

5.3 Build example

1) After completing the Azure IoT SDK example configuration, click 'build' in the status bar at the bottom of Visual Studio Code or press the 'F7' button on the keyboard to build.

2) When the build is completed, '**main.uf2**' is generated in '**RP2040-HAT-AZURE-C\build\examples**' directory.

5.4 Upload firmware

1) While pressing the **BOOTSEL** button of the device power on the board, the USB mass storage '**RPI-RP2**' is automatically mounted.
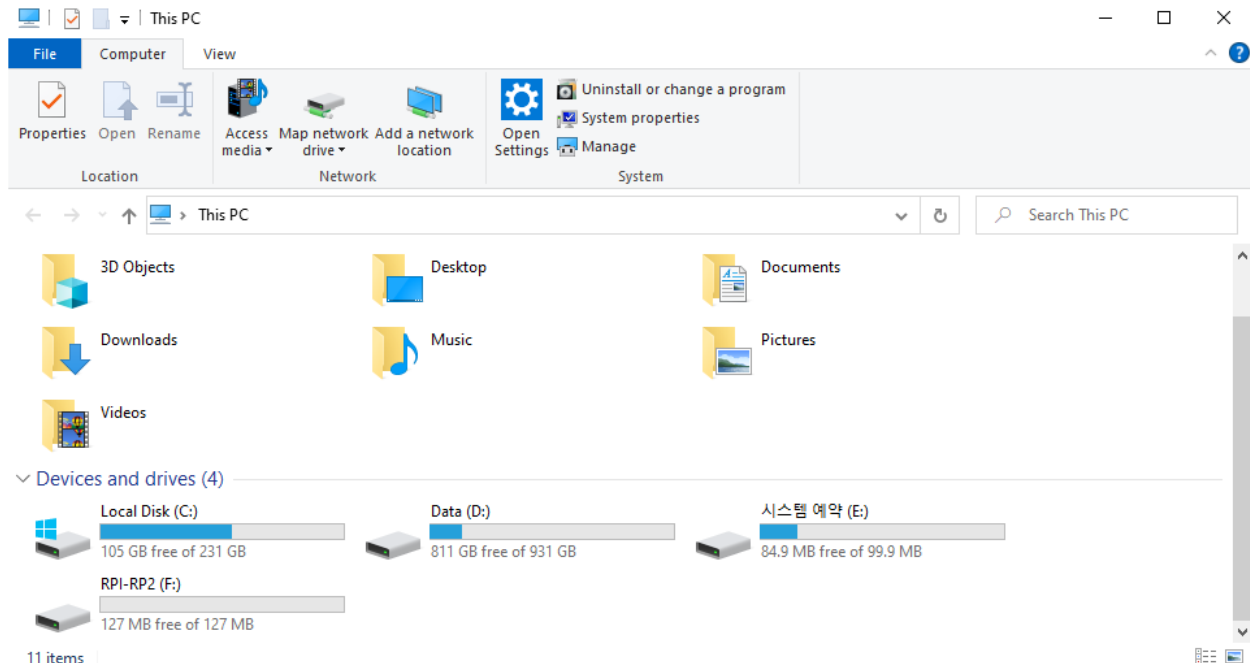
Figure 20. Automatically mounted USB mass storage 'RPI-RP2'

2) Drag and drop '**main.uf2**' onto the USB mass storage device '**RPI-RP2**'.

5.5   Run sample application

First, connect to the serial COM port of the device with terminal program.

When connecting to the serial COM port of the device, use following settings to set up the serial port.

- ·   Baud rate : 115,200
- ·   Data bit : 8
- ·   Parity bit : none
- ·   Stop bit : 1
- ·   Flow control : none
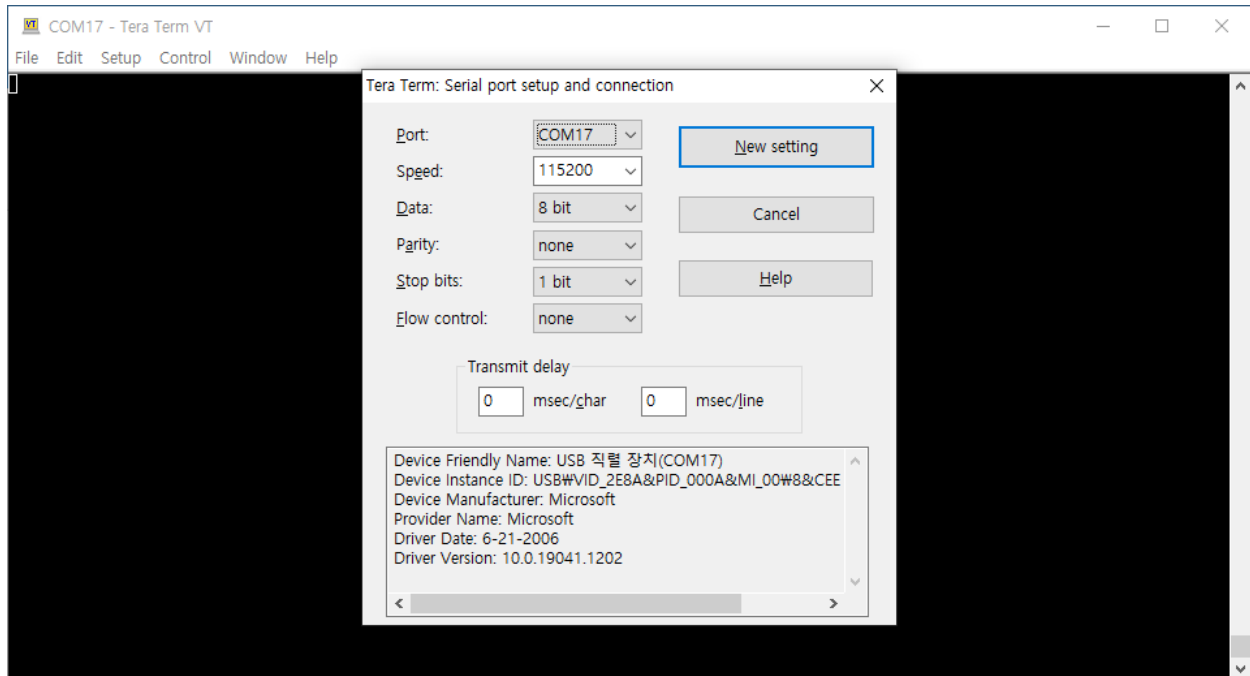
Figure 21. set up serial port

If sample application is running normally, in terminal you should be able to see the device network information, registration to the IoT Hub through DPS, and sending data from the device to the IoT Hub.
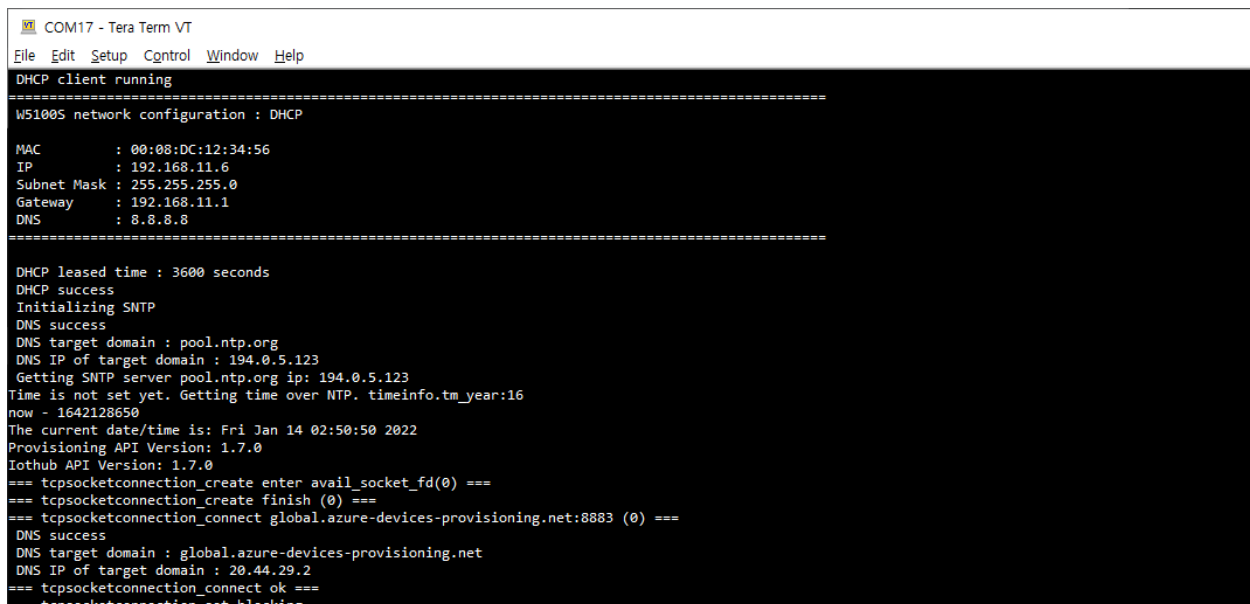
Figure 22. Device network information

Figure 23. Registration to IoT Hub through DPS

Figure 24. Sending data from device to IoT Hub

6 Integration with Azure IoT Explorer

6.1 Run Azure IoT explorer

Run Azure IoT explorer.

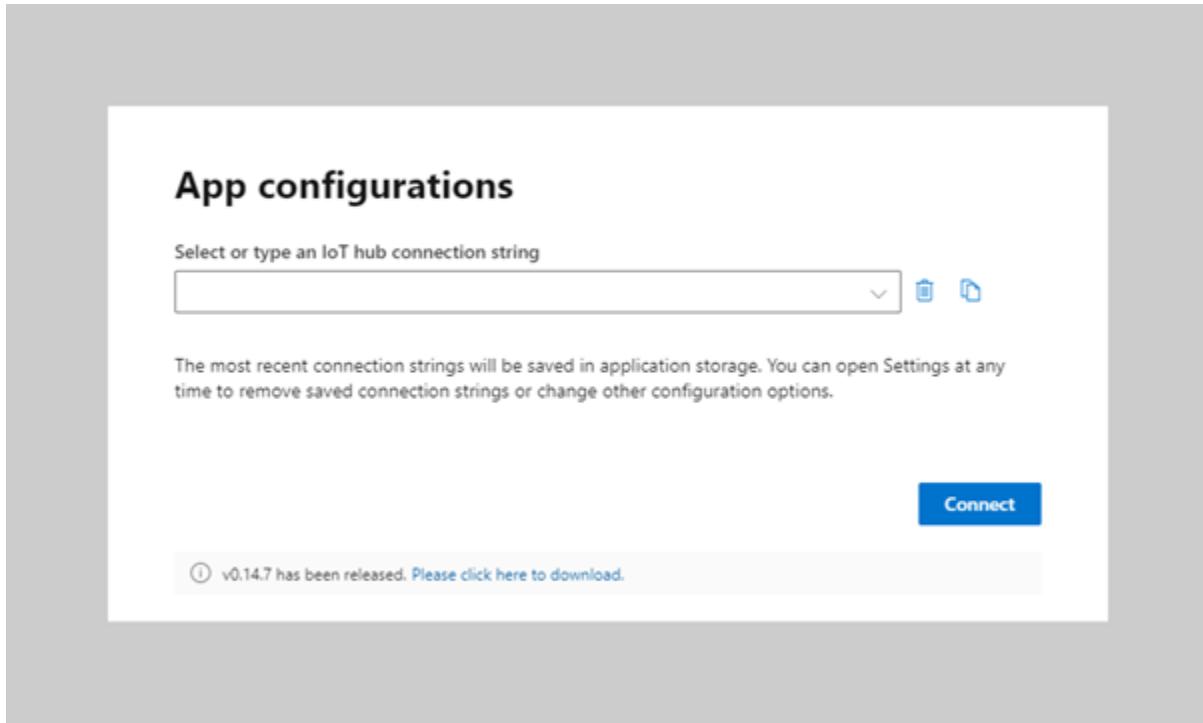*Figure 25. Run Azure IoT explorer*

If Azure IoT explorer is not installed, download the latest version Azure IoT explorer from the link below and install it.

- Azure IoT explorer

6.2 Enter IoT Hub connection string and connect to Azure IoT explorer

Enter the connection string of the IoT Hub set up in *section 3.2* and press the connect button to connect to Azure IoT explorer.
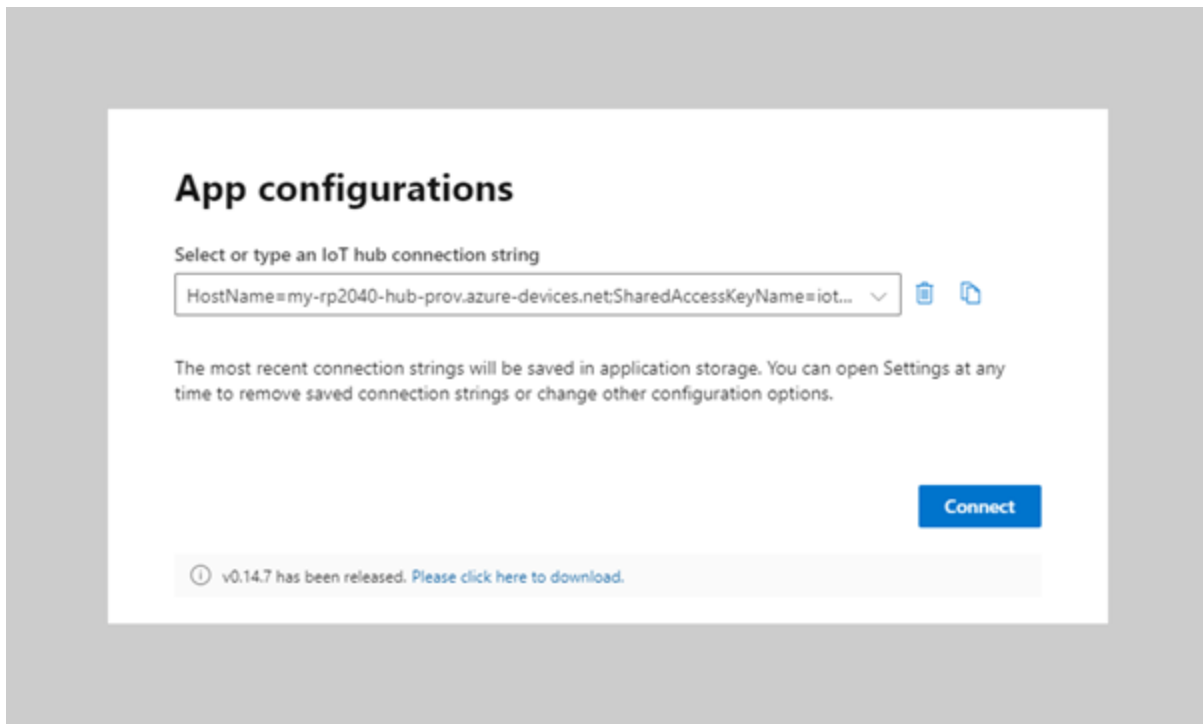
Figure 26. Enter connection string of IoT Hub

6.3   Run device

If the device is running normally, in Azure IoT explorer you should be able to see the device registering to the IoT Hub through DPS, and receiving data sent from the device to the IoT Hub.

Figure 27. Registered device

Figure 28. Receive data sent from device to IoT Hub

7 Additional Links

Basic example of the W5100S-EVB-Pico is also provided. If necessary, please refer to the example of the link below.

- RP2040-HAT-C

8 Troubleshooting

If you have any questions or problems while testing W5100S-EVB-Pico examples, please post them at the links below.

- · [WIZnet Developer Forum](#)
- · [RP2040-HAT-C Issues](#)
- · [RP2040-HAT-AZURE-C Issues](#)