SemBeacon: A Semantic Proximity Beacon Solution for Discovering and Detecting the Position of Physical Things

Van de Wynckel, Maxim; Signer, Beat

Published in:
Proceedings of IoT 2023, 13th International Conference on the Internet of Things, Nagoya, Japan, November 2023

DOI:
10.1145/3627050.3627060

Publication date:
2023

License:
Unspecified

Document Version:
Accepted author manuscript

SemBeacon: A Semantic Proximity Beacon Solution for Discovering and Detecting the Position of Physical Things

Maxim Van de Wynckel
mvdewync@vub.be
Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
Brussels, Belgium

Beat Signer
bsigner@vub.be
Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
Brussels, Belgium

ABSTRACT

Discovering smart devices in the physical world often requires some type of indoor positioning system. Bluetooth Low Energy (BLE) beacons are a well-established technique to create scalable low-cost positioning systems for indoor navigation, tracking and location awareness. While various BLE specifications aim to provide a generic way to uniquely identify a beacon and optionally detect its location, they are either deployment specific or do not broadcast enough information to be used without a proprietary database containing the locations of installed beacons. We present a novel BLE advertising solution and semantic ontology extension called SemBeacon that is backwards compatible with existing specifications such as iBeacon, Eddystone and AltBeacon. With the help of a prototype application, we demonstrate how SemBeacon enables the creation of real-time positioning systems that can describe their location as well as the environment in which they are located. In contrast to Eddystone-URL beacons which were originally used in Google's Physical Web project to broadcast web pages of physical objects, SemBeacon is a specification for broadcasting semantic data about the environment and positioning systems that are available within a beacon's proximity using linked data.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Sensor networks**; Location based services; • **Networks** → *Location based services*; • **Hardware** → Sensor applications and deployments.

KEYWORDS

Bluetooth, beacons, semantic location, specification, physical things

ACM Reference Format:
Maxim Van de Wynckel and Beat Signer. 2023. SemBeacon: A Semantic Proximity Beacon Solution for Discovering and Detecting the Position of Physical Things. In *The International Conference on the Internet of Things (IoT 2023), November 7–10, 2023, Nagoya, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3627050.3627060

IoT 2023, November 7–10, 2023, Nagoya, Japan

ACM ISBN 979-8-4007-0854-1/23/10…$15.00
https://doi.org/10.1145/3627050.3627060

1 INTRODUCTION

The discovery of smart devices in indoor environments goes beyond detecting their presence and often requires users to find and track these devices in the physical world. This level of tracking requires different positioning techniques [6, 19], as indoor positioning does not rely on a single standard. One of the early and still widely used technologies are Bluetooth Low Energy (BLE) beacons due to their low cost and good battery performance [14]. They send out uniquely identifiable information which, combined with the beacons' received signal strength, can be used to determine the location relative to one or more beacons. This type of relative positioning requires knowledge of the transmitted signal strength as well as the physical location of the beacons.

While some specifications let the beacons broadcast both their transmitted signal strength and physical location, these locations usually lack contextual information such as a reference frame, the layout of the building or information about the use of the beacon or smart device. In most implemented indoor positioning systems, the beacons are stored in a database but not publicly shared among other applications or services. Beacon specifications such as the Bluetooth Indoor Positioning Service [8] let beacons broadcast both a WGS84 location and a *local* location, but do not include information on how this local location should be interpreted.

We present a BLE advertising and semantic vocabulary specification called *SemBeacon*, where Bluetooth beacons advertise their semantic description on the Semantic Web [1] while maintaining backwards compatibility with existing specifications commonly used for indoor positioning and location awareness. Similar to earlier work such as Hewlett Packard's CoolTown™ project [26] which broadcasted semantic locations via infrared beacons, we broadcast a URI describing our beacon via BLE. Our solution supports the design of semantic environments that can be detected by any application that is able to detect, retrieve and reason on the SemBeacon data. This enables the discovery of smart devices, indoor positioning systems and other semantic information of a building. Our solution can help to discover stationary or movable Physical Things on the Web of Things [35] while also helping users to position themselves within the environments these *Things* are located in.

We demonstrate the use and performance of SemBeacons using an open source prototype application that discovers beacons and their geospatial environments. Our demonstrator showcases how beacons within the same spatial area can be discovered in real-time without having to perform intensive querying on semantic data or without the need of a predefined database or Web Service. With our extensions of the POSO ontology [33] we further enable the creation of interoperable indoor positioning systems.

2 BACKGROUND AND RELATED WORK

The semantic description of people, places and objects has already been investigated in Hewlett Packard's CoolTown™ project [26] in the early 2000s. The *semantic locations* introduced as part of HP CoolTown were represented as URIs that were encoded within barcodes, physical locations using a trusted service or CoolTown beacons. These beacons were battery-powered infrared transmitters that could broadcast a URI pointing to an XML resource with semantic data every three seconds [16].

Various related work focused on semantic location-based services [15, 18, 24] with a primary focus on navigation and environment description. However, they do not offer a method to advertise the availability of this location-based service in a physical environment or details on how applications should determine a location within these environments. Due to this limitation, applications require prior knowledge to discover environments and smart devices. Mathew et. al [21] discussed several methods such as application interfaces, knowledge servers or open ontologies, but these methods only support the discovery of semantic descriptors of devices on the Web rather than in the physical world.

Google started the *Physical Web* project in 2014 to enable seamless interactions with physical *things* and locations [22, 27]. It was based on Eddystone-URL beacons to broadcast URLs for physical objects. Compatible smartphones would receive a notification when they were near a physical thing broadcasting a URL, and the broadcasted URLs enabled direct user interaction without the need for additional applications.

With Bluetooth beacons being a well-established technology for indoor positioning systems [14], in the following sections we focus on existing efforts to realise indoor positioning systems by using BLE advertisements. BLE beacons send out *advertisements* at a fixed interval that can be picked up by any BLE receiver in range that is performing a passive scan. A BLE receiver can also perform an active scan that first sends out a scan request. Active scanning allows a response from devices that do not automatically send out advertisements and enables these BLE beacons to send out different advertisement data for passive and active scans. The proximity to the beacon is estimated by means of a radio frequency (RF) path loss model on the received signal strength indicator (RSSI) [13] which estimates the signal loss of a transmitted signal.

2.1 Positioning Techniques

Ever since the release of Bluetooth Low Energy, BLE beacons have been one of the more prominently used techniques for indoor positioning [2]. For their use in indoor positioning systems, they are often installed as small battery-powered devices in rooms, hallways or other key areas [5] but can also be embedded in smart devices [12]. During the installation, the beacon's physical location in the building is linked to its advertised identifier.

One of the simplest positioning techniques using beacons is called cell identification. With this technique, the proximity to a single beacon or a group of beacons is used to determine the receiver's position [19]. We assume that the beacons with the best signal strength are the closest. When multiple beacons are in range, multilateration can be used to compute the absolute position based on three or more beacons. In the context of positioning systems, the known fixed location of beacons is used to determine the location of a user scanning for these beacons.

Regardless of the used technique, devices are capable of computing their relative location between one or more beacons, but require knowledge of the location of these beacons in order to compute their absolute location within a building.

2.2 Bluetooth IPS Specification

In 2015, the Bluetooth SIG published a core specification for advertising an indoor positioning service via BLE [8]. This advertisement specification allows broadcasting the global WGS84 location [4], local coordinates within a building, the transmission power and additional information needed to know the transmitter's location.

The Bluetooth IPS specification describes the broadcast of the location data in 20 bytes and GATT services to configure these properties. Any changes made to the location have to be configured in the hardware itself, making remote changes impossible without additional network communication. While the specification broadcasts a location in a global reference frame and local reference frame, no information is broadcast on how to interpret these local coordinates.

2.3 iBeacon Specification

The iBeacon specification was developed by Apple Ltd. in 2013. In the specification, a total of 30 bytes is used for the *manufacturer data* to encapsulate the information it broadcasts. As illustrated in Figure 1, the iBeacon specification offers three layers of device identification (dark blue) and adds an additional byte for the reference transmission power at one metre distance. A 128-bit *proximity UUID* identifies all beacons used by the same positioning system or application. Next, an unsigned 16-bit *major* and *minor* number identify the hierarchy of the beacon with a spatial region [7]. The specification does not mention at what spatial level the hierarchical separation of the major and minor identifiers should be chosen. However, most commonly the major number defines the building or floor and the minor number represents the beacon within that floor. The iBeacon specification does not advertise a beacon's location and requires a database or a known trusted service to provide this information.

Adv Flags 3B	Len 1B	Type 1B	Company ID 2B	Beacon Type 1B	Beacon Len 1B	Proximity UUID 16B	Major 2B	Minor 2B	TX @ 1m 1B	
-	0x1A	0xFF	0x4C00	0x02	0x15	*uint8[]*	*uint8*	*uint8*	*int8*	

Figure 1: iBeacon advertisement data (30 bytes)

2.4 Eddystone Specification

Eddystone was developed by Google in 2015. Other than the iBeacon specification, it does not use custom manufacturer data, but creates a service with its own unique UUID containing the beacon information [25]. The specification offers four types of *frames*:

- **Eddystone-URL** for broadcasting short URL addresses.
- **Eddystone-UID** for broadcasting beacon identifiers similar to iBeacon. Uses a namespace and instance identifier instead of a service UUID, major and minor number.

- **Eddystone-TLM** for broadcasting telemetry data such as sensor data or battery information.
- **Eddystone-EID** is a security specification for broadcasting ephemeral identifiers.

Figure 2 shows the Eddystone specification for UID and URL Eddystone frames. In order to limit the size of the broadcasted URL in an *Eddystone-URL*, the specification adds 1 byte to specify the URL scheme prefix (e.g. `0x00` matches `"http://www."` or `0x01` matches `"https://www."`). The URL itself is a UTF-8 character array of the URL without a scheme. Commonly used UTF-8 character groups such as top-level domains can be encoded using a single byte (e.g. `0x00` matches `".com/"`) as detailed in the specification[1].

Despite the encoding techniques, a URL shortening service is often needed in order to obtain a URL that fits within the frame. This short URL redirects to the full URL, preventing the encoding of identifiable information within the domain or path without performing an HTTP request.

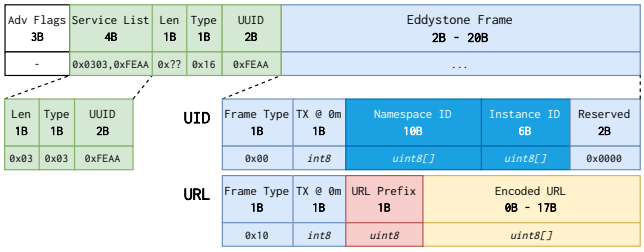

Figure 2: Eddystone-UID and Eddystone-URL advertisement data (13–31 bytes)

Multiple frames can be combined to provide additional data. For example, the Eddystone-UID frame can be combined with a scan response containing telemetry data to return additional data while identifying a beacon with its namespace and instance identifiers.

Using the Eddystone-URL frame, Seo and Yoo [30] proposed an interoperable context model advertising *Place*, *Object* and *Annotation* URIs. These URIs led to HTML web pages providing augmented reality context via HTML elements. For its use within the context of interoperable positioning systems, there is a lack of identifiable information that would prevent applications from having to access the URI of every encountered Eddystone-URL beacon.

2.5 UriBeacon

As predecessor of the Eddystone-URL beacon frame [20], UriBeacon has a similar maximum URI byte size as Eddystone (17 bytes) and uses the same encoding techniques. The main difference to the Eddystone-URL specification is the addition of 1 byte that can be used to add flags to the URI. The specification only implemented one *invisible hint* flag defining whether the URI should be accessed. In Eddystone-URL, this byte is used to indicate the type of Eddystone frame (i.e. `0x10` for Eddystone-URL). Having the ability to add flags to the beacon to indicate how the client should handle or use the URI is an important and useful feature in real-time systems, which we also decided to use in our own solution.

[1]https://github.com/google/eddystone

2.6 AltBeacon Specification

AltBeacon is an open specification by Radius Networks[2] that is backwards compatible with iBeacon scanners not specifically scanning for beacons manufactured by Apple Ltd. However, unlike iBeacon's identification through a proximity service UUID and a major and minor classification, AltBeacon uses 20 bytes as a single beacon identifier.

Figure 3: AltBeacon advertisement data (31 bytes)

While iBeacon only uses 30 bytes, the last byte in AltBeacon was added to the specification as a manufacturer-specific byte. The specification recommends using a UUID for the beacon identifier's first 16 bytes in a single *organisational unit*, enabling backwards compatibility with iBeacon scanners looking for a proximity UUID.

An AltBeacon is identified as such by setting bytes 7–8 to the `0xBEAC` hexadecimal representation which would indicate the beacon type and remaining payload length for iBeacon. In AltBeacon, these two bytes are called the *beacon code* as illustrated in Figure 3. Similar to iBeacon, the AltBeacon specification does not include information about a beacon's location. Unlike the AltBeacon specification which recommends using the same 128-bit UUID for the same organisational unit, we aim for a specification where beacons are not specifically deployed for a single proprietary application.

All the presented prominent beacon specifications used for positioning and proximity awareness are built on top of Bluetooth v4.2 compatible advertisements. While these advertisements are lightweight and sufficient to broadcast the required information, they are limited when used without an additional database or Web service mapping the beacons to other contextual data. In our own solution, we have considered the backwards compatibility with the prominent beacon specifications, as well as some of their characteristics and features that could be useful for semantically describing the locations and environments in which these beacons are placed.

3 SEMBEACON SPECIFICATION

The general principle of our *SemBeacon* specification is that uniquely identified beacons have a position and additional semantic data available on the Web which is accessible via a Unique Resource Identifier (URI), describing the beacon and its deployed environment. Beacons broadcast this URI via an advertisement, allowing any application receiving the advertisement to know the position of the beacon. Because indoor positioning systems often detect multiple of these battery-powered beacons in rapid succession, our specification is designed to limit the amount of network connections needed to retrieve information about the location of beacons and the amount of data these beacons have to broadcast. Our specification is fully backwards compatible with AltBeacon and iBeacon [17, 28] and Eddystone-URL scanners which allows existing buildings to gradually add SemBeacons to cover their spatial area.

While the Eddystone-URL specification can be used to broadcast a URI of a semantic resource providing more information about

[2]https://github.com/AltBeacon/spec

an object, there is a lack of information for its use in real-time positioning systems. An Eddystone beacon's MAC address can be used to uniquely identify an object. However, as the URL needs to be a short encoded URL in order to fit within the 17 bytes, the information that can be encoded within this URL is limited. Due to this limitation, beacons belonging to the same spatial area cannot be extracted accurately unless a local database is maintained. Being able to identify a set of beacons belonging to the same spatial area or *namespace* prevents us from having to perform network requests to retrieve this information. Combining Eddystone frames such as a UID and URL frame in an advertisement packet and scan response could offer similar capabilities as SemBeacon when linking to semantic data, but due to their packet size it is not possible to encode additional information about the type of SemBeacon that is being deployed (see Section 3.1.2).

With our solution, we also aim to create interoperable indoor positioning systems where users can obtain the location of a beacon as well as any other sensors or configured positioning systems in a building without prior knowledge of the environment. Our proposed solution for creating semantic Bluetooth Low Energy (BLE) beacons is divided into two main sections. While Section 3.1 describes the advertisement specification based on AltBeacon and Eddystone-URL, Section 3.2 introduces the semantic description of SemBeacons on the Web, including the use of the POSO [33] vocabulary for describing indoor positioning systems.

Our main advantage with SemBeacon is the ability to operate without the need of a proprietary application that defines a database or Web Service to map the beacons to contextual data. Our vocabulary allows the description of positioning systems which in turn helps applications to find these beacons in an indoor environment where GPS can not be used.

3.1 Advertisement Specification

For our semantic beacon specification we had several design requirements. The specification should be compatible with both the iBeacon as well as AltBeacon specifications to be recognised by existing deployed indoor positioning systems. In addition, existing infrastructures consisting of hardware beacons whose protocol cannot be altered should also be supported, regardless of the type or manufacturer of those beacons. Our advertising packet payload shown in Figure 4 is based on the AltBeacon specification which is backwards compatible with iBeacon. In our online technical specification[3] we also introduce a BLE v5 compatible advertisement leveraging the additional payload size and range of BLE v5 [9].

3.1.1 Identification. A SemBeacon is recognised as an AltBeacon (i.e. bytes 7–8 should be `0xBEAC`) that offers an Eddystone-URL compatible scan response. Individual SemBeacons are identified as a combination of a 16-byte *namespace identifier* and a 4-byte *instance identifier* that is unique for all beacons in the same namespace. The namespace is a 128-bit universally unique identifier (UUID) that is unique per spatial area where beacons are deployed, including buildings or even floors. The instance identifier is a 32-bit (unsigned big endian integer) uniquely identifying all beacons within the same namespace. Splitting the identifier in a namespace and instance

[3]https://sembeacon.org/docs/

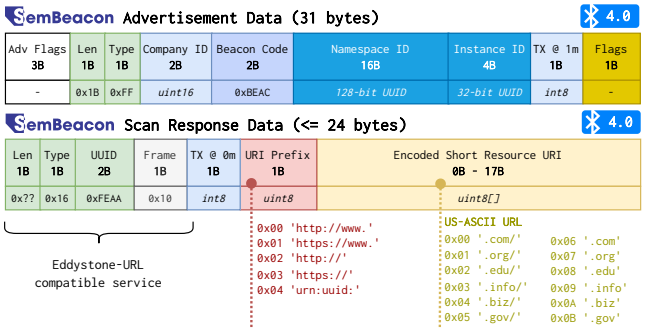

Figure 4: SemBeacon specification based on AltBeacon with an Eddystone-URL compatible scan response

is inspired on Eddystone-UID. However, unlike Eddystone-UID which recommends that the namespace is used for filtering beacons during the scanning (i.e. for a specific application), we utilise the namespace to remember which semantic resources have already been retrieved.

Similar to the iBeacon specification which does not specify the spatial relation between its major and minor identifiers, SemBeacon does not force the use of a certain spatial hierarchy between namespaces and instance identifiers. However, all beacons that can be fetched with a single HTTP GET request should be considered to be in one *namespace*, offering users the freedom to decide how to group beacons in a single resource.

With our aim for interoperability, newly developed BLE scanners should not filter on a specific namespace identifier but rather perform filtering after obtaining the semantic data. However, existing implementations that scan for certain *proximity UUID* iBeacons can be supported by using the same 16-byte UUID for the namespace identifier. These existing implementations will not be able to leverage the broadcasted URI, but can still use an internal database. Applications that want to implement filtering of SemBeacons used by one positioning system can use a prefix for the 128-bit UUID.

3.1.2 Semantic Flags. With SemBeacons we aim to provide most, if not all information on the Semantic Web [1] via the broadcasted URI. However, some beacons might not require an HTTP request to the URI if the online information is guaranteed to be unusable for the application that detects the beacons. For example, an application designed to use SemBeacons as landmarks for indoor positioning cannot make use of beacons that are attached to movable objects.

The final byte of the SemBeacon advertisement is a manufacturer-specific byte in AltBeacon. SemBeacon uses this byte to provide boolean information about the type of SemBeacon and the expected available online data, which can be used to decide whether to access the resource URI depending on the scanning application.

In Table 1 we provide an overview of the available SemBeacon flags which are based on the Bluetooth Indoor Positioning Service [8] flags, Eddystone frames and the UriBeacon [20] flag. While they repeat certain information that would be available online, the flags depend on the deployment of the beacon and would not need to change after deployment.

Bit	Description
0	Indicates if beacon has a position (0 = unsure, 1 = yes).
1	Indicates if beacon is private (0 = public, 1 = private).
2	Indicates if beacon is stationary (0 = stationary, 1 = mobile).
3	Indicates if beacon is part of a positioning system (0 = no, 1 = yes).
4	Indicates if beacon provides telemetry data (0 = no, 1 = yes).
5–7	Reserved for future use.

Table 1: SemBeacon flags

To support the positioning outlined in Section 2.1, each beacon requires a position. During the installation of beacons in the physical environment, the first bit would be set to 1 to indicate that the beacon is installed at a fixed known location. In a use case where these beacons are providing information about other beacons in the environment, they do not necessarily require a location.

If users or assets are equipped with a SemBeacon, these moving objects (indicated by bit 2) can be tracked. For some assets or persons, the semantic description might require authentication to limit access to the URI. In this case, the private flag at bit 1 can be toggled on. A use case would be that the semantic data is stored in a Solid *Pod* [29]. Solid (Social Linked Data) lets users have their own data vault called a Pod containing public or private semantic data. Users can decide to share this data with other services, including broadcasting it with SemBeacon so it is only accessible via (authenticated) users or objects within their proximity.

Finally, interoperable indoor positioning applications might only be interested in SemBeacons which are placed for indoor positioning systems, in which case bit 3 can be toggled to indicate that the beacon forms part of a positioning system. In addition, the semantic description might provide additional information on implemented positioning techniques that use other sensors such as Wi-Fi fingerprinting or Ultra-wideband beacons [6, 19, 31].

3.1.3 Scan Response Payload. Bluetooth Low Energy can respond with a scan response payload in addition to the advertisement data. The scan response consists of an additional 31 bytes of data that is sent whenever a scan request is received. In Figure 4, we see an Eddystone-URL compatible SemBeacon URI in the scan response. We have chosen this frame as a legacy scan response due to its implementation in existing libraries, applications and hardware. During the setup of the beacons, the URI can be put behind a linked data frontend that serves a web page when the URI is visited by the browser of users who are scanning for Eddystone-URL beacons. These websites might contain a visual representation of the semantic data as in the Physical Web [22]. However, unlike the Physical Web, our main goal is to access the URI via software.

A SemBeacon URI should resolve to a machine-readable document where the data is structured using triples consisting of *subjects*, *objects* and *predicates* that provide the relation between a subject and object. The namespace identifier should match with respect to the base URI, ensuring that the namespace identifier matches the base URI for all beacons within this document.

3.2 Semantic Description

For the design of our vocabulary to semantically describe beacons, we have focused on the use case where SemBeacons are used as proximity beacons in positioning systems and location-aware applications, but additional functionality and data can be added as with any vocabulary. We have chosen the Positioning System Ontology (POSO) [33] as our core ontology. POSO uses the Semantic Sensors Network (SSN) and Sensor, Observation, Sample and Actuator (SOSA) ontologies [10, 11]. It offers the ability to describe a positioning system and its internal workings, the deployment, entities tracked by the system and sensor information. The ontology already offers the terminology to describe *landmarks* used in a positioning system, such as beacons or other RF transmitters. We extended POSO (`poso-common`) with an additional vocabulary for describing the different types of beacons including SemBeacons and their related information.

As outlined in Section 3.1.1, the identification of a semantic beacon is done on a *namespace* and *instance* level. Namespace identifiers resemble the base URI of a semantic resource. In our online technical specification, we make it clear that multiple beacons within the same resource (which are retrieved with a single HTTP GET request) should use the same namespace.

http://sembeacon.org/example.ttl

```
1  @prefix :          <http://sembeacon.org/example.ttl#> .
2  @prefix hardware:  <http://w3id.org/devops-infra/hardware#> .
3  @prefix poso:      <http://purl.org/poso/> .
4  @prefix posoc:     <http://purl.org/poso/common/> .
5  @prefix sembeacon: <http://purl.org/sembeacon/> .
6  @prefix qudt:      <http://qudt.org/schema/qudt/> .
7  @prefix unit:      <http://qudt.org/vocab/unit/> .
8
9  :building_a a ssn:Deployment ;
10   rdfs:label "Building A" ;
11   sembeacon:namespaceId "e19c5e1ed6a14d..."^^xsd:hexBinary .
12
13 :room_a1_2 a sembeacon:SemBeacon ;
14   rdfs:label "SemBeacon Room A1.2"@en ;
15   rdfs:isDefinedBy <http://sembeacon.org/example.ttl#> ;
16   sembeacon:namespace :building_a ;
17   sembeacon:instanceId "beac0101"^^xsd:hexBinary ;
18   hardware:mac "00:11:22:33:44:55" ;
19   posoc:referenceRSSI [ # Reference RSSI is a ...
20     # ... factory calibrated signal strength
21     poso:hasRSS [
22         qudt:unit unit:DeciB_M ; qudt:numericValue -56 ] ;
23     # ... measured at a specific distance
24     poso:hasRelativeDistance [
25         unit:Meter ; qudt:value "1.0"^^xsd:double ] .
26   ] ;
27   poso:hasPosition [ a poso:AbsolutePosition ;
28     poso:hasAccuracy [ ... ] ; poso:xAxisValue [ ... ] ;
29     poso:yAxisValue [ ... ] ; poso:zAxisValue [ ... ] ] .
```

Figure 5: Example SemBeacon description (`example.ttl`)

In Listing 5 we show a basic description of a SemBeacon and its spatial location written as triples using the Resource Description Framework (RDF) [23]. The individual `:room_a1_2` on line 13 is a `sembeacon:SemBeacon` type. An individual is an instance of a class, in this case for a specific *Deployment* [1]. We describe the reference RSSI at 1 metre and the absolute position defined using the POSO ontology. As some positioning systems require the reference RSSI to be measured at 0 metres, the distance can be identified using `poso:hasRelativeDistance` on line 24. Further, we use the Hardware Ontology from the Devops Infrastructure Ontology Catalogue [3] to provide a beacon's physical address, which can be used

by some positioning systems to uniquely identify other types of beacons that do not offer an identifier in their advertisement data.

http://sembeacon.org/example.ttl

```
:room_a1_3 a posoc:iBeacon ;
  rdfs:label "iBeacon Room A1.3"@en ;
  rdfs:isDefinedBy <http://sembeacon.org/example.ttl#> ;
  sembeacon:namespace :building_a ;
  hardware:mac "00:55:44:33:22:11" ;
  posoc:major 10001 ; posoc:minor 12831 ;
  posoc:referenceRSSI [ ... ] ; poso:hasPosition [ ... ] .
```

Figure 6: Example iBeacon description (`example.ttl`)

Listing 6 illustrates how additional beacons defined in the same resource and namespace can be retrieved whenever the resource is fetched from SemBeacons. An iBeacon is defined with a deployment as a common namespace also shown in Listing 5 on line 16. In the online documentation[4] we provide more details on all additions to the vocabulary and its usage. Developers can decide to extend the vocabulary to describe different output data types or even different protocols for interfacing with the beacon or device.

3.3 State

We now provide an overview of the different steps an application has to take to scan and use SemBeacons. All beacons within the same spatial environment will be put in the same resource. Each beacon within this resource will have the same namespace identifier, which removes the need for unnecessary HTTP requests when multiple beacons are detected with the same namespace.

In the following, We list the six different steps of an application scanning for BLE v4 advertisements.

(1) **Passive scanning**: The application passively scans for incoming advertisements until AltBeacon compatible manufacturer data is detected that includes a 128-bit UUID (i.e. namespace identifier).
(2) **Beacon identification**: A beacon is identified with a namespace and instance identifier. The application checks if it has knowledge about the namespace identifier. If the namespace was not previously discovered, an active scan is performed.
(3) **Active scanning**: The application actively scans for new beacons using a scan request.
(4) **SemBeacon detection**: SemBeacons will respond with a scan response that includes the Eddystone-URL compatible resource URI. Compatible scanners will detect a SemBeacon when it has an AltBeacon advertisement and Eddystone-URL scan response.
(5) **Data retrieval**: In case no information about the namespace was available, the resource URI is accessed to retrieve the location of the beacon, as well as other beacons within the same namespace. Depending on the flags shown earlier in Table 1 and the implementation of the application to act on these flags, the data might not be retrieved.
(6) **Passive scanning**: The application continues the passive scan until an unknown namespace is found, in which case step (3) is performed again.

[4]https://sembeacon.org/terms/1.0/

3.4 Performance and Caching

Bluetooth beacons are often deployed as battery-powered devices. These devices send out an advertisement at a fixed interval. In between advertisements, the beacon will lower its controller processing usage in order to save power. Specifications such as the Bluetooth IPS specification [8] describe the ability to connect to the device in order to obtain more information. Connecting to a device is time consuming in a real-time system and requires more battery power.

Our SemBeacon design is similar to other beacon specifications that do not accept connections. We aimed to provide a specification that offers minimal tracking latency to obtain information, provides caching possibilities and can be used on battery-powered devices with no or minimal overhead compared to existing specifications. We maintained the minimal latency by including all the relevant information needed to perform the tracking in both the advertisement and scan response. The advertisement data includes the namespace and instance identifier which are sufficient for an application to perform real-time distance estimation with known beacons. In order to resolve a beacon's location and any other relevant information, a scan response including the resource URI is required which can in turn be mapped to the identifiable information of the advertisement data. Only one network request is required per spatial area since beacons in the same namespace are all kept in the same RDF resource.

The caching of beacon data can be handled by HTTP cache policies of the resource URIs which are already fetched using HTTP requests, allowing the reconfiguration of these policies without changing the physical beacons. With the cache policy being applied to all beacons in the same namespace, this adds an additional level of freedom to separate beacons in different namespaces based on their cache policy or when using movable beacons whose information updates more frequently.

4 DEMONSTRATOR

To demonstrate the deployment and use of SemBeacons, we developed an application[5] that can scan for iBeacon, Eddystone beacons, AltBeacon and SemBeacons. The application will retrieve the environment information that is broadcast by the SemBeacons together with the devices and deployed positioning systems in these environments. The application along with the SemBeacons was tested in a real-world environment.

4.1 Hardware and Software

Our prototype SemBeacons that were deployed within the building, were designed using an ESP32-S3 microcontroller. The Arduino code for creating and configuring a SemBeacon using an ESP32 can be found on GitHub[6]. Our scanner application is developed in Ionic Capacitor[7] which also allows SemBeacons to be discovered via Web Bluetooth Scanning [34] as well as using an Android or iOS device. A native Android library for scanning beacons is available on the SemBeacon GitHub[8].

[5]https://github.com/SemBeacon/sembeacon-app/
[6]https://github.com/SemBeacon/sembeacon-arduino-esp32/
[7]https://capacitorjs.com
[8]https://github.com/SemBeacon/sembeacon-android-library/

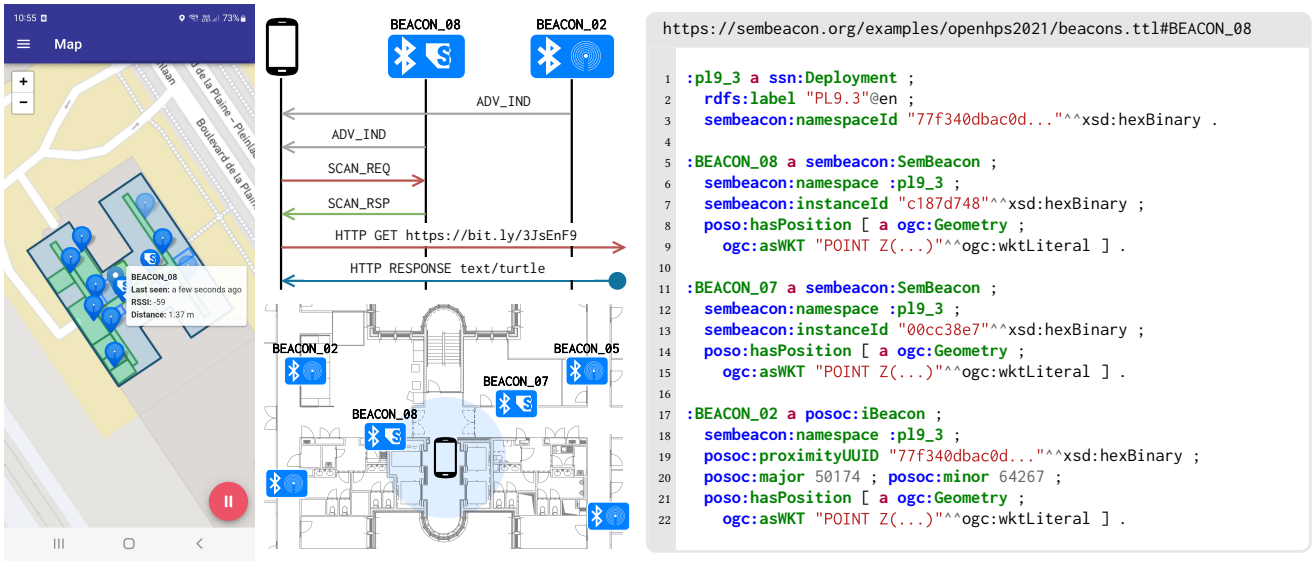

Figure 7: Example scenario using the floor plan and transformed dataset of [32]

4.2 Dataset

For our demonstrator, we transformed and extended an existing indoor positioning dataset [32] to semantic RDF data[9]. We redeployed the beacons of the dataset in the same building but replaced two of the original beacons that are closest to the entrance of the floor with SemBeacons (i.e. `BEACON_07` and `BEACON_08`). Other than existing indoor positioning systems for navigating in indoor spaces or tracking the location of physical objects, our solution does not require prior knowledge of the beacons within this dataset. We can publish the semantic data online and let the two SemBeacons broadcast their resource identifier. Any changes to the environment such as new smart devices or additional details to the geospatial description can be modified online without the need to update the application or reconfigure the beacons.

4.3 Device and Environment Discovery

We have developed a demonstrator application that is scanning for SemBeacons while showing the user a regular map with their own location. When SemBeacons are discovered and their data retrieved, the information they contain is shown on the map which may include an indoor map as well as the beacons or devices themselves. Future applications can expand on this to provide contextual actions on the beacons or devices or to visualise information on the map or when interacting with the marker. Our application uses the POSO ontology to extract the positioning system used within the environment. In our dataset, the positioning is limited to multilateration between the fixed locations of iBeacons and SemBeacons.

Let us now provide a step-by-step example based on the states described Section 3.3. In Figure 7 we illustrate our example scenario using an existing dataset [32] with `BEACON_7` and `BEACON_8` transformed to SemBeacons near the entrance. As soon as a phone arrives in the building, it will passively pick up beacon advertisements from the nearby SemBeacons as well as other beacons advertising on the floor shown as the advertisement indicator (ADV_IND) arrows in grey. Without prior knowledge on the namespace identifier, these advertisements are ignored. Once an AltBeacon-compatible advertisement with an unknown namespace identifier is detected, the phone sends a scan request which triggers SemBeacons to respond with an encoded short resource URI.

After retrieving the short resource URI, the application performs a GET request which in turn returns the resource with the description of the beacon, metadata about the environment and other beacons stored in this resource. On the left-hand side of Figure 7, we show our application visualising the GeoJSON floor layout, the beacons in range and known beacons that are not in range (semi-transparent markers). On the right-hand side, we see the semantic resource describing the SemBeacons as well as any other beacons or sensors in the namespace. In our scenario, we have two SemBeacons each with a unique instance identifier and a position. Other beacons within the same deployment are identified as iBeacons.

```
PREFIX sembeacon: <http://purl.org/sembeacon/>
PREFIX poso: <http://purl.org/poso/>
SELECT ?beacon {
    ?beacon a poso:BluetoothBeacon .
    {
        ?beacon sembeacon:namespaceId "..."^^xsd:hexBinary
    } UNION {
        ?beacon sembeacon:namespace ?namespace .
        ?namespace sembeacon:namespaceId "..."^^xsd:hexBinary
    } .
}
```

Figure 8: Example SPARQL query to retrieve all beacons belonging to the same namespace

In Listing 8, we showcase a simple SPARQL query to demonstrate how our application retrieves all beacons belonging to the same namespace. This can either be a SemBeacon with the `:namespaceId` predicate or a deployment that in turn has a `:namespaceId`.

[9] https://sembeacon.org/examples/openhps2021/beacons.ttl

5 CONCLUSION AND FUTURE WORK

We have presented the *SemBeacon* solution for Bluetooth Low Energy (BLE) advertising and the semantic description of beacons and their environments. Our solution is designed for applications where beacons are used as landmarks to detect an indoor position. The SemBeacon specification is backwards compatible with existing standards to facilitate adoption in existing positioning systems and applications. Our supplemental material further includes a work-in-progress BLE v5 extended advertisement specification which is no longer backwards compatible, but enables new BLE v5 features, including improved battery consumption and larger range, and removes the requirement for scan responses by beacons.

In contrast to existing techniques such as iBeacon that require a database or shared service to perform indoor positioning, we provide a decentralised Semantic Web-based approach. Our solution is backwards compatible with existing techniques and infrastructures, facilitating the transition to SemBeacons. Existing indoor positioning systems can be adapted by converting the location of beacons to semantic data. To make applications aware of this semantic data, SemBeacons can be placed at key areas or be included in smart devices to broadcast the URI describing these areas or devices.

We demonstrated a SemBeacon deployment using an existing dataset that was redeployed in the same building but with additional SemBeacons. We outlined how our solution can help in bridging an environment's semantic online description and the corresponding physical environment. Existing indoor positioning systems and applications can easily be modified to support SemBeacons without major infrastructure changes. Other than providing semantic data for its use in Bluetooth-based indoor positioning techniques, SemBeacons can be used to advertise information about other sensors or positioning techniques used in the environment.

Our SemBeacon specification, along with code examples, documentation and the ontology alignment with POSO is available as open source on GitHub. While the specification is aimed towards BLE beacons, the semantic part of the specification can be extended to other types of beacons such as Ultra-wideband beacons, which might enable future beacons to broadcast URIs without having to change the vocabulary needed to describe them. In addition, our specification supports the broadcasting of information about other sensors and beacons in the deployment, enabling them to describe other hardware needed to perform indoor positioning or location awareness. The extensible vocabulary allows for a large variety of future use cases. We plan to conduct further experiments where SemBeacons are installed alongside multiple existing deployments to demonstrate the transition from proprietary applications to a single interoperable application for interfacing with physical things.

REFERENCES

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The Semantic Web. *Scientific American* 284, 5 (May 2001), 34––43.

[2] Danijel Čabarkapa, Ivana Grujić, and Petar Pavlović. 2015. Comparative Analysis of the Bluetooth Low-Energy Indoor Positioning Systems. In *Proc. of TELSIKS 2015*. IEEE. https://doi.org/10.1109/TELSKS.2015.7357741

[3] Oscar Corcho et al. 2021. A High-Level Ontology Network for ICT Infrastructures. In *Proc. of ISWC 2021*. 446—-462. https://doi.org/10.1007/978-3-030-88361-4_26

[4] B. Louis Decker. 1986. World Geodetic System 1984. In *Proc. of the International Geodetic Symposium on Satellite Positioning*.

[5] Cole Gleason et al. 2018. Crowdsourcing the Installation and Maintenance of Indoor Localization Infrastructure to Support Blind Navigation. *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1 (2018). https://doi.org/10.1145/3191741

[6] Yanying Gu, Anthony Lo, and Ignas Niemegeers. 2009. A Survey of Indoor Positioning Systems for Wireless Personal Networks. *IEEE Communications Surveys & Tutorials* 11, 1 (2009). https://doi.org/10.1109/SURV.2009.090103

[7] Apple Inc. 2015. *Proximity Beacon Specification*. Specification. Apple Inc. https://developer.apple.com/ibeacon/

[8] Bluetooth SIG Inc. 2015. *Indoor Positioning Service*. Specification. Bluetooth SIG Inc. https://bluetooth.com/specifications/specs/indoor-positioning-service-1-0/

[9] Bluetooth SIG Inc. 2020. *Bluetooth Core Specification v5.1*. Specification. Bluetooth SIG Inc. https://bluetooth.com/specifications/specs/core-specification-5-1/

[10] Krzysztof Janowicz et al. 2019. SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators. *Journal of Web Semantics* 56 (2019). https://doi.org/10.1016/j.websem.2018.06.003

[11] Krzysztof Janowicz and Michael Compton. 2010. The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration Into the Semantic Sensor Network Ontology. In *Proc. of SSN 2010*. https://ceur-ws.org/Vol-668/paper12.pdf

[12] Kang Eun Jeon et al. 2018. BLE Beacons for Internet of Things Applications: Survey, Challenges, and Opportunities. *IEEE Internet of Things Journal* 5, 2 (2018). https://doi.org/10.1109/JIOT.2017.2788449

[13] Myungin Ji et al. 2015. Analysis of Positioning Accuracy Corresponding to the Number of BLE Beacons in Indoor Positioning System. In *Proc. of ICACT 2015*. https://doi.org/10.1109/ICACT.2015.7224764

[14] Zhu Jianyong et al. 2014. RSSI Based Bluetooth Low Energy Indoor Positioning. In *Proc. of IPIN 2014*. https://doi.org/10.1109/IPIN.2014.7275525

[15] Sanya Khruahong et al. 2017. Multi-Level Indoor Navigation Ontology for High Assurance Location-based Services. In *Proc. of HASE 2017*. https://doi.org/10.1109/HASE.2017.9

[16] Tim Kindberg and John Barton. 2001. A Web-based Nomadic Computing System. *Computer Networks* 35, 4 (2001). https://doi.org/10.1016/S1389-1286(00)00181-X

[17] Markus Koühne and Jürgen Sieck. 2014. Location-based Services With iBeacon Technology. In *Proc. of AIMS 2014*. https://doi.org/10.1109/AIMS.2014.58

[18] Kangjae Lee, Jiyeong Lee, and Mei-Po Kwan. 2017. Location-based Service Using Ontology-based Semantic Queries: A Study With a Focus on Indoor Activities in a University Context. *Computers, Environment and Urban Systems* 62 (2017). https://doi.org/10.1016/j.compenvurbsys.2016.10.009

[19] Hui Liu et al. 2007. Survey of Wireless Indoor Positioning Techniques and Systems. *Transactions on SMC: Applications and Reviews* 37, 6 (2007). https://doi.org/10.1109/TSMCC.2007.905750

[20] Google LLC. 2015. UriBeacon: The Web Uri Open Beacon Specification for the Internet of Things. https://github.com/google/uribeacon/tree/uribeacon-final.

[21] Sujith Samuel Mathew et al. 2011. Web of Things: Description, Discovery and Integration. In *Proc. of IoT 2011 and CPSCom 2011*. https://doi.org/10.1109/iThings/CPSCom.2011.165

[22] Dmitry Namiot and Manfred Sneps-Sneppe. 2015. The Physical Web in Smart Cities. In *Proc. of RTUWO 2015*. https://doi.org/10.1109/RTUWO.2015.7365717

[23] Jeff Z Pan. 2009. Resource Description Framework. In *Handbook on Ontologies*. Springer.

[24] Vassilis Papataxiarhis et al. 2008. MNISIKLIS: Indoor Location Based Services for All. *Location Based Services and TelecCrtography II* (2008). https://doi.org/10.1007/978-3-540-87393-8_16

[25] Rodrigo VM Pereira et al. 2017. A Digital Implementation of Eddystone Standard Using IBM 180nm Cell Library. In *Proc. of SBESC 2017*. https://doi.org/10.1109/SBESC.2017.28

[26] Salil Pradhan. 2000. Semantic Location. *Personal Technologies* 4, 4 (2000). https://doi.org/10.1007/BF02391560

[27] Dave Raggett. 2015. The Web of Things: Challenges and Opportunities. *Computer* 48, 5 (2015). https://doi.org/10.1109/MC.2015.149

[28] HS Rajashree et al. 2018. Indoor Localization using BLE Technology. *IJERT* 6, 13 (2018).

[29] Andrei Vlad Sambra et al. 2016. *Solid: A Platform for Decentralized Social Applications Based on Linked Data*. Technical Report. MIT CSAIL & QCRI.

[30] Daeil Seo and Byounghyun Yoo. 2020. Interoperable Information Model for Geovisualization and Interaction in XR Environments. *IJGIS* 34, 7 (2020). https://doi.org/10.1080/13658816.2019.1706739

[31] Hamza Soganci, Sinan Gezici, and H Vincent Poor. 2011. Accurate Positioning in Ultra-Wideband Systems. *IEEE Wireless Communications* 18, 2 (2011). https://doi.org/10.1109/MWC.2011.5751292

[32] Maxim Van de Wynckel and Beat Signer. 2021. *OpenHPS: Single Floor Fingerprinting and Trajectory Dataset*. https://doi.org/10.5281/zenodo.4744380

[33] Maxim Van de Wynckel and Beat Signer. 2022. POSO: A Generic Positioning System Ontology. In *Proc. of ISWC 2022*. https://doi.org/10.1007/978-3-031-19433-7_14

[34] Web Bluetooth Community Group. 2023. *Web Bluetooth Scanning*. W3C Working Draft. https://webbluetoothcg.github.io/web-bluetooth/scanning.html

[35] Deze Zeng, Song Guo, and Zixue Cheng. 2011. The Web of Things: A Survey. *JOC* 6, 6 (2011).