

Implementing Eddystone™ Bluetooth® Smart Beacons Using the TI BLE-Stack™

Chester Kim

ABSTRACT

Keywords

- Eddystone™
- Beacon
- Bluetooth® low energy
- Bluetooth® Smart
- BLE-Stack™
- SimpleLink™ CC2640 Bluetooth Smart wireless MCU
- SimpleLink™ CC2650 Multi-standard wireless MCU
- SimpleBLEPeripheral

Contents

1	Introduction	2
2	Abbreviations	2
3	Objective	2
4	Project Overview and Prerequisites	3
5	Design and Implementation	3
6	Test	7
7	Summary	9
8	References	9

List of Figures

1	SimpleEddystoneBeacon Operation Mode Transitions	4
2	Eddystone Validator and Eddystone-URL Config Validator	8
3	Estimote Android App	8

List of Tables

1	Eddystone Frame Formats	3
2	Properties of Advertising Packets	4

BLE-Stack, SimpleLink are trademarks of Texas Instruments.
Bluetooth is a registered trademark of Bluetooth SIG.
Eddystone is a trademark of Eddystone Designs.
All other trademarks are the property of their respective owners.

1 Introduction

As described in the *Bluetooth® low energy Beacons* Application Note ([SWRA475](#)), a Bluetooth low energy (BLE) beacon is a device broadcasting a non-connectable advertising packet carrying small pieces of information to nearby devices. Depending on the timing and the information conveyed within the packet, beacons can enable a variety of use cases, including but not limited to proximity awareness, synchronization, identification, or just informing.

Some vendors have defined a protocol or format for how the beacon is implemented in a larger ecosystem. For example, Apple has defined the iBeacon protocol, available under an MFi license, for use with iOS devices.

Eddystone is an open beacon protocol specification from Google aimed at improving proximity-based experiences, with support for both Android and iOS smart device platforms. These experiences are implemented by specifying various beacon payload formats, defined as frames-types, as well as a corresponding set of APIs used to access these payloads from the smart device (such as an Android smart phone).

This application note describes how to implement an example Eddystone beacon device using the TI BLE-Stack V2.1 SDK on the SimpleLink CC2640 Bluetooth Smart wireless MCU. The lower power consumption and wide operating voltage capability make the CC2640 an ideal platform for implementing battery-powered Eddystone beacons.

Although beacons are, by definition, broadcast-only devices, the concept of a connectable beacon will be described. Having the beacon enter a connectable state (technically, switching from the GAP broadcaster to the peripheral role) is desirable for the purposes of provisioning and updating the metadata of the beacon. The Eddystone protocol defines a mode that allows the beacon data to be updated by an authorized client.

The project files and the source code files created or modified for Eddystone Beacon device implementation are available as a GitHub repository, which can be found on the [TI BLE Wiki](#).

2 Abbreviations

ADV	Advertising Packet
API	Application Program Interface
BLE	Bluetooth low energy
CCS	Code Composer Studio
ID	Identifier
GAP	General Access Profile
MCU	Microcontroller
OAD	Over the Air Download
SDK	Software Development Kit
TI-RTOS	Texas Instruments Real Time Operating System
TLM	Telemetry
TX	Transmission
UID	Unique Identity
URL	Uniform Resource Locator

3 Objective

This application shows what must be implemented on top of the existing SimpleBLEPeripheral sample application to make a complete working Eddystone beacon example project named SimpleEddystoneBeacon.

4 Project Overview and Prerequisites

Prior to following the examples described in this application note, the designer should have a detailed understanding of the TI BLE-Stack SDK as described in the SW Developer's Guide ([SWRU393](#)), the Bluetooth low energy Beacons application note ([SWRA475](#)), and the Google Eddystone protocol specification, which can be found at <https://github.com/google/eddystone/blob/master/protocol-specification.md>.

The sample application SimpleEddystoneBeacon requires TI BLE-Stack V2.1. Either IAR Workbench for ARM 7.40.2 or CCS 6.1 IDEs can be used to build the project. The SimpleEddystoneBeacon project runs on the CC2650DK (SmartRF06EB + CC2650EM) and CC2650 SensorTag reference platforms. An optional smartphone and test application can be used to validate the Eddystone beacon implementation.

The TI GIT repository [CC2640_simpleeddystonebeacon](#) includes the minimum number of files necessary to build the project when they are put on top of the existing BLE-Stack V2.1 installation. The repository consists of project and workspace files for IAR/CCS, as well as *simpleEddystoneBeacon.c* and *simpleEddystoneBeacon.h*, which are for the application, and *eddystoneURLCfg.c* and *eddystoneURLCfg.h*, which provide URF Configuration service. The slightly modified *peripheral.c* is also included. To install the SimpleEddystoneBeacon project, copy the repository on the `<INSTALL_DIR>\Projects\ble`, where `<INSTALL_DIR>` is the location where TI BLE-Stack 2.1 is installed. By default, it is `C:\ti\simplelink\ble_cc26xx_2_01_00_44423`.

The SimpleEddystoneBeacon application projects have the configurations for CC2650DK and CC2650 SensorTag. Because SensorTag configurations are combined with an OAD option, Chapter 9 of CC2640 BLE OAD User's Guide must be read carefully before building those configurations. CC2640 BLE OAD User's Guide is available at `<INSTALL_DIR>\Documents`.

5 Design and Implementation

5.1 Operation Modes Overview

The Eddystone protocol defines three frame-type formats (referred to as "frames"), which are transmitted by the beacon device as described in [Table 1](#).

Table 1. Eddystone Frame Formats

Frame-type	Description
UID	Unique identifier that can identify a particular beacon. Useful for location aware applications.
URL	Compressed web URL, such as https://goo.gl/Aq18zF , that can be launched by the smart device application.
TLM	Telemetry information, such as battery voltage and temperature of the beacon.

These frames are described in more detail in the Eddystone protocol specification. Each frame represents an advertising packet, up to the maximum 31-byte payload defined by the Bluetooth 4.0 core specification.

The Eddystone Beacon defines two modes of operation: regular advertising mode and an optional URL configuration mode. These operating modes are related by five individual operational states. In regular advertising mode, the beacon operates as a traditional broadcaster sending non-connectable advertisements consisting of UID, URL, or TLM frames. In URL configuration mode, the beacon transmits connectable advertisements to allow a client to connect, and if authorized, update the URL content and the system parameters through the URL configuration service.

The operating state transitions for these modes of operation are depicted in [Figure 1](#), along with the corresponding actions used to transition states in the SimpleEddystoneBeacon example project. States shown with a grey circle represent URL configuration mode, while the other states represent regular advertising mode.

The Eddystone protocol does not specify the type, when, or how often a particular frame is transmitted, or the actions that trigger a particular state transition. These details are implementation-specific. An Eddystone Beacon may, for example, transmit only URL frames, or a combination of all three frames at different intervals.

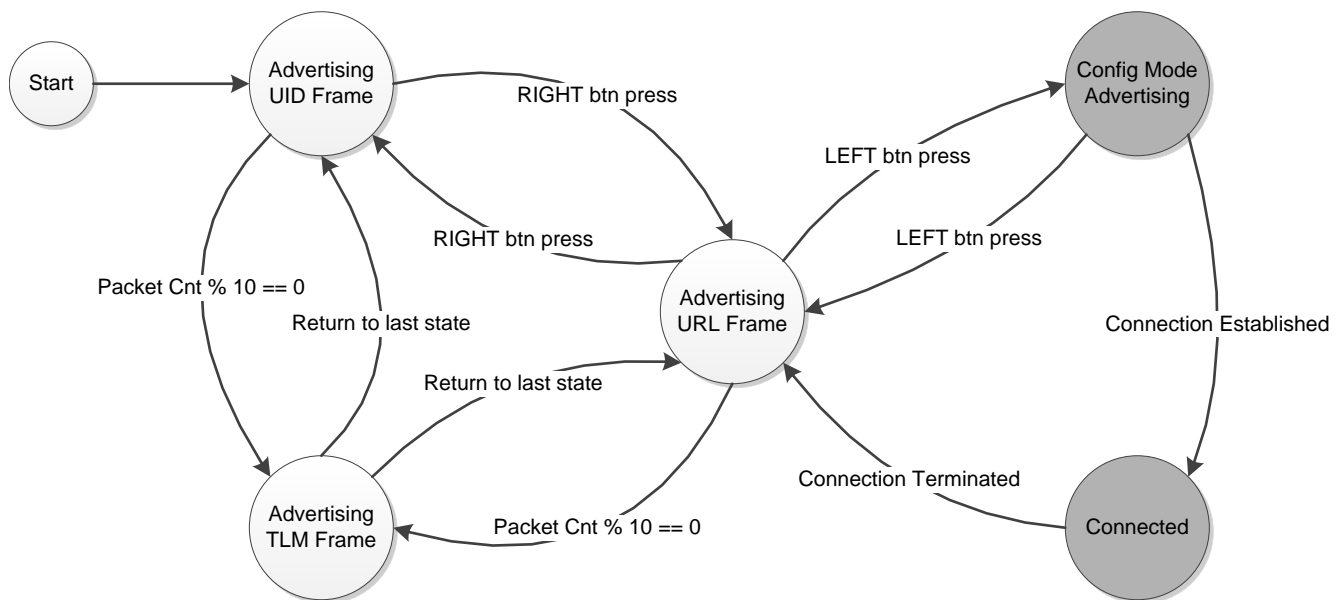


Figure 1. SimpleEddystoneBeacon Operation Mode Transitions

In this design, the beacon device sends either UID frames or URL frames regularly, and TLM frames intermittently, while in regular advertising mode. The state transitions between UID and URL are toggled by a RIGHT button press. For practical reasons, TLM frames are expected to be observed less frequently than UID or URL frames, so in this design, they are sent only every 10th advertise of UID or URL.

URL configuration mode can be entered by entering the config mode advertising state in this design. It is triggered by a LEFT button press. Exiting configuration mode can be caused by a LEFT button press, or any type of termination of the connection. Generally, the central terminates the connection when the beacon is finished updating, to let the beacon resume regular advertising mode.

To use a button press to trigger some of the transitions, the key event handler must be implemented. Button press handling is implemented slightly differently by hardware platform. In this implementation, functions in *board_key.c* are used for the SmartRF06EB, while the equivalent code for the SensorTag has been added to the application source file. In any case, *SimpleEddystoneBeacon_keyChangeHandler()* is registered as the callback and eventually the key event SEB_KEY_CHANGE_EVT is handled by *SimpleEddystoneBeacon_handleKeys()*.

5.2 Regular Advertising Mode

Advertising is used for four different purposes, regardless of the operation mode. Properties of each type of the ADVs used in this implementation are summarized in [Table 2](#).

Table 2. Properties of Advertising Packets

	UID/URL/TLM Frame	URL Configuration Mode ADV
ADV Event Type	ADV_NONCONN_IND	ADV_IND
ADV Type Flags	BREDR_NOT_SUPPORTED	BREDR_NOT_SUPPORTED GENERAL_DISCOVERABLE
ADV Interval	From URL Cfg for UID/URL, 10 times longer for TLM	Default Interval
TX Power	From URL Cfg	Medium Power Mode
Connectable	No	Yes

As switching from one advertising packet to another at run time is required, how to set the parameters to change the properties for regular advertising mode is described in this section. How to get the information to populate the payload of each packet is also covered.

5.2.1 Setting Advertising Parameters

As shown in [Table 2](#), there are five properties to be changed when toggling between regular advertising mode and URL configuration mode.

ADV Event Type can be configured using *GAPRole_SetParameter()*, with *GAPROLE_ADV_EVENT_TYPE* as the parameter ID and *GAP_ADTYPE_ADV_NONCONN_IND* as the value, because regular advertising mode is non-connectable.

ADV Type Flags is an item included in an advertising packet. The *GAP_ADTYPE_FLAGS* field of the struct variable *eddyystoneAdv* for regular advertising mode is initialized with *GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED*, because regular advertising mode is non-discoverable.

ADV Interval is set by specifying minimum and maximum intervals for limited and general discoverable ADVs. Typically, all four parameters are set to the same value to get the desired interval as follows:

```
GAP_SetParamValue(TGAP_LIM_DISC_ADV_INT_MIN, advInt);
GAP_SetParamValue(TGAP_LIM_DISC_ADV_INT_MAX, advInt);
GAP_SetParamValue(TGAP_GEN_DISC_ADV_INT_MIN, advInt);
GAP_SetParamValue(TGAP_GEN_DISC_ADV_INT_MAX, advInt);
```

That particular design should select advertising parameters that balance power usage with beacon power use. Using faster advertising intervals allows for quicker discovery by the smart device, while longer intervals allow for longer beacon battery life.

The value from the parameter corresponding with *URLCFG SVC_BEACON_PERIOD* of the URL configuration service is reflected on *advInt*. Unit conversion is necessary in this case because the value from the URL configuration service is in millisecond, while *GAP_SetParamValue()* expects a number of 625-microsecond ticks. See [Section 5.3.2](#) for more details about URL configuration service. Basically the beacon period from the URL configuration service is supposed to be applied to URL frame advertising. However, it is used for UID as well in this implementation, because no external means to setup the interval for UID is defined in the Eddystone specification. The interval for TLM depends on the interval for UID/URL, because it is based on the number of UID/URL frames that have sent since the last TLM frame in this implementation.

Use the value from the parameters corresponding with *URLCFG SVC_ADV_TX_PWR_LVL* and *URLCFG SVC_TX_POWER_MODE* of the URL configuration service to get the designated TX Power in dBm.

Once all of the above properties are set and the ADV data is updated, advertising is ready to start. [Section 5.2.2](#) describes more about updating ADV data.

Lastly, non-connectable advertising mode is enabled to start sending out the packets at the given rate and power. *GAPRole_SetParameter()*, with the parameter ID *GAPROLE_ADV_NONCONN_ENABLED*, is used to start or stop advertising in regular advertising mode.

All the procedures described in this section are done in *SimpleEddystoneBeacon_applyConfiguration()* and *SimpleEddystoneBeacon_startRegularAdv()*.

5.2.2 Populating the Eddystone Frame Payload

Data in UID and URL frames generally don't need to be changed unless the beacon device has been updated during the last configuration mode operation. The Ranging Data subfield of UID frame and all subfields of URL frame except for Frame Type must be updated upon exiting configuration mode. Updates are implemented in *SimpleEddystoneBeacon_applyConfiguration()*.

TI recommends a few algorithms to generate a 10-byte ID Namespace subfield and a 6-byte ID Instance subfield of UID by the Bluetooth core specification. This implementation example uses some meaningless numbers, but for an actual product, the number assignments in function *SimpleEddystoneBeacon_initUID()* must be replaced with proper algorithms.

For URL frame, *SimpleEddystoneBeacon_encodeURL()* facilitates conversion of regular null-terminated string URL into encoded numbers.

TLM frames require special handling compared to UID and URL frames, because the frame data must be updated with the most recent information each time a TLM frame is broadcasted. Updating the TLM payload is done in *SimpleEddystoneBeacon_updateTLM()*.

The battery voltage VBATT subfield can be obtained from *AONBatMonBatteryVoltageGet()*. To use the battery monitor function, include *aon_batmon.h* of driverlib.

A constant value is currently used for the temperature TEMP subfield, because the SmartRF06EB does not have a temperature sensor. On the SensorTag, the integrated temperature sensor can be utilized for this.

The ADV_CNT subfield needs an event from the stack, notifying the application upon completion of each advertising operation. *HCI_EXT_AdvEventNoticeCmd()* is the API to register an event that is sent upon completion of an ADV send out. Register SEB_ADV_COMPLETE_EVT and check it in the processing loop of stack events that come with the signature of 0xFFFF. The event SEB_ADV_COMPLETE_EVT is eventually processed in *SimpleEddystoneBeacon_processAdvCompleteEvt()*. This event processing function is not only for counting the number of ADVs, but also which type of Eddystone frame should be sent.

A 1-sec resolution clock tick counter is used, as an example, for SEC_CNT subfield, though the Eddystone spec requires 100-ms resolution. *UTC_clock.c*, originally used by the TimeApp example project, was added for this. Typically, a TLM frame is supposed to be sent intermittently, so in this example it is sent every 10 seconds by default, allowing for a 1-sec resolution.

When a local Eddystone frame is populated with all necessary information, it must be handed over to the BLE stack. *GAPRole_SetParameter()*, with the parameter ID GAPROLE_ADVERT_DATA, is used for that purpose.

All the procedures described in this section are handled by *SimpleEddystoneBeacon_selectFrame()*.

Dynamic change of the ADV payload in non-connectable advertising mode is not considered in *peripheral.c* by design, because it is originally intended to take care of the GAP role of the peripheral device, which normally uses connectable advertising. To enable dynamic ADV payload change in non-connectable advertising mode, make a small modification in *gapRole_processGAPMsg()* in *peripheral.c* as follows:

```
case GAP_ADV_DATA_UPDATE_DONE_EVENT:
{
    gapAdvDataUpdateEvent_t *pPkt = (gapAdvDataUpdateEvent_t *)pMsg;

    if (pPkt->hdr.status == SUCCESS)
    {
        if (pPkt->adType)
        {
            // Setup the Response Data
            pPkt->hdr.status = GAP_UpdateAdvertisingData(selfEntity,
                FALSE, gapRole_ScanRspDataLen, gapRole_ScanRspData);
        }
        else if ((gapRole_state != GAPROLE_ADVERTISING)    &&
            (gapRole_state != GAPROLE_CONNECTED_ADV)    &&
            (gapRole_state != GAPROLE_CONNECTED ||
                gapRole_AdvNonConnEnabled == TRUE)    &&
#ifdef CONNECTABLE_BEACON_SUPPORT
            (gapRole_state != GAPROLE_ADVERTISING_NONCONN) &&
#endif
            (Util_isActive(&startAdvClock) == FALSE))
        {
            // Start advertising
            gapRole_setEvent(START_ADVERTISING_EVT);
        }
    }

    ...

}
break;
```


This change is incorporated into the accompanying patch included in the repository.

5.3 URL Configuration Mode

The URL configuration mode is an optional method to update the URL content of the beacon. A designer may wish to omit this mode, or modify the accompanying service based on implementation-specific requirements.

5.3.1 Setting Advertising Parameters

To create the set up for configuration mode advertising, use the same method described in [Section 5.2.1](#) but with different properties than shown in [Table 2](#). In this section, only the differences from [Section 5.2.1](#) are noted.

For ADV Event Type, GAP_ADTYPE_ADV_IND as the value is used because configuration mode is connectable.

For ADV Type Flags, the GAP_ADTYPE_FLAGS field of the struct variable *eddystoneCfgAdv* for configuration mode is initialized with GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED, together with GAP_ADTYPE_FLAGS_GENERAL, because configuration mode is discoverable.

advInt for ADV Interval is set to GAP_ADTYPE_ADV_IND. Typically the interval for configuration mode is shorter than that for regular advertising mode.

TX Power should be set to the value corresponding to TX_POWER_MODE_MEDIUM.

When all of the above properties are set and the ADV data is updated using *GAPRole_SetParameter()* with the parameter ID GAPROLE_ADVERT_DATA, advertising is ready to start.

Lastly, connectable advertising mode is enabled by calling *GAPRole_SetParameter()* with the parameter ID GAPROLE_ADVERT_ENABLED.

All the procedures described in this section are handled in *SimpleEddystoneBeacon_startConfigAdv()*.

5.3.2 URL Configuration Service

URL Configuration service is implemented in *eddystoneURLCfg.c*. The application can access the nine characteristics of the service using *URLCfgSvc_GetParameter()* and *URLCfgSvc_SetParameter()*. The application registers a callback called when changes happen in the characteristics by the central. In this implementation, only a change in the Reset characteristic notifies the application.

Once the beacon device is connected with a central (such as a smart phone) in URL configuration mode, some of the system properties on the beacon device can be modified by the central through writing to characteristics, if the Lock State is FALSE. Only Unlock characteristic can be written to if Lock State is TRUE.

Updated characteristics are reflected on the corresponding properties in *SimpleEddystoneBeacon_applyConfiguration()* when the beacon device exits URL configuration mode to enter regular advertising mode by being disconnected for any reason, such as termination or time out. However, actions should be immediately taken when the Reset characteristic changes.

6 Test

To verify all the features and functions described in the application note, devices implementing the observer or central GAP roles are necessary. A GitHub Eddystone repository comes with validation tools for that purpose. Using [Eddystone Validator](#) and [Eddystone-URL Config Validator](#), we can verify URI/URL/TLM frames and URL configuration mode, respectively. [Figure 2](#) shows the screenshots of Eddystone Validator and Eddystone-URL Config Validator that have been used to verify SimpleEddystoneBeacon sample application. The validators used to validate SimpleEddystoneBeacon project are based on the commit SHA b5f23b3f895007c761e2303db4c10df7af2f997f of the GitHub Eddystone master branch.

Besides the validators, there are several Android/iOS applications that can be used to test Regular Advertising Mode functionalities of Eddystone-compatible beacons. For example, using [Estimote Android App](#), we can see how Eddystone-compatible advertising packets are parsed and used by applications as shown in [Figure 3](#). These smart phone applications are not developed or supported by Texas Instruments.

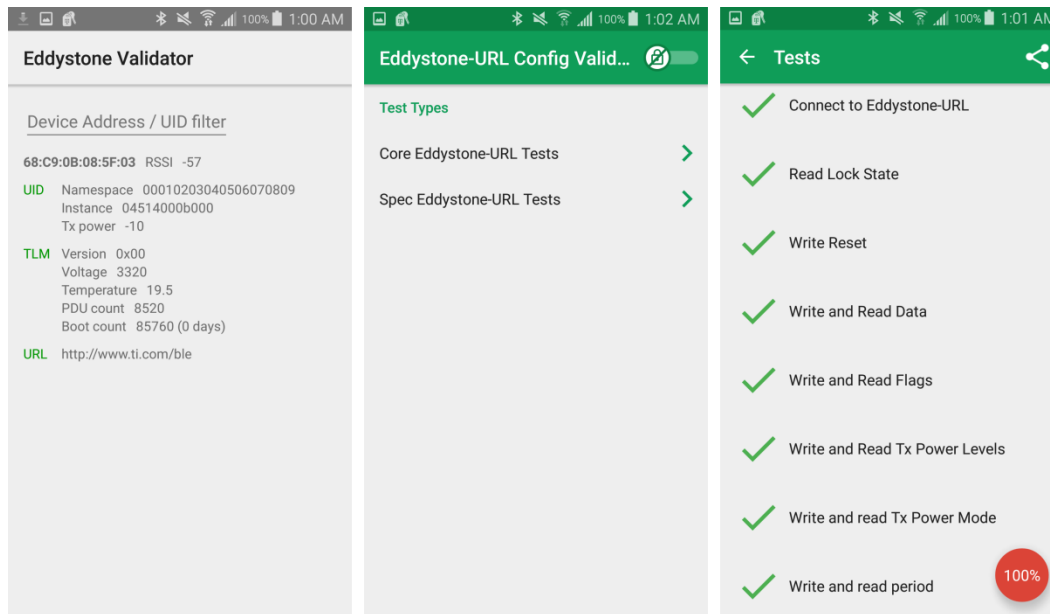


Figure 2. Eddystone Validator and Eddystone-URL Config Validator

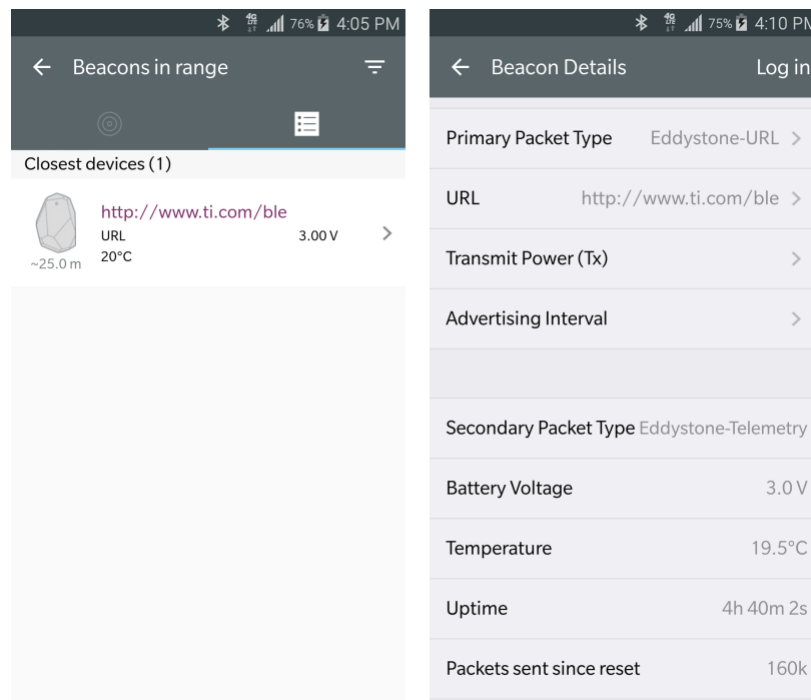


Figure 3. Estimote Android App

7 Summary

This application note describes the basic concept of Eddystone Beacon, how connectable beacon applications can be implemented on top of the existing sample application SimpleBLEPeripheral, and how the compatibility to Eddystone specification can be tested. Also, events, callbacks, and usages of hardware drivers are introduced for obtaining information to populate the TLM frame.

With a simple feature addition and modification on the sample application, connectable beacon devices such as a Eddystone-compatible beacon device can be easily developed.

8 References

1. Bluetooth low energy Software Developer's Guide ([SWRU393](#))
2. CC2640 OAD User's Guide
3. Bluetooth low energy Beacons ([SWRA475](#))
4. [Google Eddystone Github](#)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com