Indoor Positioning and Hyperlocation using Bluetooth Low Energy

# Overview

## Bluetooth Low Energy

Bluetooth low energy (Bluetooth LE, BLE, marketed as Bluetooth Smart) is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. Compared to Classic Bluetooth, Bluetooth Smart is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range.

Bluetooth Smart was originally introduced under the name Wibree by Nokia in 2006. It was merged into the main Bluetooth standard in 2010 with the adoption of the Bluetooth Core Specification Version 4.0.

## Bluetooth low energy beacons

Bluetooth beacons are devices that broadcast signals that can be heard by smart devices nearby. Paired with an app, Apple Wallet Pass, or Physical Web browsers, businesses are able to deliver contextually relevant content and information to users at very specific locations. Creating unique customer experiences through location aware applications has never been easier.

## Trilateration

Trilateration is the process of determining absolute or relative locations of points by measurement of distances, using the geometry of circles, spheres or triangles.

In addition to its interest as a geometric problem, trilateration does have practical applications in surveying and navigation, including global positioning systems (GPS). In contrast to triangulation, it does not involve the measurement of angles.

In two-dimensional geometry, it is known that if a point lies on two circles, then the circle centers and the two radii provide sufficient information to narrow the possible locations down to two. Additional information may narrow the possibilities down to one unique location.

## Triangulation

Triangulation is the process of determining the location of a point by forming triangles to it from known points.

Specifically in surveying, triangulation per se involves only angle measurements, rather than measuring distances to the point directly as in trilateration; the use of both angles and distance measurements is referred to as triangulateration.

Optical 3D measuring systems use this principle as well in order to determine the spatial dimensions and the geometry of an item. Basically, the configuration consists of two sensors observing the item. One of the sensors is typically a digital camera device, and the other one can also be a camera or a light projector. The projection centers of the sensors and the considered point on the object’s surface define a (spatial) triangle. Within this triangle, the distance between the sensors is the base b and must be known. By determining the angles between the projection rays of the sensors and the basis, the intersection point, and thus the 3D coordinate, is calculated from the triangular relations.

## Fingerprinting

Fingerprinting is a localization method comprising two phases. In the first phase—learning—vectors composed of the RSSI values and optional extra features measured by a measuring device in the known locations are collected . These reference values—the calibrated dataset—are saved together with the location coordinates into the fingerprint database for the needs of localization. In the second phase—localization itself—the device to be localized measures the RSSI values and compares them with the data in the fingerprint database using a suitable method. The most widely used algorithms or methods of comparison and estimation of the position are

(i) probabilistic methods,

(ii) -Nearest Neighbors,

(iii) neural networks,

(iv) support vector machine,

(v) smallest M-vertex polygon.

Concrete solutions based on collection of fingerprints are described by Bahl and Padmanabhan or Azizyan et al. who collect other features during the measurement, such as sound intensity, acceleration, light intensity, or color of the light. Wu et al. bring an interesting approach which assumes similarity between the so-called virtual and physical model of the interior. It automates the initial phase of learning based on clustering of the fingerprints. Then, the virtual rooms are mapped to the physical rooms.

Localization accuracy can be increased if the movement of localized objects is considered. Such methods utilise the history of previous measurements and estimate the position based on the known previous trajectory of the object. Other solutions use dead reckoning method based on collection of data from movement and orientation sensors of a mobile device (like accelerometer, gyroscope, and magnetometer). This way, the direction of movement and the distance traveled could be determined and combined with other measurements and/or estimations. Particle filters are often incorporated in the process of gathering such estimations.

## Bluetooth low energy based indoor location

Bluetooth-based indoor localization is not a novel idea. Due to the limitation of the original Bluetooth specification (now called Bluetooth Classic), this approach has not been widely used. The time required for obtaining a sufficient number of nearby Bluetooth devices was not satisfactory due to the lengthy process of discovery. Likewise, energy and economic demands of Bluetooth infrastructure were high compared to WiFi-based infrastructure, which also served other purposes.

The situation changed with the advent of Bluetooth 4.0 (including BLE/Bluetooth Smart) in 2010. Due to low energy consumption and configuration options (regarding the advertising interval and the transmitter output power), the utilisation of this technology is much more promising, not only in comparison with previous versions of Bluetooth, but also in comparison with today’s widespread WiFi-positioning. In, the authors focus on proximity estimation based on signal strength. Furthermore, directly compares the BLE-based localization to the WiFi-localization by deploying BLE beacons at the same spots where WiFi access points were originally placed. The results show that BLE is more accurate at identical places than WiFi.

## iBeacon Technology

iBeacon is Apple’s brand name of the technology based on the microlocalization and the interaction of a mobile device in the physical world. This technology can be considered to be the next development stage of the QR code technology or, alternatively, the NFC technology. iBeacon uses the Bluetooth Low Energy standard which is a part of a new version of Bluetooth 4.0. Sometimes, the terms Bluetooth Smart, Bluetooth LE, BTLE, and just BLE are used. It is a technology developed by Nokia (originally, the technology was named Wibree; in 2010 BLE was standardized) and in contrast to the previous versions of Bluetooth, dramatically lower consumption is typical for BLE.Also the way how the (peripheral) device announces its existence to the other devices is the opposite from how it is in the original Bluetooth Classic. BLE enables a peripheral device to transmit an advertisement packet without being paged by the master (central) device. Thanks to this communication model, it is possible to construct energy-efficient transmitters—BLE beacons or iBeacons according to Apple.

iBeacon is a small device which transmits particular information in a defined radius and in regular intervals. As soon as a mobile device (a smartphone) gets within this radius, it can receive such information and, based on this, it can perform an action. Considering low consumption of BLE, such a device can be powered by a coin battery for up to two years. Of course, the battery life depends on the transmitter output power (TX power) and advertising interval settings.

The iBeacon technology is going to be adopted by shop marketers. A visitor with a BLE-enabled smartphone may be notified of special offers, discounts, information, and so forth based on his/her position or proximity to a beacon. It finds similar use in museums and exhibition halls.

# requirements for this project

## Ios Device

Since the hardware part is as important as the software one, our Indoor Location is currently tuned around iPhones and iPads—these models will provide you with best location accuracy. Other Bluetooth Smart enabled iOS devices will work, but the positioning accuracy may vary. The issue is that since all the ios devices have the same antenna for Bluetooth, it is easy to do fingerprinting and trilateration because of it being standard over a group of devices.

## Beacons

We have used beacons made by Estimote in our project. Estimote beacons can be attached to any location or object. They broadcast BLE radio signals which can be received and interpreted by a smartphone, unlocking microlocation and contextual awareness. To be able to listen to these beacons, it is necessary to have a device that supports Bluetooth 4.0 or higher. The Estimote beacon contains an nRF51822 chip, a powerful, highly flexible multiprotocol System-on-a-Chip (SoC). The nRF51822 is built around a 32-bit ARM® Cortex™ M0 CPU with 256 kB/128 kB flash + 32 kB/16 kB RAMThe whole SoC is highly optimized to be energy-efficient. Thus the stable TX power of the beacon is ensured while the battery voltage may drop. When the voltage finally drops from 3 V to 1.7 V, a brown-out reset is generated and the device stops broadcasting. In its basic mode, a beacon simply transmits Bluetooth packets with identification data—so-called advertisements—in regular intervals. It does not communicate with the surrounding devices by any other sophisticated way. Advertisements contain the following data:

(i) MAC address.

(ii) Universally unique identifier (UUID)—common for a single deployment at a venue.

(iii) Major number—designated for dividing the beacon sets into smaller segments.

(iv) Minor number—designated for dividing the segments into smaller subsegments.

In the configuration mode, beacon’s broadcasting parameters (which include the above stated data transmitted in a packet and other parameters such as the TX power or the advertising interval) can be configured. In the configuration mode, beacons use advanced bidirectional communication with a master device (e.g., a smartphone) with the aid of which they are configured.

At a physical layer, BLE transmits in the 2.4 GHz industrial, scientific, and medical (ISM) band with 40 channels each 2.0 MHz wide. 37 channels are used to exchange the data among paired devices and 3 channels are designated for broadcasting advertisements. These 3 channels are thus primarily used by beacons and are chosen deliberately so that they collide with the WiFi channels as little as possible. The beacon broadcasts its advertisement packet repetitively based on the selected advertising interval while hopping over the 3 designated channels.

## Laptop with the server running on it

# Programming Basics

As mentioned earlier, we use only ios devices due to them having the same antenna for BLE signal broadcasting and receiving making it easier and more uniform for us to correlate the RSSI and distances from the beacons of the device. Hence the programming has been done in Xcode which is the coding IDE for ios. It is nowhere similar to Android Studio and requires some time to figure out due to Apple’s rigid developer policy. First you have to create a developer’s account in developer.apple.com. Create a free profile which will last for around a month. Be careful when you are programming as your provisioning profile or developer’s profile might expire which will force you to make a new profile midway.

Coming to Xcode, all you need to do is to go to the app store and download xcode using an Apple ID. If there are any issues, try making an Apple ID based out of US. For the address you can use websites which give you a fake address which can be used for creation of id.

# Steps

## Mapping out the location

Firstly, you need to map out the location which you want to place your beacons at. For this you can use the Estimote SDK to help you out. For this you can use the EILLocationBuilder Class.

In order to construct a new location you need to:

* set the shape of the location and its orientation
* add details such as beacons, walls, doors on boundary segments

The shape of the location is defined by its boundary points. For example, consider a square defined by points (0,0), (0,5), (5,5), (5,0) along with its orientation with respect to magnetic north.

EILLocationBuilder \*locationBuilder = [EILLocationBuilder new];

[locationBuilder setLocationBoundaryPoints:@[

[EILPoint pointWithX:0 y:0],

[EILPoint pointWithX:0 y:5],

[EILPoint pointWithX:5 y:5],

[EILPoint pointWithX:5 y:0]]];

[locationBuilder setLocationOrientation:0];

Points that define the shape of location also define its boundary segments. They are indexed in the same order as the points. Here there would be the following 4 segments: [(0,0), (0,5)], [(0,5), (5,5)], [(5,5), (5,0)], [(5,0), (0,0)].

The next step is to place beacons and doors in the location:

[locationBuilder addBeaconIdentifiedByIdentifier:@"aabbccddeeff"

atBoundarySegmentIndex:0

inDistance:2

fromSide:EILLocationBuilderLeftSide];

Using ESTIndoorLocationBuilder you can also place beacons inside the location (addBeaconWithIdentifier:withPosition:, addBeaconWithIdentifier:withPosition:andColor:).

I did it with more precise measurements of the room which I got from the Blueprint of the Cisco ThingQbator given to me through email.

## Updating position inside location

Once you have instance of EILLocation you can start monitoring and obtaining position updates for that location.

Monitoring location is simply determining if the user is currently inside or outside the location. In order to monitor location, first you need to create instance of EILIndoorLocationManager and start monitoring for that location.

EILIndoorLocationManager \*indoorLocationManager = [EILIndoorLocationManager new];

[indoorLocationManager startMonitoringForLocation:yourLocation];

To obtain position updates, you need to set a delegate which will be receiving the updates.

indoorLocationManager.delegate = yourDelegate;

[indoorLocationManager startPositionUpdatesForLocation:yourLocation];

In addition to position, updates provide also information about accuracy of determined position which can be visualized via EILPositionView as avatar with a circle of given radius within which the real position is expected to be.

[indoorLocationManager startPositionUpdates];

Note, that you need only one Indoor Location Manager to monitor multiple locations. However, position updates are available for only one location at the same time. If you need to enable position updates for new location, first you need to stop position updates:

[indoorLocationManager stopPositionUpdates];

In order to have Indoor Location status change and position updates without delay, you should start Indoor Location Manager and monitioring for Indoor Location early.

EILIndoorLocationManager position updates can be delivered in two different modes. Modes differ in accuracy, stability and responsiveness. Depending on the mode system resource usage may be different.

* EILIndoorLocationManagerModeNormal - Delivers most accurate and responsive position updates at the cost of high system resource usage. To achieve best results user should hold phone in hand in portrait orientation. This is default mode of EILIndoorLocationManager.
* EILIndoorLocationManagerModeLight - Delivers stable, but a bit less responsive position updates. Has a very low system resource usage.

In order to change mode simply change mode property of EILIndoorLocationManager object.

indoorLocationManager.mode = EILIndoorLocationManagerModeLight;

If the position updates delivery is in progress it will effectively restart position updates with new mode.

## Managing Locations in the Estimote Cloud

For this, you would need an estimote account with the beacons that you used for mapping attached to that account. First, go to cloud.estimote.com and log in with your account. After that go to apps and create a new web app which is user defined(don’t select the predetermined options in those templates). Now you will have to name that app and then the estimote cloud will return and app token and id which needs to be put in the code of your app, like this.

        ESTConfig.setupAppID("indoormap-erw", andAppToken: "7ad02b90ea2517123df48f6b730a3fae")

Now you must set up a delegate of the EILIndoorLocationManager object which would help you fetch the location from the cloud so that it can be used in the app. The location identifier can be obtained by going to the locations tab in the cloud.estimote.com and then clicking on settings of the corresponding location. Here the identifier field would have the location identifier which we need to use. This is done as follows:

self.locationManager.delegate = self

        let fetchLocationRequest = EILRequestFetchLocation(locationIdentifier: "shachell-s-location")

After this you can go ahead and draw the location in the app done as follows,

        fetchLocationRequest.sendRequestWithCompletion { (location, error) in

            if let location = location {

                self.location = location

                self.locationView.showTrace = true

                self.locationView.rotateOnPositionUpdate = false

                self.locationView.drawLocation(location)

                self.locationManager.startPositionUpdatesForLocation(self.location)

} else {

                print("can't fetch location: \(error)")

            }

## Exporting the position to a server

Here an Asynchronous GET request is made by making a NSMutable object and requesting a particular URL. Your current x and y positions are passed as a parameter in the request.This is done as follows:

let request = NSMutableURLRequest(URL: NSURL(string: "<http://10.196.65.165:8088/app>?"+position.x.description+"&"+position.y.description+"\*0.76094754727664&0.030401932938554")!)

        let task = NSURLSession.sharedSession().dataTaskWithRequest(request)

        task.resume()

Note that here, the static positions passed along with position.x and position.y represent the objects given to the app by WiFi based positioning carried out by CMX server. Since at this point of time, Prime Infrastructure is down, I have not been able to integrate CMX with the BLE based locationing. Hence I am passing static parameters from the CMX point of view. Once CMX is up and running one can request for the Json object from CMX and pass the positions dynamically.

## Energy Conservation

Initially the beacons are configured to have optimum advertising interval and transmission power(2000ms and -20db resp). When the app is launched, the EILIndoorLocationManager object automatically changes the advertising interval and the transmission power of the beacons ranged(it becomes 500ms and -12db). So we don’t have to worry about changing the beacons characteristics as soon as the app opens.

However, we need to worry about when the app is going to close. We have to go back to the original advertising intervals and transmission power so that we can continue to get about 18 months of battery life in the beacon. For this we need to use the writeAdvInterval function attached to the Beacon object. This property must be called in the function applicationDidEnterBackground present in the ViewController.swift file of the xcode project.

## Server to handle CMX Position and BLE Position

This server can be done in Java and is not something you should worry too much on. All it should do is take 2 positions as parameters display both and display the median of the 2 positions calling it the ideal position. No changes need to be done in the server that I have already made and is not an important part to worry about.

# Final Aim of The Project

# Remaining Work

Integration of CMX with the app is the only thing which is remaining. After this, we could probably bring in Visual Light Communication which could augment WiFi based positioning and Bluetooth based positioning to give more accuracy. Energy harvesting using the lux intensity of light can also be done to improve the BLE based positioning.