ITU ContextPhone Platform

Maytham Fahmi
IT-University of Copenhagen
masf@itu.dk

Dennis Sadeler Shapira
IT-University of Copenhagen
dssh@itu.dk

Maria Natividad Lara Diaz
IT-University of Copenhagen
mnad@itu.dk

ABSTRACT

In this paper we describe the concept, development and reflection for the first mandatory assignment of Pervasive Computing course (Spring 2015) at IT University of Copenhagen where we built a context-aware mobile phone utilizing an iBeacon infrastructure.

ACM CLASSIFICATION KEYWORDS

Ubiquitous and mobile computing, Mobile phones, Location based services, Global positioning systems, Sensor devices, Google Maps.

Author Keywords

Context-aware system; context;

INTRODUCTION

With the increase of mobile devices, the world around us offers a lot of opportunities to improve human-computer interactions through systems that adapt better to our every day routine.

The information in the physical environment creates a context for the human-computer interaction. The context of an entity (person, place or thing whether electronic or otherwise) is an aspect of its physical circumstances that is relevant to the systems behavior.

From that context it is possible to gather relatively simple values from physical sensors such as; measuring the ambient light on a given time of day, detecting a device when entering the proximity of a sensor(iBeacon) and reading the location of a device with the help a location provider. In this project the context awareness that is relevant for us is mainly gathering the location coordinates of an entity.

The objective for this assignment is to built an application for a mobile phone that is context aware by utilizing an iBeacon infrastructure.

CONCEPTUAL DESCRIPTION

A user arrives at ITU, outside the building the coordinates are registered on the android phone using the location provider through GPS. After he enters the building, the GPS is going to lose the ability to read the coordinates due to being inoperable when located indoors. The coordinates that were read were the last known coordinates of the users location. When the user reaches the iBeacon signal the application will eighter use the GPS or the network provider to check if the location is already stored in the cloud-infrastructure, if it has not been registered yet, the application stores the location of the iBeacon based on the last known coordinates. In that way, when he or any other user reaches the same iBeacon signal, instead of storing the location, the application will retrieve the coordinates from the cloud database and show its location on the map. [Figure 1]

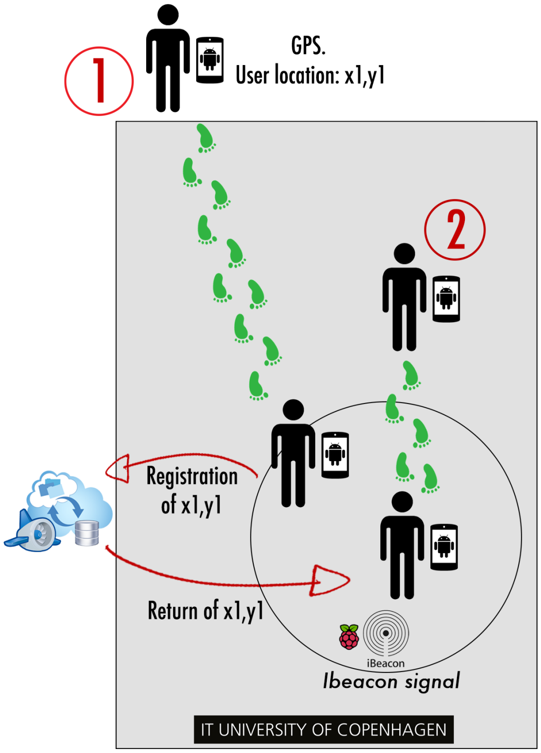

Figure 1. Interaction model

PROJECT SCENARIO

To document the architecture of the system, we divide it into a number of tasks:

- The creation of an iBeacon hardware node.
- The creation of an Android Map application.
- The creation of a cloud-based infrastructure.
- The creation of a ContextMonitor Android Service.

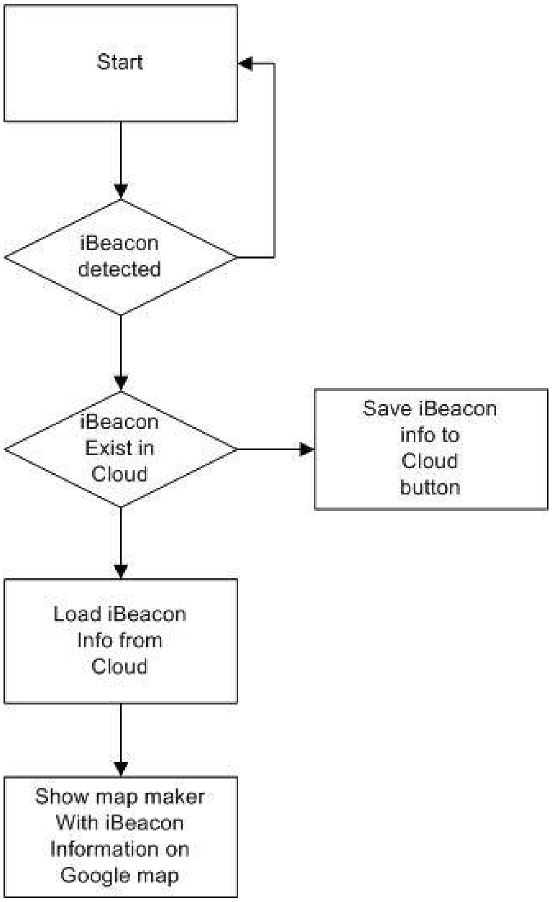
Start
iBeacon detected
iBeacon Exist in Cloud
Save iBeacon info to Cloud button
Load iBeacon Info from Cloud
Show map maker With iBeacon Information on Google map

Figure 2. Process of the system

IBEACON HARDWARE NODE

An iBeacon is a small Blutooth Low Energy (BLE) device that provides specific information to the user, such as position and a unique identifier (UUID). It is possible to use multiple iBeacons in order to build an indoor network location system that tracks a users location as he moves around inside a building without relying on a mobiles GPS. One of the major advantages of this device is that the interaction doesn't require any action from the user. Hence, it functions with automated operations running in the background that meets the context-awareness concept we talked of earlier(On our application we inserted button to manually that checks if the iBeacon was previously registered as well as a button to update the cloud. This was only done for demonstration purpose).

Besides being part of the standardization of Bluetooth 4.0, the iBeacon incorporates a communication protocol to validate itself, which is done by transmitting a unique pulse of 128-bit (UUID) to identify an entity and two values of 16-bit values better known as ”Major” and ”Minor” to identify each element in the same entity.

The Android device receives a iBeacon signal and for the this project we utilized a Raspberry Pi to emit iBeacon advertisements.

The configuration of a Raspberry Pi worked as an iBeacon. The Raspberry Pi can yield more useful advantages than the configuration itself. In creative scenarios it can be enhanced to be suitable for multiple applications. To mention a variety of ideas ranging from amusing to practical, these could be; organizing a scavenger hunt for digital treasures placing iBeacons with different tasks, marketing usability by customizing offers based on their location in a store or even simple and useful uses like alerting if someone is trying to steal your bike parked outside your house.

As a requirement we were handed a RaspberryPi that had to be configured to work as an ibeacon.
In order to set up the RaspberryPi and enabling it to emit signals as an iBeacon to our Android application, we installed some open source tools, mainly Bluez, that is the main Bluetooth stack for Linux, libusb and some helper libraries.

The raspberry by itself doesn't proportionate any kind of bluetooth communication which subsequently lead us to use a USB Bluetooth adapter version 4.0. After installing the required libraries, we were able to enable and disable the device scanning and configure the UUID identification for the devise.

However, in order to make it more simple to enable and disable the devise, we created two scripts in the RaspberryPi with functions in charge of the start/stop of the signal device.

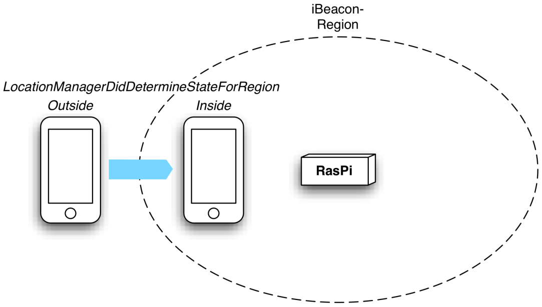

Figure 3. Bluetooth Presence Awereness

ANDROID MAP APPLICATION

The Android platform has a practical and convenient positioning system that combines two technologies:

- Global location system based on GPS - This system, as we commented on before, needs to be used outdoor in order to have direct visibility to the satellites.
- Location system - This is the solution for facilitating the access in indoor locations that is based on the information received from the cellphone towers and from Wi-Fi locations.

These two services are totally integrated in the android system.

CLOUD-BASED INFRASTRUCTURE

In order to have a better understanding of the creation of the Cloud-based infrastructure we need to separate it by explaining three main components. 1. The implementation of the cloud which is done by utilizing Google App Engine. 2. The

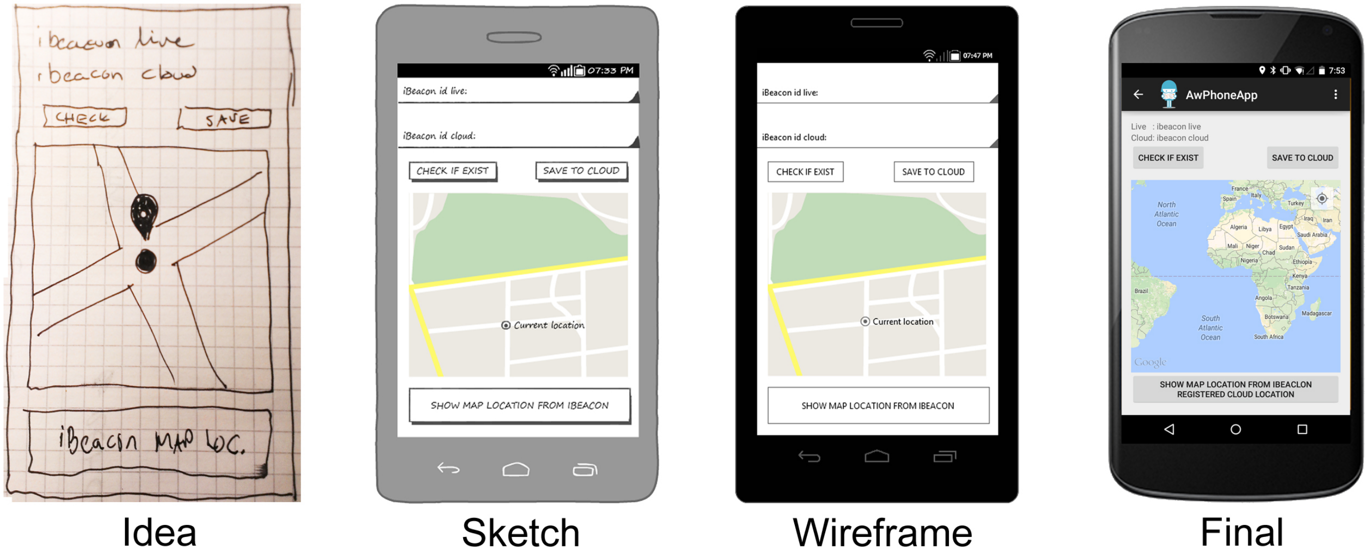

Figure 4. Interaction model

communication between client and the server is done by implementing REST Services. 3. The format that is communicated between them which is done by using JSON.

The Google App Engine that represents the cloud is internet-based and manages the services and the CRUD for storing data. This technology offers an efficient way to handle resources such as storage, memory, processing and bandwidth, to provide only the necessary resources for each request. [Figure 5]

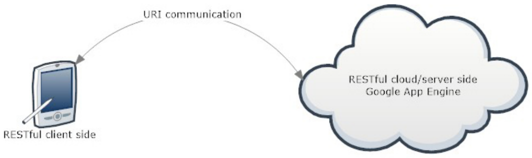

Figure 5. Cloud structure

Furthermore, being a free service that Google offers since 2008 for the production and hosting of web applications made for Python, Java, Go and PHP it seemed as convenient choice.

As mentioned above the communication between the cloud and the client is done using RESTful Services. These services differ from SOAP Services in the way that SOAP is oriented towards Remote Procedure Call(RPC) with a rigid interaction between client/server that is tightly coupled, meaning, invoking methods through a remote service with constrained and prior knowledge of an API. REST on the other hand is oriented towards resources that are available by identifiers(URIs). The client conforms to a protocol of standardized methods that can't be violated, leading to nicely decoupled system. Through the Rest API the client can generate the actions needed.

The web services following REST Services must validate some premises:

- Client/server - The interface of communication between them separate completely the responsibilities of each one.
- Cache - The content of the services can be cached so that once the first request to service has been made the following requests can rely on the cache if necessary.
- Uniform Services - They share the same invocation by using uniform methods GET, POST, PUT and DELETE.
- Architecture by layers: The services are oriented to be scalable and the client wont be able to recognize if the user is making a direct request to the server or if there's a cache system in between.

JSON (JavaScript Object Notation) is a lightweight data exchange format. It is based on a subset of the JavaScript Programming Language but is a text format completely independent from the language that it is used by. This property makes JSON an ideal language for data exchange.

CONTEXTMONITOR ANDROID SERVICE

By building an Android service that runs in the background we are able detect context information from various sensors. The service runs in an infinite loop that "scans" the context monitor which is a set of classes that encapsulate the context sensing. The classes implement sensors for sensing its surrounding environment. For each scan the service will consecutively detect whether new monitors have been added or removed from the context monitor as well as retrieving the the information provided from the sensors. In this project the context monitor utilizes the following three sensing technologies:

- A light sensor - The phones build-in light sensor measures the ambient light.
- A gravity sensor - The phones build-in gravity sensor that measures the gravity.

- A iBeacon sensor - Utilizing the phone blue tooth we are able to detect iBeacons signals when entering within their detection proximity.

On retrieval of context information the ContextEntity (which is a class that defines our data field) is used for storing the data in a database using google datastore.

REFLECTION

An important reflection regarding this application was considered during development, namely the accuracy of the location when entering the ibeacons proximity field. Since the iBeacon was located indoor(inside the building of ITU) the last known coordinates may have been fetch outside of the building(using GPS) which would result in a poor precision of the location of the iBeacon. Hence depending on the size of the building the precision could vary accordingly. Apart from the precision, various attempts could have been done in order to optimize the accuracy. The following thoughts/approaches could have been further developed in order to optimize the coordinates. In stead of just storing the coordinates the first time an iBeacon has been registered and using those coordinates for future retrievals it would have been more sufficient to design an algorithm that would use previous fetched coordinates with newly fetched coordinates an then calculate the inaccuracy either by calculating an average distance and then subsequently updating the coordinates accordingly. To be even more meticulous, an array of coordinates of a given iBeacon could have be stored and then from these coordinates a location could be calculated. This would consequently mean that the more a given ibeacon have been registered the more accurate its location would eventually be.

CONCLUSION

The requirements of this project was to develop an android application for a mobile phone that utilizes the above mentioned technologies to implement an application with context awareness. Based on the requirements of only having to register a single iBeacon the project may still seem some what superficial. But though acknowledging that the project is still in its infancy we can conclude that this has been successful as far as being in the first faze. Projecting into future, the next step would be using multiple iBeacons, proper administration of the location provider and registering more coordinates in order to acquire more precision and accuracy. By venturing into these aspect we would be capable in creating a mature and practical application.

REFERENCES

1. Paul Martin, Bo-Jhang Ho, Nicholas Grupen, Samuel Muoz, Mani Srivastava *An iBeacon primer for indoor location: demo abstract.* http://dl.acm.org/citation.cfm?id=2675028
2. Sara Perez Barragan *iBeacon Technology in Welfare : A study of Bluetooth Low Energy for Indoor Positioning.* http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A751730&dswid=-6907
3. Mattew S.Gast *Building Applications with iBeacon* http://goo.gl/1uJ209
4. Gregory D. Abowd, Chistopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, Mike Pinkerto. *Cyberguide: a mobile context-aware tour guide.* http://dl.acm.org/citation.cfm?id=272199
5. Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, Paul Webster *The anatomy of a context-aware application.* http://dl.acm.org/citation.cfm?id=506907
6. B. Schilit, N. Adams, R. Want *Context-Aware Computing Applications.* http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4624429&tag=1
7. Amit Kushwaha, Vineet Kushwaha *Location Based Services using Android Mobile Operating System.* http://www.ijaet.org/media/0001/Location-Based-Services-using-Android\-mobile-Operating-System-Copyright-IJAET.pdf
8. S. Kumar, Dept. of Comput. Eng., Aligarh Muslim Univ., Aligarh, India. *Location based services using android (LBSOID).* http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5439442
9. Claudio Riva, Markku Laitkorpi *Designing Web-Based Mobile Services with REST.* http://link.springer.com/chapter/10.1007/978-3-540-93851-4_42#page-1
10. Cesare Pautasso, Olad Zimmermann, Frank Leymann *Restful web services vs. ”big”’ web services: making the right architectural decision.* http://dl.acm.org/citation.cfm?id=1367606
11. Wikipedia *Representational State Transfer.* http://es.wikipedia.org/wiki/Representational_State_Transfer
12. IT-University of Copenhagen *LAB Exercises.* https://learnit.itu.dk/course/view.php?id=3002998
13. D. Crockford *The application/json Media Type for JavaScript Object Notation (JSON).* https://tools.ietf.org/html/rfc4627