Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Bluetooth Low Energy and Smartphones for Proximity-Based Automatic Door Locks

by

Tim Andersson

LIU-IDA/LITH-EX-G--14/050—SE

2014-06-08

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Linköpings universitet
Department of Computer and Information Science

Final thesis

Bluetooth Low Energy and Smartphones for Proximity-Based Automatic Door Locks

by

Tim Andersson

LIU-IDA/LITH-EX-G--14/050—SE

2014-06-08

Supervisor: Peter Dalenius
Examiner: Johan Åberg

Bluetooth Low Energy and Smartphones for Proximity-Based Automatic Door Locks

Tim Andersson
Studievägen 5 lgh 1405
58329, Linköping, Sweden
tim@ndersson.me

ABSTRACT

Bluetooth Low Energy is becoming increasingly popular in mobile applications due to the possibility of using it for proximity data. Proximity can be estimated by measuring the strength of the Bluetooth signal, and actions can then be performed based on a user's proximity to a certain location or object. One of the most interesting applications of proximity information is automating common tasks; this paper evaluates Bluetooth Low Energy in the context of using smartphones to automatically unlock a door when a user approaches the door. Measurements were performed to determine signal strength reliability, energy consumption and connection latency. The results show that Bluetooth Low Energy is a suitable technology for proximity-based door locks despite the large variance in signal strength.

INTRODUCTION

Bluetooth Low Energy (BLE) is a relatively new technology, originally released by Nokia in 2006 under the name *Wibree*. In 2007, the Bluetooth Special Interest Group (SIG) announced that the Wibree Forum would be merged with the Bluetooth SIG and that the Wibree specification would become part of the Bluetooth Core Specification Version 4.0 as what is today known as Bluetooth Low Energy [14]. The low power consumption of the BLE technology and its ability to connect to smartphones makes it especially suitable for low power sensor devices such as heart rate monitors, cycling computers and thermometers [4]. It is also a suitable technology for use in so called *Smart Homes* as it communicates using radio transmissions in the 2.4 - 2.485 GHz band [3]. As a result, Bluetooth Low Energy does not suffer from the same drawbacks as other technologies used in the home, e.g. infrared (IR) remote controls which require line of sight to the device that is being controlled.

An interesting result of the fact that Bluetooth uses radio waves for transmission is that a signal strength can be calculated. Using the signal strength, it should be possible to estimate the proximity between two Bluetooth devices. The fact that it might be possible forms the basis of this thesis.

Motivation

The market for applications using Bluetooth Low Energy is constantly growing. The Bluetooth SIG expects that by the year 2018, more than 90 percent of all Bluetooth enabled devices will support Bluetooth Low Energy [11]. As more and more smartphones gain support for Bluetooth Low Energy, the possibilities of using the technology for new types of applications increase. One of the possibilities is using BLE to detect proximity, which can then be used in a number of interesting areas; this paper will investigate one of them, namely automatic door locks.

Purpose

The aim of this thesis is to evaluate Bluetooth Low Energy as a technology by focusing on its use in door locks that automatically unlock based on the proximity of a smartphone. This will be accomplished by examining different properties of Bluetooth and comparing the results to what can be found in the literature. By implementing an application for a mobile operating system, the thesis also aims to evaluate what possibilities and restrictions exist in using Bluetooth Low Energy on smartphones.

Research Questions

Four questions have been formulated to aid in the evaluation of Bluetooth Low Energy. The first is a general question pertaining to how suitable of a technology BLE is for automatic door locks, while the remaining three are more concrete questions which act as guidelines.

Is Bluetooth Low Energy a suitable technology for proximity-based automatic door locks?

Is it possible to prevent the door from unlocking when a person approaches the door from the inside? A security issue may arise if the door unlocks whenever a person approaches the door regardless of where the person is located relative to it. It is not difficult to imagine several scenarios where a person would want to avoid unlocking the door automatically. One such scenario might be that an enraged person is trying to get through the door, in which case unlocking the door may result in direct danger to the person inside.
The paper will discuss the possibility of solving this problem using two Bluetooth devices located on either side of the door.

Does the implemented solution affect the smartphone's battery life negatively in a significant way? One of the main features of Bluetooth Low Energy is the low power consumption. As the implemented application is designed to run continuously in the background on a smartphone it is especially important that it has a minimal effect on the battery life. This paper measures the effect on battery life and compares the results to what can be expected based on theoretical data.

Is the Bluetooth Low Energy communication latency sufficiently small to make the technology usable in practice? The main benefit of having a door lock that automatically unlocks based on the user's proximity is that it is more time efficient for the user than unlocking the door manually. If implemented well it should be faster than Radio-frequency identification (RFID) locks that are common in today's world. If there exists inherent latency in using Bluetooth Low Energy, this benefit might be lost. As such, it is essential that the latency is sufficiently small to outperform other similar technologies.

Limitations

The paper will not discuss any security aspects related to the Bluetooth communication.
An application will only be implemented and evaluated for Apple's mobile operating system iOS and not for any other mobile operating system such as Android or Windows Phone.

BLUETOOTH LOW ENERGY

A basic understanding of what Bluetooth Low Energy is and how it works is required to fully understand this paper. This section will give a brief overview of the architecture of BLE and explain a few key concepts, such as BLE beacons. It will also cover what sets BLE apart from previous versions of Bluetooth and how Bluetooth can be used on iOS.

Bluetooth Low Energy Compared to Classic Bluetooth

Bluetooth Low Energy is optimized for devices that require maximum battery life instead of high data rates [1]. While previous versions of Bluetooth are commonly used to transmit a lot of data such as files or audio, BLE is used to transmit small pieces of data such as temperature readings. Bluetooth Low Energy was also designed to achieve much smaller latencies in connecting and transferring data than in Classic Bluetooth.

Architecture

Bluetooth Low Energy devices can either have a *central* role, or a *peripheral* role. Devices with a central role are devices that consume data from devices with a peripheral role. Drawing a parallel to a client-server architecture, a central device can be compared to a client and a peripheral device to a server. In this paper, the smartphone is a central and the Estimotes (covered in the *Beacons and iBeacon* subsection) are peripherals.
In order for a peripheral to be discoverable, peripherals *advertise* continuously by transmitting *advertising packets* at a fixed interval (usually measured in milliseconds). They advertise at all times even if no other Bluetooth device is in range. To find a device, centrals *scan* for peripherals by listening for the peripheral's advertisement packets. After discovering a peripheral, the central can establish a connection to the peripheral. If a connection is established, data can be read from the peripheral using the *Generic Attribute Profile* (GATT). All use cases do not require that a connection is established between a central and a peripheral since some data – such as the local device name and manufacturer specific data – can be retrieved from the advertisement packets. A peripheral is not permitted to have a connection to more than one central at the same time [16], but a central may have connections to several peripherals at once. Peripherals do not advertise during the period a connection exists between the peripheral and a central [16].

The Bluetooth Signal and RSSI

The Bluetooth Low Energy radio operates in the unlicensed 2.4 GHz industrial, scientific and medical (ISM) band [16]. BLE employs frequency hopping to minimize interference from other technologies in the 2.4 GHz band [2], e.g. Wi-Fi. Despite this, interference can have a significant negative effect on the signal strength.

Received Signal Strength Indicator (RSSI) is a measurement of the power in a received radio signal. In Bluetooth Low Energy, RSSI is measured in the logarithmic unit decibel-milliwatts (dBm) and has a range of -127 – $+20$ dBm with an accuracy of $\pm$ 6 dBm [16]. Since it is a measurement of power, a smaller RSSI indicates a weaker signal.
RSSI will be used to determine the smartphones proximity to the door.

A concept related to RSSI is *Measured Power*. Measured Power is the RSSI as measured by the device manufacturer at a distance of one meter and is used when determining the proximity to a device. Devices that support iBeacon (for more info on iBeacon, see the next subsection) broadcast the measured power in their advertising packets. Since an increased RSSI can be the result of decreased distance to the device or increased power to the transmitting antenna, Measured Power is used to determine which of these has occurred.

Beacons and iBeacon

Bluetooth Low Energy *beacons* are small devices that other Bluetooth devices – such as smartphones – use for context-aware applications. The beacons advertise continuously and the smartphones can use the received advertisements to estimate a proximity to the beacon.
There are several third-party beacons available for purchase, such as the *Estimote* – seen in Figure 1 – and *Kontakt* beacons. In this thesis, two Estimote beacons are used.

Revealed alongside iOS 7, *iBeacon* is Apple's trademarked indoor positioning system built on top of Bluetooth Low Energy. Devices that support the iBeacon protocol are called *iBeacons* and advertise a particular Bluetooth Low Energy-payload [10]. If a device supports iBeacon, an iOS application can use Apple's Core Location Objective-C framework to interact with the beacon.
Both the Estimote and Kontakt beacons support iBeacon.

Bluetooth Low Energy on iOS

With the release of iOS 5 in 2012, Apple introduced the Core Bluetooth framework which enables developers to communicate with Bluetooth Low Energy devices [12]. The Core Bluetooth framework is an abstraction over the BLE protocol stack and hides many of the low-level details of the specification, making it easier for developers [5].
There are two key classes in Core Bluetooth: *CBCentralManager* and *CBPeripheralManager*. For an application that needs to act as a central and communicate with peripherals,

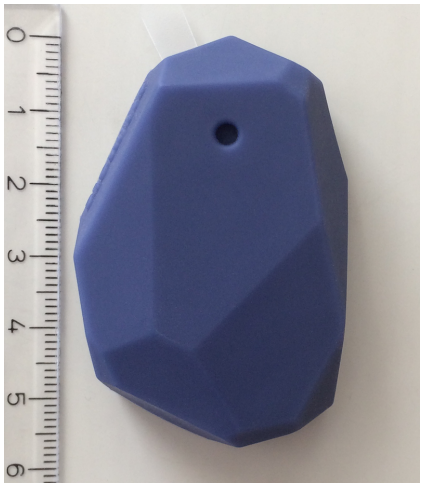

Figure 1. A purple Estimote, as seen from above. The ruler shows centimeters.

the former class shall be used. Applications that function as Bluetooth Low Energy peripherals and serve data to other devices use the latter class. Both classes use the *delegation pattern* that is commonly used in iOS frameworks. As an example, assume an application's `AppController` class wants to be notified when a Bluetooth peripheral is discovered. To accomplish this, the `AppController` creates an instance of `CBCentralManager` and sets itself as the delegate using `CBCentralManager`'s `delegate` property. The `AppController` then tells the `CBCentralManager` to start scanning for peripherals. When the central manager discovers a peripheral, it will notify the `AppController` by calling the `centralManager:didDiscoverPeripheral: advertisementData:RSSI:` instance method on its delegate (i.e. the `AppController`). This method and a few others are specified in the `CBCentralManagerDelegate` protocol.

Application States and Their Effect on Core Bluetooth

An important thing to know is that an iOS application can be in different *app states* [9]. The relevant states for this thesis are as follows:

Active
The application is running in the *foreground* and is receiving events.

Background
An application in the background state is still running and executing code, but is not visible to the user. Applications usually only enter this state briefly on their way to being suspended.

Suspended
The application is in the background but is not executing code. A suspended application might be purged by the system if memory is needed.

This is important to understand since several restrictions regarding the use of Bluetooth apply to applications that run in the background, and the application implemented in this thesis will spend almost all of its time in the background. The restrictions include not being able to scan for peripherals that do not advertise any services, not being able to receive all advertising packets, and limiting the frequency at which the central device scans for peripherals [6]. These restrictions will have a large impact on how the application is implemented, which will be covered in the *Method* section on page 4.

If the application is in a suspended state it is not executing code and cannot respond to events. However, if the application uses Core Bluetooth, the system can briefly transition the application from a suspended state to the background when a Bluetooth Low Energy event is detected. By implementing support for *state preservation* and *restoration*, the system can even re-launch a previously terminated application in response to Bluetooth events [6]. Events include connecting/disconnecting from peripherals and discovering new peripherals through scanning.

Core Location

Apart from Core Bluetooth, Apple supplies the *Core Location* framework which is used for determining a user's position. It can also be used to define geographic regions and monitor when a user crosses the boundaries of those regions [7]. As of iOS 7, Core Location supports region monitoring and ranging using Bluetooth Low Energy devices, which means a region can be defined around a Bluetooth device. Ranging can be used to determine the relative range to nearby Bluetooth devices [13]. However, in order to use Core Location with Bluetooth Low Energy devices, the devices must support *iBeacon*.

RELATED WORK

There has been a lot of research relating to the reliability of using RSSI for indoor localization, most of which has come to less than satisfactory conclusions. Qian Dong and Waltenegus Dargie concluded in a study done in 2012 [17] that the raw RSSI data is "absolutely weak" in determining the distance to a mobile node in an indoor environment. They also found that even after running the RSSI data through algorithms aimed to reduce noise, e.g. a moving average algorithm, the localization errors were too great to ignore.

Another recent study [18] – done at Linköping University in 2013 – showed that the Bluetooth Low Energy signal strength varied greatly over time, even at a constant distance. The authors concluded that this makes it difficult to infer an exact distance to the Bluetooth device based solely on RSSI data.

A similar study was done by Ambili Parameswaran, Mohammad Husain and Shambhu Upadhyaya in 2009 [19]. In the study, the authors performed practical experiments with the aim of proving or disproving that RSSI can be used as an indicator of distance between sensors with a built-in 2.4GHz antenna. The experiments were performed in a close to ideal environment in which the sensors were placed on the same surface, there were no obstacles between the sensors and any electronic equipment that could cause interference was eliminated. Despite the close to ideal environment, the authors came to the conclusion that RSSI cannot be used as a metric for distance measurements in localization algorithms due to

the difficulty in finding a correlation between RSSI and distance.

A third study [15] at the SPaRC Laboratory in Slovenia concluded that RSSI is not a reliable measurement when precise distance evaluation is needed. However, the authors believe that RSSI could possibly be of practical use in cases where the distance evaluation does not have to be very precise.

Despite a lot of research coming to the conclusion that RSSI is not well suited for indoor localization, it might work well for the use case evaluated in this thesis. Precise distance evaluation is not a requirement in estimating how close a user is to the door. However, the large variance in RSSI might be a problem when determining what side of the door a user is located on. This is discussed further under *Discussion*.

METHOD

There are two parts to the thesis: implementing an application for Apple's mobile operating system iOS, and performing measurements on the Bluetooth Low Energy communication. This section will cover how and why a certain implementation was chosen and what measurements were performed. It will also explain how the door can be unlocked from an application and how this was discovered.

Unlocking the Door Programmatically

If the implemented iOS application is unable to unlock the door, the use for the application decreases drastically. As such, the first step was to determine if and how the door could be unlocked programmatically. It was known that the lock was running a web server on the local network; the web server served a web page that was used to configure what times during the day the lock required a passcode, amongst other things. The web page also contained a button for unlocking the door briefly, known as a *pulse unlock*. It was trivial to discover what HTTP request was sent to perform a pulse unlock, and then send the same request from an iOS application. This worked well with the only drawback being that the application cannot know if the door is unlocked or not.

Choosing an Approach for the Implementation

Before deciding on an approach, several alternatives were evaluated based on how well they would perform in an iOS application that would mostly be run in a backgrounded state. The alternatives mainly differ in how Bluetooth is used.
A decision also needed to be made whether to use Core Location or Core Bluetooth.

Core Location or Core Bluetooth?

The first choice made was whether to use Core Location or Core Bluetooth. As Core Location can only be used with Bluetooth devices that support the iBeacon protocol, it was decided that Core Bluetooth would be used to make the implementation more general.

Approach A: Scanning

The first approach used scanning as the main component. As seen in Figure 2, the application would act as a central and continuously scan for advertising packets from peripherals. Whenever an advertising packet was received, an updated signal strength could be retrieved. Based on the retrieved signal strength, the application would decide whether or not to unlock the door.
As scanning is simply the process of listening for packets, this approach had the benefit of working with any number of smartphones scanning simultaneously.
The approach seemed feasible until it was discovered that when running in the background, Core Bluetooth does not allow individual advertising packets to be received. Instead, multiple discoveries of an advertising peripheral are coalesced into a single discovery event [6]. As such, a signal strength would only be retrievable once – the first time a peripheral was discovered. This made the approach infeasible and the approach was discarded.

Figure 2. An illustration of Approach A, showing two advertising Estimotes and a smartphone.

Approach B: Acting as a Peripheral

The second approach moved all the logic from the iOS application to the Estimotes. Instead of the application acting as a central it would act as a peripheral and advertise, as seen in Figure 3. The Estimotes would then listen for the advertising packets from the application and determine whether or not to unlock the door.
An iOS application is allowed to advertise data in the background; however, if all applications that are advertising are in the background, the frequency at which advertising packets are sent may decrease [6]. The minimum frequency is not detailed in the documentation.
This approach was deemed infeasible due to the fact that the Bluetooth devices used – the Estimotes – could not be programmed. If other Bluetooth devices could be used, e.g. an RFduino, this approach would most likely be the most practical.

Figure 3. An illustration of Approach B, showing two Estimotes and an advertising smartphone.

Approach C: Connecting to the Peripherals

The third and chosen approach, seen in Figure 4, was similar to the first approach in that the application would act as a central. Instead of scanning for devices, the application would connect to the Estimotes and read their signal strength using the `readRSSI` instance method defined on `CBPeripheral` in the Core Bluetooth framework.

By exploiting the fact that the Estimotes cancel the connection after ten seconds, the application could be woken up from a backgrounded or suspended state by the system every ten seconds. Upon being woken up, an application has around 10 seconds to complete a task [6]. The application could use this time to re-connect to the Estimote and read the RSSI several times.
There are two disadvantages with this approach: the RSSI can only be read once every second, and a Bluetooth Low Energy peripheral can only be connected to a single central at any one time. This means that if a person inside the building is sufficiently close to the Estimote to keep constantly connecting to it, a person approaching the door from the outside will not be able to unlock the door automatically.
Despite these shortcomings, this approach was deemed the best of the three.

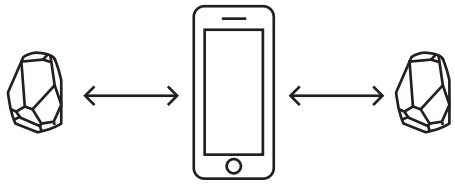

Figure 4. An illustration of Approach C, showing the connections between two Estimotes and a smartphone.

Implementing the Chosen Approach

After an approach was chosen it was implemented in the form of an iOS application; this section will briefly explain the process of implementing the application.
For technical details on the application architecture, please refer to the Results section.

Tools and Programming Language

The application was written in the object-oriented programming language Objective-C using Apple's integrated development environment (IDE) Xcode. It was tested on an iPhone 5S running iOS 7.1.

Application Overview

Before the chosen approach was incorporated into the final application, the chosen approach was tested in a standalone application to verify that it would work. After the approach was verified, work was started on the final application. However, when verifying the chosen approach it was discovered that the user would need a way to choose which Estimotes to use. As such, a solution that used scanning to display a list of Estimotes was developed. When this part of the application was completed, the chosen approach was incorporated into the application and modified to work with the Estimote selection.

Performing Measurements

Several measurements were performed during the thesis to evaluate the Bluetooth technology. This section will describe and motivate all performed measurements.

Signal Strength Variance at a Constant Distance

The signal strength (RSSI) is used to determine the user's proximity to the door and as such it is important to understand how the signal strength behaves. Two sets of measurements were performed to investigate how the signal strength varies over time at a constant distance between the smartphone and Estimote. If the signal strength varies significantly even when the distance between devices does not change, it might be difficult to determine a proximity from the raw RSSI data.
Both sets of measurements were performed in an office environment and used the same Estimote, which had an advertising interval of 200 milliseconds and a transmit power of -12 dBm.

The first set of measurements was performed by placing the smartphone and Estimote one meter apart on a flat surface with no obstacles between the devices. An application on the smartphone listened for all incoming advertising packets from the Estimote during a period of 60 seconds and recorded their signal strength and time of arrival to a comma-separated values (CSV) file. The measurement was repeated five times at an interval of three minutes, and the standard deviation was calculated for each measurement.

The second set of measurements was performed with the same setup as the previous set. However, instead of using the advertising packets to retrieve a signal strength, the application connected to the Estimote and used the `readRSSI` instance method. This was done to determine if the signal strength would be more reliable when a connection was established.

The mean standard deviation was then calculated for each set of measurements.

Energy Usage

As the application is designed to run constantly in the background, it is of great importance that the effect on battery life is minimal. When using an iOS device for development, it is possible to record energy usage for later inspection. This feature was used to record energy usage over a period of eight hours while the implemented application was running. To prevent any other application from affecting the energy usage, the iPhone was rebooted before the measurement was performed. The implemented application was then launched and immediately put into a backgrounded state, and the iPhone was locked. An Estimote was in close proximity and the application continuously connected and read the RSSI from the Estimote during the entire eight hour period. No other Bluetooth Low Energy devices were in the vicinity.
Following the measurement, the recorded data was imported from the device into Apple's performance, analysis and testing tool *Instruments*.

Connection Latency

The connection latency is defined to be the time it takes the application from initiating a connection until a connection has been established to the device. If this takes a long time, it might not be practical to rely on establishing a connection to retrieve a signal strength. By the time the connection is established, the user might already have arrived at the door.
A measurement of the connection latency was performed by writing a small application that connects and disconnects from an Estimote several times. The application

stores the time of the call to `-[CBCentralManager connectPeripheral:options:]` and calculates how much time has passed when the `centralManager:didConnectPeripheral:` delegate method is called. The application then disconnects from the Estimote, waits 60 seconds and then performs the next connection measurement.
The mean connection latency was then calculated from 60 measurements.

Signal Strength When Walking Past Two Estimotes

One problem with proximity-based door locks that this thesis tries to solve is preventing the door from unlocking when a person approaches the door from the inside. The approach taken in this thesis was using two Estimotes – one on either side of the door – to determine what side of the door the user is on. It was hypothesized that if the user is on the inside of the door, the signal strength from the interior Estimote should be greater and vice versa. To determine if this hypothesis was correct, five measurements were performed when the user walked past the door. Each measurement lasted 15 seconds – which was enough time for the user to pass the door – and during this time, an application on the user's iPhone recorded the signal strength of both Estimotes. The data was then analyzed to see if any false positives were present (i.e. the signal strength of the exterior Estimote was greater than that of the interior Estimote, despite the user constantly being inside the office). Figure 5 shows the layout of part of the office, the location of the two Estimotes and the approximate path the user walked.

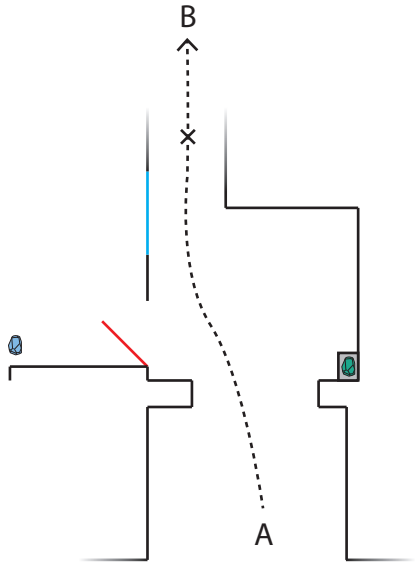

Figure 5. An illustration of the office layout. The user walked from point A to point B while the application recorded the signal strength from both the interior (right) and exterior (left) Estimote. The cross will be explained in the Discussion.

RESULTS

This section details how the iOS application is implemented and presents the results from the performed measurements.

Application Architecture

There are two parts to the application: choosing what Estimotes to use, and using the selected Estimotes to determine when to unlock the door.
When the application is first launched, the user is prompted to choose two Estimotes – one *interior* and one *exterior*. This user interaction is handled by the `PLKEstimoteSelectionTableViewController` class which presents a table view[1] of nearby Estimotes represented by instances of the `PLKBluetoothDevice` class. The table view controller uses an instance of `PLKDeviceScanner` to populate the table view using scanning. By continuously scanning for devices, the RSSI for each Estimote can be kept up-to-date as the user moves around. Since all Estimotes broadcast the same device name (`estimote`), the Estimote with the greatest RSSI is highlighted to help the user differentiate between the Estimotes.

When two Estimotes have been selected, an instance of the `PLKMainViewController` class is created. This class is responsible for observing the RSSI of the two Estimotes and determining whether or not to unlock the door. `PLKMainViewController` uses an instance of `PLKEstimoteConnectionManager` to keep the RSSI of the Estimote model objects updated; this is accomplished through continuously connecting to the Estimote devices and reading their RSSI.
When the RSSI of an Estimote is updated, the view controller determines if the door should be unlocked by comparing the RSSI of the interior and the exterior Estimote. If the RSSI of the exterior Estimote is greater, the user is assumed to be outside the door and the view controller unlocks the door. The unlocking is handled by an instance of `TADoorUnlocker` which communicates with the door lock.

Measurements

This subsection will present the results from all the measurements performed during the thesis.

Signal Strength Variance at a Constant Distance

The results from both sets of measurements show large standard deviations, with the results from measurements performed using `readRSSI` being the largest. Table 1 contains the standard deviation for all five measurements in each set, as well as the mean standard deviation calculated from each set.

Measurement	Using advertising packets	Using `readRSSI`
#1	3.028	5.713
#2	3.247	4.335
#3	4.270	4.530
#4	3.594	4.682
#5	3.731	5.878
Mean	**3.574**	**5.028**

Table 1. Standard deviation for all RSSI measurements at a distance of one meter.

[1] A table view is a user interface element used in iOS applications to display hierarchical lists of items.

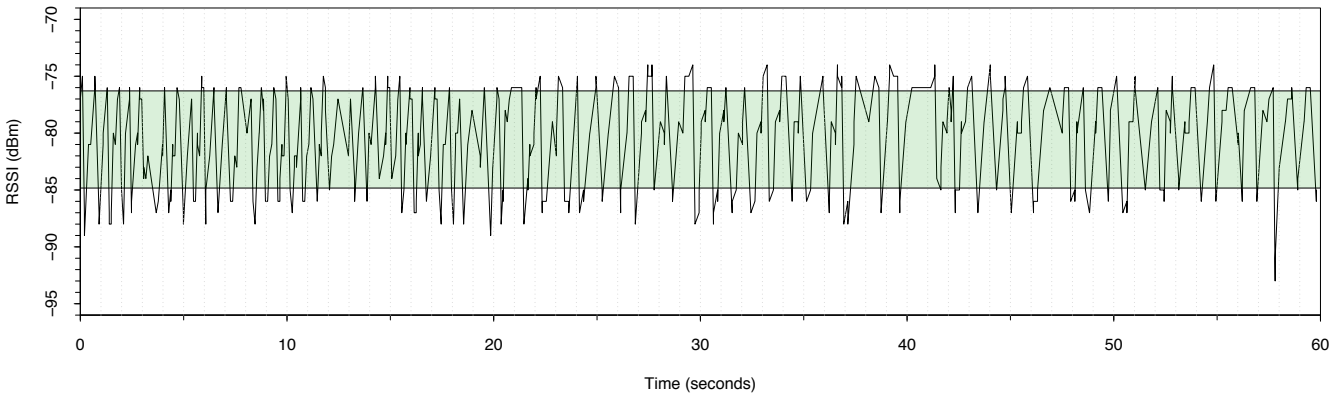

Figure 6. RSSI plotted over time during a period of 60 seconds.

To get a feel for what the RSSI looks like over time, Figure 6 plots the data from the third measurement using advertising packets. The highlighted rectangle is within one standard deviation of the mean RSSI.

Energy Usage

Using the logging built into iOS, energy usage is reported as a relative value on the scale 0 — 20 [8]. The energy usage in the recorded data is consistently reported as $1/20$, with only a few irregular spikes. This suggests that the application has a negligible effect on energy usage.

Connection Latency

The mean connection latency calculated from the first 60 measurement samples was 324 milliseconds.

Signal Strength When Walking Past Two Estimotes

In three out of five measurements, the RSSI of the exterior Estimote was reported to be greater than that of the interior Estimote at least once. One of these measurements is shown as a plot in Figure 7. If a red point is above the black line, the RSSI of the exterior Estimote is greater than the RSSI of the interior Estimote. This can be seen at approximately 10 seconds.

DISCUSSION

In which the method and results are discussed and analyzed.

Results

The following subsections discuss the results from this thesis. This includes the results from all performed measurements, as well as a discussion about the architecture of the implemented application.

Application Architecture

When implementing the final application, care was taken to make the implementation modular and reusable – something that was made easy since Core Bluetooth is a high-level framework. As such, the majority of the classes are small and responsible for only one thing, e.g. unlocking the door. The classes that could be made general without increasing the complexity were made general. A good example of this is the `PLKDeviceScanner` class which is used to find Bluetooth Low Energy devices through scanning. In the implemented application, this class is only used to find Estimotes. However, scanning for devices is a very common thing to do in a Bluetooth application. With this in mind, `PLKDeviceScanner` was not hard coded to scan for devices with a local device name of `"estimote"`. Instead, the programmer supplies a block (a.k.a. anonymous function) that acts as a filter; if the filter returns true for a supplied device, the device is kept.

The fact that it is relatively simple to write good, reusable Bluetooth code using Core Bluetooth is an indication of iOS being a mature platform for developing Bluetooth Low Energy applications.

Signal Strength Variance

Based on the related work and what is written about RSSI in the Bluetooth Core Specification, the results from measuring the signal strength were expected. The Bluetooth specification states that the reported RSSI has an accuracy of ± 6 dBm. When measuring the RSSI at a constant distance of one meter, a mean standard deviation of 3.574 dBm and 5.028 dBm were calculated – well within the mentioned accuracy. An explanation to why the standard deviation is greater for the measurements that used `readRSSI` might be that the measurements were not performed at the exact same location in the office. It is not unlikely that the new location simply had more interference in the form of WiFi. If anything, this shows how sensitive the RSSI is to the environment.

Signal Strength When Walking Past Two Estimotes

The results that showed a higher RSSI being retrieved from the exterior Estimote when walking past the door were unexpected. Even though Bluetooth uses radio waves to communicate and does not require line-of-sight to operate, the signal can be attenuated by obstacles such as walls. With this in mind, the expected result was for the exterior Estimote's signal strength to always be less than that of the interior Es-

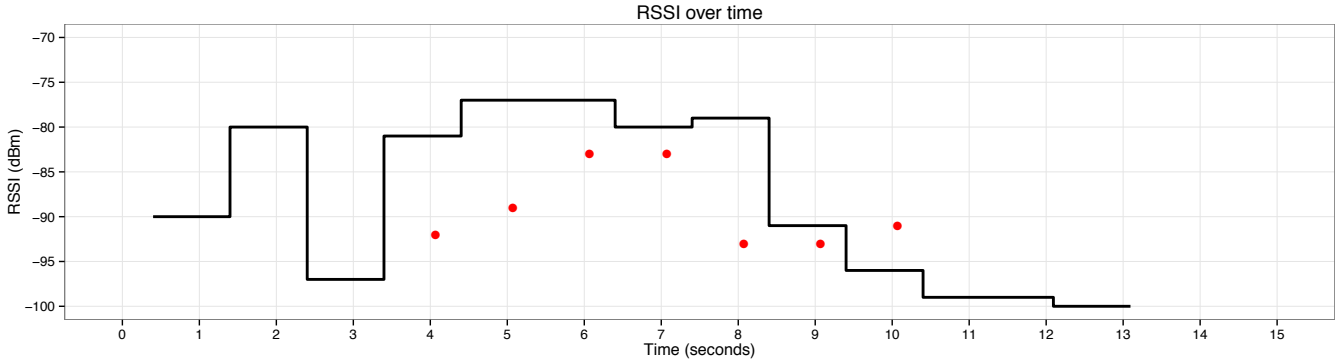

Figure 7. RSSI over time plotted for the interior (black line) and exterior (red points) Estimote.

timote. Despite this, there were false-positives reported in three out of five measurements. After having analyzed the data thoroughly, it was noted that this always happened at the end of the walk close to the cross in Figure 5. A plausible explanation to this phenomenon is that at that point, there is only a window (the blue line segment in Figure 5) between the smartphone and the exterior Estimote, but the corner of a wall between the smartphone and the interior Estimote. The window does not attenuate the Bluetooth signal as much as the wall, which would explain the exterior Estimote having a greater signal strength despite the user always walking past the door on the interior side.
This shows that developing a solution that works in any environment will be difficult. The most reliable solution would most likely be to simply use one Bluetooth device instead of two, and to make sure the device is not discoverable from the inside. This can probably be accomplished either by placing the device a short distance from the door and configuring it to use a very weak signal, or by placing the device closer to the door and preventing the radio signal – through the use of shielding – from reaching inside.

Connection Latency

The connection latency was higher than expected, yet it was deemed sufficiently low to not impact the user experience. According to the Bluetooth SIG, BLE can support connection setup in as low as 3 milliseconds [2]. According to the results, the mean connection latency was 324 milliseconds – a difference of two orders of magnitude! Since the connection attempts were performed 60 seconds apart, this could be explained by iOS putting the Bluetooth radio in a suspended state. That would mean the radio has to be powered on before a connection can be initiated, and it would not come as a surprise if this took several hundred milliseconds.

When measuring the connection latency, the application was in the foreground and the Estimote was within range of the iPhone. Thus, the measurements showed how quickly a connection can be established when the application is running in the foreground and the Estimote is within range. However, the most common scenario is for the application to be in a suspended state and for the Estimote to be out of range. As such, it would be interesting to perform a measurement which would measure the time it takes from the moment the Estimote comes within range until a connection has been established. This would include connection latency, as well as the time it takes for the iPhone to discover the Estimote. Unfortunately, due to the difficulty in performing such a measurement, this kind of measurement was not performed.

Energy Usage

Bluetooth Low Energy was developed with the goal of having ultra-low power consumption. As such, the results from the energy usage measurement were expected.
The irregular spikes in the measurement data are most likely the result of the operating system performing tasks unrelated to the application. Unfortunately, the measurement data could not be exported from Instruments in any way, which made it difficult to perform any real analysis of the data.

Method

A discussion about how the chosen approach was implemented and how the measurements were performed will follow in the coming subsections. A short paragraph on source criticism is also included.

Implementation

When implementing the final application and evaluating the three different approaches, a form of learning-by-doing was applied. The majority of what the author knows about the Bluetooth Low Energy technology was gained from hands-on experience throughout working on the thesis. This method worked well as there was a lot of practical work. However, it did have some drawbacks, and if the thesis work was to be redone there would be some changes to the method. The most significant drawback was that by not doing most of the theoretical research about Bluetooth Low Energy and Apple's frameworks before the practical work, the risk of coming to faulty conclusions increased. One faulty conclusion made was that it was not possible to scan for Bluetooth devices when the application was in a backgrounded state, even if the devices advertise services. As it turns out, this was possible but the application was restricted to only being notified once when a device is found – as detailed in the documentation. In this case, the end result was the same for the implemented application, but it resulted in the author having an incorrect understanding of the Core Bluetooth framework. By doing the

research beforehand – and being thorough – a lot of wasted time can be avoided.

Measurements
All measurements have been performed with replicability and reliability in mind; each one – except for the battery usage measurement – has been performed multiple times and statistical methods have been used on the results to increase reliability. Despite this, it will be difficult to perform the measurements again and get exactly the same results. The Bluetooth signal is relatively sensitive to interference, and all measurements relating to signal characteristics were performed in an office environment without making any effort to eliminate interference. As such, the measurement result values (such as average RSSI) will most likely differ. That being said, any difference in results will probably not be great enough to alter any conclusions.
The signal strength measurement using two Estimotes will be very difficult to reproduce as the results are tightly coupled to the location at which the measurements were performed. This does not make the measurements less valid, however; there is definite value in showing how Bluetooth behaves in a location such as the one were the measurements were performed.

The only measurement that should have been done differently is the energy usage measurement. Energy usage was only measured during one eight hour period, which reduces the reliability of the results. Even if the results were constant throughout the measurement period, it would be a good idea to verify the results with a second measurement.

Source Criticism
Care has been taken to only cite reputable and credible sources, such as the official Bluetooth SIG website and papers published at well-known conferences. Verifying all information is very difficult, but facts presented in this thesis are based solely on information gathered from more than one source.

The Work in a Larger Context
It is important for users to be aware of and understand the privacy aspect of automation as technologies such as BLE become more widespread. Automating common tasks – such as unlocking doors – can have an impact on user privacy. Using technology to automate tasks makes it easier to collect usage data without the user being notified, which can be seen as a form of privacy invasion. In the case of automatic door locks, information regarding who has unlocked the door and at what time it was unlocked can be stored; this data can then be used to approximate a persons location over time. When using Bluetooth to approximate distance, an application can store the distance data and transmit it to another party.
A user that is aware of possible scenarios can make more informed decisions regarding the use of automation technology.

CONCLUSIONS
An iOS application for unlocking a door based on a users proximity has been implemented, and several measurements have been performed to evaluate Bluetooth Low Energy in the context of automatic door locks. From this, a lot of knowledge has been gained pertaining to how suitable Bluetooth Low Energy is for automatic door locks, as well as what restrictions and possibilities exist on the iOS platform for developing Bluetooth Low Energy applications. As such, it can be concluded that the purpose of this thesis has been fulfilled.

Despite the restrictions that apply to backgrounded applications, Apple's mobile operating system iOS is a mature platform for developing Bluetooth Low Energy applications. With the updates to Core Bluetooth in iOS 7 allowing applications to be re-launched by the system even after being terminated, it is possible to write applications that the user can launch once and then forget. This is especially suitable for applications that automate tasks based on Bluetooth activity. Even though Core Bluetooth works very well for most tasks, it can be too high-level in some cases, e.g. by not exposing the raw advertising packets.

The following subsections will present the conclusions for the research questions discussed in the Introduction.

Is it possible to prevent the door from unlocking when a person approaches the door from the inside?
Measurements show that the signal strength varies too greatly over time and is too easily affected by the environment to correctly differentiate between what side of the door the user is located on using the approach discussed in this thesis. However, as discussed in the previous chapter, it should be possible to solve this problem by only using a single peripheral on the exterior side of the door.

Does the implemented solution affect the smartphone's battery life negatively in a significant way?
By measuring the energy usage of the implemented application during a period of eight hours, it was concluded that the application's affect on the battery life was negligible.

Is the Bluetooth Low Energy communication latency sufficiently small to make the technology usable in practice?
The average connection latency discovered through measurements (324 milliseconds) was two orders of magnitude higher than expected. Despite the large difference in the expected latency and the actual latency, the connection latency was concluded to be sufficiently small to not make the technology unusable.

Is Bluetooth Low Energy a Suitable Technology for Proximity-based Automatic Door Locks?
Bluetooth Low Energy provides ultra-low power consumption, sufficiently low latencies in establishing connections and – with a bit more time and work – the possibility of only unlocking the door when a user approaches the door from the outside. Based on this, the conclusion is that Bluetooth Low Energy is a suitable technology for proximity-based door locks. The difficulty in determining a user's location relative to the door is not a big problem since it is not necessarily a requirement, and it should be relatively simple to solve, as previously discussed.

Future Work

The most important aspect to improve upon in the implemented solution is the fact that only one application can be connected to the Bluetooth device at once. For the solution to be truly useful, there cannot be a limit on how many people can be in close proximity to the lock simultaneously. This will most likely require that an entirely new solution be developed as the limit is a result of the connection-oriented solution implemented in this thesis.

REFERENCES

1. A Look at the Basics of Bluetooth Technology. **http://www.bluetooth.com/Pages/Basics.aspx**.
2. About Bluetooth Low Energy Technology. **http://www.bluetooth.com/Pages/low-energy-tech-info.aspx**.
3. Bluetooth Fast Facts. **http://www.bluetooth.com/Pages/Fast-Facts.aspx**.
4. Bluetooth Smart Technology: Powering the Internet of Things. **http://www.bluetooth.com/pages/bluetooth-smart.aspx**.
5. Core Bluetooth Programming Guide: About Core Bluetooth. **https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html**.
6. Core Bluetooth Programming Guide: Core Bluetooth Background Processing for iOS Apps. **https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothBackgroundProcessingForIOSApps/PerformingTasksWhileYourAppIsInTheBackground.html#//apple_ref/doc/uid/TP40013257-CH7-SW1**.
7. Core Location Framework Reference. **https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CoreLocation_Framework/_index.html**.
8. Instruments User Reference: Energy Usage Instrument. **https://developer.apple.com/library/ios/documentation/AnalysisTools/Reference/Instruments_User_Reference/EnergyUsageInstrument/EnergyUsageInstrument.html**.
9. iOS App Programming Guide: App States and Multitasking. **https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html**.
10. Location and Maps Programming Guide: Region Monitoring and iBeacon. **https://developer.apple.com/library/ios/documentation/userexperience/conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html**.
11. Skyrocketing demand for Bluetooth appcessories for latest phones. **http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx**.
12. What's New in iOS: iOS 5.0. **https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS5.html**.
13. What's New in iOS: iOS 7.0 - Core Location Framework. **https://developer.apple.com/library/ios/releasenotes/General/WhatsNewIniOS/Articles/iOS7.html#//apple_ref/doc/uid/TP40013162-SW28**.
14. Wibree Forum Merges with Bluetooth SIG. **http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=51**, June 2007.
15. Benkič, K., Malajner, M., Planinšič, P., and Ž. Čučej. Using RSSI value for distance estimation in wireless sensor networks based on ZigBee. In *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*, IEEE (2008), 303–306.
16. Bluetooth SIG. *Bluetooth Specification version 4.0*, June 2010.
17. Dong, Q., and Dargie, W. Evaluation of the reliability of RSSI for indoor localization. In *Wireless Communications in Unusual and Confined Areas (ICWCUCA), 2012 International Conference on*, IEEE (2012), 1–6.
18. Englund, A., and Suther, M. Bluetooth Low Energy som trådlös standard för hemautomation, Bachelor's Thesis, Linköpings universitet, 2013.
19. Parameswaran, A. T., Husain, M. I., and Upadhyaya, S. Is RSSI a reliable parameter in sensor localization algorithms: An experimental study. In *Field Failure Data Analysis Workshop (F2DA09)* (2009).