

Bluetooth Low Energy Smartphone Application Example

TryBT for iOS

Overview

TryBT is an iOS sample application that can communicate with the evaluation boards for RX23W, RA4W1, or RE01B which are Renesas Electronics' MCU by Bluetooth® Low Energy wireless technology. This app is distributed as a sample project including source code, so users can customize and reuse the source code.

This document will explain how to create a development environment as well as how to perform basic customization of TryBT.

Target Devices

- iOS device (iOS 13.0 or later)

Related Documents

- RX23W Group Target Board for RX23W Quick Start Guide (R20QS0014)
- RX23W Group Target Board for RX23W module Quick Start Guide (R20QS0022)
- RA4W1 Group Evaluation Kit for RA4W1 EK-RA4W1 Quick Start Guide (R20QS0015)
- RE01B Group Bluetooth Low Energy Sample code (using CMSIS Driver Package) (R01AN5606)
- Renesas Flash Programmer V3.08 Flash memory programming software User's Manual (R20UT4813)

The *Bluetooth*® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Renesas Electronics Corporation is under license. Other trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1.	Overview.....	4
1.1	Operational Environment.....	4
1.2	NOTES	6
2.	Environment Configuration	7
2.1	Installing Xcode	7
2.2	Installing Homebrew.....	7
2.3	Installing latest version of Ruby.....	9
2.4	Installing CocoaPods.....	10
2.5	Opening a TryBT Project.....	10
2.6	Setting up an Apple Developer account.....	11
2.7	Creating a Certificate Signing Request	13
2.8	Creating and Installing Developer Certificate.....	14
2.9	Creating App ID for TryBT	17
2.10	Confirming UDID of iOS Device	20
2.11	Registering iOS device.....	21
2.12	Creating a TryBT Profile	22
2.13	Associating Profile with TryBT Project	26
2.14	Compiling and Installing TryBT	27
3.	Writing Firmware to Evaluation board	28
4.	Basic Operation of TryBT	31
4.1	Device List Screen	31
4.2	Connected Device Detail Screen	34
4.3	Light Demo Screen.....	37
4.4	Data Demo Screen.....	39
5.	Customizing TryBT	42
5.1	Customizing Application Title	42
5.2	Customizing Application Icons	42
5.3	Customizing Splash Screen	43
5.4	Customizing Icon Data on the Demo Screen	43
5.5	Enabling/Disabling Customization Mode.....	44
6.	File Composition of TryBT	45
6.1	Info.plist	46
6.2	.swift	46
7.	Screen Transition of TryBT	47

8.	Bluetooth Communication of TryBT.....	48
8.1	Performing a scan with Central Manager.....	48
8.2	Connecting to peripheral device.....	49
8.3	Service Discovery.....	50
8.4	Entering Values on the Light Demo Screen.....	52
8.5	Standby for Notifications on the Data Demo Screen.....	52
	Revision History	54

1. Overview

TryBT works on iOS devices running iOS 13.0 or later. This app connects to the Target Board for RX23W and displays a sample screen to perform demonstration. Also, source code of this app is distributed as an iOS project that can be modified.

1.1 Operational Environment

- iOS device (iOS 13.0 or later)
- Macintosh PC
- Any of the evaluation boards below:
 - [Target Board for RX23W](#) or [Target Board for RX23W module](#) ^{NOTE1}
 - [EK-RA4W1](#) ^{NOTE2}
 - [EB-RE01B](#) ^{NOTE3}

NOTE1 A firmware that can communicate with TryBT is written to Target Board for RX23W at the factory. If you write a firmware for communicating with TryBT to the board again, write the prebuilt firmware included in the quick start guide below to the Target Board for RX23W.

RX23W Group Target Board for RX23W Quick Start Guide ([R20QS0014](#))
→ble_demo_tbrx23w_profile_server_preinstall_yyyymmdd.mot file in mot folder
RX23W Group Target Board for RX23W module Quick Start Guide ([R20QS0022](#))
→ble_demo_mtbrx23w_profile_server_preinstall_yyyymmdd.mot file in mot folder

NOTE2 A firmware that can communicate with TryBT is written to EK-RA4W1 at the factory. If you write a firmware for communicating with TryBT to the board again, write the prebuilt firmware included in the quick start guide below to the EK-RA4W1.

RA4W1 Group Evaluation Kit for RA4W1 EK-RA4W1 Quick Start Guide ([R20QS0015](#))
→Restore_Factory/r20qs0015.srec included in bin.zip

NOTE3 A firmware that can communicate with TryBT is written to EB-RE01B at the factory. If you write a firmware for communicating with TryBT to the board again, write the firmware included in the document below to the EB-RE01B.

RE01B Group Bluetooth Low Energy Sample code (using CMSIS Driver Package) ([R01AN5606](#))
→ble_project_server.hex in ROM_Files folder

Hereinafter, this document will describe operations using Target Board for RX23W. These operations are same as when either EK-RA4W1 or EB-RE01B is used.

Hardware required for checking TryBT operation:

- iOS device: iOS 13.0 or later
- Macintosh PC
- Target Board for RX23W

iOS device tested:

- iPhone SE 2nd Generation (iOS 14.7)

Software required for checking TryBT operation:

- MacOS 10.14.4 Mojave or later
- Xcode 11.0 or later (refer to Section 2.1 for more information)

TryBT Project:

- Implementation Language: Swift ((<https://www.apple.com/jp/swift/>))

Before Using:

To install and check the operation of the TryBT project on iOS device, you need to become a paid member of the Apple Developer Program.

<https://developer.apple.com/programs/>

1.2 NOTES

- This document was created based on the status on 30th Sept 2021. It is not guaranteed that information described in this document supports all of the future versions of software and tools provided by our company or third parties.
- Renesas Electronics disclaims any and all liability arising from the use of information in this document and related software. Please also refer to the "Notice" in the last page of this document.

2. Environment Configuration

2.1 Installing Xcode

Install Xcode, Integrated Development Environment.

1. Open App Store and type "Xcode" in the app search field.
2. Select "Install".

2.2 Installing Homebrew

Homebrew is a package manager for installing and managing various libraries within the Mac OS environment. Install it to build your development environment.

1. Open the following URL: <https://brew.sh/>
2. Select the icon next to the shell script command shown on the screen to copy the command.



Figure 2.1 Installing Homebrew (1)

3. Open Finder, go to [Applications]→[Utilities], and run Terminal.

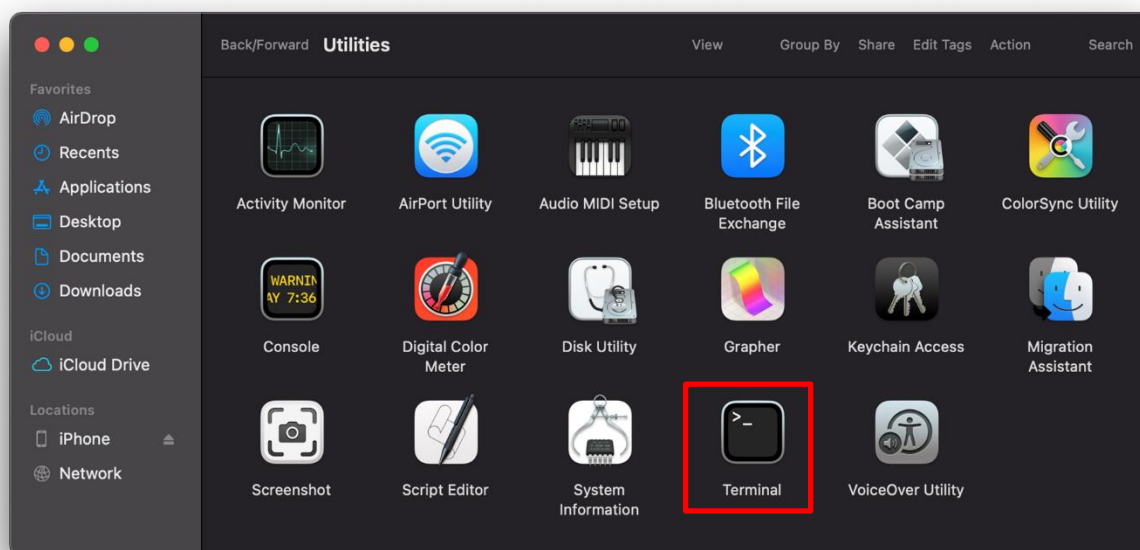
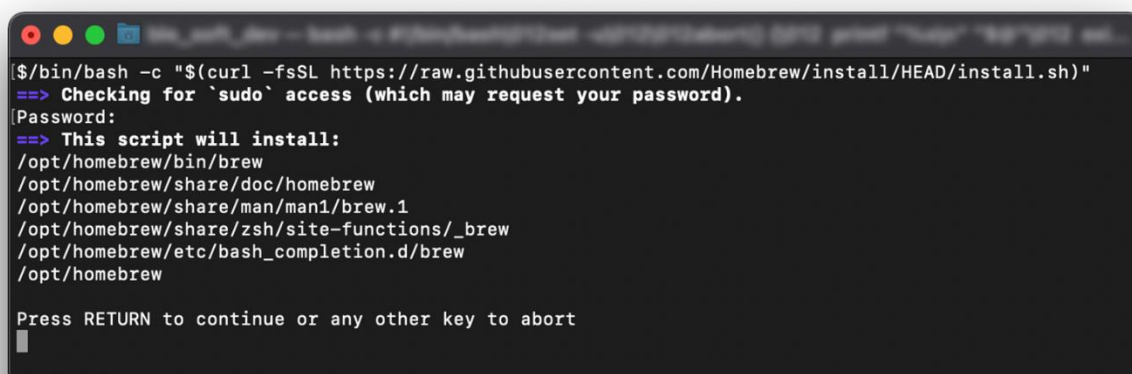


Figure 2.2 Installing Homebrew (2)

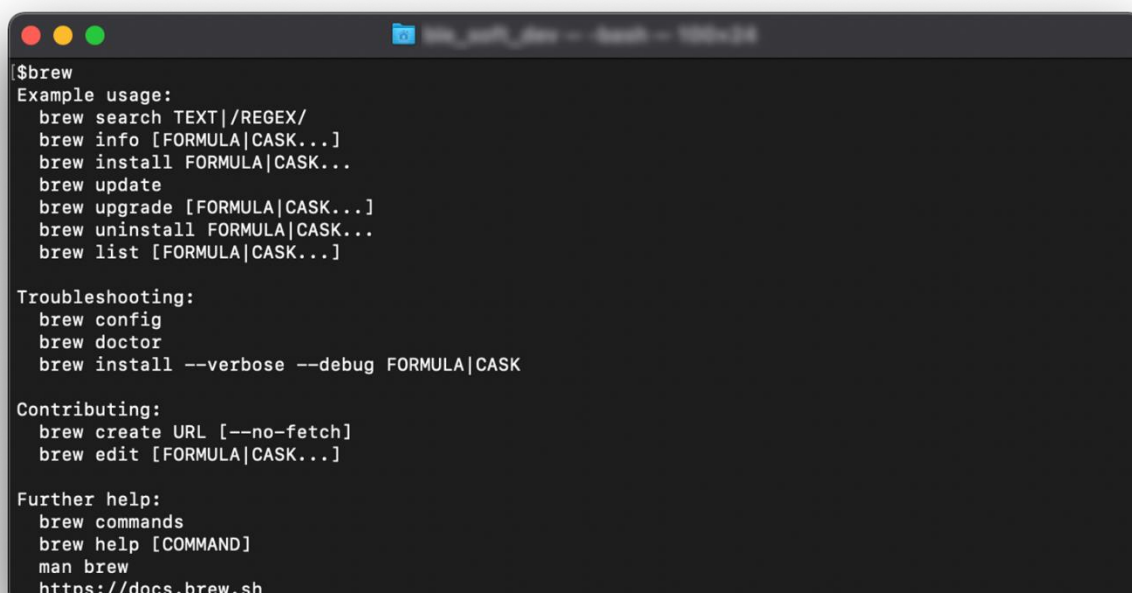
4. Paste the command copied in step 2. into the Terminal and then execute it.



```
[~/work_dev - Bash - 100x24]
[$/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)" ]
==> Checking for `sudo` access (which may request your password).
[Password: ]
==> This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew
Press RETURN to continue or any other key to abort
█
```

Figure 2.3 Installing Homebrew (3)

5. Type "brew" in the Terminal and confirm the tips are displayed.



```
[~/work_dev - Bash - 100x24]
[$brew]
Example usage:
  brew search TEXT|/REGEX/
  brew info [FORMULA|CASK...]
  brew install FORMULA|CASK...
  brew update
  brew upgrade [FORMULA|CASK...]
  brew uninstall FORMULA|CASK...
  brew list [FORMULA|CASK...]

Troubleshooting:
  brew config
  brew doctor
  brew install --verbose --debug FORMULA|CASK

Contributing:
  brew create URL [--no-fetch]
  brew edit [FORMULA|CASK...]

Further help:
  brew commands
  brew help [COMMAND]
  man brew
  https://docs.brew.sh
```

Figure 2.4 Installing Homebrew (4)

2.3 Installing latest version of Ruby

Use Homebrew to update Ruby to the latest version, then install CocoaPods to manage Xcode's libraries.

1. Open Finder, go to Applications→Utilities, and run Terminal
2. Execute the following command : `brew install ruby-build`
3. Execute the following command : `brew install rbenv`
4. To add Ruby configuration to the PATH, execute the following commands:

```
cd
echo 'export PATH="$HOME/.rbenv/shims:$PATH"' >> .bash_profile
echo 'eval "$({rbenv init -})"' >> .bash_profile
source .bash_profile
```

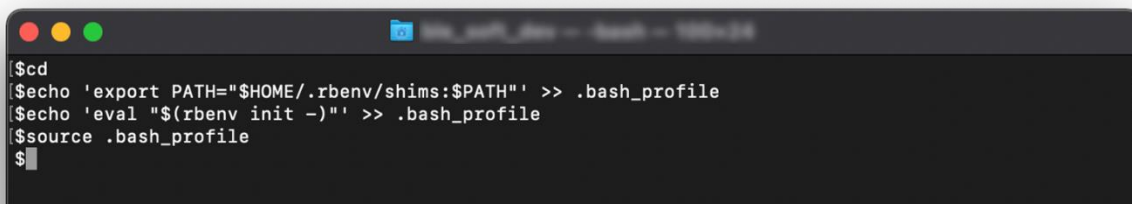


Figure 2.5 Setting for Ruby

NOTE: `/bin/bash` is used in the above as a shell. For how to change default shell, refer to the URL below:

<https://support.apple.com/guide/terminal/trml113/mac>

5. Run `ruby -v` to make sure your ruby version is 2.2.2 or later. If you are running former version, install the latest one with the following commands and reboot the terminal.
`rbenv install -l`, to display a list of installable versions
`rbenv install x.x.x`, to install the specified version
`rbenv global x.x.x`, to switch to the specified version

2.4 Installing CocoaPods

CocoaPods is a tool to manage third-party libraries for iOS apps. TryBT project uses CocoaPods to manage its libraries. Access the URL below and to install it:

<https://guides.cocoapods.org/using/getting-started.html>

2.5 Opening a TryBT Project

Open a TryBT project in Xcode.

1. Unzip the TryBT zip file attached to this document.
2. Move the unzipped folder to any folder.
3. Start Terminal and go to the above folder with `cd` command.
4. Execute the following command: `pod install`

The library used by the TryBT project will be installed automatically:

If the error below occurs by the `pod install` command, select [Xcode]→[Preferences...] in Xcode menu and configure Command Line Tools in Location tab.

xcode-select: error: tool 'xcodebuild' requires Xcode, but active developer directory '/Library/Developer/CommandLineTools' is a command line tools instance

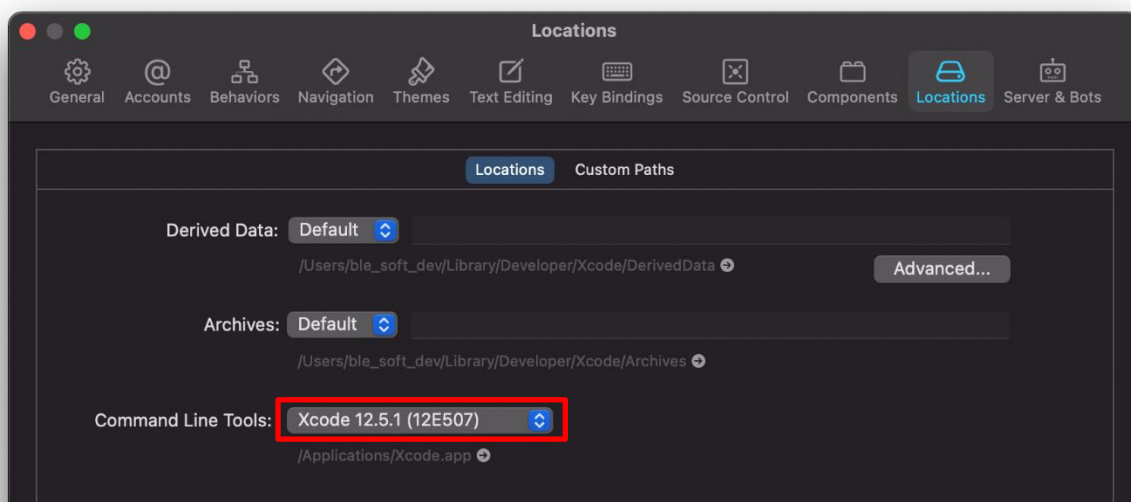


Figure 2.6 Configuring Command Line Tools

5. TryBT.xcworkspace will be created in the same folder. Double-click TryBT.xcworkspace to open the TryBT project.

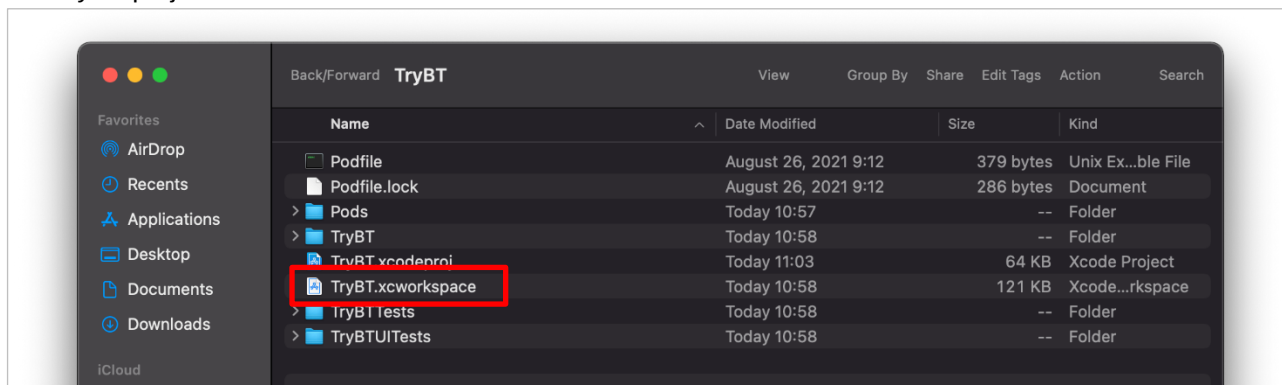


Figure 2.7 Opening TryBT Project

2.6 Setting up an Apple Developer account

To install TryBT on your device, you will need to set up a subscribed Apple Developer account in XCode.

1. Select [Xcode]→[Preferences] in Xcode menu.
2. Click the [+] button at the bottom left of Accounts tab and add your Apple Developer account.

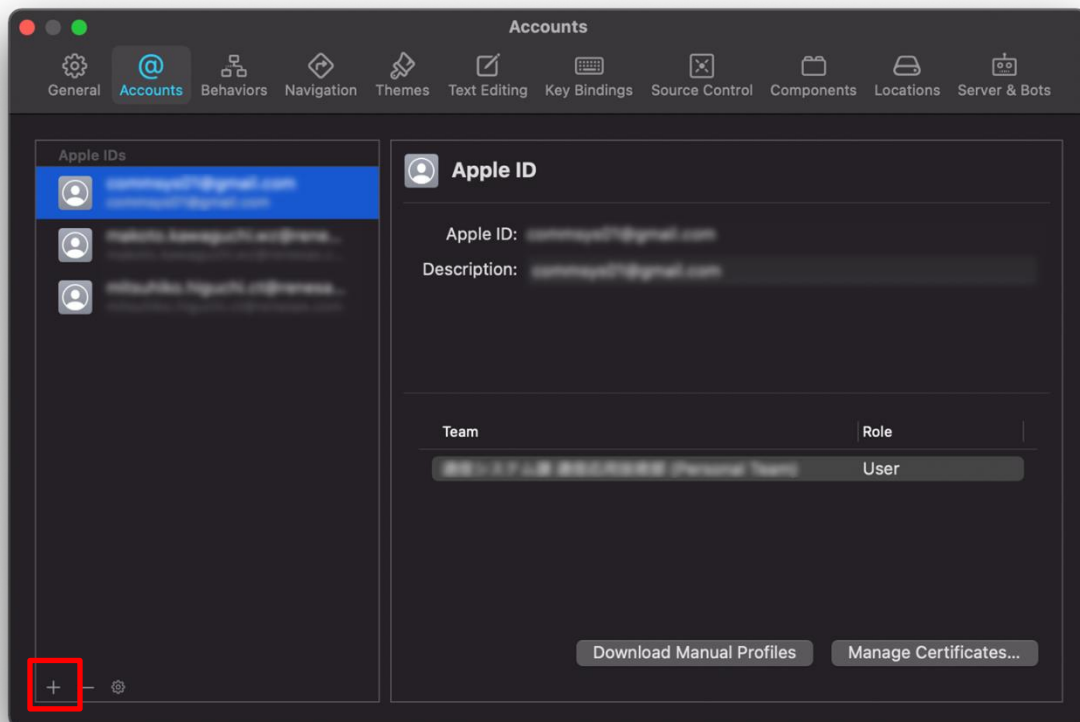


Figure 2.8 Setting-up an Apple Developer Account (1)

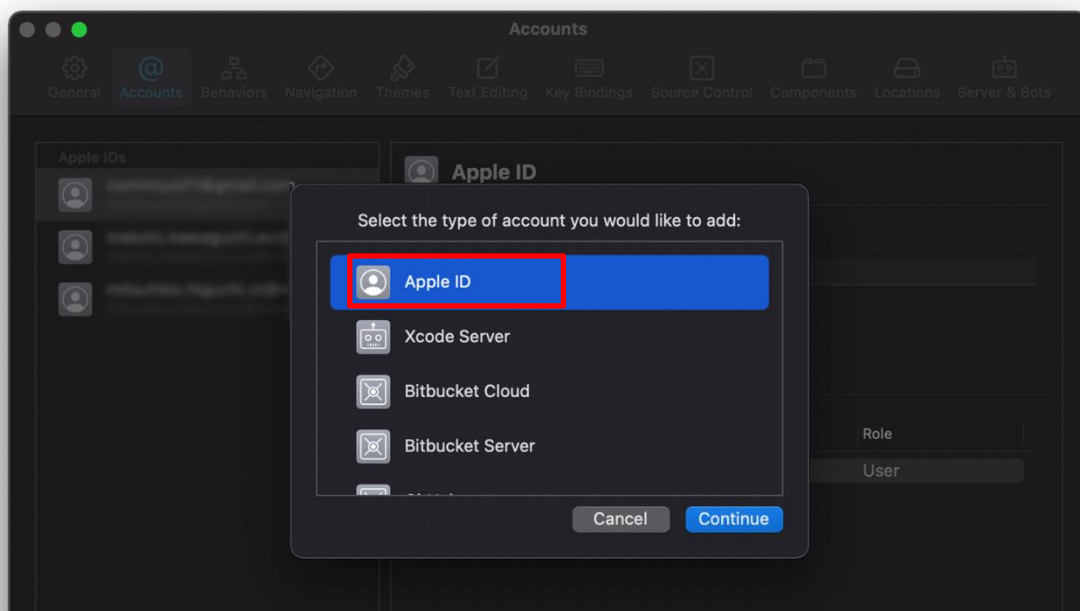


Figure 2.9 Setting-up an Apple Developer Account (2)

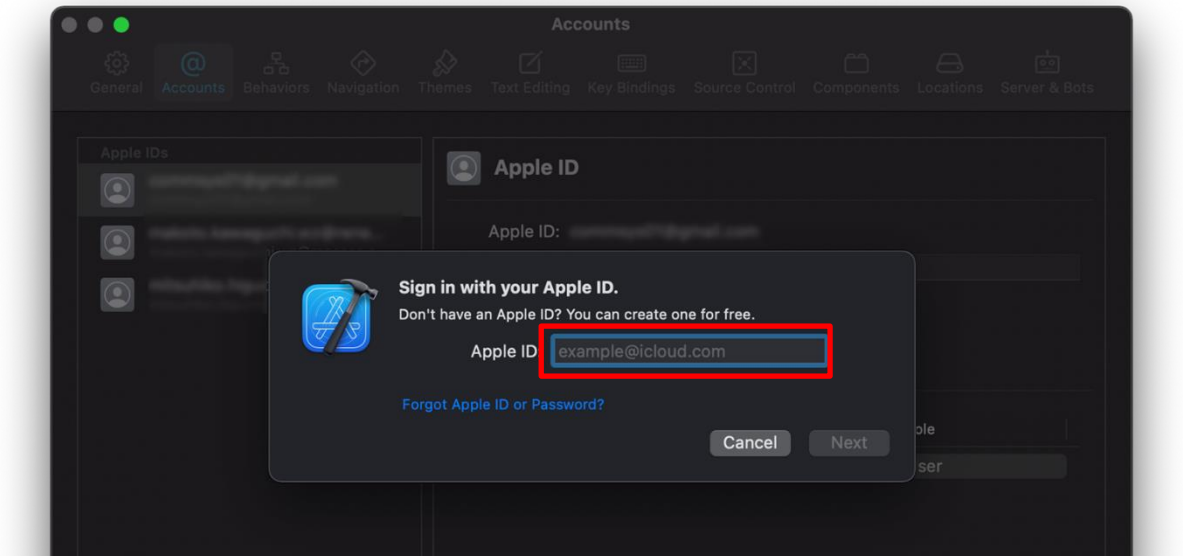


Figure 2.10 Setting-up an Apple Developer Account (3)

2.7 Creating a Certificate Signing Request

Create the Certificate Signing Request file required for installing to iOS device.

1. Launch Finder, go to [Applications]→[Utilities], and open Keychain Access.

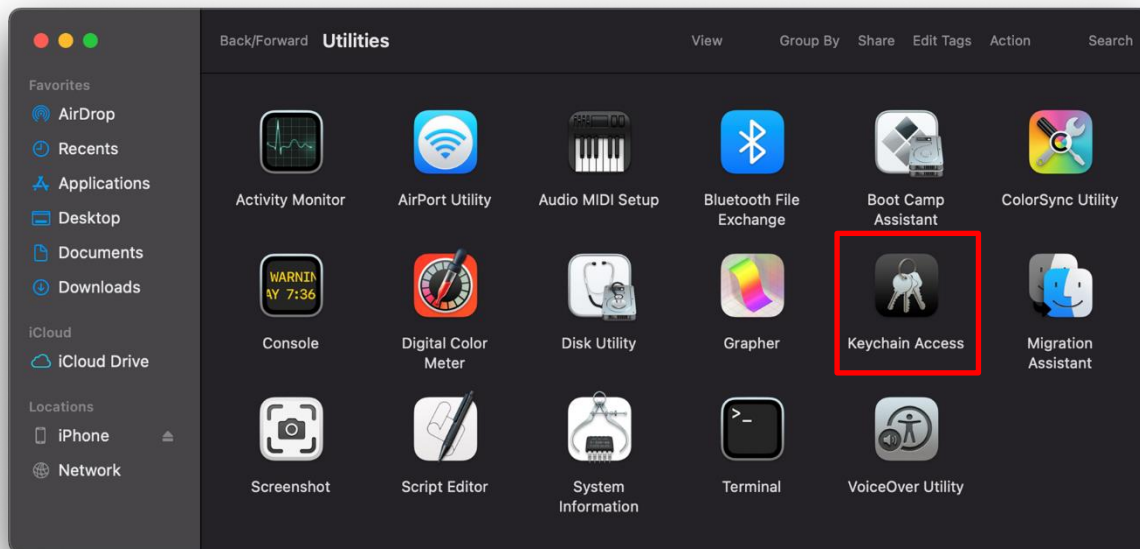


Figure 2.11 Creating a Certificate Signing Request (1)

2. Select [Keychain Access]→[Certificate Assistant]→[Request a Certificate From a Certificate Authority] in Keychain Access menu.
3. Enter your Apple Developer Program email address in the User Email Address field and enter your name in the Common Name field. Select "Save to disk". Click Continue button to save the Certificate Signing Request.

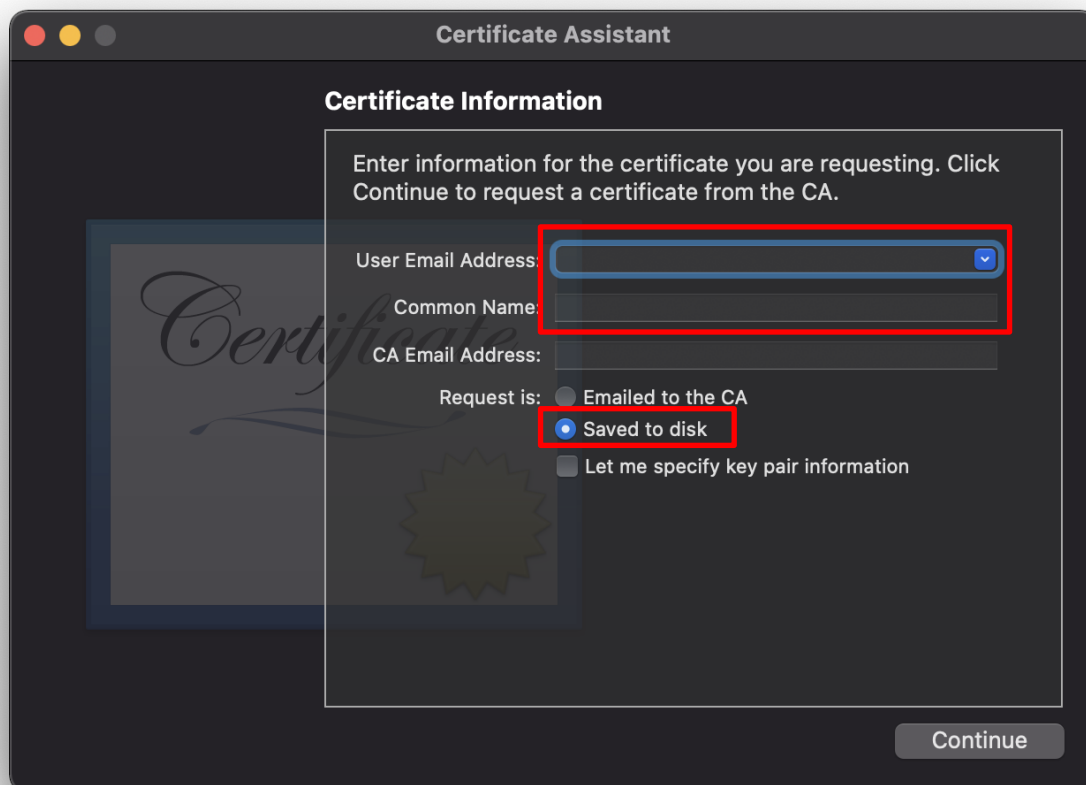


Figure 2.12 Creating a Certificate Signing Request (2)

2.8 Creating and Installing Developer Certificate

Create and install a Developer's Certificate, which is required to install TryBT on your device.

1. Sign-in to the Apple Developer and select "Certificates, IDs & Profiles" from the left side menu.

Apple Developer

<https://developer.apple.com/>

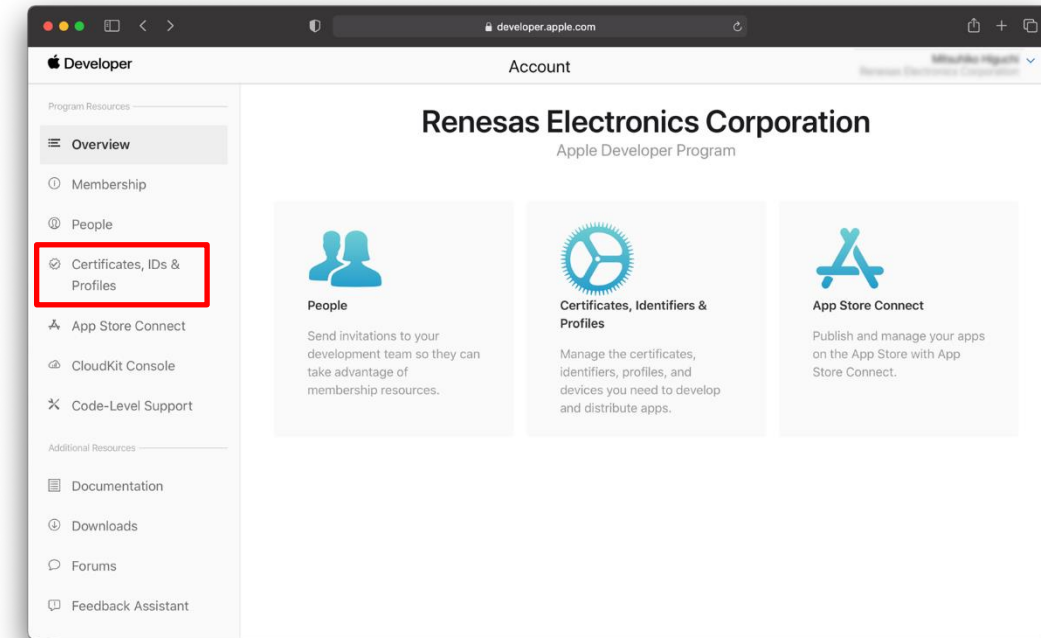


Figure 2.13 Creating Developer Certificate (1)

2. Select [Certificates] from the left menu and click blue [+] button next to Certificate.

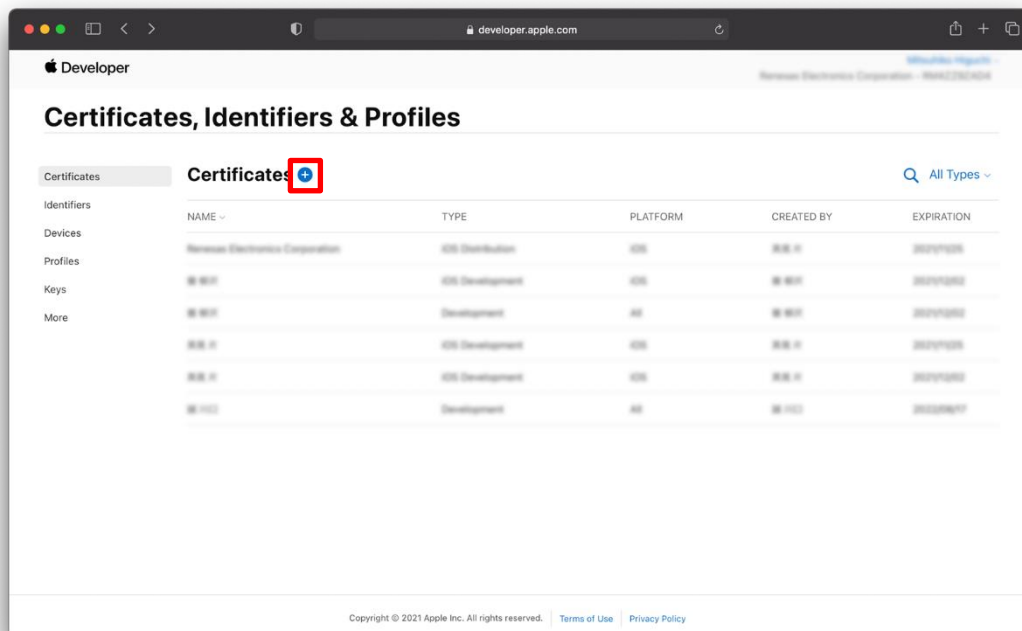


Figure 2.14 Creating Developer Certificate (2)

3. Select "Apple Development" and click Continue button.

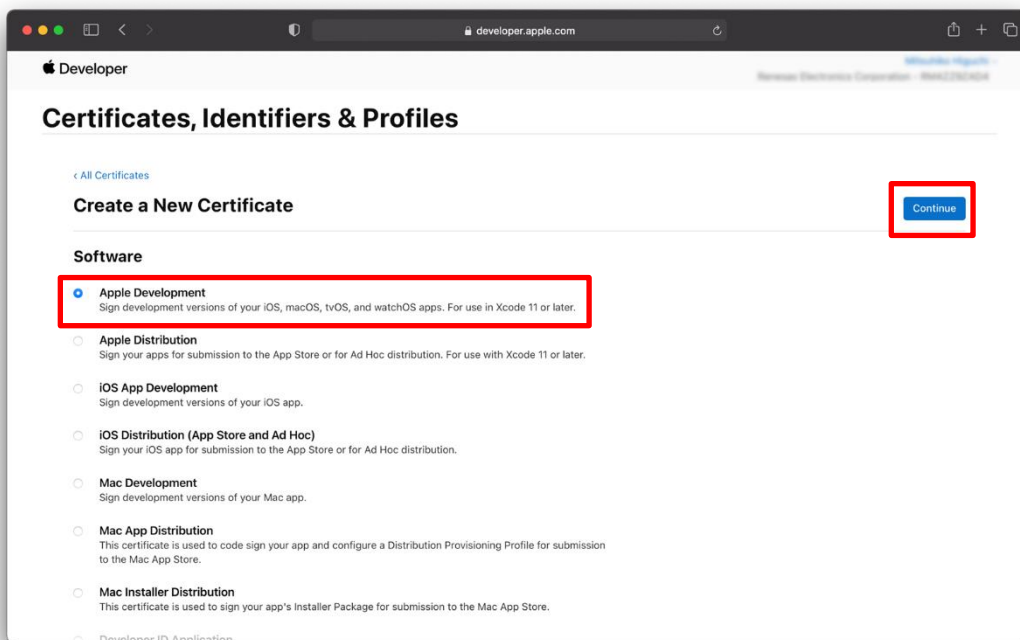


Figure 2.15 Creating Developer Certificate (3)

4. Select the Certificate Signing Request that you have already created on Choose File and then click Continue button.

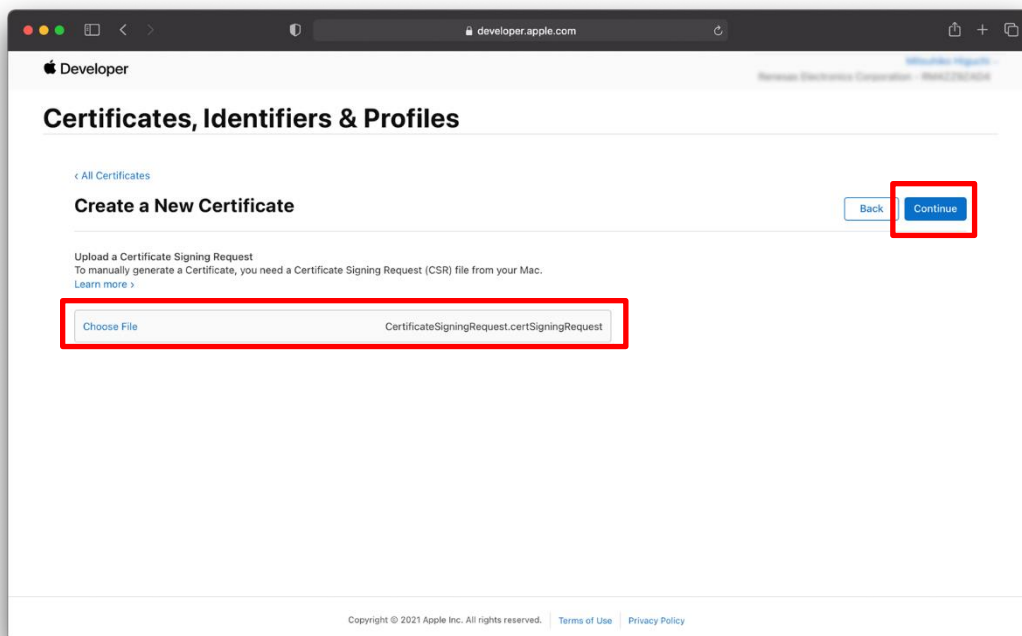


Figure 2.16 Creating Developer Certificate (4)

5. The Certificate has now been created. Click Download button to download Certificate. Double-click the downloaded Certificate to install it.

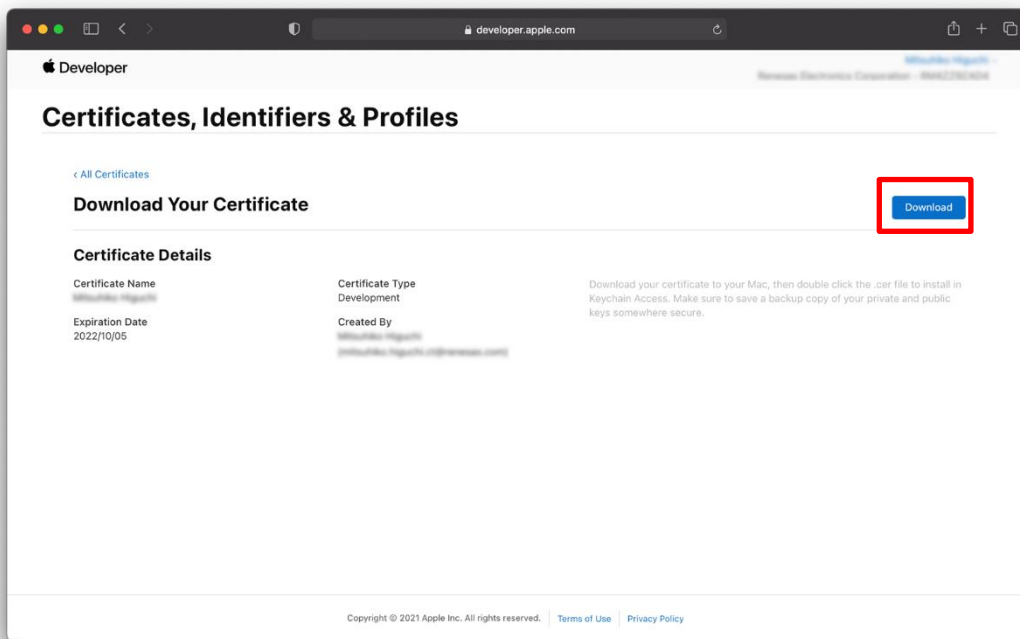


Figure 2.17 Creating Developer Certificate (5)

2.9 Creating App ID for TryBT

Create an App ID of TryBT. App ID is unique across whole iOS applications.

1. Select [Identifiers] from the left menu and click the blue [+] button next to Identifiers.

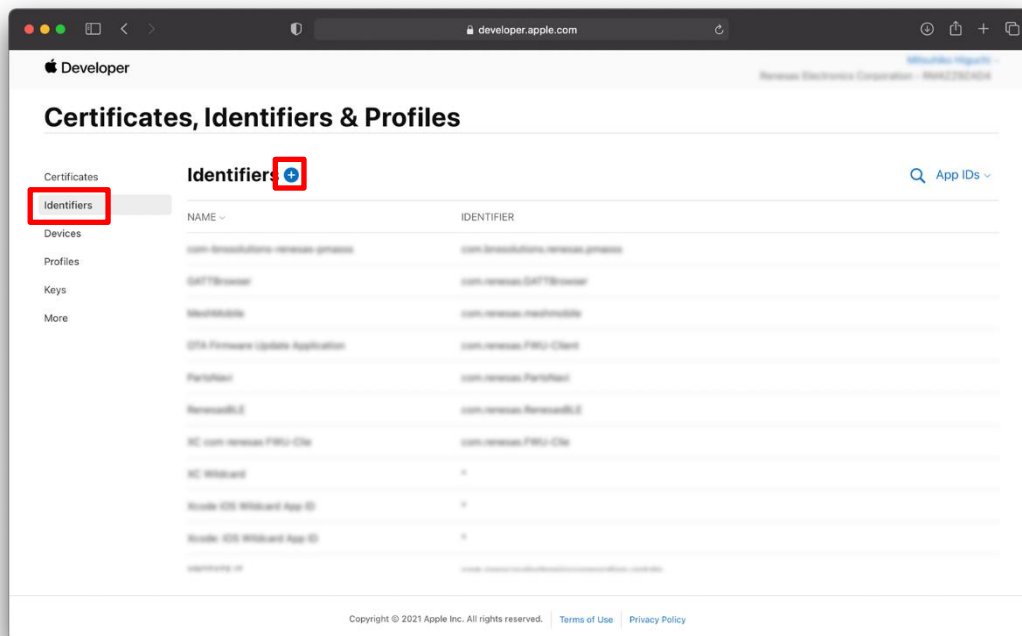


Figure 2.18 Creating App ID for TryBT (1)

2. Select "App IDs" and click Continue button.

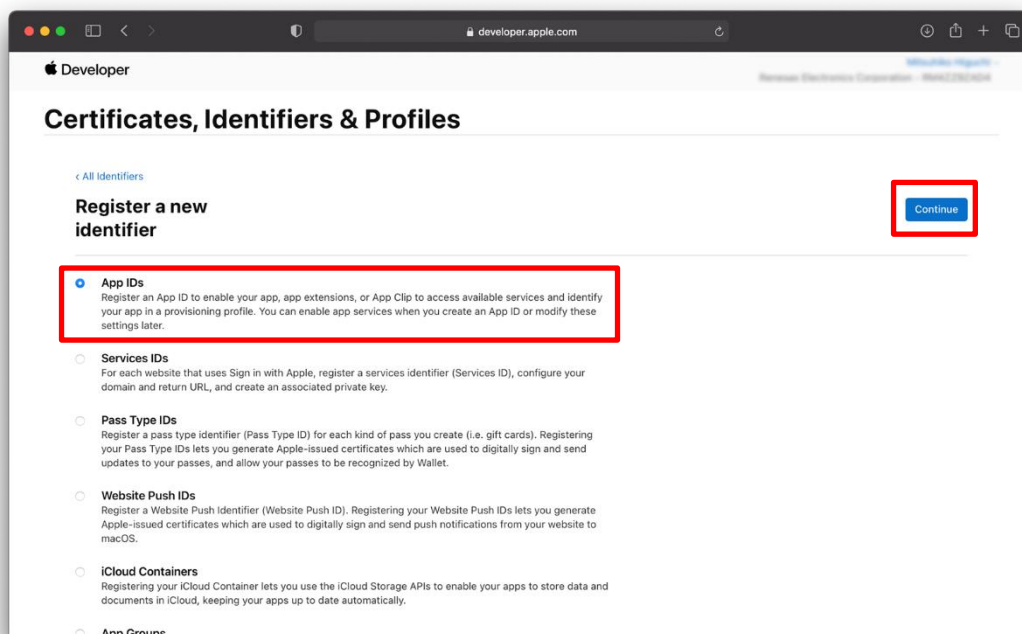


Figure 2.19 Creating App ID for TryBT (2)

3. Select "App" and click Continue button.

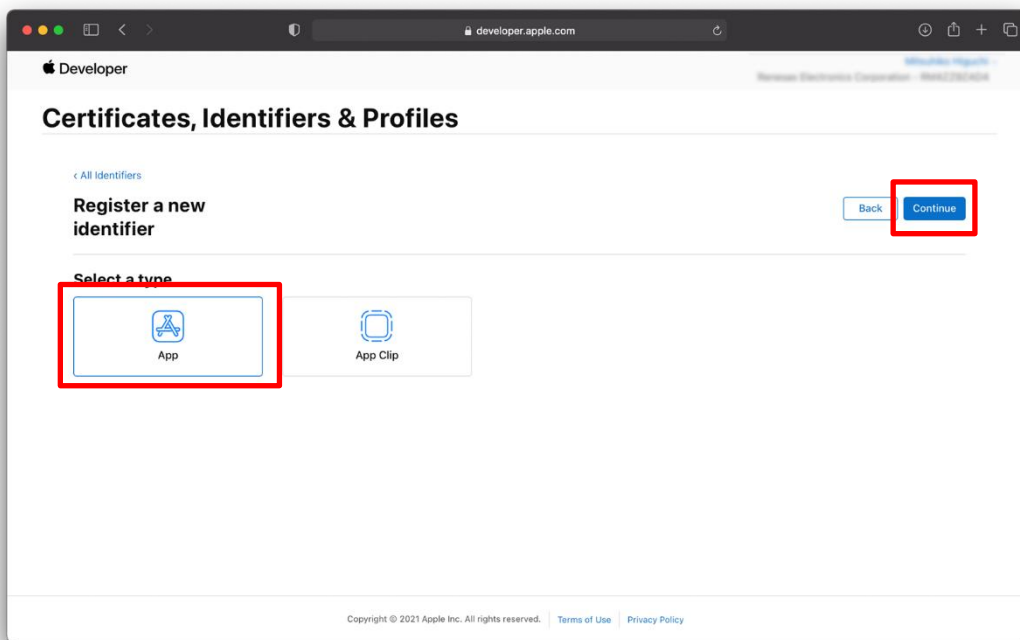


Figure 2.20 Creating App ID for TryBT (3)

4. Create an App ID. Type "TryBT Development" in the Description field. Leave App ID Prefix as it is. Type a reverse lookup of the company's domain followed by ".TryBT" in the Bundle ID field.

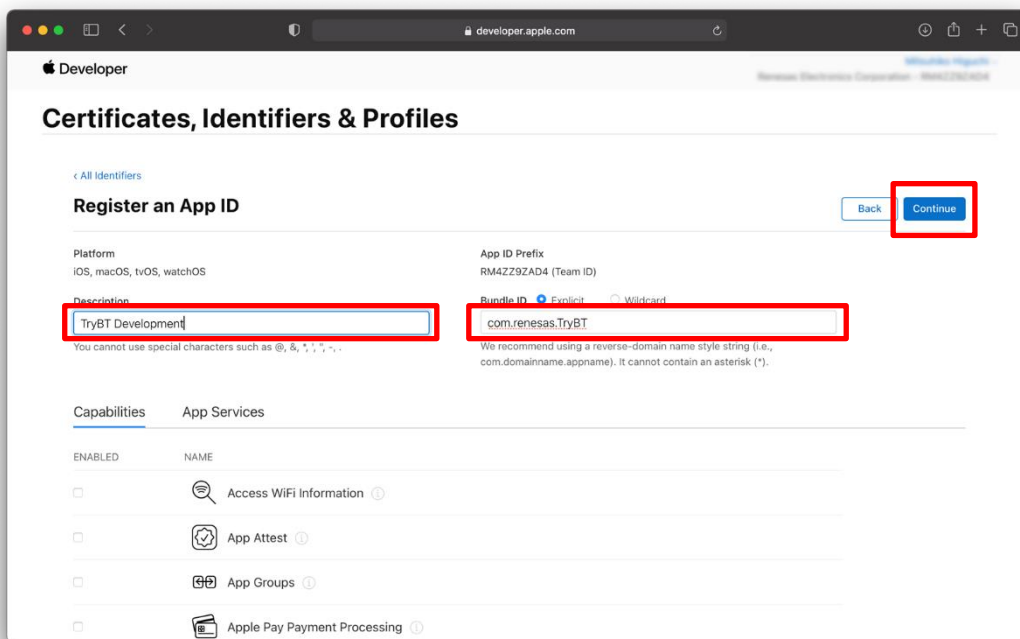


Figure 2.21 Creating App ID for TryBT (4)

5. Scroll down the screen and check Push Notifications, then click Continue button. Then, click Register button.

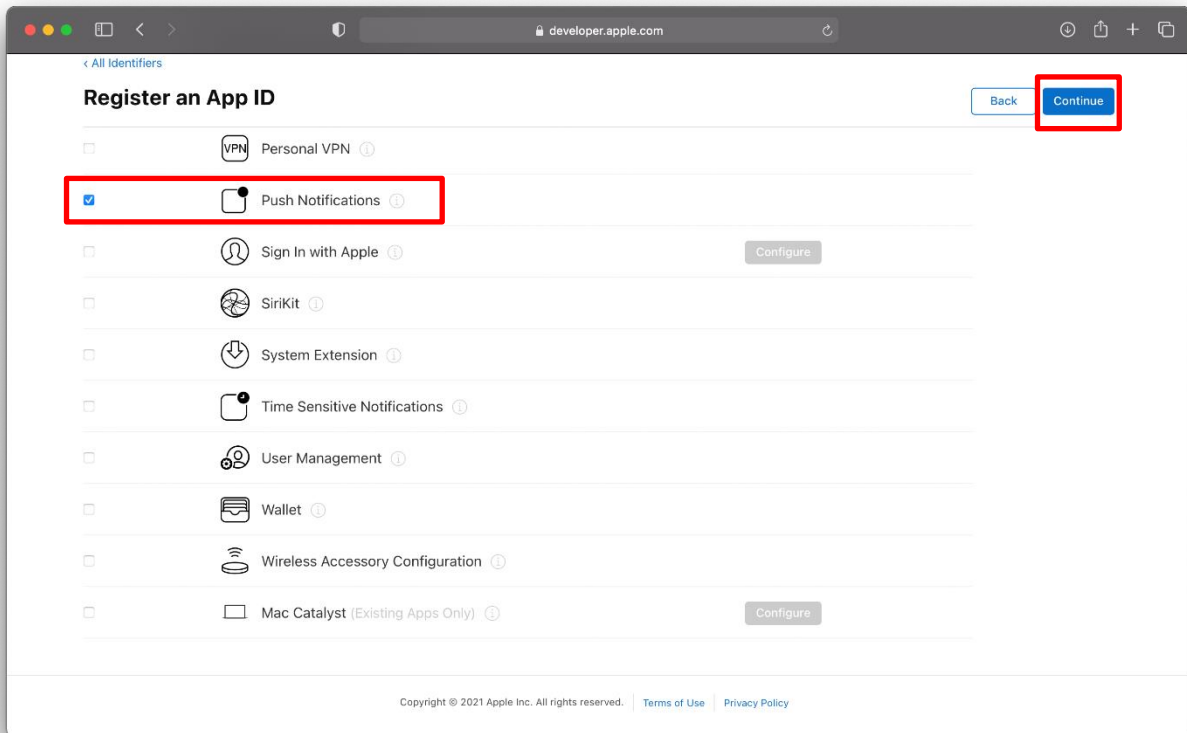


Figure 2.22 Creating App ID for TryBT (5)

2.10 Confirming UDID of iOS Device

Confirm the UDID of iOS device that you will install TryBT.

1. Connect the iOS device to the Mac via a wired connection.
2. Launch Xcode and select [Window]→[Devices and Simulators].
3. Select iOS device in Devices. The value displayed in Identifier field will be the UDID of the iOS device.

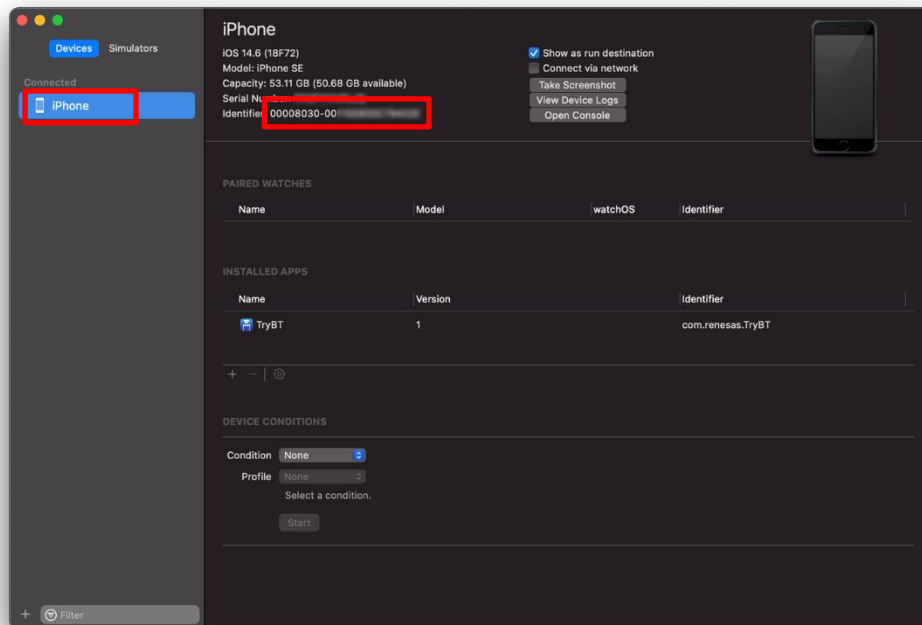


Figure 2.23 Confirming UDID of iOS Device

2.11 Registering iOS device

Register the UDID on the Apple Developer.

1. Select Devices from the left menu and click the blue [+] button next to Devices.

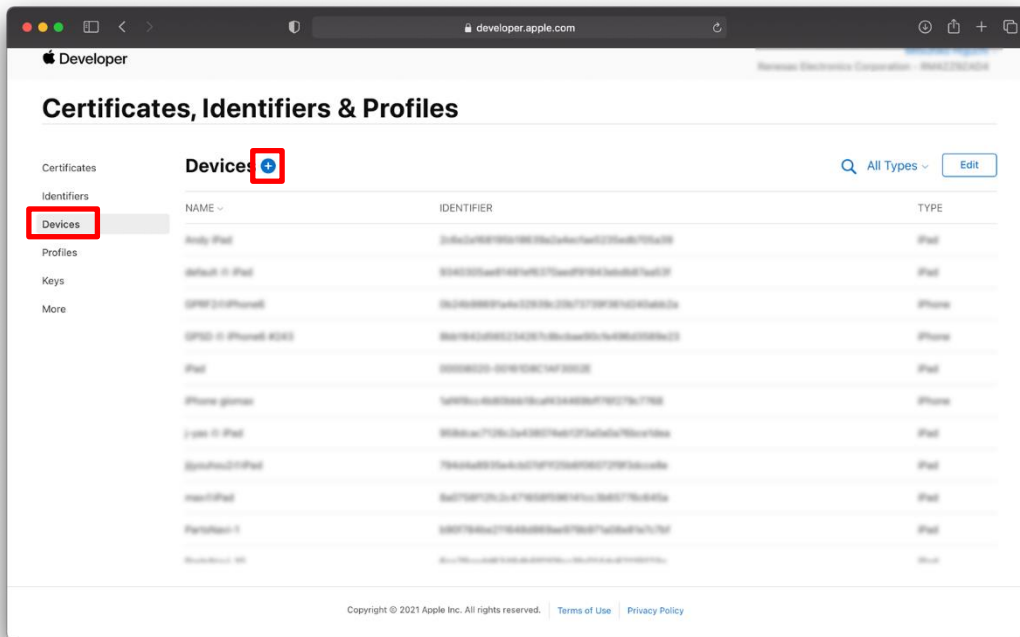


Figure 2.24 Registering iOS Device (1)

2. Select "iOS, tvOS, watchOS" from Platform field. Enter a name to identify iOS choice in Device Name field and the UDID of iOS device in Device ID (UDID) field. Click Continue button and then click Register button in the next page.

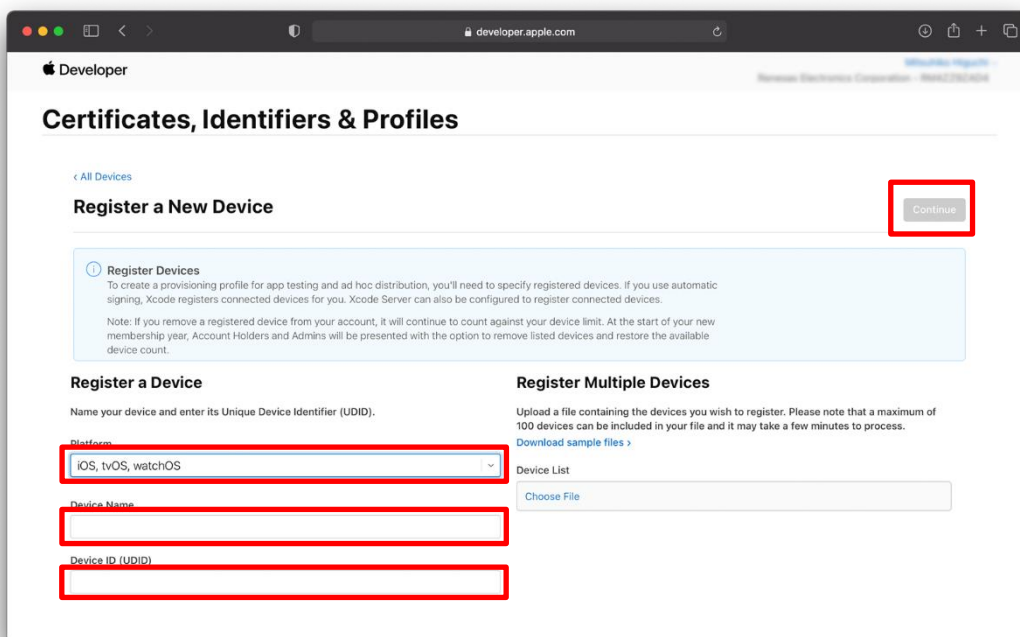


Figure 2.25 Registering iOS Device (2)

2.12 Creating a TryBT Profile

Create a Profile. The Profile is a file that centrally manages the App ID of the application, the device, etc.

1. Select Profiles from the left menu and click the blue [+] button next to Profiles.

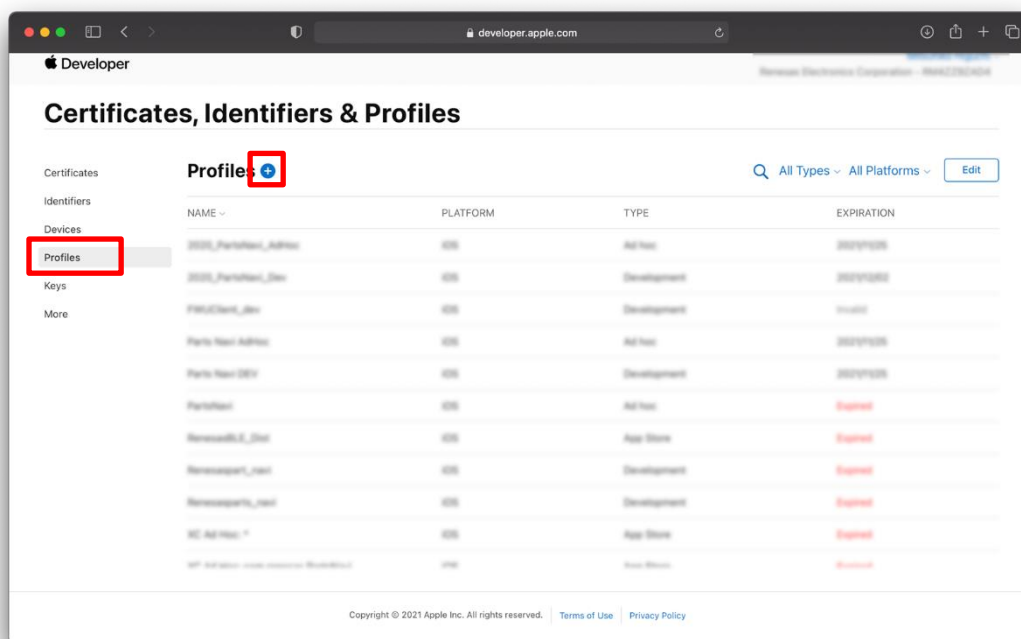


Figure 2.26 Creating a TryBT Profile (1)

2. Select "iOS App Development" and click Continue button.

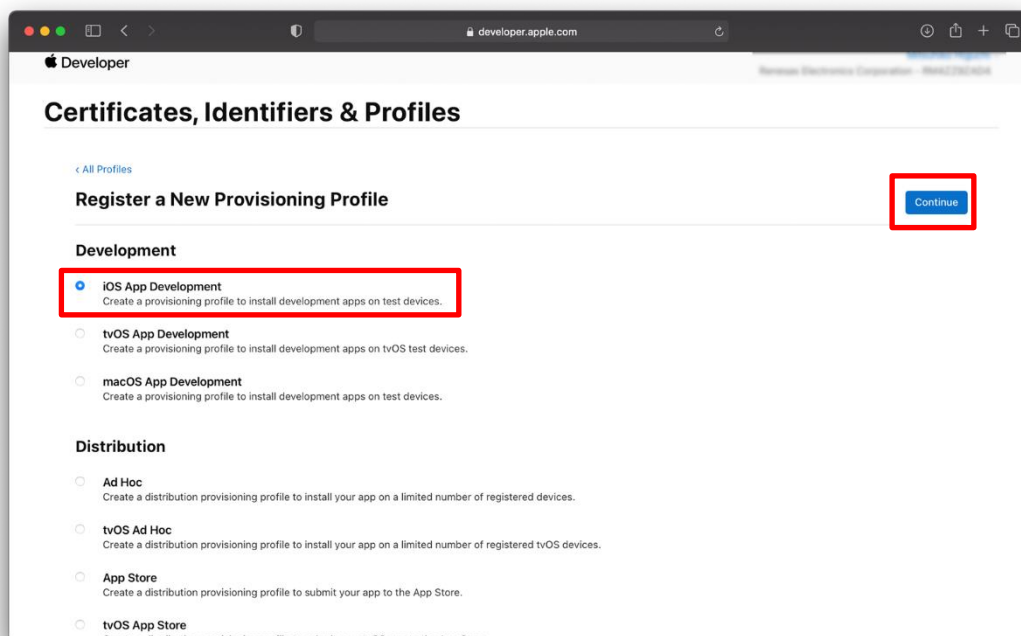


Figure 2.27 Creating a TryBT Profile (2)

3. Select the App ID for TryBT and click Continue button.

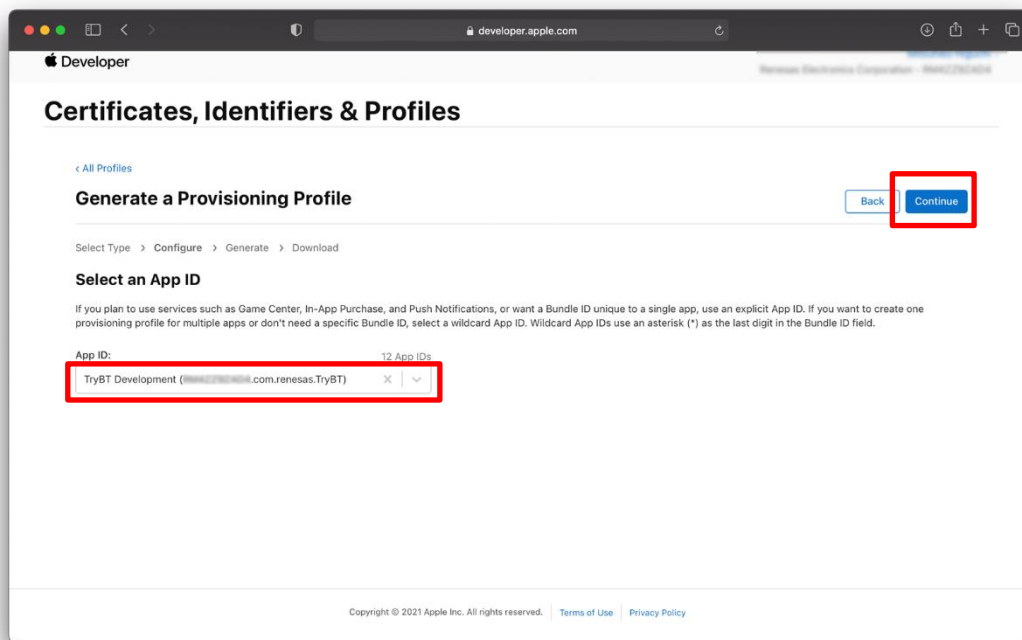


Figure 2.28 Creating a TryBT Profile (3)

4. Specify the Certificate that you created.

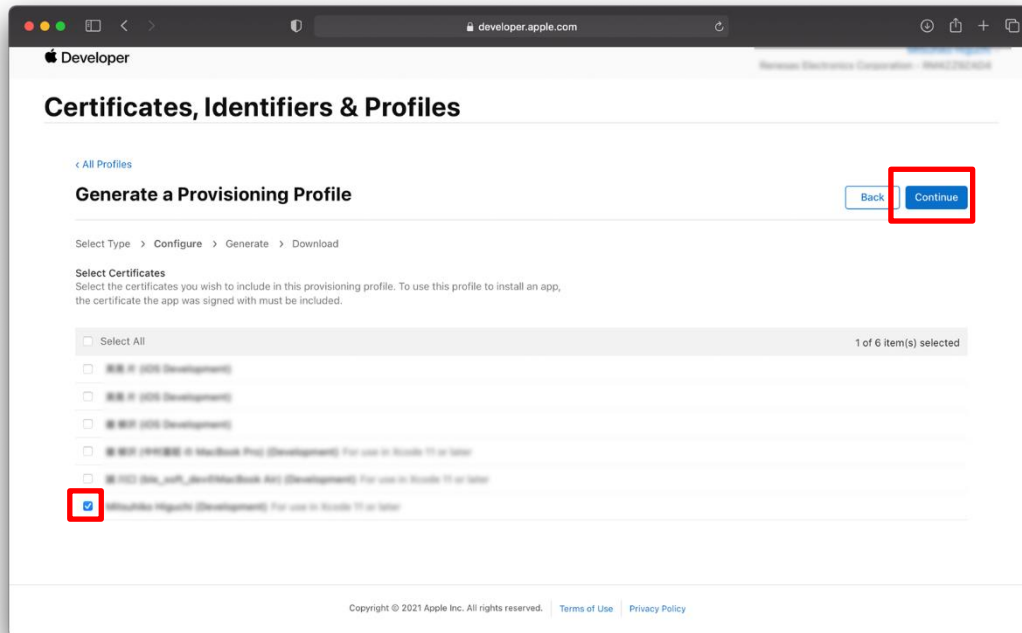


Figure 2.29 Creating a TryBT Profile (4)

5. Select iOS devices and click Continue.

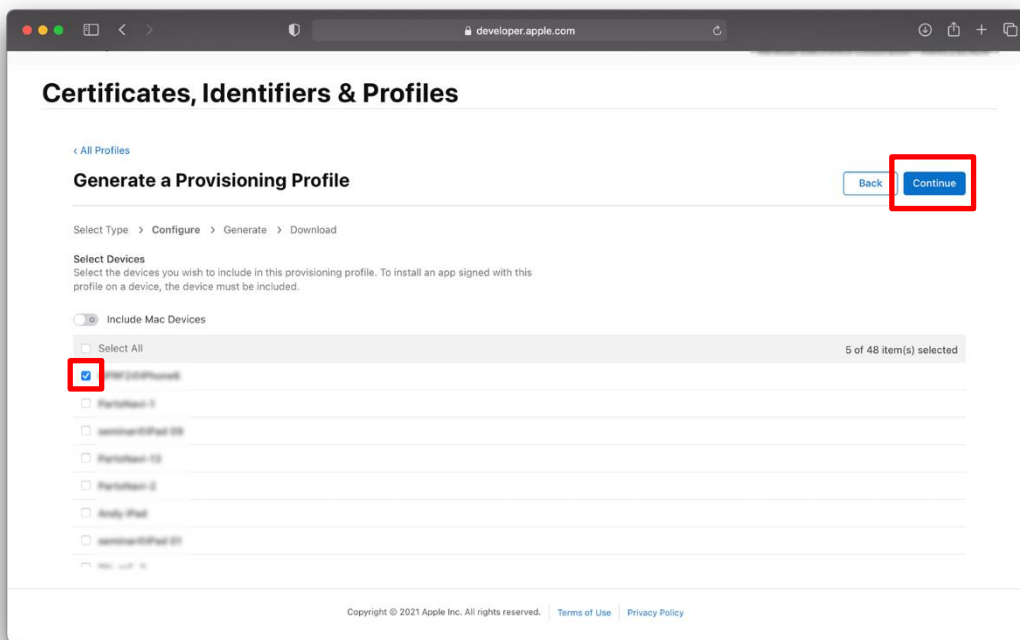


Figure 2.30 Creating a TryBT Profile (5)

6. Enter a name to identify the profile. In this example, "TryBT Profile" is entered. After entering, click Generate button.

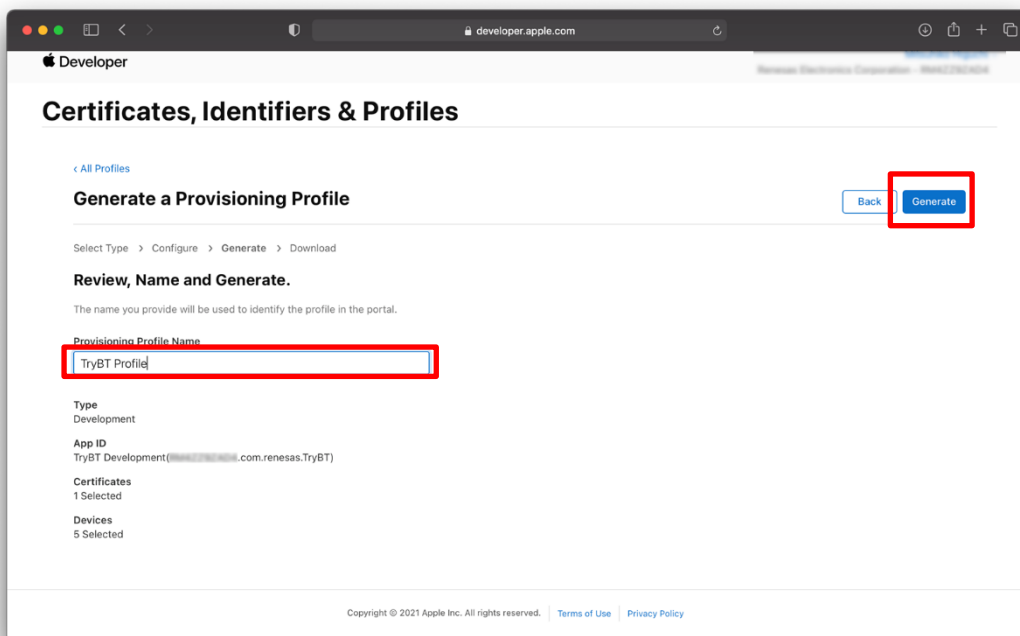


Figure 2.31 Creating a TryBT Profile (6)

7. Click Download button to download the Profile.

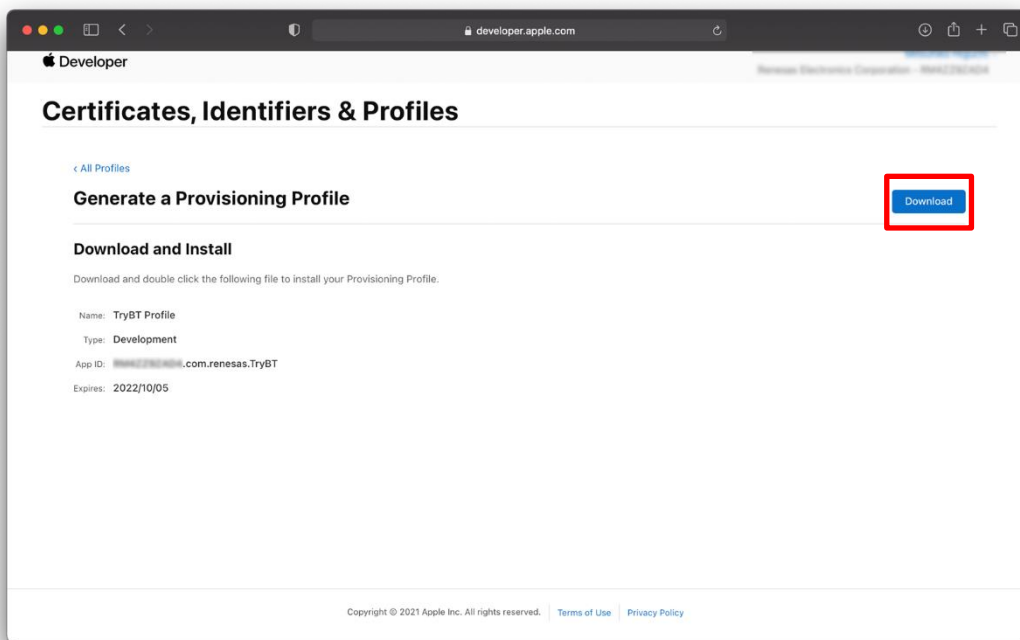


Figure 2.32 Creating a TryBT Profile (7)

2.13 Associating Profile with TryBT Project

Associate the download Profile with the TryBT project.

1. Select TryBT Project in Xcode.
2. Select TryBT under [TARGETS] and then select "Signing & Capabilities" tab.
3. Remove the check from "Automatically manage signing".
4. Enter the App ID you entered to Apple Developer in the Bundle Identifier field.
5. Click the Provisioning Profile. Select "Import Profile" and specify the Profile downloaded by Section 2.12.

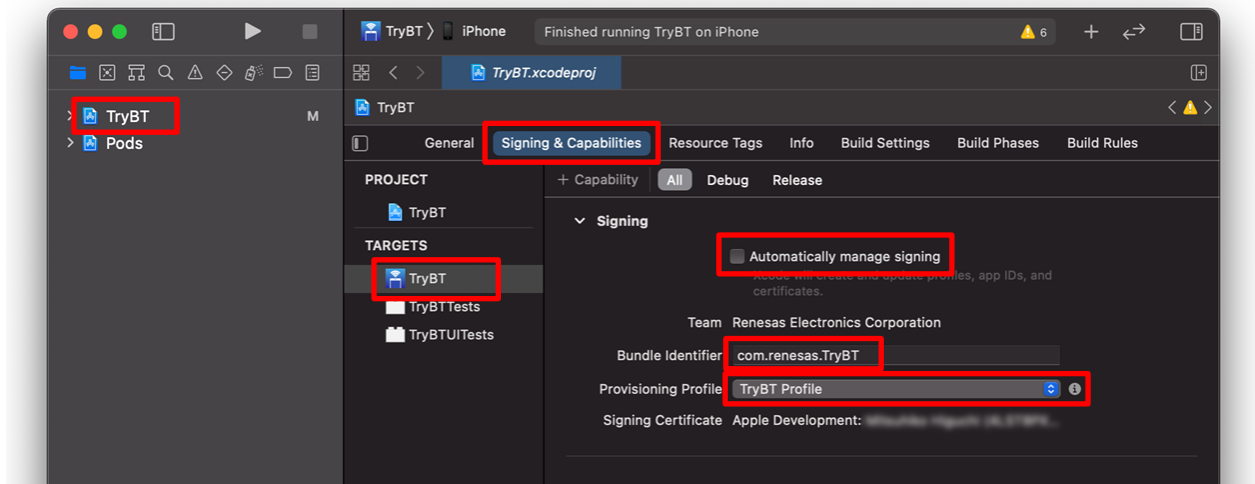


Figure 2.33 Associating Profile with TryBT Project

2.14 Compiling and Installing TryBT

Connect iOS device to Mac via wire and install TryBT.

1. Open the TryBT project.
2. Connect the iOS device to Mac.
3. When the device connects correctly, the device name will be displayed at the top of Xcode.

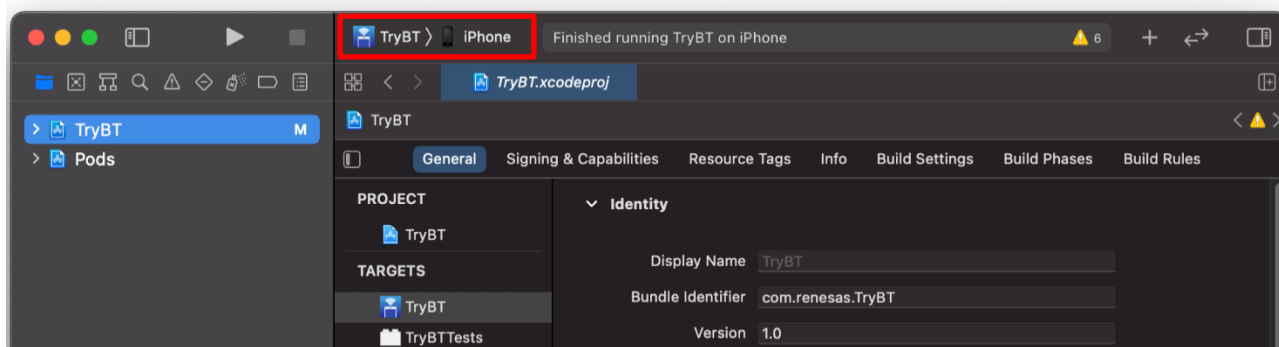


Figure 2.34 Connecting iOS Device (1)

NOTE: If the display at the top of Xcode is not changed after iOS device is connected, click the top of Xcode and select an iOS device from the dropdown list of [iOS Device].

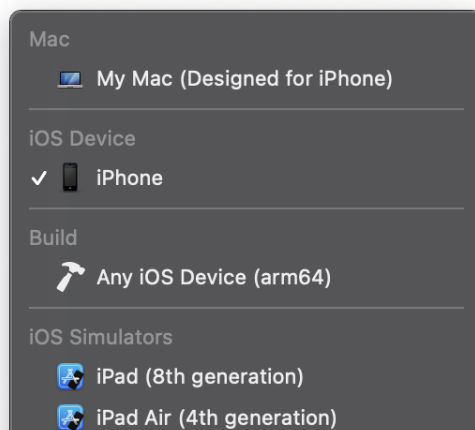


Figure 2.35 Connecting iOS Device (2)

4. Click the play button in the upper left corner of the Xcode to start compiling and installing TryBT.

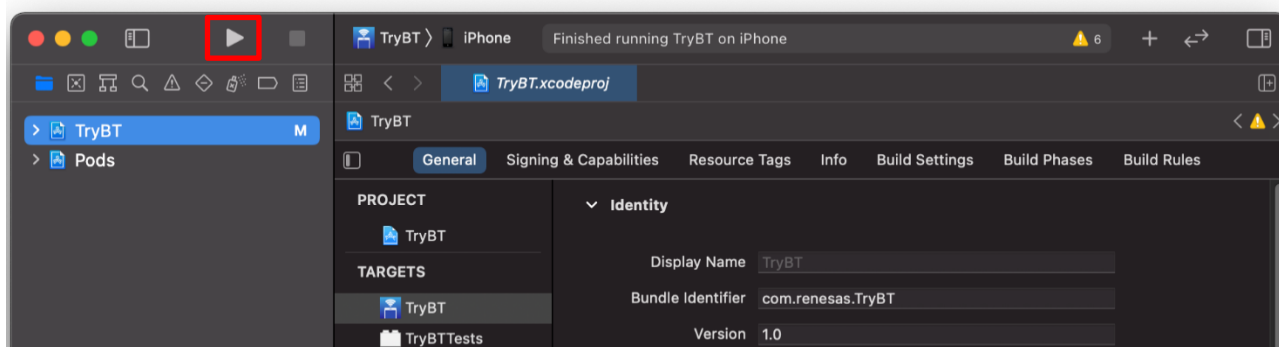


Figure 2.36 Compiling and Installing TryBT

3. Writing Firmware to Evaluation board

Here is an example of using the Target Board for RX23W as the evaluation board. As for the other boards, refer to documents introduced by section 1.1.

A firmware that can communicate with TryBT is written to Target Board for RX23W at the factory. This chapter describes how to write a firmware to Target Board for RX23W again.

1. Access the URL below. After logging in by My Renesas account and agreeing to the disclaimer, you can download a zip file.
<https://www.renesas.com/document/scd/rx23w-group-target-board-rx23w-quick-start-guide-sample-code>
2. Unzip the zip file downloaded by the step 1. Pre-built firmware is the mot file below.
./mot/ble_demo_tbrx23w_profile_server_preinstall_20191009.mot

On the following pages, Steps to write a pre-built firmware to Target Board for RX23W. To write the firmware, the tool below can be used.

Renesas Flash Programmer (Programming GUI)

<https://www.renesas.com/software-tool/renesas-flash-programmer-programming-gui>

3. When you write a firmware, change the ESW 1-2 to ON and connect the ECN1 connector to PC via USB cable.

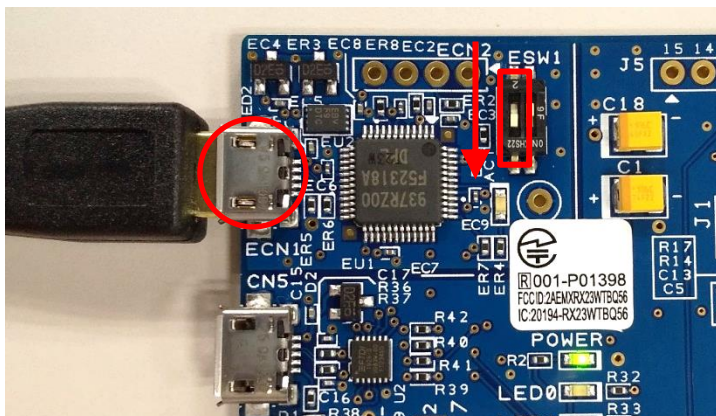


Figure 3.1 Target Board for RX23W Setting for Writing Firmware

4. Start the Renesas Flash Programmer and select "File"→"New Project".

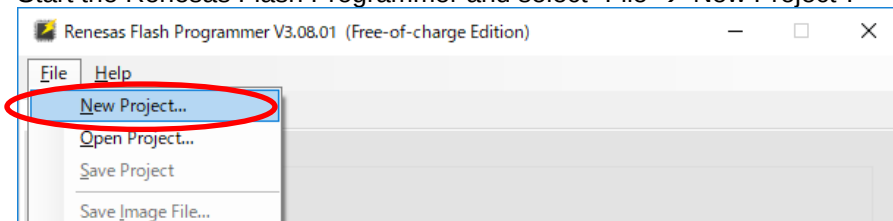


Figure 3.2 Writing Firmware to Target Board for RX23w (1)

5. In the Create New Project dialog, set the following settings and click Connect button.

Microcontroller: RX200
Project Name: any name
Project Folder: any place
Tool: E2 emulator Lite
Interface: FINE
Power: None

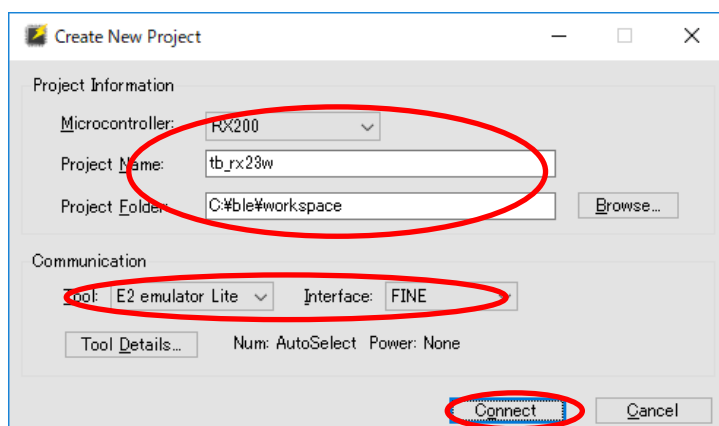


Figure 3.3 Writing Firmware to Target Board for RX23w (2)

6. Configuration steps complete when "Operation completed" is displayed.
7. Specify the following firmware included in the folder of step2 and click Start button.
Program File: mot/ble_demo_tbrx23w_profile_server_preinstall_20191009.mot

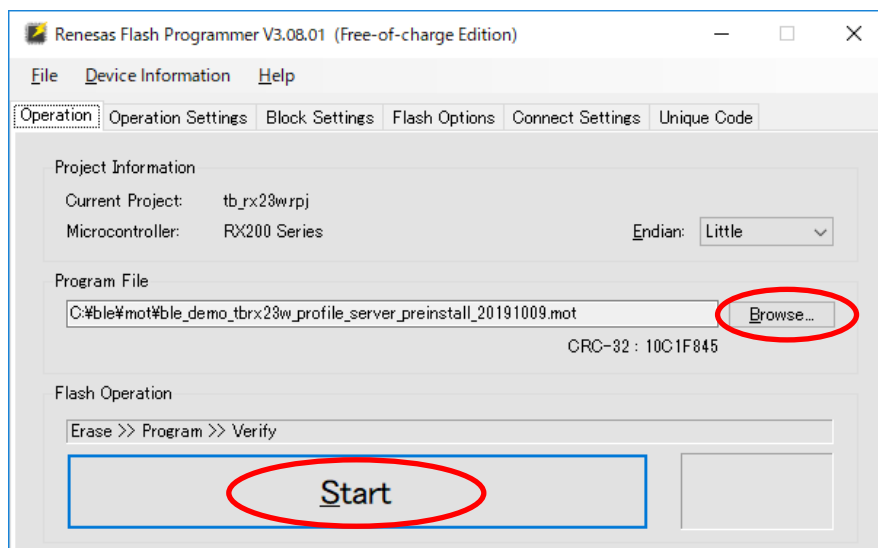


Figure 3.4 Writing Firmware to Target Board for RX23w (3)

8. "Operation completed" will be displayed after writing a firmware.
9. Detach Target Board for RX23W from PC.
10. When you run the firmware, change the ESW 1-2 to OFF and connect the CN5 connector to PC via USB cable.

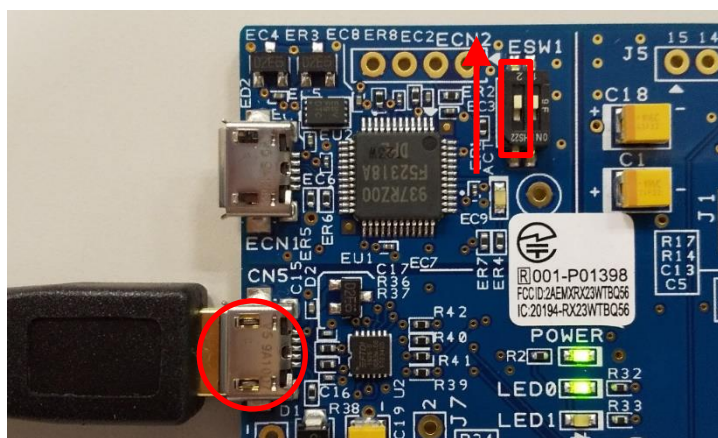


Figure 3.5 Target Board for RX23W Setting for Running Firmware

4. Basic Operation of TryBT

Here is an example of using the Target Board for RX23W as the evaluation board.

4.1 Device List Screen

When the app is launched, Device List screen is displayed. This screen shows a list of connectable devices and its connection status.

Target Board for RX23W is displayed as "RBLE-DEV" in Device List Screen. Tapping "RBLE-DEV" will establish a connection to Target Board for RX23W and display Connected Device Detail Screen.

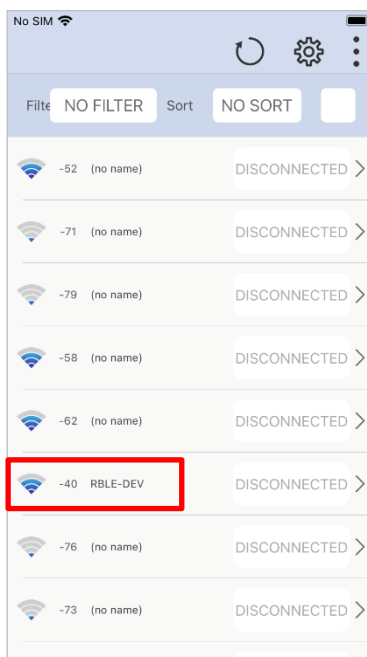


Figure 4.1 Device List Screen (1)

Filtering devices can be performed by selecting Filter type.

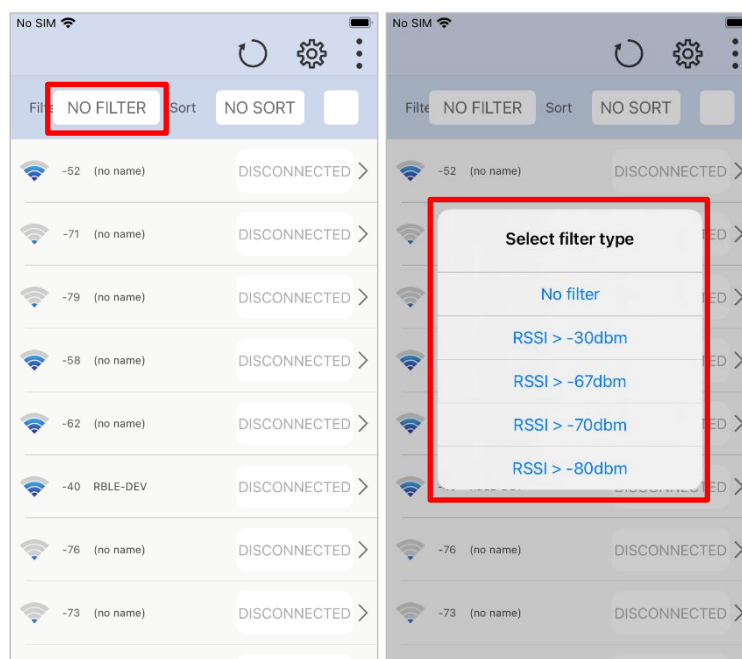


Figure 4.2 Device List Screen (2)

Order of the device list can be specified by selecting Sort order.

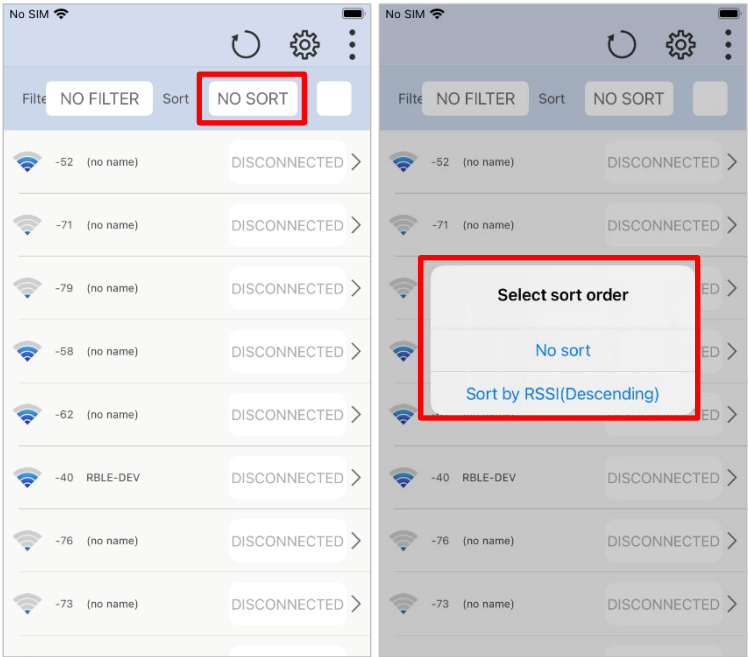


Figure 4.3 Device List Screen (3)

Device list is reloaded by tapping Reload button.

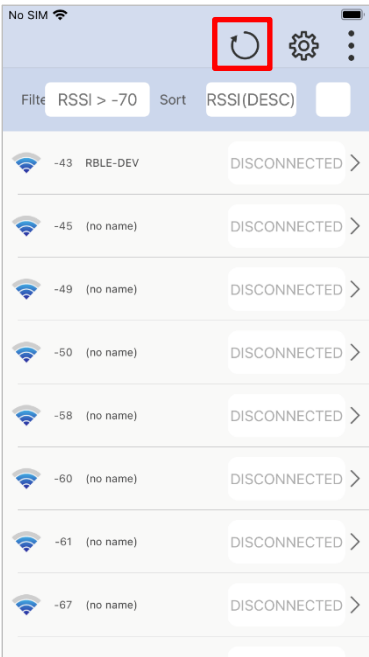


Figure 4.4 Device List Screen (4)

Tapping setting button will display "Register UUID name" and "Bluetooth Settings".

Service Name of GATT Profile and its UUID can be registered by selecting "Register UUID name". Service name registered is displayed in Detailed Information of Connected Device Detail Screen. Note that only one UUID can be registered in this setting. For display example, see

To display Service Name implemented in Target Board for RX23W, register any of the following combinations of UUID and Service Name.

00001800-0000-1000-8000-00805f9b34fb: GAP Service

00001801-0000-1000-8000-00805f9b34fb: GATT Service

58831926-5f05-4267-ab01-b4968e8efce0: LED Switch Service

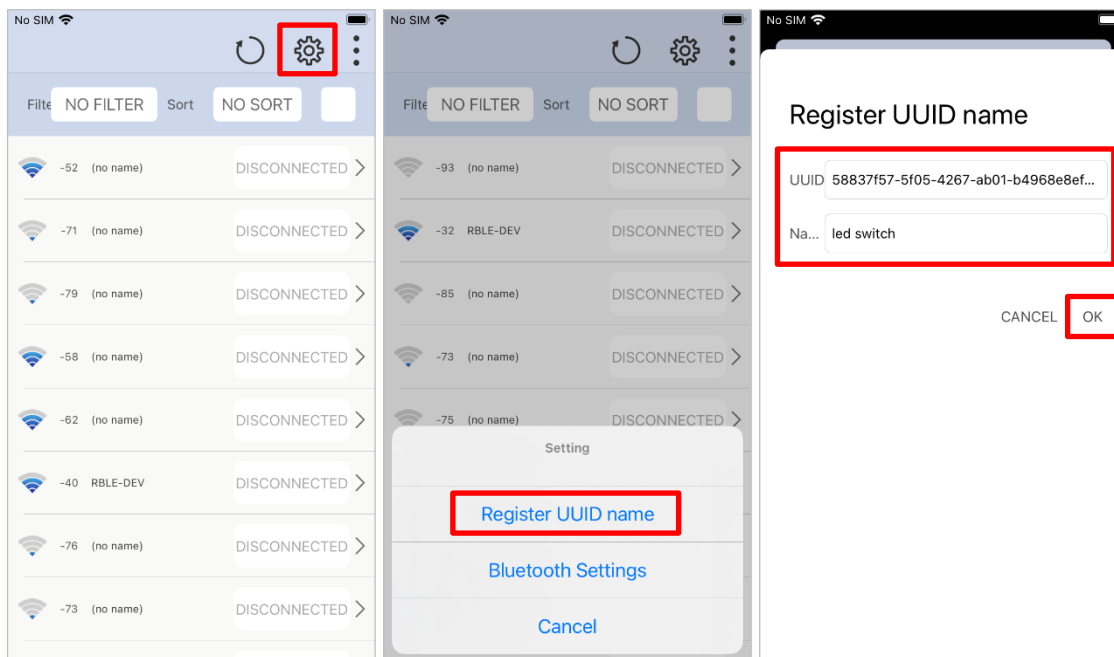


Figure 4.5 Device List Screen (5)

Bluetooth Setting screen of OS can be displayed by selecting "Bluetooth Settings".

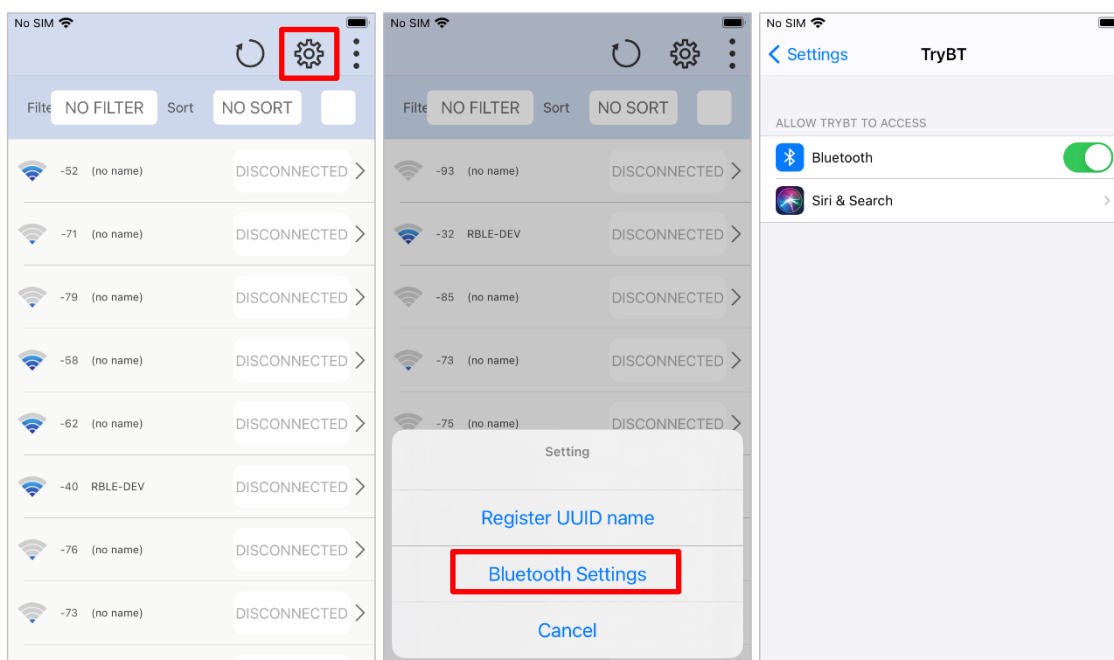


Figure 4.6 Device List Screen (6)

4.2 Connected Device Detail Screen

Connected Device Detail Screen shows the connection status of Target Board for RX23W.

"CONNECTED" indicates the board is connected. Tapping this will terminate this connection. Similarly, "DISCONNECTED" indicates the board is disconnected. Tapping this will establish a connection again.

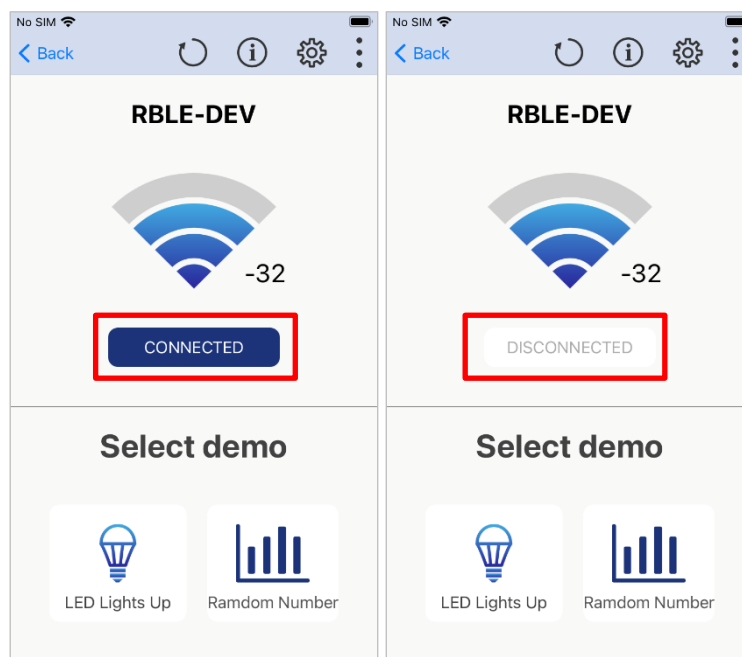


Figure 4.7 Connected Device Detail Screen (1)

Tapping the LED Lights up button will display the Light Demo Screen. To go back to the Connected Device Detail Screen, tap back button on the top left.

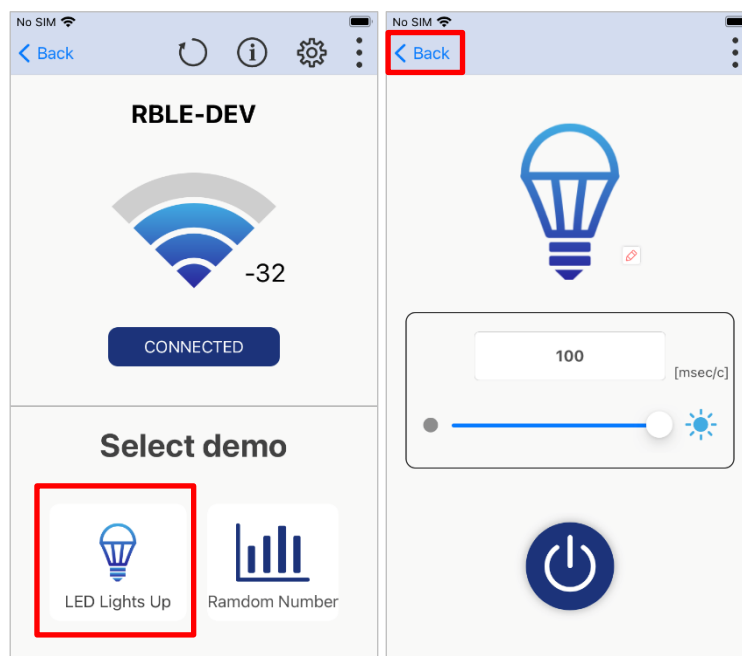


Figure 4.8 Connected Device Detail Screen (2)

Tapping the Random number button will display the data demo screen. To go back to the Connected Device Detail Screen, tap Back button on the top left.

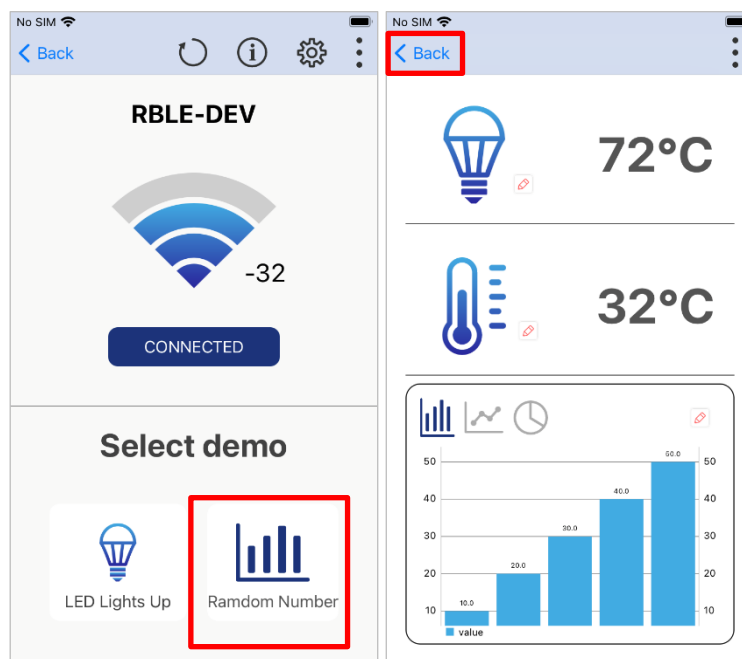


Figure 4.9 Connection Details Screen (3)

Tapping the Reload button will reload the information of Target Board for RX23W connected.

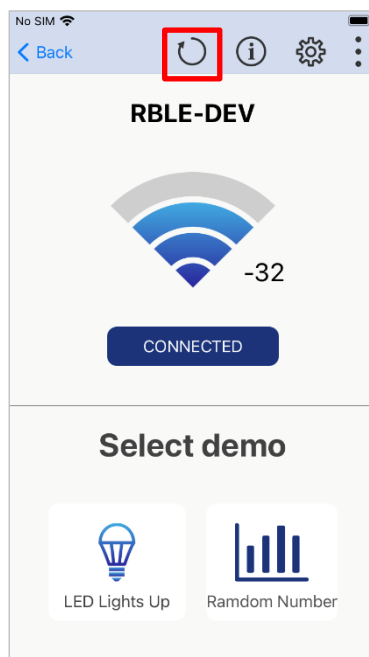


Figure 4.10 Connection Details Screen (4)

Tapping the Info button will display detailed information of evaluation board.

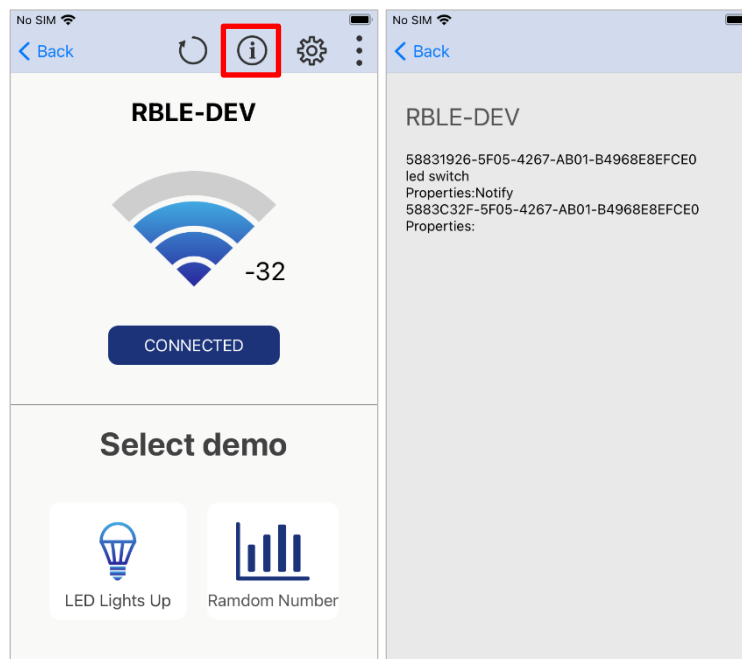


Figure 4.11 Connection Details Screen (5)

Tapping setting button will display "Bluetooth Settings".

Bluetooth Setting screen of iOS can be displayed by selecting "Bluetooth Settings".

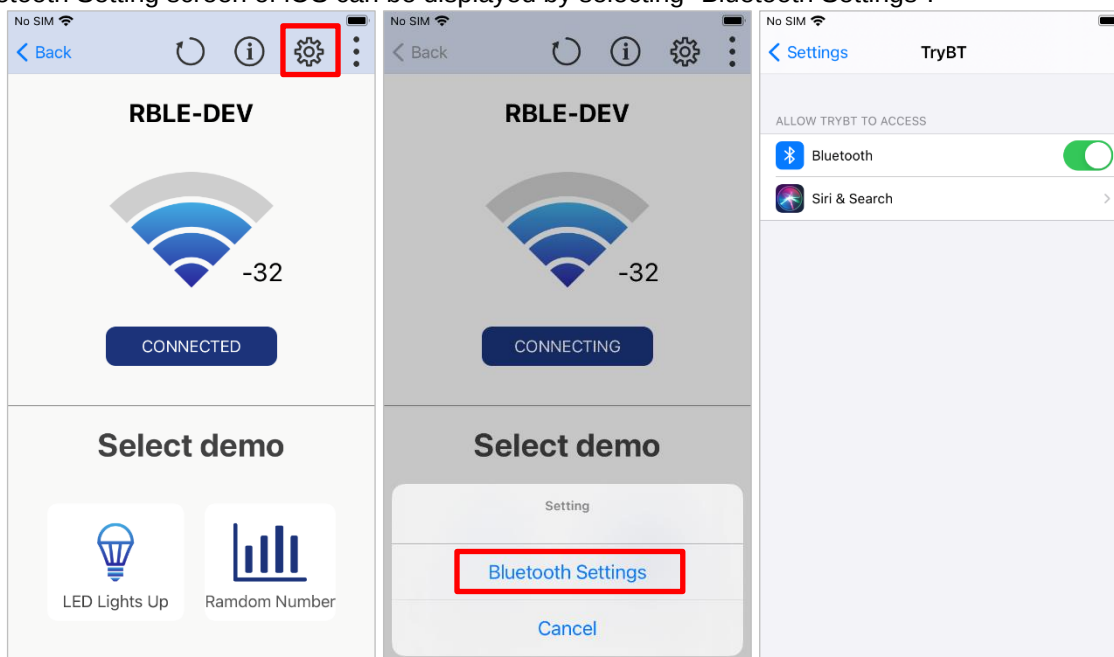


Figure 4.12 Connection Details Screen (6)

4.3 Light Demo Screen

Light Demo screen can operate the LED blinking of Target Board for RX23W LED.

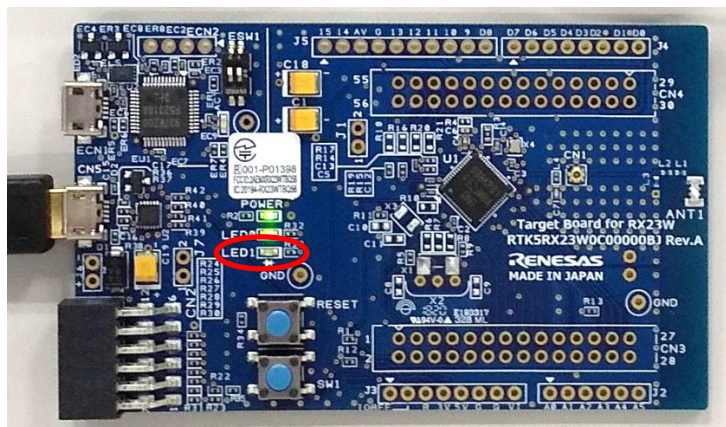


Figure 4.13 User LED on Target Board for RX23W

Specify the blinking interval in milliseconds by using text edit or the slider. (100ms to 10000ms)



Figure 4.14 Light Demo Screen (1)

Switch on and off by tapping the power button.

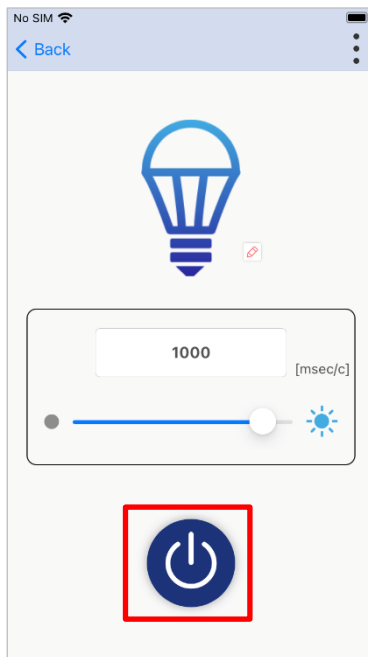


Figure 4.15 Light Demo Screen (2)

TryBT has Customization Mode that can change icon and graph color dynamically. When you tap the pen mark near lamp icon, TryBT changes into Customization Mode and the lamp icon can be changed.

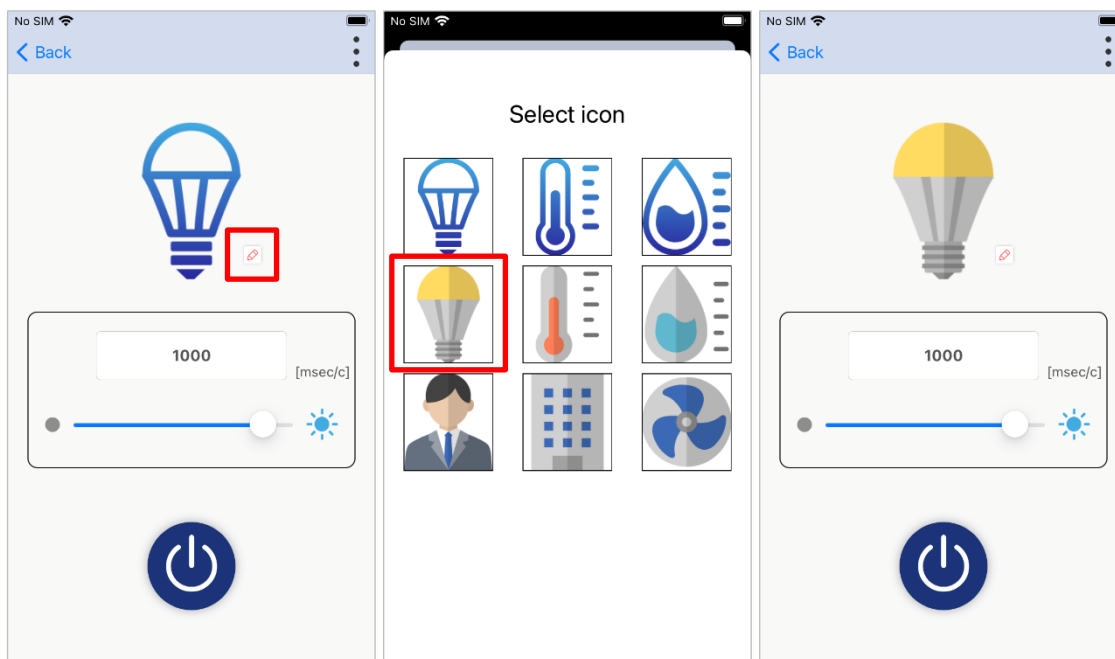


Figure 4.16 Light Demo Screen (3)

4.4 Data Demo Screen

Data Demo screen generates a randomized number and displays temperature, humidity, and graph each time the switch on the Target Board for RX23W is pressed. The graph stores up to ten data points.

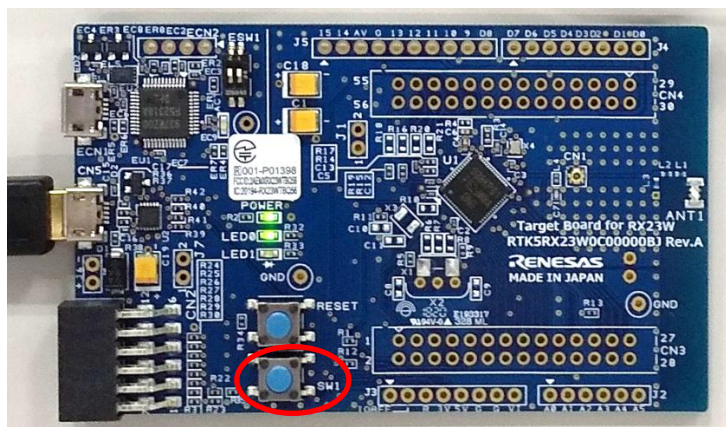


Figure 4.17 User Switch on Target Board for RX23W

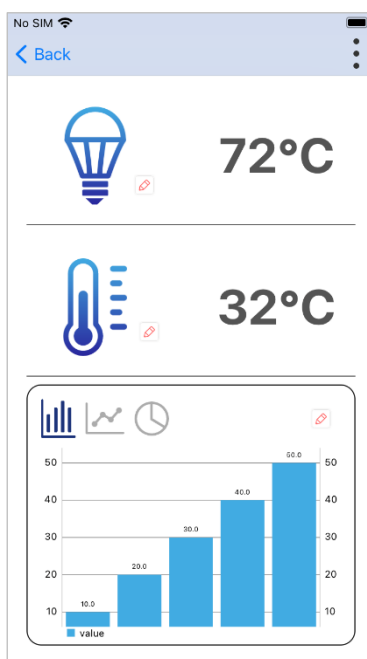


Figure 4.18 Data Demo Screen (1)

Graph format can be switched among bar graph, line graph, and pie chart.

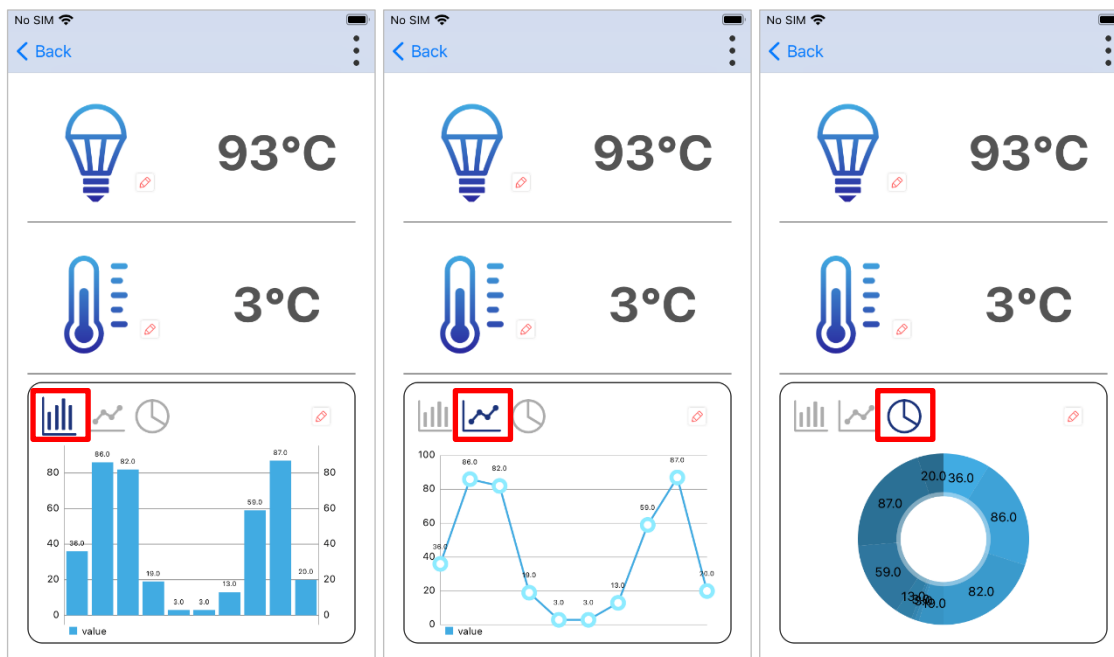


Figure 4.19 Data Demo Screen (2)

When you tap the pen mark near thermometer and hygrometer icons, TryBT changes into Customization Mode and thermometer and hygrometer icons can be changed.



Figure 4.20 Data Demo Screen (3)

When you tap the pen mark near the graph, TryBT changes into Customization Mode and color of the graph can be changed.

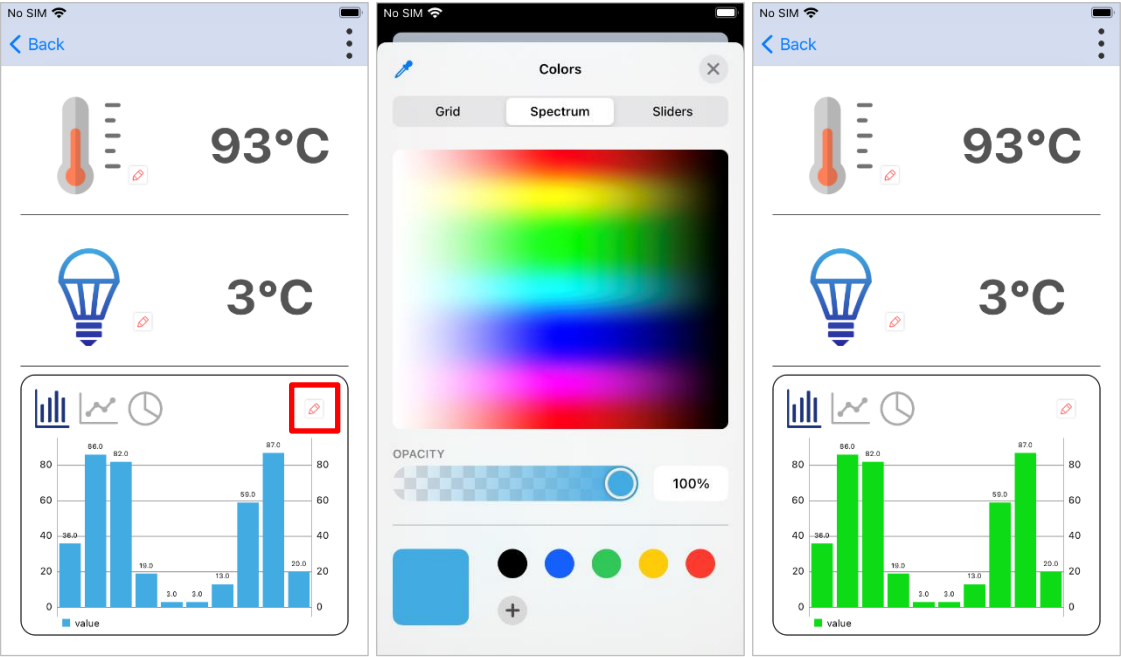


Figure 4.21 Data Demo Screen (4)

5. Customizing TryBT

This chapter describes how to change application title and application icon, splash screen, and Icon data as well as how to enable/disable customization mode of TryBT. When you customize software implementation of TryBT, also refer to description regarding TryBT project described in the following chapters.

5.1 Customizing Application Title

The app title can be changed.

1. Open the TryBT project in Xcode. Change the app title in [Info]→[Bundle Name].

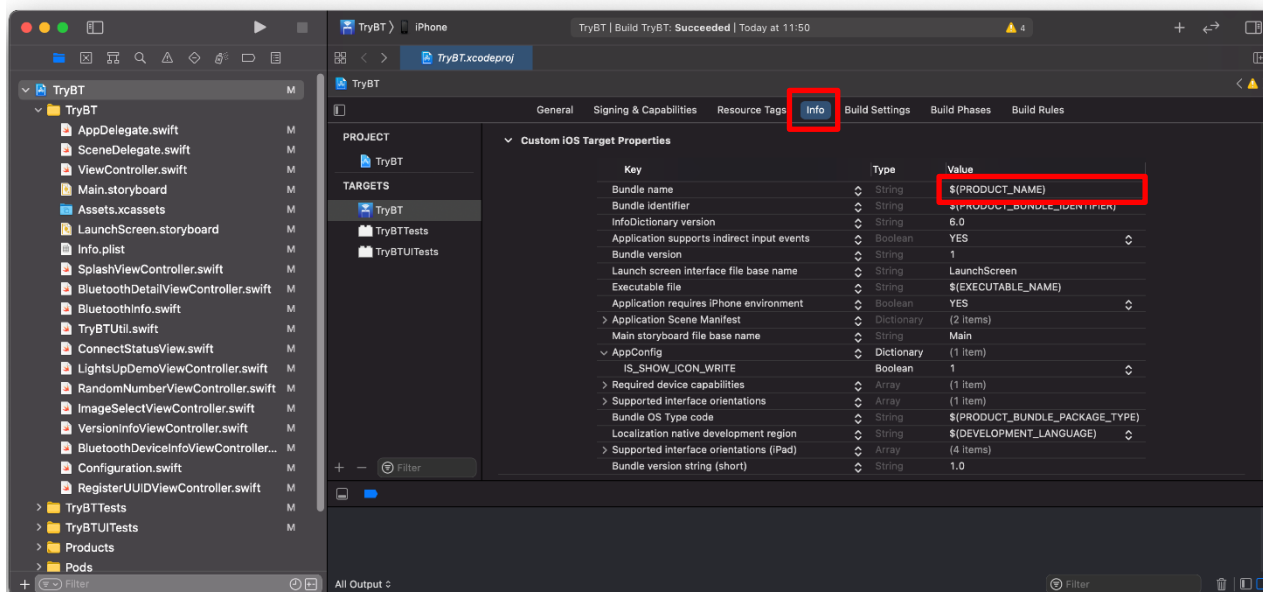


Figure 5.1 Customizing Application Title

5.2 Customizing Application Icons

The app icons can be changed.

1. Open the TryBT project in Xcode. Change the app icons in [Assets.xcassets]→[AppIcon].

NOTE: Change the icons for all resolutions.

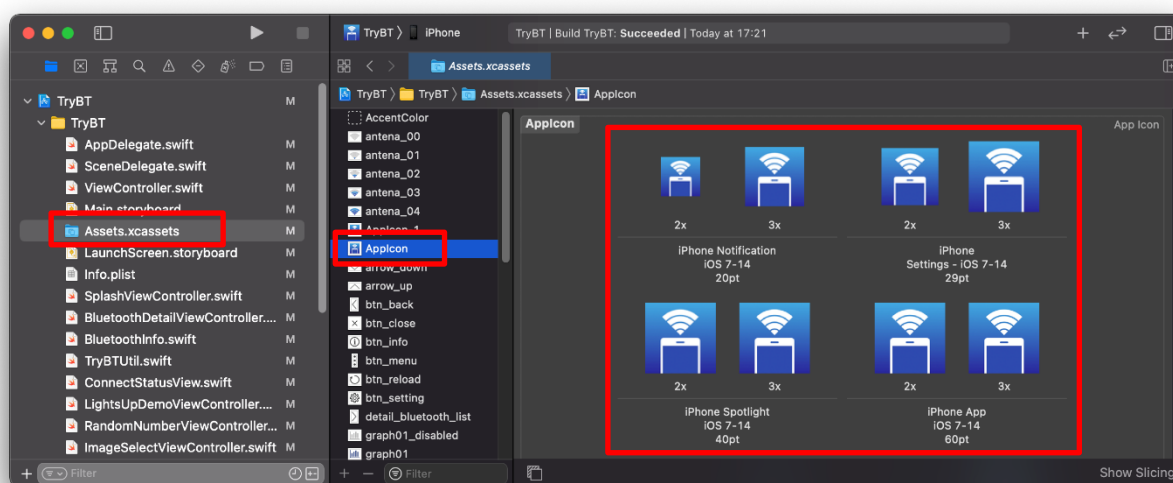


Figure 5.2 Customizing Application Icons

5.3 Customizing Splash Screen

The splash screen, shown when the app is launched, can be changed.

1. Open the TryBT project in Xcode. Change the "splash_background" and the "splash_logo" in [Assets.xcassets].

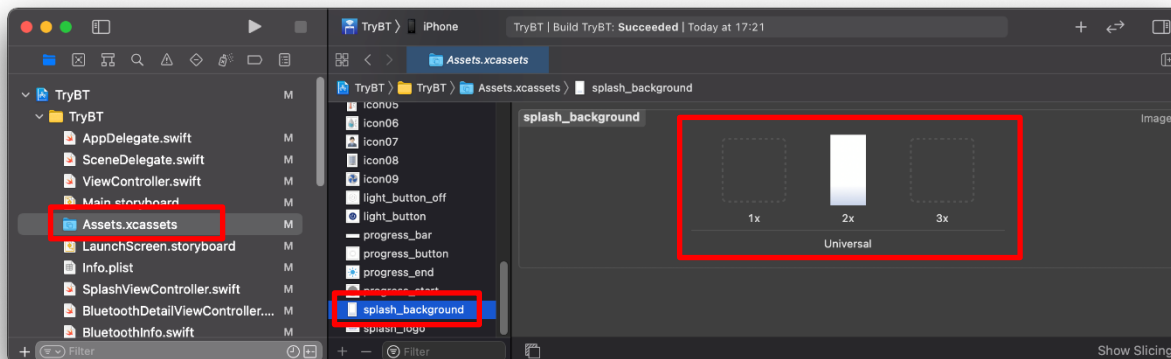


Figure 5.3 Customizing the Splash Screen (splash_background)

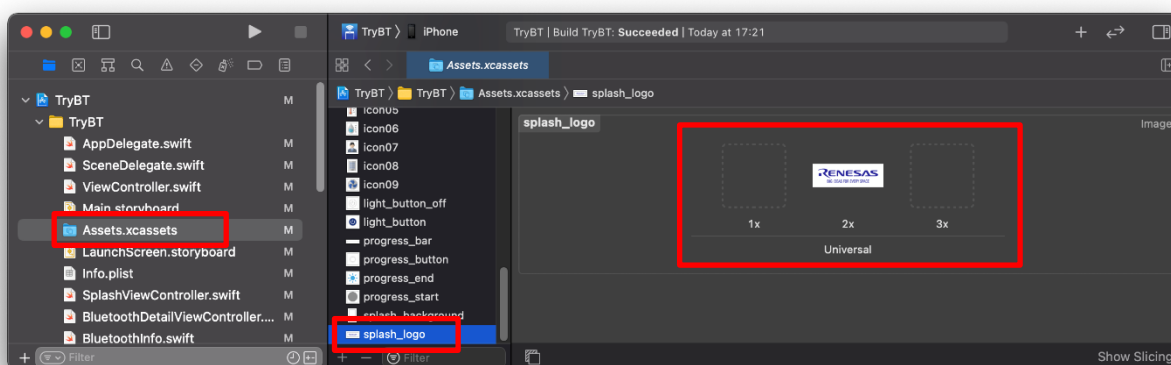


Figure 5.4 Customizing the Splash Screen (splash_logo)

5.4 Customizing Icon Data on the Demo Screen

Customizable icon on the Light Demo screen and Data Demo screen can be changed.

NOTE: Up to nine icons.

NOTE: Prepare png format icons with a resolution of 200x200.

1. Open the TryBT project in Xcode. Change icon01 to icon09 in [Assets.xcassets].

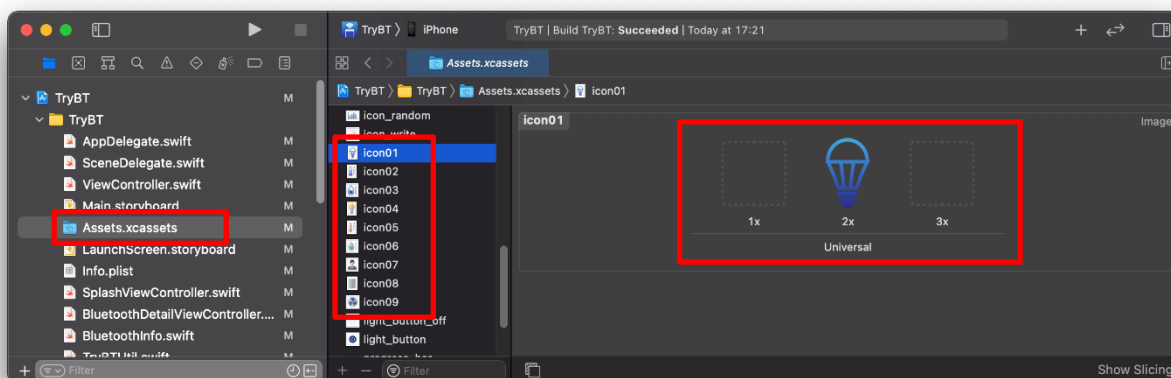


Figure 5.5 Customizing Demo Screen Icon Data

5.5 Enabling/Disabling Customization Mode

This application has a customization mode that allows you to change icons and other features while the application is running.

1. Open the TryBT project in Xcode. Change [AppConfig]→[IS_SHOW_ICON_WRITE] to either 0 or 1 in Info.plist.
 - 1: Enable Customize mode
 - 0: Disable Customize mode

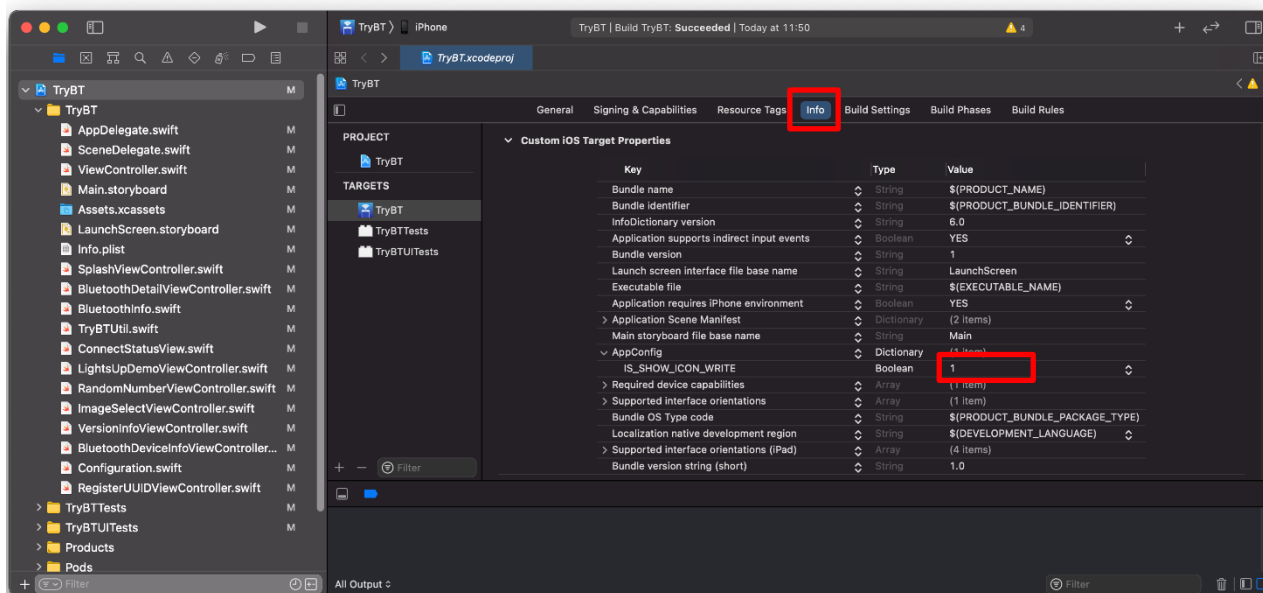


Figure 5.6 Enabling/Disabling Customization Mode

6. File Composition of TryBT

Folder and File Composition of TryBT is shown below. Info.plist, and swift files are described in this chapter.

```
TryBT:
|   Podfile
|
|—Pods/
|—TryBT
|   |   AppDelegate.swift
|   |   BluetoothDetailViewController.swift
|   |   BluetoothDeviceInfoViewController.swift
|   |   BluetoothInfo.swift
|   |   Configuration.swift
|   |   ConnectStatusView.swift
|   |   ImageSelectViewController.swift
|   |   Info.plist
|   |   LightsUpDemoViewController.swift
|   |   RandomNumberViewController.swift
|   |   RegisterUUIDViewController.swift
|   |   SceneDelegate.swift
|   |   SplashViewController.swift
|   |   TryBTUtil.swift
|   |   VersionInfoViewController.swift
|   |   ViewController.swift
|   |
|   |—Assets.xcassets
|   |—Base.lproj
|       |   LaunchScreen.storyboard
|       |   Main.storyboard
|       |
|—TryBT.xcodeproj
|—TryBT.xcworkspace
|—TryBTTests
|—TryBTUITests
```

Figure 6.1 Folder and File Composition of TryBT

6.1 Info.plist

Info.plist is a file for setting the TryBT application. The Info.plist file sets the icon, version, and app display name when the app is installed. The main setting values are shown. Please see the official website for details. (https://developer.apple.com/documentation/bundleresources/information_property_list)

Table 6-1 Info.plist

Attribute Name	Overview
Bundle version string (short)	Specifies the version (string) of the application. e.g. : 1.0.0
Bundle version	Shows the build as a serial number.
Bundle name	Specifies the display name of the application. e.g. : TryBT

6.2 .swift

Each file with a swift extension is a file that constitutes the screen and logic of TryBT. This section provides an overview of each file.

Table 6-2 .swift

Class Name	Overview
AppDelegate	This class manages the lifecycle of the entire application, including the definition of constants in TryBT.
BluetoothDetailViewController	Defines the Bluetooth detail screen.
BluetoothDeviceInfoViewController	Defines the screen displaying information about the connected device.
BluetoothInfo	This is a structure that holds the connection information for one Bluetooth device.
Configuration	This is a utility class for retrieving data from Info.plist.
ConnectStatusView	A custom view class that toggles the display between CONNECTED and DISCONNECTED.
ImageSelectViewController	Defines the screen for icon selection.
LightsUpDemoViewController	Defines the screen where the LED demo is performed.
RandomNumberViewController	Defines the screen for the graph demo.
RegisterUUIDViewController	Defines the UUID registration screen.
SplashViewController	This is the splash screen.
TryBTUtil	This class defines utility methods.
VersionInfoViewController	Defines the screen for displaying version information.
ViewController	Defines the Bluetooth list screen for initial display.

7. Screen Transition of TryBT

The screen class transition diagram for the TryBT screen is shown below.

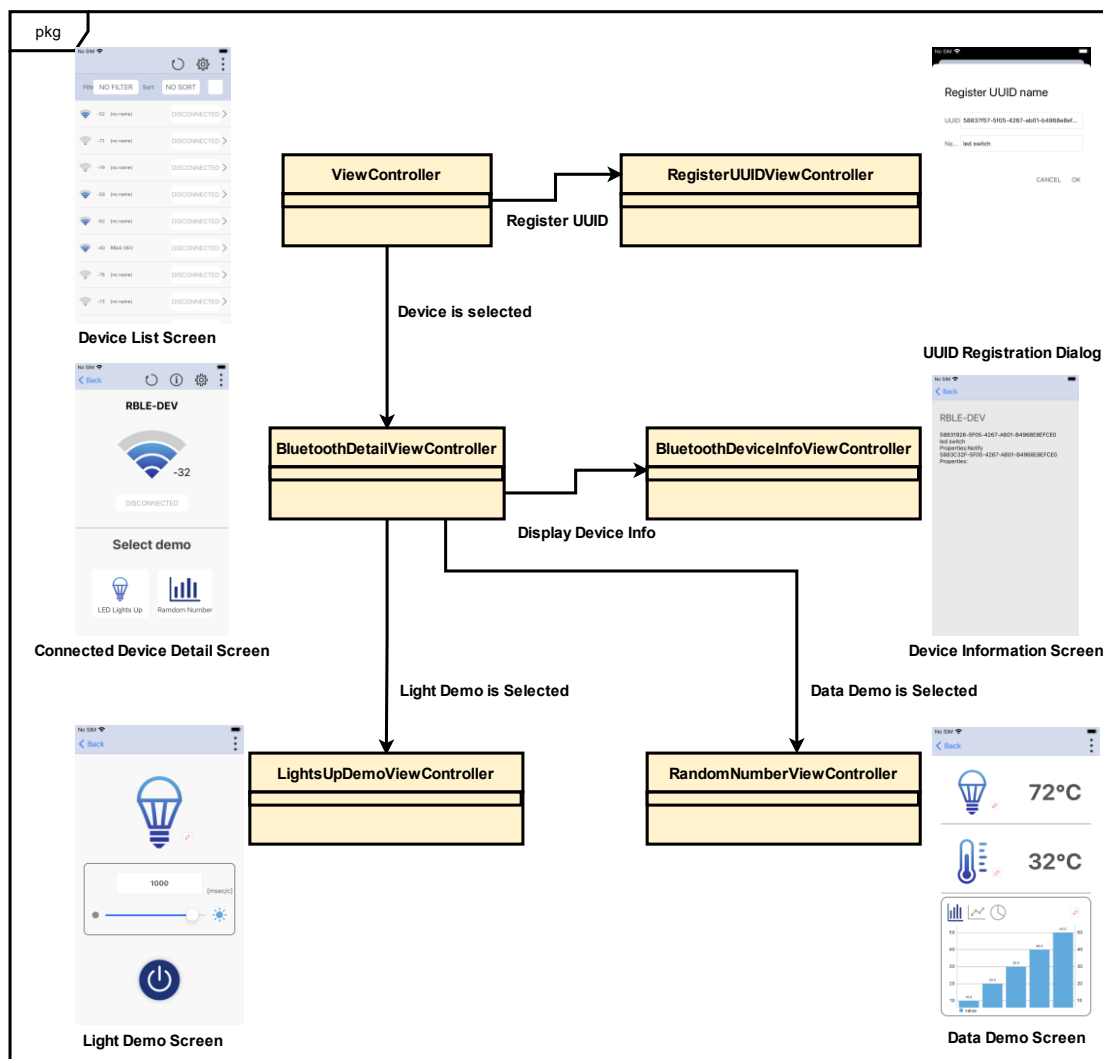


Figure 7.1 Screen and Class Transition of TryBT

8. Bluetooth Communication of TryBT

This chapter explains how to use Bluetooth with TryBT. Please also see the official website (<https://developer.apple.com/documentation/corebluetooth>) for Bluetooth on iOS.

8.1 Performing a scan with Central Manager

TryBT scan peripheral device using CBCentralManager Class.

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    ...  
    centralManager = CBCentralManager(delegate: self, queue: nil, options: nil)  
    ...  
}
```

Figure 8.1 Initializing CBCentralManager class (ViewController.swift line 167)

```
// begin to scan  
func startScan(){  
    print("begin to scan ...")  
    allBluetooth = Array<BluetoothInfo>()  
  
    centralManager.delegate = self  
    centralManager.scanForPeripherals(withServices: nil)  
}
```

Figure 8.2 Scanning (ViewController.swift, line 208)

This saves the search results and displays them in a list.

```
func centralManager(_ central: CBCentralManager,
                    didDiscover peripheral: CBPeripheral,
                    advertisementData: [String: Any],
                    rssi RSSI: NSNumber) {

    print("peripheral.name: ¥(String(describing: peripheral.name))")
    print("advertisementData:¥(advertisementData)")
    print("RSSI: ¥(RSSI)")
    print("peripheral.identifier.uuidString: ¥(peripheral.identifier.uuidString)¥n")

    let bluetoothInfo =
        BluetoothInfo(peripheral: peripheral, advertisementData: advertisementData, rssi: RSSI)

    var isAddBluetooth = true
    for addedBluetoothInfo in allBluetooth{
        if(peripheral.identifier.uuidString == addedBluetoothInfo.peripheral.identifier.uuidString){
            isAddBluetooth = false
            break
        }
    }
    if(isAddBluetooth){
        allBluetooth.append(bluetoothInfo)
        self.refreshTable()
    }
}
```

Figure 8.3 Stored scan result (ViewController.swift, line313)

8.2 Connecting to peripheral device

Connects to the peripheral device. Pass the CBPeripheral class as the first argument of centralManager.connect method.

```
func connect(){
    self.centralManager.delegate = self
    if(self.peripheral != nil){
        self.centralManager.cancelPeripheralConnection(self.peripheral!)
        self.peripheral = nil
    }
    self.centralManager.connect(bluetoothInfo!.peripheral, options: nil)
    // Do any additional setup after loading the view.

    statusLabel.status = ConnectStatusView.ConnectStatus.disconnected

    self.lightsUpDemoButton.isEnabled = false
    self.randomNumberButton.isEnabled = false
}
```

Figure 8.4 Connect to Peripheral (BluetoothDetailViewController.swift, line 168)

The connection completion event is notified to the centralManager didConnect Delegate. TryBT keeps connected Peripheral devices.

```
//call when connect peripheral success
func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral) {
    print("didConnect")
    statusLabel.status = ConnectStatusView.ConnectStatus.connecting

    UserDefaults.standard.set(peripheral.identifier.uuidString,
                              forKey: fromAppDelegate.LAST_CONNECTED_UUID)

    // set delegate
    peripheral.delegate = self
    self.peripheral = peripheral

    self.lightsUpDemoButton.isEnabled = true
    self.randomNumberButton.isEnabled = true
}
```

Figure 8.5 Connection Event (BluetoothDetailViewController.swift, line 229)

8.3 Service Discovery

In TryBT, when the "LED Lights Up" button or "Random Number" button on the connected device details screen is tapped, TryBT execute service discovery by specifying the UUID of the service.

```
@IBAction func lightsUpDemoPressed(_ sender: Any) {

    if(peripheral != nil){
        selectedDemo = .lightsUpDem
        // service-id
        let UUID = CBUUID(string: fromAppDelegate.demo_uuid)
        // 4-1. 利用可能 Service の探索開始
        peripheral!.discoverServices([UUID])
    }else{
        self.alertDisconnectMessage()
    }
}
```

Figure 8.6 Service Discovery Lights Up Demo (BluetoothDetailViewController.swift, line 41)

When the service discovery is successful, discover characteristic.

```
func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    if (error != nil) {
        print("error: %@",String(describing: error))
        return
    }

    for service in peripheral.services!
    {
        // search Characteristic
        switch selectedDemo{
        case .lightsUpDem:
            let serviceUUIDArray =
                NSArray(object: CBUUID(string: fromAppDelegate.LIGHT_UP_SERVICE_UUID))
            peripheral.discoverCharacteristics(serviceUUIDArray as? [CBUUID], for:service as CBService)
        case .randomNumber:
            let serviceUUIDArray =
                NSArray(object: CBUUID(string: fromAppDelegate.RANDOM_NUMBER_SERVICE_UUID))
            peripheral.discoverCharacteristics(serviceUUIDArray as? [CBUUID], for:service as CBService)
        }
    }
}
```

Figure 8.7 Discovery characteristic (BluetoothDetailViewController.swift line 261)

If the characteristic can be searched, the screen will change to each screen.

```
func peripheral(_ peripheral: CBPeripheral,
  didDiscoverCharacteristicsFor service: CBService, error: Error?) {

  if (error != nil) {
    print("error: ¥(String(describing: error))")
    return
  }
  if(service.characteristics!.isEmpty){
    //error
  }else{

    self.characteristics = service.characteristics![0]
    _ = self.characteristics?.uuid

    switch selectedDemo{
    case .lightsUpDem:
      performSegue(withIdentifier: "toLightsUpDemo",sender: nil)
    case .randomNumber:
      performSegue(withIdentifier: "toRandomNumber",sender: nil)

    }
  }
}
```

Figure 8.8 Notification characteristic discovery result (BluetoothDetailViewController.swift, line 283)

8.4 Entering Values on the Light Demo Screen

On the light demo screen, the value is written to peripheral when the slider and TextEdit are changed. Pass the data, characteristic, and writing option as arguments.

```
func writeValue(value:Int){
  var value:UIInt = UInt(value)
  let data: NSData = NSData(bytes: &value, length: 1)

  peripheral!.writeValue(data as Data,
    for: self.characteristics!,
    type: .withResponse)
}
```

Figure 8.9 Write Characteristic (LightsUpDemoViewController.swift, line 142)

8.5 Standby for Notifications on the Data Demo Screen

Sets the notification standby for receiving Rx23W button click events. Enables / disables notifications for Peripheral devices.

```
override func viewDidLoad() {
    super.viewDidLoad()
    let menuBarItem = UIBarButtonItem(
        image: UIImage(named: "btn_menu")?.withRenderingMode(.alwaysOriginal),
        style: .plain,
        target: self,
        action: #selector(menuButtonPressed))

    navigationItem.rightBarButtonItem = [menuBarItem]

    peripheral?.delegate = self
    peripheral?.setNotifyValue(true, for: self.characteristics!)
    ...
}
```

Figure 8.10 Enable Notify (RandomNumberViewController.swift, line51)

Notify from the evaluation board will be notified to the following didUpdateValueFor Delegate.

```
func peripheral(_ peripheral: CBPeripheral,
               didUpdateValueFor characteristic: CBCharacteristic, error: Error?) {

    if (error != nil) {
        print("Write...error: ¥(String(describing: error))")
        return
    }

    let newValue1 = Int.random(in: 0...100)
    let newValue2 = Int.random(in: 0...100)
    let newGraphValue = Int.random(in: 0...100)
    sampleData.append(Double(newGraphValue))
    if(sampleData.count > 20){
        sampleData.remove(at: 0)
    }
    UserDefaults.standard.set(newValue1, forKey: fromAppDelegate.RANDOM_VALUE_1)
    UserDefaults.standard.set(newValue2, forKey: fromAppDelegate.RANDOM_VALUE_2)
    UserDefaults.standard.set(sampleData, forKey: fromAppDelegate.RANDOM_GRAPH_VALUE)
    refreshRandomValue()
}
```

Figure 8.11 Notify Event (RandomNumberViewController.swift, line 319)

Revision History

Rev.	Date	Revised contents	
		Page	Reason for revision
1.00	15.Oct, 2021	-	First edition released

General Precautions in the Handling of Microprocessing Units 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 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 increase 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 guidelines 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 due to the false recognition of the pin state as an input signal become possible.

5. Clock signals

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.

6. 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

1. 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.
6. 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 products, 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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/.