

RX65N Group

Visualization of Sensor Data using RX65N Cloud Kit and Azure RTOS

Introduction

This document offers sample code that the RX65N Cloud Kit from Renesas and the Wi-Fi module (SX-SDMAC (from Silex Technology)) included in the kit are used to communicate with the Azure cloud service (Azure IoT Hub) using Microsoft's Azure RTOS.In addition, This document is described how to visualize the temperature data uploaded to Azure IoT Hub via Wi-Fi using a web application.

Azure RTOS is a realtime operating system for connectivity, security, and over-the-air (OTA) updates. Azure RTOS provided by Microsoft includes demo applications for demonstrating the functionality of Azure RTOS. This demo application runs on the RX65N Cloud Kit.

In addition, the RX family is certified by Microsoft as Azure RTOS certified hardware. Therefore, it is available free of charge when using Azure RTOS on RX family MCUs.

Click here for details.<<u>https://github.com/azure-rtos/threadx/blob/master/LICENSED-HARDWARE.txt</u>>

e² studio is a development environment based on the open-source Eclipse CDT (C/C++ Development Tooling) project. In addition to a debugging interface, it provides support for building projects (editor, compiler, linker control).

Target Device

RX65N Group (RX65N Cloud Kit)

• Visit the following webpage for information on boards, related programs and development environments needed for development work using RX cloud solutions. <u>https://www.renesas.com/rx-cloud</u>

• YouTube

The contents described in this application note explained with a video.

Azure RTOS Tutorial (1/3) RX65N Cloud Kit: ~ Development Environment Setup ~ -YouTube Azure RTOS Tutorial (2/3) RX65N Cloud Kit: ~ Set up the program ~ -YouTube Azure RTOS Tutorial (3/3) RX65N Cloud Kit: ~ Operate Azure Cloud ~ -YouTube

• Azure RTOS GitHub Sample Code Azure RTOS Embedded IDE samples

https://github.com/azure-rtos/samples

Azure RTOS Plug and Play sample code https://github.com/azure-rtos/samples/tree/PublicPreview/PnP

Azure RTOS ADU sample code <u>https://github.com/azure-rtos/samples/tree/PublicPreview/ADU</u>



Contents

1.	Overview	3
1.1	System Diagram	3
2.	Preparation	4
2.1	Hardware Configuration	4
2.2	About Azure and Azure RTOS	4
2.3	Software Configuration	5
3.	Connecting to Azure	6
3.1	Azure Preparation	6
3.1.	1 Sign in to Azure	6
3.1.2	2 Create an Azure IoT Hub	7
3.1.3	3 Create an IoT Device	11
3.2	Software Preparation	14
3.3	Running the Demo Program Preparation	19
3.3.	1 Hardware Preparation	19
3.3.2	2 Tera Term Preparation	20
3.4	Running the Demo Program	21
3.5	Communication confirmation by Azure IoT Explorer	22
4.	LED ON/OFF operation by Azure IoT Explorer	28
5.	Visualization of Sensor Data	31
5.1	Software Preparation	31
5.2	Running the web application	
5.3	Display of graph changes	
6.	Important Note after Running Demo Program	36
7.	Websites and Support	36
8.	Appendices	36
Rev	vision History	41



1. Overview

1.1 System Diagram

The system diagram below shows the steps from getting the sensor data of the RX65N Cloud Kit to visualization and the use of the Azure IoT Explorer service to control RX65N Cloud Kit.



Figure 1.1System Diagram



Figure 1.2 Visualization of sensor data



2. Preparation

2.1 Hardware Configuration

The hardware configuration of the system is listed in the table below.

			-
Item	Content	Provider	Description
Board used (packaged with	Target Board for RX65N	Renesas Electronics	Evaluation board mounted with RX65N MCU ^{*1}
RX65N Cloud Kit)	RX Cloud Option Board	- Corporation	Cloud communication evaluation board capable of connecting to Azure ^{*1}
	Silex Pmod Module		Communication board mounted with wireless LAN module ^{*1}
Wi-Fi	Wireless router	-	Wireless LAN standard: IEEE 802.11b/g/n (2.4 GHz) Encryption method: AES
PC	Windows 10	-	Recommended OS
	Google Chrome	-	Web browser used
Note: 1. The target	board for RX65N, R	X65N cloud option b	board, and Silex Pmod module are included in
RX65N Cl	oud Kit.		

Table 2.1 Hardware Configurations

RX65N Cloud Kit Web:

https://www.renesas.com/products/microcontrollers-microprocessors/rx-32-bit-performance-efficiencymcus/rx65n-cloud-kit-renesas-rx65n-cloud-kit

2.2 About Azure and Azure RTOS

Azure is a cloud computing service provided by Microsoft.

Azure RTOS, which is a real-time operating system for microcomputers provided by Microsoft, has a library for connecting the microcomputer and Azure, and can manage and control IoT devices connected to the cloud.



2.3 Software Configuration

The software configuration of the system is listed in the table below.

Item	Content	Version
Integrated development environment	e ² studio	2021-04
Compiler	GCC for Renesas RX	8.3.0.202004
Communication software	Tera Term	Version4.105
Tool for device interaction	Azure IoT Explorer	Version 0.14.3
Tool for web application download	Git	2.31.1
Tool for web application running	node.js (npm)	14.17.0
Emulator	E2 emulator Lite (on-board)	-

Table 2.2 Software Configurations

The software download sites used in this system are shown in the table below.

Note: Link destinations are subject to change.

Table 2.3 Tool download sites

Content	Link
e ² studio	https://www.renesas.com/kr/software-tool/e-studio
GCC for Renesas RX	https://llvm-gcc-renesas.com/rx-download-toolchains
Tera Term	https://osdn.net/projects/ttssh2/
Azure IoT Explorer	https://github.com/Azure/azure-iot-explorer/releases
Git	https://www.git-scm.com/download
node.js (npm)	http://nodejs.org



3. Connecting to Azure

The following preparation is necessary in order to connect RX65N Cloud Kit to Azure.

3.1 Azure Preparation

3.1.1 Sign in to Azure

- 1. Sign in to Azure.
- Azure Sign-in page as follows.

https://azure.microsoft.com

If you do not have an Azure account, create one from the Free account on the screen.



Figure 3.1 Sign-in screen

2. After signing in, go to the Azure portal page.



Figure 3.2 Azure Portal selection screen



3.1.2 Create an Azure IoT Hub

1. Click **Create a resource**.

= Micr	osoft Azure	€ P Search n	esources, services, and	d docs (G+/)			Σ
Az	ure services						
Г	+	K.	+	8	\$	6	32
	Create a resource	IoT Hub	Subscriptions	Users	Cost Management	Cost Management	Event Hubs

Figure 3.3 Create a resource

2. Enter "IoT Hub" and click IoT Hub from the displayed candidates.

Home >		
Create a resource	e	
Get started	P IoT Hub	X Getting Started? Try our Quickstart center
Recently created	ioT Hub ្សាក្	
Categories	IoT Hub Device Provisioning Service Device Update for IoT Hub	
AI + Machine Learning	Crosser IoT Connectivity & Streaming Analytics	
	BOUND DE VELEVOTEIS	

Figure 3.4 IoT Hub Search \rightarrow IoT Hub Selection

3. Click Create.



Figure 3.5 IoT Hub Create



RX65N Cloud Kit and Azure RTOS

4. On the Basics tab, select the **Subscription** that suitable your environment then enter the **Resource** group, Region and IoT hub name* \rightarrow Click Next: Networking >. Note: The IoT hub name must be unique across the Azure cloud.

IoT hub		
Microsoft		
Basics Networking Mana	gement Tags Review + create	
Create an IoT hub to help you con	nect, monitor, and manage billions of your IoT assets. Learn more	
Project details		
Choose the subscription you'll use organize and manage resources.	to manage deployments and costs. Use resource groups like folders	to help you
Subscription * 🕡	subscription1	\sim
Resource group * ①	(New) azure-seminar-ResourceGroup1 Create new	\checkmark
Region * ①	East US	\sim
IoT hub name * 🛈	azure-seminar-iothub-001	\checkmark

Figure 3.6 IoT Hub Basics Information Settings \rightarrow Next: Networking >

5. On the Networking tab, select **Public endpoint** \rightarrow Click **Next: Management >**.

■ Microsoft Azure	ervices, and docs (G+/)
Home > Create a resource > IoT Hub >	
loT hub	
Microsoft	
Basics Networking Management Tags	Review + create
Network connectivity	
Connect to your lot Hub using public or private endpoin	nts.
	endpoint (all networks)
0	endpoint orks will have access to this IoT hub.
	ore about connectivity methods.
Review + create < Previous: Basics	Next: Management >

Figure 3.7 Select Public endpoint \rightarrow Next: Management >



6. On the Management tab, select the **Pricing and scale tier** that suitable your environment \rightarrow Click **Next: Tags** >.

If you select a Pricing and scale tier other than the F1: Free tier, set the subsequent setting items as necessary.

	Hub >			
IoT hub … Microsoft				
Basics Networking Manag	jement Tag s R	eview + create		
Each IoT hub is provisioned with a c maximum daily quota of messages		in a specific tier. The tier and number of rn more	units determine the	
Scale tier and units				
Pricing and scale tier * ①	F1: Free tier		\sim	
		Learn how to choose the right IoT hul	o tier for your solution	
Number of F1 IoT hub units ①	0		1	
	termines how your IoT	hub can scale. You can change this later i	f your needs increase.	
Defender for IoT	off			
Defender for IoT	off	hub can scale. You can change this later i protection to IcT Hub, IoT Edge, and you		
Defender for IoT	off			
Defender for IoT T <mark>urn</mark> on Def <mark>end</mark> er fo <mark>r I</mark> oT and add	Off an extra layer of threat	protection to IcT Hub, IoT Edge, and you	ur devices. Learn more	
Defender for IoT Turn on Defender for IoT and add Pricing and scale tier ①	Off an extra layer of threat F1	protection to IcT Hub, IoT Edge, and you Device-to-coud-messages ①	ur devices. Learn more Enabled	

Figure 3.8 Select the Pricing and scale tier \rightarrow Next: Tags >

7. On the Tags tab, set tags as needed \rightarrow Click Next: Review + create >.

This document does not require any input, so do not enter anything and click Next: Review + create >.

E Microsoft Azure	𝒫 Search resources, services	s, and docs (G+/)	
Home > Create a resource	> IoT Hub >		
IoT hub … _{Microsoft}			
Tags are name/value pairs. T	Management Tags Revie categorize resources and conso ill update automatically if you ch	lidate billing, apply the same tag	
Name 🛈	Value ①	Resource	
		IoT Hub	
Review + create	< Previous: Management	Next: Review + create >	

Figure 3.9 Set tags \rightarrow Next: Review + create >



8. On the Review + create tab, review your selection \rightarrow Click **Create** if you are satisfied. Creating an IoT Hub takes a few minutes.

😑 🛛 Microsoft Azure 🛛 🔎	Search resources, services, and docs (G+/)	
Home > Create a resource > IoT	Hub >	
IoT hub … Microsoft		
Validation passed.		
Basics Networking Manag	ement Tags Review + create	
Basics		
Subscription	subscription1	
Resource group	azure-seminar-ResourceGroup1	
Region	East US	
IoT hub name	azure-seminar-iothub-001	
Networking		
Connectivity method	Public endpoint (all networks)	
Private endpoint connections	None	
Management		
Pricing and scale tier	F1	
Number of F1 IoT hub units	1	
Messages per day	8,000	
Device-to-cloud partitions	2	
	STOCK STOCK	

Figure 3.10 Review your selection \rightarrow Create

9. Once the IoT Hub is created, click **Go to resource**.



Figure 3.11 Go to resource



10. Make a note of the **Hostname** with a text editor. The Hostname information will be used later.

Microsoft Azure	arch resources, services, and docs (G+/)	E 6 0 7	R	
Acome > azure-seminar-ion IoT Hub P Bearch (Ctrl+/)	thub-001 & … « → Move ∨ ■ Delete ♡ Refresh			×
X Overview	Senetials			JSON View
Activity log	Resource group (change) azure-seminar-ResourceGroup1	Hostname azure-seminar-iothub-001.azure-devic	es.net	
Access control (IAM) Tags	Status Active	Pricing and scale tier F1 - Free		
Diagnose and solve problems	Current location East US	Number of IoT Hub units 1		
F Events	Subscription (change) subscription1	Minimum TLS Version		
iettings	Subscription ID			

Figure 3.12 IoT Hub Hostname recording

3.1.3 Create an IoT Device

1. Click **IoT devices** on the left side of the screen \rightarrow Click **Add Device**.

■ Microsoft Azure	𝒫 Search resources, services, and docs (G+/)	🖂 🕼 🖓 🎯 ?	ጽ
Home > IoT Hub > azure-s	eminar-iothub-001		
azure-semina	ar-iothub-001 IoT devices 🛷 …		
	 View, create, delete, and update devices in your IoT Hub. 		
O Pricing and scale	Device name		
Networking	enter device ID		
🔎 Certificates	Find devices		
Built-in endpoints	+ Add Device 💍 Refresh 📋 Delete		
- Sailover			
😂 Properties	Device ID Status	Last Status Update	
🔒 Locks			
Explorers	There are no IoT devices to display.		
Query explorer			
IoT devices			
\odot			

Figure 3.13 IoT devices \rightarrow Add Device



2. In Create a Device, enter and select the following information \rightarrow Click **Save**. \rightarrow Enter arbitrary name

- Device ID

Authentication type

- → Symmetric key
- Auto-generate keys \rightarrow Put a check
- connect this device to an IoT hub \rightarrow Enable

	Microsoft Azure P Search resources, services, and docs (G+/)
Hom	$_{2}$ > azure-seminar-iothub-001-76172937 $>$ azure-seminar-iothub-001 $>$
Æ	Create a device
C	Find Certified for Azure IoT devices in the Device Catalog
Devic	e ID * (i)
azu	e-seminar-deviceid
	ntication type ① metric key) X.509 Self-Signed X.509 CA Signed
Prima	ry key 🕕
Ente	r your primary key
Secor	dary key 🕕
Ente	r your secondary key
Auto-	generate keys 🕕
	ect this device to an IoT hub ① ble Disable
Paren	t device 🛈
	parent device a parent device
2	ave

Figure 3.14 Device ID information settings \rightarrow Save

3. Make a note of the **Device ID name** with a text editor and click the **Device ID**. The Device ID name information will be used later.

	Microsoft Azure	₽ Search	n resources, services, and docs (G+/)		Ŗ	2		? A?
Hom	e > IoT Hub > azure-se	eminar-ioth	ub-001					
	azure-semina	ar-ioth	ub-001 IoT devices	\$ ² ···				
100	IoT Hub							
₽ s	Search (Ctrl+/)	*	View, create, delete, and update device	es in your IoT Hub.				
0 p	Pricing and scale	-	Device name					
<-> N	Vetworking		enter device ID					
🔎 с	Certificates		Find devices					
	Built-in endpoints	- 1	+ Add Device 🖒 Refresh 🗐	Delete				
- 1 F	ailover	- 1						
P P	Properties	- 1	Device ID	Status	Last	Status Up	date	
Αu	.ocks	- 1			2001	status op		
Explo	orers		azure-seminar-deviceid	Enabled				

Figure 3.15 Click Device ID



4. Make a note of the **Primary Key** with a text editor. The Primary Key information will be used later.

😑 🛛 Microsoft Azure 🛛 🖉	Search resources, services, and docs (G+/) 🛛 💀 🤌 🛞 🦓 🧖	MY DIRECTOR	ev 🐇
Home > azure-seminar-iothub-0	001-76172937 > azure-seminar-iothub-001 >		
azure-seminar-dev	iceid &		Х
azure-seminar-iothub-001			15.52
Save 🖾 Message to Device	🗡 Direct Method 🕂 Add Module Identity 🔲 Device twin 🔍 Manage keys 🗠 🖒 Refresh		
an sone in strage to before	/ Discrimented Hod module dentity = Dence thin - Q manage wys - O terreat	-	
Device ID	azure-seminar-deviceid		0
Primary Key 🜒		0	0
Secondary Key 🌘		۲	0
rimary Connection String 🕚		۵	0
	[0	0
Secondary Connection String 👩			
Secondary Connection String 🕚			

Figure 3.16 Device ID Primary Key recording



3.2 Software Preparation

Follow the steps below to prepare the software for the demo program.

1. Download the sample project.

Download the Azure RTOS project for the RX65N Cloud Kit and GCC compiler combination from the Azure RTOS sample page on GitHub.

• GitHub sample page as follows.

https://github.com/azure-rtos/samples

Figure 3.17 GitHub sample page

2. Extract the project files from the archive and copy them to a suitable location. (In the description below, the root folder containing the project files is designated as \${base_folder}.)

Note: After extracting the project files from the archive, copy them to a location with a short file path, such as the root folder on the C: drive. If the file path is too long, a build error may result.



3. Launch e² studio and specify a workspace directory.

📴 e ^z studio Launcher	×
Select a directory as workspace	
e ² studio uses the workspace directory to store its preferences	and development artifacts.
Workspace: C:\e2workspace_azure	✓ Browse
Use this as the default and do not ask again	
	Launch Cancel

Figure 3.18 Workspace selection screen

eworkspace_azure - e ² studio File Edit New Alt+Shift+N > Open File Open Projects from File System Recent Files Close All Editors Ctrl+Shift+S Save All Ctrl+Shift+S Revert Move Rename File Refresh Convert Line Delimiters To Print Ctrl+P Print Chrypties Alt+Enter	4. Select File → Import .			
New Alt+Shift+N > h Configurations Open File Open Projects from File System File - C* Recent Files > diose Editor Ctrl+W Cbee All Editors Ctrl+Shift+W Save Ctrl+Shift+W Save All Ctrl+Shift+W Move Revert Move F2 Rename F5 Convert Line Delimiters To > Print Ctrl+P Import Ctrl+N		e2workspace_azure - e ² stu	dio	
Open File Open Projects from File System Recent Files Close Editor Ctrl+ Shift+ W Save Copen All Editors Ctrl+ Shift+ W Save As Save As Save As Revert Move Refresh Convert Lirle Delimiters To Print Ctrl+ P Import Export		File Edit Source Refactor	Navigate Search Project Renesas Views	
Close All Editors Ctrl+Shift+W Save Ctrl+S SaveAs SaveAs SaveAs Evert Move F2 Rename F5 Convert Line Delimiters To > Print Ctrl+P Import Ctrl+P		Open File Open Projects from File Sy	/stem	
Save As Save All Ctrl+Shift+S Revert Move Rename F2 Refresh F5 Convert Line Delimiters To > Print Ctrl+P Move				
Revert Move Rename F2 Refresh Convert Line Delimiters To Print Ctrl+P			Ctrl+S	
Rename F2 Refresh F5 Convert Line Delimiters To > Print Ctrl+P Import Ctrl+P Export F			Ctrl+Shift+S	
Convert Line Delimiters To > Print Ctrl+P Import Export		Rename.		
import <u> <u> </u> <u> </u> <u> </u> <u> </u> Export </u>				
Export		_	Ctrl+P	
Properties Alt+Enter				
Switch Workspace >				
Switch Workspace > Restart Exit		Restart	,	

Figure 3.19 Select File \rightarrow Import...



5.	Click	General	\rightarrow Existing	Projects	into	Worksp	ace \rightarrow N	ext >
۰.	0.000	••••••						

Import -		
Select Choose import wizard.	Ľ	
Select an import wizard: type filter text Select an import wizard: type filter text Constant of the selection of the system File System File System Preferences Preferences Preferences Renawa & Import Existing C/C++ Project into Workspace Renewas CCRX project conversion to Renewas GCC RX Renewas CCRX project for CA78K0R/CA78K0 Renewas CS+ Project for CA78K0R/CA78K0 Renewas CS+ Project for CC-RX and CC-RL Renewas CS+ Project for CC-RX and CC-RL C/C++ C/C++ Code Generator C/C++ Comph	Cancel	

Figure 3.20 General \rightarrow Existing Projects into Workspace \rightarrow Next >

6. Click **Browse...**, specify $\delta = \frac{1}{rx65n-cloud-kit folder}$ select all of the sample project extracted $\rightarrow Click$ **Finish**.

S Import	– 🗆 X
Import Projects Select a directory to search for existing Eclipse projects.	
Select root directory: C:\azure_rtos\rx65n-cloud-kit Select archive file: Projects: filex (C:\azure_rtos\rx65n-cloud-kit\e2studio_gnurx\filex) netxduo_addons (C:\azure_rtos\rx65n-cloud-kit\e2studio_gnurx\ntextudio	u Deselect All Refresh
sample_filex_ramdisk (C:\azure_tos\rx65n-cloud-kit\e2studio, sample_netxduo_ping (C:\azure_tos\rx65n-cloud-kit\e2studio sample_pnp_temperature_controller (C:\azure_tos\rx65n-cloud-kit\e2studio sample_pnp_temperature_controller (C:\azure_tos\rx65n-cloud-kit\e2studio sample_pnp_temperature_controller (C:\azure_tos\rx65n-cloud-kit\e2studio sample_nop_temperature_controller (C:\azure_tos\rx65n-cloud-kit\e2studio cloud-kit\e2studio cloud-kit\e2stud	- d
Working sets Add project to working sets Working sets:	New > Select
? < Back Next > Finish	Cancel

Figure 3.21 Project import screen



7. Define the following two macros in

{\$base_folder}/rx65n-cloud-kit/e2studio_gnurx/sample_pnp_temperature_controller/src/main.c.

- WIFI_SSID \rightarrow The SSID of the access point to connect to.

- WIFI_PASSWORD \rightarrow The password of the access point to connect to.

(Makes sure to enclose the above macro definitions in quotes (" ") as shown in the figure below.



Figure 3.22 main.c

8. Define the following three macros in

 $\{\text{base_folder}/\text{rx65n-cloud-kit/e2studio_gnurx/sample_pnp_temperature_controller/src/sample_config.h.} - HOST_NAME \rightarrow The hostname confirmed as described in 3.1, Azure Preparation.}$

- DEVICE_ID \rightarrow The Device ID created as described in 3.1, Azure Preparation.

— DEVICE_SYMMETRIC_KEY → The Primary Key confirmed as described in 3.1, Azure Preparation.

<pre>/* Required when DPS is not used. */ /* These values can be picked from device</pre>	connection string which is of format : HostName= <host1>;DeviceId=<device1>;SharedAcce</device1></host1>	essKev=4
HOST NAME can be set to <host1>,</host1>	connection set ing mach as of format i november mestar perfaction contect	, some y
DEVICE ID can be set to <device1>,</device1>		
DEVICE_SYMMETRIC_KEY can be set to <key< th=""><th>1>. */</th><th></th></key<>	1>. */	
#ifndef HOST_NAME		
#define HOST_NAME	"azure-seminar-iothub-001.azure-devices.net"	
Wendit / HOST_NAME /		
#ifndef DEVICE ID		
#define DEVICE ID	"azure-seminar-deviceid"	
<pre>#endif /* DEVICE_ID */</pre>		
<pre>#else /* IENABLE_DPS_SAMPLE */</pre>		
/* Required when DPS is used. */		
#ifndef ENDPOINT		
#define ENDPOINT		
<pre>#endif /* ENDPOINT */</pre>		
#ifndef ID SCOPE		
#define ID SCOPE	**	
<pre>#endif /* ID_SCOPE */</pre>		
#ifndef REGISTRATION ID		
#define REGISTRATION ID	**	
<pre>#endif /* REGISTRATION_ID */</pre>		
<pre>#endif /* ENABLE_DPS_SAMPLE */</pre>		
/* Optional SYMMETRIC KEY. */		
#ifndef DEVICE_SYMMETRIC_KEY		
#define DEVICE_SYMMETRIC_KEY	the second se	

Figure 3.23 sample_config.h





e2workspace_azure - sample_pnp_temperature_	contre	oller/src/sample_config.h - e ²	studio
File Edit Source Refactor Navigate Search	Pro	ject Renesas Views Run	Window Help
🍝 🔯 🔳 🎄 Debug 🗸		Open Project	
😕 🛷 👻 🗾 🗊 🖷 😫 🗸 🤸 🏷 😅		Close Project	0.1.4% 0
🎦 Project Explorer 💥 🛛 🖨 🕏 🖓 🖇 🖵	010	Build All Build Configurations	Ctrl+Alt+B
> 😂 filex > 😂 netxduo		Build Project	Ctrl+B
> 📂 netxduo_addons	_	Build Working Set	>
> 📂 sample_azure_iot_embedded_sdk > 🞏 sample_azure_iot_embedded_sdk_pnp		Clean	
> 😰 sample_azore_ior_embedded_sdk_prip		Build Automatically	
> 📂 sample_netxduo_ping		Build Targets	>
✓ Sample_pnp_temperature_controller [Hard > S Includes	^	C/C++ Index	>
✓ [™] src	e ²	Update All Dependencies	Alt+D
> 🧽 r_wifi_sx_ulpgn		Change Device	
> 🧀 smc_gen > 🔀 compat.c	\$	C/C++ Project Settings	Ctrl+Alt+P
> 🖻 demo_printf.c		Properties	

Figure 3.24 Project \rightarrow Clean...

10. Uncheck **Clean all projects**, check only **sample_pnp_temperature_controller**, click **Clean**, and confirm that **0 errors** are reported.

Clean	- 0	×		
Clean discards all build results and states. The next time a build projects will be rebuilt from scratch.	occurs the select	ed		
Clean all projects				
type filter text				
🗆 😂 filex				
🔲 🗁 netiduo				
🗌 😂 netx duo_addons				
^C sample_azure_iot_embedded_sdk				
Sample_azure_iot_embedded_sdk_pnp Sample_filex_ramdisk				
Sample Tilex_ramoisk				
Sample_pnp_temperature_controller			-	
□ 🗁 sample_threadx				
□ 😂 threadx				
Start a build immediately				
Build the entire workspace				
O Build only the selected projects				
Clean	Can	cel		_

Figure 3.25 Clean setting screen



Figure 3.26 Clean completion screen



3.3 Running the Demo Program Preparation

Prepare to run the demo program.

3.3.1 Hardware Preparation

- 1. Remove the jumper from the EJ2 pins on the target board (bottom board).
- 2. Connect the ECN1 connector on the target board (bottom board) to the PC via a USB cable.
- 3. Connect the CN18 connector on the cloud option board (top board) to the PC via a USB cable.



Figure 3.27 RX65N Cloud Kit (Top)



Figure 3.28 RX65N Cloud Kit (Bottom)



3.3.2 Tera Term Preparation Launch Tera Term and set as shown in the table below.

Table 3.1 Tera Term Settings

Item	Setting
Baud rate	115,200
Data length	8bit
Parity	none
Stop bits	1bit
Flow control	none



3.4 Running the Demo Program

Follow the steps below to run the demo program.

1. Click **sample_pnp_temperature_controller HardwareDebug** from the drop-down list in the upper left of the e² studio window.

e2workspace_azure - sample_pnp_temperature_	controller/src/sample_config.h - e² studio
File Edit Source Refactor Navigate Search	Project Renesas Views Run Window Help
🍕 🎋 🔳 🔻 Debug 🗸	💽 sample_threadx HardwareDebug 🔨 🔅 🗂 🕶 🔚 🌚
(四) 세····································	sample_threadx HardwareDebug
Image: Project Explorer ⋈ Image: Project Explorer ⋈ > ﷺ filex	sample_pnp_temperature_controller HardwareDebug
> 👺 netxduo	🔄 sample_netxduo_ping HardwareDebug
> 6 sample_azure_iot_embedded_sdk > 6 sample_azure_iot_embedded_sdk > 6 sample_azure_iot_embedded_sdk_pnp	sample_azure_iot_embedded_sdk HardwareDebug
> 😰 sample_filex_ramdisk	cs sample_azure_iot_embedded_sdk_pnp HardwareDebug
 > Sample_netxduo_ping > Sample_pnp_temperature_controller [Hardy 	sample_filex_ramdisk HardwareDebug
> 🔆 Binaries	💽 sample_netxduo_ping HardwareDebug 🗸 🗸
> 🔊 includes 🗸 🥵 src	New Launch Configuration
N 🧰 r wifi sy ulgan	

Figure 3.29 Hardware Debug selection

2. Click the D	Debug icon.	
	🔨 🐞 🔳 🔅 Debug 🗸 💽 sample_pnp_temperature_controll 🗸 🄅	
	Figure 3.30 Debug	

3. A message appears asking you to confirm that you wish to switch to the Debug perspective; click the **Switch** button.

4. Click the **Restart** icon. After a short time execution pauses at the main function; click the **Resume** icon.

ature_controll 🗸 🔅	- 🖻 -	🔊 - 🐔	-	a 💵		5 24	 i+ 👼	 *	Q .	÷ 0 ₀₀ •	***	000 100 1	2	3
		 			_								_	

Figure 3.31 Demo program execution

5. Confirm that the execution log is output on the Tera Term screen.



Figure 3.32 Demo program execution log



3.5 Communication confirmation by Azure IoT Explorer

Confirm the uploaded data is sent to the Azure cloud.

1. Access the created IoT Hub page on the Azure portal and the Shared access policies \rightarrow iothubowner \rightarrow Get Primary connection string.

ome > IoT Hub > azure-seminar-iothu azure-seminar-iothu IoT Hub	ıb-001 Shared access poli	cies \$	iothubowner azure-seminar-iothub-001 Regenerate primary key	×
O Search (Ctrl+/) «	2			
Overview	Shared access policies are used to generate sec + Add shared access policy O Refresh	Delete	Primary key)
Activity log	1 Add shared access pointy O remean	E peiere	Secondary key	
A Access control (IAM)	Policy Name	Permissions		▶ []
Tags			Primary connection string	
Diagnose and solve problems	✓ iothubowner	Registry Read, F		
Events	service	Service Connect	Secondary connection string	
ettings	device	Device Connect		∞ [
Shared access policies	registryRead	Registry Read	Permissions	
Identity			Registry Read	
Pricing and scale	registryReadWrite	Registry Read, F	Registry Write	
Networking			Service Connect	
Certificates			V Device Connect	
Built-in endpoints				
Failover				
Properties				

Figure 3.33 shared access policies \rightarrow iothubowner \rightarrow Primary connection string

2. Launch Azure IoT Explorer and click **Add connection**.

Azure IoT Explorer (preview) Azure IoT Explorer (preview) Ottifications
Home > IoT hubs IoT hubs IoT hubs IoT Plug and Play IoT Notification Center Notification Center Volume Value Volume Volume

Figure 3.34 Azure IoT Explorer



3. Enter the got **Primary connection string** in **Connection string** \rightarrow Click **Save**.

 Azure IoT Explorer (preview) File Edit View Window Hel 		– 🗆 X
Azure IoT Explorer (p	oreview)	Notifications Settings
Home > IoT hubs		Add connection string $ imes$
=	+ Add confin	Connection string *
සි loT hubs ග් loT Plug and Play O Notification Center	No conne You will nee storage and Help: Where do L	enter a connection string Where do I get an IoT hub connection string? Please do not save your hub connection string to any unsafe locations
		Save

Figure 3.35 Enter the primary connection string in Azure IoT Explorer \rightarrow Save

4. Create a folder named models in your local folder.

5. Go to the IoT Plug and Play quickstarts web page and download the two model files into the models folder.

· IoT Plug and Play quickstarts web page as follows.

https://docs.microsoft.com/azure/iot-develop/set-up-environment



Figure 3.36 Download model file



6. Click **IoT Plug and Play components** in Azure IoT Explorer \rightarrow **Add** \rightarrow Click **Local folder** in the pull-down.

Azure IoT Explorer (preview) File Edit View Window Hell	lp	-	
Azure IoT Explorer (p	preview)	Notifications	🌀 Settings
Home > IoT Plug a	No mo Model repository definitio by returned a construction of the second secon	Help to display re the application looks to find IoT Plug and Play mode ication storage and can be edited or removed at any ti	

Figure 3.37 IoT Plug and Play components \rightarrow Add \rightarrow Local folder

7. Pick a folder \rightarrow Select the models folder where you saved the model file \rightarrow Click Save.

Azure IoT Explorer (p	preview)	Notifications	🍘 Settings
Home > IoT Plug	and Play components		
 ■ ♣ IoT hubs ^g IoT Plug and Pla ^Q Notification Center 	Image: Savg + Add ∨ ? Revert ? Help We'll Rok for your model definition in the for Click 'Aot' to enable more ways to can resol Before enabling us to retrieve model definiti Statement 1 Local Folder * Selected folder C/models	ve your model definitions.	

Figure 3.38 Pick a folder \rightarrow Select the models folder where you saved the model file \rightarrow Save

8. Click **IoT hubs** on the left side of the screen \rightarrow **Created IoT Hub**.

Azure IoT Explorer (preview) File Edit View Window He		- 0 ×
Azure IoT Explorer (preview)	Notifications 💿 Settings
Home > IoT hubs		
=	+ Add connection	
品 IoT hubs ,		
5 ^g IoT Plug and Play	azure-seminar-iothub-001	1 🗊
Q Notification Center	azure-seminar-iothub-001.azure-devices.net	D
	Shared access policy name	
	iothubowner	D

Figure 3.39 IoT Hub selection



9. Click the **Device ID name**.

P Azure IoT Explorer (preview) File Edit View Window Help					- 🗆 X
Azure IoT Explorer (preview)			9	Notifications	Settings
Home > azure-seminar-io	othub-001 > Devi	ces			
New Nefresh Delete		y parameter	Last status up	loT Plug and	Edge device
azure-seminar- deviceid Enabled	Disconnected	Sas		dtmi:com:exam ple:Temperatur	Euge device
4				ple:Temperatur eController;1	

Figure 3.40 Device ID selection

10. Click the **IoT Plug and Play components**.

Azure IoT Explorer (preview File Edit View Window H		
Azure IoT Explorer (preview) 🚨 Notifications	Settings
Home > azure-se	eminar-iothub-001 > <u>Devices</u> > azure-seminar-deviceid > Device ide	ntity
=	🗟 Save 🔍 Manage keys 🗸	
Device identity		
🔁 Device twin	Device identity	
C Telemetry	Device ID ①	^ _
> Direct method	azure-seminar-deviceid	Đ
S. Direct method	Primary key 💿	
Cloud-to-device	•••••	D
🛠 Module identities	Secondary key 💿	
og IoT Plug and Play	•••••	D
Dr lot ridy and ridy	Primary connection string ①	

Figure 3.41 Device ID selection



11. Scroll down the page and click thermostat1 from Components.

Azure IoT Explorer (preview File Edit View Window H Azure IoT Explorer	Help		×
		e-seminar-deviceid > IoT Plug and Pla	
=	🕐 Refresh		
 Device identity Device twin Telemetry 	LoT Plug and Play components) Model ID	•
Direct method Cloud-to-device	Default component	dtmi:com:example:Temperatur eController;1	
🛠 Module identities 🖉 loT Plug and Pla	thermostat1	dtmi:com:example:Thermostat; 1	
	thermostat2	dtmi:com:example:Thermostat:	

Figure 3.42 thermostat1 selection

12. Click the **Telemetry**.

Azure IoT Explorer (previe File Edit View Window		-	
Azure IoT Explorer	(preview)	Notifications	Settings
<u>Home</u> > azure-s <u>components</u> > t ≡		ommands Telemetry	_
Device identity	C) Refresh	0	🕑 Back
🔁 Device twin	You model definition has been resolved from: Local Folder 🛞 Config	gure	
C Telemetry	Interface Id		
S Direct method	dtmi:com:example:Thermostat:1		D

Figure 3.43 Telemetry selection

13. Click the Start.



Figure 3.44 Telemetry started



14. Confirm that the uploaded data is displayed in the **Receiving events** field \rightarrow Click **Stop**.

Azure IoT Explorer (pre	view) Q Notifications	💮 Settings
<u>Home</u> > azure-semi <u>components</u> > therr	nar-iothub-001 > <u>Devices</u> > azure-seminar-deviceid > <u>IoT Plug and Play</u> nostat1	
=	Interface Properties (read-only) Properties (writable) Commands Telemetry	
Device identity	Stop	🕝 Back
🔁 Device twin		
Telemetry	Telemetry 🛈	
✓ Direct method	Consumer group ① \$Default	
🗹 Cloud-to-device me	Specify enqueue 🕕	
Module identities	No No	
o [⊄] IoT Plug and Play co	Use built-in event hub Yes	
	① Receiving events 〇	
	Thu Jul 08 2021 11:20:02 GMT+0900 (Japan Standard Time):	Í
	{ "body": { "temperature": 22 }, "enqueuedTime": "Thu Jul 08 2021 11:20:02 GMT+0900 (Japan Standard Time)",	
	"properties": {} }	
	Thu Jul 08 2021 11:19:57 GMT+0900 (Japan Standard Time):	
	{ "body": { "temperature": 22 }	
	}, "enqueuedTime": "Thu Jul 08 2021 11:19:57 GMT+0900 (Japan Standard Time)",	

15. Click the **Stop** icon on the e^2 studio screen.



Figure 3.46 Demo program stop



4. LED ON/OFF operation by Azure IoT Explorer

Follow the steps below to turning on/off LED1 on the RX65N Cloud Kit from the Azure cloud.

To operate LED1, you need to make changes to the sample project downloaded from the Azure RTOS sample page on GitHub in 3.2 Software Preparation.Please refer to 8. Appendices for the changes.

1. Click the **device ID** created in Azure IoT Explorer \rightarrow Click **Device twin**.

File Edit View Window He			
Azure IoT Explorer (p	preview)	fications	Settings
Home > azure-se	minar-iothub-001 > <u>Devices</u> > azure-seminar-deviceid >	Device twi	in
=	💍 Refresh 📙 Save		
Device identity			
🔁 Device twin	Device twin ①		
Telemetry	<pre>1* { 2 "deviceId": "azure-seminar-deviceid", 3 "etag": "AAAAAAAAB8",</pre>		1
✓ Direct method	<pre>4 "deviceEtag": "NTEwNzkSODU4", 5 "status": "enabled", 6 "statusUpdateTime": "0001-01-01T00:00:00Z", 7 "connectionState": "Connected".</pre>		- 1
Cloud-to-device	<pre>7 "connectionState": "Connected", 8 "lastActivityTime": "2021-06-11707:09:16.31689472", 9 "cloudToDeviceMessageCount": 0, 10 "authenticationType": "sas",</pre>		- 1
Module identities	10 addmentedition/per ses, 11 - "x509Thumbprint": (12 "primaryThumbprint": null, 13 "secondaryThumbprint": null		
S loT Plug and Play	<pre>14 }, 15 "modelId": "dtmi:com:example:TemperatureController;1",</pre>		

Figure 4.1 Device twin screen

2. Add the following lines of code into **desired properties** \rightarrow Click **Save**.

"LED": { "LED1": 1 }, Figure 4.2 Message payload to turn on LED1



 Device identity Device twin Device twin Telemetry Direct method Cloud-to-device Module identities 	r-iothub-001 > <u>Devices</u> > azure-seminar-deviceid > Device twin	tings
 Device identity Device twin Telemetry Direct method Cloud-to-device Module identities 	r-iothub-001 > <u>Devices</u> > azure-seminar-deviceid > Device twin	
Device identity Device twin Telemetry Direct method Cloud-to-device Module identities		
Device identity Device twin Telemetry Direct method Cloud-to-device Module identities		
Device twin Telemetry Direct method Cloud-to-device Module identities) Refresh 🗟 Save	
Direct method Cloud-to-device Module identities	evice twin 🛈	
Cloud-to-device	<pre>1 - { 2 "deviceId": "azure-seminar-deviceid",</pre>	-
X Module identities	<pre>3 "etag": "AAAAAAAAB0=", 4 "deviceEtag": "NTEwNik5OOU4", 5 "status": "enabled",</pre>	I
S Module identities	<pre>6 "statusUpdateTime": "0001-01-01700:00:002", 7 "connectionState": "Connected", 8 "lastActivityTime": "2021-06-11T07:09:16.3168947Z", 9 "cloudToOviceNessageCount": 0.</pre>	
	<pre>9 "cloudToDeviceMessageCount": 0, 10 "authenticationType": "sas", 11+ "x509Thumbprint": { 2</pre>	
	13 "secondaryThumbprint": null 14 },	
p for high and hops.	<pre>15 "modelId": "dtmi:com:example:TemperatureController;1", 16 "version": 212,</pre>	
	17 - "properties": { 18 - "desired": {	
	19- "thermostat1": (20 "t": "C, t": "(21 "targetTemperature": 22	
	22 }, 23 - "LED": {	
	24 "LED1": 1 25 }.	
	26 - "Smetacata": (
	27 "\$lastUpdated": "2021-06-10T02:45:36.25003772", 28 "\$lastUpdatedVersion": 29,	
	29 * "thermostat1": { 30 "\$lastUpdated": "2021-06-10702:45:36.25083772",	
	31 "\$lastUpdatedVersion": 29,	
	32 + "t": (33 "\$lastUpdated": "2021-06-10T02:45:36.25083772",	
	34 "\$lastUpdatedVersion": 29	
	35 }, 36 - "targetTemperature": { 37 "\$lastUpdated": "2021-06-10702:45:36.25083772",	

Figure 4.3 Device twin setting state

3. Confirm that RX65N Cloud Kit LED1 turn on.



Figure 4.4 LED1 turn on



4. Add the following lines of code into **desired properties** \rightarrow Click **Save**.

"LED": { "LED1": 0 },

Figure 4.5 Message payload to turn off LED1

5. Confirm that RX65N Cloud Kit LED1 turn off.



Figure 4.6 LED1 turn off



5. Visualization of Sensor Data

Follow the steps below to visualizing the sensor data* uploaded from the RX65N Cloud Kit to the Azure cloud using a web application.

Note: The demo program in this document uploads fixed values as sensor data.

5.1 Software Preparation

1. Go to the IoT Hub page you created on the Azure portal **Built-in endpoints** \rightarrow Enter arbitrary name to **Create new consumer group** \rightarrow Click **Save**.

😑 Microsoft Azure 🔎 s	earch resources, services, and docs (G+/) 🛛 🕼 🛛 🕼 🖓 🖓 🛞 🥍 MY DIRECTORY 🜷
Home > azure-seminar-iothub-001	
e azure-seminar-io	thub-001 Built-in endpoints 🖈 … 🛛 🗙
P Search (Ctrl+/)	« 🗟 Save り Undo
X Overview	1
Activity log	Bach IoT hub comes with built-in system endpoints to handle system and device messages. When you create new endpoints and routes, messages stop flowing
R Access control (IAM)	to the built-in endpoint unless you create a separate route and direct them there. Learn more
Tags	Event Hub Details
Diagnose and solve problems	
🗲 Events	Partitions D
Settings	
Shared access policies	Event Hub-compatible name ① iothub-ehublazure-semi-13259798-53385e18b7 ①
 Shared access policies Identity 	
	Retain for ①
O Pricing and scale	Days
Networking	Consumer Groups ①
🔎 Certificates	Consumer Groups
Built-in endpoints	SDefault
-* Failover	Create new consumer group
Se Properties	

Figure 5.1 Consumer group creation

2. Make a note of the **consumer group name** with a text editor.

The consumer group name information will be used later.

3. Make a note of the Shared access policies \rightarrow service \rightarrow Primary connection string with a text editor.

The Primary connection string information will be used later.

Home > azure-seminar-iothub-001	ub-001 Shared access pol	icies 🖈	service azure-seminar-iothub-001 ③ Regenerate primary key ③ Regenerate secondary key 11 Swap keys	
P Search (Ctrl+/) «	#b		Primary key	
X Overview	Shared access policies are used to generate se + Add shared access policy O Refresh	20 	enmary key	>
Activity log	T Add shared access policy O Remesh	D Delete	Secondary key	
R Access control (IAM)	Policy Name	Permissions		
Tags	a national sector and the sector of the		Primary connection string	
Diagnose and solve problems	iothubowner	Registry Read, F		Þ
🗲 Events	service	Service Connect	Secondary connection string	
Settings	device	Device Connect	<	>
Shared access policies	registryRead	Registry Read	Permissions	
🐒 Identity			Registry Read	
Pricing and scale	registryReadWrite	Registry Read, F	Registry Write	
6 Networking			Service Connect	
			Device Connect	

Figure 5.2 Shared access policies \rightarrow service \rightarrow Primary connection string



5.2 Running the web application

1. Start the command prompt and run the commands in the following order.

(1) Download the sample code of the web application for visualization from GitHub.

git clone https://github.com/Azure-Samples/web-apps-node-iot-hub-data-visualization.git

Figure 5.3 Web application sample code download

(2) Move the folder to the storage location of the sample code.

cd web-apps-node-iot-hub-data-visualization

Figure 5.4 Move sample code folder

(3) Set environment variables.

set lotHubConnectionString=[5.1 Software Preparation 3. Primary connection string]

set EventHubConsumerGroup=[5.1 Software Preparation 2. consumer group name]

Figure 5.5 Environment variable settings

(4) Launch website.

Enter Ctrl-C to exit the website.

npm install

npm start

Figure 5.6 Launch website

command Prompt	-		×
licrosoft Windows [Version 10.0.19041.867] c) 2020 Microsoft Corporation. All rights reserved.			^
:\myfolder>git clone https://github.com/Azure-Samples/web-apps-node-iot-hub-data-visualization.gi loning into 'web-apps-node-iot-hub-data-visualization' emote: Enumerating objects: 193, done. emote: Counting objects: 100% (13/13), done. emote: Compressing objects: 100% (13/13), done. emote: Total 193 (delta 6), reused 0 (delta 0), pack-reused 180 eceiving objects: 100% (13/193).es & KiB 1.69 MiB/s, done. esolving deltas: 100% (79/79), done.	t		
:\myfolder>cd web-apps-node-iot-hub-data-visualization			
:\myfolder\web-apps-node-iot-hub-data-visualization>set IotHubConnectionString=HostName=azure-sem levices.net;SharedAccessKeyName=service;SharedAccessKey=	inar-iothub-0	901.az	ure-
:\myfolder\web-apps-node-iot-hub-data-visualization>set EventHubConsumerGroup=azure_consumer_grou	p_test		
:\myfolder\web-apps-node-iot-hub-data-visualization>npm install dded 349 packages from 306 contributors and audited 352 packages in 10.488s			
packages are looking for funding run `npm fund` for details			
ound 6 vulnerabilities (3 moderate, 3 high) run `npm audit fix` to fix them, or `npm audit` for details			

Figure 5.7 Command prompt screen

2. Run the demo program according to the procedure in 3.4 Running the demo program.



3. Start your browser and go to http://localhost:3000 to see the temperature data graph.



Figure 5.8 Web application screen



5.3 Display of graph changes

The sample program allows you to change the value of the sensor data to be uploaded. The changed data can be confirmed in the graph.

1. Go to the **Device ID** you created on the Azure portal \rightarrow **IoT Plug and Play components** \rightarrow Scroll down and click **thermostat1** from Component.

Azure IoT Explorer (preview File Edit View Window H		- 🗆 X
Azure IoT Explorer (preview)	📮 Notifications 🛛 🎯 Settings
<u>Home</u> > azure-se components	eminar-iothub-001 > <u>Devices</u> > azure	-seminar-deviceid > IoT Plug and Play
=	🕐 Refresh	
 Device identity Device twin Telemetry 	LoT Plug and Play components	
✓ Direct method	Component	Model ID
🖾 Cloud-to-device	Default component	dtmi:com:example:Temperatur eController;1
Module identities 6 ³ IoT Plug and Pla	thermostat1	dtmi:com:example:Thermostat; 1
	thermostat2	dtmi:com:example:Thermostat:

Figure 5.9 Components screen

2. Click Properties (writable).

Azure IoT Explorer (previe File Edit View Window		-	
Azure IoT Explorer	(preview)	Notifications	Settings
<u>Home</u> > azure-s components > t	eminar-iothub-001 > <u>Devices</u> > azure-seminar-devicei hermostat1	id > <u>loT Plug</u>	and Play
Device identity	Interface Properties (read-only) Properties (writable) Comm	nands Telemetry	🕣 Back
Device twin	You model definition has been resolved from: Local Folder 🛞 Configure	2	
> Direct method	dtmi:com:example:Thermostat;1		D

Figure 5.10 Properties (writable) selection



3. Enter the temperature data in target Temperature \rightarrow Click Update desired value.



Figure 5.11 Enter the temperature data \rightarrow Update desired value

4. Go to http://localhost:3000 and check that the graph is changing.



Figure 5.12 Check the sensor data change on the web application screen

5. Click the **Stop** icon on the e^2 studio screen to end the demo run.



6. Important Note after Running Demo Program

Fees are incurred when are using the Azure Service. Make sure to delete your account if you will no longer be using it.

7. Websites and Support

Azure RTOS: <u>https://azure.microsoft.com/support/community/</u>

Azure RTOS GitHub: https://github.com/azure-rtos

8. Appendices

The changes required for the sample program to perform 4. LED ON/OFF operation by Azure IoT Explorer are shown below.

Note: The line numbers in the description indicate the number of lines in the sample program at the time of download.

1. Make the following changes to "{\$base_folder}/rx65n-cloud-

kit/e2studio_gnurx/sample_pnp_temperature_controller/src/sample_pnp_temperature_controller.c".

(1) Add the following variable declarations.

l.143	<pre>static const CHAR sample_device_info_component[] = "deviceInformation";</pre>		
	static SAMPLE_PNP_THERMOSTAT_COMPONENT sample_led;	7	
	<pre>static const CHAR sample_led_component[] = "LED";</pre>	Add	
	static double sample_led_last_device_reported;		

Figure 8.1 Changed sample_pnp_temperature_controller.c (1)

(2) Add elements for LEDs to the sample_components array.

l.146	static const CHAR *sample_components[] = { sample_thermostat_1_component,
l.147	sample_thermostat_2_component,
	sample_led_component, Add
l.148	<pre>sample_device_info_component };</pre>

Figure 8.2 Changed sample_pnp_temperature_controller.c (2)



	0
if (component_name_ptr == NULL component_name_len == 0)	
(昭各)	
}	
else if (sample_pnp_led_process_property_update(&sample_led,	
(NX_AZURE_IOT_HUB_CLIENT *)userContextCallback,	
component_name_ptr, component_name_len,	
property_name_ptr, property_name_len,	
&property_value_reader, version) == NX_AZURE_IOT_SUCCESS)	
printf("property updated of led1¥r¥n");	
else	Add
{	Auu
}	
	{ (略) else if (sample_pnp_led_process_property_update(&sample_led, (NX_AZURE_IOT_HUB_CLIENT *)userContextCallback, component_name_ptr, component_name_len, property_name_ptr, property_name_len, &property_value_reader, version) == NX_AZURE_IOT_SUCCESS) printf("property updated of led1¥r¥n"); else {

(3) Add a conditional expression to the following if statement of the sample_desired_property_callback() function



(4) Add conditional expression to the following if statement of sample_components_init() function.



Figure 8.4 Changed sample_pnp_temperature_controller.c (4)



 Make the following changes to "{\$base_folder}/rx65n-cloud- kit/e2studio_gnurx/sample_pnp_temperature_controller/src/sample_pnp_thermostat_component.c". 					
1) Add include settings in platform.h file.					
#include "sample_pnp_thermostat_component.h"					
#include "platform.h" Add					
Figure 8.5 Changed sample_pnp_thermostat_component.c (1)					
2) Add a constant for the data key name LED1 that communicates with Azure.					
I.36 static const CHAR temp_response_description_failed[] = "failed";					
<pre>static const CHAR led_property_name[] = "LED1"; Add</pre>					
Figure 8.6 Changed sample_pnp_thermostat_component.c (2)					
3) Add 2 functions used for Device twin operation.					
Last static VOID sample_send_led_report(SAMPLE_PNP_THERMOSTAT_COMPONENT *handle, line NX_AZURE_IOT_HUB_CLIENT *iothub_client_ptr, double temp, INT status_code, UINT version, const CHAR *description)					
{ UINT bytes_copied; UINT response_status; UINT request_id; NX_AZURE_IOT_JSON_WRITER json_writer; ULONG reported_property_version; /* Build telemetry JSON payload */ if (nx_azure_iot_json_writer_with_buffer_init(&json_writer, scratch_buffer, sizeof(scratch_buffer)))					
<pre>{ printf("Failed to create json writer¥r¥n"); return; }</pre>					
if (nx_azure_iot_pnp_helper_build_reported_property_with_status(
<pre>printf("Failed to create reported response¥r¥n"); } else { bytes_copied = nx_azure_iot_json_writer_get_bytes_used(&json_writer); if (nx_azure_iot_hub_client_device_twin_reported_properties_send(iothub_client_ptr,</pre>					
&request_id, &response_status, &reported_property_version, (5 * NX_IP_PERIODIC_RATE))) { printf("Failed to send reported response¥r¥n");					
}					
} nx_azure_iot_json_writer_deinit(&json_writer); }					

Figure 8.7 Changed sample_pnp_thermostat_component.c (3)-1



Visualization of Sensor Data using

RX65N Cloud Kit and Azure RTOS

```
UINT sample_pnp_led_process_property_update(SAMPLE_PNP_THERMOSTAT_COMPONENT *handle,
NX_AZURE_IOT_HUB_CLIENT *iothub_client_ptr,
Last
line
                                               CHAR *component_name_ptr, UINT component_name_length,
                                               UCHAR *property_name_ptr, UINT property_name_length,
                                              NX_AZURE_IOT_JSON_READER *property_value_reader_ptr, UINT version)
           {
               double parsed_value = 0;
               INT status code;
               const CHAR *description;
               if (handle == NX_NULL)
               {
                   return(NX_NOT_SUCCESSFUL);
               }
               if (handle -> component_name_length != component_name_length ||
                 strncmp((CHAR *)handle -> component_name_ptr,
                         (CHAR *)component_name_ptr, component_name_length) != 0)
               {
                   return(NX_NOT_SUCCESSFUL);
               }
               if (property_name_length != (sizeof(led_property_name) - 1) ||
                   strncmp((CHAR *)property_name_ptr, (CHAR *)led_property_name, property_name_length) != 0)
               {
                   printf("PnP property=%.*s is not supported on thermostat component¥r¥n",
                          property_name_length, property_name_ptr);
                   status_code = 404;
                   description = temp_response_description_failed;
               }
               else if (nx_azure_iot_json_reader_token_double_get(property_value_reader_ptr, &parsed_value))
               {
                   status_code = 401;
                   description = temp_response_description_failed;
               }
               else
               {
                   status_code = 200;
                   description = temp_response_description_success;
                   if (parsed_value == 1) // LED ON
                   {
                       PORTB.PDR.BIT.B0 = 1;
                       PORTB.PODR.BIT.B0 = 0;
                   }
                   else
                           // LED OFF
                   {
                       PORTB.PDR.BIT.B0 = 1:
                       PORTB.PODR.BIT.B0 = 1;
                   }
               }
               sample_send_led_report(handle, iothub_client_ptr, parsed_value,
                                       status_code, version, description);
               return(NX_AZURE_IOT_SUCCESS);
                    Figure 8.8 Changed sample_pnp_thermostat_component.c (3)-2
```



Visualization of Sensor Data using RX65N Cloud Kit and Azure RTOS

3. Make the following changes to "{\$base_folder}/rx65n-cloud-

	0	0	· · —	,	
kit/a2atudia	anury/aampla	nnn	tomporatura	controllar/ara/compla ppp	_thermostat_component.h".
KII/ezstualo	griuix/sample	prip	lemperature	controller/src/sample prip	

(1) Add t	he declaration of the sample_pnp_led_process_property_update() function.
1.65	UINT sample_pnp_thermostat_process_property_update(SAMPLE_PNP_THERMOSTAT_COMPONENT *handle,
I.66	NX_AZURE_IOT_HUB_CLIENT *iothub_client_ptr,
l.67	UCHAR *component_name_ptr, UINT component_name_length,
1.68	UCHAR *property_name_ptr, UINT property_name_length,
I.69	NX_AZURE_IOT_JSON_READER *property_value_reader_ptr, UINT version);
	UINT sample_pnp_led_process_property_update(SAMPLE_PNP_THERMOSTAT_COMPONENT *handle,
	NX_AZURE_IOT_HUB_CLIENT *iothub_client_ptr,
	UCHAR *component_name_ptr, UINT component_name_length,

UCHAR *property_name_ptr, UINT property_name_length,

NX_AZURE_IOT_JSON_READER *property_value_reader_ptr, UINT version);

Add

Figure 8.9 Changed sample_pnp_thermostat_component.h (1)



Revision History

		Description			
Rev.	Date	Page	Summary		
1.00	XX.XX.21	-	First edition		



General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

6

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).

7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

- Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
- 2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
- 3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
- 5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
- Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
 - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

- 7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
- 8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
- 9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
- 10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
- 12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
- This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
 Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas
- Electronics products. (Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries
- (Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.