BlueBeacon Gateway
Reader of BLE beacons with LAN/WLN interface

Datasheet and user manual

BlueBeacon Gateway is a reader for Bluetooth Low Energy (BLE) devices used to interface BLE beacons/tags networks with the cloud, via LAN or WiFi network (802.11.n).

BlueBeacon Gateway operates as a BLE reader capable of reading the unique identifiers of the beacons within the coverage area and making them available on the cloud (via LAN or WLAN). Through special commands, it is possible to filter the received beacons, for instance, on the technology (iBeacon or Eddystone) or their identifier (UUID, major, minor for iBeacon, ID namespace and instance for Eddystone).

BlueBeacon Gateway is available in two versions: RB/O BLE with omnidirectional antenna, and RB/D with BLE directive linearly polarized antenna with 14dBi gain.

Possible BlueBeacon Gateway applications are:
- remote monitoring of installation of beacons (such as museums, shopping centers, etc ....);
- identify and locate (at specific points) BLE beacons used as active RFID tags applied to objects or people inside a specific environment.

### Bluetooth radio

<table>
<thead>
<tr>
<th>Version</th>
<th>Bluetooth Low Energy (Bluetooth 4.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antenna</td>
<td>Two options:</td>
</tr>
<tr>
<td></td>
<td>RB/O: integrated chip antenna</td>
</tr>
<tr>
<td></td>
<td>RB/D: directive 14dBi linearly polarized panel antenna</td>
</tr>
</tbody>
</table>

### Electrical specifications

<table>
<thead>
<tr>
<th>Power voltage</th>
<th>5V DC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC current</td>
<td>&lt;2.5A (max.)</td>
</tr>
<tr>
<td>Data interface</td>
<td>Ethernet / 802.11.n WLAN</td>
</tr>
<tr>
<td></td>
<td>4x USB</td>
</tr>
<tr>
<td></td>
<td>40x GPIO (SW configurable)</td>
</tr>
</tbody>
</table>

### Mechanical and environmental specifications

<table>
<thead>
<tr>
<th>Size</th>
<th>RB/O: 100 x 73 x 30 mm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RB/D: 185 x 185 x 65 mm</td>
</tr>
<tr>
<td>IP protection</td>
<td>RB/O: IP40</td>
</tr>
<tr>
<td></td>
<td>RB/D: IP67</td>
</tr>
<tr>
<td>Operating temperature</td>
<td>10°C to +50°C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version RB/O</th>
<th>Version RB/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>![RB/O Image]</td>
<td>![RB/D Image]</td>
</tr>
</tbody>
</table>
Hardware

BlueBeacon Gateway is based on Raspberry Pi 3 Model B Single Board Computer (SBC). For documentation on the SBC, please visit the official product website: https://www.raspberrypi.org/

Configuration

The default network configuration uses DHCP. Once identified the IP address acquired by the SBC, log on the BlueBeacon Gateway can be performed via ssh (for instance, using Putty on Windows). Username and password are not changed from the default ones:
- username: pi
- password: raspberry

Once you logged into your home directory you will find two sample applications for NodeJs:
1. `bluebeacon_scanner.js`
   This application creates a TCP server that can be used to receive beacons data
2. `example/example.js`
   This application creates a Web server that shows a real-time plot of beacons RSSI (Received Signal Strength Indicator)

BlueBeacon library for Node.js

The BlueBeacon library for Node.js allows to identify the BlueUp beacons (BlueBeacon) which are present within the receiving range of BlueBeacon Gateway.

The library emulates the operation of the iOS CoreLocation framework for beacon monitoring and ranging (https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/).

In order to use the library within a Node.js application, it is sufficient to call the require() of the library at the beginning of the *.js file:

```javascript
var BlueBeacon = require('bluebeacon');
```

To scan for beacons, simply call the function:

```
BlueBeacon.startScanning(options, noDuplicates);
```

with the following parameters:

1. options: (optional) javascript object which includes parameters for filtering beacons:
   - `null`: no filtering (receive beacons in both iBeacon and Eddystone format);
   - `options.type = "ibeacon"`: only iBeacon beacons;
   - `options.type = "eddystone"`: only Eddystone beacons;
2. `noDuplicates`: boolean value that indicates if you want to receive all packets received from a beacon (false) or only the first one (true).

The library return an event whenever the BlueBeacon Gateway receives a beacon frame:

```javascript
bluebeacon.on('discover', (beacon) => {
    ...
});
```

where the beacon parameter is an object with the following properties:

- `bdaddr [String]`: Bluetooth MAC address
- `rssi [Number]`: received signal strength indicator
- `timestamp [String]`: beacon frame reception timestamp
• model [Number]: BlueBeacon model
• serial [Number]: BlueBeacon serial number
• period [Number]: advertising interval [ms]
• txpower [Number]: beacon transmission power [dBm]
• battery [Number]: battery residual capacity [%]
• type [String]: type of beacon “iBeacon” o “eddystone”
  ○ ”iBeacon”
    ▪ uuid [String]: iBeacon Proximity UUID
    ▪ major [Number]: iBeacon Major Number
    ▪ minor [Number]: iBeacon Minor Number
    ▪ measuredPower [Number]: calibrated RSSI at 1mt (according to iBeacon specification)
    ▪ accuracy [Number]: estimated accuracy for the beacon
    ▪ proximity [String]: proximity indicator: far [ > 5m ], near [2m - 5m], immediate [0 – 2m])
  ○ “eddystone”
    ▪ url [String]: Eddystone URL
    ▪ nid [String]: Eddystone Namespace ID
    ▪ bid [String]: Eddystone Instance ID
    ▪ measuredPower [Number]: calibrated RSSI at 0mt (according to Eddystone specification)
    ▪ temperature [Number]: beacon temperature [°C]
    ▪ adv_counter [Number]: advertised frames from power-up [count]
    ▪ sec_counter [Number]: time from power-up [sec]

Example 1: iBeacon beacon
{
  "bdaddr": "fc:29:c1:7f:dc:29",
  "rssi": -93,
  "timestamp": "2016-06-16T09:20:52.033Z",
  "model": 1,
  "serial": 10384,
  "txpower": -20,
  "period": 500,
  "battery": 100,
  "connectable": true,
  "type": "ibeacon",
  "uuid": "acfd065ec3c011e39bbe1a514932ac01",
  "major": 0,
  "minor": 14,
  "measuredPower": -74,
  "accuracy": 11.000753003614028,
  "proximity": "far"
}

Example 2: Eddystone beacon
{
  "bdaddr": "c1:b3:77:28:85:f1",
  "rssi": -69,
  "timestamp": "2016-06-16T09:18:16.130Z",
  "model": 1,
  "serial": 11727,
  "txpower": 4,
  "period": 700,
"battery": 100,
"connectable": false,
"type": "eddystone",
"temperature": 33,
"adv_counter": 1139684,
"sec_counter": 696475,
"measuredPower": -12,
"url": "https://www.blueupbeacons.com"
}

The library also provides a function that emulates the behavior of the beacon Ranging function in iOS:

BlueBeacon.startRanging(regions);

where the regions parameter can be:

- null, to make the ranging of any beacon;
- an array of the region type objects.

A region type object is a JavaScript object with the following fields:

- `uuid` [String]: iBeacon Proximity UUID
- `major` [Number]: iBeacon Major Number
- `minor` [Number]: iBeacon Minor Number

Example 3: region type object with UUID, major and minor

```
  "regions": [{
    "uuid": "ACFD065EC3C011E39BBE1A514932AC01",
    "major": 0,
    "minor": 14
  }]
```

Example 4: region type object with UUID and major

```
  "regions": [{
    "uuid": "ACFD065EC3C011E39BBE1A514932AC01",
    "major": 0
  }]
```

Example 5: region type object with UUID

```
  "regions": [{
    "uuid": "ACFD065EC3C011E39BBE1A514932AC01"
  }]
```

Note: The ranging function can be used only with beacons with iBeacon technology and returns an event every 1 second

```
bluebeacon.on('range', (beacons) => {
  ...
})
```

where the parameter `beacons` is an array of beacon objects.
Example A: BlueBeacon library with Node.js

This example shows how to create a TCP Server with Node.js, to transmit beacon data received through BlueBeacon library. The application does not create a Web Server but a raw TCP Server, you can connect to via netcat-type software.

TCP Server uses port 8888.
TCP Server accepts commands in JSON format and sends the beacon data in JSON format.

1. log on BlueBeacon Gateway with ssh (Putty):
   ```
   ssh pi@xxx.xxx.xxx.xxx
   ```
2. start the application
   ```
   node bluebeacon_scanner.js
   ```
3. connect to the TCP server on port 8888 through netcat software
   ```
   nc xxx.xxx.xxx.xxx 8888
   ```
4. start beacon scan
   ```
   { "command" : "stream" }
   ```
5. with active beacons in the receiving region of the BlueBeacon Gateway, you start to receive the flow of data in JSON format

Example B: BlueBeacon library with Node.js and Socket.io

In the home directory of BlueBeacon Gateway SBC, you will find a sample program that demonstrates the use of the BlueBeacon library in combination with Socket.io library for creating WebSocket with Node.js. The sample application creates a tiny Web Server that provides a web page that shows a real-time plot of the beacon RSSI. A Start/Stop button on the web page allows to start/stop receiving data via WebSocket.

1. log on BlueBeacon Gateway with ssh (Putty):
   ```
   ssh pi@xxx.xxx.xxx.xxx
   ```
2. Access to “example” directory
   ```
   cd example
   ```
3. start the application
   ```
   node example.js
   ```
4. open the URL with a browser, using the following address
   ```
   http://xxx.xxx.xxx.xxx:8000
   ```
5. Press Start to receive beacon RSSI data

BlueUp reserves the right to change product specifications at any time without prior notice.

iBeacon is a trademark of Apple Inc., Eddystone is a trademark of Google Inc.