Azure RTOS and ST Microelectronics STM32 Discovery Kit IoT (STM32L4S5)

By Sean D. Liming and John R. Malin
Annabooks – www.annabooks.com

May 2023

There are a number of Azure RTOS online guides to get started with different platforms. The STM32L4S5 Discovery Kit is one of the first platforms that demonstrated connecting to Azure IoT Central. If you follow the quick start online documents, you will be able to build the example application from the command line and get it to run. If you want to use the example applications as a basis for a project, being able to debug by stepping through the code is going to be important. In this paper, we will walk through the example but set up the development environment to use Visual Studio Code.

Target Hardware: STM32L4S5 Discovery Kit (BL-4S5I-IOT01A)

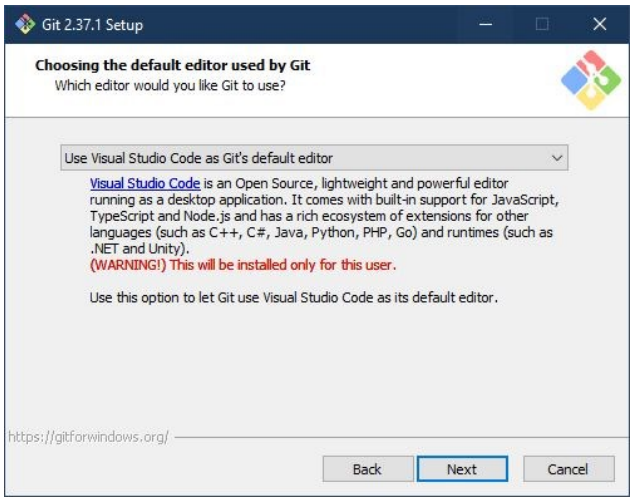
1 Tools Setup

For this setup will we need to download and install a few items.

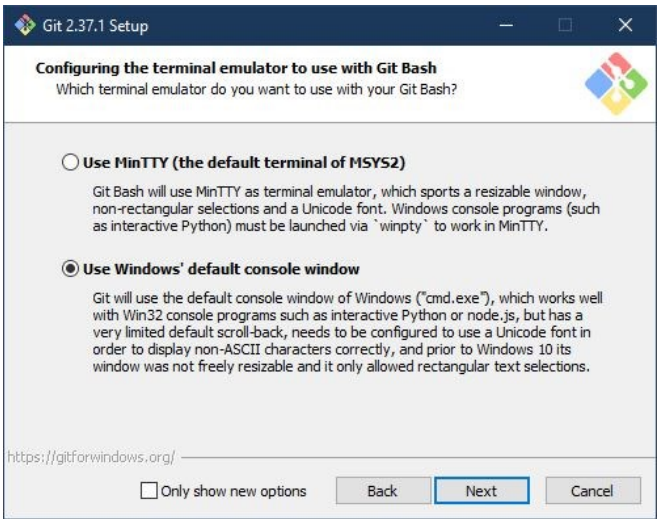
1. Download and install Visual Studio Code: Visual Studio Code - Code Editing 1.69.2.
2. Once Visual Studio Code has been installed, install the following add-ons from the Visual Studio Code marketplace:

   - C/C++ - Visual Studio Marketplace
   - CMake Tools - Visual Studio Marketplace
   - CMake - Visual Studio Marketplace
   - Cortex-Debug - Visual Studio Marketplace
   - Embedded Tools - Visual Studio Marketplace
   - Windows-arm-none-eabi – Visual Marketplace

3. Install Git so we can download the Azure RTOS to get started building the files: Git - Downloads (git-scm.com).
   a. Accept the license, and click Next.
   b. Leave the install location as is, and click Next.
   c. Leave the Selected Components as they are, and click Next.
   d. Keep the State Menu Folder as is, and click Next.
   e. Set the default editor selection to be “Use Visual Studio Code as Git’s default editor”, and click Next.



f. Keep the default for initial branches, and click Next.
g. Keep the default PATH Environment, and click Next.
h. Keep the default OpenSSH selection, and click Next.
i. Select “Use Windows’ default console window”, and click Next.



j. Keep the defaults for the next question, and click Next.
k. Select “Use Windows’ default console window”, and click Next.
l. Keep the defaults for the next question, and click Next.
m. Keep the defaults for the next question, and click Next.
n. Keep the defaults for the extra options, and click Next.
o. Keep the defaults for the experimental options, click Install.
p. Click Finish once the install completes.

4. Download and install ABCOMTERM from Annabooks.com. This will be the terminal program to see the standard output from the device.
5. Reboot the computer.

2 Visual Studio Code Sample Application

2.1 Download the Getting Started Files from GitHub

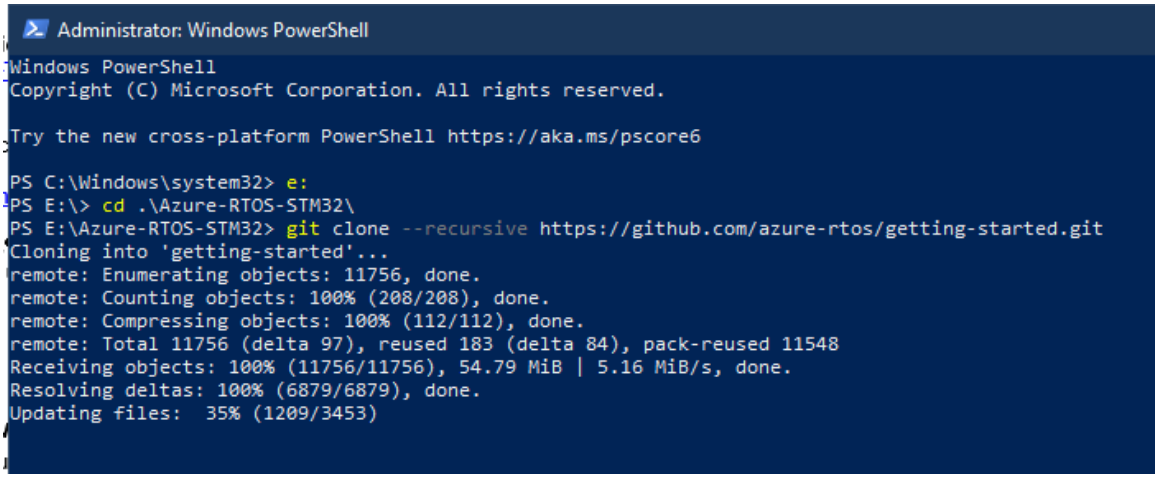
Now, we need to get the getting started repository that contains the Azure RTOS build example and the ports to the B-L4S5I-IOT01A and other development kits.

1. Create a directory called \Azure-RTOS-STM32.
2. Open PowerShell.
3. Change the directory to the newly created folder:

   ```
   cd  \Azure-RTOS-SM32
   ```

4. Run the following

   ```
   git clone --recursive https://github.com/azure-rtos/getting-started.git
   ```



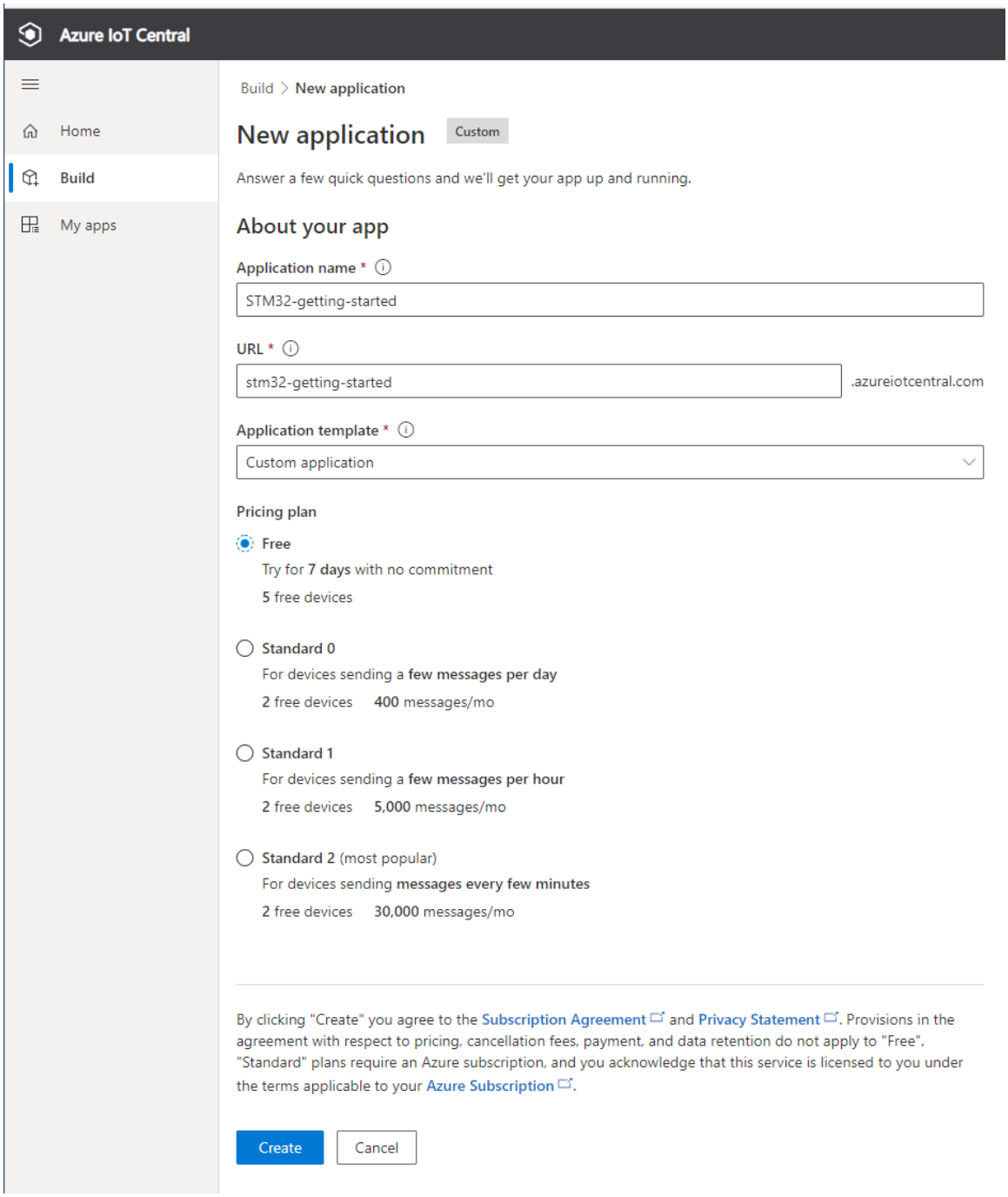
2.2 Create Azure IoT Central Application

Now we need to set up the application on Azure IoT Central.

1. In a browser, open https://apps.azureiotcentral.com/home
2. Sign into the account or create an account.
3. Click on Build App.
4. In the Custom app tile, click Create app

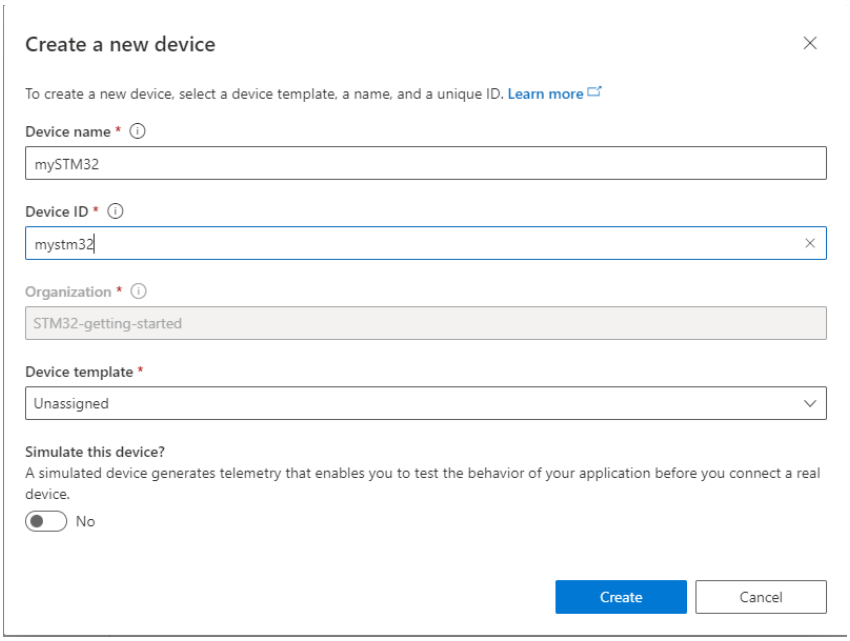
   Application Name: STM32-getting-started.
   Pricing Plan: Free.

5. Click Create.

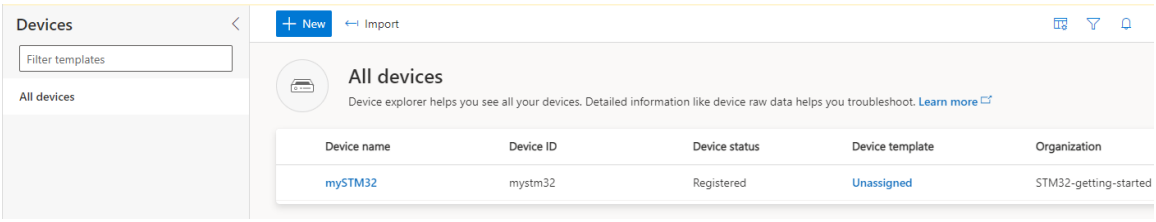


Note: Pricing plans can change.

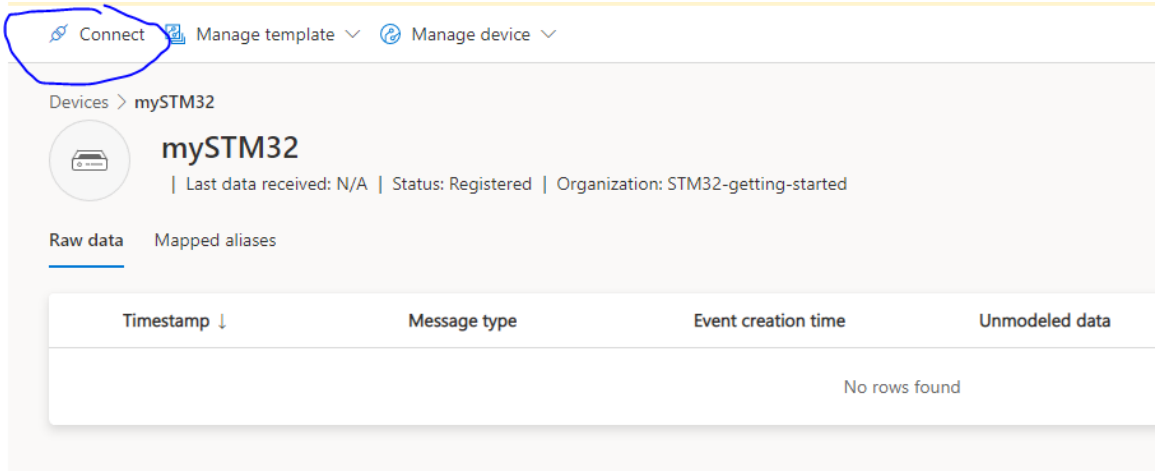
6. Now, we need to add a device to the application and click on the +New button that is above the All Devices section.
7. Enter the following:
    a. Device Name: mySTM32
    b. Device ID mystm32
8. Click Create.



9. Click Create.
10. The device will be created and listed under all devices



11. Click on mySTM32. This will be the view of the data coming in.
12. Click on Connect at the top of the bar.



13. A Device Connections group box appears. Copy the following information and paste it into a Notepad or Notepad++ temporary document. We will need this in the next section.

    - ID scope

- Device ID
- Primary Key

14. Close the dialog when finished.

No need to set up a template as pre-published template for the STM32L4S5 Discovery Kit will be used to display the data.

2.3 Building the Getting Started Sample App

With the application created in Azure IoT Central and the device information collected to make the connection, we are ready to build the example.

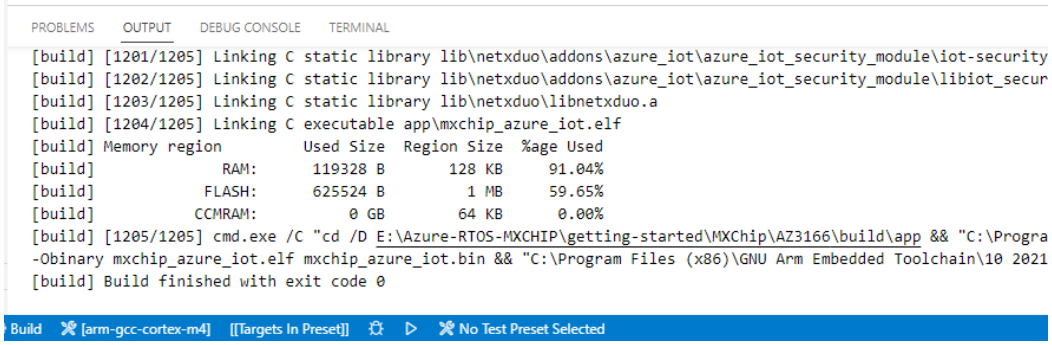
1. Open PowerShell and change the directory to \Azure-RTOS-STM32\getting-started\STMicroelectronics\B-L4S5I-IOT01A.
2. Type the following and hit enter to open Visual Studio Code:

```
code .
```

3. You will be asked to trust the authors of the code. Click Yes.
4. When asked for the toolchain at the top, accept arm-gcc-cortex-m4.
5. Under B-L4S5I-IOT01A\App, open Azure_config.h and fill in the information gathered from the Azure IoT Central application, as well as, your Wi-Fi connection settings:

Constant name	Value
IOT_DPS_ID_SCOPE	ID scope value
IOT_DPS_REGISTRATION_ID	Device ID value
IOT_DEVICE_SAS_KEY	Primary key value
WIFI_SSID	Your Wi-Fi SSID
WIFI_PASSWORD	Your Wi-Fi password
WIFI_MODE	WEP, WPA_PSK_TKIP, or WPA2_PSK_AES

6. Save the file.
7. At the bottom, click on Build. It will take a few minutes, but the build should complete successfully

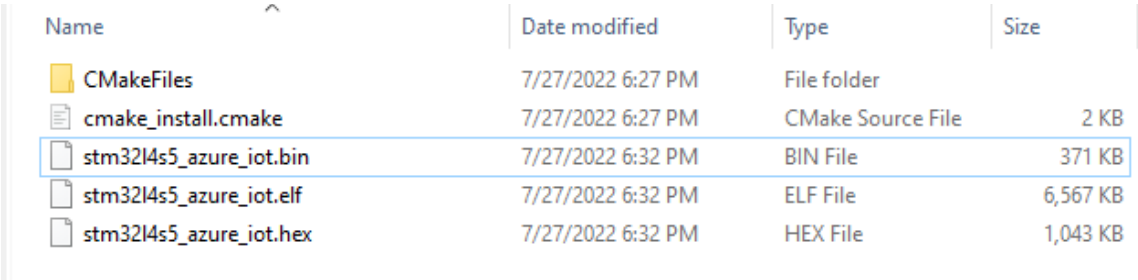


2.4 Program the STM32L4S5 Discovery Kit Board

With the stm32l4s5_azure_iot.bin build, programming the board is a simple copy and paste.

1. Open File Explorer.

2. Navigate to the \Azure-RTOS-STM32\getting-started\STMicroelectronics\B-L4S5I-IOT01A\build\app folder. The newly created stm32l4s5_azure_iot.bin file should be present.



Name	Date modified	Type	Size
CMakeFiles	7/27/2022 6:27 PM	File folder	
cmake_install.cmake	7/27/2022 6:27 PM	CMake Source File	2 KB
stm32l4s5_azure_iot.bin	7/27/2022 6:32 PM	BIN File	371 KB
stm32l4s5_azure_iot.elf	7/27/2022 6:32 PM	ELF File	6,567 KB
stm32l4s5_azure_iot.hex	7/27/2022 6:32 PM	HEX File	1,043 KB

3. Connect the USB cable from the B-L4S5I-IOT01A to your development computer.
4. Copy and paste the mxchip_azure_iot.bin into the <drive letter>DIS_L4S5VI folder. Programming starts automatically. The Red LED will be lit and go off when completed.
5. Open a serial terminal program and connect to AZ3166 COM port and set the baud rate to 115200. ABCOMTERM sets the baud rate to 115200 by default.
6. Hit the reset button on the B-L4S5I-IOT01A

If all goes well, you will see the terminal output with something similar to the following:

```
Starting Azure thread


Initializing WiFi
       Module: ISM43362-M3G-L44-SPI
       MAC address: C4:7F:51:91:44:40
       Firmware revision: C3.5.2.5.STM
SUCCESS: WiFi initialized


Connecting WiFi
       Connecting to SSID 'Net1980i8085'
       Attempt 1...
SUCCESS: WiFi connected

Initializing DHCP
       IP address: 192.168.1.41
       Mask: 255.255.255.0
       Gateway: 192.168.1.1
SUCCESS: DHCP initialized

Initializing DNS client
       DNS address 1: 192.168.1.1

       DNS address 2: 8.8.8.8
SUCCESS: DNS client initialized

Initializing SNTP time sync
       SNTP server 0.pool.ntp.org
       SNTP time update: Jul 28, 2022 2:6:40.16 UTC
SUCCESS: SNTP initialized

Initializing Azure IoT DPS client
       DPS endpoint: global.azure-devices-provisioning.net
```

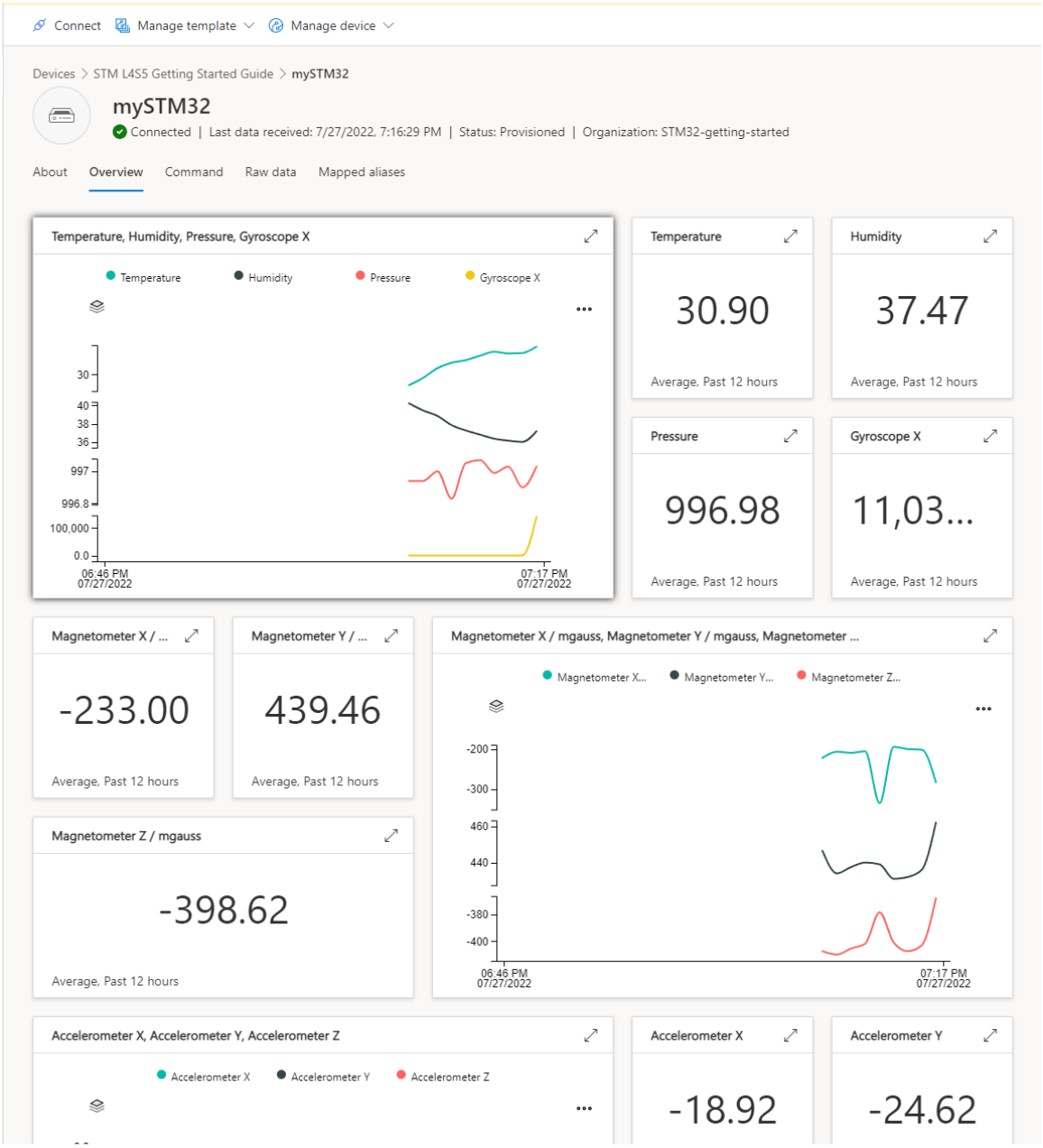
```
       DPS ID scope: 0ne006D0BC8
       Registration ID: mystm32
SUCCESS: Azure IoT DPS client initialized

Initializing Azure IoT Hub client
       Hub hostname: iotc-12a55b58-481c-4eed-a3b5-ab011ba4366b.azure-devices.net
       Device id: mystm32
       Model id: dtmi:azurertos:devkit:gsgstml4s5;2
SUCCESS: Connected to IoT Hub

Receive properties: {"desired":{"$version":1},"reported":{"$version":1}}
Sending property:
$iothub/twin/PATCH/properties/reported/?$rid=3{"deviceInformation":{"__t":"c","manufact
urer":"STMicroelectronics","model":"B-L4S5I-IOT01A","swVersion":"1.0.0","osName":"Azure
RTOS","processorArchitecture":"Arm Cortex
M4","processorManufacturer":"STMicroelectronics","totalStorage":2048,"totalMemory":640}
}
Sending property: $iothub/twin/PATCH/properties/reported/?$rid=5{"ledState":false}
Sending property:
$iothub/twin/PATCH/properties/reported/?$rid=7{"telemetryInterval":{"ac":200,"av":1,"va
lue":10}}

Starting Main loop
Telemetry message sent: {"humidity":40.86,"temperature":28.83,"pressure":996.96}.
```

"Azure IoT" will appear on the little screen; and in the browser, refresh the screen to see the mySTM32 device filled with data.



2.5 Debugging the application

Now, we will step through the code to see how it works.

1. In Visual Studio Code, hit F5.
2. The binary will be downloaded and a breakpoint will be hit within main.c.

main.c

AZ3166 > app > main.c > main(void)

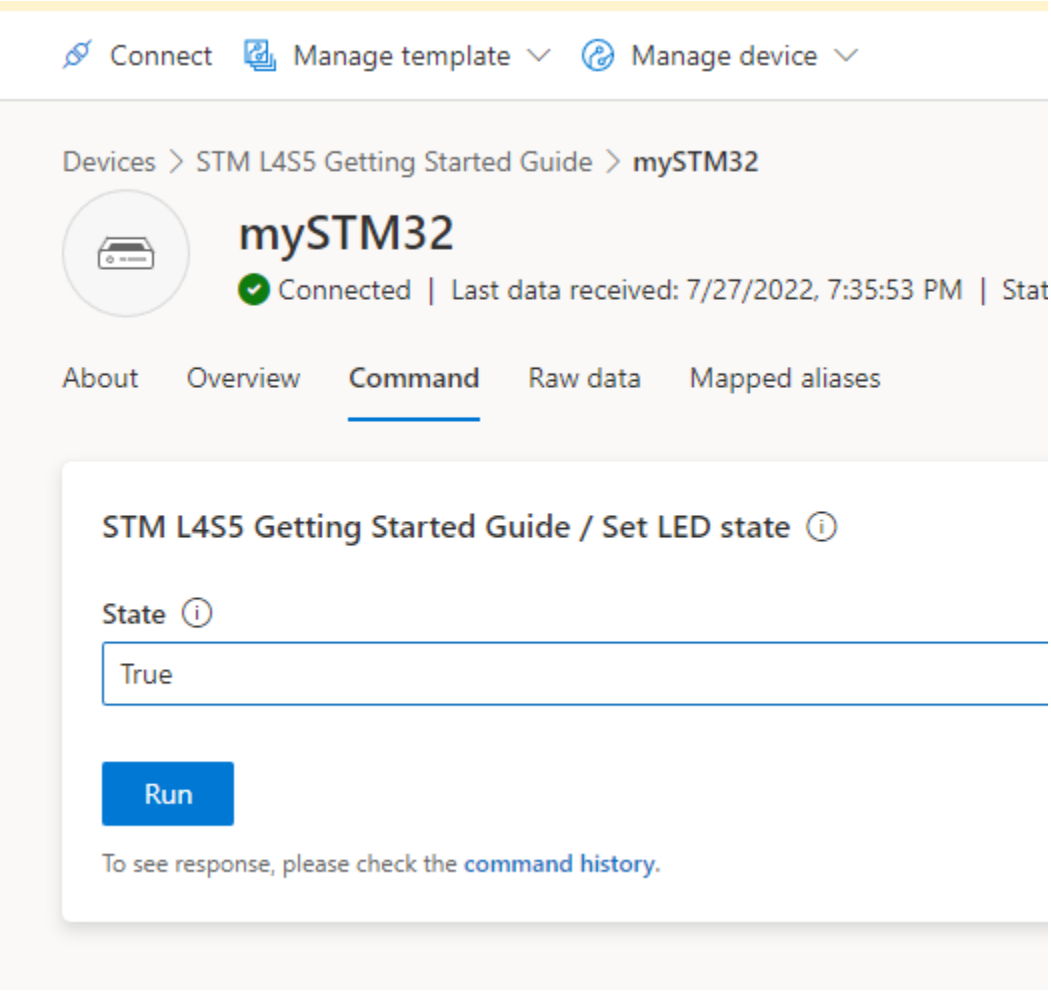
```
49      systick_interval_set(TX_TIMER_TICKS_PER_SECOND);
50
51      // Create Azure thread
52      UINT status = tx_thread_create(&azure_thread,
53          "Azure Thread",
54          azure_thread_entry,
55          0,
56          azure_thread_stack,
57          AZURE_THREAD_STACK_SIZE,
58          AZURE_THREAD_PRIORITY,
59          AZURE_THREAD_PRIORITY,
60          TX_NO_TIME_SLICE,
61          TX_AUTO_START);
62
63      if (status != TX_SUCCESS)
64      {
65          printf("ERROR: Azure IoT thread creation failed\r\n");
66      }
67  }
68
69  int main(void)
70  {
71      // Initialize the board
72      board_init();
73
74      // Enter the ThreadX kernel
75      tx_kernel_enter();
76
77      return 0;
78  }
79
```

3. Click Step Over (F10) to move past the board initialization call.
4. Click Step Over (F10) and the application thread will kick off and run.
5. Stop the debugger (Shift+F5).

The files comprise the core functionality of the application are:

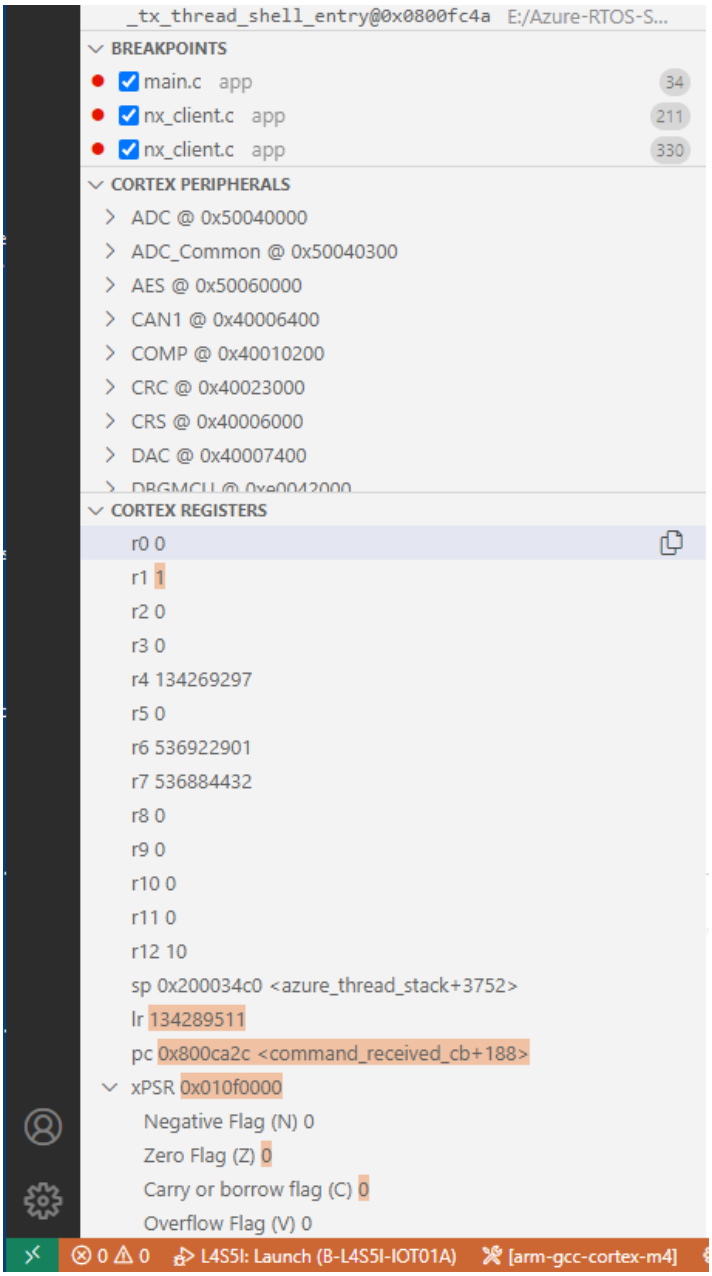
- main.c – sets up and runs the thread.
- nx_client.c – creates the callback to send telemetry and handle receive commands.
- Azure_iot_nx_client.c – this file has the main loop client_run(), which connects to Azure IoT Central and handles communications between the local application and the application on Azure IoT Central.

6. In main.c, set a breakpoint at line 34, which is the call to azure_iot_nx_client_entry.
7. In nx_client.c, set a breakpoint at line 330, which is within the azure_iot_nx_client_entry.
8. Also, in nx_client.c, set another breakpoint at line 211, which is the call to turn the LED on or off.
9. Hit F5.
10. When the breakpoint hits in Main.c, hit F10 twice.
11. The debugger will break at line 34. Hit F11 to step into the to azure_iot_nx_client_entry call.
12. The debugger opens nx_client.c and hits the breakpoint at line 330.
13. Continue to hit F10, but at Line 370, hit F11 to step into azure_iot_nx_client_dps_run.
14. Continue to hit F10, and at line 1199 at the return hit F11.
15. The debugger is now in the main loop in Azure_iot_nx_client.c. In Azure IoT Central, click on Command, set the LED State to True, and click Run.

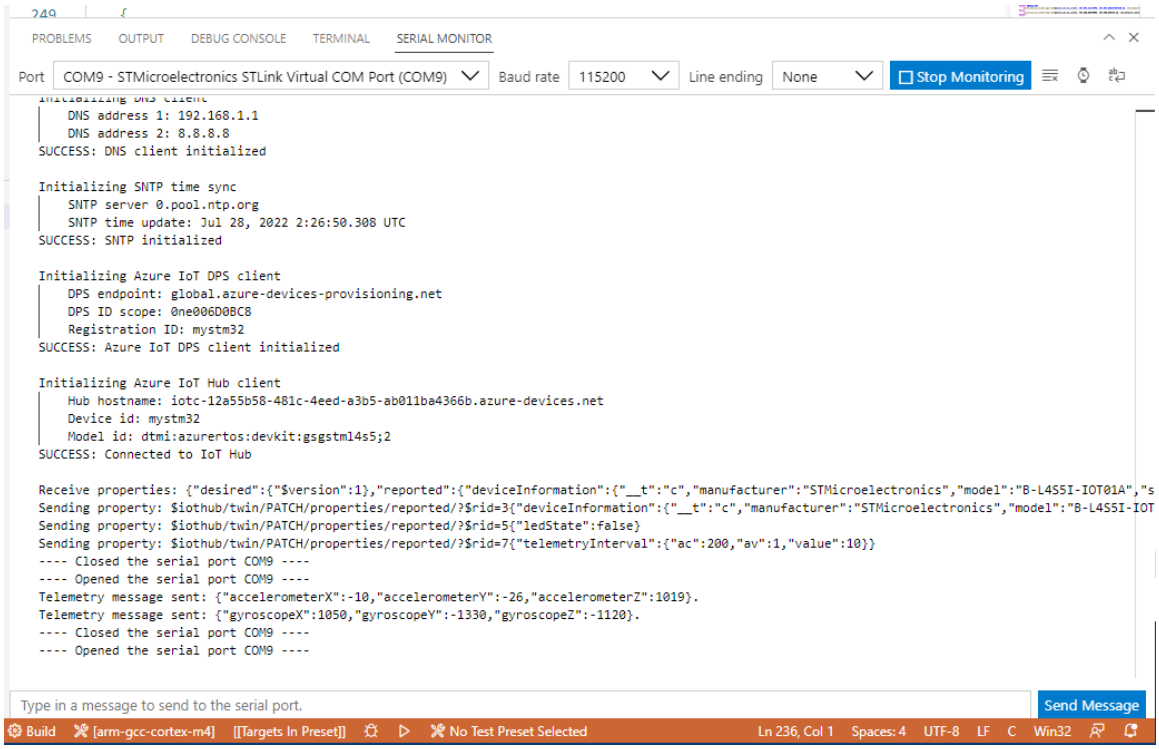


16. Go back and continue to hit F10. Eventually you should hit the breakpoint at line 211 in nx_client.c.
17. Hit F5 to continue debugging, and the LED should turn on.

If you have installed the embedded tools into Visual Studio Code, you will be able to see the Peripherals and Cortex Registers in the Debug section.



In addition, there is a serial monitor that can read the standard output from the board.



18. Hit Shift+F5 to stop debugging.

3 Conclusion

Sample projects are good starting points to get familiar with the software. The ability to step through the code and see the API calls in operation provides good insight when documentation is lacking. The paper here covered debugging with Visual Studio Code, but further development should be using the STM32Cube Integrated Development Environment that provides a richer development experience and direct support for all the STM32 MCU family..

References

More information on the Azure IoT SDKs can be found here.

Introduction to THREADX - stm32mcu