

W5500-EVB-Pico

Getting Started Guide

for AWS IoT Core

Version 1.0.0



<http://www.wiznet.io/>

Table of Contents

1	<i>Document Information</i>	<i>3</i>
2	<i>Overview</i>	<i>4</i>
3	<i>Hardware Description.....</i>	<i>4</i>
4	<i>Set up your Development Environment</i>	<i>6</i>
5	<i>Set up your hardware</i>	<i>16</i>
6	<i>Setup your AWS account and Permissions.....</i>	<i>18</i>
7	<i>Create Resources in AWS IoT.....</i>	<i>19</i>
8	<i>Provision the Device with credentials</i>	<i>19</i>
9	<i>Build the demo</i>	<i>20</i>
10	<i>Run the demo</i>	<i>20</i>
11	<i>Debugging</i>	<i>25</i>
12	<i>Troubleshooting</i>	<i>29</i>

1 Document Information

1.1 Revision History (Version, Date, Description of change)

Version	Date	Description of change
V1.0.0	2022-06-23	Initial Release

2 Overview

The W5500-EVB-Pico is a microcontroller evaluation board based on the Raspberry Pi RP2040 microcontroller chip and full hardwired TCP/IP controller W5500 chip. The W5500-EVB-Pico has the same role as the Raspberry Pi Pico platform and includes W5500, so the Ethernet function is basically included.

- Raspberry Pi Pico Clone
- Ethernet (W5500 Hardwired TCP/IP CHIP)

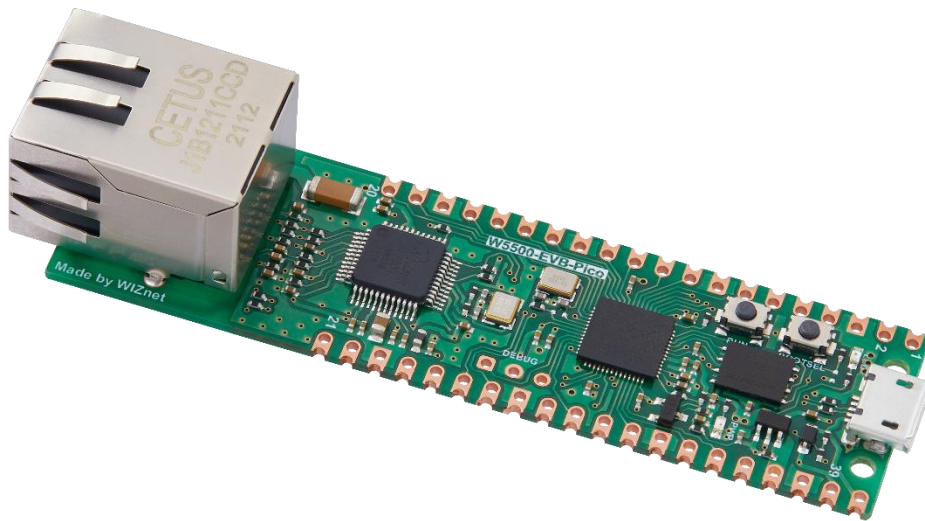


Figure 1. W5500-EVB-Pico

3 Hardware Description

3.1 DataSheet

https://docs.wiznet.io/assets/images/w5500_evb_pico_schematic-69ae2bc3858a5c48bc88c17b04f30967.png

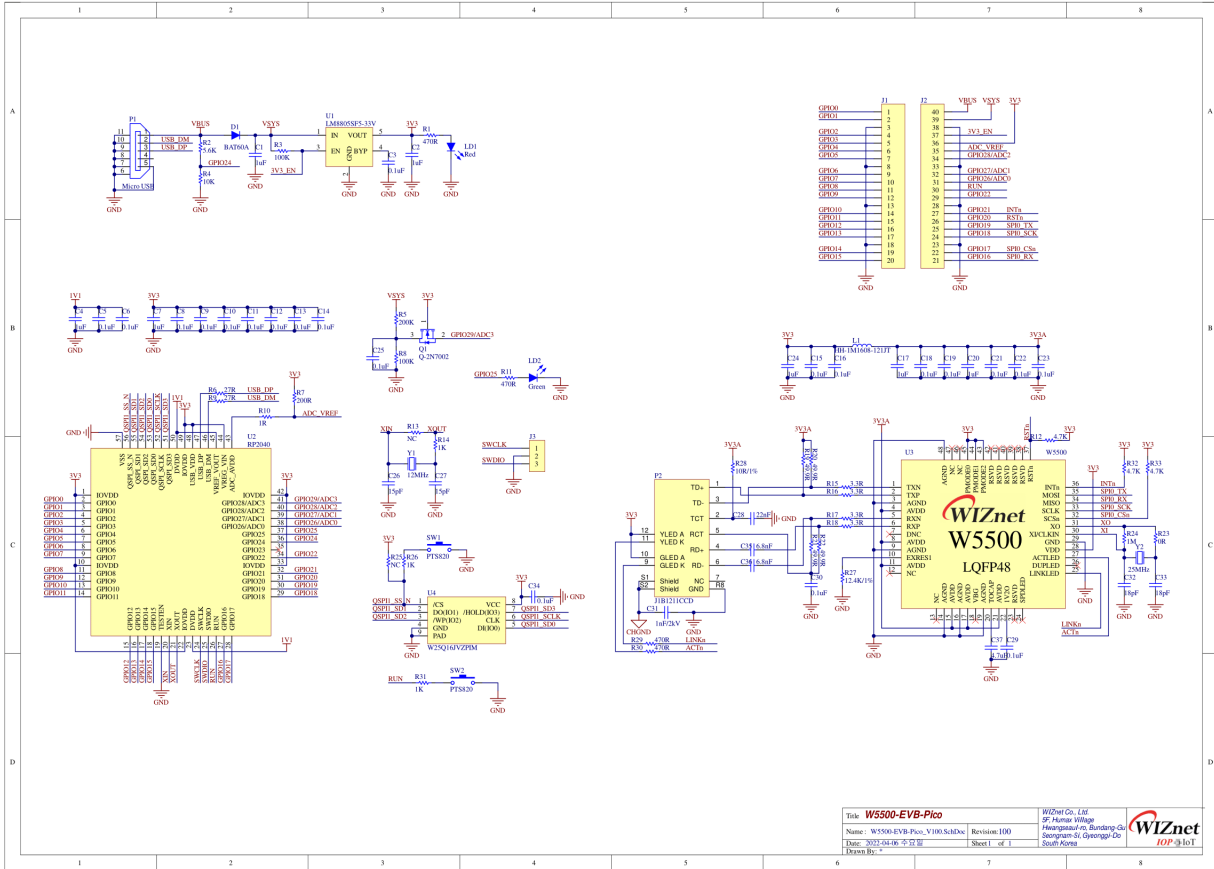


Figure 2. W5500-EVB-Pico schematic

3.2 Standard Kit Contents

- W5500-EVB-Pico : 1EA
- 1 x 20 2.54mm pitch pin header : 2EA

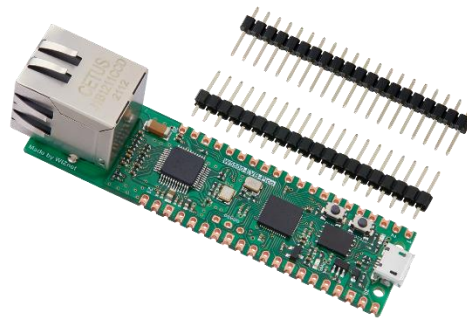


Figure 3. W5500-EVB-Pico package

3.3 User Provided items

- USB type A to USB micro B cable



Figure 4. USB type A to USB micro B cable

- Ethernet cable
- Desktop or laptop

3.4 3rd Party purchasable items

None

3.5 Additional Hardware References

<https://docs.wiznet.io/Product/iEthernet/W5500/w5500-evb-pico>

4 Set up your Development Environment

4.1 Tools Installation (IDEs, Toolchains, SDKs)

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 4.4 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

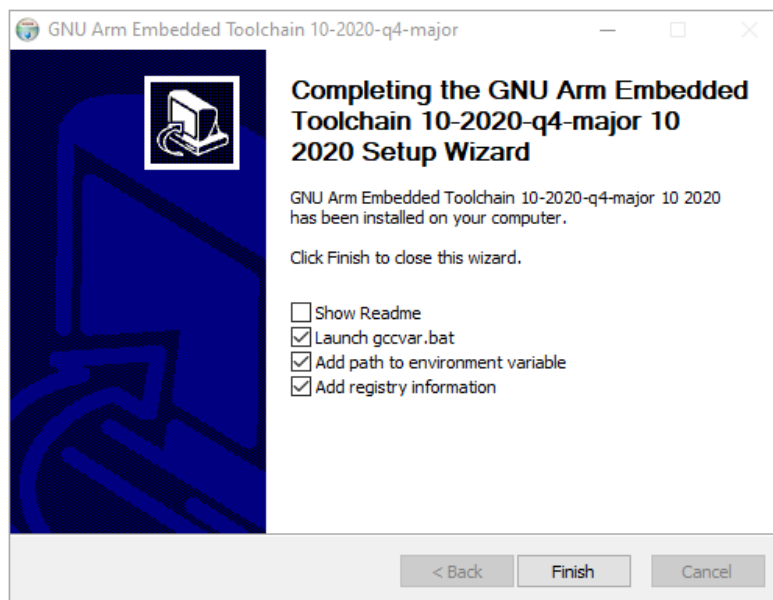


Figure 5. Install ARM GCC compiler

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

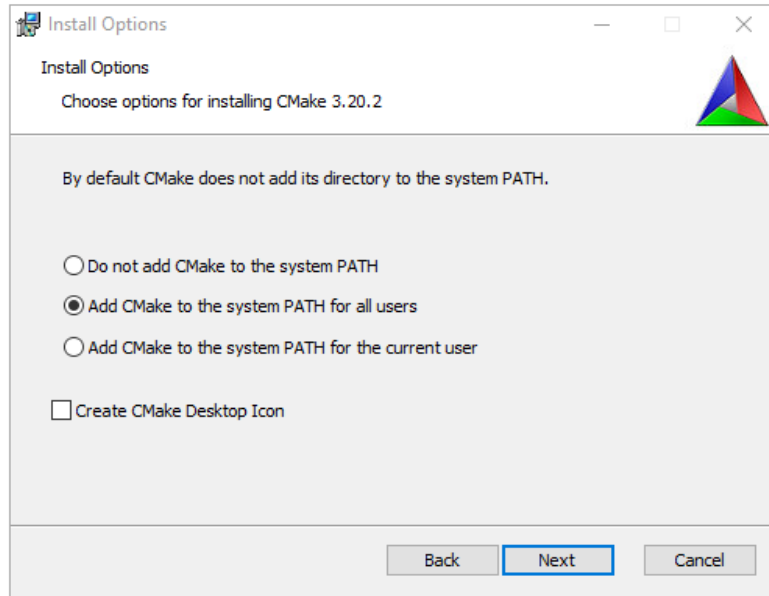


Figure 6. Install CMake

During the installation add CMake to the system PATH for all users when prompted by the installer.

③ Install Build Tools for Visual Studio

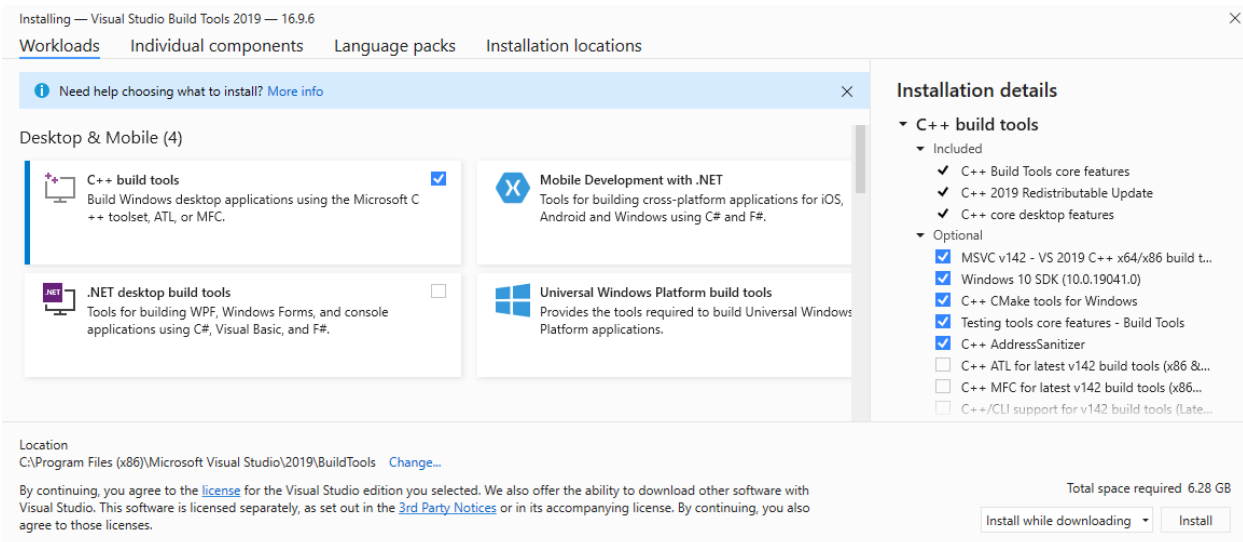


Figure 7. Install Build Tools for Visual Studio

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.

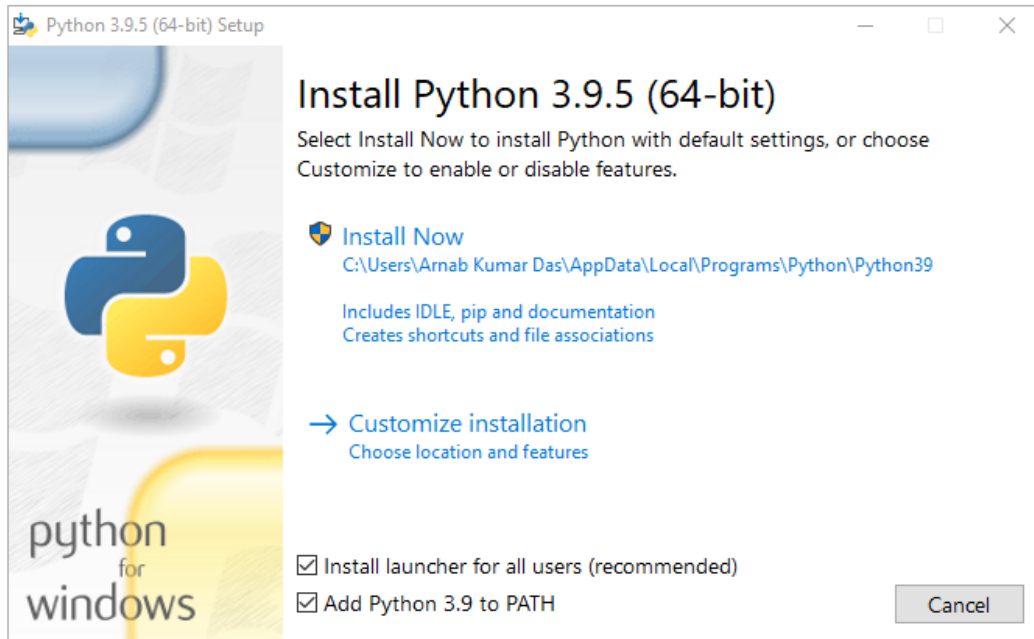


Figure 8. Install Python

⑤ Install Git

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

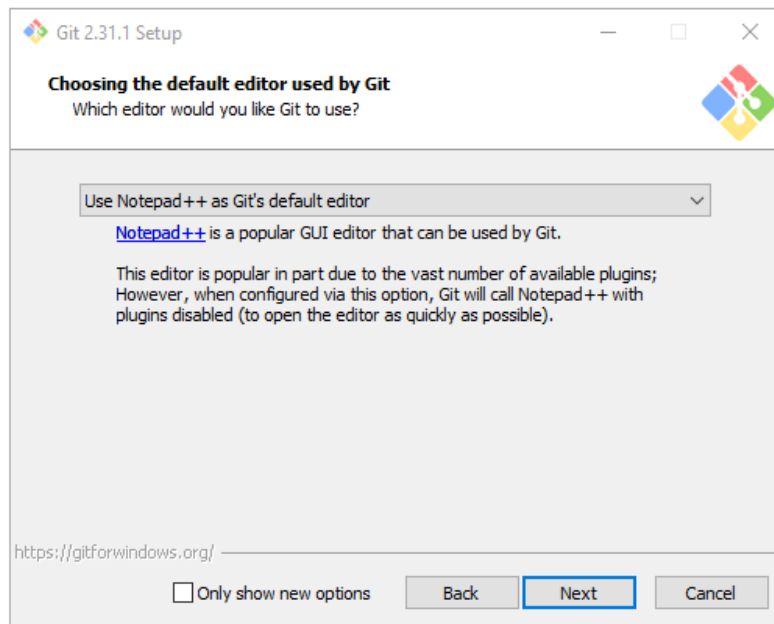


Figure 9. Install Git

⑥ Install Visual Studio Code

During the installation add Visual Studio Code to the system PATH.

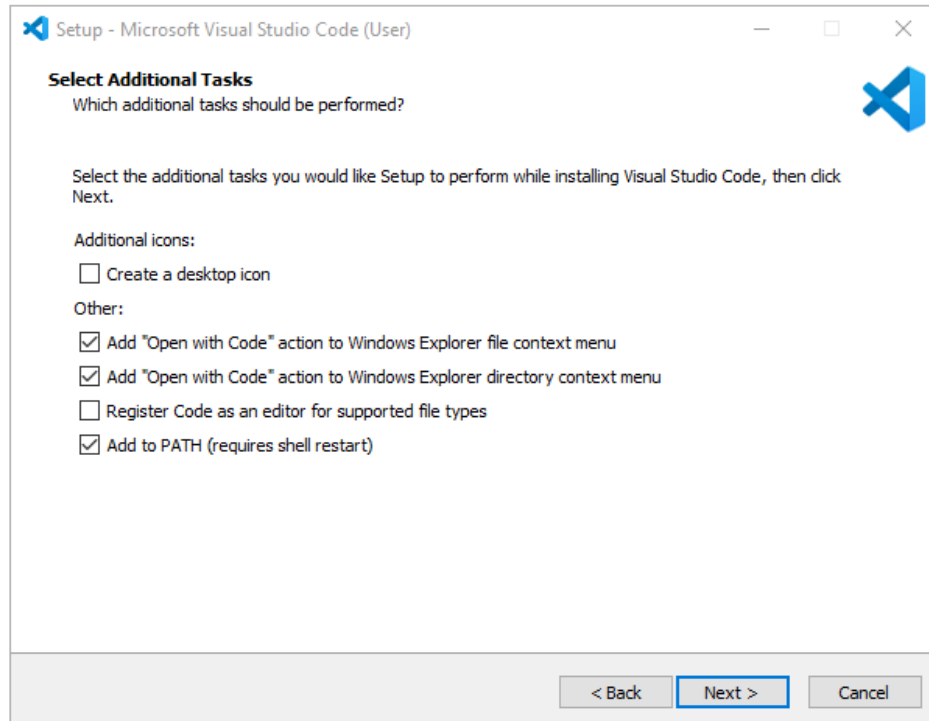


Figure 10. Install Visual Studio Code

2) Clone the Raspberry Pi Pico SDK and WIZnet example repository using below commands

- SDK : <https://github.com/raspberrypi/pico-sdk>
- Example : <https://github.com/Wiznet/RP2040-HAT-AWS-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-AWS-C.git
D:\RP2040> cd RP2040-HAT-AWS-C
D:\RP2040\RP2040-HAT-AWS-C> git submodule update --init
```

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1348]
(c) Microsoft Corporation. All rights reserved.

D:\>mkdir RP2040

D:\>cd RP2040

D:\RP2040>git clone -b master https://github.com/raspberrypi/pico-sdk.git
Cloning into 'pico-sdk'...
remote: Enumerating objects: 4600, done.
remote: Counting objects: 100% (1797/1797), done.
remote: Compressing objects: 100% (888/888), done.
remote: Total 4600 (delta 1074), reused 1208 (delta 747), pack-reused 2803
Receiving objects: 100% (4600/4600), 2.39 MiB | 23.06 MiB/s, done.
Resolving deltas: 100% (2254/2254), done.

D:\RP2040>cd pico-sdk

D:\RP2040#pico-sdk>git submodule update --init
Submodule 'tinysub' (https://github.com/hathach/tinysub.git) registered for path 'lib/tinysub'
Cloning into 'D:/RP2040/pico-sdk/lib/tinysub'...
Submodule path 'lib/tinysub': checked out '4bfab30c02279a0530e1a56f4a7c539f2d35a293'

D:\RP2040#pico-sdk>cd ..

D:\RP2040>git clone -b main https://github.com/Wiznet/RP2040-HAT-AWS-C.git
Cloning into 'RP2040-HAT-AWS-C'...
remote: Enumerating objects: 3045, done.
remote: Counting objects: 100% (3045/3045), done.
remote: Compressing objects: 100% (2097/2097), done.
remote: Total 3045 (delta 931), reused 2989 (delta 881), pack-reused 0
Receiving objects: 100% (3045/3045), 8.93 MiB | 23.86 MiB/s, done.
Resolving deltas: 100% (931/931), done.

D:\RP2040>cd RP2040-HAT-AWS-C

D:\RP2040#RP2040-HAT-AWS-C>git submodule update --init
Submodule 'libraries/aws-iot-device-sdk-embedded-C' (https://github.com/aws/aws-iot-device-sdk-embedded-C.git) registered for path 'libraries/aws-iot-device-sdk-embedded-C'
Cloning into 'D:/RP2040/RP2040-HAT-AWS-C/libraries/aws-iot-device-sdk-embedded-C'...
Submodule 'libraries/IO_Library_Driver' (https://github.com/Wiznet/IO_Library_Driver.git) registered for path 'libraries/IO_Library_Driver'
Cloning into 'D:/RP2040/RP2040-HAT-AWS-C/libraries/IO_Library_Driver'...
Submodule 'libraries/mbedtls' (https://github.com/ARMmbed/mbedtls.git) registered for path 'libraries/mbedtls'
Cloning into 'D:/RP2040/RP2040-HAT-AWS-C/libraries/mbedtls'...
Submodule path 'libraries/aws-iot-device-sdk-embedded-C': checked out '9de6cc4e2cfda4cfff8b6a5a3df5b9ab3a90e8f2c'
Submodule path 'libraries/IO_Library_Driver': checked out 'e285249784ce6c09b869502bb6715337b45278e3'
Submodule path 'libraries/mbedtls': checked out '662deb38d61bb1fc6392c55a5134d1bd1a116118'

D:\RP2040#RP2040-HAT-AWS-C>

```

Figure 11. 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
```

- ③ Opening Visual Studio Code from Developer Command Prompt
- ④ Open Extensions

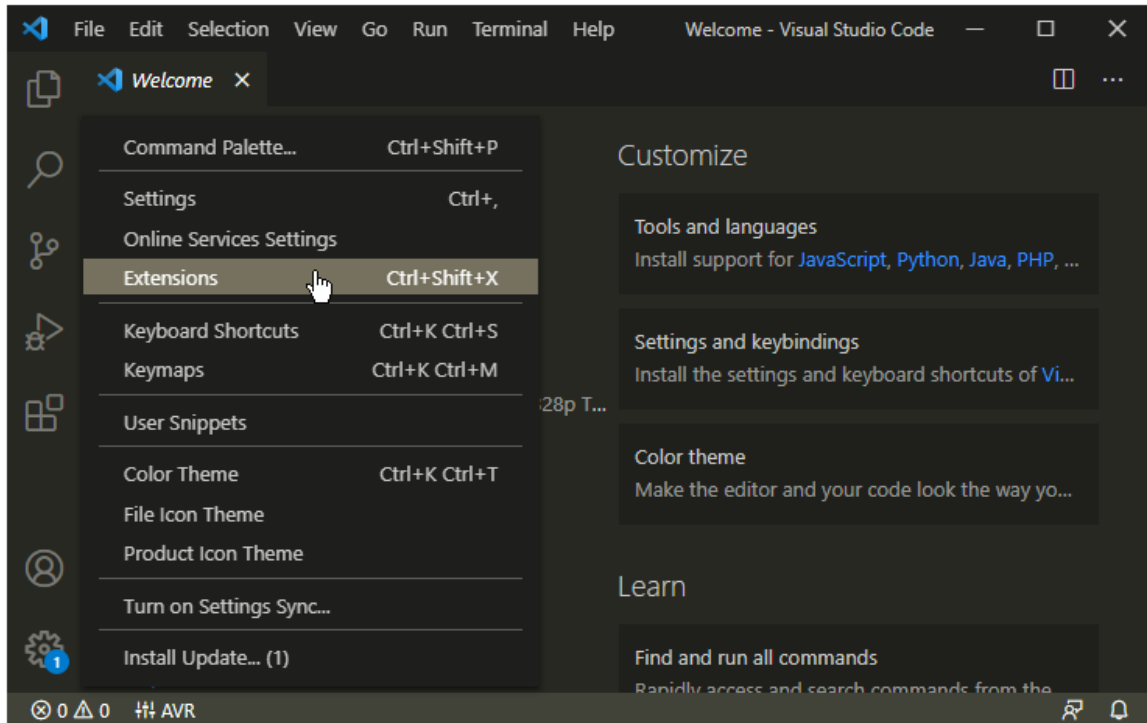


Figure 12. Install Extensions in Visual Studio Code

- 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

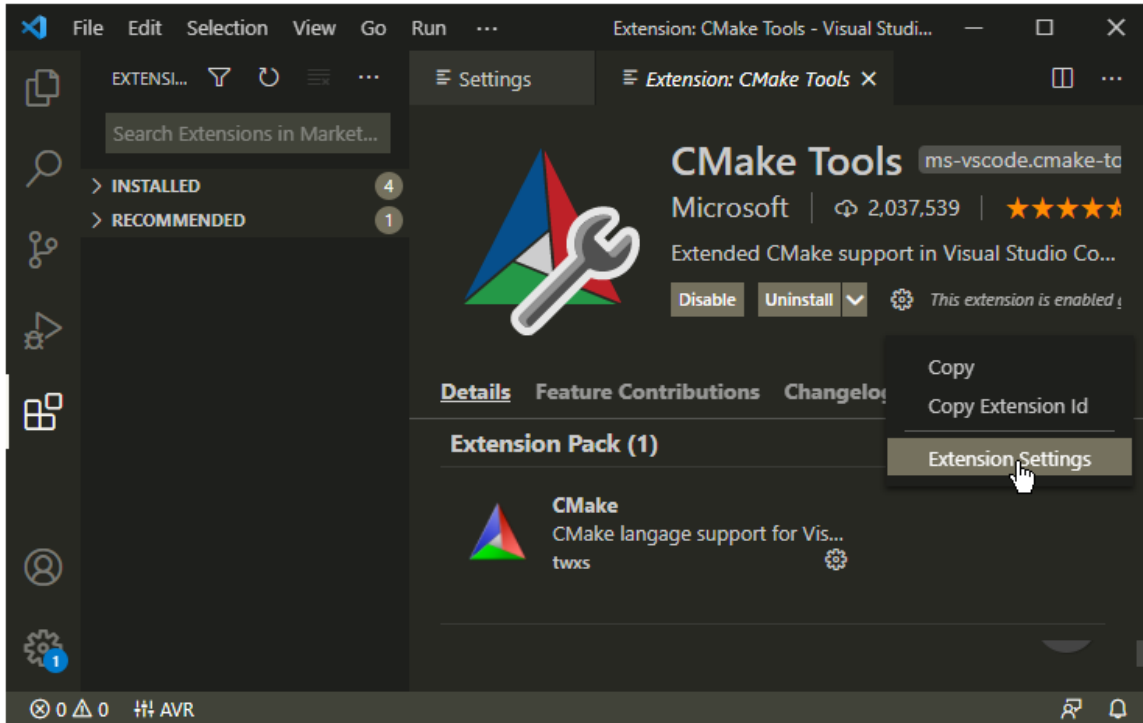


Figure 13. CMake Tools Extension Settings in Visual Studio Code

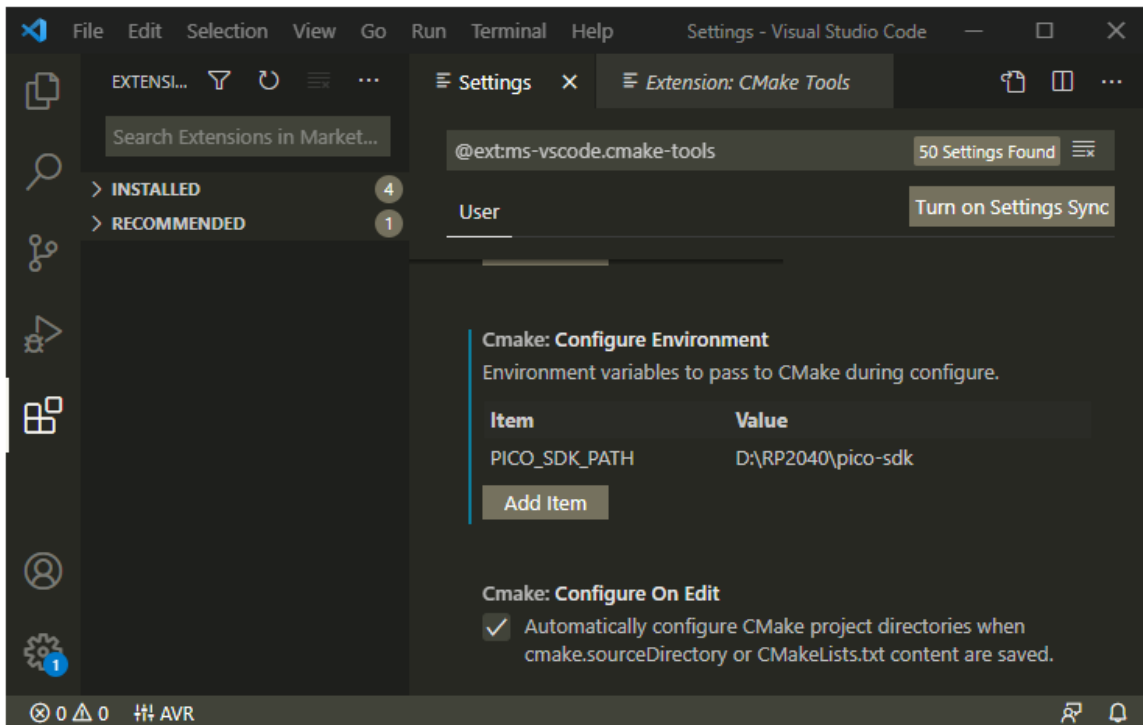


Figure 14. Add CMake Configure Environment path

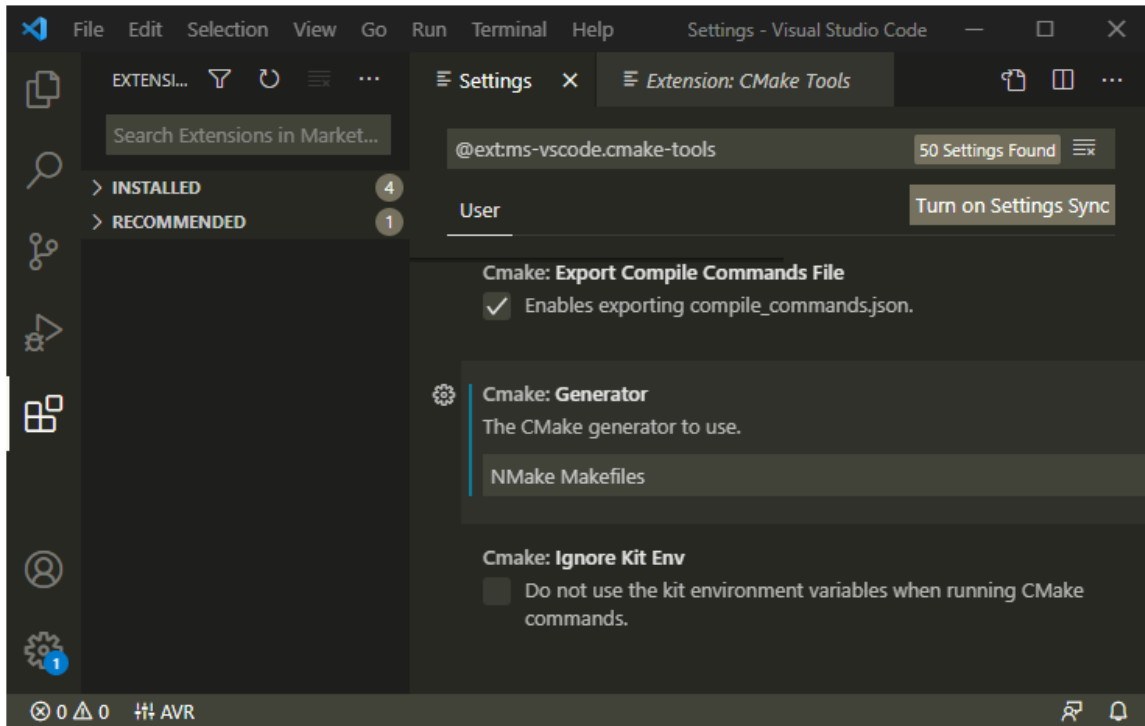


Figure 15. Add CMake Generator name

- ⑧ Add folder RP2040-HAT-AWS-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-AWS-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-AWS-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

4.2 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.

4.3 Other pre-requisites

None

4.4 Additional Software References

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

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

And refer to the links below for instructions how to setup environment and proceed with tests.

- [Getting Started with AWS IoT SDK Examples](#)
- [Connect AWS IoT through MQTT](#)

5 Set up your hardware

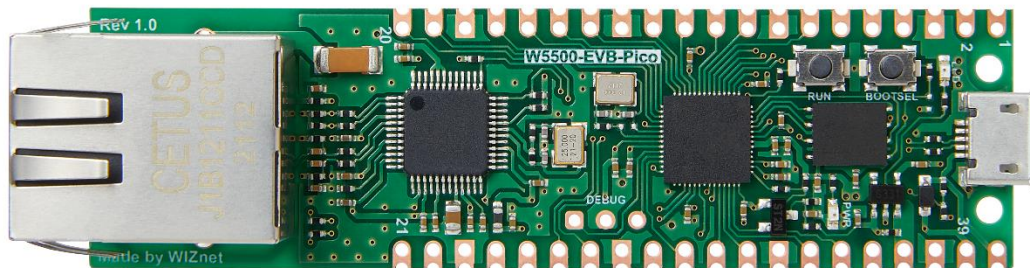


Figure 16. W5500-EVB-Pico front

The diagram illustrates the pinout for the W5500-EVB-Pico board, showing connections for various protocols and power. The board features a W5500 Ethernet controller and a WIZnet W5500 chip. The pin headers are numbered 1 to 40. The legend at the bottom right provides a color key for the pin functions:

- Power:** Red
- Ground:** Black
- UART (default):** Purple
- UART:** Light purple
- GPIO, PIO, and PWM:** Green
- ADC:** Dark green
- SPI:** Pink
- I2C:** Blue
- System Control:** Light pink
- Debugging:** Orange
- Connected to W5500 in board:** Dark blue

Copyright 2022 WIZnet Co., Ltd. All rights reserved.

W5500-EVB-Pico pin-out is directly connected to the GPIO of RP2040 as shown in the picture above. It has the same pinout as the Raspberry Pi Pico board. However, GPIO16, GPIO17, GPIO18, GPIO19, GPIO20, GPIO21 are connected to W5500 inside the board. These pins enable SPI communication with W5500 to use Ethernet function. If you are using the Ethernet function, these pins cannot be used for any other purpose.

The RP2040 GPIO used inside W5500-EVB-Pico is as follows.

I/O	Pin name	Description
I	GPIO16	Connected to MISO on W5500
O	GPIO17	Connected to CSn on W5500
O	GPIO18	Connected to SCLK on W5500
O	GPIO19	Connected to MOSI on W5500
O	GPIO20	Connected to RSTn on W5500
I	GPIO21	Connected to INTn on W5500
I	GPIO24	VBUS sense - high if VBUS is present, else low
O	GPIO25	Connected to user LED
I	GPIO29	Used in ADC mode (ADC3) to measure VSYS/3

Apart from GPIO and ground pins, there are 7 other pins on the main 40-pin interface.

Pin no.	Pin name	Description
PIN40	VBUS	Micro-USB input voltage, connected to micro-USB port pin 1. Nominally 5V.
PIN39	VSYS	Main system input voltage, which can vary in the allowed range 4.3V to 5.5V, and is used by the on-board LDO to generate the 3.3V.
PIN37	3V3_EN	Connects to the on-board LDO enable pin. To disable the 3.3V (which also de-powers the RP2040 and W5500), short this pin low.
PIN36	3V3	Main 3.3V supply to RP2040 and W5500, generated by the on-board LDO.
PIN35	ADC_VREF	ADC power supply (and reference) voltage, and is generated on W5500-EVB-Pico by filtering the 3.3V supply.
PIN33	AGND	Ground reference for GPIO26-29.
PIN30	RUN	RP2040 enable pin, To reset RP2040, short this pin low.

6 Setup your AWS account and Permissions

Refer to the instructions at [Set up your AWS Account](#). Follow the steps outlined in these sections to create your account and a user and get started:

- Sign up for an AWS account and
- Create a user and grant permissions.
- Open the AWS IoT console

Pay special attention to the Notes.

7 Create Resources in AWS IoT

Refer to the instructions at [Create AWS IoT Resources](#). Follow the steps outlined in these sections to provision resources for your device:

- Create an AWS IoT Policy
- Create a thing object

Pay special attention to the Notes.

8 Provision the Device with credentials

You need to enter the root certificate, client certificate and private key that were downloaded earlier.

Root certificate uses the RSA 2048 bit key, Amazon Root CA 1, and does not use the public key.

Device certificate and key can be set in 'mqtt_certificate.h' in 'RP2040-HAT-AWS-C/examples/aws_iot_mqtt/' directory.

```
uint8_t mqtt_root_ca[] =
"-----BEGIN CERTIFICATE-----\r\n"
"... \r\n"
"-----END CERTIFICATE-----\r\n";

uint8_t mqtt_client_cert[] =
"-----BEGIN CERTIFICATE-----\r\n"
"... \r\n"
"-----END CERTIFICATE-----\r\n";

uint8_t mqtt_private_key[] =
"-----BEGIN RSA PRIVATE KEY-----\r\n"
"... \r\n"
"-----END RSA PRIVATE KEY-----\r\n";
```

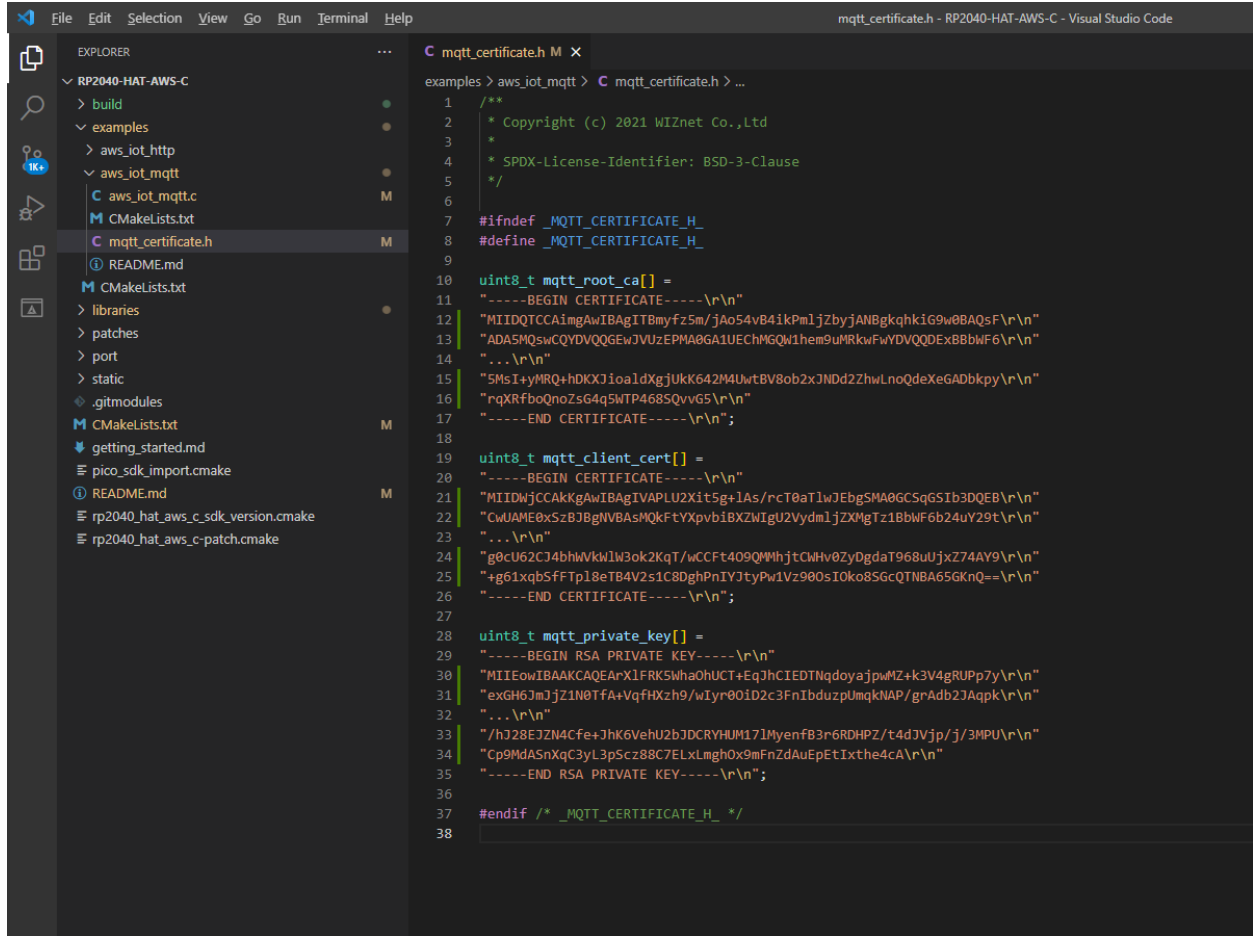


Figure 19. Set up certificates

9 Build the demo

- ① 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, 'aws_iot_mqtt.uf2' is generated in 'RP2040-HAT-AWS-C/build/examples/aws_iot_mqtt/' directory.

10 Run the demo

- ① While pressing the BOOTSEL button of Raspberry Pi Pico or W5500-EVB-Pico power on the board, the USB mass storage 'RPI-RP2' is automatically mounted.

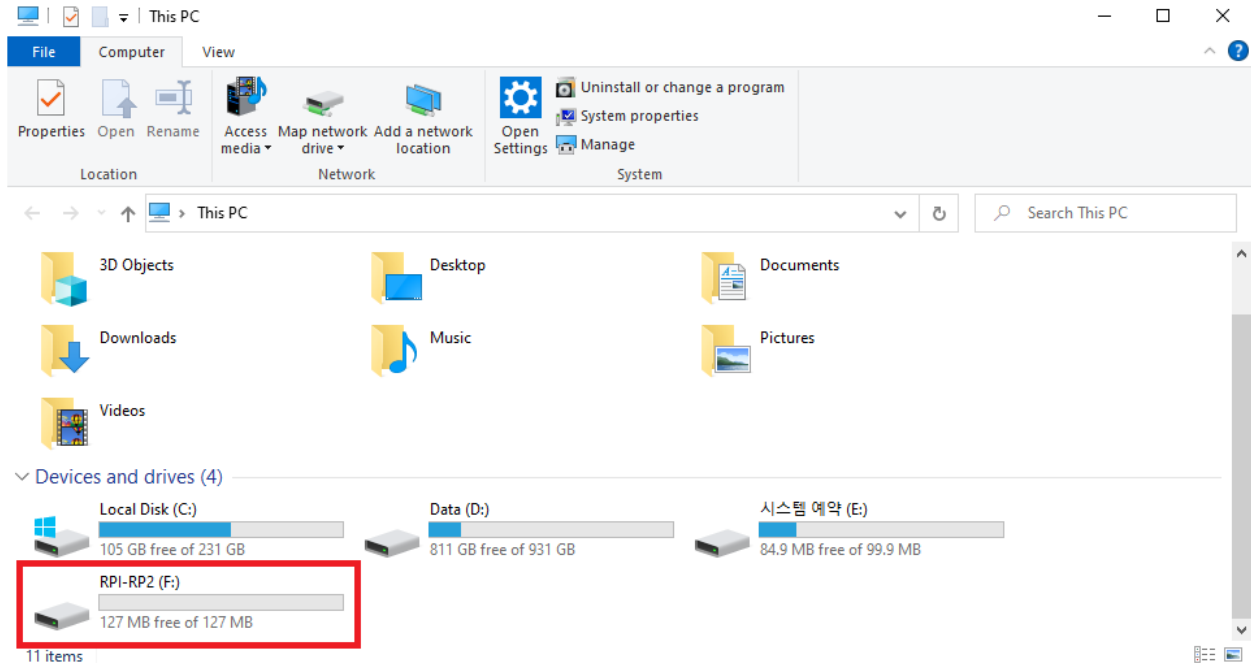


Figure 20. Automatically mounted USB mass storage 'RPI-RP2'

- ② Drag and drop 'aws_iot_mqtt.uf2' onto the USB mass storage device 'RPI-RP2'.
- ③ Connect to the serial COM port of Raspberry Pi Pico or W5500-EVB-Pico with Tera Term.

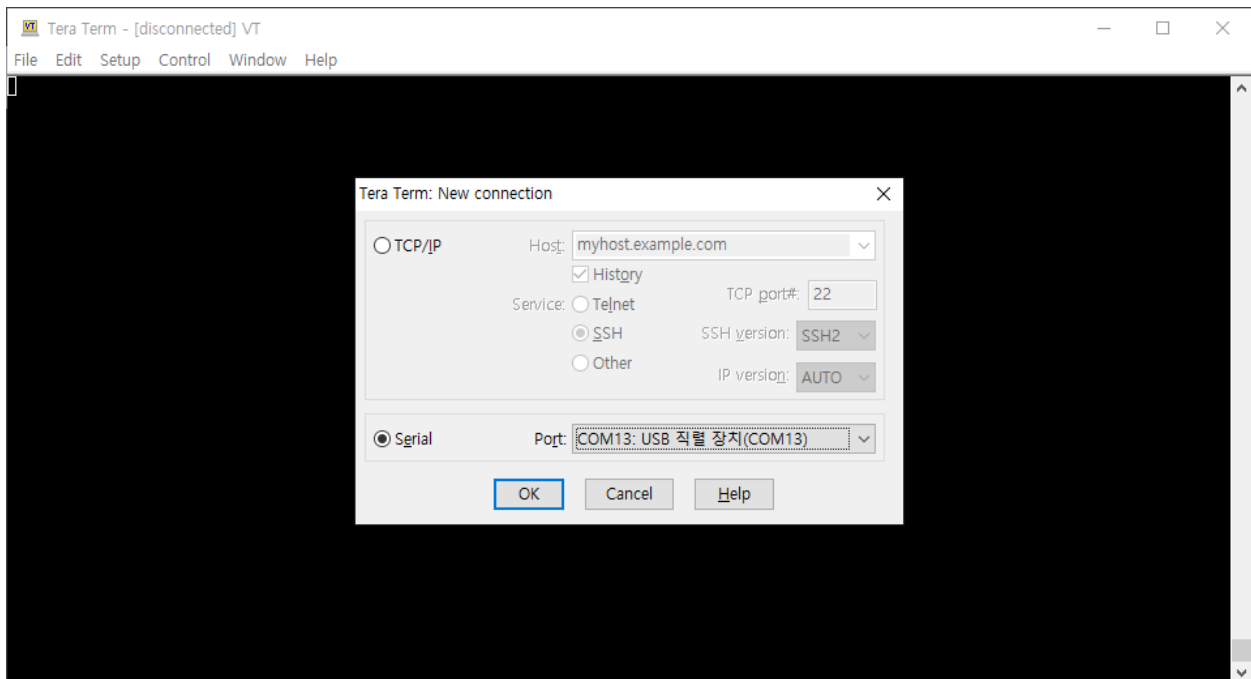


Figure 21. Connect to the serial COM port of W5500-EVB-Pico

④ When connecting to the serial COM port of W5500-EVB-Pico, 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

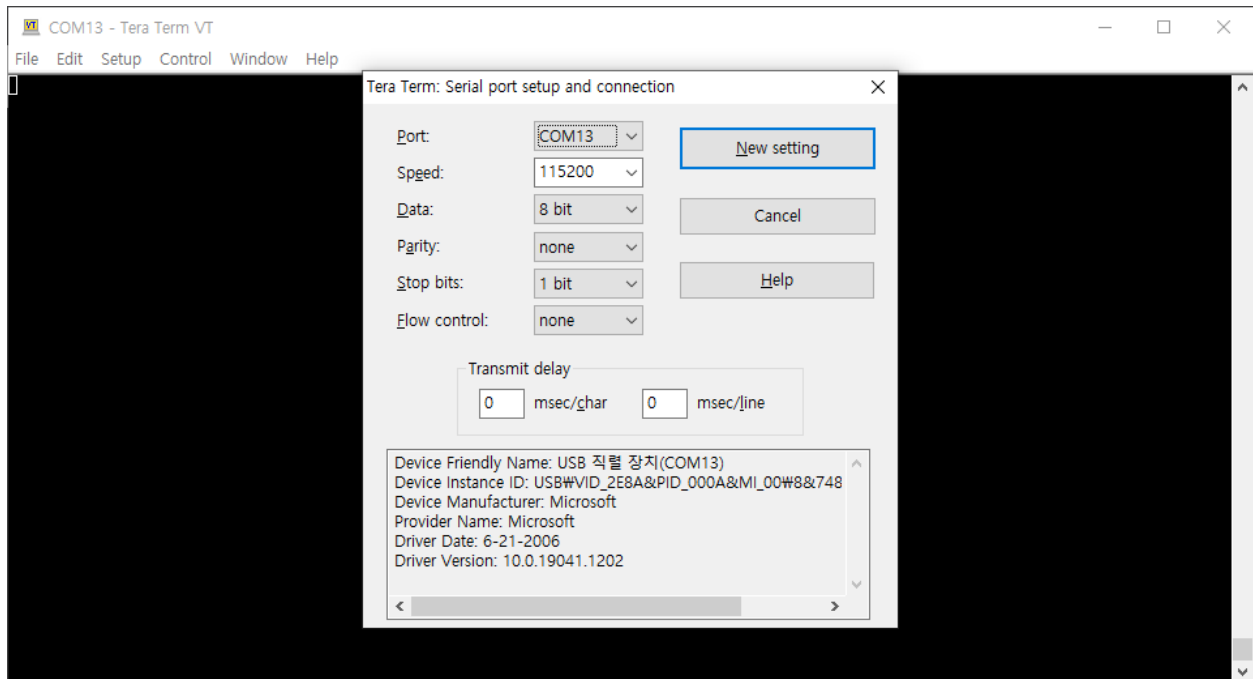


Figure 22. Set up serial port

⑤ Reset your board.

⑥ If the Connect AWS IoT through MQTT example works normally on W5500-EVB-Pico, you can see the network information of W5500-EVB-Pico, connecting to the AWS IoT and publishing the message.

```

COM13 - Tera Term VT
File Edit Setup Control Window Help
DHCP client running
=====
W5500 network configuration : DHCP

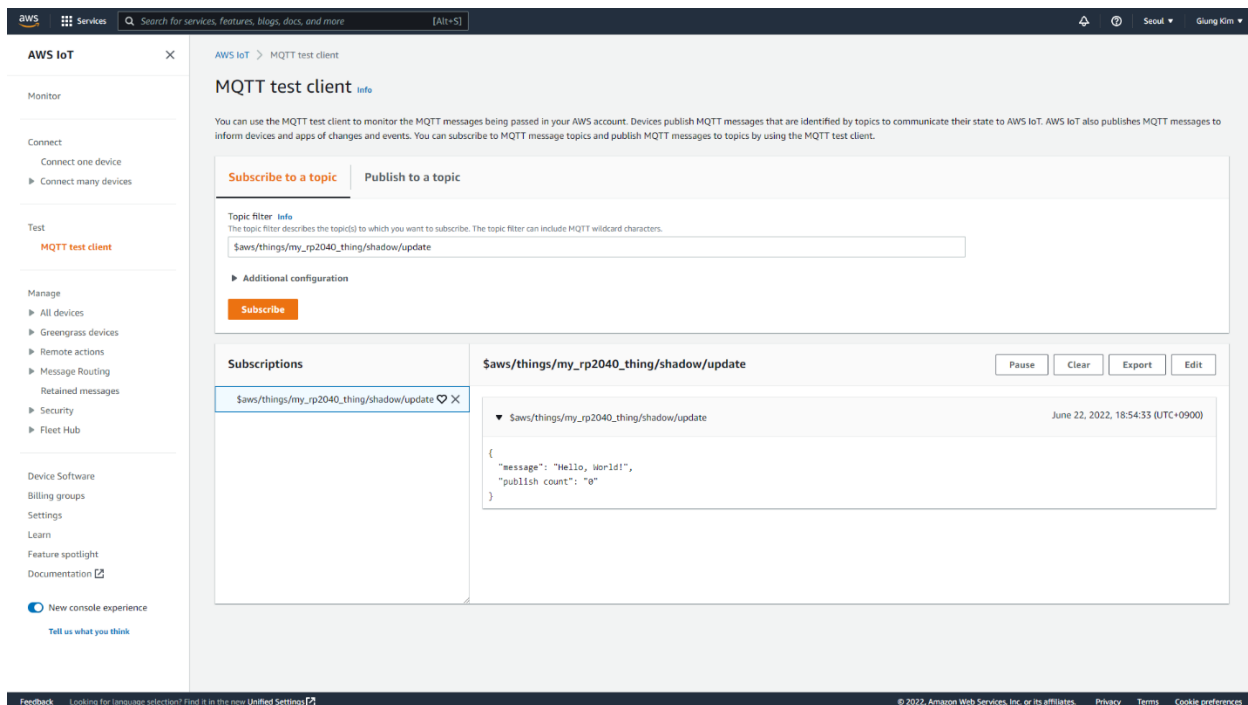
MAC       : 00:08:DC:12:34:56
IP        : 192.168.11.209
Subnet Mask : 255.255.255.0
Gateway   : 192.168.11.1
DNS       : 8.8.8.8
=====

DHCP leased time : 3600 seconds
dns while
dns while
dns while
dns while
dns while
dns while
- DNS: [aqzlwsttrwzrm-ats.iot.ap-northeast-2.amazonaws.com] Get Server IP - 3.34.116.52
ok! mbedtls_x509_crt_parse returned -0x0 while parsing root cert
ok! mbedtls_ssl_set_hostname returned 0
ok! mbedtls_x509_crt_parse returned -0x0 while parsing device cert
ok! mbedtls_pk_parse_key returned -0x0 while parsing private key
Root CA verify option 2
ok! mbedtls_ssl_conf_own_cert returned 0
SSL initialization is success
Performing the SSL/TLS handshake...
ok

[ Ciphersuite is TLS-RSA-WITH-AES-128-GCM-SHA256 ]

SSL connection is success
MQTT initialization is success
MQTT connection is success
MQTT subscription is success
Received SUBACK: PacketID=1
MQTT publishing is success
PUBLISH OK
  
```

Figure 23. Publish the message from W5500-EVB-Pico



The screenshot shows the AWS IoT console's MQTT test client interface. On the left is a navigation sidebar with options like Monitor, Connect, Test (selected), and Manage. The 'Test' section includes 'MQTT test client'. The main area is titled 'MQTT test client' and contains a 'Subscribe to a topic' section. The topic filter is set to '\$aws/things/my_rp2040_thing/shadow/update'. Below this, a 'Subscriptions' table lists the active subscription. To the right, a message is displayed, received on June 22, 2022, at 18:54:33 (UTC+0900). The message content is a JSON object: {'message': 'Hello, world!', 'publish count': '0'}.

Figure 24. Receive the message about subscribed topic from IoT Core

- ⑦ If you publish the message through the test function in AWS IoT Core to the subscribe topic, you can see that the W5500-EVB-Pico receive the message about the subscribe topic.

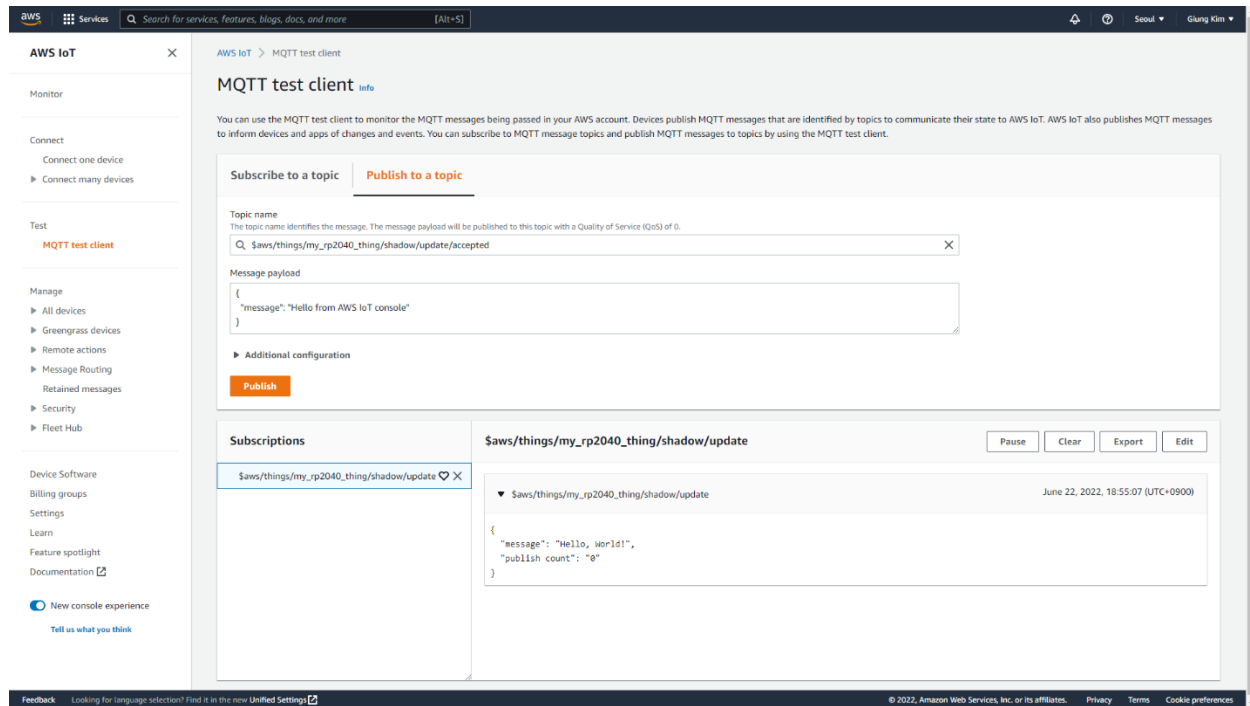
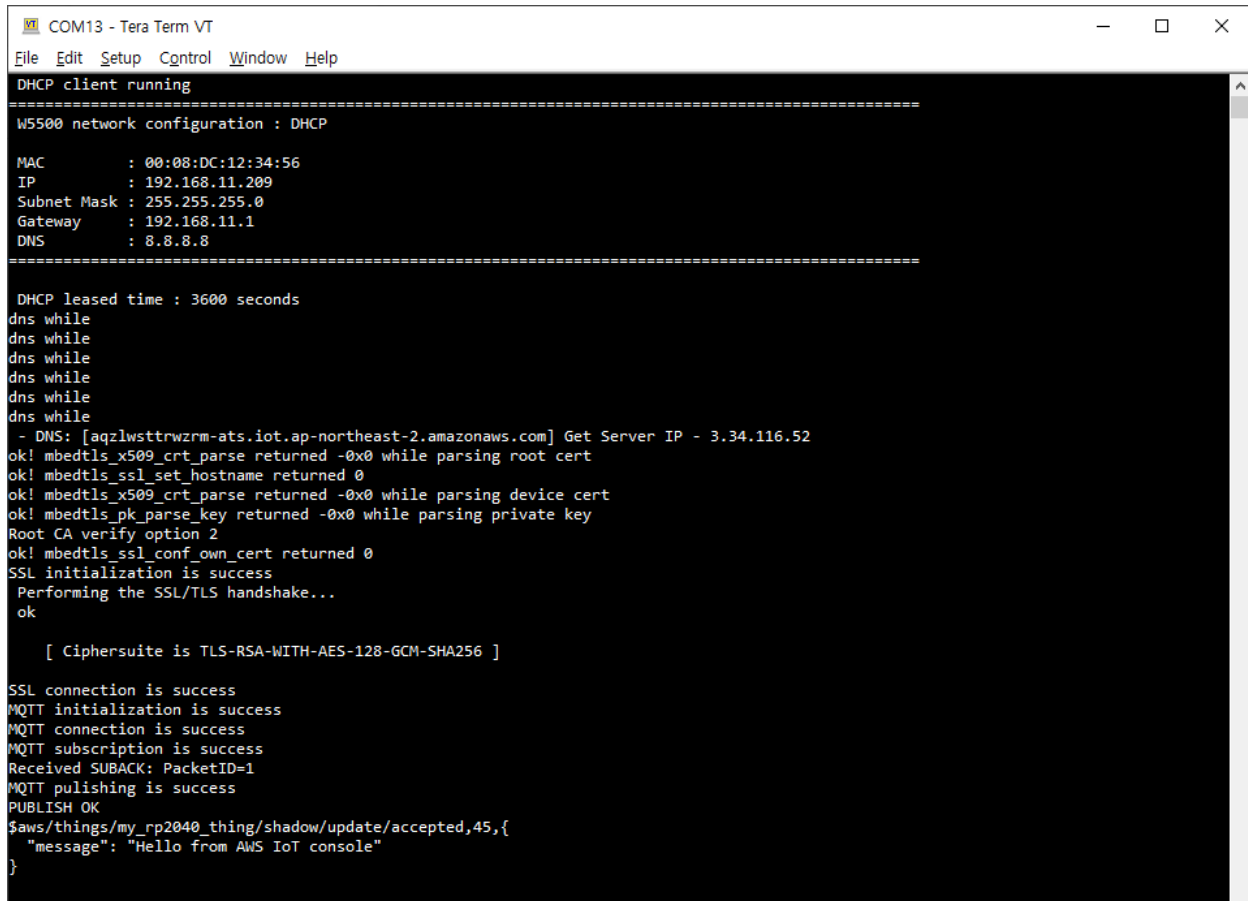


Figure 25. Publish the message from IoT Core



```

COM13 - Tera Term VT
File Edit Setup Control Window Help
DHCP client running
=====
W5500 network configuration : DHCP

MAC       : 00:08:DC:12:34:56
IP        : 192.168.11.209
Subnet Mask : 255.255.255.0
Gateway   : 192.168.11.1
DNS       : 8.8.8.8
=====

DHCP leased time : 3600 seconds
dns while
dns while
dns while
dns while
dns while
dns while
- DNS: [aqzlwsttrwzrm-ats.iot.ap-northeast-2.amazonaws.com] Get Server IP - 3.34.116.52
ok! mbedtls_x509_crt_parse returned -0x0 while parsing root cert
ok! mbedtls_ssl_set_hostname returned 0
ok! mbedtls_x509_crt_parse returned -0x0 while parsing device cert
ok! mbedtls_pk_parse_key returned -0x0 while parsing private key
Root CA verify option 2
ok! mbedtls_ssl_conf_own_cert returned 0
SSL initialization is success
Performing the SSL/TLS handshake...
ok

[ Ciphersuite is TLS-RSA-WITH-AES-128-GCM-SHA256 ]

SSL connection is success
MQTT initialization is success
MQTT connection is success
MQTT subscription is success
Received SUBACK: PacketID=1
MQTT publishing is success
PUBLISH OK
$aws/things/my_rp2040_thing/shadow/update/accepted,45,{
  "message": "Hello from AWS IoT console"
}

```

Figure 26. Receive the message about subscribed topic from W5500-EVB-Pico

11 Debugging

- ① Connect to the serial COM port of W5500-EVB-Pico with Tera Term to view logs and debugging.

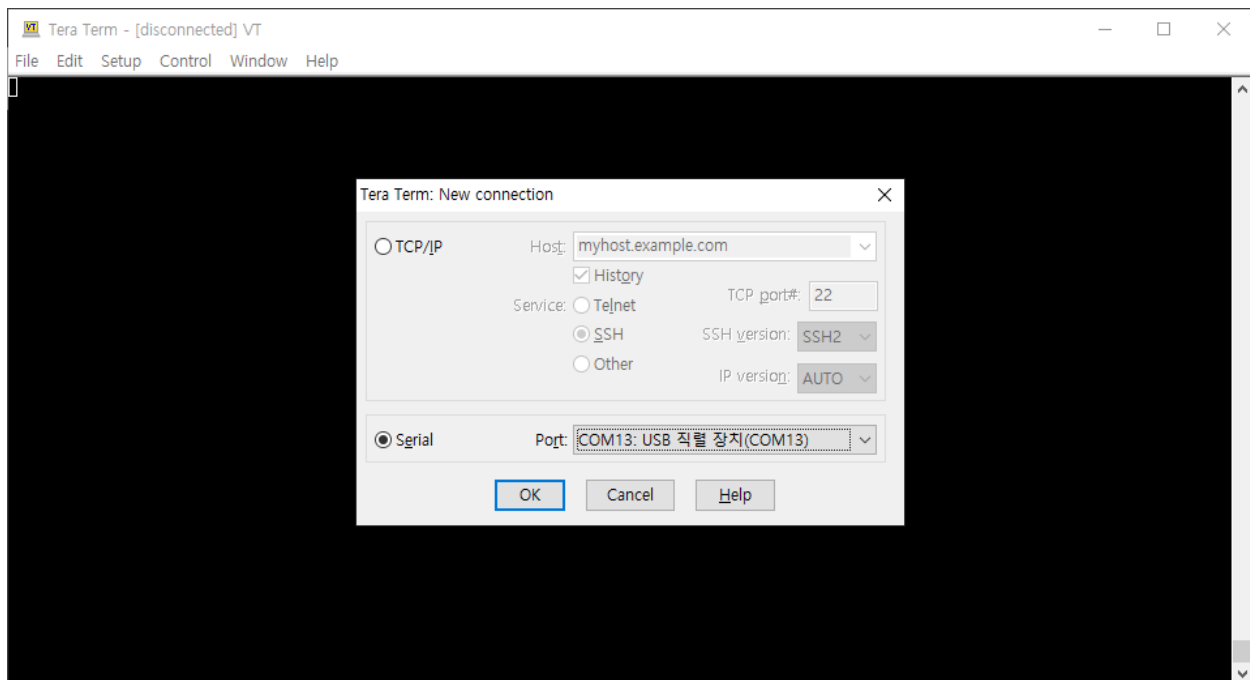


Figure 27. Connect to the serial COM port of W5500-EVB-Pico

② When connecting to the serial COM port of W5500-EVB-Pico, 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

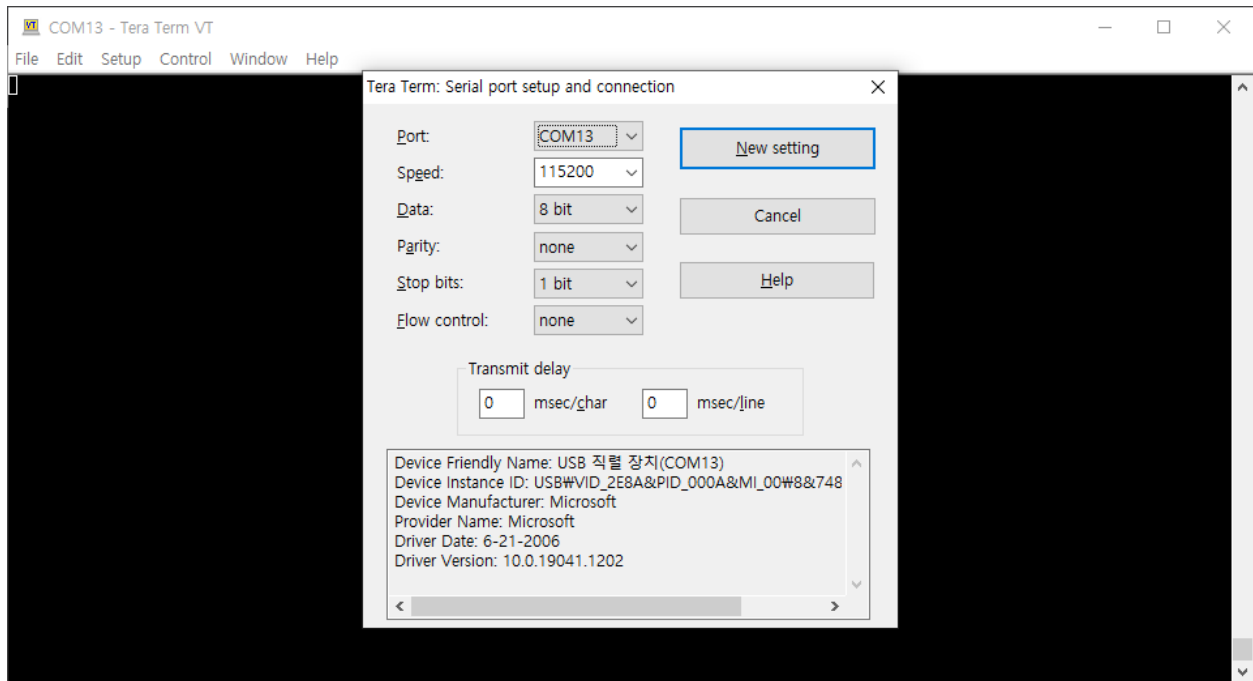


Figure 28. Set up serial port

- ③ If everything works normally, you can see the W5500-EVB-Pico's network information, publishing the message at intervals of 10 seconds and receiving the message about the subscribe topic through logs.

```

COM13 - Tera Term VT
File Edit Setup Control Window Help
DHCP client running
=====
W5500 network configuration : DHCP

MAC       : 00:08:DC:12:34:56
IP        : 192.168.11.209
Subnet Mask : 255.255.255.0
Gateway   : 192.168.11.1
DNS       : 8.8.8.8
=====

DHCP leased time : 3600 seconds
dns while
dns while
dns while
dns while
dns while
dns while
- DNS: [aqzlwsttrwzrm-ats.iot.ap-northeast-2.amazonaws.com] Get Server IP - 3.34.116.52
ok! mbedtls_x509_crt_parse returned -0x0 while parsing root cert
ok! mbedtls_ssl_set_hostname returned 0
ok! mbedtls_x509_crt_parse returned -0x0 while parsing device cert
ok! mbedtls_pk_parse_key returned -0x0 while parsing private key
Root CA verify option 2
ok! mbedtls_ssl_conf_own_cert returned 0
SSL initialization is success
Performing the SSL/TLS handshake...
ok

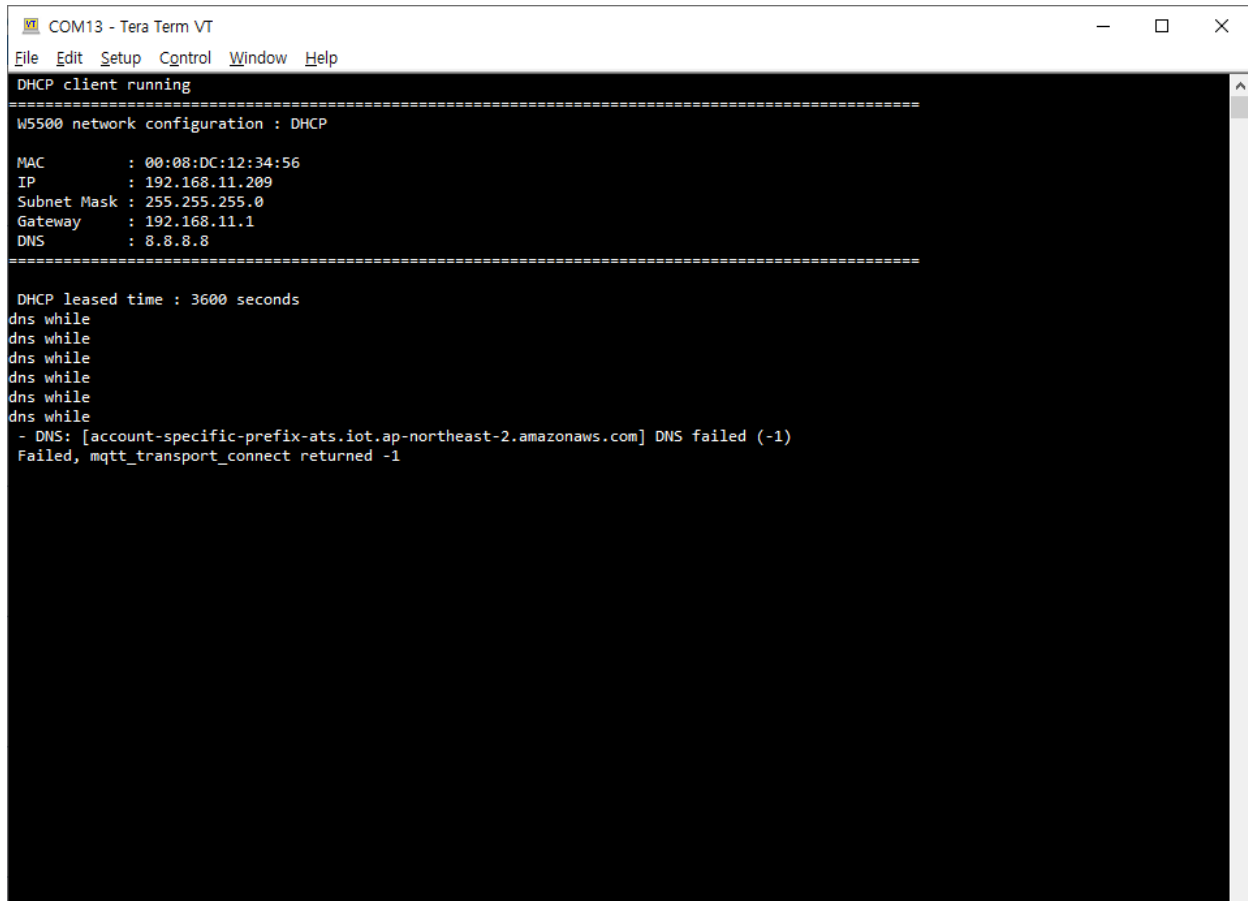
[ Ciphersuite is TLS-RSA-WITH-AES-128-GCM-SHA256 ]

SSL connection is success
MQTT initialization is success
MQTT connection is success
MQTT subscription is success
Received SUBACK: PacketID=1
MQTT publishing is success
PUBLISH OK
$aws/things/my_rp2040_thing/shadow/update/accepted,45,{
  "message": "Hello from AWS IoT console"
}

```

Figure 29. works normally

- ④ If there occurs any problem, the error log is printed showing function what the problem has occurred.



```
COM13 - Tera Term VT
File Edit Setup Control Window Help
DHCP client running
=====
W5500 network configuration : DHCP

MAC       : 00:08:DC:12:34:56
IP        : 192.168.11.209
Subnet Mask : 255.255.255.0
Gateway   : 192.168.11.1
DNS       : 8.8.8.8
=====

DHCP leased time : 3600 seconds
dns while
dns while
dns while
dns while
dns while
dns while
- DNS: [account-specific-prefix-ats.iot.ap-northeast-2.amazonaws.com] DNS failed (-1)
Failed, mqtt_transport_connect returned -1
```

Figure 30. Failed in DNS

12 Troubleshooting

If you have any questions or problems while using the W5500-EVB-Pico examples, please leave them at the link below.

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