Microsoft

Azure RTOS Samples for STM32L475-DISCO IoT Node using STM32CubeIDE User Guide

Published: May 2020

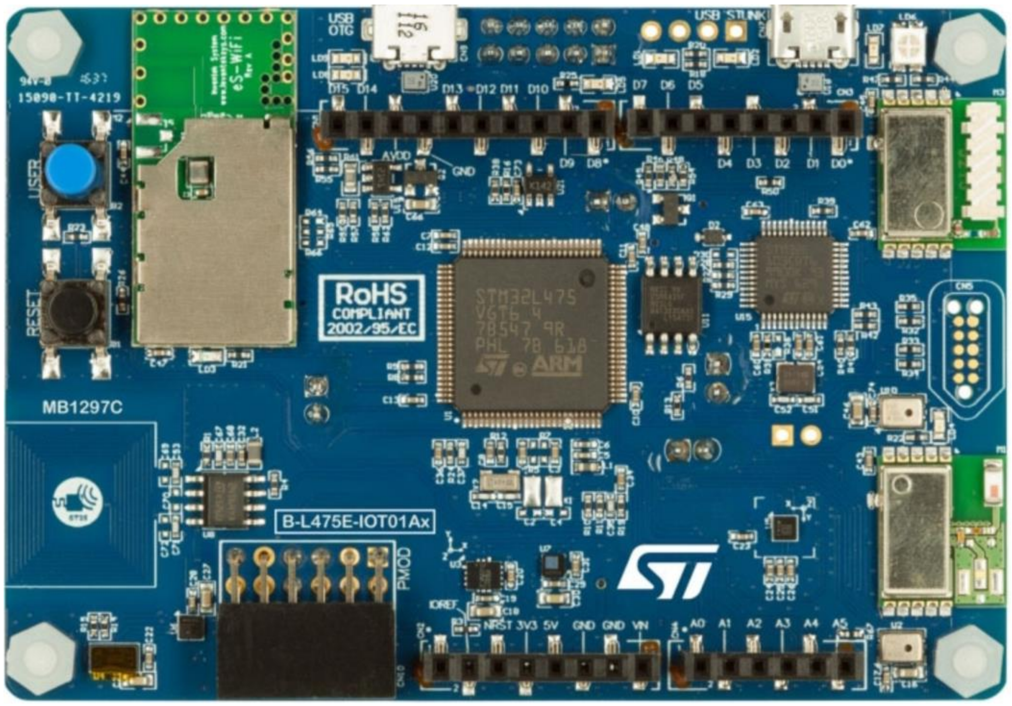
For the latest information, please see
azure.com/rtos

Part Number: 000-1059

Revision 6.0

Table of Contents

Overview



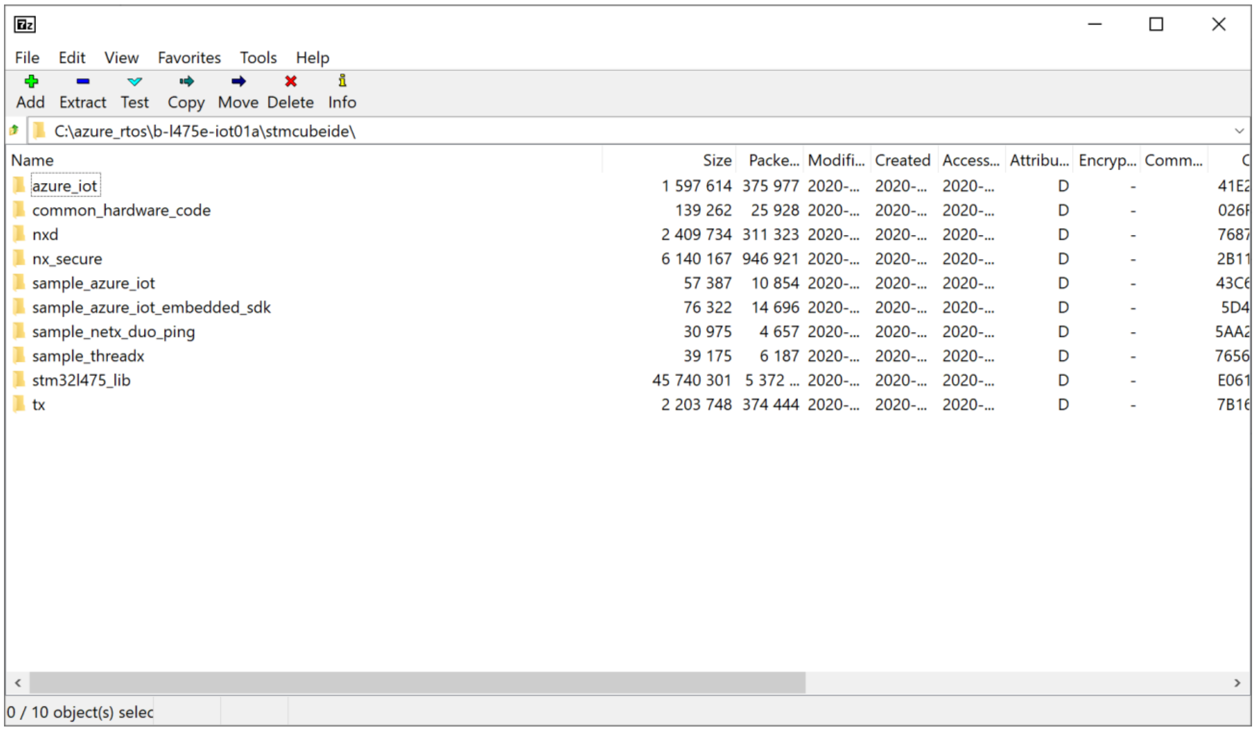
STM32L475-DISCO IoT Node board

Azure RTOS Samples for each component (Azure connectivity, ThreadX and NetX Duo) are designed to run on the STM32L465-DISCO IoT Node "out-of-the-box." Each sample project is described later in this document along with links to further information as necessary.

All samples are designed to run using the STM32CubeIDE 1.3.0 or later, with the on-board ST-LINK debugger (Debug USB port). The default factory jumper selections are assumed. It can be downloaded free from this page:

https://www.st.com/en/development-tools/stm32cubeide.html

The sample distribution zip file has following organization:


File Edit View Favorites Tools Help
Add Extract Test Copy Move Delete Info

C:\azure_rtos\b-l475e-iot01a\stmcubeide\

Name	Size	Packe...	Modifi...	Created	Access...	Attribu...	Encryp...	Comm...	C
azure_iot	1 597 614	375 977	2020-...	2020-...	2020-...	D	-		41E2
common_hardware_code	139 262	25 928	2020-...	2020-...	2020-...	D	-		026F
nxd	2 409 734	311 323	2020-...	2020-...	2020-...	D	-		7687
nx_secure	6 140 167	946 921	2020-...	2020-...	2020-...	D	-		2B11
sample_azure_iot	57 387	10 854	2020-...	2020-...	2020-...	D	-		43C6
sample_azure_iot_embedded_sdk	76 322	14 696	2020-...	2020-...	2020-...	D	-		5D4
sample_netx_duo_ping	30 975	4 657	2020-...	2020-...	2020-...	D	-		5AA2
sample_threadx	39 175	6 187	2020-...	2020-...	2020-...	D	-		7656
stm32l475_lib	45 740 301	5 372 ...	2020-...	2020-...	2020-...	D	-		E061
tx	2 203 748	374 444	2020-...	2020-...	2020-...	D	-		7B16

0 / 10 object(s) selec

The root directory contains the workspace as well as following sub-folders:

Folder	Contents
stm32cubeide	Contains the following sub-folders
common_hardware_code	Contains common code for STM32FL475-DISCO board
docs	Contains user guides and supporting documentation
nxd	Contains NetX Duo source code and pre-built NetX Duo library (nxd.a)
nx_secure	Contains NetX Secure source code and pre-built NetX Secure library (nx_secure.a)
sample_azure_iot	Contains sample project to connect Azure RTOS to Azure IoT Hub
sample_netx_duo_ping	Contains NetX Duo ping sample project
sample_threadx	Contains ThreadX sample project
stm32l475_lib	Contains STM32L475 drivers
tx	Contains ThreadX source code and pre-built ThreadX library (tx.a)

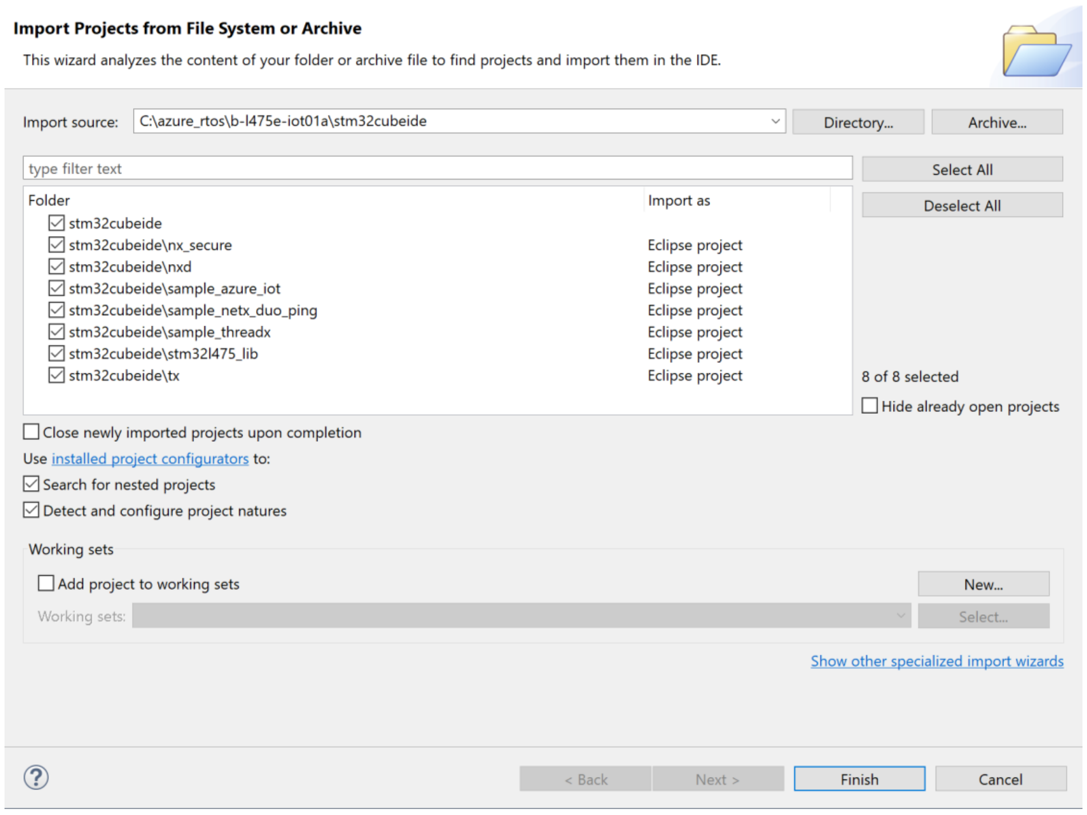
Getting Started

1) Unpack the sample zip file into a folder of your choice, we recommend:

   C:\azure_rtos\b-l475e-iot01a\stm32cubeide

2) Open STM32CubeIDE, select ***File > Open Projects from File System…***, select the ***stm32cubeide*** folder and select ***Finish*** to open the workspace. If you don't have STM32CubeIDE 1.3.0 or above, it can be downloaded from this page:

   https://www.st.com/en/development-tools/stm32cubeide.html



Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\azure_rtos\b-l475e-iot01a\stm32cubeide

Folder	Import as
☑ stm32cubeide	
☑ stm32cubeide\nx_secure	Eclipse project
☑ stm32cubeide\nxd	Eclipse project
☑ stm32cubeide\sample_azure_iot	Eclipse project
☑ stm32cubeide\sample_netx_duo_ping	Eclipse project
☑ stm32cubeide\sample_threadx	Eclipse project
☑ stm32cubeide\stm32l475_lib	Eclipse project
☑ stm32cubeide\tx	Eclipse project

8 of 8 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

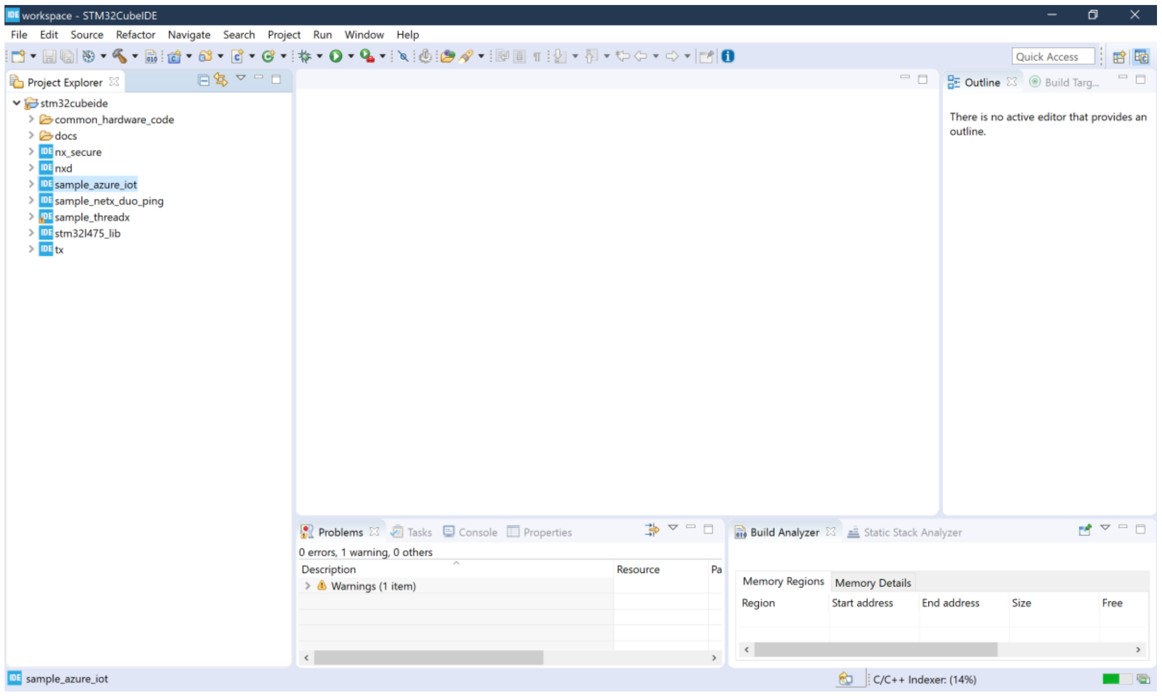
☑ Search for nested projects

☑ Detect and configure project natures

Working sets

☐ Add project to working sets

3) Select the desired sample project in the Project Explorer.



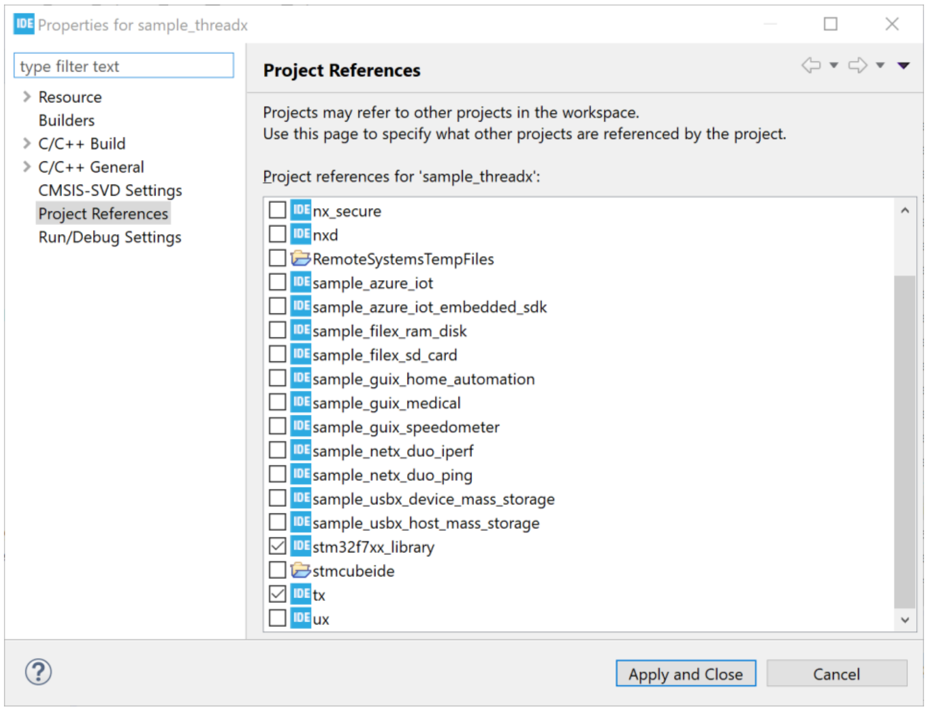
There is no active editor that provides an outline.

0 errors, 1 warning, 0 others

The ***sample_azure_iot*** project is shown above as the currently active project.

4) Select ***Build*** button or ***Project > Build Project*** to build the selected project. You will observe compilation and linking of the selected sample project.

5) Make sure the correct set of Project references are selected. Right click on the project, select ***Properties > Project References*** and select the projects that are required for the sample project.



Properties for sample_threadx

type filter text

- Resource
- Builders
- C/C++ Build
- C/C++ General
- CMSIS-SVD Settings
- Project References
- Run/Debug Settings

Project References

Projects may refer to other projects in the workspace.
Use this page to specify what other projects are referenced by the project.

Project references for 'sample_threadx':

- [ ] nx_secure
- [ ] nxd
- [ ] RemoteSystemsTempFiles
- [ ] sample_azure_iot
- [ ] sample_azure_iot_embedded_sdk
- [ ] sample_filex_ram_disk
- [ ] sample_filex_sd_card
- [ ] sample_guix_home_automation
- [ ] sample_guix_medical
- [ ] sample_guix_speedometer
- [ ] sample_netx_duo_iperf
- [ ] sample_netx_duo_ping
- [ ] sample_usbx_device_mass_storage
- [ ] sample_usbx_host_mass_storage
- [✓] stm32f7xx_library
- [ ] stmcubeide
- [✓] tx
- [ ] ux

6) Select ***Debug*** to download and start execution of the project. By default, execution stops at a breakpoint set at ***main***.

7) Select ***Resume*** to start execution of the demonstration. Please review the sample descriptions later in this guide for additional setup and expected behavior.

Sample Descriptions

Azure IoT HUB Connectivity Sample

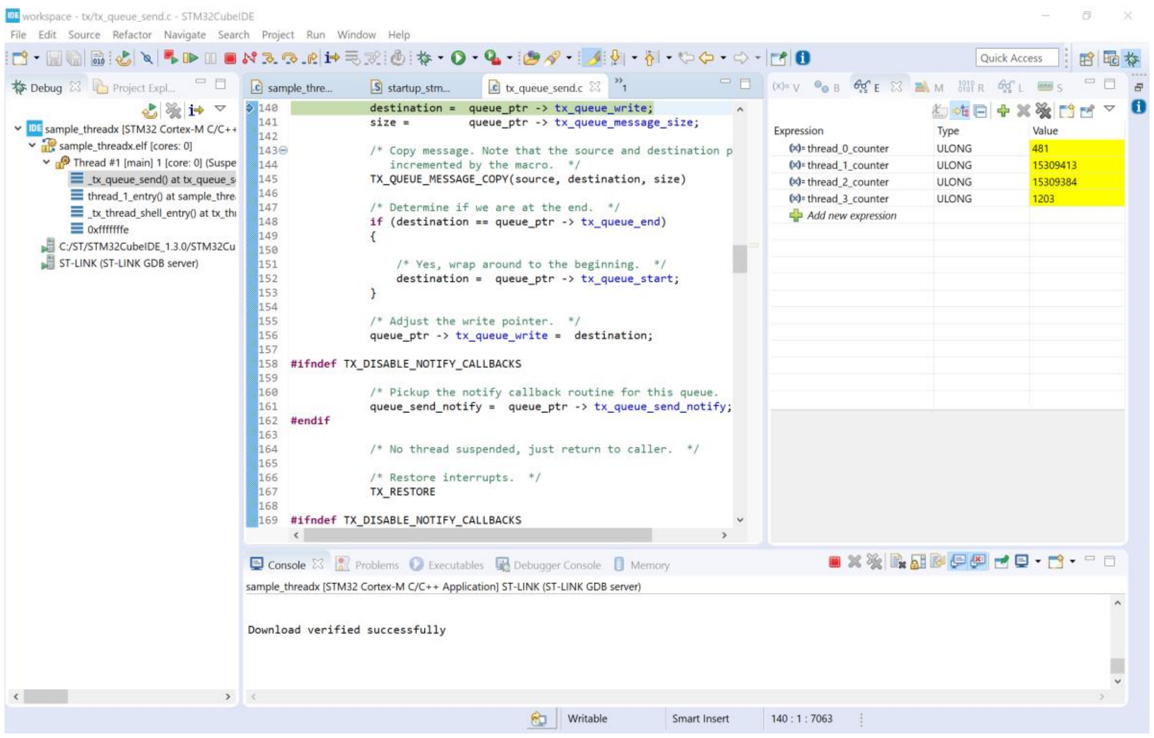
This sample show how easy it is to connect to Azure IoT using Azure RTOS. Please see the ***Azure_RTOS_STM32L475-DISCO_Azure_IoT_Quick_Connect_For_STM32CubeIDE.pdf*** for a detailed description of the sample as well as a step-by-step set of instructions on how to connect to Azure IoT.

ThreadX Sample

This sample is the standard 8-thread ThreadX example, that illustrates the use of the main ThreadX services, including threads, message queues, timers, semaphores, byte memory pools, block memory pools, event flag groups, and mutexes. This demonstration is fully described, including a source code listing, in Chapter 6 of the ***Azure_RTOS_ThreadX_User_Guide.pdf*** (also provided in this distribution).

To run the ThreadX Sample project, simply follow these steps (assuming the workspace is already open):

1. Click on the ***sample_threadx*** project to make the project active.
2. Select ***Build*** button to build the project selected. You will observe compilation and linking of the selected sample project.
3. Select ***Debug*** to download and start execution of the demonstration. The sample will initially stop at ***main***. Select another ***Resume*** to execute the sample.



sample_threadx [STM32 Cortex-M C/C++ Application]
sample_threadx.elf [cores: 0]
Thread #1 [main] 1 [core: 0] (Suspe
_tx_queue_send() at tx_queue_s
thread_1_entry() at sample_thre
_tx_thread_shell_entry() at tx_thi
0xfffffffe
C:/ST/STM32CubeIDE_1.3.0/STM32Cu
ST-LINK (ST-LINK GDB server)

```
            destination =  queue_ptr -> tx_queue_write;
            size =         queue_ptr -> tx_queue_message_size;

            /* Copy message. Note that the source and destination p
               incremented by the macro.  */
            TX_QUEUE_MESSAGE_COPY(source, destination, size)

            /* Determine if we are at the end.  */
            if (destination == queue_ptr -> tx_queue_end)
            {

                /* Yes, wrap around to the beginning.  */
                destination =  queue_ptr -> tx_queue_start;
            }

            /* Adjust the write pointer.  */
            queue_ptr -> tx_queue_write =  destination;

#ifndef TX_DISABLE_NOTIFY_CALLBACKS

            /* Pickup the notify callback routine for this queue.
            queue_send_notify =  queue_ptr -> tx_queue_send_notify;
#endif

            /* No thread suspended, just return to caller.  */

            /* Restore interrupts.  */
            TX_RESTORE

#ifndef TX_DISABLE_NOTIFY_CALLBACKS
```

Expression	Type	Value
thread_0_counter	ULONG	481
thread_1_counter	ULONG	15309413
thread_2_counter	ULONG	15309384
thread_3_counter	ULONG	1203
Add new expression		

sample_threadx [STM32 Cortex-M C/C++ Application] ST-LINK (ST-LINK GDB server)

```
Download verified successfully
```

After hitting ***Suspend*** ⏸ the STM32CubeIDE debugger screen shot above shows various counters incremented by the ThreadX sample as each of the main components of the ThreadX are exercised. You want view the counters from ***Window > Show View > Expressions.***

To learn more about Azure RTOS ThreadX, view ***Azure_RTOS_ThreadX_User_Guide.pdf*** and https://azure.com/rtos.

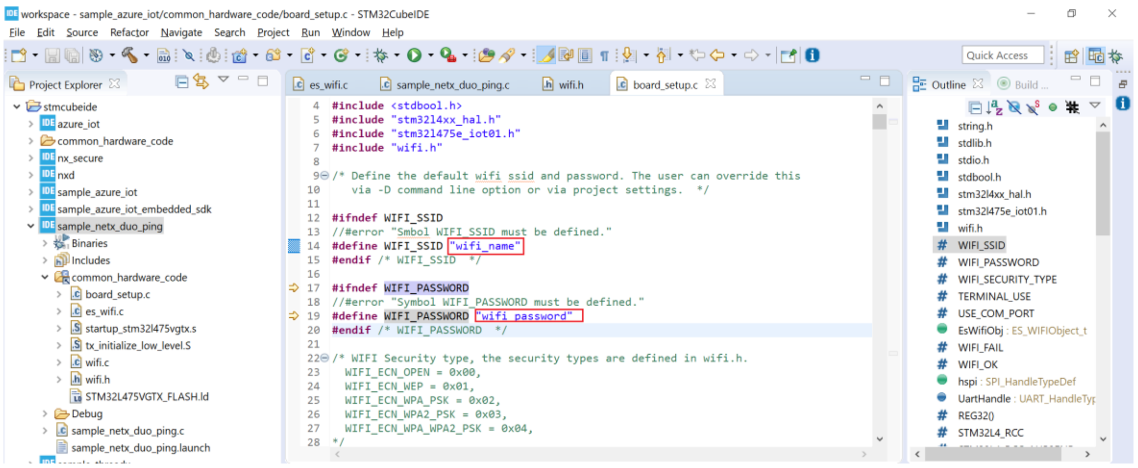
NetX Duo Simple Ping Sample

This sample project illustrates the setup and use of NetX Duo IPv4/IPv6 TCP/IP stack via ping from another node on the local network. By default, this demonstration requests an IP Address via DHCP, and displays the status and assigned IP Address via Terminal I/O in the IAR debugger. In addition, the retrieved IP address is displayed in the watch window of the IAR debugger.

To run the NetX Duo Ping Sample project, simply follow these steps (assuming the workspace is already open):

1. Click on the ***sample_netx_duo_ping*** project and make the project active.

2. Find "***board_setup.c***" within the "***common_hardware_code***" folder. Update your WiFi settings:

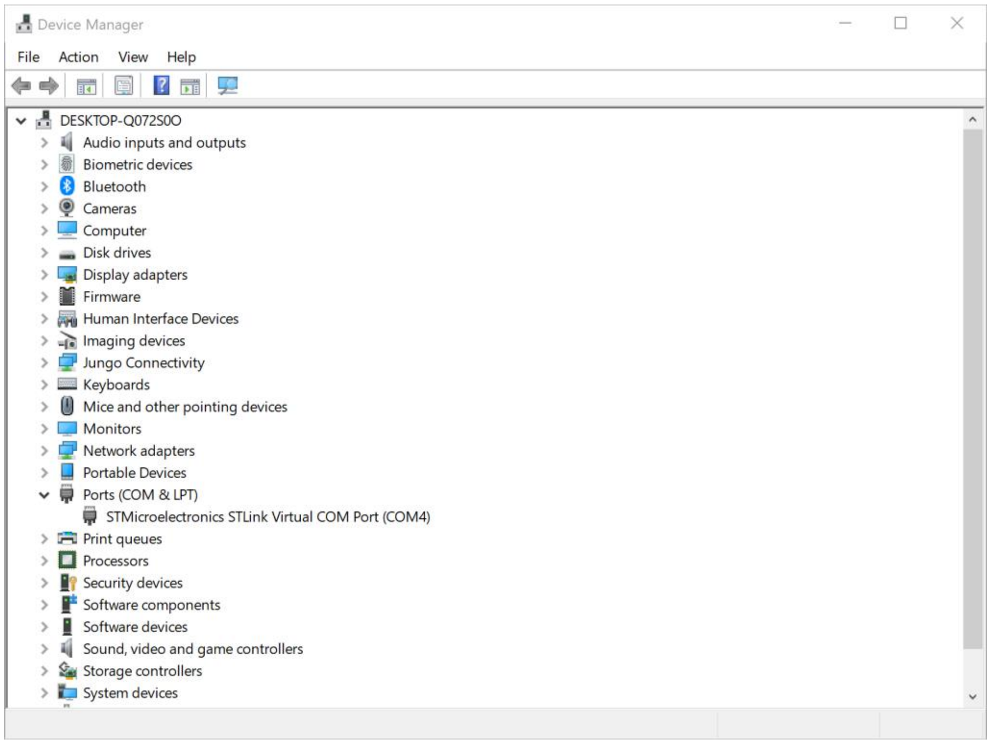


```
#include <stdbool.h>
#include "stm32l4xx_hal.h"
#include "stm32l475e_iot01.h"
#include "wifi.h"

/* Define the default wifi ssid and password. The user can override this
   via -D command line option or via project settings.  */

#ifndef WIFI_SSID
//#error "Smbol WIFI_SSID must be defined."
#define WIFI_SSID "wifi_name"
#endif /* WIFI_SSID  */

#ifndef WIFI_PASSWORD
//#error "Symbol WIFI_PASSWORD must be defined."
#define WIFI_PASSWORD "wifi_password"
#endif /* WIFI_PASSWORD  */

/* WIFI Security type, the security types are defined in wifi.h.
  WIFI_ECN_OPEN = 0x00,
  WIFI_ECN_WEP = 0x01,
  WIFI_ECN_WPA_PSK = 0x02,
  WIFI_ECN_WPA2_PSK = 0x03,
  WIFI_ECN_WPA_WPA2_PSK = 0x04,
*/
```

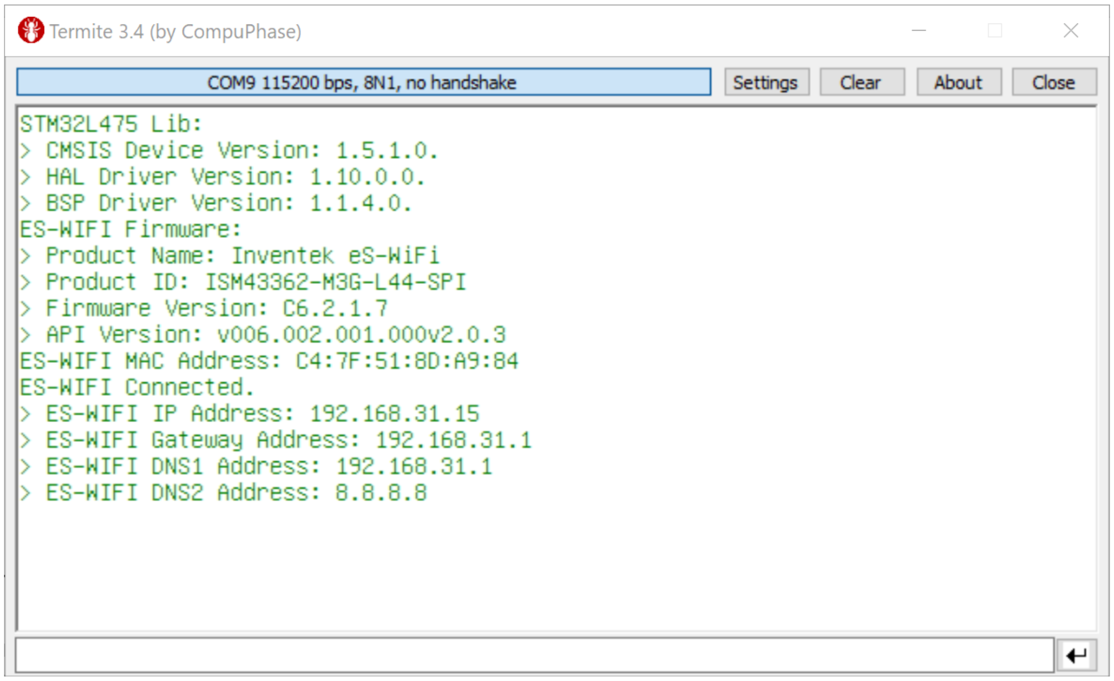
3. Select ***Build*** button to build the project selected. You will observe compilation and linking of the selected sample project.

4. Select ***Debug*** to download and start execution of the demonstration. The sample will initially stop at ***main***. Select another ***Resume*** to execute the sample.

5. Verify the serial port in your OS's device manager. It should show up as a COM port.

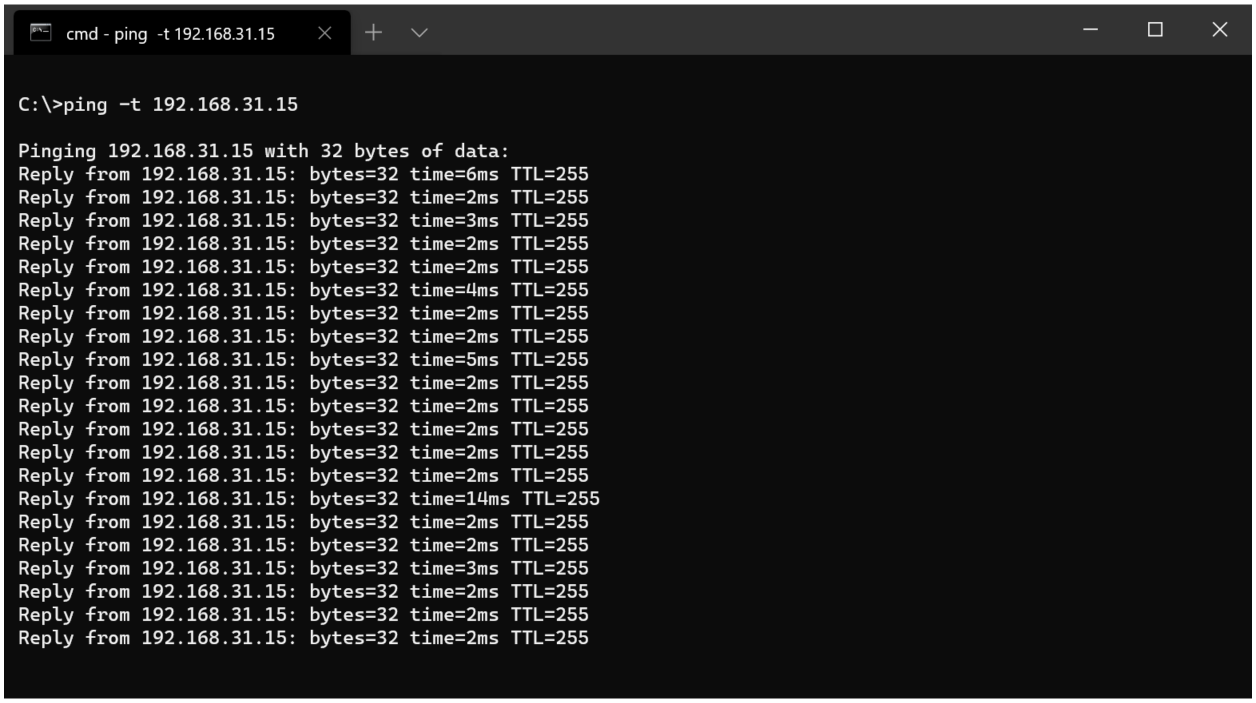


6. Open your favorite serial terminal program such as Termite and connect to the COM port discovered above, should observe the IP address assigned via DHCP in the Terminal output window.


Termite 3.4 (by CompuPhase)
COM9 115200 bps, 8N1, no handshake
Settings
Clear
About
Close

```
STM32L475 Lib:
> CMSIS Device Version: 1.5.1.0.
> HAL Driver Version: 1.10.0.0.
> BSP Driver Version: 1.1.4.0.
ES-WIFI Firmware:
> Product Name: Inventek eS-WiFi
> Product ID: ISM43362-M3G-L44-SPI
> Firmware Version: C6.2.1.7
> API Version: v006.002.001.000v2.0.3
ES-WIFI MAC Address: C4:7F:51:8D:A9:84
ES-WIFI Connected.
> ES-WIFI IP Address: 192.168.31.15
> ES-WIFI Gateway Address: 192.168.31.1
> ES-WIFI DNS1 Address: 192.168.31.1
> ES-WIFI DNS2 Address: 8.8.8.8
```

The example above shows that the assigned IP address of the STM32L475-DISCO IoT Node board is 192.168.31.15. When the demonstration is running it can be pinged by any machine on the network. The following is an example of a ping from a Windows machine on the same local netword (using the DOS command window):



```
C:\>ping -t 192.168.31.15

Pinging 192.168.31.15 with 32 bytes of data:
Reply from 192.168.31.15: bytes=32 time=6ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=3ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=4ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=5ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=14ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=3ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
Reply from 192.168.31.15: bytes=32 time=2ms TTL=255
```

To learn more about Azure RTOS NetX Duo, view ***Azure_RTOS_NetX_Duo_User_Guide.pdf*** and https://azure.com/rtos.