

Get Started with EM Beacons

This article is a QuickStart guide for working with EMBC01 & EMBC02 Bluetooth Low Energy beacons.

Each beacon comes programmed from the factory with a unique MAC address and a standard set of advertising options. Pressing the button repeatedly (squeezing the beacon in the center) cycles through the different advertising states. The set of standard states depends on the model:

```
EMBC01
1. Off
2. Proximity, outputPowerLevel 1, beaconPeriod 100 ms.
3. Proximity, outputPowerLevel 3, beaconPeriod 500 ms.
4. Proximity, outputPowerLevel 5, beaconPeriod 1000 ms.

Where 'Proximity' = IDData + AltBeacon + E-UID

EMBC02 (EMBC01 + Accelerometer)
1. Off
2. Sensor Data, outputPowerLevel 4, beaconPeriod 1000 ms.
3. IDData + E-UID + E-TLM, outputPowerLevel 3, beaconPeriod 100 ms.
4. IDData + E-URL + E-TLM, outputPowerLevel 7, beaconPeriod 100 ms.
```

With the right tools and know-how, these advertising options are fully customizable without modifying or recompiling the source code. You can change characteristics such as

```
* Packet types. Advertisements can be any combination of
  * EM sensor data - advertising with live data like acceleration or temperature
  * ID Data - compatible with Apple's iBeacon technology
  * AltBeacon - Radius Networks' open proximity beacon standard
  * Eddystone - Google's open standard, including -UID, -URL, and -TLM varieties.
* State machine. Simple on/off or cycle through the states.
* Interval. The period between beacon advertisements
* Transmit power over a range from -18 to +4 dBm
* Beacon identity values including MAC address, UUID, MajorID and MinorID
* The content of sensor-data packets
```

Taking *custom* one step further, EM provides the EMBCxx SDK with full source downloadable from this forum. Registration is required.

This article is focused on the tools and techniques for customizing advertising options as described above. It covers the items you need and where to get them. It guides you through the steps necessary to customize parameters and program multiple beacons to operate exactly the way you want them to.

What You Need

The five necessary components are listed here. You can order and/or download these items using the links in the list.

1. An RLink 6819 Debugger/Programmer, either the [Standard](#) (P/N EMRLK6819) or [Pro](#) (P/N EMRLKP6819) model. The Standard RLink has limited debug capability. If you plan to make FW source code modifications, consider buying the Pro model up front. Unfortunately there's no option to upgrade a Standard to a Pro.
2. A BLE Beacon Development Kit which can be either the [EMBC01 Proximity Beacon](#) (P/N EMBC01DVK) or the [EMBC02 with Acceleration Sensor](#) (P/N EMBC02DVK).
3. The latest [EM Beacon Software Development Kit \(SDK\)](#) (requires EMDeveloper forum registration)
4. A PC running Windows 7, 8 or 10 to run the programming and update scripts
5. [A Ruby language interpreter](#)

Let's start with a look at each of these items. The hardware prerequisites are pictured below with the RLink Programmer/Debugger above and to the right. The DVK components are below and to the left.



The Rlink Programmer Debugger

Beacon programming uses the GASP interface to talk to the EM6819 controller. RLink models that support other processors like ARM and the 8051 will not work with EM Beacons. As mentioned above, the Standard model EMRLK6819 can compile, link and program firmware images of any size but debugging is limited to images of up to 2K. The PRO model EMRLKP6819 removes the 2K restriction but costs more. The standard model is sufficient for all the customization activities described in this article. Consider the PRO model if you plan to modify the firmware.

Compiling the firmware can be accomplished using the Ride7 IDE and the RKit-C816. The toolchain is free with registration at the [Raisonance Support web site](#). They are not required for the activities described in this article.

BLE Beacon Development Kits (DVK)

Each DVK ships with five coin-shaped beacons like the one shown in the picture above, protective plastic housings, an adapter board for the RLink level-shifter, and a 6-pin Tag Connect cable.

The AA battery holder shown in the picture is not included in the DVK. You must supply 3.0V to the target. You have the option of using a power supply, adding the battery holder as we did, or powering the beacon with the included CR2032 coincell battery. For occasional use and small numbers of beacons, the coincell battery is perfectly acceptable. The trick is to bend the battery retention clip back, then slide the coincell battery deeper into its cage so that power is still applied but the battery no longer interferes with the pins on the programming cable. For debugging or programming larger volumes, it's much more convenient to use a bench supply or battery holder as we have done.

Also for hands free debugging, it's convenient to use the [TC2030-CLIP Retaining Clip](#) available directly from Tag Connect. We've also been known to use the rubber band trick as shown above! (However the retaining clip is more reliable.)

The EM Beacon Software Development Kit (SDK)

SDKs include schematics and layout plots, source code, libraries, Intel hex files and a set of programming utilities. All we need for this article are the hex files and utilities but you must download the whole SDK to get them. If you haven't already, create an account at [forums.emdeveloper.com](#). Your membership should be approved within 24 hours, often faster. This gives you full access to all developer discussions and downloads. The latest SDKs are in the [Downloads Forum](#).

"Beacon Firmware Dev Kit V3.0.0" is current as of this writing. However, the methodology described here generally applies to all revs including the Over-the-Air (OTA) SDK. Parameters and their range of values may differ. Always check in the `...util` directory for the `ParameterDictionary.pdf` revision that applies to your SDK.

Treat the SDK as a workspace. Download and unzip it wherever you would like to work. After unzipping, you should have a folder named `EM6819_Beacon`. The two items you need are contained in this folder.

1. The `util` directory and it's content
2. Either a. `project\EMBC01\build\EMBC01.hex` for the EMBC01 proximity beacon b. `project\EMBC02\build\EMBC02.hex` for the EMBC02 with acceleration sensor

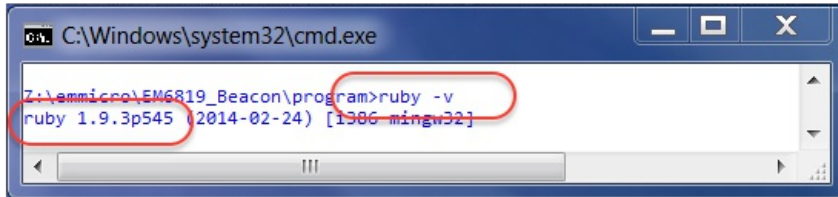
Windows and a Ruby Interpreter

The beacon programming and update tools in the SDK are written in the Ruby language. They have been tested on Windows 7, 8 and 10. Ruby is

not installed by default on Windows, so the first step is to check to see whether or not you have a useful version installed.

1. Open a command line window. (Windows Start menu...Run. Type `cmd`, then press the ENTER key.)
2. Type `ruby -v` at the DOS command prompt. Press ENTER.

If all goes well, the response will be something like this



```
C:\Windows\system32\cmd.exe
Z:\emmicorp\EM6819_Beacon\program>ruby -v
ruby 1.9.3p545 (2014-02-24) [i386-mingw32]
```

This article assumes you have Ruby version 1.8.7 or above. If you need to install or upgrade, I highly recommend the [RubyInstaller for Windows site](#). Running an installer from there ([rubyinstaller.org](#)) will put Ruby on your path and give you the option of associating ".rb" files with the interpreter.

Set Up the Programmer

Connect the level-shifter to the RLink programmer. Some RLink units ship with 2 accessory boards. The one you want is labeled `ADP-LV` and has the EM Microelectronic logo. The connectors are keyed and fit with the RLink only one way.

Similarly, connect the DVK adapter board to the RLink+LevelShifter. The adapter board is keyed as well and fits only one way.

Connect the Tag-Connect cable to the adapter board. The red wire on the ribbon cable should be aligned with pin 1 which is labeled on the adapter board.

Insert the tag connect pins into the programming pads on the EMBC01/02 beacon. We won't program yet. This is just to verify that the CR2032 battery is not interfering with the programming pins.

Create a CSV Parameter File

The default EMBC01 broadcasts proximity packets (idData + AltBeacon + E-UID). It changes RF power level and broadcast intervals with each button push as it cycles between three different active states and off. Instead, let's alter this behavior to a simple on/off configuration where the beacon alternates between 1) Sensor + AltBeacon packets and 2) off. We'll program 5 beacons, each with a different transmit power setting and beacon period in order to test them in the field.

Consulting [the parameter dictionary](#), we determine the following.

- A `beaconMode` parameter value of `0x05` will enable the Sensor and AltBeacon packet types as desired.
- A `beaconMachines` value of `0x01` will give the CUSTOM (on/off) behavior.
- We'll use a different power setting for each of the five beacons. Legal values are from 0 (-18 dBm) to 7 (+4 dBm) so we'll use 0, 1, 3, 5 and 7.
- Likewise we'll vary the `beaconPeriod` value, choosing periods of 50, 100, 250, 500 and 750 ms.

We need to encode our choices in a CSV (comma separated value) file. We'll call it `FieldTest.csv`. Note that spaces are **not** used before, after or between the values. Otherwise, errors will result.

```
beaconMode,beaconMachines,outputPowerLevel,beaconPeriod
0x05,0x01,0,50
0x05,0x01,1,100
0x05,0x01,3,250
0x05,0x01,5,500
0x05,0x01,7,750
```

This example illustrates the use of both hexadecimal and decimal values. Only the values that need to be altered are given in the CSV file. Others, like the deviceAddress already programmed in the beacon devices, will remain unchanged.

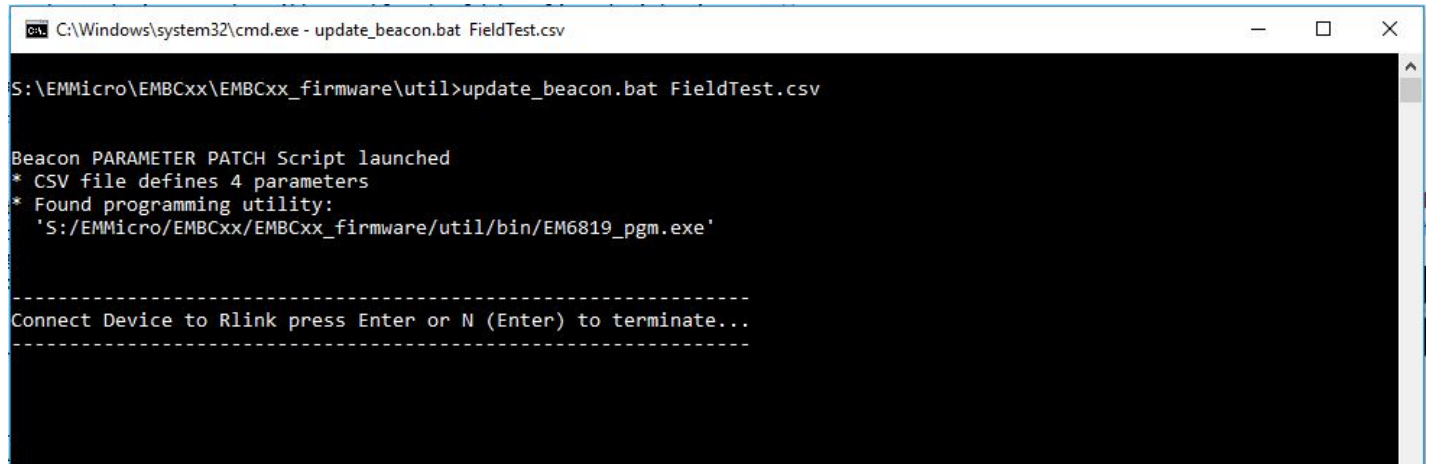
Run the Program Script

Drop the `FieldTest.csv` file in the SDK `..util` directory and open a DOS shell in that directory. (Tip: In Windows, navigate to the parent directory, hold the shift key down and right-click on util, select 'Open command window here')

Command options are fully described in \util\USAGE.pdf. Since we're only altering parameter values and not the firmware itself, we will use the `update_beacon` script like this.

```
update_beacon FieldTest.csv
```

`Update_beacon` invokes the required Ruby scripts. You should see a display something like this:



```
C:\Windows\system32\cmd.exe - update_beacon.bat FieldTest.csv

S:\EMMicro\EMBCxx\EMBCxx_firmware\util>update_beacon.bat FieldTest.csv

Beacon PARAMETER PATCH Script launched
* CSV file defines 4 parameters
* Found programming utility:
  'S:/EMMicro/EMBCxx/EMBCxx_firmware/util/bin/EM6819_pgm.exe'

-----
Connect Device to Rlink press Enter or N (Enter) to terminate...
-----
```

Now connect the Tag-Connect programming pins to the beacon PCB using the registration pins for proper alignment. Maintain enough pressure to ensure good contact between the spring-loaded pogo pins and the lands with one hand and press ENTER. The scripts will do the following:

1. Launch the Raisonance programming utility to create a hex file by dumping the parameter region of the device's memory
2. Verify that it can find a valid version number in the flash memory dump
3. Create a second hex file with the new parameter values
4. Re-launch the Raisonance programming utility to update the flash-memory parameters
5. Scan the programming utility output for errors and display the full set of parameter values (not just the ones you changed)
6. Prompt for the next device to update

The resulting console output and prompt for the next device looks like this.

```
C:\Windows\system32\cmd.exe - update_beacon.bat FieldTest.csv
* Searching for parameter block in the device to be updated...
* (This will take several attempts for older firmware versions)

* Initiating flash memory dump to S:/EMMicro/EMBCxx/EMBCxx_firmware/util/Beacon_SN_0x452A5134435210_dump.hex

>S:/EMMicro/EMBCxx/EMBCxx_firmware/util/bin/EM6819_pgm.exe d6f40,60,S:/EMMicro/EMBCxx/EMBCxx_firmware/util/Beacon_SN_0x452A5134435210_dump.hex
EM6819_pgm: software for programming EM6819 chips using a RLink as programmer.
Copyright 2007-2009 Raisonance S.A.S..

Connecting to RLink... OK
RLink Serial Number: dngGASP00000217

Connecting to EM6819... OK

Dumping Data Flash range 0x6F40-0x6F9F to file S:/EMMicro/EMBCxx/EMBCxx_firmware/util/Beacon_SN_0x452A5134435210_dump.hex ... OK
* Writing updated flash memory image to file
S:/EMMicro/EMBCxx/EMBCxx_firmware/util/Beacon_SN_0x452A5134435210_update.hex

- - - - - Starting programmer - - - - -
>S:/EMMicro/EMBCxx/EMBCxx_firmware/util/bin/EM6819_pgm.exe OS:/EMMicro/EMBCxx/EMBCxx_firmware/util/Beacon_SN_0x452A5134435210_update.hex S

EM6819_pgm: software for programming EM6819 chips using a RLink as programmer.
Copyright 2007-2009 Raisonance S.A.S..

Connecting to RLink... OK
RLink Serial Number: dngGASP00000217

Connecting to EM6819... OK

Overwriting file S:/EMMicro/EMBCxx/EMBCxx_firmware/util/Beacon_SN_0x452A5134435210_update.hex to Flash... OK

Starting execution... OK
- - - - - Programming complete - - - - -

EM6819SerialNumber: 0x452A5134435210
deviceAddress: 0EF3EE010205
idDataUUID: 0x699EBC80E1F311E39A0F0CF3EE3BC012
idDataMajorId: 0x005A
idDataMinorId: 0x354D
beaconPeriod: 0x000032
outputPowerLevel: 0x00
beaconMode: 0x05
beaconMachines: 0x01
openSensor: 0x01
temperatureOffset: 0x0000
EddyStoneUUID: 0x699EBC80E19A0F0CF3EE050201EEF30E
EddyStoneURL: https://www.emdeveloper.com
dateTimeProgrammed: 20160731T195928
FirmwareVersion: 454D0300

-----
Connect Device to Rlink press Enter or N (Enter) to terminate...
-----
```

Note that the new parameter values are displayed in hexadecimal format regardless of how you entered them in your CSV file.

Set the programmed device aside, connect the next beacon and press enter to program. Repeat for all five beacons. Each programming cycle will use the next row in the CSV file.

Device Self-test and Normal Operation

Once a device is programmed, the Rlink releases its reset line. Before entering the sleep state for the first time, the beacon performs its Power-On Self Test (POST) functions. This behavior will be repeated again each time a device is programmed or a battery is reinserted.

Both LEDs flash on during POST and off upon successful completion. If the LEDs remain on, POST has failed. Remove and reinsert the battery. If

the problem persists, the device should be reflashed. If POST still fails, contact EM for a replacement.

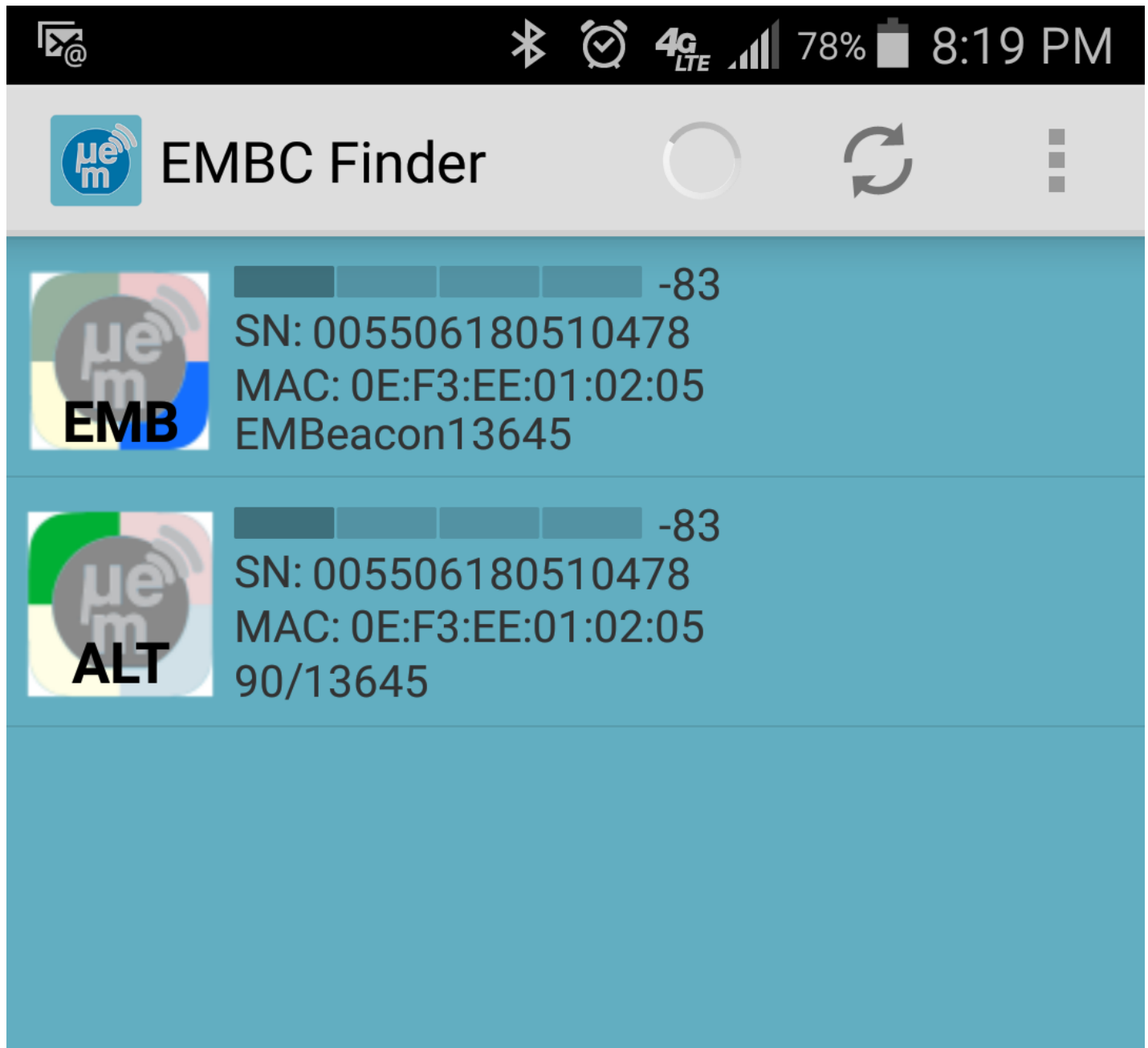
After POST is finished, the COiN Custom state machine sleeps until the beacon is activated by squeezing it to push the button located underneath the battery. When activated, the beacon transmits one packet each each wake-up cycle and sleeps for the remaining duration of the beacon interval. This interval is extended by a random increment of up to 10 mSec, per BT specification, to prevent two or more accidentally synchronized beacons from continually colliding.

For example, the beacon programmed with parameters from the first line in our CSV file will wake up every 50-60 ms and transmit one packet type on all three advertising channels. The very first wake up cycle produces a Sensor Data packet and the next an AltBeacon packet. In our example, the period between any two packets *of the same type* is two times the beacon interval or 100-120 ms. Only one packet type is transmitted in each wake-up cycle so enabling all six packet types results in (at least) six times the beacon interval between similar packet types.

Display the Beacons using EMBC Finder

At this point, you've programmed one or more beacons but they're off. Press the button(s) to turn them on. The green LED flashes briefly as they go on.

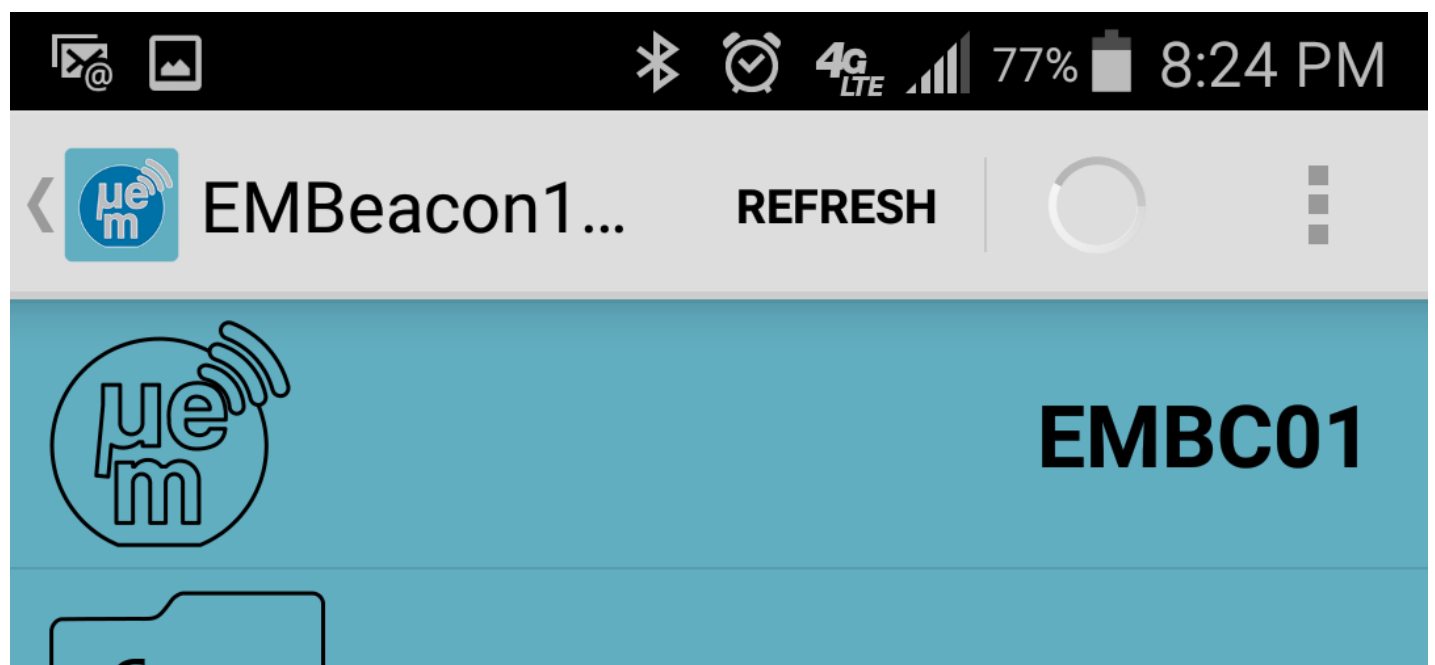
Download and run the latest EMBC Finder app for Android from the Google Play Store. The main screen automatically scans and displays beacons in the immediate area. You should see at least the two - EMBeacon (Sensor) packets and Altbeacon - that we programmed in our example.



Logging

OFF

Now Select "EMB" to see details from the EMBeacon packet. The display should be similar to the following:

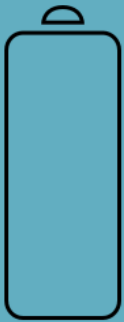




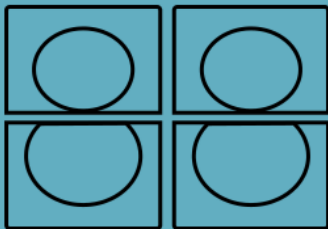
3.0.0



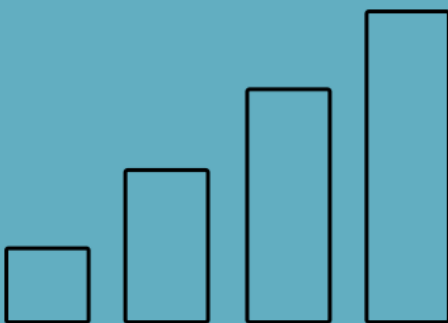
1 Button Presses



3.00 V



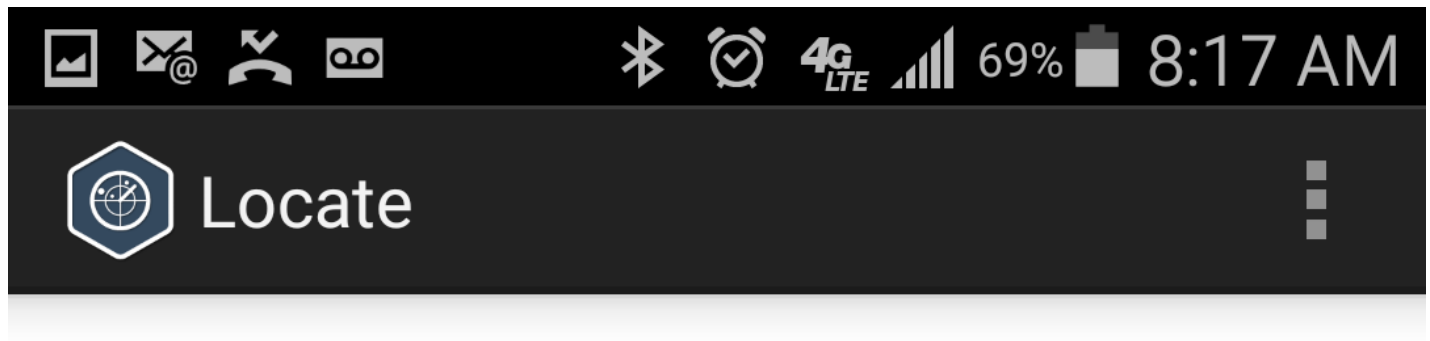
370 Packets



-80 dBm

The EMBC01/02 are standards-based beacons capable of transmitting a variety of well-known packet types detectable by many Android and iOS apps. Here are just a couple of suggestions.

The Radius **Locate** app, available for Android and iOS, does a nice job of detecting AltBeacon, iBeacon, and Eddystone packets.



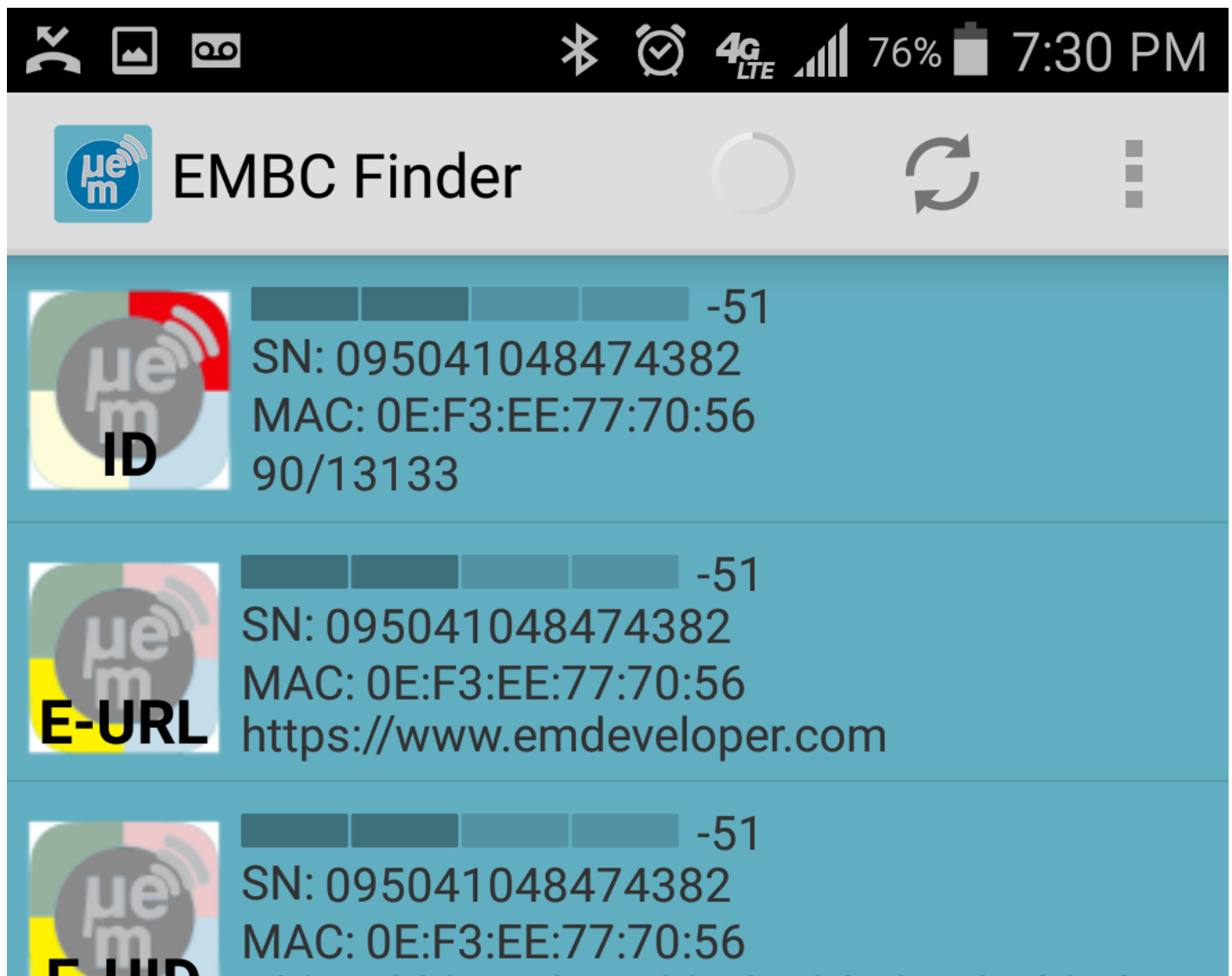
Beacon Type: Eddystone
0x00010203040506070809
ID2: (Beacon ID) 0x0a0b0c0d0e0f
proximity: near
(1.83 meters)
RSSI: -84
Tx Power: -71
Telemetry version: 0
Battery: 13.3 V
Temperature: 51.4 degrees C
Advertisement Count: 0

Uptime: 85570355.2 secs

Distance to Beacon

Calibrate Beacon

The **EMBC Finder** App for Android detects all of the EMBC01/02 packet types. In addition, its the only app that displays detailed sensor readings (battery, acceleration, drop count, etc.) from EM Sensor packets.



E-UID

699EBC80E1F311E39A0F-0CF3EE3BC012



■■■■■ -59

SN: 095041048474382

MAC: 0E:F3:EE:77:70:56

Logging

OFF

If you plan to do BLE Application or FW development, treat yourself to a sniffer. We like the TI CC2540EMK-USB shown here detecting a sequence of AltBeacon and Sensor Data packets. When you believe your beacon is transmitting but your application is not detecting, that's when a sniffer comes in handy. It helps you determine whether or not the beacon is transmitting, and if so what the actual packet contents are. In this way you can quickly zero in on the cause when your devices aren't communicating.

P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
					Type	TxAdd	RxAdd	PDU-Length					
1	+0	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	0F 09 45 4D 42 65 61 63 6F 6E 31 32 38 37 37 00	0xB90C0F	-66	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
2	+50427	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	02 01 06 1B FF 5A 00 BE AC 69 9E BC 80 E1 F3 11	0xB90D8BC	-66	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
3	+50364 =100791	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	0F 09 45 4D 42 65 61 63 6F 6E 31 32 38 37 37 00	0xEE6CAC	-66	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
4	+50398 =51189	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	02 01 06 1B FF 5A 00 BE AC 69 9E BC 80 E1 F3 11	0xB90D8BC	-67	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
5	+51430 =202619	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	0F 09 45 4D 42 65 61 63 6F 6E 31 32 38 37 37 00	0x17CD49	-67	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
6	+50445 =253064	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	02 01 06 1B FF 5A 00 BE AC 69 9E BC 80 E1 F3 11	0xB90D8BC	-67	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI (dBm)	FCS
7	+50468 =253304	0x25	0x8E89BED6	ADV_NON_CONN_IND	2	0	0	37	0x0EF3EE777057	0F 09 45 4D 42 65 61 63 6F 6E 31 32 38 37 37 00	0x40AD3A	-66	OK

Questions, Comments, Errors in this article?

This article explored a small subset of the options available for programming and updating EM Microelectronic COiN beacon products. For the complete and most up to date list of options, please download the FW SDK and review ParameterDictionary.pdf in the util directory. To go further and modify the firmware, you'll need the Ride7 IDE. Ride7 is freely downloadable following registration on the Raisonance site. You can view the Ride7 download & installation instructions here:

<https://forums.emdeveloper.com/downloads/6819/InstallationAndQuickStartEM6819.pdf>

For questions and comments, please post to forums.emdeveloper.com or send email to support@emdeveloper.com. Your feedback is appreciated.

EMBC01/02 Programming QuickStart, Last updated 02 August 2016