Contextual Computation and Context Awareness
Occupancy and Traffic Monitoring in the new Mary Idema Pew Library

By
Sam Serpoosh
Contextual Computation and Context Awareness
Occupancy and Traffic Monitoring in the new Mary Idema Library

By
Sam Serpoosh

A project submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Information Systems

at
Grand Valley State University
April 2014
Dedication

This master’s project is dedicated to my parents and sister. They always supported me throughout my career and education, respect my decisions and motivate me to improve myself and reach for a higher level.
Acknowledgements

My special thanks to Professor Jonathan Engelsma for all of his great commitment to this project and my career. It has been a tremendous pleasure to meet him and work under his supervision. Also the Mary Idema Pew Library, especially Carlos Rodriguez, Kyle Felker and Kristin Meyer for their help and willingness to cooperate to this project when it was needed.

To Ehsan Valizadeh for his dedication to collaborate in testing the project.

To Davoud Qorashi and all the other students who cooperated in this experimental project by using the application.

I hope the collected data, application and its server would help the students to find less crowded areas to study and work conveniently. I know that there were lots of improvements to be made but due to the time constraints application and sever features kept under a tight scheduled implementation.
# Table of Contents

Abstract .......................................................................................................................... 1

Introduction ..................................................................................................................... 2
  Looking for a book in bookshelves ................................................................. 3
  Campus Tour Guide ................................................................. 4
  Meeting with Library Staff ................................................................. 4

Architecture ..................................................................................................................... 5
  Client .......................................................................................................................... 5
    Region-Monitoring .................................................................................. 5
    Beacon Ranging .................................................................................. 6
  Estimote Beacon .................................................................................. 6
  Web Server ................................................................................................ 6
  Overall Workflow ............................................................................... 7

Implementation .............................................................................................................. 8
  Client Side ........................................................................................................ 8
    Region Monitoring and Beacon Ranging Frequency ................................ 10
    Application is running in foreground .................................................. 10
    Application is running in background ............................................. 10
  Server Side ................................................................................................... 11

Data Interpretation ........................................................................................................ 13
  How long users occupy a given area ..................................................... 14
  Taking a Break while Studying .............................................................. 14
  Walking by a monitored area ............................................................... 15
  Hourly Report of Areas Occupancy ........................................................ 15
    Closer look at hourly population in different areas ................................ 16
  Number of devices in areas ................................................................. 17

Lessons Learned .......................................................................................................... 17

Bibliography ................................................................................................................. 21
Abstract

Nowadays, handheld gadgets are equipped with sensors, which enable them to provide different kinds of information about the people who use them. Various algorithms and techniques use this information in order to provide users with useful and more relevant data and content. Some of these techniques fail to take into account an important element when it comes to human-computer interaction, and that is the current context of the user. Humans are very effective at using implicit situational information or the context to improve the interaction and communication. Applications can utilize contextual information to improve human-computer interaction as well.

For the purpose of this study, any information that can be used in order to describe the situation of an entity relevant to the interaction between user and application is considered context. [1] An application that takes into account this contextual information is referred to as a context-aware application. This study involves a context-aware application that monitors the occupancy of particular places on campus. In particular, we set out to automate monitoring occupancy, a task previously completed manually, in GVSU’s new Mary Idema Pew Library. We installed beacon devices transmitting Bluetooth Low Energy (BLE) signals in two test areas in the library and an additional beacon in our laboratory in Mackinac Hall. We incorporated additional software in an iOS application (Laker Mobile), which is widely used on campus. This new software runs in the background and senses collocated beacons and reports a privacy preserving universal identifier representing that particular iOS device to a network based server that was also implemented as part of this project. Our implementation uses a new technology called iBeacons, introduced by Apple in iOS 7. iBeacons utilize the BLE features that have been included in recent iOS devices, to create context-aware application experiences. Based on the collected data the server can provide real-time information about the population of the monitored areas and a heat map associated with them. Ultimately, this information can be used to help people find less crowded areas to study in the library.
Introduction

A large amount of data is collected by software applications these days. Applications and websites use this information and apply different algorithms on them in order to provide users with more relevant content and useful information. This information is dynamic and specific to each individual user based on the information available about them, instead of the traditionally fixed and static content, which was the same for all the users of the application. Presenting a relevant and more appropriate content for each individual is a key to a better user experience and providing value to users. In fact, it is so important that companies like Netflix setup a competition for programmers to come up with a Recommendation Engine, which gives relevant movie suggestions to users. [2] Some of these techniques and algorithms miss an important element in this equation and that is the current context of the user.

Current context is the important missing element when it comes to human-computer interaction. Contextual and situational information can make a big difference and help the application to provide even more relevant content for each specific user. Context and situation that we are talking about here can be different based on the functionality the application is providing and the problem it is trying to solve.

Imagine a smart phone application, which is designed for The Museum of Arts and Design in New York City. Its main functionality is to provide explanation about museum and different items it has on display. It would be unacceptable to receive information about a painting of 18th century while looking at a maquette of Statue of Liberty! The way that most applications in this category avoid this problem is by asking the user to act manually and select the section related to what they are looking at while they are looking at it when they want more information about it. This way the user is manually providing the context to the application in order for the application to act appropriately and shows relevant content. In the world of human-computer interaction, the less the user needs to do the better. So a better approach would be to solve this problem automatically. The context of the user in this situation is what they are looking at or in other words which item they are close to. If the application can get that information automatically then it can act upon it appropriately without any need for the user to provide the context. [3]

For another example, imagine an application equipped with a recommendation system, which suggests group activities to users. Most of the applications in this category consider the location of the user and what sorts of activities are interesting from the user’s perspective. But they miss a very important element here, and that is who are the people the user is hanging out with at the moment. Different groups are interested in different activities. The context of the user in this situation is where is the user, what he
likes and whom he/she is hanging out with. The graph of this social group can be detected in real-time and based on previous activities of this group together and where they are at the moment, appropriate suggestions can be made by the recommendation engine.

As we mentioned before, the current context can be different based on the problem at hand. The context can be related to where, when, what, who, whom, etc. or a combination of those and how it can affect the human-computer interaction.

For the purpose of this study we did several brainstorming sessions in order to come up with an idea that can help solving an existing problem on the university campus, which is related to contextual computing and context-awareness and those are the keys to solve it. The following shows few of the options that came to mind.

**Looking for a book in bookshelves**

Looking for a book in bookshelves is a frustrating task for most people. Even if the person knows what bookshelf the book is on, there are still many sections on a single bookshelf to search through. The user ends up walking along the shelf, or perhaps standing on a stool in order to see the upper portion of the shelf. This can be a time consuming task.

![Figure 1. Looking for a book in shelves](image)

In order to conveniently solve this problem, we can install beacon devices in library shelves and detect where is the user and between which two shelves he/she is standing through a mobile application, then present the list of books on that shelf and their specific location (e.g. third section on the left) and then show the list of the books only in those shelves and where they are. Now the user can scroll through the list instead of moving around and jumping on a stool in order to find the book and when they find it, they go to exactly where it is and pick it.
Campus Tour Guide

Often when you’re walking on campus you see a group of people visiting the university led by a tour guide who is providing information about different locations on campus. We discussed installing beacon devices near points of interest around campus and developing a \textit{Campus Tour} mobile application that would automatically provide information about different locations, buildings, statues, etc. on campus to visitors based on the user’s current context. This can be something similar to what we described in the scenario of the application for The Museum of Arts and Design in New York City.

Meeting with Library Staff

After a number of brainstorming sessions we met with the staff of the new \textit{Mary Idema Pew Library} on campus in order to find out more about the problems they have, that might be solved in an efficient way with a context-aware solution. The following were the two main outcomes of the meeting:

- Notifying students about the new books in each section while they are walking by different sections in third and fourth floors.
- Monitoring the occupancy of certain areas of the library and traffic activities in those areas. Particularly the study room in fourth floor and the innovation zone in third floor.

Eventually we came to the conclusion that monitoring the occupancy of those areas in the library is a better candidate since first of all, it is a more important problem that needs to be solved from the library staff perspective. Secondly, this task is done manually at the moment. Library staff goes to those areas every hour and records how many people are in each of the areas being monitored.

We decided to install beacon devices in those areas of the library and one additional device in our laboratory in Mackinac Hall for testing purposes and gathering data for analysis. Then we chose an already existing iPhone application (Laker Mobile), which is being used by the majority of students on campus in order to add a context-aware feature for solving this problem. The feature works in the background and is supposed to monitor the traffic activities (e.g. Enter, Exit) for those particular areas and submit the information to a web server implemented for this project.

Based on the collected information, the people at the library will be able to get real-time information about the population of those areas and also a heat map associated with them. Eventually this information can be used to help students to find less crowded areas to study.
Architecture

Overview of the architecture and infrastructure for our solution is shown in Figure 2.

![Figure 2. High-level overview of the client and server collaboration](image)

Client

The client side of this project is a new feature added to the already existing iPhone application called Laker Mobile. This feature is using the *iBeacons* technology that was introduced by Apple in iOS 7. *iBeacons* is part of the iOS *CoreLocation* library and utilize the Bluetooth Low Energy (BLE) features included in recent iOS devices. Two of the main functionalities provided by *iBeacons* are called *region monitoring* and *beacon ranging*. These are done by scanning BLE signals transmitted by beacon devices (*Estimote* beacons in the case of this project). The frequency of BLE scanning varies based on different statuses of the application. For instance when the application is running in background and the iPhone is locked, BLE scanning happens up to every 15 minutes or so for battery efficiency reasons. Or when home or shoulder button is pressed the phone starts scanning for few seconds and stops.

Region-Monitoring
iBeacons have the capability of monitoring the exit and entrance to an area covered by a beacon device. The region monitoring can be done whether the application is running in foreground or background. This activity is detected whenever a device crosses the border of a region covered by a beacon device.

**Beacon Ranging**

iBeacons can also range beacon devices in a given region. That means they can detect the beacons and the proximity of the client device (in this case an iPhone device) to them. There are four different proximity levels including Far, Near, Immediate and Unknown. The beacon ranging can only happen when the application is running in foreground.

**Estimote Beacon**

The iBeacons we use are called Estimote. They are small, wireless devices that can be placed in an area. iBeacons transmits tiny radio signals to smart devices. It is like a small lighthouse. Smart devices can hear these signals and communicate with the beacon and estimate their proximity from it precisely. [4] We installed one of these beacons in the study room on the fourth floor of the Mary Idema Pew Library, one in the third floor innovation zone and one in our laboratory in Mackinac Hall building. They all share the same identifier in the form of a UUID, which represents the manufacturer of these beacon devices (in this case Estimote). Then each beacon device has a unique combination of major and minor number. For instance in one department, the beacon devices can have the same major number, which represents they belong to that department and each one has a different minor number that determines where in the department that device is installed. [5]

![Figure 3. Estimote Beacon installed on a wall](image-url)

**Web Server**
The web server of this project is written in Ruby on Rails and deployed on a Heroku Dyno. There is an API endpoint for the client iPhone application to submit the information about region monitoring and beacon ranging, essentially the traffic data. This API endpoint only accepts data from an authenticated client application and is not an open endpoint. The server receives the data from clients and does some preprocessing before storing them. It can provide traffic information about the monitored areas during a requested period. It also provides real time information about the population of the monitored areas and a heat map associated with them. The main endpoints of the server are specified in the following table:

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>/api/device Traffics</td>
<td>Receives traffic data form the client application</td>
</tr>
<tr>
<td>/api/traffics?from &amp; to</td>
<td>Returns traffic data in a given period</td>
</tr>
<tr>
<td>/api/areas_population</td>
<td>Returns real-time information of monitored areas population</td>
</tr>
<tr>
<td>/areas/heatmap</td>
<td>Shows a heat map associated with current population of monitored areas</td>
</tr>
</tbody>
</table>

**Overall Workflow**

The overall workflow of the interaction between the client and the server is pretty straightforward. When a device enters a monitored region, this activity will be detected by the application running on the device and the information related to this traffic activity (e.g. time of the event, the name of the area, the activity that is Enter in this case, etc.) is submitted to the server. The server receives this information and does the required processing on the data and stores it. Similarly when the device exits the area, it will be detected by the application and relevant information is submitted to the server and so on. While the device is inside the monitored area, it will be included in that area’s population and the heat map associated with it. Of course in order for the device to detect the traffic activities and act upon them, it needs to have the Bluetooth on and running the application either in background or foreground. In order to inform the user that turning on the Bluetooth improves the location accuracy and user experience, a pop up dialog is prompted when they start the application and the device’s Bluetooth is off so they can directly go to settings and turn the Bluetooth on. Something like the following picture:
Implementation

In this section, we briefly talk about some of the interesting parts in the implementation of client and server for this project.

Client Side

In order to implement features related to region monitoring and beacon ranging provided by iBeacons in iOS 7, we needed to use CoreLocation library. In the case of this project we used Estimote SDK for iOS, which is a very thin wrapper around iOS CoreLocation library and adds a few extra helper methods related to their own beacon devices. But clearly you can implement the same functionality using only iOS CoreLocation. [6]

In order to monitor regions and range beacons we need a module called ESTBeaconManager. It has the responsibility of monitoring regions for enter and exit activities and also for ranging beacons in a given region. This beacon manager module works with three important callback functions associated with region monitoring and beacon ranging. The main functionality of the application, which is aware of the context
and act differently based on the current context of the user, would be inside those
callback functions or will be triggered inside them. The followings are the prototype for
those three callbacks in Objective C: [7]

- (void)beaconManager:(ESTBeaconManager*)didEnterRegion:(ESTBeaconRegion)
- (void)beaconManager:(ESTBeaconManager*)didExitRegion:(ESTBeaconRegion)
- (void)beaconManager:(ESTBeaconManager*)didRangeBeacons:(NSArray*)inRegion:(ESTBeaconRegion *)

While monitoring regions, the beacon manager detects enter or exit activities to a certain
region and invokes the appropriate callback function for that region. For instance,
imagine a device entered the innovation zone on the third floor then the didEnterRegion
callback would be invoked for that area. The code for this callback in the project looks
somewhat like the following:

- (void)beaconManager:(ESTBeaconManager*)beaconManager
didEnterRegion:(ESTBeaconRegion*)region {
    int major = [[region major] unsignedShortValue];
    int minor = [[region minor] unsignedShortValue];
    NSString* areaName = [self getAreaNameBasedOnMajor:major andMinor:minor];
    NSString* activity = @"Enter";
    NSString* proximity = [self calculateProximityIfAvailable];
    [self submitTrafficActivityInformationToServer];
}

The scenario can be simply laid out in the following pictures:

Figure 5. Workflow of the application in Enter to and Exit from a monitored region
**Region Monitoring and Beacon Ranging Frequency**

As mentioned before, the frequency of region monitoring and beacon ranging depends on few factors. First of all, region monitoring can happen whether the application is running in background or in foreground, but beacon ranging can only happen when the application is running in foreground.

When the device is not in a region, the beacon ranging is not happening at all in order not to waste the resources and being battery efficient. But when the device is inside a region, beacon ranging happens quite frequently and every few seconds the information about ranged beacons would be submitted to the server. Things like proximity of the device to the beacons etc.

Region monitoring has more scenarios.

**Application is running in foreground**

When the application is running in foreground and it is ranging, the region monitoring is happening constantly and when the device crosses a border of a region the appropriate callback will be invoked.

**Application is running in background**

When the application is running in background, the time it takes to detect a region change is specified in the following table.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Max Time to detect Range Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>notifyEntryStateOnDisplay = YES &amp; Phone awakened</td>
<td>1 second</td>
</tr>
<tr>
<td>notifyEntryStateOnDisplay = NO &amp; Phone awakened</td>
<td>NEVER</td>
</tr>
<tr>
<td>Application is just running in background</td>
<td>Up to 15 minutes</td>
</tr>
</tbody>
</table>

As you can see it depends on few different factors. [8]

This new feature in Laker Mobile application is mostly a background service and it has almost no User Interface associated with it. Although, there is the ability for user to turn on and off the occupancy-monitoring feature in the library section of the application. It looks like the following:
Server Side

On the server, the information about region change detection or ranged beacons is received, processed and stored on a PostgreSQL database. There are few things on the server that are worth mentioning.
A lot of information about region change detection and ranged beacons is submitted to the server. This information can be redundant. For instance when an iOS device is near a beacon device and the application is ranging beacons, it will keep sending this information to the server. For the purpose of this application the only important information is whether a device entered or exited a certain region. So on the server side, we check the received information activity and area with the latest activity and area for the device and if they are the same, the received information on the server would be ignored and not stored in the database. In other words the server only accepts and stores the received traffic information about a device if it is different from the latest traffic information of the same device for the same area.

Real-time population calculation of monitored areas has some subtleties:

- Device can be on the border of a region and because of that there is a possibility for the server to receive and store two subsequent traffic data at the exact same time, one for Exit and one for Enter. When the device enters a region, it will still submit Enter traffic data up to every 15 minutes or so but in the case of Exit, there is only one Exit data that will be submitted. For that reason, when it comes to calculation of the population for that area, the Exit activity has more priority between Exit and Enter traffic data with the same time. Because Enter data will be submitted again if the device will be in the region but Exit wont since the device already left the region.

- Imagine a scenario where a device enters the innovation zone on the third floor and all of a sudden the device dies or the user turns off the Bluetooth. From now on, no traffic data will be submitted to the server. Later the Bluetooth is on and the device enters the study room on the fourth floor. On the server, there are two Enter activities for two different areas for the same device. When it comes to calculation of the population for areas, the one that happened more recently will be considered more accurate and that device will be considered in the population of the study room on the fourth floor. Also in order to improve the accuracy of the traffic data for the device, an Exit activity will be inserted for the innovation zone on the third floor since the device clearly exited that area and it is in another area at the moment.

- Inaccurate traffic information obviously can affect the population calculation. As we mentioned before, there can be a scenario that a device enters a region and then dies or the user turns off its Bluetooth. In that case
the Exit traffic data for that device never come to the server. When that happens, one extra device will be counted any time user requests the population of that area. In this implementation, this inaccurate information only lasts until the end of the day. Because in order to calculate the population of areas, only the devices of the current day are considered.

- The heat map for the monitored areas is presented based on their capacity and population at any given time. Upon the request, users can see each monitored area in one of the four green, yellow, orange and red colors. Respectively, these colors determine that the population is in the first, second, third or last quarter of the area’s capacity. The following is a snapshot from the heat map:

![Heat Map](image)

**Data Interpretation**

There is different information that can be extracted out of the collected and processed traffic data on the server. We mention few of them here along with real data from the server.
How long users occupy a given area

From the traffic data, we can understand how long a person was occupying an area. For instance, consider the following traffic data:

```json
{
  "device_id": "device id 1",
  "area": "3rd Floor Innovation Zone",
  "activity": "Enter",
  "occurance_time": "2014-04-20 13:46:55"
}
{
  "device_id": "device id 1",
  "area": "3rd Floor Innovation Zone",
  "activity": "Enter",
  "occurance_time": "2014-04-20 16:14:02"
}
```

Note that the user entered the Innovation Zone on third floor around 13:46 and exit that area around 16:14. That shows the user spent two and half hours in the innovation zone on third floor.

Taking a Break while Studying

When someone is studying in an area and then they leave for a short time and come back to that area again, that can mean they were taking a break, meeting someone in another place, picking up a book, etc. For instance this kind of information can be extracted out of the following traffic data:

```json
{
  "device_id": "device id 2",
  "area": "3rd Floor Innovation Zone",
  "activity": "Exit",
  "occurance_time": "2014-04-20 18:35:37"
}
{
  "device_id": "device id 2",
  "area": "3rd Floor Innovation Zone",
  "activity": "Enter",
  "occurance_time": "2014-04-20 18:44:49"
}
```

Notice that the user was in the innovation zone on third floor and then exit the area at 18:35 and entered (came back) again at 18:44. This can mean the user left the area temporarily for some reason, maybe taking a break or the like.
**Walking by a monitored area**

Sometimes a person is just walking by a monitored area and that data is technically a noise and needs to be ignored in the information extraction and data analysis. It is easy to detect such information based on the gap between the Enter and Exit traffic activities. Something like the following shows this scenario:

```json
{
    "device_id": "device id 3",
    "area": "4th Floor Study Room",
    "activity": "Enter",
    "occurance_time": "2014-04-20 17:35:07"
}
{
    "device_id": "device id 3",
    "area": "4th Floor Study Room",
    "activity": "Exit",
    "occurance_time": "2014-04-20 17:35:44"
}
```

If you look at the above data, you can see that the user entered the study room on fourth floor at 17:35:07 and then exited the area seconds later. This can only mean the user walked by the monitored area and within seconds crossed monitored region border twice.

**Hourly Report of Areas Occupancy**

One of the several different reports extracted out of the collected and processed data, is an hourly report of areas occupancy in the last two weeks of monitoring areas. This diagram shows that as we get closer to the exam week, the population of the areas grows. It has its highest value during the exam week until it drops again in the last day of exams since a lot of students are done with the semester at this point.
Closer look at hourly population in different areas

The diagram above shows the population of monitored areas in the last two weeks. The following diagram presents a closer look at the hourly population of each of the monitored areas in one day from 9:00 AM to 9:00 PM.

Figure 8. Hourly report of monitored areas population in the last two weeks. The peek of the population is in the exam week and it drops again at the end of it.

Figure 9. Hourly population of three monitored areas between 9:00 AM and 9:00 PM in a day.

From this diagram, further analysis can be applied. For instance, we can figure out around what time of the day those areas are pretty crowded or during what periods of the
semester the areas are the busiest. Also some already known facts are visualized in this kind of diagram, for example around the lunch time, the population drops suddenly or close to 6:00 PM since a lot of the students have classes from 6:00 PM to 9:00 PM.

**Number of devices in areas**

Another interesting information from the traffic data is the number of devices noticed in areas on a daily basis. The total number of devices noticed during the occupancy monitoring was 80. The following diagram shows a distribution of these devices in different days during this time.

As you can see, there are some interesting information that can be extracted out of the traffic data, which ultimately can help students to find less crowded areas for studying in a real-time matter based on different situations, events, contexts and help the library staff to get a perspective about the occupancy of the areas and related information.

**Lessons Learned**

There were many interesting learning and experiences involved in this project. Few of them are mentioned here.
Advertisement matters more than you think. We did another research project last year, which was about the impacts of *Gamification and Shared Situated Display on User Engagement* and we did quite a good advertisement on that application and how it works. [9] We spent a fair amount of time promoting the application and it had a very good results. A lot of people started using the application and we collected more and more data. This time for the *occupancy monitoring* application, we did not spend that much time promoting the application and introducing the features to the users and for that reason, many users were not aware of its existence and how it works. Specially, since the feature is dealing with Bluetooth and most people have their device’s Bluetooth off naturally, it was an issue and we did not collect a big amount of data. Most of the times, the number of people that were noticed at the same time in monitored areas was less than 20 or so which shows not many people know about the feature or they are concerned about the battery life and using Bluetooth.

It is amazing how fast *Apple* users upgrade their installed applications. Few days after launching the new version of the application, which includes the features associated with *context awareness* and *iBeacons*, majority of users upgraded to this version. This shows the impact of simplicity and ease in installing updates and upgrading applications handled by *Apple*. The following diagram is from Google Analytics page of the Laker Mobile application and shows the percentage of the users running the latest version of the application and the previous version.

![Diagram showing the percentage of users running the latest version of the application and the previous version.](image)

Figure 11. The green area shows the latest version and the orange area shows the previous version.

One very important lesson for us was that in software development there is a possibility of facing an obstacle that is generated by another department or team. In the case of this project it was the IT department. As we mentioned before the *iBeacons* and beacon devices utilize the BLE signals. For that reason, they were
concerned about the interference of these signals with WiFi and creating a problem for students Internet connection. This is not an issue with beacon devices at all, BLE, WiFi and many other technologies are all using the 2.4 GHz frequency. But Bluetooth and WiFi have different dedicated channels, so it is unlikely for them to interfere each other. According to the fact that WiFi is everywhere nowadays, Bluetooth® Smart (BLE) is designed not to use a specific frequency and it uses a technique called Adaptive Frequency Hopping which reduces the interference dramatically. For that reason, Bluetooth® Smart and WiFi can coexist with no problem. [10] This conversation with IT department delayed our final installment of beacon devices couple days.

- When it comes to dealing with a fairly new technology, one side of it is technical learning and solving problems and removing unfamiliar obstacles. Another very important side of it is promoting the technology and the application using that technology correspondingly. A lot of people do not know about a new technology and how it can improve the situation for them and create a better user experience. Specially, when that technology or application is dealing with some sensitive areas. In the case of this application there were two major concerns, a) a unique identifier included in the submitted traffic data and b) utilizing the iOS device Bluetooth.
  - The Unique Identifier, which is included in the submitted traffic data, is preserving the privacy of the user completely. This identifier is being generated by idForVendor method in Objective C and is accepted by Apple. It is based on the application, vendor, etc. and if the user re-installs the application the value of this identifier will be different. There is no way to associate an identifier with a person like MAC address or the similar identifiers.
  - Bluetooth Low Energy is a sensitive resource of an iOS device from a user’s perspective. They are worried about battery efficiency and think that since the Bluetooth is being used by the application, it makes the device dies sooner than usual. But that is not the case. The CoreLocatoin library BLE scanning for monitoring regions and ranging beacons is smart and efficient and it works in a battery efficient way depending on different situations of the device and application. We did some tests and monitoring on this issue and the difference on an iPhone 4S, which had the Bluetooth on for a whole day along with the application running in background was thirty minutes earlier than usual. But that is not even an issue in the case of this application, because users need to turn on the Bluetooth only when they enter the library in order to provide data and get accurate population and heat map information.
• Working with a new technology means there are not much information and examples in production in order to learn from. Mostly experimental projects are under development and the best source of learning about the technology and how to work with it would be its source code, mailing lists, issue trackers, developer forums, etc. In the case of this iBeacons, relatively speaking, there was not much information on StackOverflow and developers of the Estimote and other companies working on this technology answered most of the questions asked by developers. Hence, the ability to read source code and extract relevant information out of it based the problem at hand is a very important skill that needs to be improved further when it comes to work with new technologies and techniques.

• When dealing with a new and unfamiliar technology, majority of the time is spent on getting to know the concepts, ideas, problems associated with that technology, its potential for solving the problem at hand, etc. When the technology is becoming more and more familiar and it has a potential of being useful in different areas and domains such as the technology in the case of this project, iBeacons and Context-Awareness, it is a good idea to create an infrastructure and sort of a platform which is as abstract as possible and extracts out the common and reusable elements. This will make it easier to apply this general solution and platform to different domains and problems and the only parts specific to the new domain and problem needs to be fed to this platform and infrastructure. In the case of this project, the interfaces for interacting with iBeacons, beacon manager monitoring and ranging, client-server interaction, etc. are all crafted in a reusable fashion. The core logic of the application, which solves the problem at hand, is easily interchangeable and in order to solve another problem in a different domain, most of this framework and infrastructure can be reused with the new logic and solution.
Bibliography


2. Netflix Prize was a Machine Learning and Data Mining contest setup by Netflix in order for programmers to come up with a great Recommendation Engine for Netflix. Details on this contest can be found at [http://en.wikipedia.org/wiki/Netflix_Prize](http://en.wikipedia.org/wiki/Netflix_Prize).

3. Prophet Kitchen developed a prototype of an application for The Rubens House in Antwerp, Belgium similar to this example. A demo of this application can be seen at [http://vimeo.com/84760383](http://vimeo.com/84760383).


5. Beacon devices proximity UUID, major and minor number and the relationship between them are explained in more details in the API documentation of Estimote at [http://estimote.com/api/index.html](http://estimote.com/api/index.html).

6. The Source Code for Estimote iOS SDK is at this GitHub repository [https://github.com/Estimote/iOS-SDK](https://github.com/Estimote/iOS-SDK).


9. You can read the complete report/article of this research project at [http://scholarworks.gvsu.edu/cgi/viewcontent.cgi?article=1154&context=cistechlib](http://scholarworks.gvsu.edu/cgi/viewcontent.cgi?article=1154&context=cistechlib).