V.BTTN Developer Quick Start Guide (2014)

1. V.BTTN - Feature Summary:

   V.BTTN is a wearable Bluetooth button with limitless applications. By pairing it with a Bluetooth Smart Ready (Bluetooth 4.0) mobile phone or a PC, it can be programmed to do anything – from ordering dinner, to sending a text message, to remotely activating the camera shutter.

   Additional information:

   Wireless (Up to 75' Indoors, up to 300' Outside)

   Waterproof (Meets IP 67 (depth of 1 meter for 30 minutes)

   Up to One year battery life – no need to charge

   Accelerometer for gesture control

   Audible Tones and LED visible feedback

   Range of accessories including money clip, pendant, and wristband

   Tiny – size of a poker chip

2. V.BTTN – Available Bluetooth features.

   The V.BTTN can be programmed to:

   a. Detect a short press and release. When this feature is enabled, upon pressing the button, the app will be notified. On a button release, the app will be notified again with a different notification value.
   b. Detect a long press-release. Example, the V.BTTN can notify your application if the user pressed-and-released the button for 2-9 seconds. With this feature, only one indication is sent to the app, only if the user press and released the button between 2-9 seconds (Press-and-release 10-23 seconds places the V.BTTN in advertising mode)
   c. To notify the app when a fall is detected. If this feature is enabled, and the V.BTTN is dropped from about 4 feet, a notification will be sent to the application.
   d. To disable all sounds and LED functions on the V.BTTN (Stealth Mode)
   e. Disable just sound or disable just LED functions.
   f. Enable the accelerometer, and report x, y, z accelerations to the application (note that enabling the accelerometer greatly reduces battery life).
   g. To buzz and turn on LED using the Immediate Alert Service (Find Me feature)
   h. Report battery level to the application.

   See the Bluetooth 4.0 Service descriptions in this document for technical details.

3. Bluetooth BLE background

Bluetooth low energy (BLE) technology was introduced in 2011, through the Bluetooth Specification v4.0 (Bluetooth v4.0). With its extremely low power consumption, unique characteristics, and new features, Bluetooth low energy technology enables new applications that were not practical with Classic Bluetooth technology. Coin cell battery-operated sensors and actuators in medical, industrial, consumer, and fitness applications (also known as “Smart”) can now smoothly connect to Bluetooth low energy technology–enabled smart phones, tablets, or gateways (also known as “Smart Ready”). Bluetooth low energy technology is ideal for applications requiring periodic transfer of small amounts of data...Bluetooth low energy technology is easy to set up, robust, and reliable in tough environments.

The behavior of a Bluetooth connection—whether Classic or low energy—is determined by the Bluetooth profiles a device has implemented. Devices can only communicate if they both have the same Bluetooth profile implemented.

Application developer must implement the corresponding profiles on the mobile app side. These profiles must match the profiles on the BLE device (i.e. V.BTTN).

There are some important differences between what profiles are available for Classic Bluetooth technology and for Bluetooth low energy technology.

A good example of the differences is seen in serial port emulation. Classic Bluetooth technology provides the serial port profile (SPP) for emulation of serial data connections. Bluetooth low energy technology provides no such support in the standard Specification v4.0; Many other profiles are not offered for Bluetooth low energy technology because of the differences in the connection models. The Classic Bluetooth scenarios that are not part of Bluetooth low energy technology include headset (HSP), object exchange (OBEX), audio distribution (A2DP), video distribution (VDP), and file transfer (FTP).

Home automation is a good example where devices can be controlled using LE technology; Health and Fitness is another where wearable sensors can monitor things like heart rate. They might even be coupled with location awareness to allow correlations to be drawn between effort and heart rate. This list is endless and is probably only limited by imagination.

From a topology perspective Bluetooth low energy (BT 4.0 LE) devices can take on a number of roles:

I. Scanner
II. Advertiser
III. Master
IV. Slave

Devices such as phones or tablets would generally (but not exclusively) adopt the role of “Scanner” and would “discover” other devices that have adopted the “Advertiser” role by a process called discovery which can be active (“are there any

devices out there?") or passive ("I'll listen whilst devices advertise their presence"). Devices that adopt the "Advertiser" role are generally (but not exclusively) smaller footprint devices such as heart rate monitors or temperature sensors or the V.BTTN device.
Once devices have discovered one another one will act as an "Initiator" (typically the phone or tablet type device) and attempt to connect to one of the devices that it has discovered. If successful it will adopt the role of "Master" and the other will adopt the role of "Slave". "Master" devices will initiate commands and requests to "Slave" devices which will respond.

4. After Connection:

   Once a device has been discovered the next task is to figure out what services does the device offer. So, what's a service? Well, a service consists of: a collection of characteristics.

   It is recommended that the developers go through the information from Bluetooth SIG related to Low Energy: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx

5. Development platforms:

   Windows 8 – Any PC that includes Bluetooth 4.0 LE (Bluetooth Smart Ready). Developers can use the standard Bluetooth Profile Driver functions and the new Bluetooth Low Energy functions, introduced in Windows 8, to create BLE client applications. Windows 8 BLE reference: http://msdn.microsoft.com/en-us/library/windows/hardware/jj159880(v=vs.85).aspx

   Android - Android phones with Bluetooth 4.0 hardware and Android version 4.3 or later, i.e. Samsung Galaxy S4, S5, HTC One (M8), Nexus 4 & 5, Moto X. The following resources will help when developing for Android: Android BLE Developer Reference: http://developer.android.com/guide/topics/connectivity/bluetooth-le.html

   Android has a very good sample BLE app, which it is recommended to start development with. Look for the sample project called "BluetoothLeGatt" in the Android SDK.

   iOS and Mac OS - iPhone 4S and newer

   Mac OS X support: the 2011 MacBook Air or newer, the 2012 MacBook Pro or newer, the 2012 iMac or newer, the 2011 Mac Mini or newer and the 2013 Mac Pro.

   iOS BLE Developer Reference:
   https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html

6. List of Profiles (SIG Services) that V.BTTN implements:

   V.BTTN implements the following Bluetooth SIG GATT based services. Your application has to implement the matching services (profiles) on the phone/tablet side. Note that not all the services have to be implemented; Only the ones that contain the interested functionality or information. For detail information on Bluetooth SIG adopted specification, please go to:

   https://www.bluetooth.org/en-us/specification/adopted-specifications

   a. Device Information Service

      The Device Information Service exposes information about V.BTTN like Model Number, Firmware/Software version, Hardware version, etc. Using the official SIG link above, you may get UUID details for this service, and the UUID for the characteristic within this service.

   b. Link Loss Service

      The Link Loss Service uses the Alert Level characteristic to cause an alert in the V.BTTN when the link is lost. Using the official SIG link above, you may get UUID details for this service, and the UUID for the characteristic within this service.

      The alert behaviors are as follow:

      00 No Alert

      01 Two short chirp once + blink red LED every 10 seconds for 15 minutes

      02 Blink red LED every 10 seconds for 15 minutes

      The default Alert Level on power up is 01.

   c. Immediate Alert Service

      The Immediate Alert Service cause an alert in the V.BTTN when it is written with a value other than “No Alert.” Using the official SIG link above, you may get UUID details for this service, and the UUID for the characteristic within this service.

      The V.BTTN alert behavior is as follow:

      00 No Alert.

      01 Chirp every 2 seconds for 20 seconds.

      02 Chirp + blink red LED every 2 seconds for 20 seconds

   d. Battery Service

      The Battery Service exposes the Battery Level of the coin cell in V.BTTN. Battery level are reported as a percentage, e.g. 5C is 92%. Using the official SIG

link above, you may get UUID details for this service, and the UUID for the characteristic within this service.

7. VSN GATT Service (UUID: 0xFFFFFFF0-00F7-4000-B000-000000000000)

VSN GATT Simple Service implements characteristics that allow applications to interact with V.BTTN's specific functions like button short/long press detection, fall detection, silent/stealth mode, etc. This is a custom VSN service (profile) so the UUID details for this service will not be found on the official Bluetooth SIG site. We provide you a table below with the VSN Service UUID and the characteristic UUIDs within the service.

UUID	Mnemoic	Value(default)	Permission	Notes
0x2800	<<Primary Service>>	0xFFFFFFF0-00F7-4000-B000-000000000000 (VSN SERVICE UUID)	READ	Start of VSN Service
0x2803	<<Characteristic>>	0A (properties: read/write) F1 FF (UUID: 0xFFFFFFF1-00F7-4000-B000-000000000000)	READ	Stealth Mode Configuration Characteristic Declaration
0xFFFFFFF1…	<<Stealth Mode Configuration>>	0 (1 byte)	READ | WRITE	Bit Mask field: 00 : Normal Mode 01 : Stealth Mode (Sound) 02 : Stealth Mode (LED) (e.g. 03 is Stealth Sound+LED)
0x2901	<<Characteristic User Description>>	"Stealth Config" (17 bytes)	READ	
0x2803	<<Characteristic>>	0A (properties: read/write) F2 FF (UUID: 0xFFFFFFF2-00F7-4000-B000-000000000000)	READ	Detection Configuration Characteristic Declaration
0xFFFFFFF2…	<<Detection Mode Configuration>>	0 (1 byte)	READ | WRITE	Bit mask to enable Alerts 0x01 - Enable button press detection 0x02 - Enable button alert detection 0x04 - Enable Fall alert detection 0x08 - Enable High-G alert detection 0x10 - Enable raw accelerometer reporting
0x2901	<<Characteristic User Description>>	"Detection Config" (17 bytes)	READ	
0x2803	<<Characteristic>>	08 (properties: write only) F3 FF (UUID: 0xFFFFFFF3-00F7-4000-B000-000000000000)	READ	Urgent Alert Characteristic Declaration
0xFFFFFFF3…	<<Urgent Alert>>	0 (1 byte)	WRITE	0 - No Alert 1 - Urgent Alert
0x2901	<<Characteristic User Description>>	"Urgent Alert" (17 bytes)	READ	
0x2803	<<Characteristic>>	10 (properties: notify) F4 FF (UUID: 0xFFFFFFF4-00F7-4000-B000-000000000000)	READ	Detection Notification Characteristics Declaration
0xFFFFFFF4…	<<Detection Notification>>	0 (1 byte)	(none)	Notifications 00 - button release detected 01 - button press detected 03 - Button press-release between 2-10 secs 04 - Fall event detected 05 - High-G event detected
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	0 = Disable Notification 1 = Enable Notification
0x2901	<<Characteristic User Description>>	"Detection Notify" (17 bytes)	READ	
0x2803	<<Characteristic>>	08 (properties: write only) F5 FF (UUID: 0xFFFFFFF5-00F7-4000-B000-000000000000)	READ	Verification Characteristic Declaration
0xFFFFFFF5…	<<Verification>>	00:00:00:00:00 (5 bytes)	WRITE	Verification key: 80:BE:F5:AC:FF
0x2901	<<Characteristic User Description>>	"" (17 bytes)	READ	

Table 1. VSN Service, Attribute Table

a. Stealth Mode Configuration (UUID: 0xFFFFFFF1-00F7-4000-B000-000000000000)

The first characteristic of the VSN GATT Service is the Stealth Mode Configuration characteristic (0xFFFFFFF1…). This characteristic is used to control the V.BTTN's LED and buzzer behavior. The following values can be written to this characteristic.

Bit Mask field:

00 : Normal Mode (Sound and LED are active on the V.BTTN)

01 : Stealth Sound Mode (Sound is disabled)

02 : Stealth LED Mode (LED is disabled)

(e.g. 03 is Stealth Sound+LED both are disabled)

b. Detection Configuration (UUID: 0xFFFFFFF2-00F7-4000-B000-000000000000)

The Detection Configuration Characteristic enables/disables V.BTTN's detection modes. The following values can be written to this characteristic to enable/disable the button detection, and accelerometer modes. Once the detection mode is configured, V.BTTN will notify the phone/PC using the Notification Characteristic described in section (d):

Note: The accelerometer modes (0x04, 0x08 and 0x10 are mutually exclusive).

Bit mask to enable detection

0x01 - Enable button (short) press detection

0x02 - Enable button (long) press detection

0x04 - Enable Fall detection

0x08 – Enable High-G alert detection

0x10 – Enable raw accelerometer reporting. The individual x, y, z axis data will be reported thru the accelerometer service described in section 8.

CAUTION: Enabling accelerometer service greatly reduces the V.BTTN's battery life.

c. Urgent Alert (UUID: 0xFFFFFFF3-00F7-4000-B000-000000000000)

The V.BTTN Alert Level Characteristic causes an "urgent" alert in the V.BTTN when it is written with a value 01. The V.BTTN alert behavior are as follow:

00 No Alert.

01 One chirp + alternating red-green LEDs.

d. Notification Characteristic (UUID: 0xFFFFFFF4-00F7-4000-B000-000000000000)

The Notification Characteristic uses the Client Characteristic Configuration (0x2902) to enable/disable notifications. Once enabled, V.BTTN notifies the application (using characteristic 0xFFFFFFF4-00F7-4000-B000-000000000000) when an event like button press or fall is detected by the V.BTTN.

Notifications:

00 - Button release detected

01 - Button press detected

03 - Button press-release between 2-10 seconds

04 - Fall event detected

05 – High-G event detected

e. V.BTTN Verification (UUID: 0xFFFFFFF5-00F7-4000-B000-000000000000)

The V.BTTN Verification Characteristic enables apps to communicate with V.BTTN. Verification key value is 80:BE:F5:AC:FF. Refer to section 5, Connecting to V.BTTN for details requirements for connecting to V.BTTN

8. Accelerometer Service

The V.BTTN includes a 3-axis accelerometer. The Accelerometer Service exposes raw accelerometer data to the application.

CAUTION: Enabling accelerometer service greatly reduces the V.BTTN's battery life.

Once raw reporting is enabled (see UUID 0xFFFFFFF2-00F7-4000-B000-000000000000), individual (X, Y, Z) axis notification can be configured by writing "01 00" to the Client Characteristic Configuration for the corresponding axis. Once notification is enabled, changes to X, Y, Z axis will be received thru characteristics 0xFFFFFFA3-00F7-4000-B000-000000000000, 0xFFFFFFA4-00F7-4000-B000-000000000000 and 0xFFFFFFA5-00F7-4000-B000-000000000000, respectively.

NOTE: V.BTTN default Bluetooth wake-up interval is 1 second. For applications that require shorter wake-up interval, see Connection Control Service to update the connection interval.

UUID	Mnemoic	Value(default)	Permission	Notes
0x2800	<<Primary Service>>	0xFFFFFFA0-00F7-4000-B000-000000000000 (ACCEL SERVICE UUID)	READ	Start of Accelerometer Service
0x2803	<<Characteristic>>	10 (properties: notify) A3 FF (UUID: 0xFFFFFFA3-00F7-4000-B000-000000000000)	READ	X-Axis Characteristic declaration
0xFFFFFFA3…	<<X-Axis Value>>	00:00 (2 bytes)	(none)	X-Axis value
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	X-Axis notification configuration
0x2901	<<Characteristic User Description>>	"X-Axis" (17 bytes)	READ	
0x2803	<<Characteristic>>	10 (properties: notify) A4 FF (UUID:0xFFFFFFA4-00F7-4000-B000-000000000000)	READ	Y-Axis Characteristic declaration
0xFFFFFFA4…	<<Y-Axis Value>>	00:00 (2 bytes)	(none)	Y-Axis value
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	Y-Axis notification configuration
0x2901	<<Characteristic User Description>>	"Y-Axis" (17 bytes)	READ	
0x2803	<<Characteristic>>	10 (properties: notify) A5 FF (UUID: 0xFFFFFFA5-00F7-4000-B000-000000000000)	READ	Z-Axis Characteristic declaration
0xFFFFFFA5…	<<Z-Axis Value>>	00:00 (2 bytes)	(none)	Z-Axis value
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	Z-Axis notification configuration
0x2901	<<Characteristic User Description>>	"Z-Axis" (17 bytes)	READ	

9. Connection Control Service

   The Connection Control Service exposes the V.BTTN's connection parameters: connection interval, slave latency and supervision timeout.

   How to convert connection interval to HEX parameter

   1. Connect interval unit (decimal) = connection interval time (ms) / 1.25 ms
   2. Convert connection interval decimal unit to HEX
   3. Reverse 2 byte HEX value

      For example: Connection interval 300ms.
      Connection interval unit (dec) = 300 ms / 1.25 ms = 240
      Convert 240 to Hex. 0x00F0
      Reverse 2-byte hex value: 0xF000

      Use 0xF0 0x00 as (min/max) connection interval in the byte array

UUID	Mnemoic	Value(default)	Permission	Notes
0x2800	<<Primary Service>>	0xFFFFCCC000F74000B0000000 00000000	READ	Start of Connection Control Service
0x2803	<<Characteristic>>	10 (properties: notify) C1 CC (UUID: 0xFFFFCCC1-00F7-4000-B000-000000000000)	READ	Connection Parameter Characteristic Declaration
0xFFFFCCC100F74000-B000000000000000	<<Connection Parameter>>	00:00:00:00:00:00 (6 bytes)	(none)	Connection Interval, Slave Latency and Supervision Timeout (2-bytes each)
0x2902	<<Client Characteristic Configuration>>	00:00 (2 bytes)	READ | WRITE	Connection Parameter Notification configuration
0x2901	<<Characteristic User Description>>	"Connect Parameter"	READ	
0x2803	<<Characteristic>>	08 (properties: write) C2 CC (UUID: 0xFFFFCCC2-00F7-4000-B000-000000000000)	READ	Request Connection Parameter Characteristic Declaration
0xFFFFCCC200F74000-B000000000000000	<<Request Connect Parameter>>	00:00:00:00:00:00:00:00 (8 bytes)	READ | WRITE	Min/Max Connection Interval, Slave Latency and Supervision Timeout (2-bytes each)
0x2901	<<Characteristic User Description>>	"Request Parameter" (17 bytes)	READ	

10. Connecting to V.BTTN

    Following are steps to connect the V.BTTN to the application:

    a. Enable Bluetooth on the phone.
    b. Press-release V.BTTN button between 10-24 seconds to make visible to the app (discoverable)
    c. Application performs scan for BLE devices and connects to it.
    d. Application performs service discovery
    e. Application sends verification key. This must be done within 30 seconds of connecting to V.BTTN.