

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

By Sean D. Liming and John R. Malin Annabooks – <u>www.annabooks.com</u>

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 <u>guick start online documents</u>, 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: <u>Git -</u> <u>Downloads (git-scm.com)</u>.
  - 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.

🚸 Git 2.37.1 Setup	-		×
Choosing the default editor used by Git Which editor would you like Git to use?			<b>&gt;&gt;</b>
Use Visual Studio Code as Git's default editor		~	
Visual Studio Code is an Open Source, lightweight and p running as a desktop application. It comes with built-in TypeScript and Node, is and has a rich ecosystem of ex- languages (such as C++, C#, Java, Python, PHP, Go) .NET and Unity). (WARNING) This will be installed only for this user.	powerful editor support for Jav. tensions for oth and runtimes (s	aScript, Ier uch as	
https://gitforwindows.org/ Back	Next	Can	cel

- 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.

Use MinTTY (the default terminal of MSYS2) Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.  Use Windows' default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node, js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 s window was not freely resizable and it only allowed rectangular text selections.	Configuring the terminal emulator to use with G Which terminal emulator do you want to use with you	i <b>it Bash</b> ur Git Bash?		
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via 'winpty' to work in MinTTY. <b>(e) Use Windows' default console window</b> Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.	O Use MinTTY (the default terminal of MSYS	52)		
Use Windows' default console window Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.	Git Bash will use MinTTY as terminal emulator, v non-rectangular selections and a Unicode font. as interactive Python) must be launched via `v	which sports a resize . Windows console p winpty`to work in M	able win program linTTY.	dow, is <mark>(</mark> such
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.	• Use Windows' default console window			
	Git will use the default console window of Wind with Win32 console programs such as interactiv very limited default scroll-back, needs to be co order to display non-ASCII characters correctly window was not freely resizable and it only allo	lows ("cmd.exe"), w ve Python or node. nfigured to use a U y, and prior to Wind owed rectangular te	hich wa is, but h nicode f lows 10 xt selec	rks well las a ont in its tions.
	war il and a state of the			
tps://gitforwindows.org/	tcps;//gitrorwindows.org/			

- j. Keep the defaults for the next question, and click Next.
- k. Select "Use Windows' default console window", and click Next.
- I. 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.

٢	Azure IoT Central				
≡		Build > New application			
ŵ	Home	New application Custom			
61	Build	Answer a few quick questions and we'll get your app up and running.			
₽≞	My apps	About your app			
		Application name * ①			
		STM32-getting-started			
		URL * ①			
		stm32-getting-started .azureiotcentral.com			
		Application template * (i)			
		Custom application			
		Pricing plan			
		<ul> <li>Free</li> <li>Try for 7 days with no commitment</li> </ul>			
		5 free devices			
		Eor devices sending a few messages per day			
		2 free devices 400 messages/mo			
		Standard 1			
		For devices sending a tew messages per hour			
		2 nee devices 5,000 messages/mo			
		Standard 2 (most popular)			
		For devices sending messages every few minutes			
		2 free devices 30,000 messages/mo			
		By clicking "Create" you agree to the Subscription Agreement 🖾 and Privacy Statement 🖾. Provisions in the			
		agreement with respect to pricing, cancellation fees, payment, and data retention do not apply to "Free". "Standard" plans require an Azure subscription, and you acknowledge that this service is licensed to you under			
		the terms applicable to your Azure Subscription 다.			
		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

Devices <	+ New	← Import				🖽 🛆 Ö
Filter templates		All devices				
, in defices		Device explorer helps you	see all your devices. Detailed inforr	nation like device raw data helps	you troubleshoot. Learn more	
		Device name	Device ID	Device status	Device template	Organization
	n	nySTM32	mystm32	Registered	Unassigned	STM32-getting-started

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

🔗 Connect 💁 Manage templat	e \vee 🕜 Manage device 🗸		
Devices > mySTM32 mySTM32   Last data received Raw data Mapped aliases	: N/A   Status: Registered   Organi	ization: STM32-getting-started	
Timestamp ↓	Message type	Event creation time	Unmodeled data
		No rows fo	ound

- 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

PROBLEMS OUTPUT DEBUG CONS	DLE TERMINAL
[build] [1201/1205] Linking	C static library lib\netxduo\addons\azure_iot\azure_iot_security_module\iot-security
[build] [1202/1205] Linking	C static library lib\netxduo\addons\azure_iot\azure_iot_security_module\libiot_secur
[build] [1203/1205] Linking	C static library lib\netxduo\libnetxduo.a
[build] [1204/1205] Linking	C executable app\mxchip_azure_iot.elf
[build] Memory region	Used Size Region Size %age Used
[build] RAM:	119328 B 128 KB 91.04%
[build] FLASH:	625524 B 1 MB 59.65%
[build] CCMRAM:	0 GB 64 KB 0.00%
[build] [1205/1205] cmd.exe	/C "cd /D E:\Azure-RTOS-MXCHIP\getting-started\MXChip\AZ3166\build\app && "C:\Progra
-Obinary mxchip_azure_iot.e	f mxchip_azure_iot.bin && "C:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2021
[build] Build finished with	exit code 0

#### 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	Туре	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
Copyright © 2023 Annabooks, LLC. All rights reserved
Windows is a registered trademark of Microsoft Corporation
```

All other copyrighted, registered, and trademarked material remains the property of the respective owners.



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.

C main.	с	>□ C ↑ ♥ ♥ < 4 0 Ⅲ
AZ3166	> ap	b > C main.c > 分 main(void)
49		<pre>systick_interval_set(TX_TIMER_TICKS_PER_SECOND);</pre>
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
D 72		board_init();
73		
74		// Enter the ThreadX kernel
75		<pre>tx_kernel_enter();</pre>
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.

evices	> STM L4S5 Getting Started Guide > mySTM32
(o)	mySTIVI32
	Connected   Last data received: 7/27/2022, 7:35:53 PM
oout	Overview Command Raw data Mapped aliases
STM	L4S5 Getting Started Guide / Set LED state (i)
STM	L4S5 Getting Started Guide / Set LED state (i)
STM State	L4S5 Getting Started Guide / Set LED state (i)
STM State True	L4S5 Getting Started Guide / Set LED state (i) (i)
STM State True	L4S5 Getting Started Guide / Set LED state (i) (i)
STM State True	L4S5 Getting Started Guide / Set LED state (i) (i)

- 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.

	tx_thread_shell_entry@0x0800fc4a E:/Azure-RTO	S-S
	✓ BREAKPOINTS	
	● 🗹 main.c app	34
	● 🔽 nx_client.c app	211
	● 🗹 nx_client.c app	330
	✓ CORTEX PERIPHERALS	
	> ADC @ 0x50040000	
	> ADC_Common @ 0x50040300	
	> AES @ 0x50060000	
	> CAN1 @ 0x40006400	
	> COMP @ 0x40010200	
	> CRC @ 0x40023000	
	> CRS @ 0x40006000	
	> DAC @ 0x40007400	
	DRGMCLL@_0ve0042000	
	✓ CORTEX REGISTERS	•
	r0 0	Ç
	r1 <mark>1</mark>	
	r2 0	
	r3 0	
	r4 134269297	
	r5 0	
	r6 536922901	
	r7 536884432	
	r8 0	
	r9 0	
	r10 0	
	r11 0	
	r12 10	
	sp 0x200034c0 <azure_thread_stack+3752></azure_thread_stack+3752>	
	lr 134289511	
	pc 0x800ca2c <command_received_cb+188></command_received_cb+188>	
	✓ xPSR 0x010f0000	
Ø	Negative Flag (N) 0	
0	Zero Flag (Z) 0	
ዲሜ	Carry or borrow flag (C) 0	
ഹ്	Overflow Flag (V) 0	
×	⊗ 0 🛆 0 🔥 L4S5I: Launch (B-L4S5I-IOT01A) 🛛 🎇 [arm-gcc-corte	ex-m4] 📢

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



249	Į												300		
PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	SERIAL MONI	TOR										^
ort COM9	- STMicroe	electronics STLink \	/irtual COM Po	rt (COM9) 🚿	<ul> <li>Baud</li> </ul>	rate	115200	$\sim$	Line ending	None	$\sim$	Stop Monit	oring	≡× 〈	D ab
DNS a DNS a SUCCESS: Initializ	address 1: address 2: DNS clien ing SNTP server 0.	192.168.1.1 8.8.8.8 t initialized time sync pool.ntp.org	2,26,50,200	176											
SUCCESS:	SNTP init	ialized	2.201301300												
DPS e DPS I Regis	endpoint: D scope: tration I	global.azure-dev: 0ne006D0BC8 0: mystm32	ices-provisio	ning.net											
SUCCESS:	Azure IoT	DPS client init:	lalized												
Initializ Hub M Devic	ing Azure nostname: te id: mys	IoT Hub client iotc-12a55b58-483 tm32	lc-4eed-a3b5-	ab011ba4366b	.azure-d	evice	s.net								
SUCCESS:	Connected	to IoT Hub	.:gsgstm1455;	2											
Receive p Sending p Sending p Sending p Clos Oper Telemetry	properties property: property: ed the se ned the se med the se	: {"desired":{"\$\ \$iothub/twin/PAT( 5iothub/twin/PAT( 5iothub/twin/PAT( 5iothub/twin/PAT( 5ial port COM9 5ial port COM9 5ial ; {"acceleror	version":1}," CH/properties CH/properties CH/properties  meterX":-10,"	reported":{" /reported/?g /reported/?g /reported/?g acceleromete	deviceIn rid=3{"d rid=5{"l rid=7{"t rY":-26,	forma evice edSta eleme "acce	tion":{"_ Informati te":false tryInterv lerometer	_t":"c on":{" } al":{" Z":101	","manufactu: t":"c","man ac":200,"av" 9}.	rer":"STMi nufacturer :1,"value"	lcroele ":"STM ':10}}	ctronics","mode	1":"B- ","mod	L4S5I- el":"B	IOT01A -L4S5I
Telemetry Clos Oper	/ message sed the se ned the se	sent: {"gyroscope rial port COM9 rial port COM9	⊵X":1050,"gyr  	oscopeY":-13	30,"gyro	scope	Z":-1120}								
ype in a me	ssage to se	nd to the serial po	rt.											Send	Messa
Build 🎇 [a	rm-gcc- <u>cor</u> t	ex-m4] [[Targets	n Preset]] 🛱	D 🎾 No	Test Prese	t Seleo	ted		Lr	1 236, Col 1	Space	es: 4 UTF-8 LF	C I	Nin32	8

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 <u>STM32Cube Integrated Development Environment</u> 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

Windows is a registered trademark of Microsoft Corporation All other copyrighted, registered, and trademarked material remains the property of the respective owners.