2015

Context-Aware Mobile Apps using iBeacons: Towards Smarter Interactions

Edward R. Sykes
Sheridan College, ed.sykes@sheridancollege.ca

Stephen Pentland
Sheridan College, stephen.pentland@sheridancollege.ca

Saverio Nardi
Sheridan College

Context-Aware Mobile Apps using iBeacons: Towards Smarter Interactions

Edward R Sykes
Sheridan College
1430 Trafalgar Road
Oakville, Ontario, Canada
+1 (905) 845 9430 Ext 2490
ed.sykes@sheridancollege.ca

Stephen Pentland
Sheridan College
1430 Trafalgar Road
Oakville, Ontario, Canada
+1 (905) 845 9430
stephen.pentland

Saverio Nardi
Sheridan College
1430 Trafalgar Road
Oakville, Ontario, Canada
+1 (905) 845 9430
saverio.nardi

ABSTRACT
In this paper we describe four mobile apps for iOS devices that use Bluetooth Low Energy iBeacons to provide contextual relevance and personalized experiences for the user. The applications span a number of vertical markets including asset tracking, food transportation logistics and health care. We developed these apps in collaboration with an industry partner located in Mississauga, Ontario, Canada. In this paper we present the relevant background of work in this area, the architectural framework that we designed and developed to support these context-aware apps, the apps themselves, and report on the findings of real use test case scenarios.

Categories and Subject Descriptors
C.5.3 **[Microcomputers]**: Portable devices—*personal digital assistants (e.g., smartphones).* H.1.2 [**User/Machine Systems**]: Human Factors – *smart interactions, Bluetooth Low Energy Beacons.* H.5.2 **[User Interfaces]**: User-centered design. J.0 **[Computer Applications]:** Mobile application design and development. J.3 **[Life and Medical Sciences]**: Health, Medical information systems.

General Terms
Experimentation, Human Factors.

Keywords
Smart interactions; context-aware computing; beacons; BLE mobile apps; mobile computing; ubiquitous computing.

1. INTRODUCTION
Location awareness is the heart of virtually all context-aware mobile apps [1, 2]. Although GPS technologies have enabled rough estimates of a person's location, unfortunately, it does not provide the accuracy required for context-aware in indoor environments [3]. This is especially true in large multi-level buildings such as retail, hospital and educational environments [4].

The new trend of Internet of Things (IoT) theorizes that *things* (i.e., objects) and people will connect wirelessly. Bluetooth Low Energy (BLE), or *Bluetooth Smart*, is enabling the explosion of an incredible array of devices [5]. In fact, it is predicted that by 2020 30 billion devices will enter into the IoT ecosystem according to ABI Research [6]. Analysts from all over the world recognize BLE as a key enabler in the Internet of Things [2]. The key to *beacons* is in fact this Bluetooth Low Energy communication specification.

The motivation behind this work is to shed light on the following research questions: (1) determine a good architectural model for context-aware mobile apps that leverage iBeacons, and (2) determine the strengths and limitations of iBeacons for context-awareness by designing and developing several prototypes for real-world settings. In this paper, four iOS applications are presented that were designed and developed using iBeacons. The applications span a number of vertical markets including asset tracking, food transportation logistics and health care. We present the architecture that was designed and developed to support this ubiquitous computing framework, and report on the findings of real test use cases. This paper is structured as follows: section 2 presents a background review of similar work in this area, section 3 discusses our initial experiences with iBeacons prior to designing and developing our apps, section 4 presents a high-level description of the apps, section 5 presents the architectural model that we created, section 6 discusses the particulars of the apps, section 7 presents findings and discussion and section 8 provides the conclusion and discusses future work.

2. BACKGROUND
Bluetooth Low Energy became part of the Bluetooth standard in 2010 with Bluetooth Core Specification 4.0 [7]. BLE enables a mobile device to use Bluetooth networking at lower energy levels in an effort to reduce smartphone battery consumption.

In 2013, Apple introduced the iBeacon standard that enables new location awareness possibilities for apps. Leveraging Bluetooth Low Energy, a device, such as a smartphone or tablet with iBeacon technology can be used to establish a region around an object (e.g., shoe in a shoe store). This allows a device to determine when it has entered or left the region, along with an estimation of proximity to a beacon.

The application of beacons has been significant to a variety of markets and its growth is poised to accelerate within the next 5 years [5, 6]. From welcoming and assisting sports fans to their seats in a soccer stadium to providing interesting facts about a nearby museum exhibit, iBeacons provide a gateway to a world of new possibilities for location awareness and smart interactions between devices and iBeacon hardware.

Beacons that are compatible with the iBeacon standard will work with devices that have Bluetooth 4.0—currently these are virtually

all Apple and Android devices (iOS 7.0+, OS X 10.9+, Android 4.3+), collectively representing 96.3% of the current smartphone OS market share worldwide [3, 8].

Implementations of the iBeacon standard advertise the following information via BLE: the Universally Unique Identifier (UUID), major and minor values [9, 10]. Collectively, the UUID, major and minor values provide the identifying information for a beacon. The information is hierarchical in nature with the major and minor fields permitting subdivision of the identity established by the UUID (please refer to Table 1).

For example, consider a national-wide retail store. The UUID would be shared by all locations. This allows a device to use a single identifier to recognize any of the stores with a single region. Each specific store in San Francisco, New York and Boston would be assigned a unique major value, allowing a device to identify which specific store it is in. Within each individual store, departments would be assigned unique minor values. In this way, app developers can customize the user experience based on context-aware information provided by beacons.

Table 1. iBeacon advertisement information via BLE

Field	Size	Description
UUID	16 bytes	Specific to the app and deployment use case.
Major	2 bytes	Specifies a specific iBeacon and use case. For example, this could define a sub-region within a larger region defined by the UUID. (e.g., 1=San Francisco, 2=New York, 3=Boston).
Minor	2 bytes	Further subdivision of region or use case, specified by the application developer (e.g., 10=Sporting Goods, 20=Automotive, 30=Housewares).

At the onset of this research, we recognized that there are only a few hardware companies that build iBeacon compliant beacons. We surveyed the market for products based on the following critieria: (1) availability, (2) price, (3) well-designed and supported SDK, (4) ease of use and configuration and (5) no extraneous continuous fee for services. We evaluated the following implementations of iBeacons using this criteria: Pixie [11], Bleu Station Beacon Series 100 [12], Estimote [13], Roximity [14] and Gimbal [15]. The results are shown in Table 2. During the review and evaluation process, it became clear that the Estimote beacons were the most feasible since they provided the greatest potential to satisfy the requirements of the proposed research. Furthermore, Estimote above all the others has gained significant popularity because of its indoor location SDK and support commitment for applications developers.

Table 2. Evaluation of various iBeacon products

Criteria	Pixie	Bleu	Estimote	Roximity	Gimbal
Availability	✓✓	✓	✓	✓	✓
Price	✓✓	✓	✓✓	✓	✓✓
Well designed / supported SDK	—	✗	✓✓✓	✓	✗
Ease of use / configurable	✓	✓	✓✓	✓✓	✓
no extraneous services	✓	✗	✓	✓	✗

Legend: ✓: acceptable, ✓✓: good, ✓✓✓: very good, ✗: poor, ✗✗: very poor, —: Not applicable.

2.1 Estimote Beacons and Nearables

Estimote offers two distinct products that satisfy the iBeacon specification: *Beacons* and *Nearables* (also referred to as *Stickers*). Both comply with the iBeacon standard but also provide additional information beyond this standard. Table 3 presents the prominent characteristics of Estimote beacons and nearables. Nearables offer additional information including orientation, temperature and motion. Collectively, this provides more opportunities for personalization of location-aware and context-aware apps [2, 16, 17].

Table 3. Estimote's Beacons and Nearables Characteristics

Characteristic	Beacons	Nearables
UUID	✓	✓
Major + Minor	✓	✓
Type of Nearable	✓	✓
Received Signal Strength Indication	✓	✓
Orientation in space	✗	✓
Temperature	✗	✓
Motion in *x*,*y*, and *z*	✗	✓

Legend: ✓: supported, ✗: unsupported.

3. INITIAL TESTING WITH iBEACONS

We set out initially conducting a variety of real-world tests to determine the strengths and limitations of Estimote beacons and nearables. We set up an indoor test room where 4 beacons were positioned. We erected 4 lightweight panels, with each one opposite one other in a rectangular configuration. Each panel then had a single beacon placed in the center of the crossbar on each panel. This configuration allowed easy modification of the size and configuration of the testing area. We were able to test functionality such as region entry and exit events, ranging accuracy and position reading. Please see Figure 1.

Figure 1. Test site: An indoor room equipped with iBeacons.

We initially focused on location / proximity of a device from a beacon since this is its cornerstone functionality. We later

explored the other characteristics (orientation, temperature, motion). The way in which signals from a beacon are detected influences the design of a mobile app in terms of the user experience. When a device detects a beacon's signal, it uses the strength of the signal Received Signal Strength Indication (RSSI) to determine both proximity to the beacon, as well as the accuracy of its estimation of proximity. The stronger the signal, the more confident the device will be about the proximity.

Estimote beacons can be configured and modified easily. This facilitated rich experimentation of accuracy determination under a variety of indoor conditions such as, configuring walls and obstacles that would interfere with the signals, varying the frequency of beacon advertisements, varying the transmission power and, of course, the location of beacons relative to the receiving device. Table 3 presents the prominent findings from our experimentation under good conditions (i.e., iOS device had line-of-sight to the beacon and as few as possible obstructions in the vicinity). We discovered that the RSSI value tended to fluctuate significantly partially attributed to external factors such as absorption, interference, reflections (i.e., multi-path fading) or diffraction [18]. We also discovered that our measurements were quite different than Estimote's reported specifications for their nearables [13, 16].

Table 4. Estimote's Nearables: RSSI, distance classification, and actual distance from empirical measurements.

RSSI range *(deviation)*	**Distance Classification (Zone)**	**Actual Distance** range *(deviation)*
-60 to -75 dBm *(10dBm)*	Immediate	0-2.5cm *(2.5cm)*
-75 to -85 dBm *(10dBm)*	Near	2.5-15cm *(10cm)*
-85 to -100 dBm *(10dBm)*	Far	15cm-7m *(30cm)*

Legend: Distance Classification ("Zones" according to Estimote): "immediate", "near" and "far." [13]

We also investigated the energy consumption of mobile devices running apps that use iBeacons. Studies have shown that energy consumption is dependent on the BLE chipsets in the mobile device and is also directly proportional to the number of iBeacons the device is currently scanning [19]. Furthermore, energy consumption is also dependent on the sampling rate (i.e., the rate at which the beacon's advertised signals are received by the app on the mobile device). For instance, the output rate for Estimote nearables is 1Hz to 5kHz and the sampling rate within the app can be set from 950ms to 10s. Using a 950ms sampling rate consumes the most battery power. Collectively, these findings and experiences helped guide us in the design and development of the architectural model and context-aware apps in this project.

4. HIGH LEVEL DESCRIPTION OF APPS

4.1 Asset Tracking App

The purpose of this application was to track assets for end-users. The user's iOS device calculates its position in relation to the tracked asset (identified by the iBeacon). When the asset is outside of the device's range, a notification triggers on the device alerting the user. If the user does not take steps to relocate the asset to a closer proximity, additional notifications are issued. Our application applies this functionality to the logistics industry. A nearable is attached to a set of keys and our application on the iOS device frequently computes the distance between the two. When the keys fall outside of a preprogrammed range, notifications appear on the iPod. As long as the device remains outside of the range, notifications continue to appear. This system can better protect a company's assets and increase asset retention.

4.2 Food Transportation App

The initial motivation for this application was to explore the possibilities of using BLE technology combined with connected mobile devices to offer low-cost temperature monitoring for food delivery vehicles. By using the existing mobile device provided to the delivery driver, and one or more low-cost iBeacon products capable of reporting ambient air temperature, this solution could possibly result in a solution that was not only cheaper, but also more configurable and modular than many current solutions [2, 5, 13].

In this solution, the iBeacon devices would, at regular intervals, broadcast their identifying information and the current ambient temperature of the space it is currently deployed in (e.g., refrigerated section of a food truck). The driver's device would take and process these readings to analyze if any particular reading either fell below or rose above the acceptable range.

4.3 Mobility Assistance App

The purpose of the mobility assistance app is to provide support for persons requiring a mobility assistant device (e.g., wheelchair, walker, etc.). Using a Bluetooth Smart device with contextual awareness, a Mobile Assistant (MA) device would know when it was in a position other than upright. An alert would be provided to the user on a device (e.g. smartphone, tablet, iPod, etc.) if the MA device were in an undesirable position. The application notifies the proper authority after a preprogrammed period of time if the user does not interact with the alert. The Bluetooth Smart device has an accelerometer included with it allowing it to know its current position in three-dimensional space. The software enumerates each of the 6 possible orientations (i.e., on its front/back, on its legs/head, on its right/left side). This enables specific positions of the device to be communicated in the form of an alert to the user. Our application specifically uses the example of a wheelchair using an iOS device. Any incidents where the wheelchair is not in an upright, normal position causes an alert to be issued to the user and gives him/her a set amount of time to respond before dialing the proper authority. Our application also sends any incidents to a database for future analytics or management within a central care facility.

4.4 Home Healthcare App

The goal of this app is to explore how best to support a homecare professional while performing routine healthcare activies with the patient in his/her home. Patient locations and lists of medical equipment that should be present at that location are registered on a back-end server. The application uses readings obtained from the nearable to determine if the items are indeed present and to present tasks for the homecare practitioner to perform.

Each patient has a master beacon assigned to him/her and the identifier for that particular beacon maps back to a set of data corresponding to that patient's name[1]. This master beacon also

[1] Estimote nearable identifiers can take on one of the following values: Dog, Car, Fridge, Bag, Bike, Chair, Bed, Door, Shoe or Generic.

enables a list to be generated for both the particular health care items that are supposed to be present at this location and the tasks that must be done by the health care provider. In the list of equipment, each piece is again mapped to a particular beacon identifier value. This means that as the practitioner enters the location, the device can read the surrounding nearable and check that all required items are in fact present at the location.

Using this system, we can provide a way to reduce human error in health care. This implementation provides a way for a health care worker to have patient information that is always up to date and ensures that the information that the practitioner is working with belongs to the proper individual.

5. Architectural Model

This section presents the architectural model that was designed and developed for this suite of context-aware apps.

Our systems were comprised of 4 general parts (please refer to Figure 2):

1. The server, which was shared among all concept apps
2. The client (i.e., context-aware mobile app)
3. The database
4. The iBeacons (Estimote beacons or nearables)

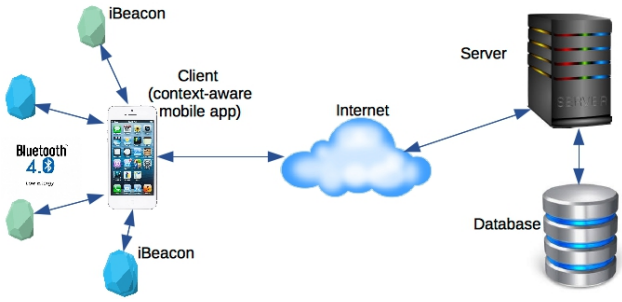

Figure 2. High-level architectural model.

Each app was designed to read location context information from surrounding iBeacon devices and combine it with the current state of the environment around them. When the conditions fell into specified ranges, the apps would then communicate with the back-end server to record the exceptional state (temperature out of range, devices not present, at a patient's home, etc.).

The applications used HTTP POST and GET requests to transmit or receive data from the server. POST requests were sent using the x-www-form-urlencoded format, and all server responses were sent in JavaScript Object Notation (JSON) format [20]. We chose to use JSON for multiple reasons; JSON:

- is an open standard for data-interchange
- is easy for humans to read and write
- is easy for machines to parse and generate
- enables flexibility and ease in designing our document structure for data required in our mobile apps

Upon receipt of data from one of the client applications, the server, which was written using Python 3 and Flask 0.10.1 would parse the incoming information and store it in a SQLite database file [21-23]. In the other direction, a client app would make a GET request to a particular URL endpoint and the server would get the appropriate data from the SQLite instance and convert it to JSON format for easy consumption by the client.

When designing the initial prototype, we took inspiration from the RESTful design [24]. This can be observed in the way the GET endpoints are structured. We used this to provide a simple interface for our client applications to retrieve data from the server and provide new data for future consumption.

Another feature of the server is to host multiple simple web pages with map data to demonstrate the relative ease with which such a system could collect and present data that, prior to BLE and connected devices, required specialized and complex eqipment such as GPS and temperature sensors from multiple vehicles in a fleet, or when a patient's wheelchair had fallen over.

5.1 Server Characteristics

The core responsibilities for the server are:

- Provide HTTP endpoints for client communication
- Accept properly formatted POST requests
- Return properly formatted JSON data for correct GET requests
- Offer a mechanism to display the stored data
- Store all received data in a persistent format
- Provide robost scalablity

The server component was built on commonly used technologies such as Linux, Nginx, Python and SQLite. These technologies inherently provide the desired characteristics, particularly robustness and scalability. The server scales to meet the number of mobile devices that are currently asking for data related to a particular beacon.

The client, on the other hand, must handle the possibility of reading multiple beacons simultaneously. While Apple's iBeacon standard is equipped to handle this, there is the potential for increased battery usage. Informal testing has shown that there can be a moderate increase in battery usage, ranging from one or two percent to over ten percent excess drain [19]. This is always dependent on the individual device and scanning interval [19].

5.2 Client Characteristics

The client characteristics presented below are required for the architecture designed (please refer to Figure 2). These client-side properties are common amongst all of the context-aware apps created in this project:

- Establish and maintain a data connection
- Properly create and send HTTP requests, either GET or POST
- Provide access to a Bluetooth radio capable of implementing the BLE standard
- Include the capacity to read information from BLE iBeacon items and process contextual data in relation to its desired operation

Collectively, these properties represent a cohesive foundation upon which the context-aware apps designed and developed are based. The diagram in Figure 2 shows the features that all of the applications share. The mobile device receives contextual data from multiple beacons that are in the proximity of the device.

Using this data, the device can process the input and communicate over the internet to the back end server. At this point the server can process the incoming request and provide an appropriate response for the client.

6. DESIGN CHARACTERISTICS

This section discusses in depth the design and key characteristics of the apps created.

6.1 Asset Tracking App

The objective of this application is to provide notifications to drivers through their iOS device when they are too far from their keys to their vehicle.

Keys for delivery vehicles include a BLE device with a universally unique identifier. Our application executes on a driver's iOS device. When the application is opened it attempts to find the driver's keys as shown in Figure 3. Once the keys are located by the application (see Figure 4), the application can be placed in the background.

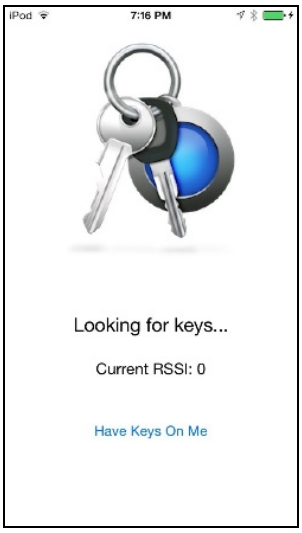

Figure 3. Key Loss App – Ranging (searching)

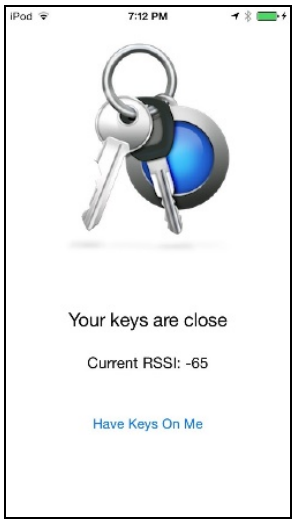

Figure 4. Key Loss App – within range of keys

The application continues running in the background, monitoring for the key's location approximately once-per-second. As long as the keys remain within the short range threshold, based on the RSSI from the BLE device, the application will simply continue to monitor.

When the keys become located too far from the driver's iOS device, the application icon changes (see Figure 5) and an internal timer begins to count down.

If the application cannot register the key's BLE device again within a predefined time, the driver receives an alert on their iOS device (see Figure 6). Notifications will continue to occur at regular pre-determined intervals if the keys are not retrieved.

This application can be easily repurposed to a number of different scenarios where asset tracking is needed (e.g., dogs, cats, purses, wallets, etc.). Furthermore, information such as GPS location and time of day can be sent back to the asset owner, thus informing him/her of the situation and allowing him/her to analyze precisely where and when the asset was left behind.

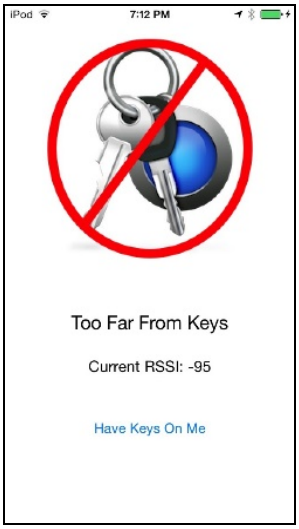

Figure 5. Key Loss App – Ranging

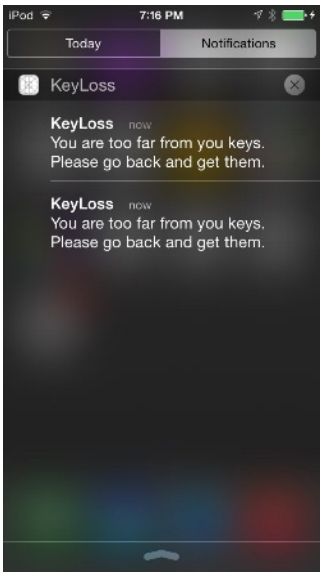

Figure 6. Key Loss App – Notification Centre message

6.2 Food Transportation App

The goal of this application was to examine the feasability of incorporating BLE devices combined with mobile devices to offer reliable, low-cost temperature monitoring integration.

The layout for the system was to place one or more BLE devices into the refrigerated compartment of a food delivery truck. The other end of the system involves a mobile device that most delivery drivers would already have. On a schedule of approximately once-per-second, each beacon would transmit its current temperature data to the mobile device.

Using the transmitted temperature data, the mobile device could analyze the readings and, if one were outside the specified range, an alert was issued to the driver and also to the back-end server system.

One of the proposed uses for this type of system was to have the mobile device transmit GPS data to the server during a food-temperature emergency. This would allow the company operating the system to determine if it were possible to complete the day's deliveries or whether the truck would have to come in for service immediately. Please refer to Figure 7 for a demonstration of the in-app alert.

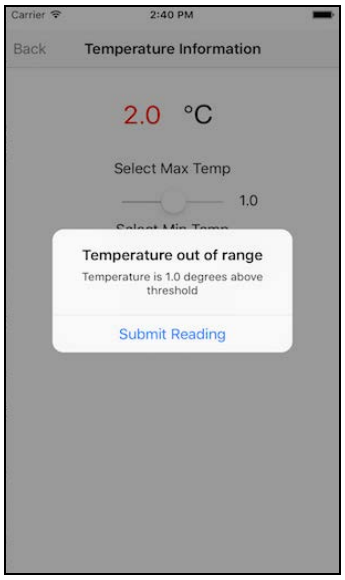

Figure 7. Food transportation app – in-app alert

When a reading that is outside the desired range is observed, the application notifies the driver that an anomaly has been observed with the readings and states the deviation of the temperature. At the same time, a POST request is sent to the server. In our demonstration, the request contains data on the temperature reading, and the current location of the vehicle as observed via GPS location request. Please refer to Table 5 for a general view of the data transmitted presented in an HTML table.

This context-aware app shows how, with nearables, a modest amount of work and inexpensive hardware, temperature monitoring can be added to a fleet of vehicles for a food transportation company that will enable logistics via mobile devices data transmitted by the beacons to optimize their management and transportation of food goods.

Table 5. Food Transportation Central Reporting Web Page.

Readings from Truck 3

Date	Temperature Reading	Latitude	Longitude	Map View
2015-05-01 11:42:46.107382	19.44	43.468757	-79.698409	Map
2015-05-05 16:33:38.332978	22.5	43.437909	-79.722187	Map
2015-05-05 16:34:33.592784	22.5	43.437931	-79.722221	Map
2015-05-05 17:20:07.697639	28.38	43.437825	-79.722248	Map
2015-05-05 19:35:08.409914	21.94	43.437825	-79.722126	Map
2015-05-05 20:08:57.533440	23.31	43.437895	-79.7222	Map

6.3 Mobility Assistance App

The motivation for this application is to illustrate through continuous monitoring for individuals requiring a wheelchair, their safety can be dramatically increased. Current solutions for individuals that require emergency assistance also require physical input. An example is the medical alert pendants with buttons to be pressed in emergency situations. Our application attempts to solve the problem of an individual being required to press a button or through voice activation, trigger the appropriate response.

A BLE device is attached to an individual's wheelchair that is in a normal upright position. The BLE device's current orientation is registered as normal in our application running on an iOS device. The application monitors the BLE device's orientation and motion (see Figure 8) on the wheelchair once-per-second.

Figure 8. Wheelchair App – (wheelchair in motion)

As long as the wheelchair is in a normal upright position (see Figure 9) the application continues to monitor the BLE device's context.

Figure 9. Wheelchair App – (wheelchair in normal position)

When the wheelchair is in a non-upright position, the application registers the BLE device's incorrect orientation. The image in our application changes to match the orientation of the BLE device, turns red (see Figure 10) and a short internal timer begins. The timer is used to account for a potential incorrect reading or accidental incorrect wheelchair position (e.g., when the wheelchair is being stowed away in a car for instance).

Figure 10. Wheelchair App–wheelchair has fallen on its right

If the wheelchair position is not righted, an alert appears requesting input from the individual, similar to the emergency pendant (see Figure 11). If the individual feels they are okay, they inform the application by pressing the correct response to the alert and the application begins monitoring again. However, if the user is able to interact with the alert, but feels they require assistance or the user is unable to interact with the alert, the application will dial the local emergency services (see Figure 12).

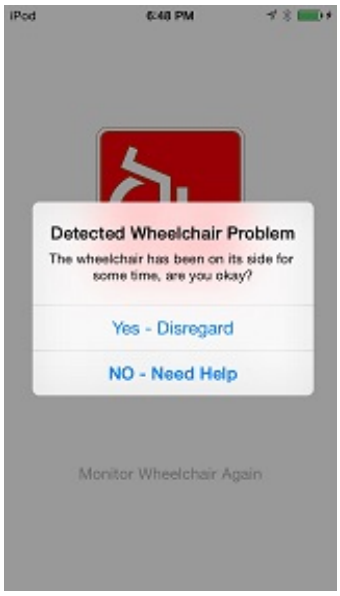

Figure 11. Wheelchair App – Detected Wheelchair Problem.

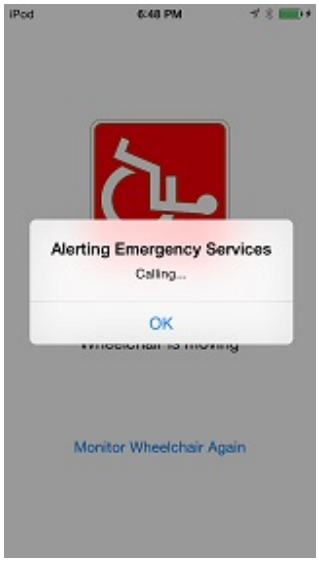

Figure 12. Wheelchair App – (detected Wheelchair Problem -- Alerting Emergency services)

Each incident is also reported to a database. Our application sends the date, time and incorrect orientation of the wheelchair to the database for future analysis and reporting.

While our application is applicable only to a wheelchair, it could be used for multiple scenarios. We considered the possibility of fully replacing the pendants so the individual could have peace of mind knowing that in an emergency scenario where they were unable to physically press a button, emergency responders would be notified.

A final consideration is a large health center or retirement community (i.e., Center) could provide similar applications to their residents. Staff could be alerted 24 hours a day of an incident with any resident. The application could also register and use BLE devices mounted on hallways and rooms to provide coordinates to

the Center's staff allowing responders to know exactly where to find the resident.

Each BLE device's UUID, major and minor could be mapped to specific rooms, hallways and residents in the Center. When an incident occurs, the iOS device would send all ranged BLE device UUIDs to staff.

6.4 Home Healthcare App

When creating the home healthcare application, our goal was to examine ways to mitigate a potentially dangerous problem in home healthcare scenarios—human error. When dealing with potentially life-threating conditions, or when handling medication tasks, it is easy to imagine the potential impact that human error could have.

The design of this system leverages both the universally unique identifier and limited range of the beacons to simulate the scenario of a home healthcare worker arriving at a patient's residence. Once the worker's mobile device is within range of a beacon that is transmitting, a GET request is sent to the server containing the UUID value of the beacon that was just encountered.

Should a match be found on the server that corresponds to the provided identifier, then a JSON formatted response is returned to the client. In our current example, the response consists of the following elements:

1. The patient's name
2. A list of all medical items that should be present at the patient's residence
3. Checklist of tasks to be performed by the healthcare provider when attending to the patient

Each item in a particular location is mapped to a UUID for a different beacon that would be attached to the device itself. As the surroundings are scanned, each individual beacon is read, and the corresponding item is marked as present on the list view. During each scan interval, the beacon details are read and iterated through to determine which items are in the vacinity of the mobile device.

This allows for real-time updates as to whether all items are present, and provides the practitioner easy access to information about equipment that is missing without having to resort to guesswork or intense searching.

Additionally, this method removes the possible implications of the practitioner failing to account for an expensive piece of equipment, ordering or bringing a new unit to the patient's home at a later time. It can also serve as a reminder for equipment checks that could have otherwise been overlooked when the practitioner is busy or trying to get their duties done in as efficient manner as possible.

As the UUID for each attached nearable is read, the status of the device is changed to indicate that it is present. During scanning, should a particular piece of equipment no longer be within the scanning proximity of the device, its status will revert to not being present. Figure 13 depicts the user interface during scanning.

At any time during or after the scanning, the healthcare practitioner may transition to the *Tasks* tab. This view presents a table containing all of the individual tasks that should be completed while the practitioner is on the premises. For easier tracking, each task may be deleted from the list presented on the device. This deletion has no effect on the original list that resides on the server. Please refer to Figure 14 for more detail on how this looks to the practitioner.

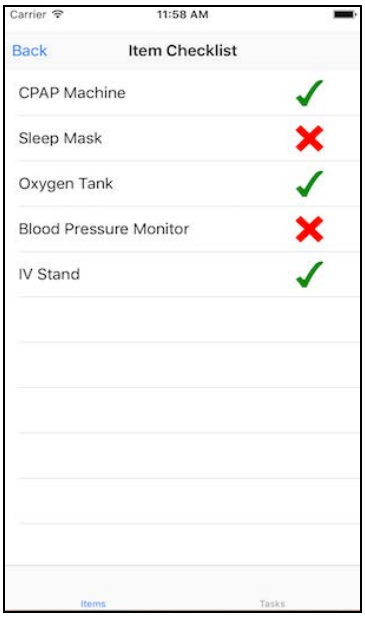

Figure 13. Home Healthcare App, *Item* checklist.

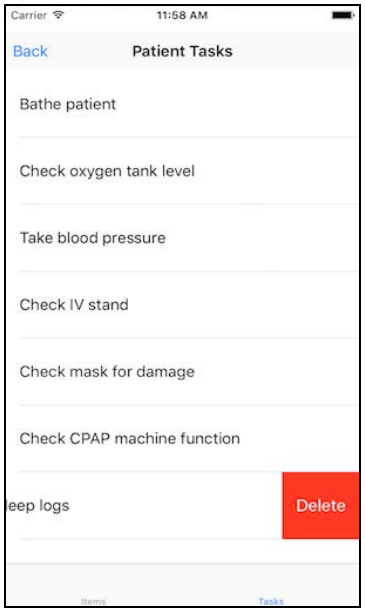

Figure 14. Home Healthcare App, *Task* list.

The data regarding the patient's items and tasks are transmitted from the server in JSON format, with an array of objects map each item to a given UUID value. Tasks are also included in the same fashion, with a simple array of string values denoting each individual task that should be completed. The only other information that is transmitted is the patient's name. Please refer to Figure 15 for the layout of the transmitted information.

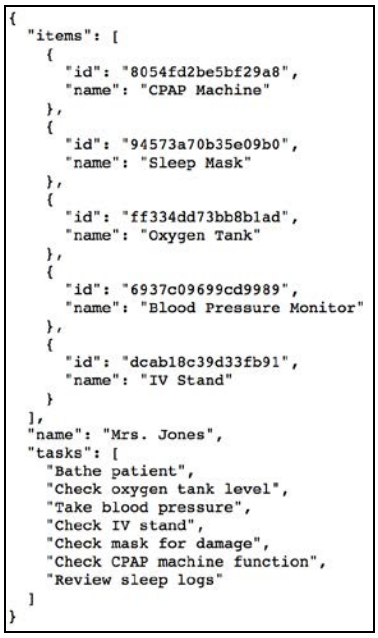

Figure 15. Home Healthcare App, sample data.

7. FINDINGS AND DISCUSSION

This section presents the findings from our research. We found the strengths of the Estimote beacons and nearables were:

- Beacons are more accurate for indoor positioning than wireless access points. They achieve a margin of error of approximately 2 meters, whereas access points are greater than 10 meters [5, 25].
- iBeacons offer a low energy alternative to traditional GPS and provide accurate indoor positioning.
- iBeacons provide contextual information including identification, orientation, temperature and motion.
- iBeacons range well within a 50 meter radius in perfect conditions, however absorption, interference or diffraction from external objects affects them negatively, and in some situations renders them unusable [7, 16, 25].
- While optimized for one-to-one connections, beacons can handle one-to-many connections as well [11].

We found the following weaknesses with the Estimote beacons and nearables:

- Signal strength is prone to fluctuate depending on its environment, negatively impacting proximity accuracy.
- Beacons can only broadcast and will not allow for private communication.
- Packet size for communication is small at 27 bytes, however communication bursts can occur every 3ms, depending on the configuration specified.
- Current nearable hardware is prone to failure.
- Battery life is negatively affected by beacon settings that provide greater accuracy (increased power transmission and frequencies dramatically affects battery consumption).

There is also a caveat regarding the indoor positioning when compared to conventional GPS outdoor positioning. While GPS is focused on fixing a specific location on the Earth's surface, iBeacon contextual awareness is very different. It is up to the developer of the application to determine what a particular context means when read. Generally, it is of greater importance to know the proximity of the user and device to a particular location than the actual global position.

Our experimentation with both the Estimote beacons and nearables revealed that plug-and-play application in real use case scenarios (e.g., retail store, hospital or education institution, etc.) is unlikely. Settings for both of Estimote's products require broadcast timing and signal strength adjustment to provide more accurate interaction with the iOS device.

The Software Development Kit (SDK) from Estimote also has some problems. Contrary to the documentation, we found that ranging for any beacons and nearables did not work with our iPod devices. This includes Estimote's own example applications. We created our own ranging algorithms instead which identified specific beacons or nearables. That is, we discovered that if we ranged on a particular *type* of nearable (e.g., car, or dog for instance), it would fail. As a result, we resorted to specifying the precise hardcoded identifier for the nearable instead. This approach proved to be much more reliable.

In our test office environment (see Figure 1) we experienced some interference and reflection of signals that caused our applications to incorrectly locate the iOS device in the room. Adjusting the signal strength helped to alleviate this problem.

We found that beacons and nearables work best in open spaces and large rooms where reflection and interference is minimal. We experimented in a large public space and found the beacons and nearables were able to range quite accurately.

We experimented with the speed our applications and iPod devices took to find beacons and nearables. A regular walking pace allowed our application to find these iBeacons. However, when a user runs by an iBeacon, the application will fail to recognize that the iBeacon was passed. As a result of our experimentation we wouldn't recommend the current technology for speeds greater than typical walking pace.

We did not experiment with many devices attempting to interact with a single beacon or nearable at a time, although this is possible in the specification.

Estimote nearables offer a wider range of application than their counterpart beacons as they include better context awareness. The nearables also add the benefit of providing their current orientation (e.g., face up/down, etc.), raw *x*, *y*, and *z* accelerometer readings (for motion detection) and ambient temperature readings all available up to 100ms refresh rates.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented four context-aware mobile apps that use Bluetooth Low Energy iBeacons to provide contextual relevance and personalized experiences for the user. The applications span a number of vertical markets including asset tracking, food transportation logistics and health care. In this paper we presented the current state of the art in this area in terms

of off-the-shelf iBeacons and SDKs available, the architectural framework that we designed and developed, and the context-aware apps created. Lastly, we discussed our findings from real test case scenarios and included the strengths and limitations of the iBeacons in our context-aware apps.

8.1 Future Work

The next phase of this research is already in progress. Recognizing that there are many beacons available in the market, we are currently testing a variety of different implementations of iBeacons, namely, Pixie, Roximity, Gimbal and Kontakt [26]. We plan on testing these beacons from different vendors individually and in combination to determine strengths and limitations. We believe these experiences will be useful to share in future reports.

We are also in the planning stages to conduct a comprehensive usability study of the mobile apps involving real participants. For example, we plan on working with food delivery companies and truck drivers for the asset tracking app and the food transportation logistics app. We are also in discussions with homecare facilities and practitioners to determine how to test and evaluate the health care app in real use case scenarios. A Research Ethics Board application is in progress and we are planning on continuing the research and development in collaboration with the industry partner.

9. ACKNOWLEDGMENTS

Our thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC), Canada's federal funding agency for university based research, http://www.nserc-crsng.gc.ca/index_eng.asp for funding this research under the Engage granting program.

10. REFERENCES

[1] Weiser, M. 1991. The Computer for the 21st Century. *Scientific American*, pp. 94–104.

[2] Perkins, E. 2015. The Identity of Things for the Internet of Things, Retrieved from: https://www.gartner.com/doc/2975217.

[3] Carmody, B. 2015. Beacons: Get to the Point, Retrieved June 14, 2015 from: http://www.trepoint.com/whitepaper/Beacons-GetToThePoint/.

[4] Gissiner, B. 2015. Beacons In Hospitals - Adding value to patients and Staff, Retrieved June 14, 2015 from: https://www.linkedin.com/pulse/beacons-hospitals-adding-value-patients-staff-bryan-gissiner.

[5] Ashton, K. 2014. That 'Internet of Things' Thing. *RFID Journal*.

[6] Cooney, P. 2013. More Than 30 Billion Devices Will Wirelessly Connect to the Internet of Everything in 2020, Retrieved from: https://www.abiresearch.com/press/more-than-30-billion-devices-will-wirelessly-conne/.

[7] Bluetooth. 2015. Bluetooth Smart Technology: Powering the Internet of Things, Retrieved May 2, 2015 from: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx.

[8] IDC. 2015. International Data Corporation (IDC) Worldwide Quarterly Mobile Phone Tracker, Retrieved from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[9] Apple. 2014. Getting Started with iBeacons, Retrieved from: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf.

[10] Newman, N. 2014. Apple iBeacon technology briefing. *Journal of Direct, Data and Digital Marketing Practice*, vol. 15, pp. 222-225.

[11] Pixie. 2015. Pixie – Location of Things Platform Retrieved May 23, 2015 from: https://pixie-production-jarome.netdna-ssl.com/wp-content/uploads/2015/01/Pixie-Location-of-Things-Platform-Introduction.pdf.

[12] Twocanoes. 2015. Bleu Station Beacons, Retrieved June 12, 2015 from: http://twocanoes.com/bleu.

[13] Estimote. 2015. Estimote: Real-world context for your apps, Retrieved Feb 2, 2015 from: http://estimote.com.

[14] Roximity. 2015. Roximity, Retrieved April 2, 2015 from: http://roximity.com.

[15] Gimbal. 2015. Retrieved May 22, 2015 from: http://www.gimbal.com.

[16] Etherington, D. 2014. Estimote Wants To Pioneer 'Nearables' With New Stickers Beacon Hardware. *TechCrunch*.

[17] Stawecki, M. 2015. The state of iBeacons, Retrieved June 14, 2015 from: https://stawecki.wordpress.com/2015/06/11/the-state-of-ibeacons-june-2015/?utm_campaign=iOS%2BDev%2BWeekly&utm_medium=rss&utm_source=iOS_Dev_Weekly_Issue_203.

[18] Rappaport, T. 2001. *Wireless communications: principles and practice*: Prentice Hall.

[19] Aislelabs. 2014. iBeacon Battery Drain on Apple vs Android: A Technical Report, Retrieved July 20, 2015 from: http://www.aislelabs.com/reports/ibeacon-battery-drain-iphones/.

[20] JSON. 2015. Introducing JSON, Retrieved June 1, 2015 from: http://json.org.

[21] Miller, B., and Ranum, D. 2014. How to Think Like a Computer Scientist: Learning with Python: Interactive Edition 2.0, Retrieved March 20, 2015 from: http://interactivepython.org/courselib/static/thinkcspy/index.html.

[22] Ronacher, A. 2015. Flask: web development, one drop at a time, Retrieved May 1, 2015 from: http://flask.pocoo.org/docs/0.10/.

[23] SQLite. 2015. SQLite, Retrieved April 10, 2015 from: https://www.sqlite.org.

[24] IBM. 2015. RESTful Web services: The basics, Retrieved Feb 21, 2015 from: http://www.ibm.com/developerworks/library/ws-restful/.

[25] Apple. 2015. iBeacons for Developers, Retrieved Feb 24, 2015 from: https://developer.apple.com/ibeacon/.

[26] Aislelabs. 2015. The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs, Retrieved August 1, 2015 from: http://www.aislelabs.com/reports/beacon-guide/.