Evaluation of Bluetooth 4.0 and its aptitude for a reliable indoor positioning system on today's smartphones

Bachelorarbeit

Im Studiengang Medieninformatik

Vorgelegt von

Marius Heil

Matrikel-Nr: 21591

Am 11.03.2012

An der Hochschule der Medien Stuttgart

Erstprüfer: Prof. Dr. Ansgar Gerlicher

Zweitprüfer: Prof. Dr. Joachim Charzinski

Eidesstattliche Versicherung

Name: Heil	**Vorname:** Marius
Matrikel-Nr.: 21591	**Studiengang:** Medieninformatik Bachelor

Hiermit versichere ich, Marius Heil, an Eides statt, dass ich die vorliegende Bachelorarbeit mit dem Titel “Evaluation of Bluetooth 4.0 and its aptitude for a reliable indoor positioning system on today’s smartphones” selbständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der eidesstattlichen Versicherung und prüfungsrechtlichen Folgen (§ 26 Abs. 2 Bachelor-SPO bzw. § 19 Abs. 2 Master-SPO der Hochschule der Medien Stuttgart) sowie die strafrechtlichen Folgen (siehe unten) einer unrichtigen oder unvollständigen eidesstattlichen Versicherung zur Kenntnis genommen.

Auszug aus dem Strafgesetzbuch (StGB)

§ 156 StGB Falsche Versicherung an Eides Statt

Wer von einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Stuttgart, den 11.03.2013

Ort, Datum

Unterschrift

Note of Thanks

"I hereby want to express my gratitude to Bluegiga for they have provided me with the necessary hardware to finish my thesis when all their distributors were out of stock."

Kurzfassung

Diese Arbeit stellt die Grundlagen von Bluetooth Low Energy, im Hinblick auf die Implementierung eines Positionierungssystems für Innenräume, dar. Hierzu werden die Hard- und Softwareseitigen Möglichkeiten aufgezeigt und die nötigen Grundlagen erläutert.

Bluetooth Low Energy ist seit Version 4.0 ein fester Bestandteil der Bluetooth Spezifikation. Es bietet einige grundlegenden Vorteile gegenüber klassischem Bluetooth. Einige wichtige Eigenschaften sind Energieeffizienz, günstige Hardware und eine offene Spezifikation. Momentan besitzt Bluetooth 4.0 noch eine geringe Verbreitung und mehrere Konkurrenztechnologien mit denen es in direktem Wettbewerb steht. Diese werden ebenfalls kurz präsentiert. Gesucht wird eine Lösung die billig, akkurat und einfach zu Warten ist. Insgesamt sollte die Lösung klare Vorteile gegenüber den bestehenden Systemen bieten.

Nach einer kurzen Einleitung in der Motivation und Ziele dargelegt werden, wird der Bluetooth 4.0 Standard erläutert. Im weiteren Verlauf werden einige Konkurrenztechnologien im direkten Vergleich vorgestellt. Ein Überblick über die derzeitig verfügbaren Smartphones soll dann Klarheit über die derzeitige Verbreitung, die Möglichkeiten und Probleme bei der Umsetzung bieten. Ein besonderer Fokus liegt hierbei auf dem Betriebssystem iOS, welches auch für die Implementierung eines Prototyps verwendet wurde. Die anderen notwendigen Komponenten für das Positionierungssystem werden ebenfalls vorgestellt. Schlussendlich wird die Frage beantwortet, ob sich Bluetooth 4.0 wie eingangs angenommen für die Aufgabe eignet oder nicht.

Abstract

This thesis provides an in depth look at Bluetooth 4.0 and more specifically at its new feature set called Bluetooth Low Energy. It evaluates the possibilities on hardware and software side with regard to its suitability as a cost efficient indoor positioning system.

The new low energy protocol has many benefits over Bluetooth 4.0 which are the foundation stones of my work. Energy efficiency, low hardware cost and the open nature of the underlying standard. One shortcoming is that at this stage there has not been a wide market acceptance. If the Bluetooth Low Energy (BLE) Standard does not succeed, other technologies have to be used. Some of these will be examined in a short overview.

The sought-after solution should have three key characteristics. It should be cost efficient, accurate and easy to maintain. Overall it should have clear advantages over the systems that are available today.

After an introduction that points out motivation and goals, the second and third chapter will start out by explaining the Bluetooth Low Energy standard and provide a short overview of competing products. Next, the availability of Bluetooth 4.0 in today’s smartphones will be evaluated with a focus on iOS. Implementation possibilities and caveats will be presented as a result of the measurements and tests that have been made along with a presentation of the other components that are relevant for the system. Shortly thereafter the software that has been developed for this purpose will be presented. A conclusion will then be drawn if Bluetooth Low Energy is indeed a viable solution for implementing an indoor positioning system.

Table of Contents

1 Introduction

1.1 Preface

Technology is a never ending race. It is a race defined by product quality, marketing and market penetration. It is not always the superior product that takes the lead. There have been high quality products over the past that got scrapped because of poor marketing or lobbyism. This is not a race that is won without taking risks. If you decide to invest too early, you risk that the technology is never established. If you wait for too long you will miss the initial hype and in most cases your entire chance.

Bluetooth 4.0 is by no means a recent invention. It is a standard that has been published by Nokia and its partners back in 2006 under the initial name WiBree. But only now, beginning around 2012 some manufacturers are starting to deploy compatible chipsets in consumer technology. The possibilities are virtually infinite. This technology, if widely available, provides opportunities that have not been possible before.

1.2 Motivation

There have been various attempts at building a cheap and versatile indoor navigation system. A myriad of different technologies have been employed. Yet none of these have had wide success. One method that has seen much research over the last years has been Indoor navigation based on Wi-Fi access points. But installation costs and lack of precision have impeded success[1]. There have been others based on magnetic field characteristics[2], sound waves[3], employing optical transmitters[4] and dead reckoning[5] to just name a few. Some of these systems use a combination of two or more features to improve accuracy. All of these systems have standout features and drawbacks. The most important aspects are: installation cost, accuracy, compatibility with today's smartphones, being unobtrusive to the user and additional capabilities. None of the above technologies work perfect under all the given requirements which is why this work evaluates the use of Bluetooth Low Energy as a possible candidate.

Up to this point, information on Bluetooth 4.0 is sparse and the amount of published articles decreases by a lot when indoor positioning is taken into account. Basic research on signal propagation and the features of the protocol itself has to be done to lay the cornerstones.

[1] Real world performance is discussed here http://www.mobile-zeitgeist.com/2012/02/21/indoor-navigation-zwischen-wunsch-und-wirklichkeit/ and here http://www.economist.com/news/technology-quarterly/21567197-navigation-technology-using-satellites-determine-your-position-only-works

[2] http://www.indooratlas.com/

[3] http://www.ics.uci.edu/~lopes/documents/ccnc%2005/ccnc05Beep.pdf

[4] http://www.bytelight.com/

[5] http://www.motorola.com/sites/motodev/library/indoor_location_manager.html

1.3 Goals

The primary goal of this work is to analyse the opportunities that Bluetooth 4.0 provides for indoor positioning systems. Analyse if indoor positioning is feasible with current technology, determine the accuracy that can be achieved and point out target markets where the system is useful.

The final system will consist of many Bluetooth Low Energy devices that have to be installed in the desired area. I will refer to them as nodes or beacons. The user will use a smartphone that displays a map of the area with his position.

1.4 Structure

In order to evaluate the possibilities, it is critical to understand the Bluetooth Low Energy standard to a certain extent. This is what Chapter 2 focuses on. The following chapter then provides a comparison with alternative near field radio technologies to determine if Bluetooth Low Energy is the best technology for the portrayed needs. After a short survey on the available devices that are needed for the implementation, an actual prototype implementation called FindMilk is outlined with a focus on a variety of measurements and observations that will help to tailor an algorithm for position determination. Finally a conclusion is drawn if the initial choice of Bluetooth Low Energy as the protocol of choice has been valid.

1.5 Electronic Sources and Statements

In computer science, it is often necessary to reference electronic sources because press releases and statements are frequently only available on webpages. This has been the case for many the sources mainly because Bluetooth 4.0 is still a very new technology. As the internet constantly changes over time, it is a common issue that content changes or ceases to exist. Hence, all electronic resources and statements presented in this work have been verified as of the 11.03.2013. Resources and Statements that are not supplemented with a date are all considered to be valid at this point in time.

2 Bluetooth Low Energy 101

When Nokia had developed a new low cost and low energy technology coined WiBree more than 6 years ago, it did not make a splash in the industry. Around 2007, it was decided to integrate it into the already existing Bluetooth Standard[6] which already possessed wide market adoption and thus the potential to make the new technology available to a wide market. Since then, its name has been changed to Bluetooth Low Energy. The new Bluetooth specification is called Bluetooth 4.0 and consists of all the features found in Bluetooth 3.0+HS[7] and the new low energy protocol. In order to distinguish between those two distinct feature sets, they will be referred to as Bluetooth Classic and Bluetooth Low Energy or simply BLE. The consumers have come to know the new low energy standard under the terms Bluetooth Smart and Bluetooth Smart Ready. In technological terms they are called Single- and the latter Dual Mode Chipsets. While Single Mode Devices only allow connections with the new low energy standard, Dual Mode Devices have the additional benefit of being compatible with Bluetooth 3.0[8].

As both technologies – Bluetooth Classic and Bluetooth Low Energy - make use of the same frequency range, it was possible to share core components like the antenna and parts of the Bluetooth 3.0 stack, thus further reducing implementation costs. Many Bluetooth chipsets available today are Dual Mode chipsets because the advantages justify the low additional costs.

The newest generation of Bluetooth is still in its infancy. Good documentation is still rare. Despite the completion of the standard in 2010[9], there is just one published book available by now which is called the “Bluetooth Low Energy Developers Handbook” by Robin Heydon. He was one of the persons behind the development of WiBree. There are many resources available on the web but most of them do not go into great detail about technical and low level functionality. As Bluetooth is an open standard, the specification is publicly available from the Bluetooth SIG – the special interest group that oversees the standard. It is however very extensive with its 2300 pages and not so much of a pleasant read. It is more intended as a reference book and not as a primer for Bluetooth. But with little information available elsewhere, it was used as a primary source to investigate Bluetooth 4.0.

2.1 Features

What is special about Bluetooth Low Energy? As the name implies, it uses a lot less power than its big brother. But not only does its continuous current draw stay under 1µA but the peak current is not allowed to exceed 15mA according to the specification. This enables a new use case because it

[6] http://www.electronicsweekly.com/Articles/12/06/2007/41582/Wibree-becomes-ULP-Bluetooth.htm
[7] Sometimes Bluetooth 3.0+HS is also referred to as Bluetooth 3.1. It combines the Bluetooth 3.0 Standard with the added benefits of high speed data transfer by handing it off to other protocols such as Wi-Fi.
[8] http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices-List.aspx
[9] https://www.bluetooth.org/Technical/Specifications/adopted.htm

can now be used with traditional coin cell batteries that don’t allow for high peak currents. And with its low energy demand, it can operate for around 1-5 years on a single CR2032 battery depending on the use case. This would also allow usage of photovoltaic cells to allow for a low maintenance cycle.

As stated earlier, the BLE architecture is able to share some key components with Bluetooth Classic in dual mode setups. This allows Single Mode chipset prices as low as 2 US$ if bought in quantities of thousands[10]. The additional cost for a Dual Mode over a Single Mode Chipset is no more than 20% (1). Shared components include the antenna, L2CAP and HCI implementation. The latter two will be explained in Section 2.3.3 along with the other protocols that make up the stack. The antenna can be shared because both Bluetooth variants transmit and receive within the same frequency band.

One of the improvements for developers over Bluetooth Classic is the open protocol architecture that enables programmers to develop custom protocols without the need for certification. A developer can write his own service that identifies itself with a so call UUID. A UUID is a 128bit universally unique identifier standard that is usually represented in hexadecimal notation and is commonly used in software construction. This id is user generated and is considered to be sufficiently unique. There are online services that allow the creation of UUIDs[11]. They use a combination of hardware mac address, timestamp and a unique clock id to seed a random generator and create the identifier.

There are two types of services a BLE device can offer. These are user defined services and services that are standardized by the Bluetooth SIG. Only the latter can use a 16-bit instead of a 128-bit identifier. Every service is built on top of the ATT and GATT protocol that are illustrated in Section 2.3.3. These protocols define a simple architecture where values can be read and written using value access identifiers.

Inbuilt security features offer Link Layer encryption with AES. Bluetooth LE further uses a concept of public and random UUIDs to protect devices from being tracked. A random UUID might change every 15 minutes while the public UUID can be used to identify a device over a long time. The public id can only be resolved by connected devices.

Bluetooth LE uses the unlicensed ISM 2.4 GHz band which is available internationally. With the restrictions that apply, it can provide a range of up to 300 m while the most common range is around 100m. Transmission characteristics will be detailed in Section 2.3.1.

Bluetooth only allows for a star topology in connected scenarios while it also features an unconnected broadcast mode. The need for complex topologies is fulfilled by its very fast link establishment in under 3 milliseconds (2). In a connection there are two different roles a device can

[10] http://www.ti.com/product/cc2540
[11] http://www.famkruithof.net/uuid/uuidgen

occupy in a connection. It can be either a Peripheral or a Central. The Peripheral usually represents a device that provides data to a Central. This is the case with most Bluetooth Smart devices that face the role of being devices that supply data while having a restricted energy supply. They can either broadcast this data for everybody or they can be in a connection with one Central at maximum. Meanwhile, the Central consumes this data as a device with more processing power and energy resources. A Central can be connected to multiple Peripherals at once. Devices that broadcast advertising packets are called Advertiser while scanning devices are Observers.

2.2 Typical Application Fields

There are a myriad of use cases. The medical sector is one that is more and more pushing in the consumer market with smartphone enabled devices like heart rate monitors and running shoes equipped with electronic pace detection. This is where Bluetooth is in a direct competition with Ant+, a commercially successful protocol that will be introduced in Section 3.1. There are temperature sensors equipped with Bluetooth LE, remote controllable light bulbs and the home automation sector in general. Smartphone accessories, animal tagging, consumer tracking and much more application fields are waiting to exploit the new possibilities. Accessories like mice and keyboards that have since relied on classic Bluetooth can now be outfitted with Bluetooth Low Energy to further extend their battery lives.

New projects come to live every day and the online portal Kickstarter has already provided many of those projects with the necessary funding. For example, the Arduino BLE shield that can be used to equip an Arduino controller with Bluetooth Low Energy capabilities[12]. Or “Find My Car Smarter” that helps to remember where you parked your car by saving the GPS position as soon as the smartphone loses connectivity with the BLE module inside the car[13].

[12] http://www.kickstarter.com/projects/rowdyrobot/arduino-ble-shield-connecting-the-ios-and-the-ardu
[13] http://www.kickstarter.com/projects/1628273204/find-my-car-smart

2.3 Protocol Stack

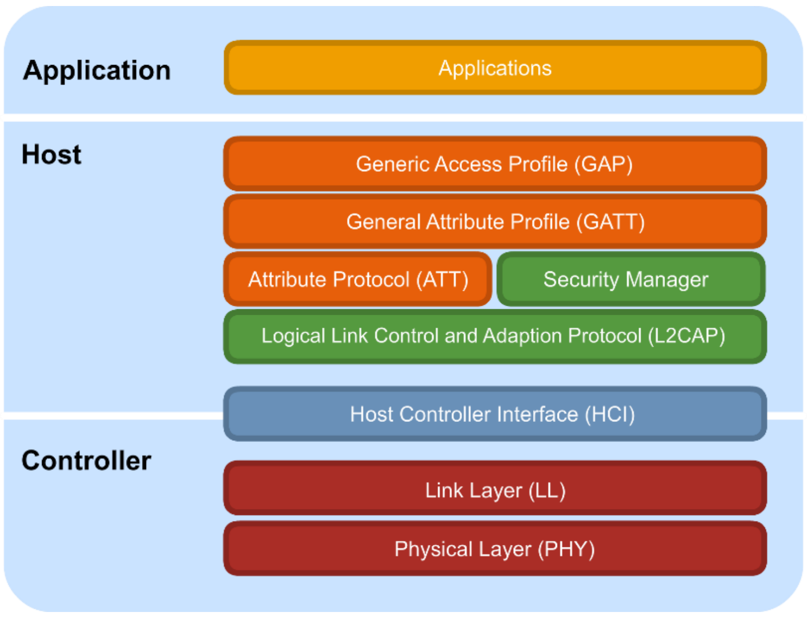

Application
Applications
Host
Generic Access Profile (GAP)
General Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager
Logical Link Control and Adaption Protocol (L2CAP)
Host Controller Interface (HCI)
Controller
Link Layer (LL)
Physical Layer (PHY)

Figure 2-A: Bluetooth Low Energy Stack

Above is a representation of the Bluetooth Low Energy stack. The stack is separated in three parts which allows for split-chip architectures. The BLED112 module used in Chapter 6 is a completely programmable system on a chip solution that covers the stack as a whole. It is however possible to use it as a USB dongle connected to a PC. In that scenario the application layer is served by the desktop computer. The host can communicate with the controller via the Host Controller Interface (HCI) over any of the interfaces that are defined in the Bluetooth 4.0 standard which are USB, UART or SDIO[14]. This opens the possibility to run the host on another chip or on the same without redesigning the system thus keeping it manufacturer independent.

When using a Dual Mode solution with Bluetooth Classic and Bluetooth Low energy in one chip, the physical layer and the Logical Link Control and Adaption Protocol or short L2CAP are shared between the two implementations.

Extensive research has shown that most of the above pictured stack is not necessary to implement an indoor navigation system. The necessary advertising packets can be built byte by byte as explained in Chapter 6. Therefore, the following chapters focus on the important parts. These are the Physical Layer and the Link Layer. The Link Layer is one of the most complex parts that make up the Bluetooth LE stack. It was however a crucial part of this work to examine the physical and link layer in order to gain a better understanding of the data that is transmitted and how it is transmitted.

[14] While USB is probably familiar to the reader, the latter two are more specific. The Universal Asynchronous Receiver/Transmitter (UART) and (Secure Digital Input Output) SDIO interfaces are mainly used as inter-chip communication technologies and provide a much better power margin compared to the more complex USB interface.

2.3.1 Physical Layer

The 2.4 GHz ISM band is an unlicensed band for industrial, scientific and medical purposes and is one of the few bands that are available internationally. Therefore it is shared with many other wireless technologies such as Wi-Fi, Ant+, ZigBee, classic Bluetooth and many more. With Wi-Fi being the most prominent representative, Bluetooth Low Energy needs a way to avoid radio interference with Wi-Fi transmitting devices virtually anywhere. BLE uses 2 MHz wide bands - instead of the 1 MHz band ranges that Bluetooth Classic uses - to achieve a better resistance against interference while also allowing the use of cheaper hardware. There is a total of 37 data channels and 3 advertising channels (labelled 37, 38, and 39). Bandwidth is sacrificed by using a wider band and renouncing on modulation in order to achieve higher robustness and simplicity. Furthermore, the three advertising channels have been laid out in between the most common used Wireless LAN channels, spread as far as possible. The advertising channels are those channels where all devices have to meet at a certain point to establish a connection. That means they are a critical part of the infrastructure. Interference on all three of these channels will block any device from receiving advertisements and successfully establishing a connection. All channels can be seen in the following diagram along with the mentioned Wi-Fi channels.

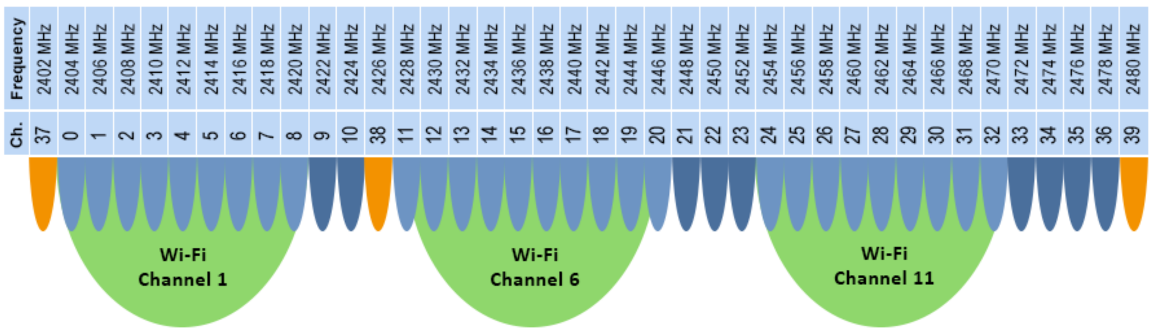


Figure 2-B: Channel usage of Bluetooth Low Energy

Gaussian Frequency Shift Keying - or short GFSK – is used to broadcast data. This means smoothing out the crossings between digital ones and zeros. By smoothing the frequency changes with a Gaussian filter, it is possible to generate little noise and keep the impact on neighbouring frequencies to a minimum. The symbol rate at which Bluetooth LE transmits amounts to 1 Mbps. This is a rather high frequency and has been chosen by design. A higher symbol rate allows for faster transmission of data and therefore reduces the time that the transmitter has to stay on which on the other hand improves battery life.

In contrast to WLAN, Bluetooth LE does not employ any sophisticated algorithms for collision detection. It enables time and frequency multiplexing based on a random hopping scheme determined by a list of blacklisted channels which the connection initiator can optionally provide. This is called adaptive frequency hopping (ADFH). No ADFH is used on the advertising channels.

Advertising devices are advised to send their packets on all of the three designated channels in short succession to achieve the best detection probability.

2.3.2 Link Layer

As mentioned earlier, the link layer is one of the most complex layers of the Bluetooth Low Energy stack. It is responsible for all data handling through the physical channel and is the final layer before information is just considered to be ones and zeros. It is responsible to process and prepare the data that is then modulated and sent over the air as positive and negative frequency deviations. In order to prepare the data for its transmission it performs a process referred to as whitening. It is an elaborate process where the data is modified using a pseudo random generator to clean it from successive sequences of either ones or zeros. Receiving sequences of more than six similar symbols in a row will cause the receiver to lose frequency lock and prevents proper transmission (3). The receiver will then perform a de-whitening with the known seed to reverse the process.

2.3.2.1 States

There are five possible states the link layer can occupy. These are shown below with all possible transitions of the state machine.

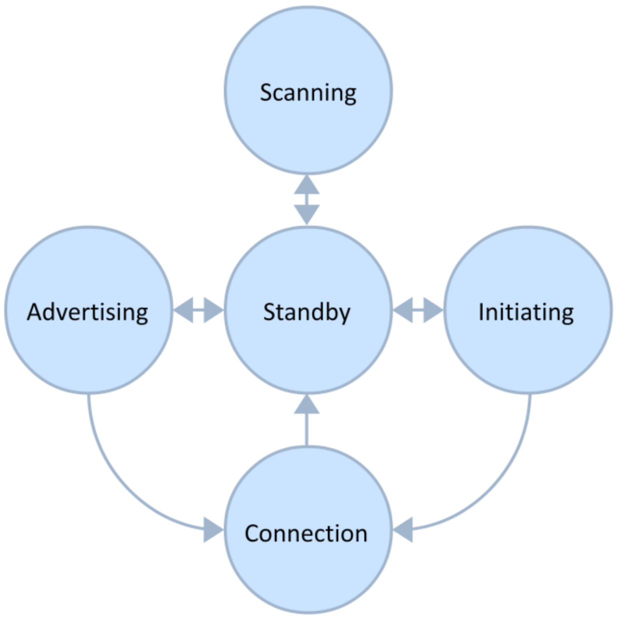


Figure 2-C: Link Layer states

The standby state represents an idle state where every device resides after it has been turned on. Higher layers can then decide to change the link layer state.

When a device is in the advertising state, it is periodically transmitting available data to every other device in scanning mode. An advertising device sequentially broadcasts data on a selected number of advertising channels. All three advertising channels should be used for best results as a channel might be jammed and adaptive frequency hopping is not used on advertising channels. After each broadcast on a channel, it has to listen for 150 µs to listen for an answer before it moves on to the next channel. After the advertising packet has been broadcasted on each channel it waits for the preconfigured advertising interval plus a random time between 0 and 10 ms before it restarts its cycle. The random delay between each cycle reduces the possibility of recurring collisions with other advertisers. The following graphic pictures an advertising event.

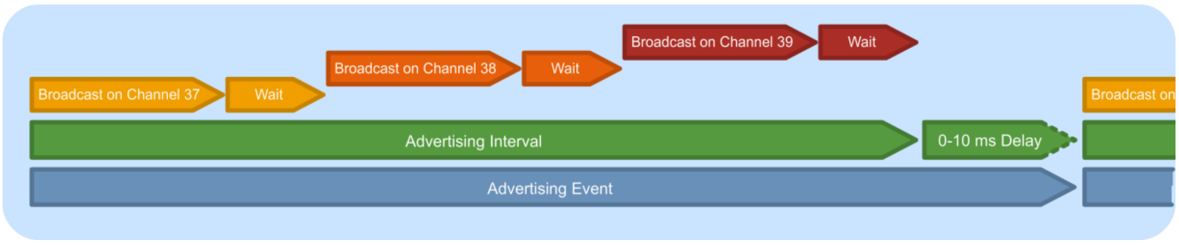


Figure 2-D: Advertising event

A scanning device uses a predefined scan window and scan interval. Similar to the advertising procedure it listens on each of the advertising channels for the time of the scan window. When it is done scanning on all channels it is interested in, it waits for the amount of time set in the scan interval before starting another scan. There are two different scan modes, active and passive. When active scanning is used, the device sends a scan request to every scannable device that it discovers. This packet is sent on the same channel as the advertising packet was received and immediately after it received the packet. It should then continue to listen on this channel as the advertising device will send a scan response. Passive scanning on the other hand does not make use of the scanning device’s transmitter. After a connectable device has been discovered, the device may go into the initiating phase and send a connection initiating packet.

While initiating a connection, there are several parameters that can be configured. In this phase, the device that initiated the connection (master) can send a channel blacklist and set up the hopping algorithm that will then be used during the connection. The device that was asked to set up the connection (slave) can then send an acknowledgement. Connection interval and slave latency are set by the master in the connection request packet. While the connection interval defines the interval at which subsequent data transfer will take place, the slave latency allows the slave to skip some connection intervals where it can save power if it does not have data to send. This enables very low duty cycles and saves power resources. The slave latency has to be chosen according to the connection initiating packet that is sent by the master, the slave may however define a Peripheral Preferred Connection Parameters GAP characteristic.

2.3.2.2 Link Layer Packet Structure

The link layer has only one packet format that is used for both advertising and data channel packets. It is shown in the next diagram.

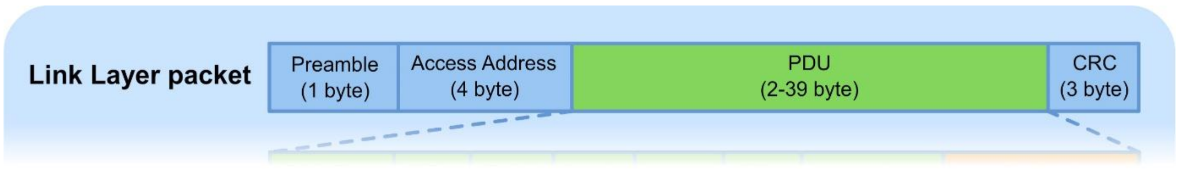


Figure 2-E: Link Layer packet structure

The preamble is chosen based on the first bit of the access address and may either be 01010101 or 10101010. The alternating bit sequence synchronizes both transmitter and receiver and is additionally used for automatic gain control. This is needed to initiate the amplifier and is the first step to calculate the (received signal strength index) RSSI. The RSSI is a measure that indicates the strength of the received signal in dBm and is one of the most important characteristics for indoor navigation.

The access address for advertising packets has been defined as 0x8E89BED6 while it is generated randomly for data channel packets. It has to be compliant to certain rules though. The access address is used to link individual packets to their corresponding connection.

A packet data unit or short PDU represents an amount of data that is different between advertising and data channel packets. A closer look at an advertising PDU is taken in the following section.

A 24 bit CRC finalizes the link layer packet structure and protects its integrity. The length of the cyclic redundancy checksum has been bumped up from the 16 bits used in classic Bluetooth because of observed problems. It can now protect against any odd bit errors and against all subsequent errors up to 5 bytes (3).

2.3.2.2.1 Advertising PDU

An Advertisement PDU consists of a number of header fields and a payload.

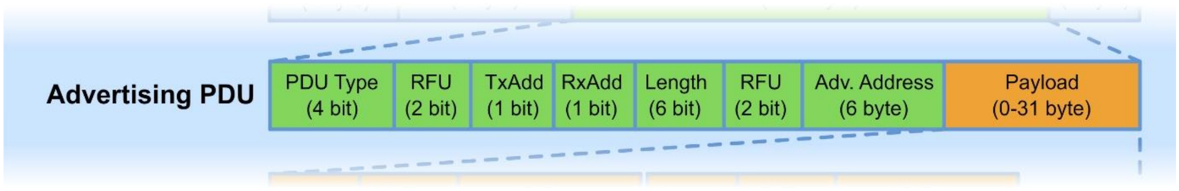


Figure 2-F: Advertising PDU

The Advertisement PDU type can be one of the following:

- **ADV_IND:** Observers are allowed to send scan requests and connect to the device.
- **ADV_DIRECT_IND:** Only the addressed device is allowed to connect.
- **ADV_NONCONNECT_IND:** No connection or scan requests are allowed.
- **ADV_SCAN_IND:** The device accepts scan requests but does not allow a connection.

There are three more PDU types allowed on advertisement channels, these are **SCAN_REQ**, **SCAN_RSP** and **CONNECT_REQ**. The scanning request and response are transmitted over the advertising channel. Any further communication that happens after a connect request has been received will be sent on the data channels specified in the connect request.

Fields labelled with **RFU** are reserved for future use and should be transmitted empty. The transceiver address flag (**TxAdd**) and receiver address flag (**RxAdd**) are chosen in accordance with the PDU type and indicate if the advertisers and receivers address respectively are either public or random.

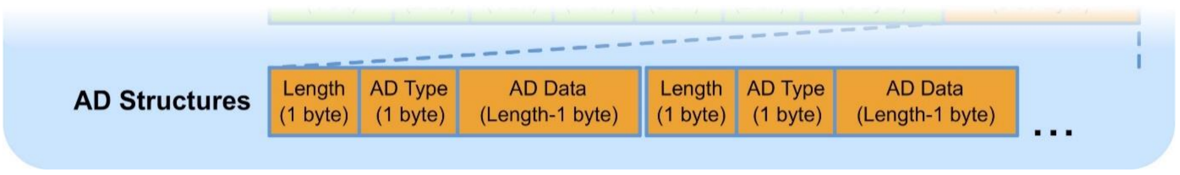


Figure 2-G: Ad Structures

The payload consists of a variable number of AD structures, each labelled with a type flag and a length. There are many different AD structure types like Flags, Lists of Services, Local name and TX power level. The identifiers that are used to label each AD structure are listed in the Bluetooth 4.0 standard Volume 3, Part B, Section 18. AD structures can either be sent in the advertising packet or put in a scan response packet. For optimal use of the advertisement channel and to save power it is advisable to include the most relevant data in the advertisement packet. Using the ADV_NONCONNECT_IND PDU to disallow scan requests can be used additionally to save power and reduce the packet load on the advertising channels.

The TX power level is an important AD structure used for indoor positioning. It should contain the power level in dBm that was used to transmit the packet. The receiver can then calculate the received signal strength indicator (RSSI) by summing it up with the received power of the advertisement packet.

2.3.2.2.2 Data PDU

The Data PDU is only important for connected devices and will therefore not be detailed in an extended manner. It consists of a header and a payload just like the advertisement PDU. Encrypted packets must include an additional 32 bit message integrity check (MIC) which makes them less subjected to undiscovered bit errors than unencrypted packets. It is also possible to validate the correctness of data by transmitting a write prepare request first. The write will be done only after the data has been sent back, validated and a write request is sent. Data PDUs make use of the higher protocol layers that are briefly explained in the next section.

2.3.3 Layers Above the Controller

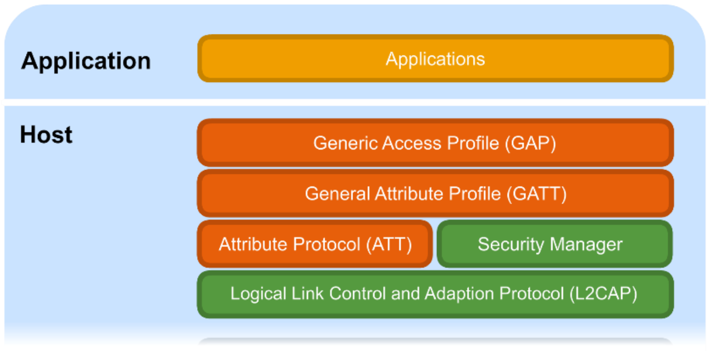

Application
Applications
Host
Generic Access Profile (GAP)
General Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager
Logical Link Control and Adaption Protocol (L2CAP)

Figure 2-H: Higher Layers

The Attribute protocol (ATT) is built on top of the multiplexing layer known as Logical Link Control and Adaption Protocol (L2CAP). The L2CAP layer is shared with Bluetooth Classic but only some of its basic functionality is used in Bluetooth Low Energy. The ATT layer describes a simple architecture where every attribute can be accessed with a known UUID. On top of it is the generic attribute profile (GATT) which defines the terms service and characteristic to organize attributes in a tree structure. A device can advertise any number of services and each service can have any number of subservices and characteristics. A characteristic is nothing more than a value that can be read, written or both. Characteristics can require security restraints like authentication or authorization. Bluetooth LE is a mostly stateless protocol except for transactions. Transactions can be used to write a set of data by sending the data in a write prepare request first and then transmitting a commit to write the data.

On top of these layers and just one layer below the application layer resides the general access profile (GAP). It is a standardized profile with the service UUID 0x1800 that has to be provided by every BLE device. It requires the inclusion of a readable characteristic with the UUID 0x2a00 containing the device name and an appearance characteristic with the UUID 0x2a01. The latter can be used by applications to show an icon that corresponds to the device type. There are other characteristics such as the peripheral preferred connection parameters that can optionally be included in the GAP profile. These characteristics can be read by every connected device using the appropriate UUID. User defined services and characteristics must use 128 bit UUIDs while services and characteristics that are Bluetooth SIG standardized use 16 or 64 bit UUIDs.

The list of services and characteristics can also be enumerated by any connected device. Connected device is called paired after they exchanged keys for an encrypted connection. If the devices store the keys for later use they are called bonded. Storing the keys reduces setup time for subsequent connections.

The Application layer can define more services and characteristics in addition to the general access profile or simply extend it. A set of services and characteristics is called a profile when its definition includes the expected behaviour as well.

3 Alternative Near Field Radio Technologies

There is a wide range of technologies in direct competition with Bluetooth Low Energy. This chapter provides a brief overview of Ant+, NFC, ZigBee and Wi-Fi with a focus on their suitability for indoor positioning. Important characteristics include low energy consumption, cost per chipset, maximum range, supported topologies, licencing fees and availability.

3.1 Ant+

Ant+ is very similar to Bluetooth Low Energy in a wide range of its features. Therefore, some more investigation was done to see how the two standards match up against each other. It is a wireless protocol that was developed as early as 2004[16] by Dynastream. In 2006, Dynastream was acquired by Garmin, a company renowned for their GPS navigation systems. It is a proprietary standard and features standardized profiles like Bluetooth Classic. In contradiction to Bluetooth Low Energy custom profile development is not possible, the only available profiles are the ones that are defined in the ANT+ standard. Furthermore all Ant+ devices have to be certified. Ant+ operates in the same frequency range as Bluetooth Low Energy and shares similarities in data rate and power consumption

Ant+ has already had its share of success in the mobile market. It has been mostly used for medical accessories like heart rate monitors and pedometers which is also its primary target market. Ant+ uses standardized protocols that enable devices from different manufacturers to seamlessly work together[17]. There are some handsets available that come with integrated ANT+ support. This includes many handsets from Sony and some selected ones from other manufacturers. These mostly run the android operating system. Starting with the iPhone 3GS, there is an Ant+ dongle available that can be connected to the dock connector. USB dongles can be used with any PC and many android devices. While this is not a convenient solution there are already more than 100 ANT+ compatible apps available in the Apple App Store. The count of ANT+ capable phones is said to be over 250 million as of October 2012.[18] Ant+ is very flexible in terms of supported network topologies.

The fusion of Bluetooth Classic and Bluetooth Low Energy is a strong indicator for the success of Bluetooth LE. The low cost adder over Bluetooth Classic is something that sets it apart from ANT+ which has to be implemented on a separate chip. Having the ability to design own protocols with BLE while being able to use standardized ones as well is an opportunity for developers but has the inherent problem of incompatibility between different manufactures. A dedicated protocol is however exactly what we want for indoor positioning and it is not yet available in ANT+. It is possible to calculate an RSSI value with ANT+ devices as well but currently ANT+ lacks a protocol for indoor

[16] http://www.thisisant.com/company/d/history
[17] http://www.thisisant.com/consumer/ant-101/what-is-ant/
[18] http://www.thisisant.com/business/opportunities/mobile

positioning, chipset and implementation are more expensive due to licencing fees that have to be paid. The comparison therefore shows a clear advantage for Bluetooth Low Energy.

3.2 NFC[19]

NFC has been primarily targeted at mobile payment, information sharing and augmented advertising where the user is able hold his smartphone near an NFC tag to get additional information. NFC tags can be produced at a very low cost of a few cents per tag and do not require a battery as they gain their power by induction from the presence of an NFC reader. However NFC can be quickly ruled out for an indoor positioning system because of its very short range which is typically less than five centimetres. It would place too much of a burden on the user to hold his smartphone close to available NFC tags on his way. This would resemble a paper chase rather than indoor navigation.

3.3 ZigBee

ZigBee is based on the IEEE standard 802.15.4 for personal area networks. While the IEEE standard just defines the physical and the MAC layer, ZigBee accounts for the higher communication protocols. It is primarily targeted towards smart meters, home automation and remote controls but has already been successfully evaluated for indoor positioning[20]. It has a few drawbacks when compared to Bluetooth LE. Namely that it is more complex which results in a higher cost per chipset[21]. Its power consumption is a little higher and it does not use frequency hopping which makes its communication less solid (1). One of the main problems however in using it for an indoor positioning system is the complete lack of support in today's smartphones.

3.4 Wi-Fi

Wi-Fi has been evaluated in numerous research papers over the last years. The results range from mediocre to good precision. In summary it may be said that none of these solutions can achieve a good position fix that is accurate by 1m if there is no really dense population of visible Access points. With the cost of Access Points comparatively much higher than the typical cost of a Bluetooth LE or ZigBee transmitter and the additional difficulty of having to wire them it has not had a broad market success over the past years.

Google is one of the leaders in indoor navigation with Wi-Fi. In an interview from Techworld Australia a Google software engineer stated that "Maps can currently estimate a user's location

[19] http://www.developer.nokia.com/Develop/NFC/
[20] http://www.freescale.com/files/microcontrollers/doc/brochure/PositionLocationMonitoring.pdf
[21] http://www.ti.com/product/cc2531

inside within 10 metres."[22] This may be enough for wide ranging malls and to find a particular shop but it is not sufficient for a smooth experience and for finding exhibitors at a fair.

Wi-Fi navigation is only viable if the available infrastructure in terms of Access Points is already available. Distributing even more Access Points to enable higher accuracy is costly and deteriorates the primary usage experience of Wi-Fi because signal collisions are bound to happen more often.

[22] http://www.techworld.com.au/article/441838/google_takes_maps_indoors_challenges_remain

4 Client Side Implementation Barriers

In order to develop a widely usable indoor navigation the implementation possibilities on client devices have to be evaluated. That means evaluating some of the most widely used mobile operating systems and devices. At the time of writing, only a fraction of mobile operating systems supported Bluetooth Low Energy. The chipsets in newer mobile devices on the other hand had the support readily integrated. There have not been any statements from the manufacturers to explain the lack of software support. So it can only be speculated that it might partly be a result of driver implementation costs or the result of handset makers who want to push a different technology like NFC.

4.1 Windows Phone 8

When Windows Phone 8 was about to be released to the market, there were rumours that it would support the new Bluetooth 4.0 specification. But despite the fact that one of the flagship phones, the Nokia Lumia had been outfitted with a Bluetooth 4.0 capable chipset[23], the operating system was announced without Bluetooth 4.0 support[24]. It is not clear why Microsoft chose to ignore this feature. Not only did Bluetooth 4.0 have a wide media support at that time but it also involved Nokia - a close partner on Windows Phone 8 - in creating the standard. Furthermore, the kernel of Windows Phone 8 has a strong relationship with the one used in Windows 8. Windows 8 in turn ships with native support for the latest Bluetooth standard. Microsoft has not yet commented on this issue.

4.2 Android

Concerning Bluetooth Low Energy on Android, there is few information available about when it will be supported by the operating system. There have been moves from individual phone manufacturers like Motorola[25] to provide Bluetooth 4.0 support for their respective devices. In order to leverage these APIs, interested developers have to use SDK add-ons that provide separate support for individual phones. This greatly complicates development work.

There was Bluetooth low energy support in Android 4.0 available via the BlueZ stack which has since been dropped in favour of a new Bluetooth stack developed by Broadcom. Apparently, this move has been made to “provide improved compatibility and reliability”[26]. This new stack has now been published under the Android open source project[27]. However, it must be noted, that the

[23] https://www.bluetooth.org/tpg/EPL_Detail.cfm?ProductID=23788
[24] http://support.microsoft.com/kb/2770012/en-us
[25] http://www.motorola.com/sites/motodev/library/bluetooth_apis.html
[26] http://developer.android.com/about/versions/jelly-bean.html
[27] http://www.broadcom.com/press/release.php?id=s721534

committed code does not yet provide support for Bluetooth Low energy, which had been available in the old BlueZ stack. According to Broadcom's press release and the responses from official Google employees, Bluetooth Low Energy support will shortly return as "the next major feature [they] are going to add" [28] in an upcoming release of Android. Though no specific timeline has been announced so far.

4.3 Blackberry OS 10

One of the latest additions to the market of mobile operating systems has been BlackBerry OS10, which is a complete revamp of their mobile operating system. Both, devices[29] and operating system[30] provide support for Bluetooth low energy. As the first BlackBerry OS10 devices reached the market in the finishing stages of this thesis, the possibilities of the API have not been evaluated.

4.4 Apple iOS

Apple has introduced BLE support since Version 5 of iOS[31], with improved support available in iOS 6. Supporting devices include the iPhone 4S, iPhone 5 and the latest iPads, starting with the iPad 2. Because of the good support of BLE in iOS, it has been chosen as a development platform for the prototype. Following is a close look at the features and caveats.

4.4.1 CoreBluetooth API Basics

The dedicated framework to provide Bluetooth Low Energy support in iOS is called CoreBluetooth. In a nutshell it is split into the classes CBCentralManager, CBDescriptor and two delegates called CBManagerDelegate and CBPeripheralDelegate. CBCentralManager is the class that provides the ability to scan for devices, get a list of discovered devices and connect to those that have been found. All the services and characteristics of a device are accessible as soon as a connection has been set up. Available peripherals or new data will result in the invocation of the corresponding delegates.

The CBPeripheralDelegate contains a UUID to identify the peripheral. This is a confusing behaviour because Bluetooth Low Energy uses an aforementioned combination of random and public ids. Therefore it cannot be guaranteed that a device will continue to be available at the same address. The UUIDs are generated by iOS to relieve the app developers from dealing with the internals of Bluetooth Low Energy[32]. A CBDescriptor is just a descriptor for a BLE characteristic.

[28] https://groups.google.com/d/msg/android-platform/CYtxCmtZ-WI/3V7GXLZza0MJ
[29] https://www.bluetooth.org/tpg/QLI_viewQDL.cfm?qid=19361
[30] http://devblog.blackberry.com/2012/09/blackberry-10-native-sdk-update/
[31] https://developer.apple.com/library/ios/#documentation/CoreBluetooth/Reference/CBCentralManager_Class/translated_content/CBCentralManager.html#//apple_ref/doc/uid/TP40011284
[32] http://lists.apple.com/archives/bluetooth-dev/2012/Jan/msg00035.html

In order to receive the required RSSI values, it is possible to set up a connection with all advertisers in reach or to allow duplicate discovery of devices. This should be passed as an option parameter to the method `scanForPeripheralsWithServices:options:`. Connecting with the devices would cause a negative impact on battery life of the peripherals. Reading the RSSI value from the advertising packets is a more elegant way but poses some difficulties in iOS that are explained in the next section.

Other than the mentioned API calls, there are no known ways of accessing Bluetooth Low Energy features. This means that it is not possible to see the advertising channel number of a received packet and there is no access to the signal-to-noise ratio or anything in the actual protocol stack. The only usable information is thus the data that is transmitted in our packets and the calculated RSSI values. This has to be taken into account when researching the possibilities for position determination.

Apples Bluetooth Design Guidelines (4) state that all advertising devices should use all three available advertising channels. The accessory should not use the ADV_DIRECT_IND PDU type to address the iOS device directly on the advertising channel. Further, the broadcasted advertising data should at least contain the AD structures Flags, TX Power Level, Local Name and available Services. The Design Guidelines allow the exclusion of some of these options from the advertising packet but require them in the scan response packet. It is however noted that "depending on its state, the Apple product may not always perform active scanning". The document advises the use of the shortest advertisement interval that is possible which is 20 ms. If the advertisement interval is lower, it is recommended to choose specific intervals that provide an increased detection probability.

iOS 6 allows iOS devices to become a Peripheral themselves. It was also made possible to register for events with the peripheral. As soon as the peripheral has updated data available it will be sent to the iOS device.

4.4.2 Caveats

Apple's iOS is a closed source operating system. This poses some disadvantages for developers. The developer only has control using the provided APIs and everything behind those APIs is a black box. There is no possibility of accessing any low level features that do not have a documented method. Some of the work that happens behind the curtains can only be comprehended by investigating or taking advantage of resources like the Apple mailing list.

Some of the limitations, bugs or peculiarities discovered during testing are listed below. All tests have been carried out with iOS version 5. Some of this behaviour associated with Bluetooth low Energy has changed since iOS version 6.

Tests have shown that when initiating a scan an iPhone will discover advertising devices after a short time and approximately 70% of subsequent advertising packets are registered. This scanning behaviour however changes over time. After a time of more than 10 seconds a higher loss of packets has been observed. This can only indicate a change in scanning parameters. To counteract the deterioration in packet reception a timer has been employed to call the scan method every 10 seconds which was the minimum timespan in which no deterioration had been observed. After this change the percentage of packet loss did not increase over time. And in fact an apple engineer has acknowledged that iOS changes the scan behaviour according to some criteria[33]. The application has no possibility to check for changed conditions with the current API.

According to the official documentation, it is possible to have an app running in the background while listening for BLE events. This is only partially true. While it works well for connected Peripherals, it does not work as advertised for advertising devices. For once, the advertising interval for background applications is very high. According to posts on the official apple developer mailing list, the average detection probability for a device is 60 times the advertising interval while the scanning app is in the background[34]. For a detection probability of 95%, this value increases to 300. If an Advertiser were broadcasting 10 advertising packets every second, that would work out as 30 seconds until the device has been detected at a 95% probability. In order to be able to provide notifications right on time it is possible to set up one or more dedicated nodes that broadcast at the smallest interval possible and set them up in strategically good positions.

Another caveat that is not officially documented is the necessity to supply an array of service UUIDs to the `scanForPeripheralsWithServices:options:` function. If the service UUIDs are not provided it will work as stated in the documentation while the app is in the foreground. But as soon as it moves to the background there will be no more calls to the provided delegate. This has been tested extensively and a workaround is provided in Section 8.2.1. It should be noted that it does not affect the desired use very much because all nodes will change their address over time and will be discovered as new devices afterwards.

It is not clear if it is a bug or a design decision that iOS does not report Advertisers that have already been discovered once while an app is in the background. It does keep track of all the Peripherals that have been discovered since the last factory reset of an iOS device and does not provide further discovery events for applications in the background. This is until an app has attempted to connect to this Peripheral in the meantime[35]. Own research has proven that this behaviour is even valid if the used Peripherals are non-connectable. By connecting and immediately disconnecting it was possible to receive background notifications.

One more bug that prevents apps from scanning for Advertisers in the background has been mentioned on the official mailing lists and has been confirmed by an Apple employee[36]. An app that

[33] http://lists.apple.com/archives/bluetooth-dev/2012/Feb/msg00033.html
[34] As stated by http://lists.apple.com/archives/bluetooth-dev/2012/Apr/msg00051.html and http://lists.apple.com/archives/bluetooth-dev/2012/Aug/msg00143.html
[35] http://stackoverflow.com/questions/9896562/what-exactly-can-corebluetooth-applications-do-whilst-in-the-background/10096244#10096244
[36] http://lists.apple.com/archives/bluetooth-dev/2013/Jan/msg00009.html

runs in the background is eventually killed by the memory manager. This essentially prevents any useful implementation of BLE background support up to the current version of iOS. As with other apps, there is the possibility of using other background modes. This will however impact battery life and will not get approved by Apple in accordance with their app store guidelines.

Some of the behaviour with Bluetooth Low Energy has changed since iOS 6. The first advertising packet that is received no longer includes a UUID for the device[37]. This has led to many crashing BLE apps in the apple store that did not implement an appropriate check. In Addition, iOS 6 has introduced a one minute disconnection delay. A BLE device that has been connected will only disconnect one minute after the connection has been made[38]. This eradicates any possibility to make use of the short connection periods that are possible with Bluetooth Low Energy.

There are many more uncertainties with the way Bluetooth Low Energy is implemented in iOS. An extended list has been gathered by Alastrair Tse and is posted on GitHub[39]. This is just one resource of many where developers attempted to gain a deeper understanding of how BLE works in iOS.

The behaviour of Bluetooth Low Energy in iOS is widely undocumented and poses many difficulties and threats to the future working. Apple may change the operating manner of methods at any point and consequently indispose any application. This has to be taken into account but cannot be worked around.

4.5 Desktop Operating Systems

Most major desktop operating systems already support the latest Bluetooth specification. This includes Windows 8, OS X 10.7 and Linux. Most desktop operating systems can be BLE enabled with suitable drivers after their release as well.

4.6 Conclusion

Concluding this study about Bluetooth 4.0 support in the mobile sector, the only viable solution at the onset of this work had been iOS which provides good support for the standard but poses many difficulties. It is reasonable to think that other mobile operating systems will follow over long or short time. As most modern smartphones already have the chipset implemented, a software update might enable this feature in the future. This is one of the fundamental thoughts that went into the design philosophy of the prototype that is presented later, resulting in a split architecture.

[37] http://olesitune.mine.nu/blelogg/?p=221
[38] http://forum.mkroll.mobi/viewtopic.php?f=1&t=527
[39] https://github.com/liquidx/CoreBluetoothPeripheral

5 Available Bluetooth Low Energy Beacons

Hardware wise, the final prototype will consist of many nodes that are installed in the building to provide the necessary navigation data. Different options have been evaluated and are discussed in this chapter. Critical features are cost, development effort, possibilities and availability. There are dual mode and single mode devices available while the latter is best suited for our purpose because we do not need the additional Bluetooth Classic capabilities. These chipsets exist in soldered or in chipset only solutions.

5.1 TI CC2540

Figure 5-A: TI CC2540
Source: http://www.ti.com/product/cc2540

The TI CC2540[40] is a fully fledged system on a chip that provides a solid development foundation. On an area of not more than 6x6mm and a height of under 1mm it comes with an integrated 128kb flash memory, 8kb of ram and a programmable Intel 8051 core. It supports USB, 128bit AES encryption and all the necessary Bluetooth Low Energy circuits. Texas instruments provides an extended toolset and documentation together with a stack that is royalty free but closed source. If the application grows too big for the internal ram, an external controller can be attached using the provided interfaces. All of these features come in a development package that is available for 99 US$ which includes one USB Dongle, one coin cell powered device and one Debugger that is needed to flash software to the chips. The chipset itself is only available in larger quantities but for as little as 2 US$ per piece.

[40] http://www.ti.com/ww/en/analog/ant_ble/index.shtml

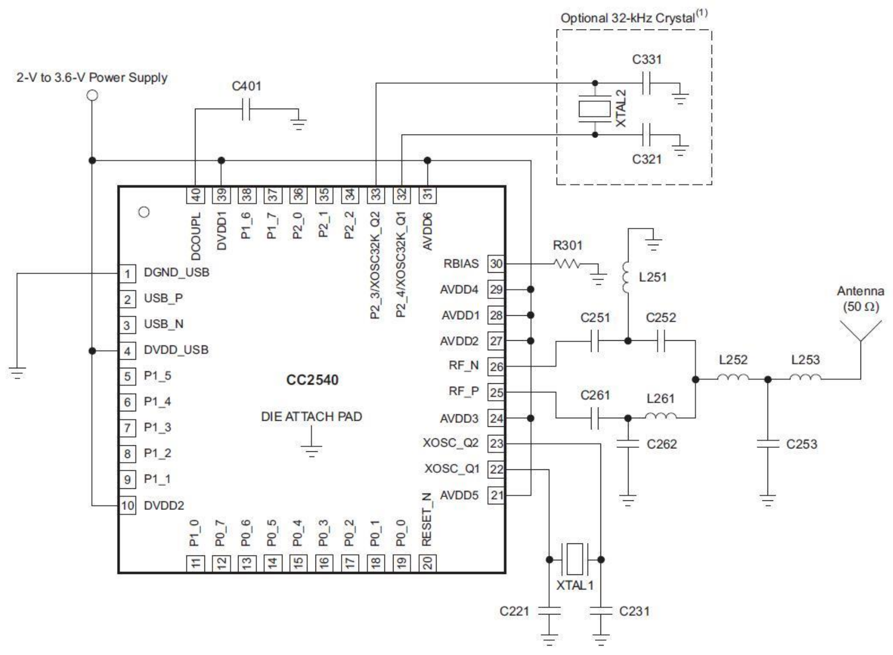


Figure 5-B: Additional components needed
Source: http://www.ti.com/product/cc2540

There are however some drawbacks of using the raw chipset. The chipset comes in a so called quad flat no-lead package that can only be soldered to circuit boards in a complicated process. It has to be outfitted with an antenna, a crystal and some more external components to make it work. An additional voltage regulator has to be added in order to power it via USB. A USB port provides 5V whereas the chipset is only designed to work in the range of 2V – 3.6V. In order to flash new software to the flash memory, debug pin headers have to be added to the circuit. One more drawback of using the CC2540 chipset is that it has to be programmed using an expensive programming environment with a steep learning curve called IAR Workbench. Using the standalone version of the CC2540 chip is not meant for small projects as there will be additional licencing fees to get the final product tested for FCC compliance and similar requirements outside the United States.

5.2 Bluegiga BLE112 / BLED112

Figure 5-C: BLE112

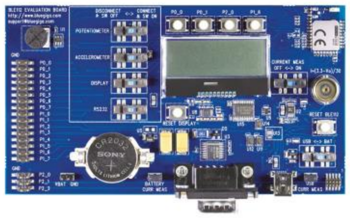


Figure 5-D: DKBLE112

Figure 5-E: BLED112

Source: http://www.bluegiga.com

Bluegiga offers solutions based on the TI CC2540 that have the chipset and the additional components soldered on a PCB. This makes the end product a little more expensive (around 10 US$ for a single BLE112 module in quantities above 1000 according to distributors) but it still stays within a realistic budget. Bluegiga also offers a development kit and a Dongle version which is called BLED112. The dongle offers some development advantages. Software can be flashed in a special direct firmware update mode (DFU) via USB. There are many things to consider when relying on this mode for development which are elaborated upon in Chapter 6. Bluegiga provides a versatile development platform with the DKBLE112 that supports an accelerometer, script debugging, display and many interface connectors at a cost of approximately 300 US$. After considering all the available options, the BLED112 has been chosen for being the cheapest and most easy to use option. It provides an easy interface over USB that is not provided by the BLE112 which would have to be customized to provide a usable interface. One of the disadvantages of using a USB dongle is that the battery drain is not representative for the final product as the USB interface consumes a high amount of power.

Development with the Bluegiga line of products is made easy with the included BGScript and BGApi that allow access to basic functionality on the chip. The BGApi allows to control the device by using an external controller over any of the provided interfaces. BGScript on the other hand is interpreted and executed on chip. The stack is proprietary to Bluegiga and accessed via either the BGApi or BGScript. These are the only methods while a C API for on device development has been announced but is not yet available.

According to Bluegiga's support, a new module aptly named BLE113 is due to come out in Q1 2013 with additional capabilities provided by the TI CC2541 chipset.

Bluegiga's solutions provide a clear advantage over the standalone CC2540 for developing small applications. With the integrated BGScript interpreter it is possible to reduce development time significantly. While the BLED112 provides a good platform for development, the BLE112 is better suited for use in the final product because it is cheaper.

5.3 Nordic Semiconductors nRF51822

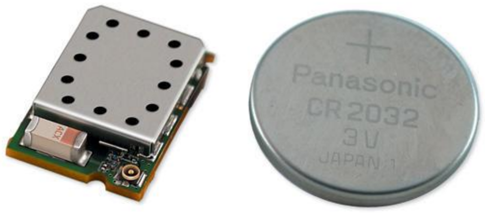


Figure 5-F: nRF51822 soldered to PCB
Source: http://www.nordicsemi.com

Nordic Semiconductors offers a BLE Single Mode system on a chip, the NRF51822[41]. It combines a 32bit ARM Cortex M0 core with 256kb of flash memory, 16kb of ram and a full Bluetooth Low Energy stack with an integrated radio. It is even capable of allowing custom protocols within the 2.4 GHz band. The nRF51822 offers similar functionality compared to the CC2540 and needs to be outfitted with external components as well. The development kit includes five nRF51822 samples, two soldered modules with external components and one USB dongle. Programming is more complex than using BGScript and the provided samples would have to be soldered to a PCB which makes this solution impractical for evaluation.

5.4 Blueradios BR-LE4.0-S2A

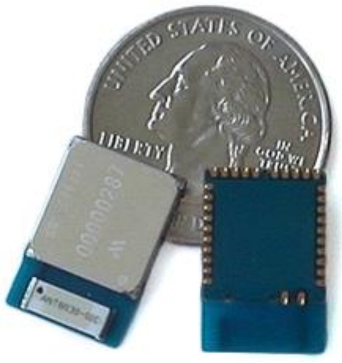


Figure 5-G: BR-LE4.0-S2A
Source: http://blueradios.com (modified)

Another reseller of the CC2540 chipset is Blueradios[42]. Modules soldered on a PCB and outfitted with the required components will set you back 11 US$ if bought in large quantities. Development kits and USB dongles are available as well but at a much higher price of 120 US$. All programming has to be done in the expensive IAR Workbench and using the CC Debugger from Texas Instruments. Blueradios does however provide an API that acts as a simplification layer between the TI stack and

---

[41] http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF51822
[42] http://blueradios.com/hardware_LE4.0-S2.htm

the programmer. The higher cost for development modules and additional complexity have excluded this module as a viable solution.

5.5 Final Choice

The final choice was an easy one. The BLED112 dongles from Bluegiga technologies provided the least cost and development effort for a simple application. Development is possible with just the USB dongles on a Windows machine. No external components are needed and all the software is provided for free. Programming is made easy by using BGScript.

It did prove to be complicated to buy four modules from Bluegiga as all resellers were out of stock. Supply shortages on behalf of component suppliers were to blame for the poor availability. Bluegiga’s support proved to be an invaluable resource for technological questions and for providing me with four modules that were sent out straight from their headquarter in Finland.

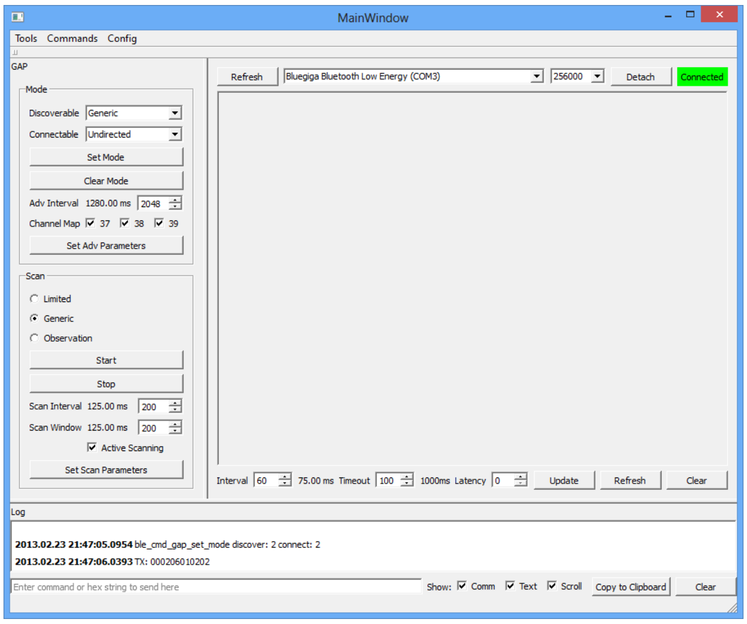
6 Programming the BLED112

The BLED112 comes with its own Bluetooth Low Energy stack and supports the proprietary BGScript which is the only way to enable autonomous usage of the dongle as of now. BGScript is somewhat limited but is capable enough to support everything that was needed for this project. Once the dongle is powered by using the two outer pins of the USB connector it should immediately start broadcasting a specific advertising packet at a high interval. This packet must include the transmitting power to allow an RSSI to be calculated by the client. It must furthermore include a unique identifier for each device because the access address might change any time according to the standard.

The BLED112 allows a transmission power up to +3 dBm. That corresponds to a power of about 2 mW. The exact sensitivity of the iOS receiver has not been published by Apple but research has shown that a maximum distance of about 100m is to be expected.

6.1 Tools and Documentation

The BLED112 can be programmed with the same tools used for the other BLE modules from Bluegiga. The included Blegui2 is an appropriate starting point for development. After all drivers have been installed, it can be used to connect to the dongle by taking advantage of the manufacturer specific BGApi.



Discoverable: Generic
Connectable: Undirected
Adv Interval 1280.00 ms 2048
Channel Map ☑ 37 ☑ 38 ☑ 39
Scan: ( ) Limited (•) Generic ( ) Observation
Scan Interval 125.00 ms 200
Scan Window 125.00 ms 200
☑ Active Scanning
Bluegiga Bluetooth Low Energy (COM3) 256000
Interval 60 75.00 ms Timeout 100 1000ms Latency 0
Show: ☑ Comm ☑ Text ☑ Scroll

```
2013.02.23 21:47:05.0954 ble_cmd_gap_set_mode discover: 2 connect: 2
2013.02.23 21:47:06.0393 TX: 000206010202
```

Figure 6-A: Blegui2

The right pane is used to enable advertising mode and set its parameters as well as putting the dongle in scan mode. Bluegiga did not adhere to the terms that are used in the Bluetooth 4.0 standard which makes some things more complicated than they need to be. This is unfortunately the same in some parts of BGScript. When another device has been discovered, it shows up in the empty area and the software can be used to initiate a connection, enumerate the available services, characteristics and values. There are many more tools available in the menu bar with which it is possible to use most of the features supported in the standard. One of the more important features is the BGApi command that allows booting the dongle in direct firmware update (DFU) test mode.

A project is compiled by using the provided build and compile command line tools. Another command line utility can then be used to flash the custom image to the device. This is needed to use the dongle by itself without being connected to a USB host. When flashing a custom image any error will be deadly to the device. If for some reason it is not possible to boot the dongle in DFU mode, no more firmware updates can be flashed and it will be stuck with whatever software it was flashed. It is possible to open the device and flash another image using a CC Debugger but the soldering work required is very difficult.

Bluegiga does provide some documentation for the toolset but it is rather limited and missing some important things. It was written by people that are familiar with the Bluetooth 4.0 standard and it often implies this knowledge. Some of the terminology does not conform to the official standard and is often hard to guess. The official support provided additional documents and example projects that were needed to implement certain features.

Bluegiga also offers a C API that can be used to control the dongle by using BGApi commands to invoke functions. This API is however strictly for developing on a separate host and cannot be used to program the chip itself. BGScript was used for programming the dongle to enable its usage without a host.

6.2 Developing a Simple Advertiser

A simple project consists of an assortment of XML files that describe the behaviour at runtime. Necessary are:

- Project.xml which lists all of the other files that are necessary to compile the project
- Cdc.xml which is a hardware descriptor file and can be copied over from an example project
- Hardware.xml which is used to configure device behaviour, e.g. transmission power
- Gatt.xml which describes the GATT profile explained in Section 2.3.3 and mirrors its structure

A very basic example of a valid gatt.xml is provided in the listing below. It should be self-explanatory if the basic structure of a GATT profile has been understood. The following gatt.xml includes the necessary GAP service and its two mandatory characteristics in addition to the custom service that is used for indoor positioning.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>

    <service uuid="1800">
      <description>Generic Access Profile</description>

      <characteristic uuid="2a00">
        <properties read="true" const="true" />
        <value>NodeM1</value>
      </characteristic>

      <characteristic uuid="2a01">
        <properties read="true" const="true" />
        <value>512</value>
      </characteristic>


      <service uuid="67a45dd0-4533-11e2-bcfd-0800200c9a66">
        <description>FindMilkService</description>
        <characteristic uuid="88f74010-4533-11e2-bcfd-0800200c9a66">
                  <description>A Value to read</description>
            <properties read="true" const="true"/>
            <value>TestValue</value>
        </characteristic>
    </service>

</configuration>
```

Listing 6-A: gatt.xml example

All these xml files are then compiled with the bgbuild command line utility which outputs a binary file that can be flashed to a device in DFU mode. After flashing of the dongle it can be used in conjunction with the Blegui2 to broadcast the configured values. But the dongle will not start advertising without the correct BGApi command transmitted over the USB interface.

In order to start advertising as soon as the device is powered up, extra steps must be taken. This information is not included in the official documentation because it disables access to the dongle via BGApi in order to enable BGScript access. The dongle is then run under sole discretion of BGScript and will not allow booting into DFU mode if appropriate measures have not been taken. Enabling script access requires the addition of some xml tags and a proper BGScript file. A very simplistic version is outlined in Figure 6-B.

```
# System Started
event system_boot(major, minor, patch, build,
ll_version,protocol_version, hw)
# Set advertisement parameters. Intervals are multiple of 625µs
# First parameter: minimum interval, second: maximum interval
# 7 equals the use of all three advertising channels
call gap_set_adv_parameters(160, 160, 7)
#Enable advertising mode
call gap_set_mode(gap_general_discoverable,gap_directed_connectable)
end

# Event listener for USB Data. If data on the USB interface is
received,
# Dongle will boot to DFU mode
event system_endpoint_rx(endpoint, datalen, data)
call system_reset(1)
end
```

Figure 6-B: Simple BGScript example

The last event listener is called as soon as data is received over the USB interface. Whatever data might be received will be ignored and the dongle will boot straight to DFU mode. This enables further reprogramming of the dongle.

One of the problems of using this approach is that it will make the device connectable which is not the intention. The device was not using the advertising packet to broadcast all data but relied on scanners to send a scan request. It responded to these requests with an appropriate scan response that contained the remaining data. This resulted in increased power usage and used the advertising channel more than required. In an attempt to make the device non connectable, essential data had been missing and no scan responses were sent out anymore. The technical support proved to be a reliable source once more and provided a code sample for custom advertising. This was used as a starting point to write the final script.

6.3 An Advanced Advertiser

The following code listing in Figure 2-A was used in the final project and defines the advertising packets byte by byte. It follows the link layer advertising structure that is explained in Section 2.3.2.2.1. Defining the advertising packet ourselves enables fine granular control over what is being sent over the air.

```
dim adv_data(31) # custom advertisement data

# System Started
event system_boot(major, minor, patch, build, ll_version, protocol_version, hw)

    # flags for discoverable/connectable
    adv_data(0:1) = $02 # ad field length = 2 bytes
    adv_data(1:1) = gap_ad_type_flags   # ad field type = 0x01 (Flags)
    adv_data(2:1) = $06 # flags = 0x06, Single mode BLE / general discoverable

    # tx power field advertisement, needed for iOS and used in the FindMilk
Client
    adv_data(3:1) = $02 # ad field length = 2 bytes
    adv_data(4:1) = gap_ad_type_txpower    # ad field type = 0x0A (TX Power)
    adv_data(5:1) = $03 # TX power = +3 dBm, corresponds to a tx_power = 15 in
xml

    # Set Node name to ASCII String 'NdeM1'
    adv_data(6:1) = $06 # ad field length = 7 bytes
    adv_data(7:1) = $09 #ad field type = 0x09 (Set complete local name)
    adv_data(8:1) = $4e # 'N'
    adv_data(9:1) = $64 # 'd'
    adv_data(10:1) = $65 # 'e'
    adv_data(11:1) = $4d # 'M'
    adv_data(12:1) = $31 # 0x31='1', 0x32='2', 0x33='3', 0x34='4'

    #Broadcast  service  ID  in  initial  package  67a45dd0-4533-11e2-bcfd-
0800200c9a66
    adv_data(13:1) = $11 # ad field length = 17 bytes
    adv_data(14:1) = $07 #ad field type = 0x07 (Complete list of 128bit UUIDs)
    adv_data(15:1) = $66

     […]

    adv_data(30:1) = $67

    # set advertisement interval, use all three advertisement channels (7)
    # (note min/max parameters are in units of 625 uSec)
    call gap_set_adv_parameters(32, 32, 7)

    # set custom advertisement data (type=adv_data, data_length, data)
    call gap_set_adv_data(0, 31, adv_data(0:31))
end
```

Listing 6-B: Extended BGScript example

The package contains four AD structures. The first AD structure contains the flags field. This field must be included if the device contains a Single Mode chipset. It may be omitted under special circumstances that are explained in the standard. Secondly, the transmit power is written in the TX power field. It has to be set to the same value used in the hardware configuration xml file. There is a discrepancy in the number format used, so the transmission power of 15 that was used in the hardware.xml corresponds to a 0x03 (+3 dBm) value that is sent over the air. This allows the client to properly determine the RSSI by summing it up with the power that the packet was received with. Next in line is the AD structure that holds the node name. Different firmware has to be flashed to all of the 4 nodes to properly name them. The last AD structure contains the 128 bit service UUID for the FindMilk service. It has been shortened in the code extract. It has to be sent in least significant octet first order according to the standard.

After the buffer has been filled with the data, a call to the function `gap_set_adv_data` sets it as the advertising packet to be transmitted. The buffer uses the full 31 bytes that are available for any Advertising PDU payload.

Using this version over the one presented in the last section relieves the advertising device from sending two packets instead of one. It thereby allows the client to do without active scanning and use passive scanning instead. Active scanning will result in one scan request after each advertising packet (1). As a result the advertising device does not have to listen to the advertising channel anymore and results in a reduced power consumption. Less collisions on the advertising channel will occur because the package count is lowered from three packets to only one. If there was a Bluetooth LE API that allowed developers to choose between active and passive scanning it could be possible to remove some of the information from the Advertising PDU and put the information in the Scan Response PDU instead. This information could be cached until the advertiser changes its address and will therefore be considered a new device. This would help to reduce the load on the advertising channels even further. Forcing passive scanning is however not possible with iOS. Sending all the information in the Advertising PDU is therefore the best solution for now.

A low advertising interval is important to improve accuracy but also results in a higher collision probability. This has been examined in Section 7.2.6. First tests with four packets per second have proven to be inaccurate while 10 packets per second have provided good results. For most of the measurements, an advertising rate of 50 packets per second has been chosen to allow for short measurements.

6.4 The Finished Nodes

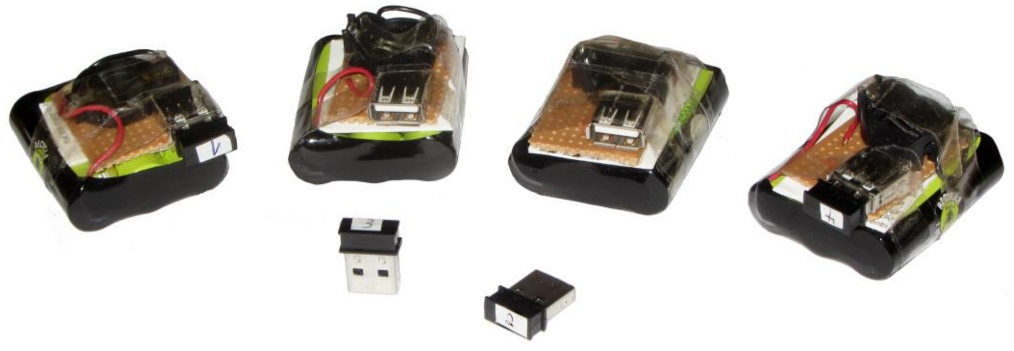


Figure 6-C: Nodes with battery packs

Some soldering work was done to outfit a USB jack with a CR2032 coin battery holder. The 3 volts provided by the coin cell battery have however proven insufficient and resulted in random device restarts. The voltage regulator used in the dongle version had lowered the voltage to an unusable amount to still power the chip. After the initial plan had been foiled it was decided to up the voltage and include three industry standard AA batteries which should provide enough juice to power the dongle for a few years to come.

7 Bluetooth Low Energy Analysis

7.1 Initial Thoughts

Before any programming was done a collection of thoughts to important topics has been gathered. What are possible ways of calculating a user's position? What possibilities remain when using Bluetooth 4.0?

There are a few different methods for determining a position based on radio signals.

Using the RSSI to calculate the distance and then simply triangulate the position by using the known positions of the advertising beacons will probably not work because of signal reflections and dampening caused by persons, objects and building structure in general. Moreover, every physical antenna has an individual sending characteristic that would have to be taken into account when positioning the nodes and doing the calculations. The behaviour of BLE signals in different scenarios have been measured in the next section.

Fingerprinting is a commonly used technology for Wi-Fi positioning and provides better results over simple triangulation in complex scenarios. Fingerprinting data has to be collected in a calibration phase and has to be linked with position data for each measurement. The trade off in comparison to triangulation is that a map has to be created first. This requires a calibration which can only be done manually by doing a measure every few metres and annotate it with the current position. In contrast to outdoor positioning using Wi-Fi, it is not possible to crowdsource the data because no position data is available as compared to GPS which can be used outside to match up the Wi-Fi fingerprint with the required position data.

There is no additional information apart from the RSSI that could be used to determine the location. None of the smartphone operating systems that support Bluetooth Low Energy enable direct access to the receiver to record data like the signal-to-noise ratio or access to the physical layer itself.

Complex algorithms exist that take signal expansion into account. Setting these up in a complex indoor location that is bound to change over time however presents difficulties.

Another method that could provide reliable results in more open environments uses the direction of departure. Every packet that is send should be sent through a highly directed antenna. This would enable to calculate the crossing at which two packets have been received by incorporating the known position of the sending nodes. This is an interesting thought and has been considered but directed signals where hard to realize with the available hardware. It has therefore not been investigated further.

There are two methods to gather the needed RSSI data. Advertisement packets can either be collected by a smartphone or the smartphone itself broadcasts the advertisement packets which get picked up by beacons. Using the second method has some drawbacks. The beacons must have a way of sending the acquired data to a processing machine. Therefore they have to be linked to some infrastructure which makes the setup more complex. In a crowded environment the packet load on the advertising channels will go up and the rising number of collisions might degrade the performance of the system. Therefore it has been decided that the beacons broadcast the advertising packets and the user's device simply acts as a passive scanner. Keeping the packet length as low as possible will ensure minimal use of the advertising channel.

Using data gathered by other indoor positioning methods like the ones mentioned in Section 1.2 can be used in addition and contribute to a higher robustness and accuracy.

The direction of the antenna used for navigation will probably have a high impact on the received signal strength and so will people blocking the antenna from one direction. Requiring every user to hold his smartphone in a specific way is not feasible and shielding of the antenna by a hand or other persons will affect the position estimate as well.

7.2 Measurements

In order to inspect Bluetooth LE many measurements were made to find characteristics that can be used to determine the position. Before every measurements, some assumptions have been made. These are presented together with the results achieved by measuring. BLED112 devices were used as transmitters with a transmit power of +3 dBm. An iPhone 4 was used as signal receiver.

7.2.1 A Simple Measurement

As with all radio signals, the measured RSSI will vary over time. A measurement was conducted to observe the deviation and stake out the RSSI values that are to be expected.

The RSSI fluctuation was between 3-6 dB for most measurements. Some few RSSI values have a very high deviation and should be filtered out. At some points, RSSI values such as +8 dBm were received, these must be filtered out because they are impossible when only sending with a transmit power of +3 dBm. In general, the highest valid RSSI that could be measured was -14 dBm by placing the antenna as close to the receiver as possible. The lowest RSSI that could be observed was -96 dBm. These values outline the bounds for usable values.

To get a reliable distance indicator, an average has to be calculated. Some mathematical models have been compared against each other. Median, arithmetic average have been tested and both of those while first filtering out ricochets first have been tested as well. Two measurements were compared by using the different averaging methods for each and then comparing the results. Filtering out a small percentage of values with the highest deviation and then calculating the arithmetic average resulted in minimally better results and has then be used to calculate averages for further measurements. There are other statistical methods like the cornish-fisher method to calculate averages that may be better suited. These were however not tested.

7.2.2 Field Study

The path loss does not have a linear relation to the distance when measured in dB. To get a better relation between RSSI and distance a field study was made. As opposed to field studies in a figurative manner, the purpose was to find a uniform surrounding.

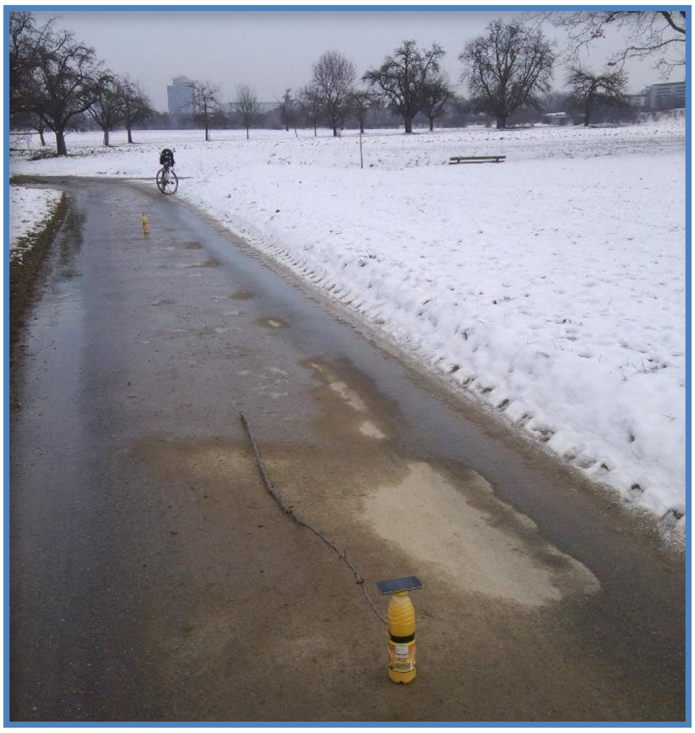


Figure 7-A: A literal field study

A measurement has been taken every length of a branch or in scientific terms every 2.5 metres. After calculating the average of all measurements they were visualised in the diagram below.

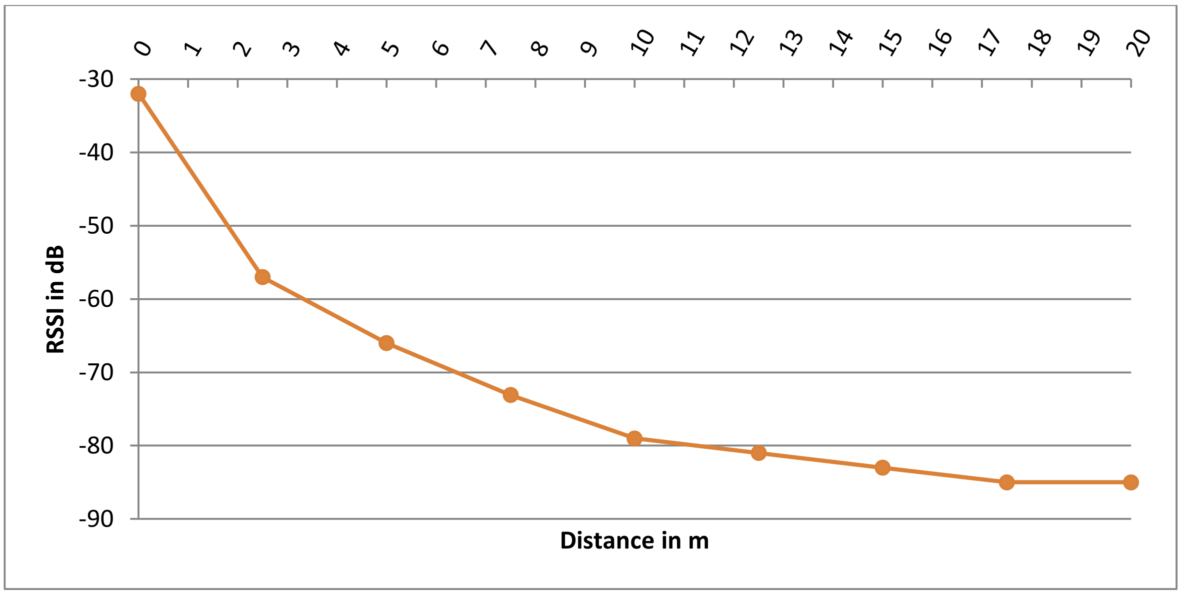


Figure 7-B: Measured RSSI by distance

In the above diagram the RSSI shows significant changes for the first 10 metres but not thereafter. Looking at a mathematical derivation strengthened this assumption. In order to provide good results, nodes should therefore be placed at a maximum distance of 20 metres or even better 10 metres to improve the result by having more signals to choose from. Problems that might evolve from choosing a high node density are evaluated in Section 7.2.6.

The path loss for an optimal spherical transmitter can be determined with the following formula according to Heydon (3).

$$path\ loss = 40 + 25 * \log(d)$$

Using the above measures a path loss formula p_1 was determined that mirrors the measured conditions up to a maximum range of 20m.

$$p_1(x) = \begin{cases} 32, & x < 0 \\ 32 - 0.01721x^2 + 19.8256 * \log(1 + x), & x > 0\ and\ x < 20 \\ 85, & x \geq 20 \end{cases}$$

This function has been plotted together with the measured data along the x axis in metres while the path loss is represented on the y axis in dB.

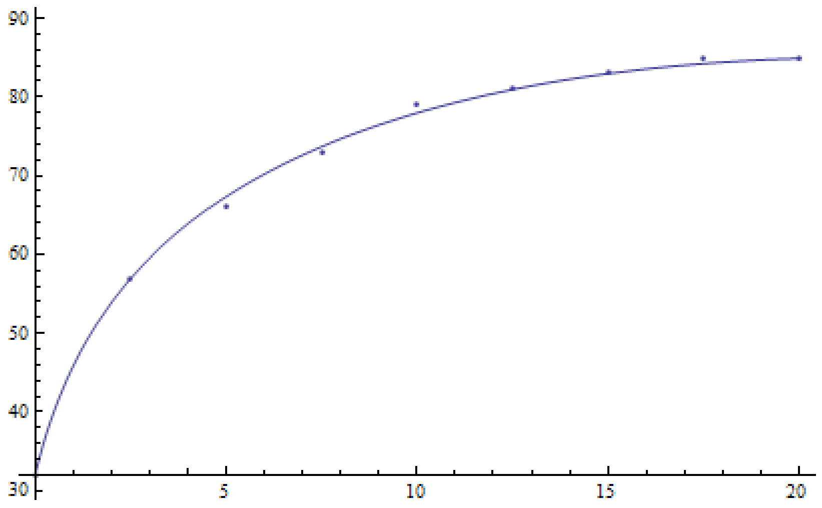


Figure 7-C: Plot of P_1 path loss function

In order to get a better relation between the distance in metres and the matching signal strength, the above data has been taken and x and y axis have been interchanged to calculate an inverted function. A regression was made to determine a good fit. The following graphs show the resulting function on the left and its regression residuals to the right.

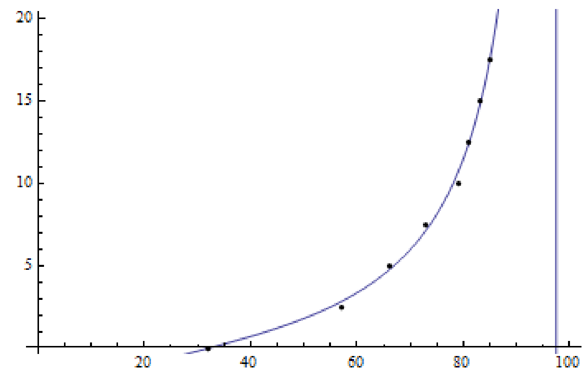


Figure 7-D: Plot of path loss inversion function P_2

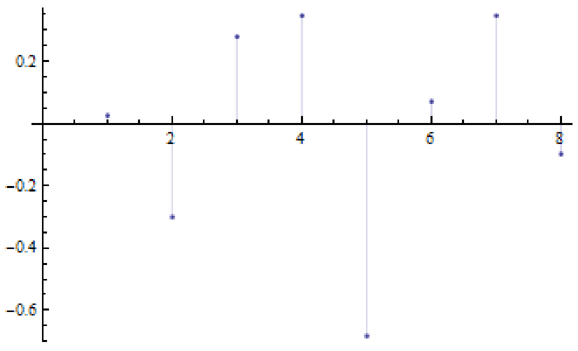


Figure 7-E: Residuals of P_2

The left graph shows the measured data plot and the resulting function p_2 that matches the data plot as close as possible. The residuals are shown in the right graph with the highest deviation from the plotted data at 0.6 m distance. The equation below can be used to linearize any measurement and convert any RSSI value to an approximate distance in metres. It outputs an approximate equivalent in the range of 0 to 20 metres.

$$p_2(x) = -3.2584 - \frac{262.4872}{-97.4118 + x} - \frac{25}{x}$$

The deviation from the measured data by up to 0.6 metres is not critical as the function itself is never used to estimate the distance itself, but rather to transform the measurements to a scale that is roughly linear.

Below is a graph that shows the measured RSSI values after they have been transformed.

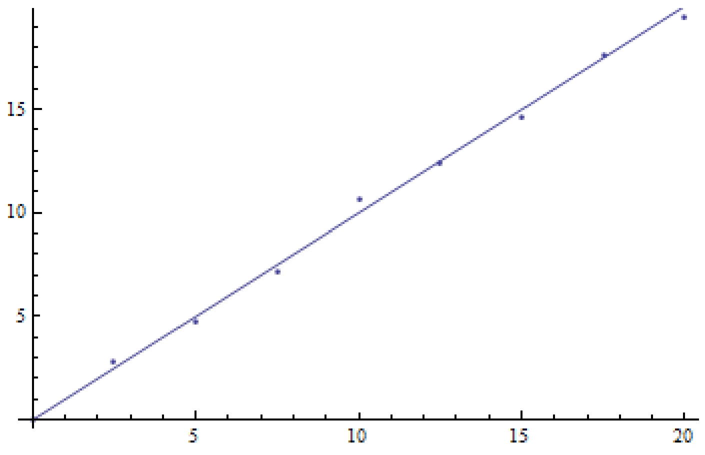


Figure 7-F: Transformed RSSI data

The above graphic conforms to the desired result. The function follows a linear path if called with a series of measures that have been taken at a multiple of a fixed distance.

The next graph shows two series of measurements. The first one was taken every 1.25 metres for 1 second while the second one was taken every 2.5 metres but for a much longer 30 seconds. The maximum RSSI difference is 3 dB. This would produce an error of about 1 metre.

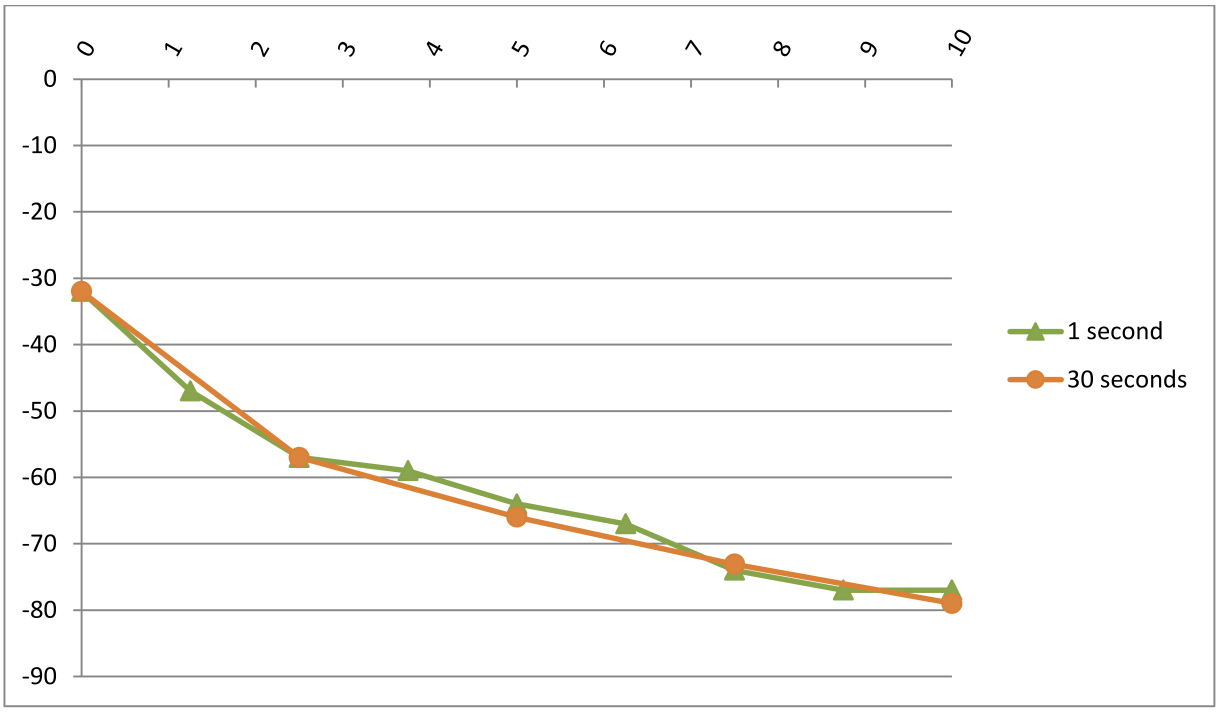


Figure 7-G: Comparing two measurements

The error rises severely after 10 metres which makes an RSSI below -80 dBm an unpredictable indicator for distance. This can be observed in the following graph that compares the 30 second calibration measurement to the second measurement. While the error stays within about 1 metre to either side, it grows significantly at high distances.

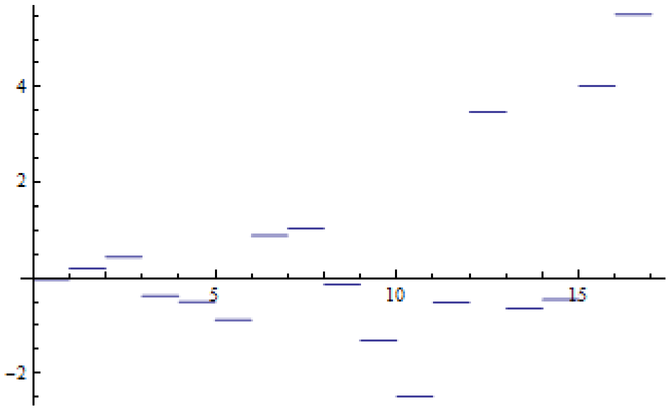


Figure 7-H: Error in metres depending on the distance

7.2.3 Advanced Characteristics

Further measurements have been taken in a building. Some of the questions raised were if a fingerprint of a location based on the RSSI distribution could provide usable results. Another assumption was, that packet loss should grow with farther distances or complex surroundings. The last hypothesis that needed an answer was if the variance of the RSSI values can be used in the fingerprint.

Individual measurements that were taken at the exact same place but at a different time were compared against each based other on their RSSI distribution. Two examples can be seen below. The RSSI values are denoted on the horizontal axis while the sum is shown on the vertical axis. No visible resemblance has been found. This means that the RSSI distribution is not a usable indicator for a fingerprint.

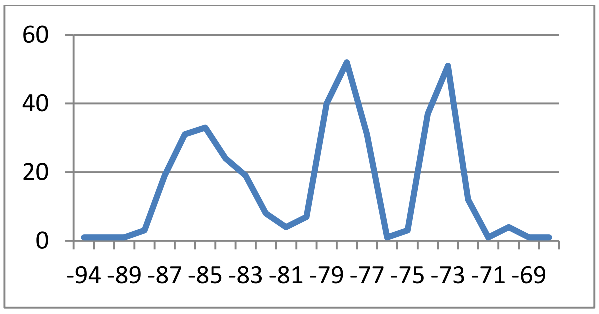


Figure 7-I: First measurement at 1m distance

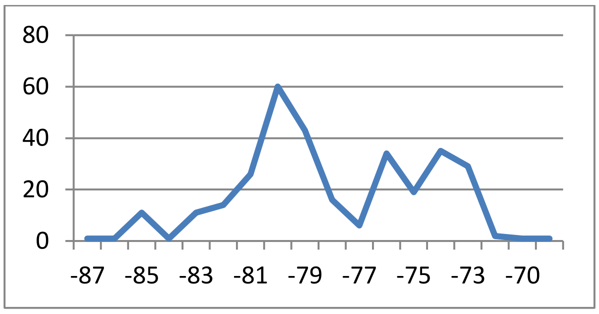


Figure 7-J: Second measurement 1m distance

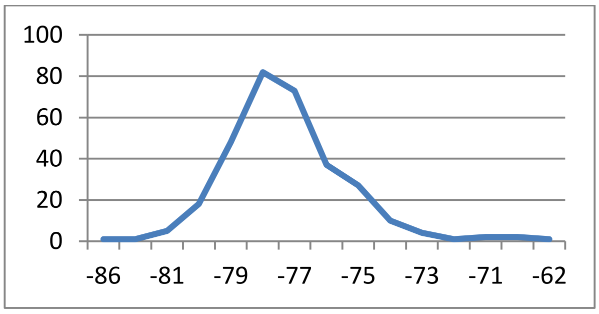


Figure 7-K: First measurement at 7m distance

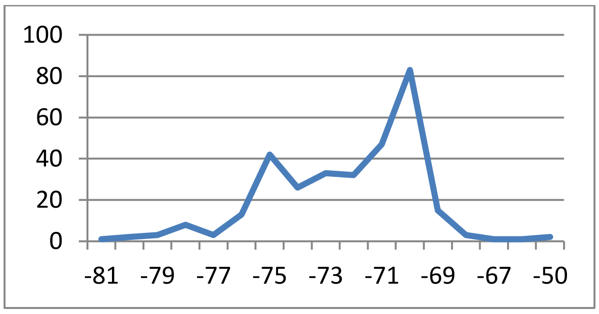


Figure 7-L: Second measurement at 7m distance

Packet loss has been observed to increase at larger distances and when doing measurements in places which a lot of furniture or other objects. It is however also influenced by the scanning behaviour of the client device which cannot always be altered or monitored. Packet loss is also bound by probability and will change with increased interference on the advertising channels. It can therefore not be easily used in a fingerprint.

The variance of the signal did also increase with higher distance and in complex surroundings. Measurements very close to the transmitter usually provided a very low RSSI variance of not more than 3 dB for the individual values but measurements in the range of 1-10m usually provided an equal variance. The first metre can usually be measured well enough without using an added parameter. This means there is no need to include the variance in the fingerprint.

7.2.4 Rotating the Antenna

There is no physical antenna that behaves as a perfect receiver in any angle to the transmitter. The same is true for the signal transmission. The graphic below shows the top view radiation pattern of a BLE112 as an example.

Transmitter and receiver have been positioned at a distance of one metre and the receiver has been turned by 360° at a constant speed. And indeed the test shows that the antenna direction has a very high relation to the RSSI.

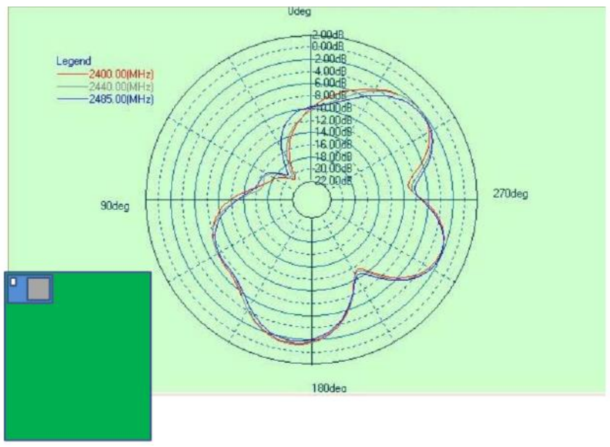


Figure 7-M: BLE112 radiation pattern top view (5)

The following graph shows the RSSI over time that has been recorded in this experiment. The black line visualizes the average RSSI.

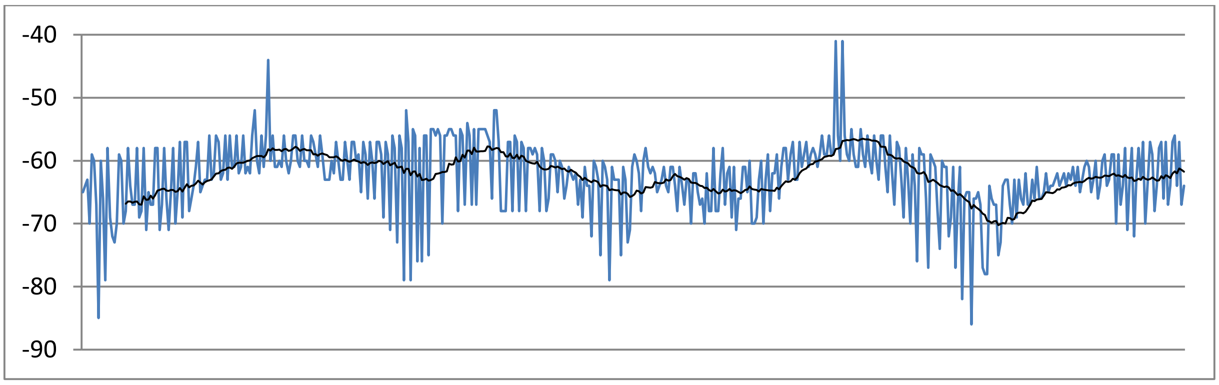


Figure 7-N: Turning the antenna

This measurement exhibits a strong relation between antenna direction and RSSI. The variance seems to have a strong relation with the direction as well. The average RSSI varies by about 13 dB. This suggests to do the initial calibration in different directions and use the magnetometer to get a better position fix.

7.2.5 Shielding the Antenna

There are many ways a radio signal can be obstructed. This happens by unintentional shielding with a hand or by a person that is standing in between a node and the receiver. Some of these possibilities have been tested and evaluated.

For the absorption tests, the receiver was 1 metre away from the transmitter. The average RSSI that was measured without any obstruction was -64 dBm. Placing a softcover book with 300 pages about 10 cm in front of the receivers lowered the average RSSI to -67 dBm. Replacing the book with a human arm resulted in an RSSI of -69 dBm. Water absorbs radio waves in the 2.4 GHz spectrum quite well (6). This is why the band is also used for microwave ovens. The human body consists to a high percentage of water. In short this means that different humidity levels will also have an effect on the RSSI. Comparing relative RSSI values will therefore provide better results than comparing absolute values.

A human body placed in between transmitter and receiver at about 20 cm away from the receiver lowered the average RSSI by 6 dB while a placement about 5 cm in front of the receiver lowered it by a total of 15 dB. This presents some serious deterioration of the signal. By placing the nodes at about 2-3 metres above the floor, a line of sight between receiver and transmitter can be achieved in most cases. Mounting the nodes on a ceiling at that height will provide very good results as a line of sight is possible in most cases. However mounting them higher will decrease reliability. To further improve the position estimate, the signal that is blocked by the body of the user can be identified by using the inbuilt magnetometer. Signals that are influenced at a lower probability can then be favoured.

Different grip positions of the smartphone were tested as well. The first third shows the smartphone being held with no obstruction of the antenna at all. The second third shows it being held in a normal way while the last third shows a firm grip that covers most of the antenna.

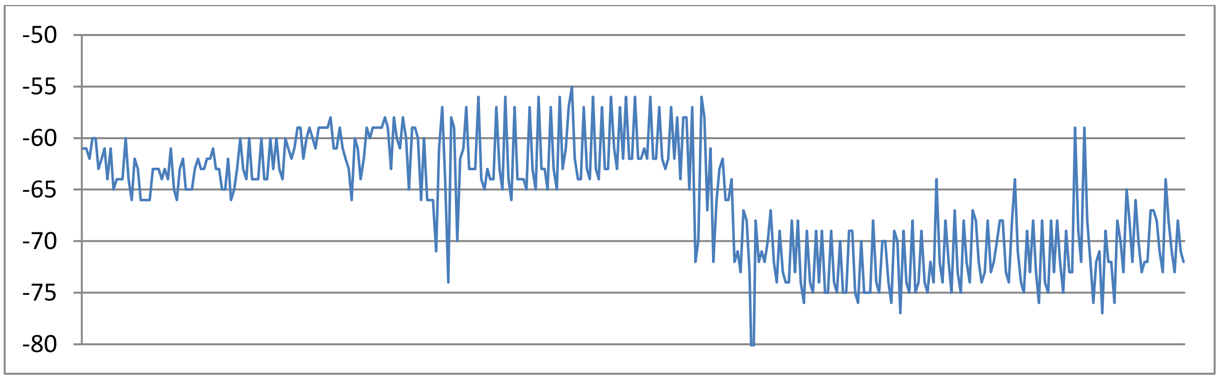


Figure 7-O: Hand grip measurement

The first two versions of holding the phone were deemed to be very common amongst smartphone users while the third was a more artificial version of holding it and will not affect a lot of users. The position of the hand seems to be less critical than assumed. The first two versions provide nearly the same signal strength. This suggests that smartphone manufacturers have tried their best to place the antenna in a way that will not deteriorate signal strength across different users.

7.2.6 Packet Collision Probability

The packet collision probability on the advertising channels is quite high when using a high number of nodes in open environments. Office buildings on the other hand will most likely not be faced with the problem of overused advertising channels as the building structure prevents the signals from spreading over a large range.

When placing one node with a receivable range of 100 metres every 10 metres in both vertical and horizontal direction, we end up with approximately 250 devices that can be received by an optimal spherical receiver at one place. Doing the same with nodes that are placed every 15 metres results in about 130 devices. The used advertising packets have a length of 47 bytes at the physical channel. That corresponds to a transmission time of 376 µs at the Bluetooth Low energy symbol rate of 1 MHz. After adding the 150 µs inter-frame gap specified by Bluetooth LE, we end up with a maximum of about 1900 packets that could be transferred under optimal conditions on one channel.

This starts to get more complex when we consider the collisions that will take place in the simple TDMA scheme that the advertising channels rely on. The calculations that have been done can only provide approximate values. Getting exact values would require many measurements with a multitude of nodes and could only be described by a complex mathematical model.

The probability for one packet to be transmitted on an advertising channel is 1. A second packet that is transmitted within the same second on the same channel will have a probability of:

$$P_{tx2} = \frac{1\,000\,000\ \mu s - (376\ \mu s + 150\ \mu s)}{1\,000\,000\ \mu s}$$

The average count of packets that are sent without collisions can thus be calculated with the following formula where a is the number of packets sent and S equals to the result.

$$S = \sum_{i=0}^{a-1} \left(\frac{1\,000\,000\ \mu s - (376\mu s + 150\mu s) * i}{1\,000\,000} \right)$$

An optimal value can be found for a set amount of nodes in receiving distance. The optimal packet transmission rate for 250 nodes that are in receiving distance was determined to be 7.6 packets per second. This results in an average of 3.8 packets being sent without collision on one channel. The optimal transmission rate with 130 nodes in range is 14.6 packets per second which results in an average of 7.3 collision free packets per second. These numbers can be multiplied by three when using all advertising channels. The amount of received packets has to be considered independently and is a function that is dependent on scan window and scan interval. The iOS scanner for example managed to receive about 2/3 of the packets that were sent.

Not all colliding packets within the receivers range are bound to cause bit errors. According to the Bluetooth specification Volume 6, Part A, Section 4.1, the co-channel interference performance ratio is 21 dB. This means that packets from a node with a 4 m range will be received even if they collide with packets that were sent from a distance of 20 metres. This can be calculated by using the appropriate path loss formula.

In conclusion every system has to balance between node density and advertising interval. After the node distance has been determined, the optimum advertising interval can be calculated in order to get the highest throughput and use as little power as possible.

7.3 Results

The only reliable characteristics that have been observed were the RSSI values themselves. The variance, packet loss and distribution have not shown a reliable correlation with the position over time. The signals are of most value when received close to a transmitter. Higher RSSI values should be preferred for calculating the position. Distances above 10 metres have shown significantly worse distinguishing features with the used system. The calibration data can be used to calculate the area that one node was able to service. The smaller this area, the better the estimate that is gained from this node.

The RSSI values show a high variance and therefore need to be averaged. Filtering out a low percentage of ricochets and then calculating the arithmetic median has provided best results but there are many more mathematical filters that could be used to calculate more reliable averages. Using a Kalman filter would be a good starting point. In order to provide a steady position estimate between two points, a ring buffer can be used.

Completely covering the antenna will cause worse reception from all nodes. This has to be taken into account. Using an algorithm that depends on relative values might solve that problem.

Signals have not been severely deteriorated based on the user’s smartphone grip. However, they are strongly influenced by persons in the transmission line. In order to provide good results the magnetometer will have to be used. When additionally used in conjunction with multidirectional calibration measurements the results will be much better.

When using the system in an office building, the advertising channel will provide enough bandwidth to support a high advertisement rate. When used in open environments a compromise between node density and advertisement rate has to be made.

Antennas with a strongly directed beam could be used to enable an angle of departure calculations which would provide superior results in open environments. These would however probably provide inferior results when used in dense surroundings.

When a measurement is received and compared to the calibration measurements, the strength of each node will in most cases be lower than what the system was calibrated with. Radio signals in the 2.4 GHz range have a higher probability of being absorbed than being reflected. This means that we can prefer matches where the calibration measurement has the same or higher RSSI. Additionally, the likelihood that the average RSSI at a certain point will increase is a lot lower than the likelihood that it decreases.

The system performance can be increased by using additional methods like pace detection, an accelerometer and other sensors. A reliability factor has to be calculated for every position fix in order to incorporate the other systems only for position fixes that are not accurate.

8 Developing the Client App

An app for iOS has to be developed that is able to scan for advertisers. The acquired data has to be processed and saved. The caveats that have been discussed in Section 4.4.2 must be accounted for. It should then be possible to use the algorithm that was developed in the previous chapter to determine the user's location based on test measurements. The algorithm has not been implemented with all of its presented features because of its complexity which had been out of scope for this work.

8.1 Technology

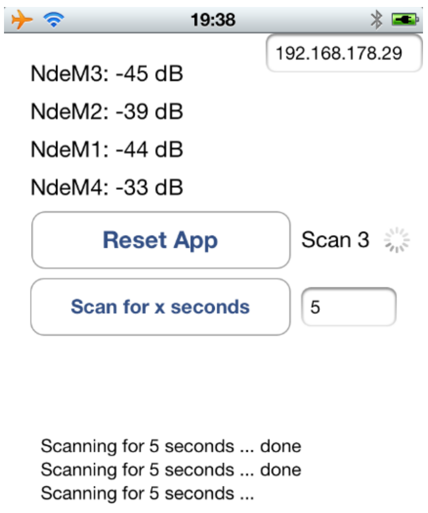
When developing on mobile devices, there are a few things to consider. One can choose from developing a native app which is then compiled and run on the device, writing an App that is run in a virtual machine or combining two technologies which results in a hybrid app. Native apps usually only work on the platform they were developed on because they rely on many APIs and frameworks that are unique to one platform. Interpreted apps and web apps rely on a virtual machine that executes the intermediate code and user interface layouts and transforms it to machine readable code. This allows being platform agnostic but comes at the cost of performance and often features. The third approach has been chosen for developing the smartphone app because it provided many benefits. The app was developed as a combination of a native app written in Objective-C and an Air app which is developed using the Adobe Flex tool chain and was programmed in ActionScript 3.

The Objective-C app is used to initiate scanning and connects to the entered IP address using a TCP/IP socket. A computer that owns this IP address constantly listens on port 7777 for an incoming connection. Once the connection is established, the received RSSI signals together with the node names are transmitted over that socket for further processing. The beauty of this division in two parts is that the algorithm and all the data management can be done on a desktop computer while the smartphone client does not have to be updated for any changes in the algorithm. This enables a fast prototyping workflow and allows easy porting of the final app to other platforms.

Adobe Air is capable of using native extensions. The native client app can therefore be transformed in a library that enables Air to communicate with the BLE controller in the smartphone. The Air app together with the native extension will then be compiled and packed into a working smartphone application. Adobe Air right now works on iOS, Android, Windows, OS X and BlackBerry OS10. In order to publish the app for a specific platform, all that has to be done is writing a native library that allows communication between Air and the BLE controller. The rest of the code enables a write once, use everywhere architecture and this includes the full user interface.

8.2 The Native App

8.2.1 Usage



NdeM3: -45 dB
NdeM2: -39 dB
NdeM1: -44 dB
NdeM4: -33 dB

```
Scanning for 5 seconds ... done
Scanning for 5 seconds ... done
Scanning for 5 seconds ...
```

Figure 8-A: The iOS App

Usage of the app is pretty straightforward. Once the Air application on a computer has been opened and is listening for incoming connections. The IP of that computer should be entered in the text field placed at the upper right corner of the screen. The connection is established by pressing the “Reset app” button. After entering an arbitrary amount of seconds, the scan process can be started and all data will be sent over the socket.

8.2.2 Software Architecture

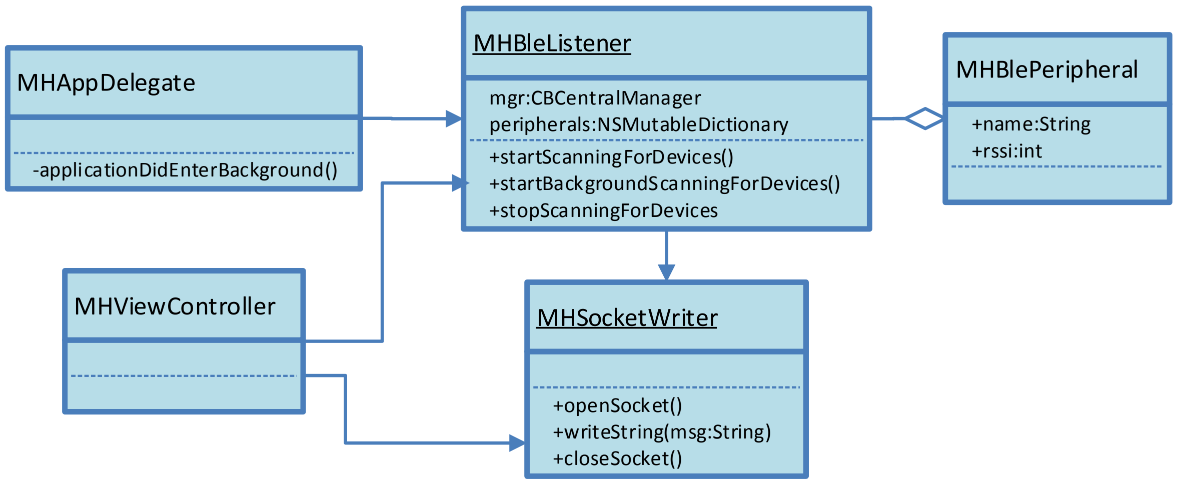
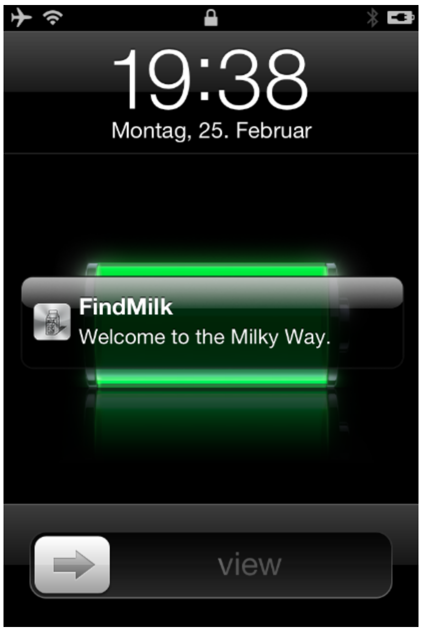


Figure 8-B: iOS App UML diagram

Figure 8-B shows the basic class structure of the iOS application. The class MHViewController is responsible for handling user interaction and messages the MHSocketWriter for all socket based operations like opening and closing a connection. It is also responsible for calling the MHBleListener to start and stop scans. The MHBleListener class is a delegate that is responsible to handle all messages it receives from the CBCentralManager class. The CBCentralManager class is provided by the iOS CoreBluetooth framework and allows scanning and managing Bluetooth Low Energy devices. There is only one exception where the MHBleListener receives a message from the MHAppDelegate. This is as soon as the app enters the background. This can happen if the user presses the home button or turns off the display. In this case, the MHBleListener performs a reconnect to all registered BLE devices to work around one of the iOS anomalies explained in Section 4.4.2. This ensures that further notifications for devices will be received in the background even if they had already been discovered while the app was active.

The CBCentralManager's method scanForDevicesWithServices:options: is provided with the UUID for the FindMilk service. This allows scanning in the background and allows us to only detect BLE devices that have been set up for the purpose of indoor positioning. Duplicate advertisers are not discovered more than once by default. This has to be enabled by passing the correct setting to the method first. When a device is detected, the delegate, the MHBleListener class in this case, will be called. In case the delegate has been called with new RSSI data, it will promptly write this data to the socket.

In order to allow background operation, an entry has to be created in the .plist configuration file. The entry UIBackgroundModes should then contain the value "Bluetooth-central". When the app is residing in the background while the iOS device receives an advertising packet, a notification is created and presented to the user. Tapping the notification will open the app. This feature could be used to notify the user as soon as he enters a building. He could then use the app to find his way around the premises.



FindMilk
Welcome to the Milky Way.

Figure 8-C: Background notifications

The BLE listener does not only call the CBCentralManager to start scanning for devices but also starts a timer at an interval of 10 seconds. This timer ensures that the scanning method is called every 10 seconds to prevent iOS from reverting to a lower scan interval.

8.3 The Air Application

8.3.1 Usage

All analysis has been done on a computer. The main menu allows access to all test modes. The first button takes the user to a simple full screen text view that can be used to output debugging instructions and socket communication. The second and third button takes the user to test scenarios where measurements can be made and evaluated. A simple game has been implemented to test the algorithm. It is accessible under FindMilk game. The most relevant parts of the workflow can be seen in the flow diagram below.

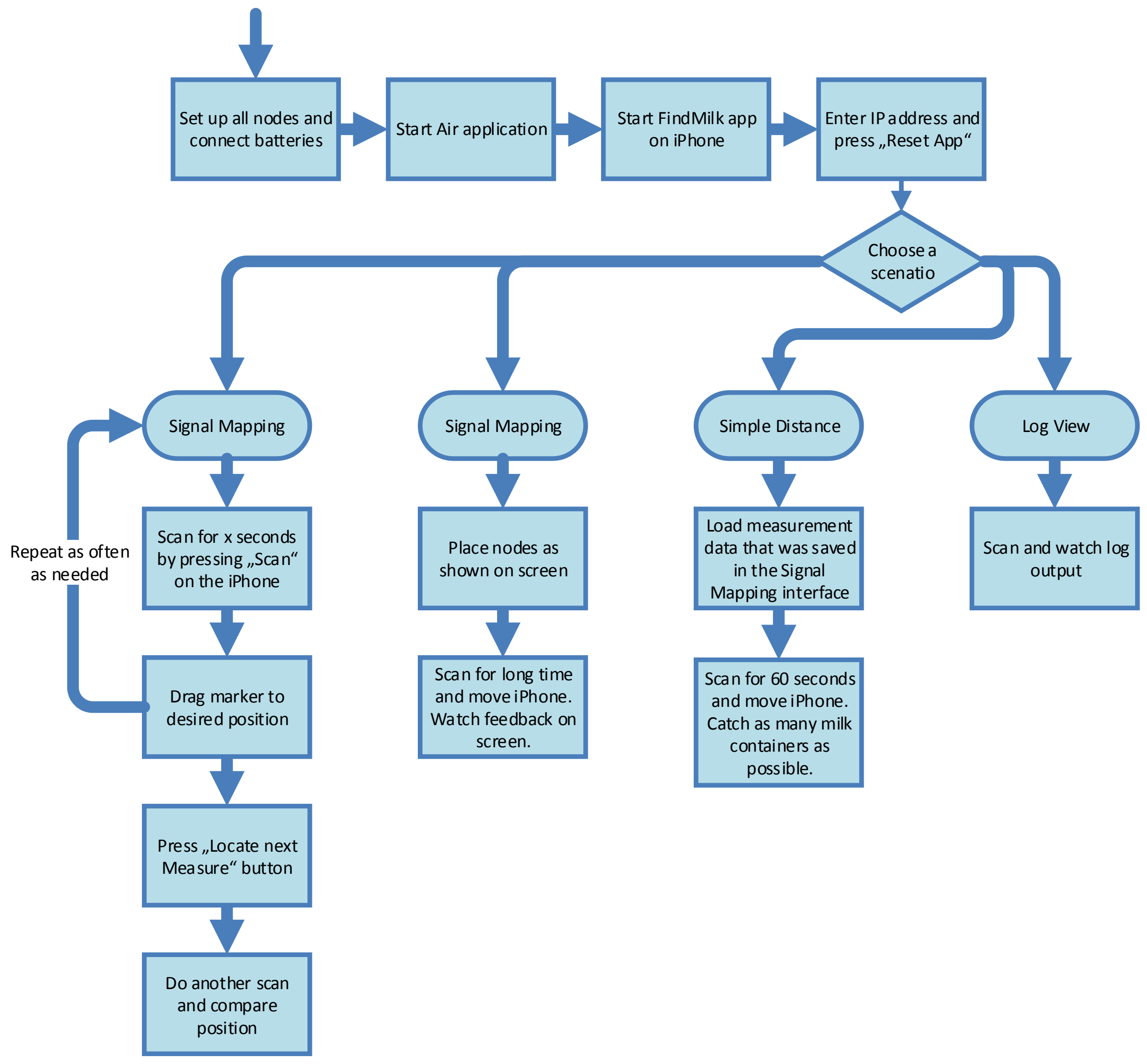

Set up all nodes and connect batteries
Start Air application
Start FindMilk app on iPhone
Enter IP address and press „Reset App“
Choose a scenatio
Signal Mapping
Scan for x seconds by pressing „Scan“ on the iPhone
Drag marker to desired position
Press „Locate next Measure“ button
Do another scan and compare position
Repeat as often as needed
Signal Mapping
Place nodes as shown on screen
Scan for long time and move iPhone. Watch feedback on screen.
Simple Distance
Load measurement data that was saved in the Signal Mapping interface
Scan for 60 seconds and move iPhone. Catch as many milk containers as possible.
Log View
Scan and watch log output

Figure 8-D: Flow diagram of program usage

8.3.1.1 Simple Distance

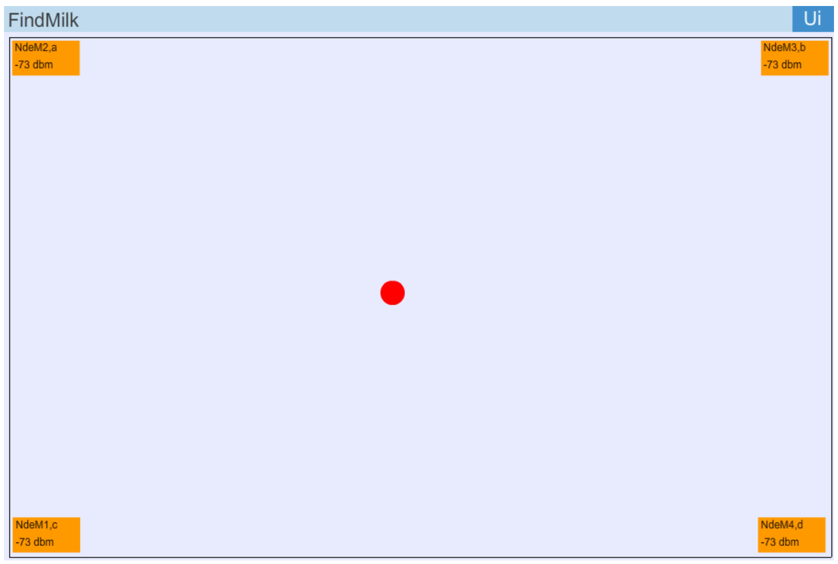


Figure 8-E: Simple Distance screen

After the socket communication was intact and first measurements started to come in, a simple percentage based position approximation was programmed. All four nodes have to be laid out in the correct order as presented on screen. The node RSSIs were then averaged in x and y direction independently and a red dot was used to represent the user’s position. This had not provided meaningful results. But it can be used to gain a basic understanding by shielding the antenna and watching the stability of the measurements. It does not provide correct results because the RSSI values are dependent on antenna direction and do not relate to the actual position in a linear manner.

8.3.1.2 Signal Mapping

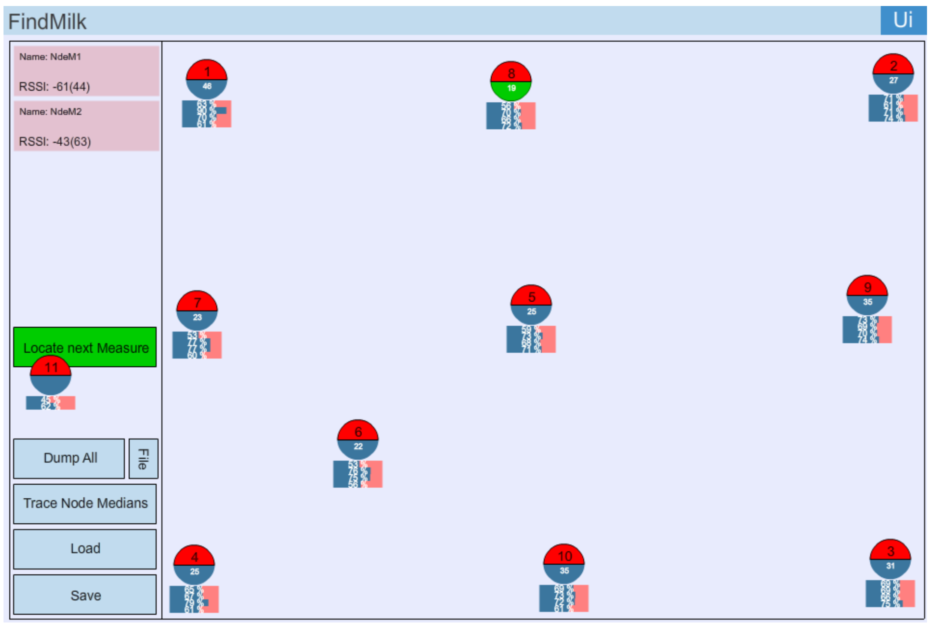

Name: NdeM1
RSSI: -61(44)
Name: NdeM2
RSSI: -43(63)

Figure 8-F: Signal Mapping screen

In order to have the means to collect all measurements and analyse them, the Signal Mapping interface was created. It collects all RSSI values, groups them according to their node names and displays them gathered as a measurement. If the iPhone client is set to a five second scan time, all these values will be gathered in one measurement. Each measurement is numbered and can be moved on screen to represent the physical measurement location. An indicator for every Advertiser is shown with 100% being the best reception and 0% being the worst. The indicators are sorted by node names. All RSSI values can be viewed by clicking at the lower half of a measurement marker which will show a text field with all values.

The left side provides an overview of all nodes that have been found, including the last received RSSI and an average in brackets. After the "Locate next Measure" button has been pressed, any further measurement will be shown on the left side and the distance to all of the previously recorded measurements will be calculated. The lowest distance equals the highest resemblance and will be coloured green. The algorithm that was used is very simple and only calculates the similarity by comparing the sample with the calibration measurements. The distance value is shown in the lower half of the measurement circles.

All RSSI values can be exported to the console for further analyses using different software. This is done by pressing "Dump All" whereas pressing "Trace Node Medians" will just print the RSSI median for every Advertiser in every measurement. All values can be written in a tab separated structure to a file, this is achieved by using the "File" button. All measurements can be exported and loaded as well in JSON format with "Load" and "Save".

8.3.1.3 FindMilk Game

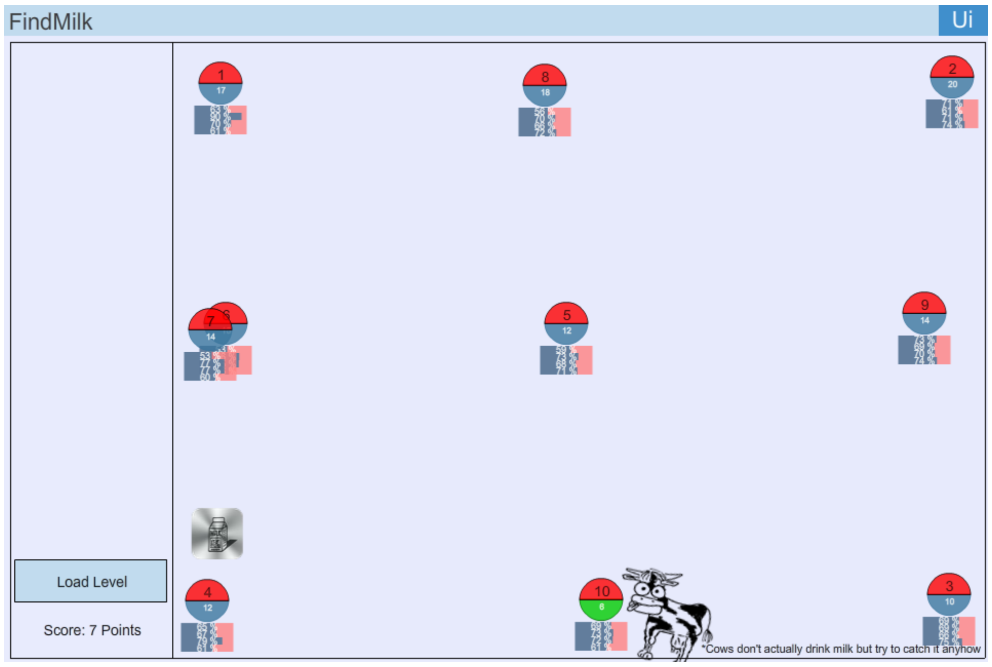


Figure 8-G: FindMilk Game

More as an afterthought to explain the name of the application, a game has been implemented. A level has to be loaded first by pressing “Load Level”. This level can be created by using the Signal Mapping interface. A 60 second scan should then be started and it is then possible to move the cow. The goal is to collect the most milk containers in a time of 60 seconds. There is no interpolation done to determine a position between two points so the player has to think of an appropriate path to catch a milk container that is in between two measurements.

8.3.2 Software Architecture

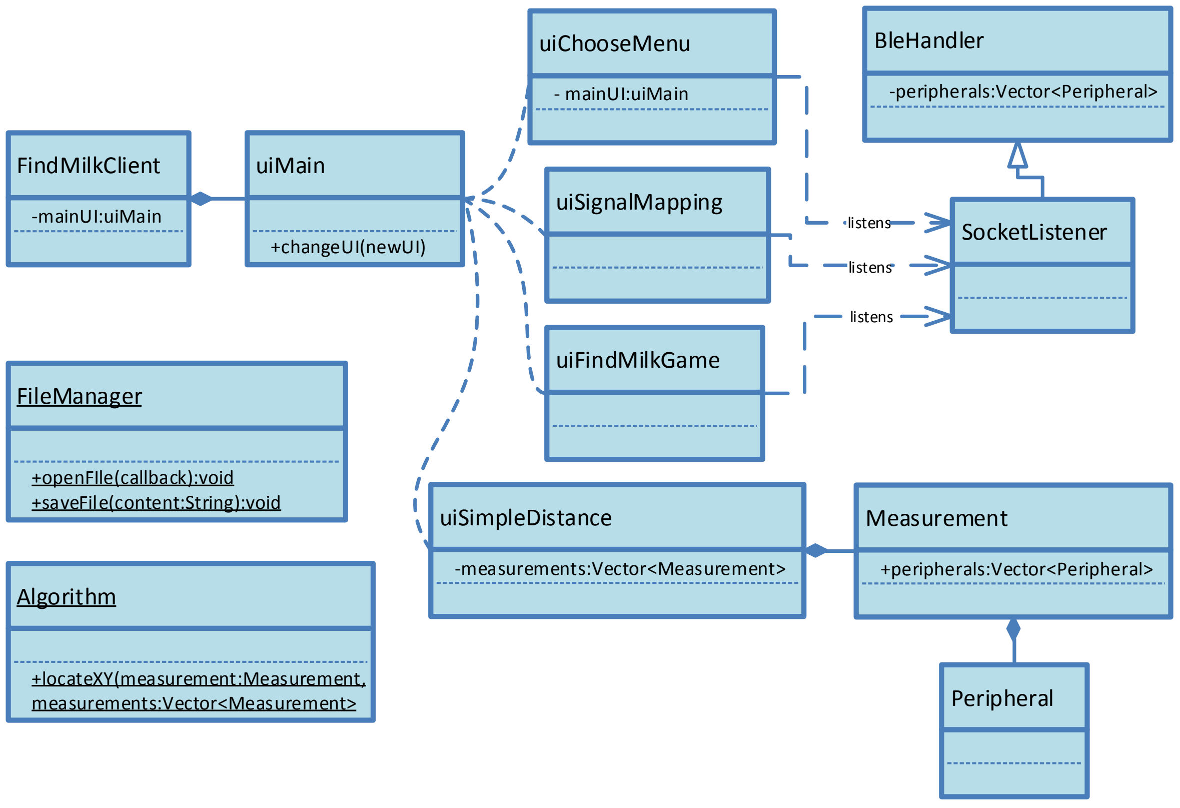


Figure 8-H: Simplified UML diagram for the Air part

The software has been designed in a class based OOP manner with the future use as an Air native extension in mind (ANE). ActionScript 3 does not rely on delegates like Objective-C, but provides a different design pattern that works with event listeners. A class can broadcast events that will then be received by every event listener attached to this class. The BleHandler class which can be seen in the upper right of Figure 8-H is able to broadcast RSSI and PeripheralFound events. The SocketListener extends on this functionality and broadcasts the same events as soon as it receives the corresponding commands from the connected socket. Interested classes like the uiSignalMapping class in the above diagram receive the event and can then read the included data that the event provides. In case of an RSSI event, this is a received RSSI value and the corresponding node name. When building an ANE it is now possible to simply replace the SocketListener class with a class that receives the RSSI values from its native extension and then uses the method of its subclass to broadcast the same events.

The main class is FindMilkClient. It holds a reference to the class that handles the user interface. This class in turn provides a method to change the user interface that is shown on stage. All the other classes prefixed with “ui” extend an asset that was created within Flash Professional and enliven it with functionality. All these classes have a data management of their own if they need one at all. The uiSimpleDistance class for example holds a Vector of Measurements which in turn each hold a reference to all the Peripherals that have been found during that measurement. Each of them has an RSSI log that is stored in a ring buffer.

The classes FileManager and Algorithms are purely used in a static manner. The first provides methods to open and save a String to a file while the latter cumulates the algorithms used for position calculation at a convenient place.

Many classes provide serialization and deserialization to and from JavaScript object notation (JSON). This is used to allow loading and saving of test scenarios. Some variables have been made public in order to allow easy serialization. Some classes can return a string representation of their data. This is used to output measurement data to the console which can then be analysed in applications such as Microsoft Excel or Wolfram Mathematica.

9 Advanced Implementation Possibilities

There are virtually endless possibilities that can be implemented. Some more thoughts have been gathered and are presented in this chapter.

Not only is it possible to enable indoor positioning with Bluetooth 4.0 but also tracking. If the client uses an internet enabled smartphone, the current position can be uploaded in intervals. Another method is to configure the smartphone to send out advertisement packets at a specific interval. These could be picked up periodically by the same nodes used for the navigation system itself or by dedicated nodes that would have to be installed. This information could then be sent using a peer-to-peer approach with an algorithm to calculate a path between all nodes and one special node that is connected to a host where all the information would be evaluated. Among some rather uncomfortable uses, this could be used to generate heat maps from customer movement and would enable shop owners to rethink the shop layout based on the data acquired.

The nodes could be set up to broadcast battery information once a day. The mobile client could be set up to upload that information to a server. A maintenance worker could then be notified of the nodes’ name and position as soon as the battery is low.

The most successful approach is probably an implementation targeted at a particular shop or building. This has many reasons. It is not always possible to get indoor maps for any building. Creating them can only be done with a lot of effort and using existing ones might even present copyright problems. The owner of a building will however have access to those maps and even more important to the data that provides added value for navigation. In a shop this would be the places where certain products are positioned. For a fair, this data would consist of all the exhibitors and their products. And for an office building it would make all rooms searchable for persons and functions. Making this data searchable and visible on the map would provide the best benefit for both user and provider. Because the Bluetooth technology is already available on the consumer side it is just a matter of finding a customer that wants to offer indoor navigation for his clients. After the building has been outfitted with the necessary technology, an app has to be created to supply the customer with all the added benefits.

Another possibility is to outfit shopping carts with a small BLE beacon. This enables tracking of all customers with ease and could provide all the necessary data needed for an optimal shop layout.

These are just some of the possibilities. There are a lot more benefits to an indoor navigation system and much more value adding and position dependant data that could be provided. But these are well above the limits and goals of this work which was just intended to evaluate the basics.

10 Conclusion

Coming back to the initial question: Is Bluetooth 4.0 suitable to be used for indoor positioning? Yes it is, but no, the implementation is not easy. While a simple implementation can provide indoor positioning with the accuracy of few metres at low cost, it is a lot more complex to design a solution that provides better precision in every environment. This work presented the basics of Bluetooth 4.0 and a lot of testing was undertaken to show its behaviour. A simple prototype was implemented to show what is possible with little effort and solutions were presented on how to refine the algorithm to build a robust solution that can be used in real world scenarios.

It will not be much longer until indoor navigation finds its way into many public buildings, airports, fairs and more. The benefits for both consumers and providers are much too big to be ignored. With the low cost of Bluetooth 4.0 devices it is easy to enable a better accuracy as Wi-Fi can provide nowadays while still keeping costs down. But it remains to be seen if future systems will be based on a single standard or if they will make use of many methods merged in some way to enable a seamless user experience.

Bibliography

1. **Smith, Phill.** *Comparisons between Low Power Wireless Technologies.* s.l. : CSR, 2011. CS-213199-AN.

2. **Bluetooth SIG.** Bluetooth Low Energy Technology, How it works. [Online] 18 05 2011. http://www.bluegiga.com/files/bluegiga/LE%20Public/LowEnergy_HowItWorks.pdf.

3. **Heydon, Robin.** *Bluetooth Low Energy: The Developer's Handbook.* s.l. : Prentice Hall, 2012. 978-0132888363.

4. **Apple Inc.** *Bluetooth Accessory Design Guidelines for Apple Products R6.* 2012.

5. **Bluegiga.** *BLE112 Data Sheet Version 1.11.* 2011.

6. **Flock, Warren L.** *Propagation Effects on Satellite Systems at Frequencies Below 10 GHz, A Handbook for Satellite Systems Design (Second Edition).* 1987. NASA Reference Publication 1108(02).

7. **Bluetooth SIG.** *Bluetooth Specification Version 4.0.* 2010.

8. **Bluegiga.** *BGScript Scripting Language Developer Guide Version 2.4.* 2012.

9. **Carles Gomez, Joaquim Oller and Josep Paradells.** *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology.* 2012. ISSN 1424-8220.

10. **Bluegiga.** *Developing Bluetooth 4.0 single mode applications Version 1.5.* 2011.

11. —. Bluetooth Smart Getting Started Version 1.5. 2012.

12. —. *Bluetooth 4.0 (BLE) Single Mode Stack Api Documentation Version 1.3.* 2012.

13. **Texas Instruments.** *Texas Instruments CC2540/41 Bluetooth Low Energy Software Developer's Guide v1.2.* 2012.

14. **Bluegiga.** *Executing BGscript on BLED112.*

15. **Bluetooth Smart.** How to Choose a Bluetooth Smart / Low Energy Development Kit. [Online] 05 10 2011. http://blog.bluetooth-smart.com/2011/10/05/bluetooth-low-energy-development-kits-2/.

16. **Texas Instruments.** LPRF San Diego Bluetooth Low Energy Deep Dive Presentation. [Online] 10 04 2012. http://e2e.ti.com/support/low_power_rf/m/videos__files/653593.aspx.

List of Abbreviations

AD structure: Advertising structure

ADFH: Adaptive Frequency Hopping

ANE: Air native Extension

API: Application Programming Interface

ATT: Attribute Protocol

BL+HS: Bluetooth with High Speed

BL/EDR: Bluetooth with Enhanced Data Rate

BLE: Bluetooth Low Energy

CRC: Cyclic Redundancy Check

DFU: Direct Firmware Update

GAP: General Access Profile

GATT: General Attribute Profile

GPS: Global Positioning System

HCI: Host Controller Interface

L2CAP: Logical Link Control and Adaption Protocol

MIC: Message Integrity Check

NFC: Near Field Communication

PDU: Packet Data Unit

RFU: Reserved for Future Use

RSSI: Received Signal Strength Index

SDIO: Secure Digital Input Output

UART: Universal Asynchronous Receiver / Transmitter

UUID: Universally Unique Identifier

List of Illustrations

Code Listings