ウェアラブルリングスキャナ

WRS-100 シリーズ

シリアル通信アプリケーション
開発ガイド

AIMEX
Corporation

はじめに

この度はウェアラブルリングスキャナ「WRS-100 シリーズ」をお買い上げ頂き、誠にありがとうございます。

「WRS-100」は、ホスト機器との通信手段として Bluetooth SIG が規定する標準サービスである HID（Human Interface Device）Service と、Nordic Semiconductor 社が独自に規定する NUS（Nordic UART Service）の 2 種類をサポートしています。

HID の場合は OS が標準的に対応しているため、特に追加でソフトウェアをインストールする必要はありませんが、Nordic UART Service を使って通信を行う場合は別途受信プログラムを開発し、対象 OS 上で実行する必要があります。

このマニュアルでは、Nordic UART Service 用のシリアル通信アプリケーションを開発する方法と、各 OS 別のサンプルプログラムの動作について解説します。

本マニュアルはソフトウェア開発者を対象に書かれています。

各 OS 用の開発環境や開発言語については説明しませんので、ご不明な点がございましたら市販の参考書などをご覧ください。

本開発ガイドは WRS-100 シリーズ専用です。その他の端末に使用するアプリケーションを作成する際には、必ず機種毎のドキュメントをご覧ください。

目次

1. WRS-100 の通信仕様

1-1. Bleetooth LE の概要

「WRS-100 シリーズ」は、Bluetooth Low Energy の通信モジュールを搭載しています。Bluetooth Low Energy は、BLE や Bluetooth Smart ともいわれ、Bluetooth SIG が規定するバージョン 4.0 規格で新たに追加された通信規格です。従来の Bluetooth 規格は Bluetooth BR/EDR またはクラシック Bluetooth と呼ばれ、Bluetooth 3.0 以前とも互換性があります。一方で、BLE はクラシック Bluetooth とは通信プロトコル上の互換性が無い、全く新規の通信規格になります。

BLE の特徴は次の通りです。

・超低消費電力

・2.4GHz 帯を使用して最大 1Mbps の通信が可能。

・ブロードキャスト通信を利用した近接通知が可能（例…iBeacon 等）。

・ペアリングレスでの接続が可能。

・通信規格上は同時接続数に制限なし。

本マニュアルでは、以降 Bluetooth Low Energy を BLE、Bluetooth BR/EDR をクラシック Bluetooth と呼びます。

1-2. 接続形態

BLE では、ホスト機器となるコントローラ側をセントラル、デバイスをペリフェラルと呼びます。

Windows、Android、iOS などのホスト機器がセントラル、WRS-100 がペリフェラルとなって通信を行います。WRS-100 シリーズでの代表的な接続形態を下図に示します。

1-3. プロファイル/サービス/キャラクタリスティック

BLE では、セントラルとペリフェラルは GAP（Generic Access Profile）というプロファイルで接続し、GATT（Generic Attribute Profile）というプロファイルに従ってデータ通信を行います。GATT は、データ構造の読み出し、データの読み書き、データの変更通知など BLE デバイス間のデータ交換の論理的な仕様を規定してます。

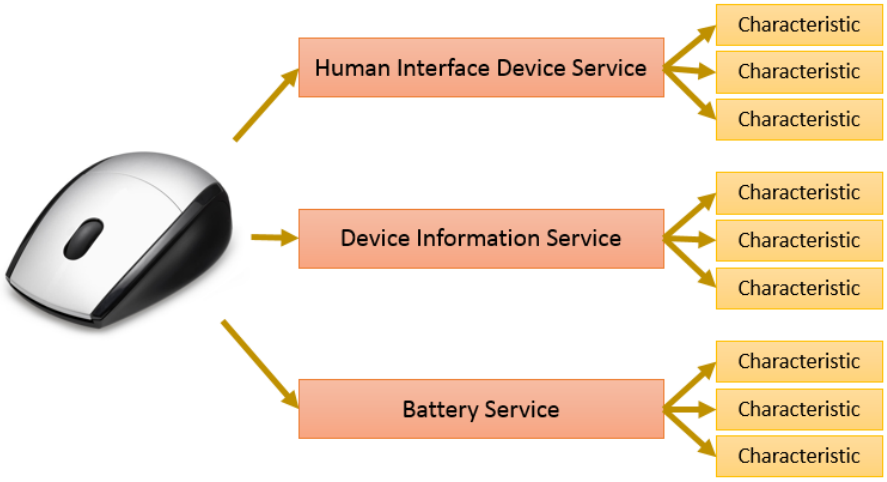
サービスは BLE 機器が提供する一連のデータ交換に関する機能を取りまとめた単位です。サービスは GATT プロファイル上に実装されます。各サービスにはサービスを識別するための 16byte の UUID が割り当てられています。Bluetooth SIG で定義された標準的なサービスは短縮型の UUID が使用可能で、例えば温度計サービスは 2byte の 0x1809、心拍計サービスは 2byte の 0x180F が割り当てられています。

以下は、標準サービスの例です。

サービス名	UUID	サービス概要
Automation IO	0x1815	デジタル入出力制御
Battery Service	0x180F	バッテリー情報取得
Blood Presure Service	0x1810	血圧計情報の取得
Current Time Service	0x1805	現在時刻取得
Cycling Speed and Cadence	0x1816	サイクリング時のスピードや回転数伝達
Device Information	0x180A	デバイス情報の取得
Health Thermometer	0x1809	体温計情報の取得
Heart Rate	0x180D	心拍計情報の取得
Human Interface Device	0x1812	キーボードやマウスなどの入力情報取得
Location and Navigation	0x1819	位置情報やナビゲーション情報の伝達
Phone Alert Status Service	0x180E	電話の着信情報などの伝達
Running Speed and Cadence	0x1814	ランニング時のスピードや歩数伝達
Weight Scale	0x181D	体重計情報の取得

各サービスの中には、複数のキャラクタリスティックという属性単位があります。実際のデータのやり取りは、キャラクタリスティックに対して読み書きすることで行われます。キャラクタリスティックにもサービス同様に固有の UUID が割り当てられています。例えば心拍計サービスの場合は Heart Rate Measurement（0x2A37）、Body Sensor Location（0x2A38）、Heart Rate Control Point（0x2A39）の 3 つのキャラクタリスティックから構成されています。

一つのデバイスは、複数のサービス、複数のキャラクタリスティックを持つことができます。



バッテリ駆動の BLE ワイヤレスマウスの例

キャラクタリスティックに対するアクセス方式としては、write、read、notify の 3 種類があり、どのアクセス方式をサポートしているかはキャラクタリスティックにより異なります。キャラクタリスティックに write するとセントラルからペリフェラルへのデータの送信、read するとペリフェラルからのデータの受信が行われるイメージです。notify がサポートされている場合、セントラルから都度 read でアクセスしなくても、ペリフェラルからセントラルにイベント通知の形式でデータを伝送することができます。

1-4. WRS-100 のサービス仕様

「WRS-100 シリーズ」が搭載するサービスについて説明します。

読み取ったバーコードデータを転送するためのサービスとして、Nordic UART Service と HID Service の 2 種類をサポートしており、どちらかをコマンドバーコードで排他的に選択して使用します。出荷時設定は「Nordic UART Service（シリアル通信モード）」です。

サービス名	概要
Nordic UART Service	Nordic Semiconductor 社が独自に仕様を定義したサービスです。 シリアル通信(UART)をエミュレーションするサービスで、スキャンデータの送信に使用します。
HID Service	Bluetooth SIG 標準で定義されたサービスです。 キーボード入力をエミュレーションするサービスで、スキャンデータの送信に使用します。
Battery Service	Bluetooth SIG 標準で定義されたサービスです。 スキャナの電池残量の通知に使用します。
Device Information Service	Bluetooth SIG 標準で定義されたサービスです。 スキャナのデバイス情報（メーカ名、ソフトバージョン番号、シリアル番号など）の取得に使用します。

①Nordic UART Service

NUS（Nordic UART Service）は Bluetooth SIG が定義する汎用サービスではなく、Nordic Semiconductor 社が独自に定義したサービスです。WRS-100 からホスト端末へ読み取ったバーコードデータを転送するために使用します。

本マニュアルでは、この Nordic UART Service を使用する際のプログラム作成方法について解説しています。

Nordic UART Service と、キャラクタリスティックの UUID は次の通りです。

サービス/キャラクタリスティック	UUID
Nordic UART Service	6E400001-B5A3-F393-E0A9-E50E24DCCA9E
RX Characteristic	6E400002-B5A3-F393-E0A9-E50E24DCCA9E
TX Characteristic	6E400003-B5A3-F393-E0A9-E50E24DCCA9E

読み取ったバーコードデータは、TX Characteristic を通じて取得します。TX Characteristic の notify 機能を使ってイベント駆動で読み出しを行います。

WRS-100 シリーズでは、RX Characteristic を使ったホスト端末→スキャナ方向へのデータ伝送はサポートしていません。

BLE の GATT 通信では、1 パケットあたりのデータサイズは 20byte が上限になります。

このため、WRS-100 では、大きなバーコードを読み取った場合は、複数のパケットに分割して送信します。各パケットは以下のデータ構造となります。

1byte 目	2byte 目	3byte 目	4byte 目	5byte 目	6byte 目	…	20byte 目
分割数	番号	Data[x]	Data[x+1]	Data[x+2]	Data[x+3]	…	Data[x+17]

1byte 目	データを分割した数。 送信データ長が 18Byte を超えるときに分割が発生する。
2byte 目	分割されたデータのうちの、何番目のパケットかを示す。
3byte 目～20byte 目	受信データ本体。 受信データの詳細は「1-6. データフォーマット」参照。

例 1）データ長が 18Byte の場合

データ部は分割せず、1 パケットで送信する。

1 パケット目	0x01	0x01	Data[0]	…	Data[17]

例 2）データ長が 40Byte の場合

データ部を 18Byte, 18Byte, 4Byte の 3 パケットに分割して送信する。

1 パケット目	0x03	0x01	Data[0]	…	Data[17]
2 パケット目	0x03	0x02	Data[18]	…	Data[35]
3 パケット目	0x03	0x03	Data[36]	…	Data[39]

ホスト端末の受信プログラム側では、TX Characteristic の読み出し時に受信データが分割されているかどうかを判断し、分割されている場合は結合して 1 つのデータに再構築します。

②HID Service

HID (Human Interface Device) Service は Buetooth SIG が定義する汎用サービスです。キーボードやマウスなど入力デバイスをエミュレートするためのサービスであり、WRS-100 からホスト端末へ読み取ったバーコードデータを転送するために使用します。

HID Service は各 OS で標準的にサポートされているため、このサービスを使用する場合は新規にプログラムを作成することなく WRS-100 からのデータを取り込むことができます。

HID Service と、キャラクタリスティックの UUID は次の通りです。

サービス/キャラクタリスティック	UUID
Human Interface Device Service	00001812-0000-1000-8000-00805F9B34FB
Protocol Mode	00002A4E-0000-1000-8000-00805F9B34FB
Report	00002A4D-0000-1000-8000-00805F9B34FB
Report Map	00002A4B-0000-1000-8000-00805F9B34FB
Boot Keyboard Input Report	00002A22-0000-1000-8000-00805F9B34FB
Boot Keyboard Output Report	00002A32-0000-1000-8000-00805F9B34FB
HID Information	00002A4A-0000-1000-8000-00805F9B34FB
HID Control Point	00002A4C-0000-1000-8000-00805F9B34FB

※青字部分が短縮型 UUID

HID Service に関する詳細は、Bluetooth SIG の Web サイトに記載されています。

https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.service.human_interface_device.xml

③Device Information Service

Device Information Service は Buetooth SIG が定義する汎用サービスです。デバイス情報を取得するために使用します。

Device Information Service と、WRS-100 が有効な値を設定しているキャラクタリスティックの UUID は次の通りです。

サービス/キャラクタリスティック	UUID
Device Information Service	0000180A-0000-1000-8000-00805F9B34FB
Manufacturer Name String	00002A29-0000-1000-8000-00805F9B34FB
Serial Number String	00002A25-0000-1000-8000-00805F9B34FB
Software Revision String	00002A28-0000-1000-8000-00805F9B34FB

※青字部分が短縮型 UUID

Manufacturer Name String	製造ベンダ名を取得します。 WRS-100 では、「AIMEX」が読み出されます。
Serial Number String	シリアル番号を取得します。 スキャナのシリアル番号が UTF-8 の文字列で読み出されます。
Software Revision String	ソフトウェアのバージョン番号を取得します。 スキャナのファームウェアバージョンが UTF-8 の文字列で読み出されます。

Device Information Service に関する詳細は、Bluetooth SIG の Web サイトに記載されています。

https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.service.device_information.xml

④Battery Service

Battery Service は Buetooth SIG が定義する汎用サービスです。電池の残量を管理するために使用します。

Battery Service と、キャラクタリスティックの UUID は次の通りです。

サービス/キャラクタリスティック	UUID
Battery Service	0000180F-0000-1000-8000-00805F9B34FB
Battery Level	00002A19-0000-1000-8000-00805F9B34FB

※青字部分が短縮型 UUID

Battery Level	電池残量を 0～100(単位:%)の数値で取得することができます。 WRS-100 では、0, 20, 40, 60, 80, 100 の 6 段階の数値で表現されます。 Battery Level キャラクタリスティックは、notify によるイベント通知が可能です。

Battery Service に関する詳細は、Bluetooth SIG の Web サイトに記載されています。

https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.service.battery_service.xml

1-5. 接続フロー

BLE デバイスの通信接続フローは”アドバタイズ”という動作から始まります。「WRS-100 シリーズ」などの機器（ペリフェラル）は、電源を投入すると自身のデバイス名やデバイスタイプなどの情報を一定のインターバルで周囲にブロードキャスト発信します。この情報を載せたパケットをアドバタイズパケットと言います。

OS または OS 上で動作するプログラム（セントラル）は、このアドバタイズパケットをスキャンして、コネクション要求を受け付けているペリフェラルを探します。目的のペリフェラルが見つかったらコネクション要求を送信し、ペリフェラルから応答があればコネクションを確立します。

コネクション確立後、セントラルはペリフェラルが必要なサービスをサポートしているかどうかを UUID で確認し、サポートしていればサービスが提供する機能（WRS-100 の場合は読み取ったバーコードデータの取得）を利用できるようになります。

1-6. データフォーマット

WRS-100 からホスト機器へ送信するデータのフォーマットについて解説します。

工場出荷時の設定では、バーコードを読み取ったデータがそのままホスト機器に送信されます。

コマンドバーコードでオプション設定を行うことで、読み取りデータの前後にプリフィックスやサフィックス、電池残量、読み取ったシンボル種別を示す ID キャラクタを付加できます。

コマンドバーコードは、ユーザズマニュアルを参照してください。

データフォーマット

プリフィックス	電池残量	ID キャラクタ	読み取りデータ	サフィックス

プリフィックスとサフィックスは、シリアル通信モード時と HID 通信モード時でそれぞれ別個の設定となっています。また、両モードで選択肢が異なります。

電池残量と ID キャラクタの付加設定は、シリアル通信モード時と HID 通信モード時で共通設定となっています。

プリフィックス

なし、または STX のどちらかを選択します。

シリアル通信モード時と HID 通信モード時でそれぞれ設定できます。

電池残量

なし、または付加するのどちらかを選択します。

電池残量のおおよその目安を、0～100 までの 20 刻みの数字と、カンマ（,）で付加します。

100 が満充電、20 がローバッテリ、0 がバッテリエンプティです。

ID キャラクタ

なし、シンボルコード ID キャラクタを付加する、AIM コード ID キャラクタを付加する、の 3 つ

から選択します。

読み取ったシンボルの種別を示すキャラクタを、データ先頭に付加できます。

「シンボルコード ID キャラクタ」は、読み取ったシンボルの種別を示す、1Byte の英字です。

「AIM コード ID キャラクタ」は、読み取ったシンボルの種別を示す、3Byte の英数記号です。

ID キャラクタの詳細は、ユーザーズマニュアルの『ID キャラクタ』を参照してください。

読み取りデータ

読み取ったバーコードのデータです。必ず送信されます。

サフィックス

なし、または選択肢の中から最大 4 つまで付加できます。

付加する場合は、選択肢を読み取った順番でデータ付加されます。

順番を変更したいときは、一度サフィックスを「なし」に設定してから、再度順番に読み取ってください。

2. サンプルアプリケーションについて

2-1. シリアル通信サンプル

「WRS-100 シリーズ」では、第一章で説明した Nordic UART Service を使ってシリアル通信を行うためのサンプルアプリケーションプログラムを公開しています。

プログラムは対象となる OS 毎に以下の 3 種類を用意しています。

プログラム種別	型番
Windows 用サンプルアプリケーション	WRS-Sample-W
Android 用サンプルアプリケーション	WRS-Sample-A
iOS 用サンプルアプリケーション	WRS-Sample-i

次章以降で、各サンプルアプリケーションの詳細について解説します。

・サンプルプログラムは機能の実現方法を例示することを目的としており、エラー処理や例外処理は考慮されていません。業務プログラムを作成する際には、十分配慮して設計してください。
・サンプルプログラムはお客様の責任において、自由に改変してご使用いただいて構いませんが、当社は動作や得られる結果に関していかなる保証もいたしません。
・プログラムの内容および仕様に関しては、将来予告無しに変更することがあります。

2-2. その他の参考アプリ

サンプルアプリケーションの仕様は、Nordic 社が公開しているサンプルアプリケーションや各種 BLE ツール類を参考にしています。

Nordic 社製のオリジナルは以下のサイトから閲覧することができます。

【nRF UART】

BLE の検索・接続・切断と、Nordic UART Service によるデータ送受信のサンプル。

Android 用アプリ

https://play.google.com/store/apps/details?id=com.nordicsemi.nrfUARTv2&hl=ja

Android Studio 用プロジェクト

https://github.com/NordicSemiconductor/Android-nRF-UART

【nRF Toolbox】

BLE の検索・接続・切断と、各種サービスのサンプル。「Heart Rate Monitor (HRM)」メニュー内に、Battery Service を用いた電池残量取得機能あり。

Anrdroid 用アプリ

https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrftoolbox&hl=ja

メーカーサイト

https://www.nordicsemi.com/eng/Products/Nordic-mobile-Apps/nRF-Toolbox-App

【nRF Connect for Mobile】

BLE の検索・接続・切断と、接続されたペリフェラルのサービス解析等が可能なアプリ。

Android 用アプリ

https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=ja

メーカーサイト

https://www.nordicsemi.com/eng/Products/Nordic-mobile-Apps/nRF-Connect-for-mobile-previously-called-nRF-Master-Control-Panel

3. Windows サンプル

3-1. 環境条件

Windows 用サンプルプログラム「WRS-Sample-W」の動作環境および前提条件は下記の通りです。

OS	Windows10 以上
開発ツール	Microsoft Visual Studio Community 2015 Version 14.0.25431.01 Update 3
開発言語	Microsoft Visual C# 2015
アプリケーション形態	ユニバーサル Windows プラットフォーム (UWP) アプリ
Bluetooth アダプタ	アイ・オー・データ製　USB-BT40LE

・「WRS-Sample-W」は、デスクトップ Windows10 64/32bit 環境で動作確認を行っています。タブレットやスマートフォンでの動作確認は行っていません。

・Bluetooth アダプタは、Broadcom 製のドライバのみインストールし、Windows 標準のプロトコルスタックを使用しています。アダプタ付属のプロトコルスタックを使用した場合は動作しません。

・Windows では BLE デバイスと GATT 通信（Nordic UART Service 通信）を行うためには必ず事前にペアリング処理を行い、デバイスを登録する必要があります。BLE 機器とのペアリングレスでの接続は現在サポートされていません。

・「WRS-Sample-W」は Jeffrey Chen 氏の ”Blutooth GattDeviceDemo” および Microsoft 社が提供している ”Windows-universal-samples”の”BluetoothAdvertisement” を参考に作成しています。
これらのオリジナルを確認したい場合は、下記のサイトからダウンロードすることが可能です。

Bluetooth GattDeviceDemo

https://github.com/dream-365/winapp

BluetoothAdvertisement

https://github.com/Microsoft/Windows-universal-samples/tree/master/Samples/BluetoothAdvertisement

3-2. サイドローディング

「WRS-Sample-W」は、ユニバーサル Windows プラットフォーム（UWP）形式のアプリケーションです。VisualStudio2015 がインストールされた環境であれば、プロジェクトファイルを読み込むことで編集や実行が可能となります。

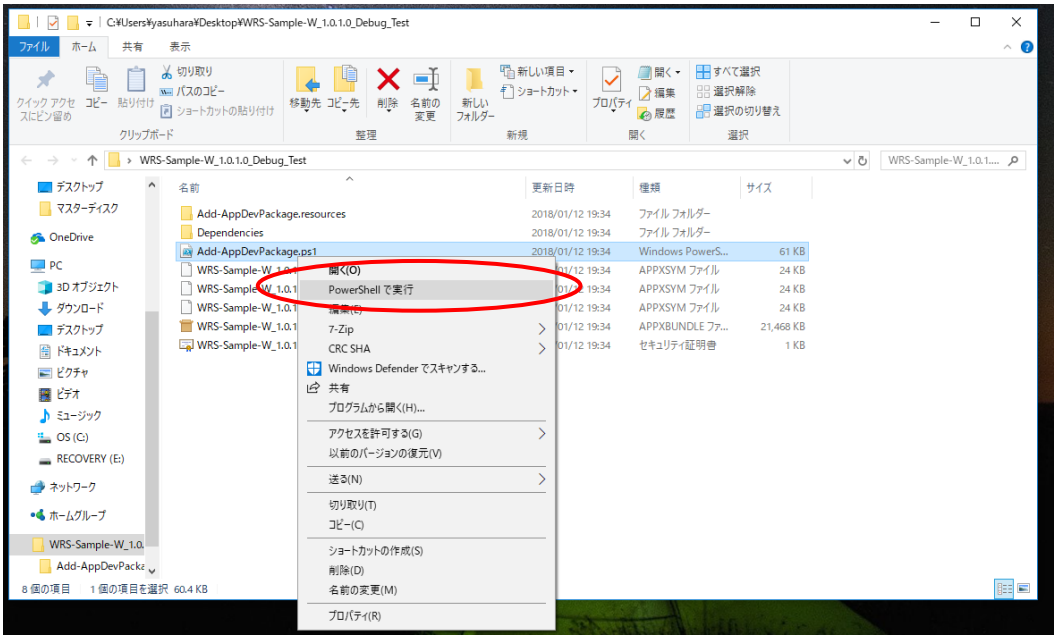
VisualStudio2015 がセットアップされていないターゲットで実行したい場合は、PowerShell を使ったサイドローディングという方法でインストールすることができます。

①サンプルプログラムのプロジェクトを展開したツリーに含まれる"AppPackages"というフォルダを、丸ごとインストールしたいターゲットの任意のフォルダにコピーします。"AppPackages"はプロジェクトルートから辿ると以下の場所にあります。

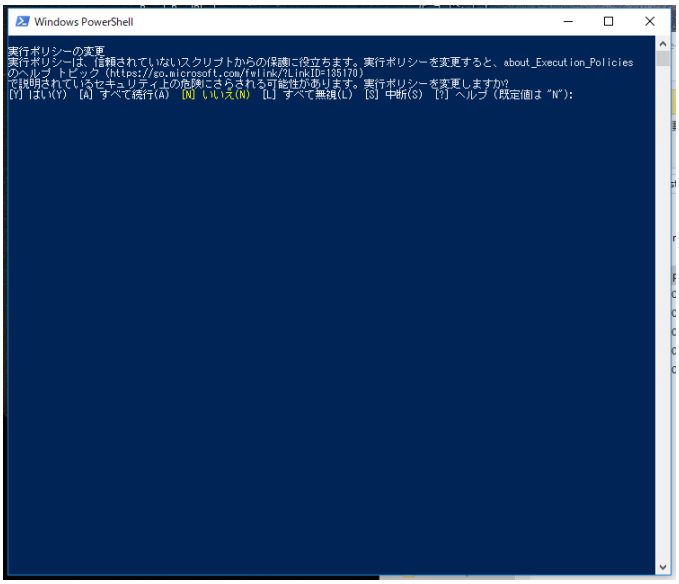
（場所の例）WRS-Sample-W-25¥WRS-Sample-W¥AppPackges

②ターゲットにコピーした"AppPackages"の 2 階層下にある PowerShell のスクリプトファイル"Add-AppDevPackage.ps1"を右クリックし、「PowerShell で実行」を選択します。

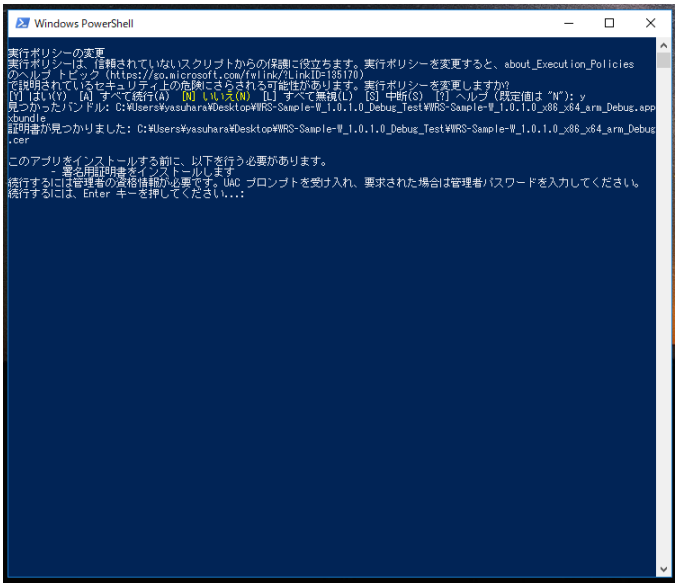
（場所の例）AppPackges¥WRS-Sample-W_1.0.1.0_Debug_Test¥Add-AppDevPackage.ps1



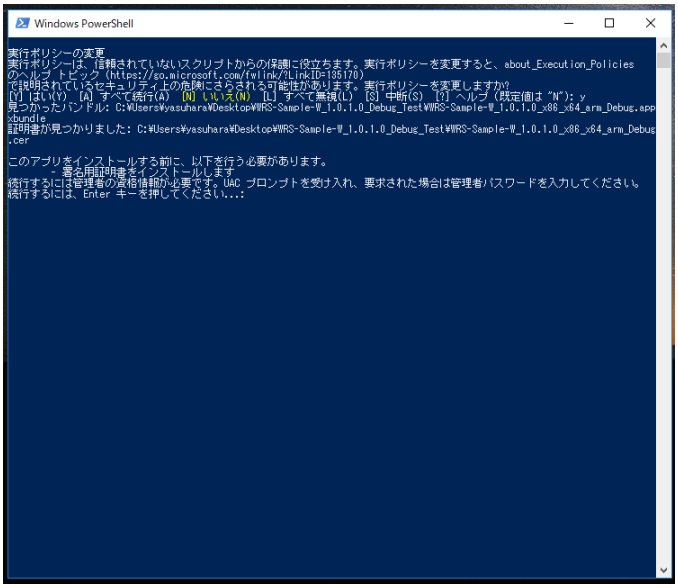
③PowerShell のウィンドウが起動し、"実行ポリシーの変更"が表示されますので、"Y"キーを押します。



④"署名用証明書"のインストール画面になりますので、"Enter"キーを押します。

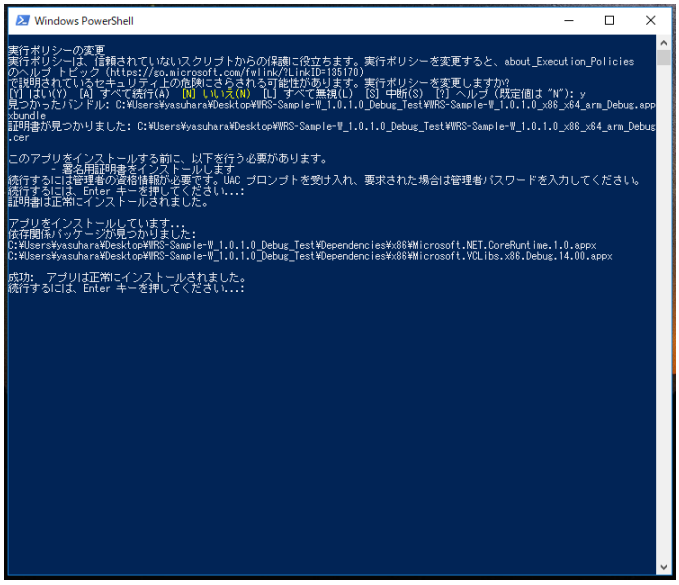


⑤証明書の発行元を信頼し、続行するか聞かれますので、"Y"キーを押します。



⑥証明書のインストール後、アプリケーションのインストールが開始されます。

インストールが完了すると以下の画面が表示されますので、"Enter"キーを押します。



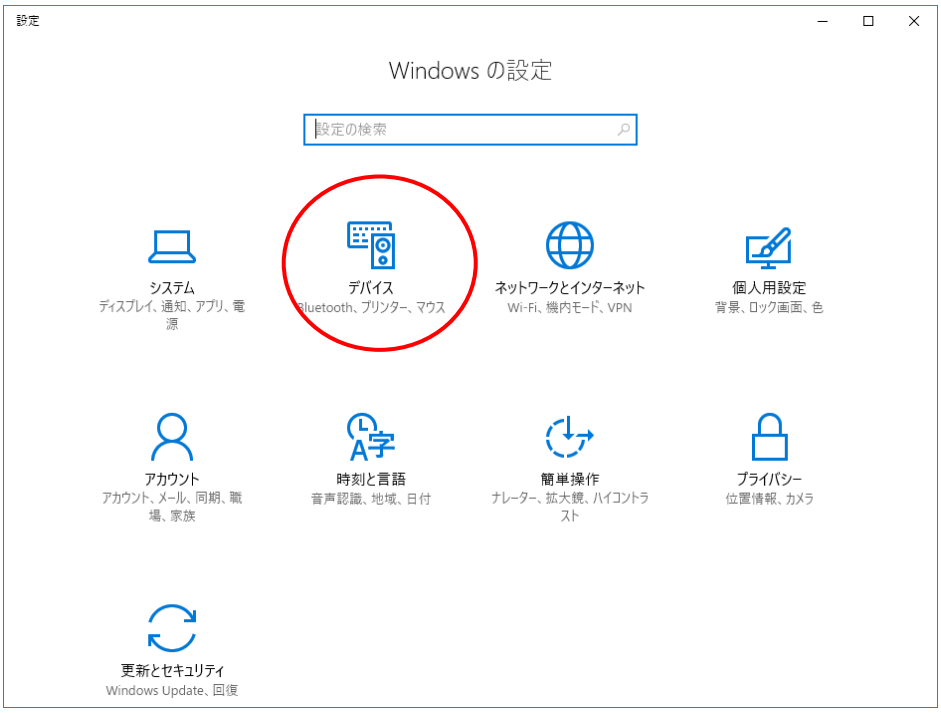
⑦アプリケーションに「WRS-Sample-W」が追加されていることを確認します。

3-3. 使用方法

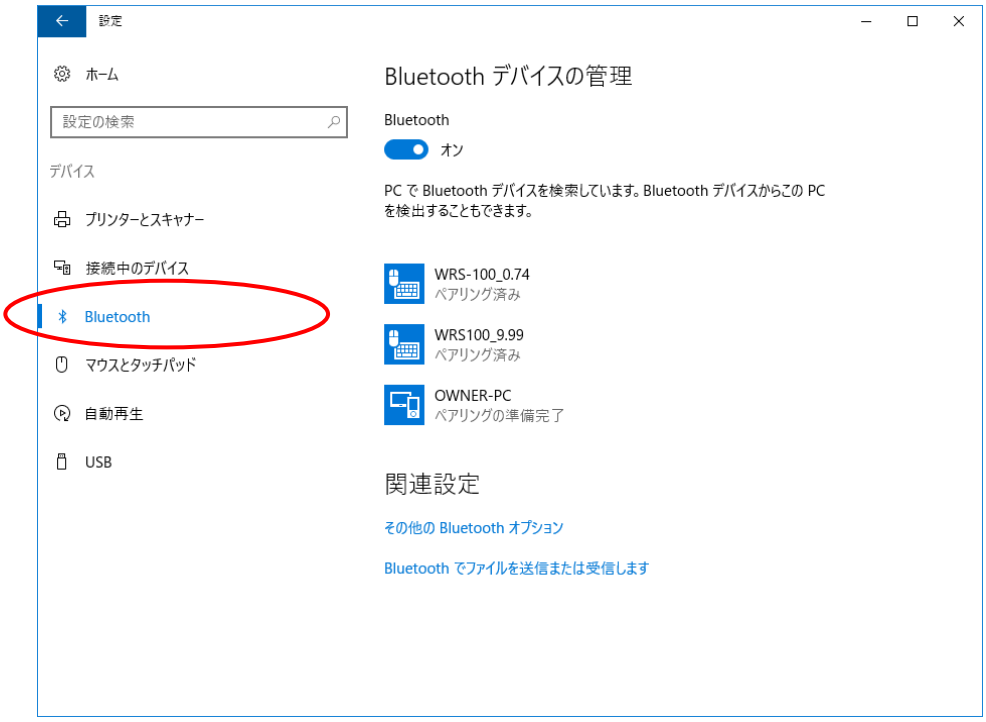
サンプルプログラムの使用方法を説明します。

まず、プログラムの起動前に Windows で通信対象となる WRS-100 とペアリングを行います。

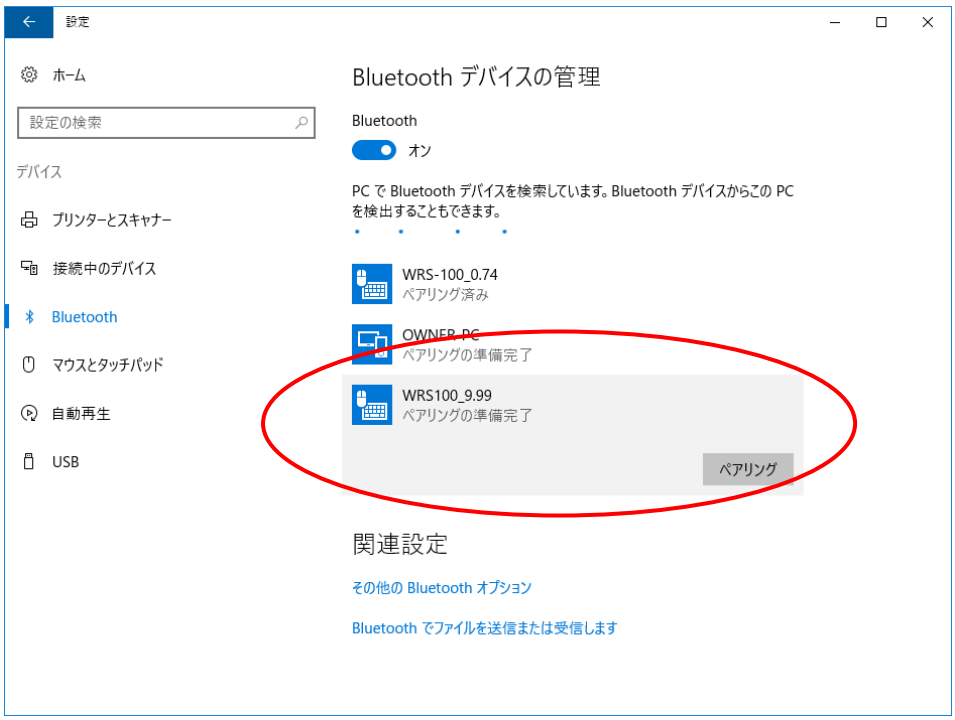
Windows10 のスタートメニューから「設定」→「デバイス」を開きます。



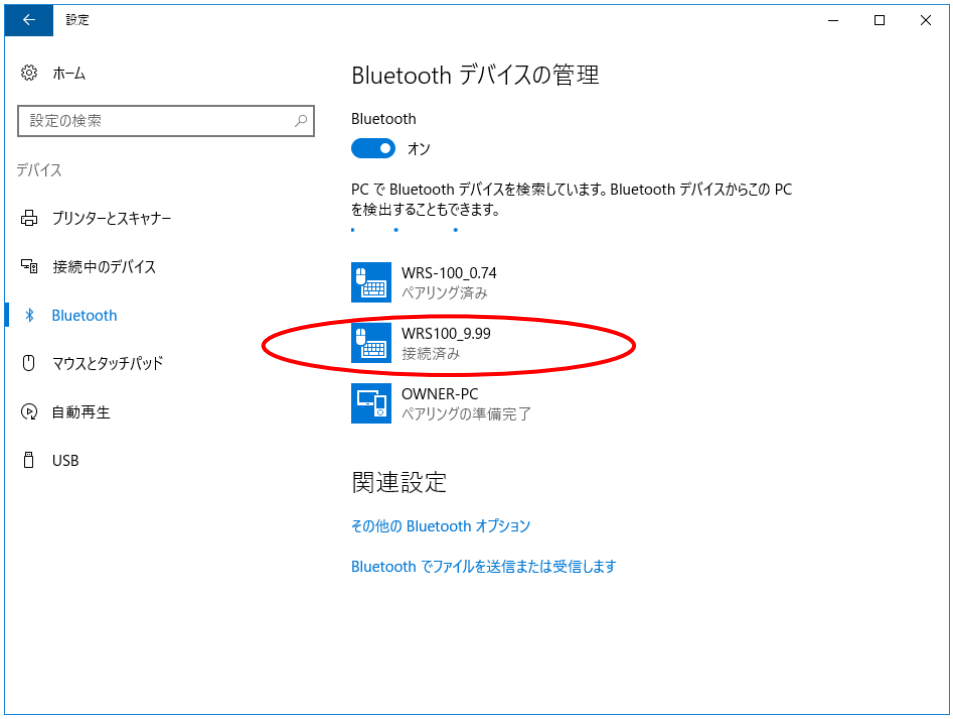
「Bluetooth デバイスの管理」を選択します。



WRS-100 を”シリアルモード”の設定で起動するとこの画面にリスト表示されますので、クリックで選択して「ペアリング」ボタンを押します。

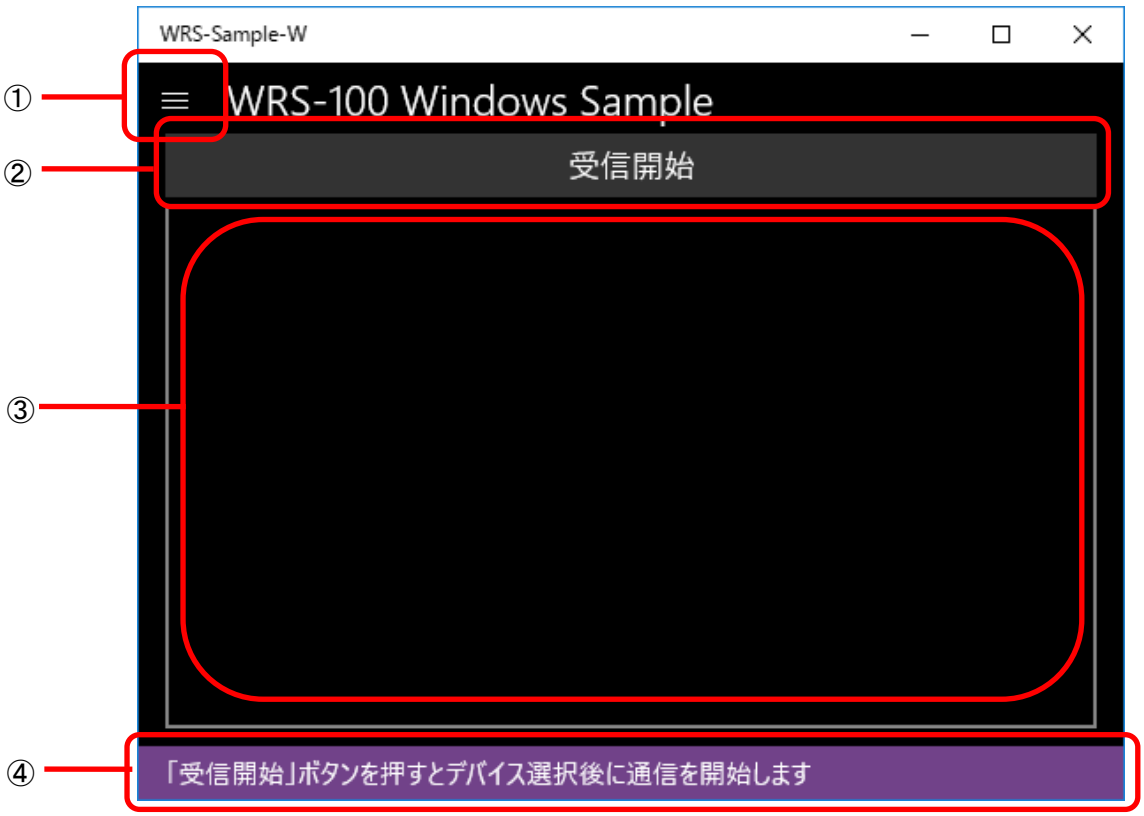


しばらくして”接続済み”または”ペアリング済み”の表示に変わればペアリングは完了です。



ペアリングが完了したら、サンプルプログラム「WRS-Sample-W」を起動します。Visual Studio でプロジェクトを開き、”デバッグ”→”デバッグの開始”または”デバッグ無しで開始”で実行するのが簡単です。Windows ストア経由の配布や、サイドローディングで実行ファイルを配布することも可能ですが、ここでは説明しません。

メインページの構成は次の通りです。



① ハンバーガーメニュー

プログラムで実行する処理を選択します。

次の 3 つの処理が選択できます。

「デバイス検索」

「デバイス情報」

「Advertisement」

② 受信開始ボダン

Nordic UART Service の通信を開始/停止する際に押すボタンです。

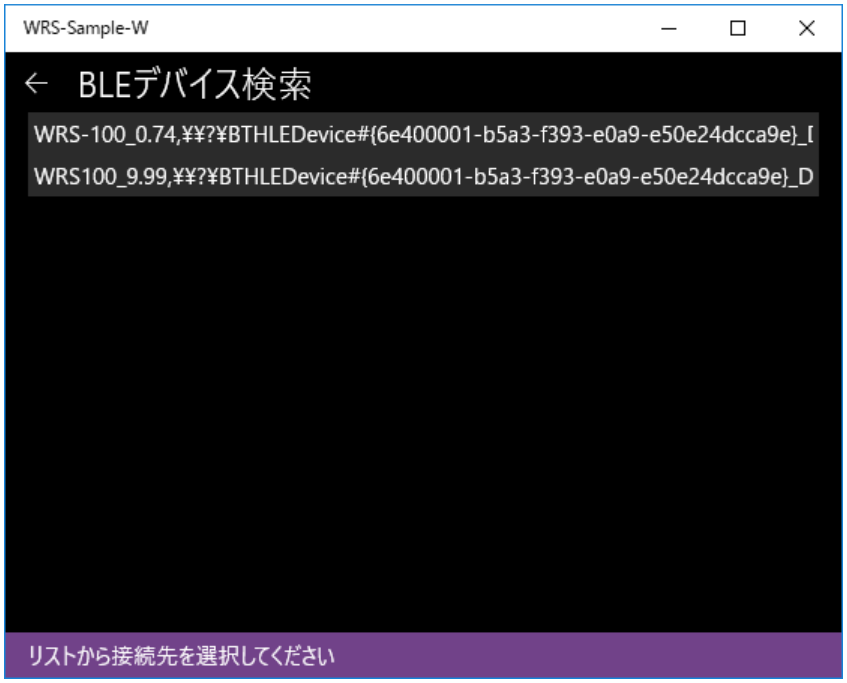
③ テキストボックス

デバイスのリストや、受信したデータを表示する領域です。

④ ステータス行

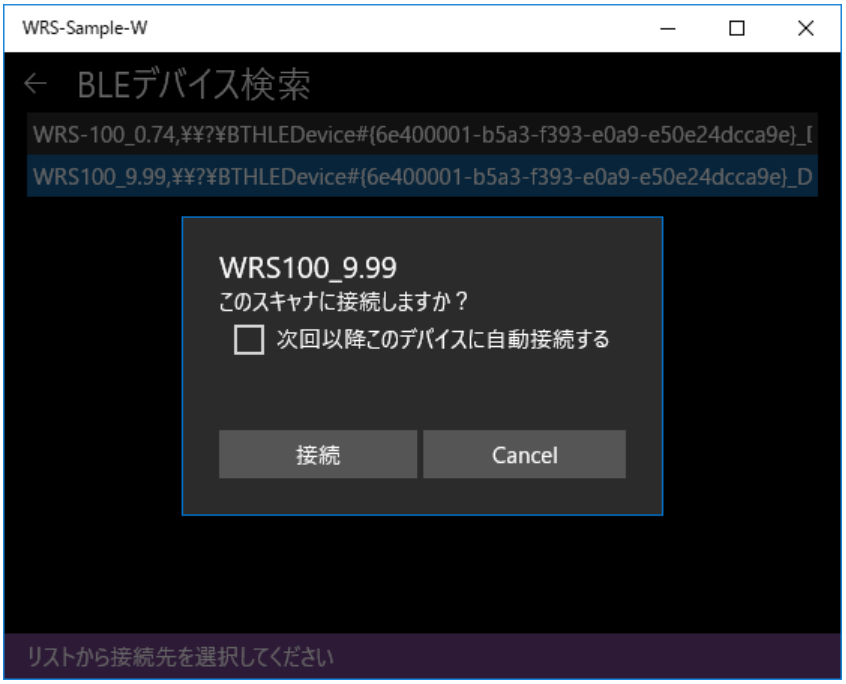
プログラムの実行ステータスやガイダンスを表示する領域です。

「受信開始」ボタンを押すとデバイス検索メニュー画面に推移し、Windows にペアリング登録済みのデバイスが表示されます。ここで表示されるデバイスの条件は、”Nordic UART Service”をサポートしており、かつ、ペアリングが完了したデバイスが対象となります。BLE キーボードや、他の BLE 機器は、サービス（クラシック Bluetooth で言えばプロファイル）が異なりますので、リスト表示されません。



クリックして接続対象を選択すると、接続ダイアログメッセージが表示されます。

WRS-100 を複数台お持ちの場合は、リストの最初の項目として表示される”デバイス名”で識別してください。

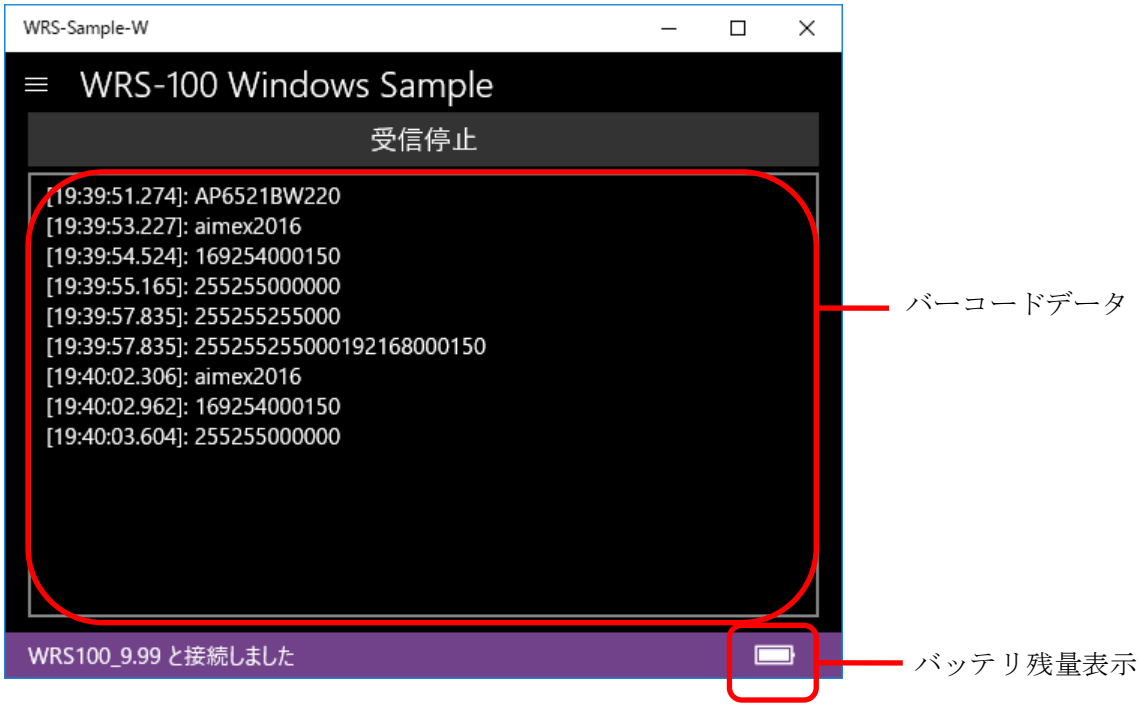


“次回以降このデバイスに自動接続する”にチェックを入れると、サービス ID がアプリケーションローカル記憶域に記憶され、次回以降自動的に接続することができます。チェックを入れない場合はテンポラリな接続として、都度接続相手を選択する動作となります。

「接続」ボタンを押すと、対象デバイスとコネクションが確立され、Nordic UART Service の利用が開始されます。

「Cancel」ボタンを押すと、処理はキャンセルされ、デバイス検索メニュー画面に戻ります。

接続を選択した場合、画面はメインページに推移し、以降読み取ったバーコードデータが画面のテキストボックスに表示されるようになります。同時に Battery Information Service の接続も行われ、デバイスのバッテリ残量がステータス行の右側にアイコン表示されます。残量は 0, 20, 40, 60, 80, 100% の 6 段階のアイコンで表現されます。



「受信停止」ボタンを押すとコネクションが切断され、初期画面に戻ります。再度「受信開始」ボタンを押すと同じデバイスと接続されます。

別のデバイスに接続し直す場合は、”デバイス検索”メニューから再度接続先の選択をやり直してください。

接続時にステータス行に”デバイスとのペアリング情報を一旦解除し、再度ペアリングしてください”というメッセージが表示された場合は、Windows とデバイスのペアリング情報がアンマッチとなっている状態です。(デバイス側でボンディング情報が消去された等)

ペアリングをやり直してから接続を試みてください。

3-4. ファイル構成

「WRS-Sample-W」は以下のファイルで構成されています。

Package.appxmanifest	マニフェストファイル
App.xaml	アプリケーションのスタートアップページ定義
App.xaml.cs	App.xml のコードビハインド
MainPage.xaml	「メインページ」の定義
MainPage.xaml.cs	MainPage.xaml のコードビハインド
DeviceList.xaml	「デバイス検索」ページの定義
DeviceList.xaml.cs	DeviceList.xaml のコードビハインド
DeviceInformation.xaml	「デバイス情報」ページの定義
DeviceInformation.xaml.cs	DeviceInformation.xaml のコードビハインド
Advertisement.xaml	「Advertisement」ページの定義
Advertisement.xaml.cs	Advertisement.xaml のコードビハインド

マニフェストファイルには、以下の Bluetooth の使用宣言が必要です。

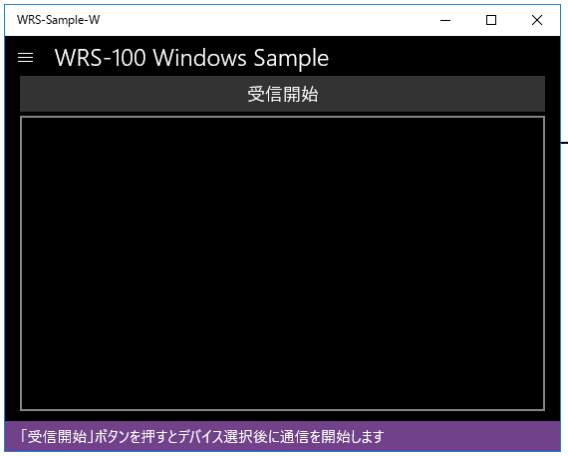
```
<Capabilities>
    <Capability Name="internetClient" />
    <DeviceCapability Name="bluetooth" />
    <DeviceCapability Name="bluetooth.genericAttributeProfile" />
</Capabilities>
```

3-5. ページ体系

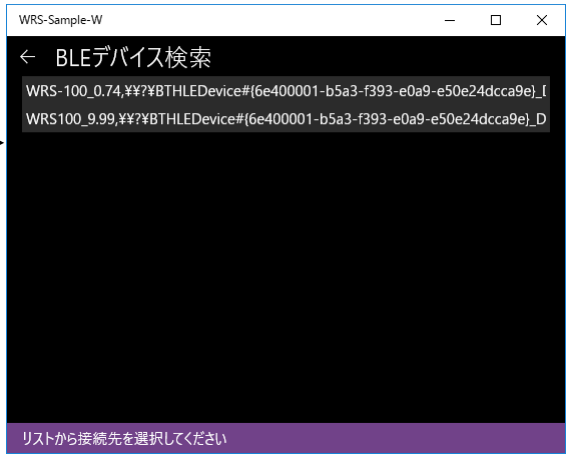
「WRS-Sample-W」は 4 つのページで構成されています。

メインページから各機能ページへは、ハンバーガーメニューから推移することができます。

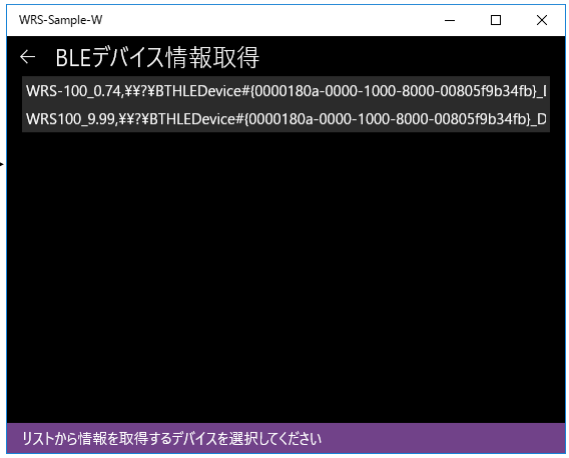
また、各機能ページからメインページへ戻る場合は、画面左上の「戻る」ボタンをクリックします。



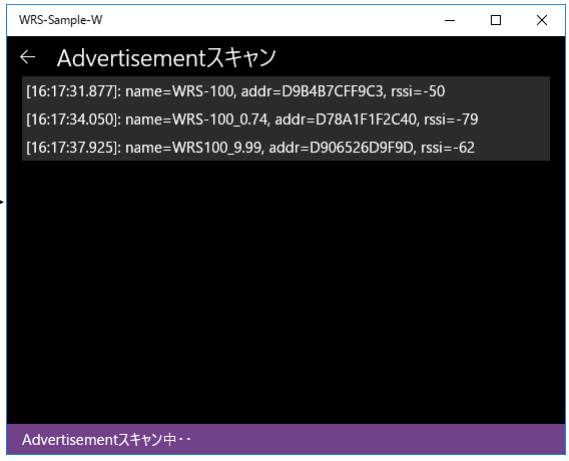
メイン　ページ



デバイス検索　ページ



デバイス情報取得　ページ



Advertisement　ページ

3-6. BLE 機器の検索

BLE 機器と GATT 通信を行う場合は、まず、DeviceInformation.FindAllAsync でデバイス・サービスを検索します。

検索する条件として GattDeviceService.GetDeviceSelectorFromUuid を指定することで、目的のサービス種別を絞って列挙します。WRS-100 の場合は、Nordic UART Service でシリアル通信を行いますので、この UUID を指定します。DeviceInformation.FindAllAsync でサービスを検索する際は、事前にペアリングが完了し、Windows に登録されたデバイスが対象となります。このタイミングではデバイスが通信可能状態にあるかどうかは無関係です。

Windows では、GATT 通信を行うためには事前にペアリングが完了している必要があります。

3-7. UART サービスでの通信

その後、取得したデバイス・サービスからサービス ID を取得し、その ID をもとに GattDeviceService.FromIdAsync でサービスを扱う GattDeviceService のインスタンスを取得します。また、GattDeviceService.GetCharacteristics でキャラクタリスティックを扱う GattCharacteristic のインスタンスを取得します。Nordic UART Service の場合、TX Characteristic がデバイス→ホスト機器へのデータ転送、RX Characteristic がホスト機器→デバイスへのデータ転送となります。WRS-100 では読み取ったバーコードデータを一方通行でホスト機器に送信するため、TX Characteristic を使って通信を行います。

TX Characteristic では Notify（イベント通知）機能をサポートしています。GattCharacteristic.WriteClientCharacteristicConfigurationDescriptorAsync で Notify 機能の使用をイネーブルにします。Notify がサポートされるキャラクタリスティックの場合、ValueChanged イベントにイベントハンドラを登録しておくことで、Notify 受信時にイベントハンドラが実行されるようになります。「WRS-Sample-W」では、ValueChanged イベント内でバーコードデータの受信、複数パケットの場合のデータの結合、テキストボックスへの表示処理を行っています。

3-8. その他のサービスの利用

WRS-100 は GATT サービスとして、Nordic UART Service の他、Battery Service、Device Information Service をサポートしてます。これらは Bluetooth SIG で規定された標準的なサービスです。

Battery Service、Device Information サービスを使用する場合も、Nordic UART サービスを使用する場合と同様にデバイス・サービスを検索し、キャラクタリスティックを指定して通信する流れとなります。

「WRS-Sample-W」では、メインページでバッテリ残量アイコンを表示するために Battery Service を使用しています。また、デバイス情報ページでデバイスの情報を読み出す際に Device Information Service を使用しています。

3-9. Advertisement メニュー

Windows10 では、Bluetooth Advertisement (BLE ビーコン) がサポートされるようになりました。

WRS-100 は電源を ON すると、一定間隔で Advertisement パケットを不特定多数の機器に対して発信(ブロード キャスト)します。このパケットは、BluetoothLEAdvertisementWatcher クラスを使って受信することができます。Received イベントにイベントハンドラを登録しておくことで、Advertisement パケット 受信毎にイベントハンドラが実行されるようになります。

「WRS-Sample-W」では、Advertisement ページで Advertisement パケットの受信を行い、画面にリスト表示しています。Advertisement パケットは実際には連続して受信されますが、同じ発信元の場合は重複して表示しないようチェックをかけています。

Android や iOS の場合、Adivertisement を受信して、そのままペアリングレスで GATT 接続することができますが、現状は Windows ではペアリングが必須となっています。

このため、「WRS-Sample-W」では Advertisement は参考扱いです。Advertisement パケットを受信し、表示するのみであり、実際の通信には使用していません。

4. Android サンプル

4-1. 環境条件

Android 用サンプルプログラム「WRS-Sample-A」の動作環境および前提条件は下記の通りです。

OS	Android Version 4.4 (KitKat) 以上
開発ツール	Android Studio 2.2.2 Build 145.3360264
SDK Version	compileSdkVersion : 23 minSdkVersion : 19
開発言語	Java version 1.8.0_111

・「WRS-Sample-A」は、CipherLab RS30、Sony Mobile Communications Xperia(TM) Z1、EPSON MOVERIO BT-300 等で動作確認を行っています。古い機種では BLE がサポートされていないことがあります。

・Android では BLE デバイスとペアリング無しで GATT 通信（Nordic UART Service 通信）を行うことができます。

・機器によって搭載している Bluetooth チップやデバイスドライバの実装が異なるため、接続が切れやすい、通信が遅延するなどの症状が出ることがあります。システム導入時には十分な検証を行ってください。

・「WRS-Sample-A」は Nordic Semiconductor 社が提供している ”nRF UART” をベースにカスタマイズを加えて作成しています。

オリジナルを確認したい場合は、下記のサイトからダウンロードすることが可能です。

nRF UART（実行ファイル）

https://play.google.com/store/apps/details?id=com.nordicsemi.nrfUARTv2&hl=ja

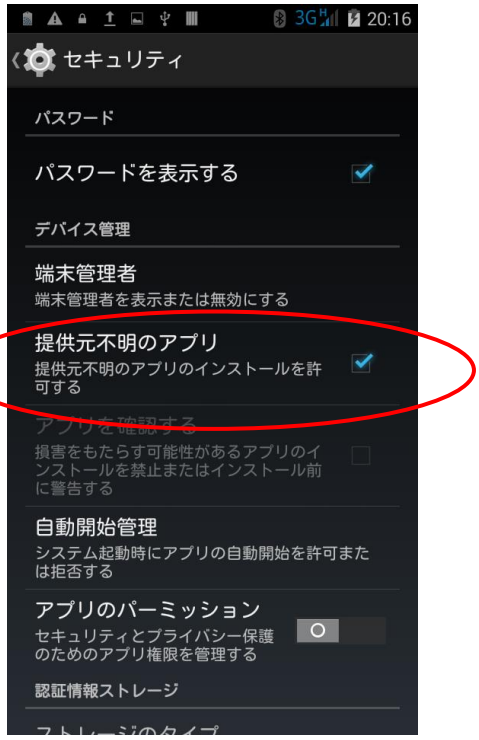
Android-nRF-UART（ソース）

https://github.com/NordicSemiconductor/Android-nRF-UART

4-2. 使用方法

サンプルプログラムの使用方法を説明します。

まず、プログラムを実行するターゲット端末で「設定」→「セキュリティ」→「提供元不明のアプリのインストールを許可する」にチェックを入れて有効化してください。この項目が無効になっていると Android Market 経由以外の方法では apk ファイルをインストールできません。



次に「設定」→「Bluetooth」を ON にし、Bluetooth 機能を有効化してください。

※「WRS-Sample-A」を使ったシリアル通信を行う場合、事前にペアリングする必要はありません。

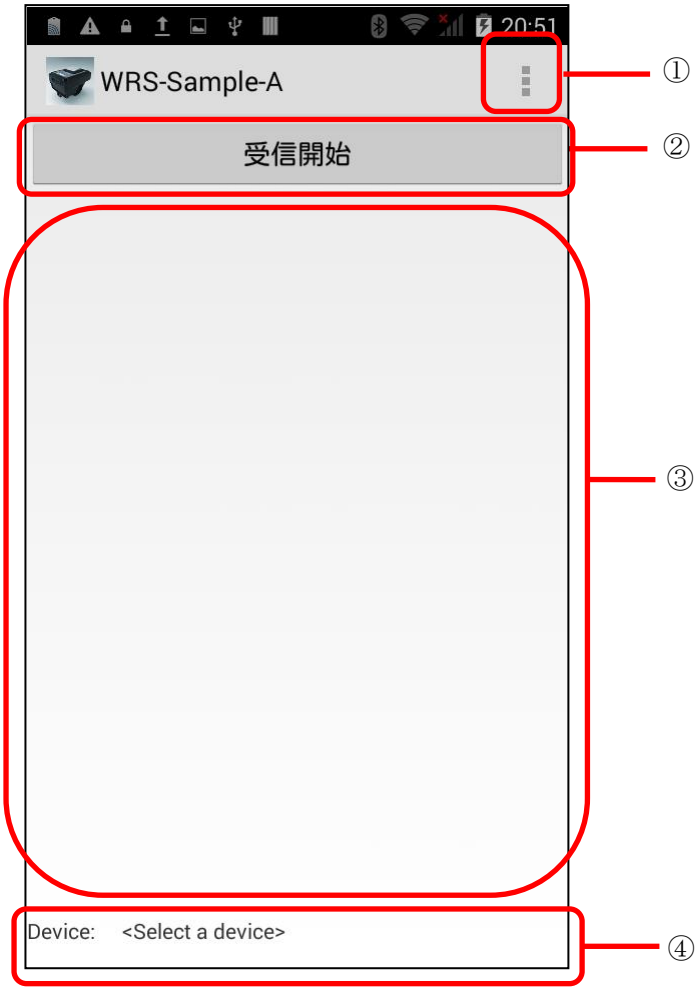
続いて ¥app¥build¥outputs¥apk フォルダにある apk ファイルを、端末に以下のいずれかの方法で転送します。

・Gmail などの添付ファイルで端末に転送する。
・PC と端末を USB で接続し、リムーバブルディスクとしてマウントして apk ファイルをコピーする。
・SD カードなどのメディアを経由して転送する。
・Andoroid Studio の「実行」または「Debug」機能を使って直接実行する。

以下は apk ファイル単独でインストールする場合の説明です。
apk ファイルを端末に転送したら、ファイラ―アプリなどからファイルをタップして展開します。
次の画面が表示されたら”インストール”ボタンをタップしてください。

インストールが完了したら、サンプルプログラム「WRS-Sample-A」を起動します。

メイン画面の構成は次の通りです。



① ハンバーガーメニュー

プログラムで実行する処理を選択します。

次の 3 つの処理が選択できます。

「デバイス検索」

「自動再接続」

「デバイス情報」

② 受信開始ボダン

Nordic UART Service の通信を開始/停止する際に押すボタンです。

③ テキストウィンドウ

受信したデータを表示する領域です。

④ ステータス行

デバイスとの接続状態を表示する領域です。

「受信開始」ボタンを押すとデバイス検索アクティビティ画面に推移し、Advertisement パケットのスキャンを行います。検出された BLE デバイスが画面にリスト表示されます。

接続対象をタップして選択すると、接続ダイアログメッセージが表示されます。

WRS-100 を複数台お持ちの場合は”デバイス名”で識別してください。

“次回以降このデバイスに自動接続する”にチェックを入れると、接続先情報がシェアードプリファレンスに記憶され、次回以降自動的に接続することができます。チェックを入れない場合はテンポラリな接続として、都度接続相手を選択する動作となります。

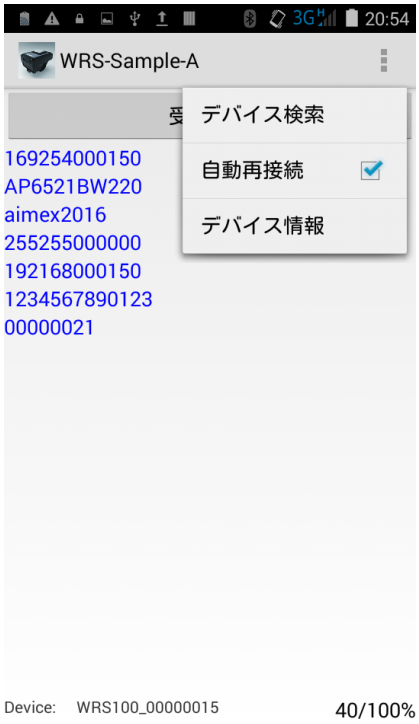
「接続」ボタンを押すと、対象デバイスとコネクションが確立され、Nordic UART Service の利用が開始されます。

「Cancel」ボタンを押すと、処理はキャンセルされ、デバイス検索画面に戻ります。

接続を選択した場合、画面はメインアクティビティに推移し、以降読み取ったバーコードデータがテキストウィンドウに表示されるようになります。同時に Battery Information Service の接続も行われ、デバイスのバッテリ残量がステータス行の右側にテキスト表示されます。残量は”XX/100%”の形式で、XX の部分は 0, 20, 40, 60, 80, 100 の 6 段階で表現されます。

「受信停止」ボタンを押すとコネクションが切断され、初期画面に戻ります。再度「受信開始」ボタンを押すと”自動接続”にチェックが入っている場合は同じデバイスと接続されます。テンポラリな接続の場合はデバイス検索画面からやり直しとなります。

”自動接続”の選択は、デバイス接続後にハンバーガーメニューの”自動再接続”のチェック状態を操作することで変更することもできます。



また、ハンバーガーメニューから”デバイス情報”を選択すると、接続中のスキャナのデバイス情報を読み出し、画面に表示します。

4-3. ファイル構成

「WRS-Sample-A」は以下のファイルで構成されています。

AndroidManifest.xml	マニフェストファイル
MainActivity.java	アプリケーションのメインアクティビティクラス
DeviceListActivity.java	デバイス検索アクティビティクラス
UartService.java	Nordic UART Service 制御クラス
main.xml	メインアクティビティのビュー定義
device_list.xml	デバイスリストのビュー定義
device_element.xml	デバイスリストのデバイス毎ビュー定義
com_dialog.xml	接続ダイアログのビュー定義
dis_values.xml	デバイス情報のビュー定義
Info_item.xml	デバイス情報リストの項目毎ビュー定義
title_bar.xml	タイトルバーのビュー定義
Message_detail.xml	バーコード表示リストのビュー定義

マニフェストファイルには、以下の Bluetooth の使用許可宣言が必要です。

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

また、Android6.0 (Mashmallow)以降は上記に加えて位置情報のパーミッションが必要となります。

```
<uses-permission android:name="android.permission. ACCESS_COARSE_LOCATION" />
または
<uses-permission android:name="android.permission. ACCESS_FINE_LOCATION" />
```

4-4. アクティビティ構成

「WRS-Sample-A」は 2 つのアクティビティで構成されています。

メインアクティビティからデバイス検索アクティビティへは、受信開始ボタンをタップするか、ハンバーガーメニューから”デバイス検索”を選択することで推移します。

メインアクティビティ　　デバイス検索アクティビティ

ハンバーガーメニューの”自動再接続”チェック処理、および、”デバイス情報”の取得・表示処理は、メインアクティビティ内で行っています。

UartService.java は、Nordic UART Service、Device Information Service、Battery Service などの GATT サービスを利用するためのライブラリである UartService クラスを定義しています。

4-5. BLE 機器の検索

まず、bluetoothManager.getAdapter()で BluetoothAdaper を取得します。端末の Blueooth 設定が有効かどうかは BluetoothAdapter.isEnabled()で判定できます。

BLE 機器の検索は BluetoothAdapter.startLeScan(BluetoothAdapter.LeScanCallback callback)で開始します。startLeScan()のパラメータにはスキャン結果を処理するコールバックの実装を指定します。

BLE 機器のスキャンにはタイムアウトが必要ですが、startLeScan() にはタイムアウトの設定がありません。このため Handler を用いてタイムアウト処理を実現しています。「WRS-Sample-A」では 180 秒のタイムアウト時間を設けています。タイムアウト後に"Scan"ボタンをタップすると再度スキャンを開始します。

スキャンにより BLE 機器が見つかり次第、順次 LeScanCallback.onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord) が呼ばれ、画面にリスト表示しています。Advertisement パケットは、一つの機器から連続で発行されるため、複数回 onLeScan イベントが発生することになります。同じデバイスを重複して表示しないように、BD アドレスでチェックをかけています。

検索で見つかったデバイスリストのうち、接続したい相手をタップすると接続確認ダイアログを表示します。ここで"接続"を選択すると intent へ putExtra()で接続先デバイスの BD アドレスをセットし、メインアクティビティへ引き渡します。"Cancel"を選択した場合はダイアログを消去し、デバイスリスト表示画面へ戻ります。

4-6. デバイスの接続

対象の機器が決定されたら、BD アドレスをキーにして BluetoothDevice.connectGatt(Context context, boolean autoConnect, BluetoothGattCallback callback) で GATT 接続します。autoConnect を false で呼び出すとすぐに機器への接続を開始します。接続完了や機器からのデータ受信などの GATT サービスは BluetoothGattCallback で処理します。

connectGatt()の結果として接続が確立すると、まず onConnectionStateChanged()がコールバックされます。ここでパラメータで渡された newState が BluetoothProfile.STATE_CONNECTED であれば正常に接続できたことを表しています。

接続の確立後は、次のステップとして BluetoothGatt.discoverServices() を実行し、サービスを検索します。サービスの検索が完了すると、onServicesDiscovered()がコールバックされます。

「WRS-Sample-A」では、これら GATT サービスのコールバック一次処理は UartService クラス内で行っており、メインアクティビティへ Broadcast でイベントの発生を通知しています。

4-7. UART サービスでの通信

onServicesDiscovered が呼ばれた後は、メインアクティビティから Nordic UART Service の使用開始を指示しています。

Nordic UART Service の場合、TX Characteristic がデバイス→ホスト機器へのデータ転送、RX Characteristic がホスト機器→デバイスへのデータ転送となります。WRS-100 では読み取ったバーコードデータを一方通行でホスト機器に送信するため、TX Characteristic を使って通信を行います。

TX Characteristic では Notify（イベント通知）機能をサポートしています。

BluetoothGatt.getService および BluetoothGatt. getCharacteristic で対象のサービスと Characteristic を取得できていることを確認したら、Characteristic に対して Notification の受信要求を設定します。BluetoothGatt.setCharacteristicNotification で Nortification をイネーブルにし、対象の Characteristic の Descriptor に対して BluetoothGattDescriptor. ENABLE_NOTIFICATION_VALUE を設定します。これら一連の処理を UartService クラスの enableTXNotification()で行っています。

Notification が有効化された後は、バーコード読み取り毎に onCharacteristicChanged()がコールバックされます。「WRS-Sample-A」では OnCharactaristicChanged()が呼ばれると ACTION_DATA_AVAILABLE イベントが Broadcast され、メインアクティビティ内の BroadcastReceiver でデータの取得、複数パケットの場合のデータの結合、テキストウィンドウへの表示処理を行っています。また表示の際にはビープ音とバイブレータで通知します。

4-8. その他のサービスの利用

WRS-100 は GATT サービスとして、Nordic UART Service の他、Battery Service、Device Information Service をサポートしてます。これらは Bluetooth SIG で規定された標準的なサービスです。

Battery Service、Device Information サービスを使用する場合も、Nordic UART サービスを使用する場合と同様にデバイス・サービスを検索し、キャラクタリスティックを指定して通信する流れとなります。

「WRS-Sample-A」では、ステータス行にバッテリ残量を表示するために Battery Service を使用しています。また、ハンバーガーメニューの”デバイス情報”でデバイスの情報を読み出す際に Device Information Service を使用しています。

Battery Service は Nordic UART Service と同様に Notification でデータを取得します。Device Information Service では、BluetoothGatt.readCharacteristic()を使って各キャラクタリスティックの読み出しを行っています。

Discriptor への書き込みや、Chaacteristic の読み出しは前の処理が完了したことを確認した後に次の処理を行わないと正常に受け付けられないため、ご注意ください。

5. iOS サンプル

5-1. 環境条件

iOS 用サンプルプログラム「WRS-Sample-i」の動作環境および前提条件は下記の通りです。

OS	iOS 8.0 以上
開発ツール	Xcode 8.2.1 (8C1002)
開発言語	Swift version 3.0.2

・「WRS-Sample-i」は Apple iPhone6 用および iPhone5 用の 2 種類があり、それぞれの LCD 解像度に合わせて画面をレイアウトしています。これら以外の機種では、画面表示が右上寄せで配置されますので、適時 Storyboard 等でレイアウトを変更してビルドしてください。

・「WRS-Sample-i」は、iPhone6（iOS 10.2.1)、iPhone5（iOS 10.1.1)、iPod Touch MD720J/A（iOS 9.2.1）で動作確認を行っています。上記以外の機種では、前述の通り画面表示がずれたり、一部機能が正しく動作しないことがあります。

・iOS では BLE デバイスとペアリング無しで GATT 通信（Nordic UART Service 通信）を行うことができます。

・機器の種類や OS のバージョンによっては搭載している Bluetooth チップやデバイスドライバの実装が異なるため、接続が切れやすい、通信が遅延するなどの症状が出ることがあります。システム導入時には十分な検証を行ってください。

・「WRS-Sample-i」は Nordic Semiconductor 社が提供している ”nRF Toolbox” をベースに、WRS-100 のシリアル通信に必要な機能のみを抜き出して作成しています。

オリジナルを確認したい場合は、下記のサイトからダウンロードすることが可能です。

IOS-nRF-Toolbox（実行ファイル）

https://itunes.apple.com/jp/app/nrf-toolbox/id820906058?mt=8

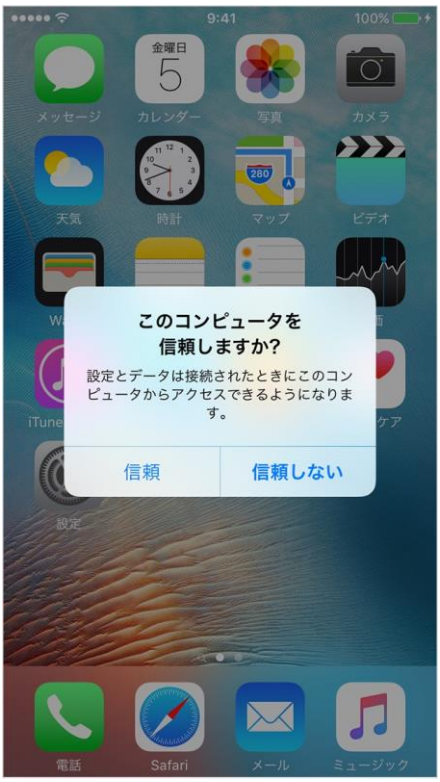
IOS-nRF-Toolbox（ソース）

https://github.com/NordicSemiconductor/IOS-nRF-Toolbox

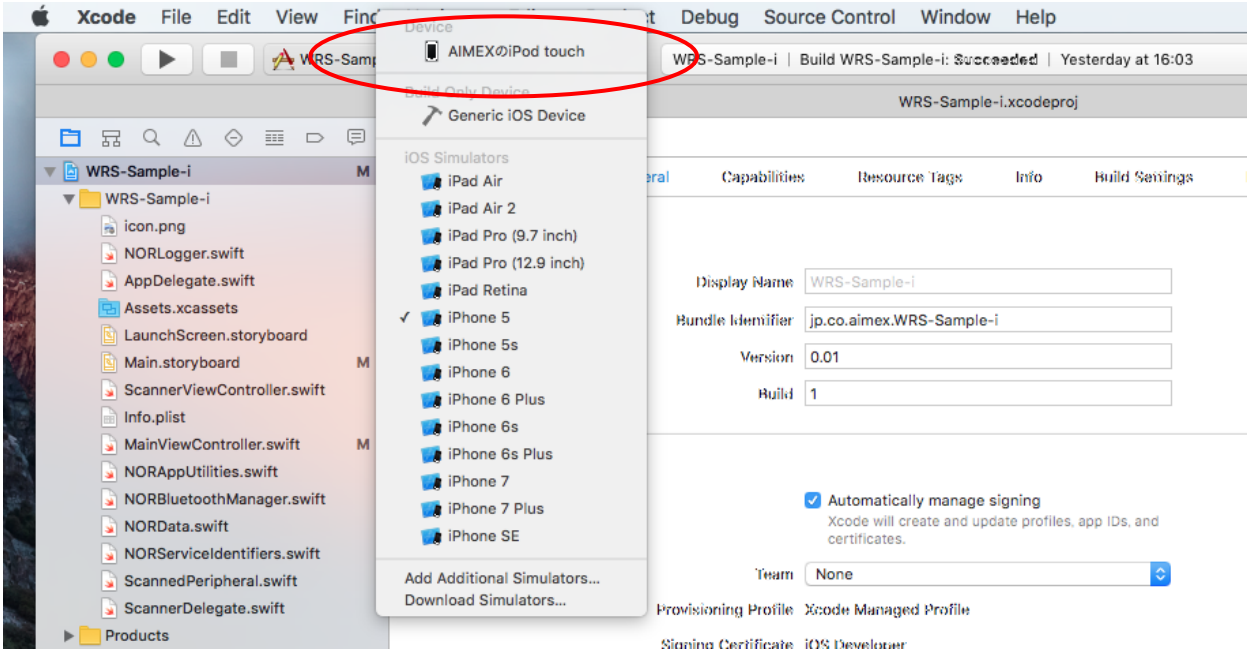
5-2. 実行準備

サンプルプログラムの実行準備手順を説明します。

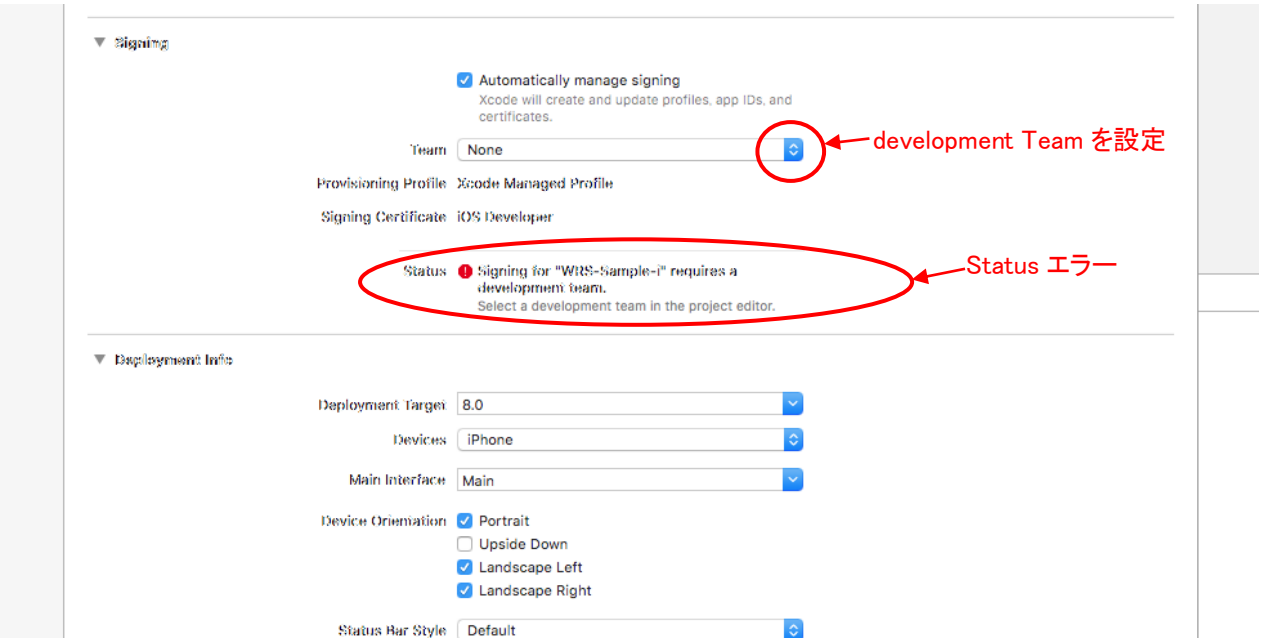
まず、Xcode で「WRS-Sample-i」のプロジェクトをオープンし、プログラムを実行するターゲット端末を USB で PC に接続します。すると、コンピュータを信頼するかどうかを確認するメッセージが表示されますので、"信頼"を選択します。



ターゲット端末との接続が認証されると、Xcode のシミュレータ選択画面で実機が選べるようになりますので、選択します。

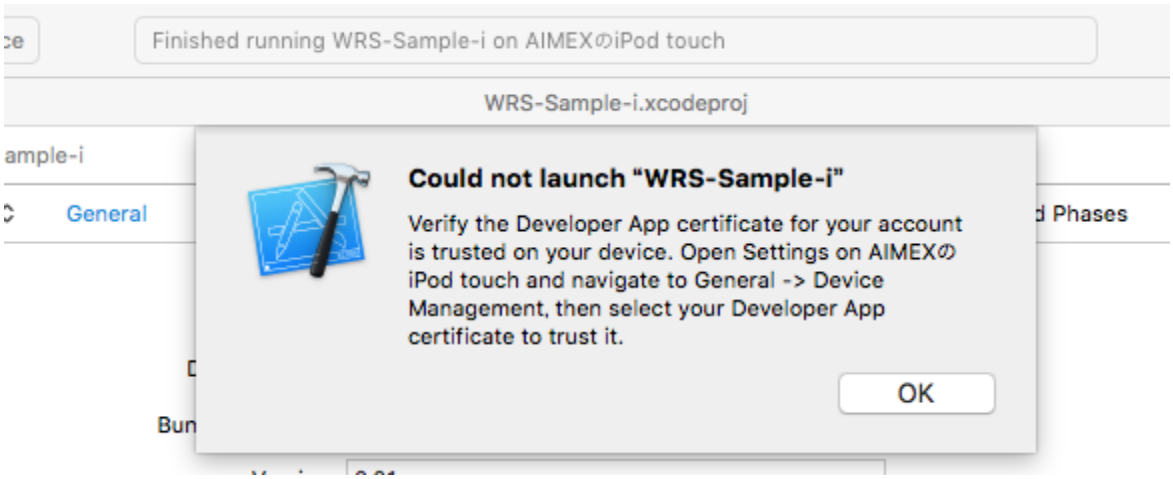


プロジェクトの「Signing」の項目で Status エラーが出ている場合は、development team を設定します。Apple ID に登録したアカウント名の Team が選択岐に現れますので、これを選択します。



Development team の設定後、Status エラーが消えたことを確認し、Xcode の「Run」ボタンを押します。

すると、シンボルの処理中～アプリケーションのインストール処理中であることを表すメッセージが表示されしばらく待たされます。その後、初回インストール時は以下のエラーが出力されます。



ターゲット端末の「設定」→「一般」→「デバイス管理」を開くと、Xcode で設定した development team の Apple ID が ″デベロッパ APP″の項目に表示されますので、タップします。

すると、そのデベロッパ ID に関する設定画面が開きますので、″〇〇を信頼″をタップします。

確認ダイアログがポップアップ表示されますので、"信頼"をタップします。

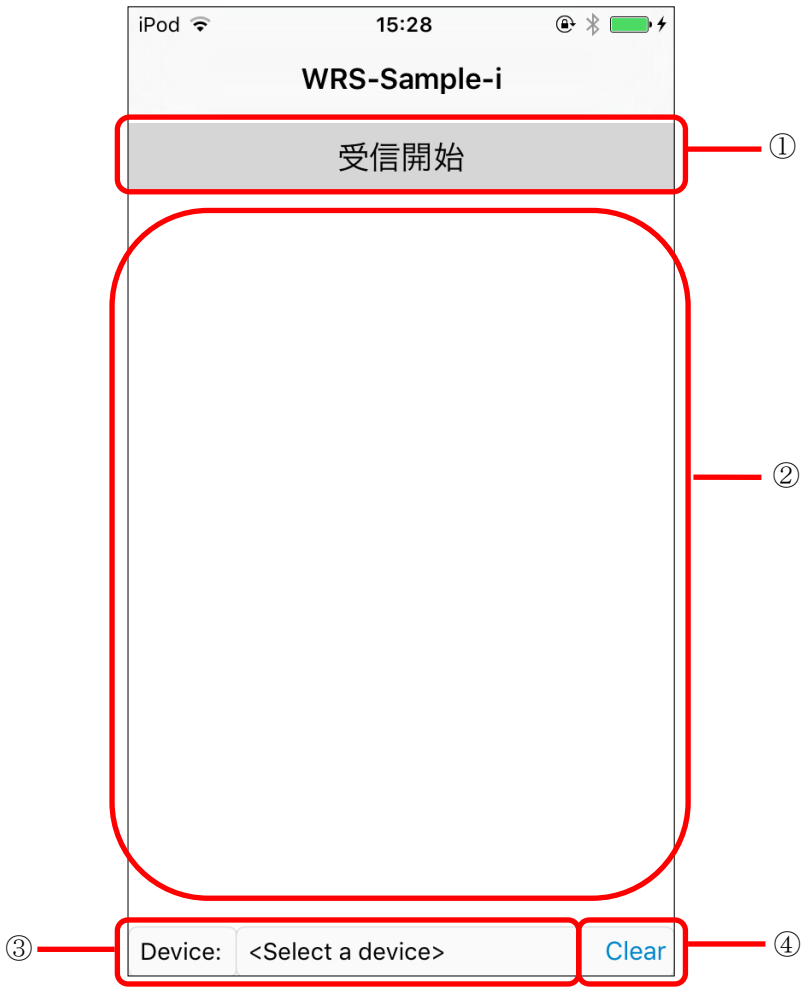
以下の画面になれば、アプリケーションプログラムのインストールは成功です。

デベロッパ ID の認証が完了したら、「設定」→「Bluetooth」を ON にし、Bluetooth 機能を有効化してください。

設定が完了したら、Xcode の「Run」ボタンを押すと、実機上でアプリケーションプログラムが起動できるようになります。

5-3. 使用方法

「WRS-Sample-i」のメイン画面の構成は次の通りです。



①受信開始ボダン

BLE 周辺デバイスをスキャンし、Nordic UART Service の通信を開始/停止する際に押すボタンです。

②テキストウィンドウ

受信したデータを表示する領域です。

③ステータス行

デバイスとの接続状態を表示する領域です。

③クリアボタン

テキストウィンドウをクリアするボタンです。

※iPhone6 用はテキストウィンドウは自動でスクロール表示されますが、iPhone5 用は手動スクロール動作となります。受信データがウィンドウ最下行まで到達した場合は、クリアボタンを押して一旦ウィンドウを消去すると確認が容易です。

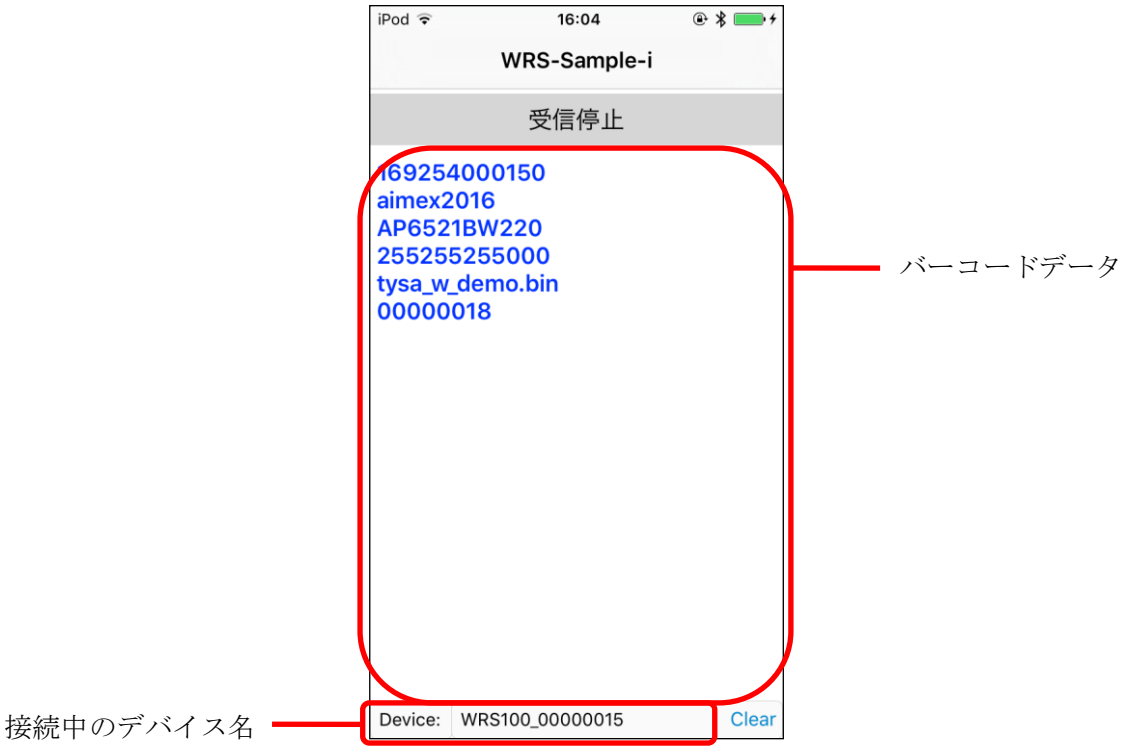
「受信開始」ボタンを押すとデバイス検索画面に推移し、Advertisement パケットのスキャンを行います。検出された BLE デバイスが画面にリスト表示されます。

接続対象をタップして選択すると、対象デバイスとコネクションが確立され、Nordic UART Service の利用が開始されます。「Cancel」ボタンを押すと、メイン画面に戻ります。

iOS では、セキュリティ確保のためデバイスアドレスは取得できません。WRS-100 を複数台お持ちの場合は”デバイス名”で識別してください。

接続デバイスを選択後、以下の画面が表示されることがあります。"OK"を押してください。

デバイスを選択して接続が完了すると、メイン画面に推移し、以降読み取ったバーコードデータがテキストウィンドウに表示されるようになります。



「受信停止」ボタンを押すとコネクションが切断され、初期画面に戻ります。再度「受信開始」ボタンを押すと、デバイス検索画面に推移します。

5-4. ファイル構成

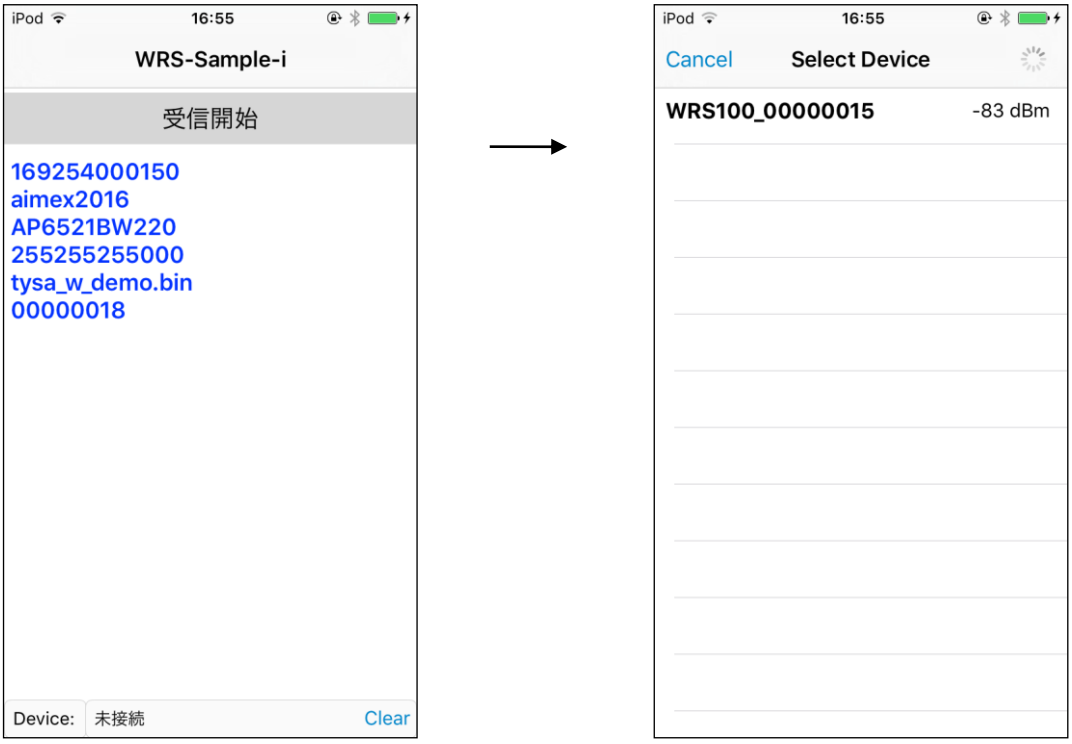
「WRS-Sample-i」は以下のファイルで構成されています。

WRS-Sample-i.xcodeprj	プロジェクトファイル
AppDelegate.swift	アプリ全体のライフタイムイベントを管理するためのクラス
MainViewController.swift	メイン画面のビューコントローラ
ScannerViewController.swift	デバイス検索画面のビューコントローラ
ScannerDelegate.swift	デバイス選択時のデリゲートプロトコル定義
ScannerPeripheral.swift	デバイスリスト表示に使用するライブラリクラス
NORBluetoothManager.swift	Bluetooth GATT 接続サービスを管理するクラス
NORAppUtilities.swift	ユーティリティライブラリクラス
NORServiceIdentifiers.swift	GATT サービスに使用する UUID の定義
NORData.swift	データ変換ライブラリクラス
NORLogger.swift	アプリケーションログ管理のプロトコル定義
LaunchScreen.storyboard	起動画面ストーリーボード
Main.storyboard	アプリケーション画面ストーリーボード

5-5. ViewController 構成

「WRS-Sample-i」は 2 つの UIViewController で構成されています。

メイン画面(MainVIewController)からデバイス検索画面(ScannerViewController)へは、受信開始ボタンをタップすることで推移します。



メイン VIewController　　デバイス検索 ViewController

各 ViewController の画面レイアウトは、ストーリーボードを使って定義しています。

NORBluetoothManager.swift は、Nordic UART Service などの GATT サービスを利用するためのライブラリである NORBluetoothManager クラスを定義しています。このファイルは Nordic Semiconductor 社が提供している ”nRF Toolbox” で使われているソースをそのまま流用していますが、メイン画面で TX Characteristic の notify データを受け取れるようデリゲードメソッドを一つ追加しています。

5-6. BLE 機器の検索

iOS で BLE 通信を行うためには、CoreBluetooth.framework を使用します。

スマートデバイス側がセントラル、WRS-100 がペリフェラルとして動作しますので、まず CBCentralManager を初期化し、Bluetooth 機能を有効化します。ここでパラメータとして Delegate をセットします。

次に BLE 機器の検索を行います。検索には scanForPeripherals メソッドを使用します。任意のサービスを持った機器のみを対象としたい場合にはそのサービスの CBUUID を作り配列として第 1 引数に渡します。この指定を行うことで不要な機器の検出を抑止できます。指定しない場合には nil を渡します。第 2 引数はオプションです。Scan 時にペリフェラルの重複を許可したりできます。

スキャンにより BLE 機器が見つかると、CBCentralManagerDelegate の centralManager メソッドが呼ばれます。ここで見つかったペリフェラルの CBPeripheral インスタンスや取得出来たアドバタイズデータ、RSSI（電波強度）などが取得出来ます。新規に検出されたデバイスはテーブル View に追加されます。表示する情報は、デバイス名と RSSI 値です。iOS ではデバイスのアドレス情報は取得できない仕様となっています。

検索で見つかったデバイスリストのうち、接続したい相手をタップすると stopScan でスキャンを停止し、MainViewController 内の centralManagerDidSelectPeripheral デリゲードメソッドが呼び出されます。その後アプリケーション処理はメイン画面(MainViewController.swift) へ移行します。

5-7. デバイスの接続

対象の機器が決定されたら、CBCentralManager.connectPeripheral で GATT 接続します。
WRS-Sample-i では、GATT 通信関連は NORBluetootManager クラスで処理しており、アプリケーションプログラムは NORBluetoothManager を介して通信制御を行っています。

接続が完了すると、centralManager:didConnectPeripheral が呼ばれます。このイベントが発生するとメイン View の didConnectPeripheral デリゲートが実行されます。そして、次のステップとして peripheral.discoverServices を呼び出し、接続したペリフェラルの持つサービスから Nordic UART Service を検索します。

5-8. UART サービスでの通信

Nordic UART Service の場合、TX Characteristic がデバイス→ホスト機器へのデータ転送、RX Characteristic がホスト機器→デバイスへのデータ転送となります。

WRS-100 では読み取ったバーコードデータを一方通行でホスト機器に送信するため、TX Characteristic を使って通信を行います。

TX Characteristic では Notify（イベント通知）機能をサポートしています。

peripheral.discoverServices で目的のサービスの検索が終了すると、didDiscoverServices が呼ばれます。Nordic UART Service がサポートされていることを確認した後、今度は discoverCharacteristics を呼び出し、キャラクタリスティックを検索します。キャラクタリスティックの検索結果として、didDiscoverCharacteristicsForService が呼ばれます。TX Charactristic と RX Characteristic がサポートされていることを確認したら、setNotifyValue で TX Characteristic に対して Notification の受信要求を設定します。Notification が有効化された後は、バーコード読み取り毎に didUpdateValueFor characteristic が呼び出されます。

didUpdateValueFor characteristic は NORBluetootManager クラス内で定義されており、キャラクタリスティック値の読み出しと、UTF-8 形式の文字列への変換を行い、メイン View の didReceivedData デリゲートメソッドを呼び出します。didReceivedData 内で、複数パケットの場合のデータの結合、テキストウィンドウへの表示処理を行っています。また表示の際にはシステム音を使ってビープ音とバイブレータで通知します。

WRS-100 シリーズ

シリアル通信アプリケーション

開発ガイド

2018 年 1 月 17 日　第 2 版発行

アイメックス株式会社