NTNU – Trondheim
Norwegian University of
Science and Technology

Context-awareness for Mobile Ticketing Systems

Øyvind Høisæther
Eirik Vigeland

Master of Science in Computer Science
Submission date: June 2014
Supervisor: Asbjørn Thomassen, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Eirik Vigeland, Øyvind Høisæther

Context-awareness for Mobile Ticketing Systems

Master Thesis, spring 2014

Software Engineering
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

One of the most pervasive technological devices that exist is the mobile smartphone. With the emergence and embedding of multiple sensors and connection possibilities found in today's phone, the boundaries of what we can accomplish are nearing a limitless horizon.

This thesis looks at how we can advance an application of a smart device to achieve another degree of functionality. By exploring context-awareness, its approaches and practical uses, we wish to enhance the User Experience (UX) of such an application. To do this, we survey and explore context, context-awareness, the roots of how it came about, and what context is considered to be.

We focus on a case-study of mobile ticketing systems, and review the current situation of these in Norway. By providing a foundation of approaches and technology that can benefit context-aware applications, we confirm our research questions with a Proof of Concept (POC) application that take advantage of a certain level of context-awareness. With the help of Bluetooth Low Energy (BLE) beacons, we show how to accomplish validation of tickets in our system to provide a seamless process for users and event organizers. We end by verifying our hypotheses that context-awareness can give a better UX by accomplishing an above average score in a System Usability Scale (SUS) test.

Preface

This master thesis is written by Eirik Vigeland and Øyvind Høisæther at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway throughout the spring semester of 2014. We are writing in collaboration with WTW AS, and under the supervision of assistant professor Asbjørn Thomassen at NTNU. With this, we conclude our master degree in Software Engineering at the Department of Computer, and Information Science (IDI), and the Faculty of Information Technology, Mathematics, and Electrical Engineering (IME).

We would like to express our gratitude and appreciation to Asbjørn Thomassen for valuable advice, encouragement, and enthusiasm in guiding us through the process of writing. We also want to thank Christian Aune Thomassen and WTW for contributing with ideas, technical advice, and means to experiment with, and lastly, to our friends and families.

Thank you!

Eirik Vigeland, Øyvind Høisæther

Trondheim, June 11, 2014

Contents

List of Figures

List of Tables

Acronyms

ACK Acknowledgement. 52

AD Advertising Data. 93, 97–103

AI Artificial Intelligence. 39, 44, 50

AIR Activity-Based Information Retrieval. 41, 47

AKT Agder Kollektivtrafikk. 15

API Application Programming Interface. 55, 63, 64, 72, 81, 88, 92, 94, 109, 134, 138, 145, 152

AR Augmented Reality. 62

ATT Attribute Protocol. 76

BCBP Bar-coded Boarding Pass. 22

BLE Bluetooth Low Energy. 24, 73–76, 79, 85, 92–95, 107, 108, 119, 125, 129, 131, 132, 151, 152, 156, 158

BR Basic Rate. 76

CRC Cycle Redundancy Check. 97

dBi Decibels-isotropic. 150

dBm Decibel-milliwatts. 104, 150, 151

DC Direct Current. 96

DHCP Dynamic Host Configuration Protocol. 52, 139

DNS Domain Name System. 51, 52

EAN International Article Number. 21

EDR Enhanced Data Rate. 76

ER Entity-Relationship. 43, 146

FFK Finnmark Fylkeskommune. 14, 24

GAP Generic Access Profile. 76, 97–100

GATT General Attribute Profile. 76

GPS Global Positioning System. 34, 40, 50, 61, 62, 68, 81, 83, 89, 90, 107, 110, 119, 122, 152

GUI Graphical User Interface. 49, 50, 80

HCI Human-Computer Interaction. 30, 56, 62

HF High Frequency. 69, 70

I/O Input and output. 110

IATA International Air Transport Association. 22

IEC International Electrotechnical Commission. 20

IoT The Internet of Things. 64, 74

IR Infrared. 72, 76, 93

ISM Industrial, Scientific, and Medical. 73

ISO International Organization for Standardization. 20

JSA Japanese Standards Association. 20

L2CAP Logical Link Control and Adaptation Protocol. 96, 97

LBS Location Based Services. 61

LE Low Energy. 76, 96, 100

LED Light-Emitting Diode. 49

LF Low Frequency. 69

LSB Least Significant Bit. 96

MAC Media Access Control. 156

MIC Message Integrity Check. 97

NACK Negative-Acknowledgment. 52

NFC Near Field Communication. 24, 59, 70, 71, 93, 119

NSB Norges Statsbaner. 14, 25

PDU Protocol Data Unit. 96, 97

PHY Physical Layer. 96, 97

POC Proof of Concept. 1, 3, 82, 88, 89, 92, 93, 101–104, 106, 111, 112, 114, 115, 119–122, 124, 146, 149, 168, 169

POI Points of Interest. 79, 122, 124

POS Point of Sale. 10, 25, 80

PwAD Persons with Alzheimer's Disease. 61, 123

QA Quality Assurance. 89

QR Quick Response. 10, 22–24, 52, 59, 60, 79, 86, 87, 90, 93, 106, 108, 109, 112, 116, 119, 120, 124

REST Representational State Transfer. 94, 135, 145

RFID Radio Frequency Identification. 52, 59, 61, 64, 69, 70, 119

RS Received Signal. 150

RSSI Received Signal Strength Indicator. 102–104, 122, 123, 149, 150, 152, 156, 159, 163–166, 168, 169

SAS Scandinavian Airlines System. 15, 18, 24

SDK Software Development Kit. 55, 72, 90, 110

SIG Special Interest Group. 73

SOA Service-Oriented Architecture. 63

SUS System Usability Scale. 4, 111–114, 120, 121

TA Travel Assistant. 16, 18, 24

TCP Transmission Control Protocol. 52

UHF Ultra-High Frequency. 70

UML Unified Modeling Language. 43

UPC Universal Product Code. 21

UUID Universally Unique Identifier. 98, 100, 101, 131

UWB Ultra-Wide Band. 70

UX User Experience. 55, 89, 119, 121, 124, 125

WPAN Wireless Personal Area Network. 73

WSN Wireless Sensor Network. 103

XML Extensible Markup Language. 43, 60, 89

1

Introduction

"The way to get started is to quit talking and begin doing."

– Walt Disney , *co-founder of The Walt Disney Company*

In 1859 Charles Darwin published the book *On the Origin of Species* where he explained that the species who survive are not the ones that are strongest or most intelligent, but those that respond to change. Being aware of the environment, and adapting to it, is not only central for humans, but has in recent years become a major part of software applications. Adding the concept of context-awareness to applications opens up a world of possibilities for both users and developers as the application can be perceived as *smarter* by being attentive and responsive to the world outside of its confinement.

Our work will start by looking at mobile ticketing systems in Norway. Having a mobile ticketing application for events can be beneficial for both attendees and the event organizers, and might help the event save money, time, and effort. We will investigate whether or not current systems use any form of context-awareness, and limitations in general. Also, we will attempt to convey the meaning of context and the term *context-awareness*, how they are normally used and how they could be utilized in a mobile ticketing system for events.

Context-awareness, as a part of ubiquitous- and pervasive- computing, has had a large impact on distributed- and mobile- computer systems, and with this the emergence of many interesting technologies and approaches, which will be discussed in this thesis.

We shall also describe a POC for ticket validation using theory, practices, and technology, that we use to approach context-awareness.

1.1 Background and motivation

This project is written in collaboration with WTW AS, a company based in Trondheim, Norway. WTW is one of the leading companies in mobile ticketing in Norway, which mainly focuses on public transportation and parking services, in both cases allowing the users to pay for a ticket using their smartphone.

Throughout the fall of 2013, we, the authors of this thesis, were working on our *specialization project*, also in collaboration with WTW, which focused on how Neo4j compares with MySQL when implementing a simple recommender system for events. Our focus has since shifted and we are now looking into how mobile ticketing systems can be improved with context-awareness, especially for events. Context-awareness is an exciting, emerging, and a future-oriented way of working with software. In everyday life, smartphones and computers are found everywhere and the possibilities of sensing with such devices are vast. In research, context-awareness and pervasive technology is still considered to be somewhat in its infant stage, which makes it both a challenging and interesting subject to look into. To date, there exist no de facto standard for doing or using context-awareness; technology, definitions, and approaches exist in many *shapes and colors* and are therefore highly debatable.

We knew before starting this thesis that there are a limited number of mobile ticketing applications for events in Norway, but unlimited possibilities (so to speak) to create the next-generation ticketing system for events, or at least the foundation of one.

1.2 Goals and research questions

Goal *Explore context-awareness, its approaches and practical uses for mobile ticketing systems.*

Research question 1 *Review mobile ticketing in Norway; are there any advantages and disadvantages?*

Research question 2 *How has context-aware computer systems developed through history up until now?*

Research question 3 *What approaches, technology and practices can be used for context-awareness in mobile devices?*

Research question 4 *How can context-awareness improve the mobile ticketing experience?*

1.3 Research method

Our approach for this thesis is based on literary, analytic, and experimental research. We must first of all provide a theoretical foundation. We should do this by:

- Investigate the current market of commercial mobile ticketing systems.
- Survey and study context-aware research.
- Explore present technological practices and possibilities to implement context-awareness for smartphones.
- Validate context-aware practices with a POC of selected approaches for mobile ticketing.

1.4 Contributions

We will briefly summarize the contributions that this thesis can be expected to provide. The contributions will be further evaluated in chapter 9.

1. A thorough survey of the field of context-awareness, featuring its origin, development, and approaches.
2. An attempt to classify context on the basis of the combined effort of research articles, papers, and books that we used for our survey of context-awareness. We will see that the classification will build on previous attempts and provide additions to these.
3. A guide to approach context-awareness for mobile devices, on the background of the sensors they incorporate and the smart spaces available for them to sense.
4. The implementation and evaluation of a POC to demonstrate the ease and seamlessness that context-awareness can provide to smartphones, in contrast to other solutions.

1.5 Thesis structure

In the next chapter, we will present a concept for a mobile ticketing system that we have partly based our work on. This includes mockups and textual descriptions of a product design concept created for WTW by an industrial design master's degree student.

In chapter three, we review mobile ticketing systems currently used in Norway, in the transportation- and event- categories. We also describe the various methods of validation used, as well as how tickets can be purchased. At the end of the chapter, we present and discuss the limitations of current ticketing systems, and try to convey the benefits of having a mobile ticketing system, as opposed to a paper-based one.

Further on, in chapter four, we explore context-awareness, its roots and development up until now. We will look at which branches of computer science that the field originated from, and discuss general ways of handling, modeling, acquiring, and adapting to context. At the end, we will conclude the chapter by looking at the research challenges that exist.

Chapter five starts by explaining the different approaches to achieving context-awareness, followed by the technologies that are available to us through modern devices.

In chapter six, we describe a prototype of the ticketing application, *Pling*, in which we take advantage of the context-aware approaches and technologies discussed in the previous chapter. We will explain its capabilities through screenshots and user walkthroughs, as well as portraying a future vision of what *Pling* has the potential to become.

Chapter seven dives into the implementation-specific details of the prototype, by explaining how the context-aware features are implemented and built.

In chapter eight, we present the SUS, what it was originally used for, how it works, and the scores from our user testing.

Chapter nine contains a thorough discussion of our work, our contributions, and the future work that can be conducted.

2

Original concept

> *"We choose to go to the moon in this decade and do the other things. Not because they are easy, but because they are hard."*
>
> – John F. Kennedy , *35th President of the United States*

In this chapter, we begin by explaining the original concept that we, to some extent, base our work on.

2.1 Concept and mockups

In 2012, an industrial design master's degree student created a product design concept for WTW which allows people to purchase tickets to events, as well as products offered at the event (e.g. food and beverages), by using their smartphones. The concept focused on sporting events such as soccer and handball, but should also be suitable for other events, i.e. concerts.

This section contains selected mockups, to help the reader gain a better understanding of the original concept, and how the system was intended to work.

From the event attendee's perspective:

- Browsing events and purchasing tickets.
- Purchasing at-event items, e.g. food and beverages.

From the event organizer's perspective:

- Validation of tickets when users arrive at the event.
- Validation of purchases of at-event items.

The text within the mockups is written in Norwegian, but the captions and images themselves should enable the reader to grasp the core idea of the system.

2.1.1 User perspective

All mockups in this section originate from the original concept.

Figure 2.1: The user has just started the app, content is loading.

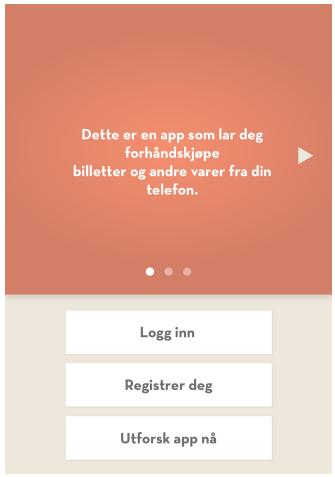

Figure 2.2: Main menu, the user needs to register or log in.

Once the app has loaded, as in Figure 2.2, the user is looking at the main menu where he or she has the options to log in, register, or just explore the app without being logged in. However, if the user registers, then he or she will be automatically logged in the next time the app is launched.

Once logged in, or the user selects to explore the app without being logged in, he or she will be presented with a list of upcoming events.

Figure 2.3: The user is browsing the events in the near future, and selects one of them.

Figure 2.4: The user chooses to purchase tickets.

Figure 2.5: The user has selected to purchase 2 tickets for adults.

The following screenshots illustrate how the user can have his or her ticket validated by an event organizer, after having confirmed the purchase of tickets.

Figure 2.6: The user has the receipt of the ticket available in the app.

Figure 2.7: The user has clicked on the QR-code to enlarge it prior to validation.

Figure 2.8: The ticket has been validated, and now marked as used.

Once the ticket has been purchased, and the user has had his or her ticket validated by the event organizers, it's possible for the user to view and purchase at-event items, such as food and beverages.

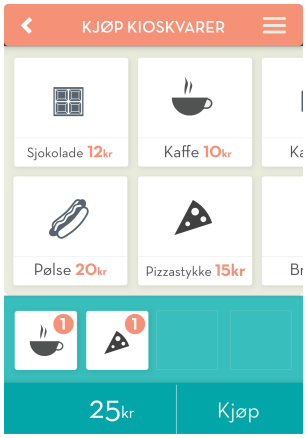

Figure 2.9: The user wants to purchase one coffee, and one slice of pizza.

Figure 2.10: The user has confirmed the purchase, and a receipt with scannable QR-code is displayed.

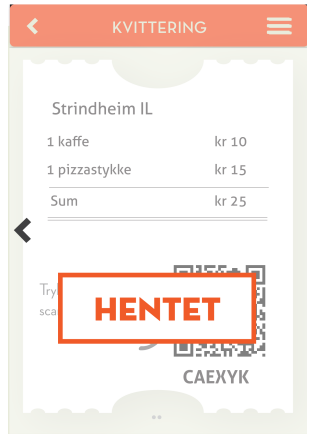

Figure 2.11: The user has had his QR-code scanned at the POS. The order is marked as completed.

There are several other aspects of the system which are not covered in this section, such as how the user logs in, adds credit-cards, creates a mobile account, and how he or she manages the settings of the app. We only covered the key aspects of how the system is to be used and what it aims to achieve.

2.1.2 Event organizer perspective

All mockups in this section originate from the original concept.

Upon the arrival of an attendee at the event, the organizer will have to validate ticket(s) of the attendee. This is done by having the attendee place his or her mobile screen (with the Quick Response (QR)-code of his ticket enlarged) in front of the camera of the tablet used by the organizer. The app will read the QR-code, and confirm the validity of the ticket.

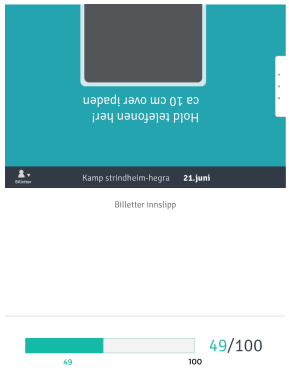

Figure 2.12: The organizer's mobile app is ready to scan tickets.

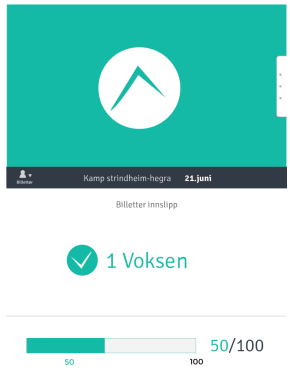

Figure 2.13: Attendee has placed his mobile phone displaying the ticket QR-code in front of the camera of the tablet.

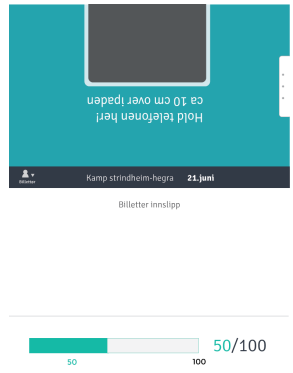

Figure 2.14: Once validated, the app is ready to continue validating new tickets with updated attendee count.

At-event purchases are validated in the same fashion by having the attendee enlarge the QR-code of the receipt he received once the purchase was confirmed. The mobile account of the attendee will not be debited until he or she has appeared at the Point of Sale (POS) and had his or her receipt scanned. Once scanned, and purchase is confirmed by the POS cashier, the transaction is complete - and the mobile account will be debited.

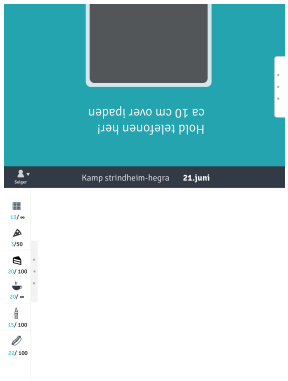

Figure 2.15: The organizer's mobile app is ready to scan receipts.

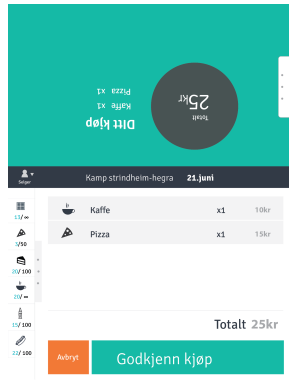

Figure 2.16: Attendee has placed his mobile phone displaying the receipt QR-code in front of the camera of the tablet. The cashier needs to confirm the purchase or cancel it.

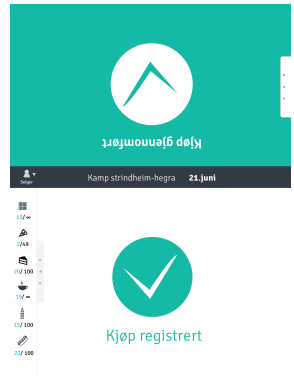

Figure 2.17: Once the purchase has been validated, and confirmed, the attendee is given his ordered food and beverages.

There are many more aspects to the system than covered in this section, such as, but not limited to:

- Managing the event.

  Adding/Removing merchandise from the inventory.

- Creating/deleting an event.
- Altering an event.

3

Current ticketing systems

"Computing is not about computers any more. It is about living."

– Nicholas Negroponte, *co-founder MIT Media Labs*

This chapter will take a closer look at the mobile ticketing systems which are currently, per February of 2014, in use in Norway, the way users can buy tickets, how tickets are validated, and the limitations that exist, if any.

3.1 Overview of mobile ticketing systems

This section will provide an overview of the mobile ticketing systems we have investigated, mostly focusing on the Norwegian market, as systems used abroad might be applicable to other privacy laws and regulations. Another reason is the fact that Norway is ranked among the top 10 countries regarding smartphone penetration in several rankings [1, 2].

3.1.1 Public transportation

AtB Mobillett
AtB is the management company for public transport in Sør-Trøndelag, Norway, and had 21.4 million passengers in 2012. In November 2011, AtB started selling tickets via their mobile application *AtB Mobillett* for Android and iPhone,

which at the end of 2012 stood for almost 30% of ticket revenues in Trondheim, accounting for 55% of single ticket sales [3].

Skyss Billett
Skyss is a public authority, owned by Hordaland county council, which administers the public transport (trains, buses, ferries, and express ferries) in Hordaland county. The passenger growth has been steady in recent years, and in 2012 the total number of passenger-boarding was 47.2 million. On January 23rd 2013, Skyss launched *Skyss Billett*, a mobile ticketing application for Android and iPhone which allows passengers to purchase single tickets (EnkeltsSkyss) within the Bergen zone [4].

FFK Mobillett
Finnmark Fylkeskommune (FFK) administers the public transport in Vadsø, Kirkenes, Honningsvåg, Hammerfest, and Alta. On January 2nd 2014, FFK launched *FFK Mobillett*, a mobile ticketing application which allows passengers to purchase single tickets on both Android and iPhone devices [5].

Kolumbus Billett
Kolumbus is a county management company for public transport in Rogaland county and has about 65 000 daily passengers [6]. There were 18.5 million trips by bus in Sør-Rogaland in 2012, while the number of trips in northern Sør-Rogaland declined by 0.4% compared to 2011. In total, the number of trips increased by 4.1% [7] for the whole county. On the 19th of December 2013, Kolumbus launched a mobile ticketing application for Android and iPhone.

RuterBillett
Ruter AS is a management company for public transport in Oslo and Akershus, owned by the city of Oslo (60%) and Akershus Fylkeskommune (40%) [8], and had in total 301 million boarding in 2012 [9]. On the 17$^{\text{th}}$ of December 2012, Ruter launched their mobile ticketing application *Ruterbillett* for Android, iPhone, and Windows Phone. Ruterbillett allows users to purchase single tickets for public transport. In 2013, an updated version of the app allowed for purchase of periodic/seasonal tickets [10].

NSB
Norges Statsbaner (NSB), owned by the Ministry of Transport, is the largest railroad actor in Norway. In November 2011, NSB launched their mobile application NSB Billett which allowed users to purchase train tickets [11]. In the annual report for 2012, NSB reported to have increased its number of train passengers by 2.6% to 53.8 million from 2011. NSB's bus operations reported to have transported 126 million travelers, an increase by 5.6% from 2011 [12].

AKT Mobilbillett
Agder Kollektivtrafikk (AKT) is the administration company for public transport in Agder. AKT is owned by Vest-Agder fylkeskommune (40%), Aust-Agder fylkeskommune (40%), and Kristiansand kommune (20%) [13]. In November 2012, AKT launched their mobile application *AKT Mobilbillett*, which allowed users to purchase bus tickets in the Kristiansand area [14].

Flybussen
Flybussen is a collaboration between Scandinavian Airlines System (SAS) and bus companies to transport passengers to and from airports throughout Norway. In 2013, Flybussen launched an app which allowed passengers to pay for their bus fare to and from the airport in both Oslo [15] and Bergen [16].

Kystbussen
Kystbussen (*the coast bus* in Norwegian) is part of NOR-WAY express buses, a leading company in the express bus service in Norway. NOR-WAY express buses operate nationwide routes in the southern part of Norway with 3.5 million passengers annually. In 2013, Kystbussen accounted for almost 460 thousand passengers on its route from Stavanger, via Haugesund, to Bergen [17, 18]. As well as providing ticket services online and with SMS, Kystbussen launched a mobile application in June 2012 [19].

SAS Scandinavian Airlines
SAS is a multinational corporation operating in Norway, Sweden, and Denmark. Its major shareholders are the Swedish government (21.4%), Danish government (14,3%), and Norwegian government (14,3%) [20].

Additional facts about SAS, according to their corporate presentation [21]:

- SAS has 136 destinations, covering all continents.
- 28 million passengers annually.
- 45 new routes in 2013.
- 30-50% market share in home markets.

In 2013, SAS launched a mobile application which allowed users to purchase airplane tickets. Although SAS already had mobile applications in App Store / Google Play, these versions did not support ticket purchasing [22].

Norwegian Travel Assistant
Norwegian is the second largest airline in Scandinavia and the third largest low-cost airline in Europe with around 3500 employees.

Quick facts about Norwegian, according to [23]:

- 416 routes to 126 destinations in Europe, North-Africa, the Middle East, Thailand, and USA.
- Over 20 million passengers in 2013.
- Publicly listed in 2003.

In July 2012, Norwegian launched the mobile application *Norwegian Travel Assistant (TA)* which contains all relevant information associated with your flight, such as travel documents, boarding pass, and which gate the flight is scheduled for [24]. It does not currently support ticket purchasing, but according to [22], this is something Norwegian is looking into.

3.1.2 Events

This section contains short descriptions of the ticketing systems we found for events in Norway. Public transportation is currently the main sector which uses mobile ticketing applications in Norway. This means that most of the following systems do not offer an app for the users to allows in-app purchases of tickets. With this section, we hope to show that ticketing systems for events should follow more so in the footsteps of public transportation by supplying mobile ticketing applications.

BillettService (TicketMaster)
BillettService makes 5 million tickets available to over 1000 events, rendering it the leading marketplace for events in sports, culture, music, and festivals [25]. BillettService is owned by TicketMaster, and even though they provide customers with a mobile application in countries like USA and Mexico, there is no application for the Norwegian market. However, BillettService does allow for the ticket to be sent to your smartphone for scanning at the event, which also means that the event needs to have scanning equipment available to validate the ticket [26].

TicketCo
TicketCo was founded in 2011, and is headquartered in Bergen. The founders of TicketCo had previously been event managers, but as they could not find a decent system to manage the events, they decided to create an alternative themselves. Shortly after, TicketCo was made available for everyone who needs a ticketing system for their events [27]. TicketCo offers an application for event hosts, to scan and validate tickets.

Hoopla
Hoopla was founded in 2011 with a vision and goal to make it easier for people to manage events [28]. Hoopla offers an application for those hosting events which can scan and validate tickets.

eArrangement
eArrangement AS is a company focusing on developing an Internet-based event management system for festivals [29]. eArrangement AS is the only ticketing system for events that we investigated which offers an application to customers which allows for purchasing- and viewing- tickets.

3.2 Ways of purchasing tickets

In this section, we will provide an overview of how each of the ticketing systems mentioned in the previous section support purchasing tickets.

3.2.1 Transportation

Based on our findings after investigating the different mobile ticketing applications, the following are the categories we chose for mobile payment methods in the transportation category:

- **SMS:** The mobile subscription will be charged when the user receives an SMS, which completes the transaction.
- **Credit-card (CC):** The user's credit-card will be charged.
- **Mobile account:** The user has a mobile payment account which will be charged when purchasing tickets.
- **Online:** The user can (or must) purchase tickets online via a website, which can then be downloaded to the mobile ticketing application.

Ticketing system	SMS	CC	Mobile Account	Online
AtB Mobillett	✓	✓	✓	✘
Skyss	✓	✓	✓	✘
FFK Mobillett	✓	✓	✓	✘
Kolumbus Billett	✘	✓	✓	✘
RuterBillett	✘	✓	✘	✓
NSB	✘	✓	✘	✓
Akt Mobilbillett	✓	✓	✘	✘
Flybussen	✘	✓	✘	✘
Kystbussen	✓	✓	✘	✓
SAS	✘	✓	✘	✓
Norwegian TA	✘	✘	✘	✓

Table 3.1: Purchasing methods in the transportation category.

3.2.2 Events

The following are the categories we chose for payment methods in the event category:

- **Online:** Tickets can be bought online through the ticketing system website.
- **In App:** Tickets can be bought through a mobile application.
- **In Store:** Tickets can be paid for, or picked up, at a store/kiosk.

Ticketing system	Online	In App	In Store
BillettService	✓	✘	✓[1]
TicketCo	✓	✘	✘
Hoopla	✓	✘	✘
eArrangement	✓	✓	✘

Table 3.2: Purchasing methods for events.

[1] Narvesen/7-11

3.3 Ways of validating tickets

In this section, we take a closer look at the main ways of validating tickets.

3.3.1 Barcodes

The definition of a barcode is given in International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) (ISO/IEC 1976 2-2), but only covers 1D-codes, and as such the definition by Japanese Standards Association (JSA) is preferred: "A barcode is a machine-readable representation of information that is formed by combinations of high and low reflectance regions of the surface of an object, which are converted to '1's and '0's" [30].

In the beginning, information was encoded into arrays of adjacent bars and spaces where the width of the barcode would differ depending on the design. This type of barcode is called linear one-dimensional barcode, and can be read by a scanner that sends a beam of light across the barcode. In 2D-codes, the bars and spaces have been replaced with dots and spaces arranged in a matrix or array, resulting in increased density of data. To be able to read a 2D-code, the scanning equipment needs to be able to read in two dimensions, as opposed to reading 1D-codes in one dimension [30].

1D-barcodes

The very first barcode was created by Norman J Woodland and Silver Bernard in the late 1940s, and in 1952 their patent (US 2612994 A) was published [31]. Barcodes as we know them usually consist of straight lines, but the barcode described in [31] can also be modified into a circular pattern, thus the orientation of the package carrying the barcode is not of importance when scanning it.

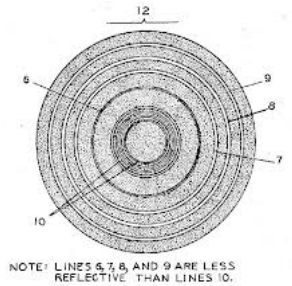

Figure 3.1: The first barcode.

Although the concept of barcodes was invented in the late 1940s, they were first successfully used commercially in 1974. On June 26th the same year the world would change, as the first upc-barcoded merchandise, a 10-pack of Wrigley's Juicy Fruit gum, would be placed on a conveyor belt. The pack of gum, once scanned, would be identified by the cash register to a price of 67 cents [32].

The barcode has since then become ubiquitous, and many varieties exist. Probably the most important 1D-barcodes are the Universal Product Code (UPC)- and International Article Number (EAN)- barcodes, as depicted below.

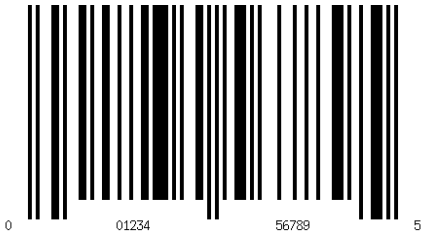

Figure 3.2: Example of an UPC-A-barcode.

Figure 3.3: Example of an EAN-barcode.

Although 1D-codes are used throughout the world in retailing, we don't see them being used for validation of tickets anymore, or at least, in very little degree. Ticketmaster (BillettService) is the only ticketing system that we found which still uses 1D-barcodes on their paper-based tickets.

2D-barcodes

In recent years, 2D-codes have become more widespread and common in our daily lives, and are more sophisticated than the 1D-codes. Whereas data is only stored horizontally in 1D-codes, data is stored both vertically and horizontally in 2D-codes. As a consequence, 2D-codes are capable of holding a lot more data, but require more advanced scanning equipment [33].

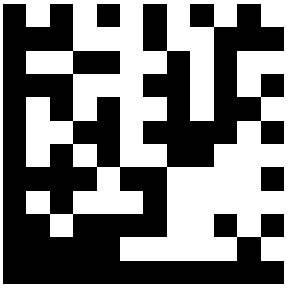

Figure 3.4: Example of a Datamatrix-code.

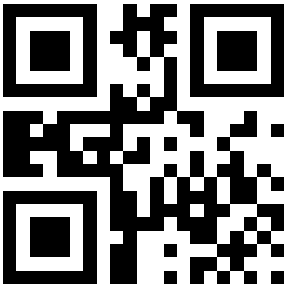

Figure 3.5: Example of a QR-code.

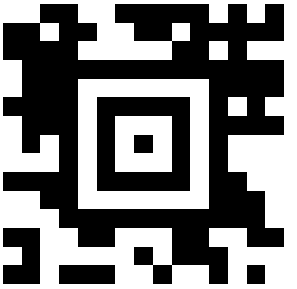

Figure 3.6: Example of an Aztec-code.

The QR-, Datamatrix-, and Aztec- codes are part of the Bar-coded Boarding Pass (BCBP) standard used by more than 200 airlines, which defines the 2D-barcode either printed on your boarding pass or sent to your mobile phone. In 2007, the International Air Transport Association (IATA) announced a global standard for mobile phone check-in using 2D-barcodes [34], and set a deadline of the end of 2010 to implement 100% bar coded boarding passes. Once fully implemented, BCBP was expected to save the industry over $500 million annually.

In the BCBP standard, the QR-, Datamatrix-, and Aztec- codes are meant to be used on mobile phones, while the PDF417 standard was selected by the industry for paper-based boarding cards in 2005 [34].

Figure 3.7: PDF417-barcode.

A QR-code is a 2D- matrix code introduced in 1994 which has gained wide acceptance in many industries such as retailing, marketing, health-care, and transportation.

The QR-code can hold much more data than other 2D-codes, and certain characteristics of the QR-code, like the advanced error-correction method, makes reading QR-codes more reliable and at higher speeds than other codes. In fact, depending on the error-correction level, up to 30% of a QR-code can be dirty or damaged and still be possible to decode [35].

The QR-code is used for validation in mobile ticketing systems such as AtB mobillett, Kolumbus billett, and Skyss billett. The QR-code contains information that a scanner checks to verify that the ticket is legitimate.

The biggest difference between the QR-code and Aztec (as seen in Figure 3.6), is that QR-codes are better for being read by consumer's phones, while Aztec is better for being displayed on consumer's phones [33].

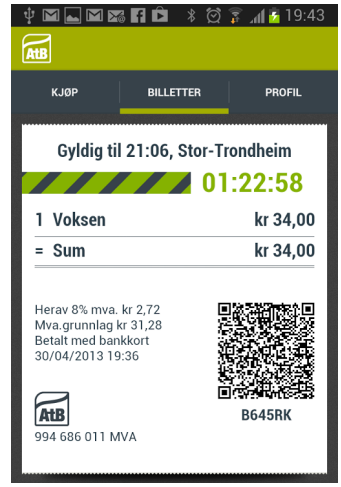

Figure 3.8: A ticket in the AtB Mobillett application.

Another difference is where the finder pattern is located - the squares that help the scanner find and orient the barcode, adjust for any skew, rotation or scaling. For QR, all four corners are needed - as the finder pattern consists of three squares in three of the corners. In addition, QR- and Datamatrix- barcodes need a quiet zone, an area of white around the code to distinguish itself from what's surrounding the code. For the Aztec code, the finder pattern is placed in the center of the code, and there's only one square the scanner needs to find to orient the barcode. All the user has to do is to place the phone screen roughly in the center of the reader to scan the barcode. According to [33], this gives Aztec fewer problems and faster scan times, making it ideal for mass-market use as a mobile ticket.

Advantages and disadvantages of barcodes

The 1D- and 2D- barcode technology works differently, each with their own strengths and weaknesses, but all barcodes have the following advantages and disadvantages [30].

Advantages

1. Data entry can be achieved fast, accurate, and reliable.
2. Barcode technology can be used for multiple purposes with low cost since it is based on paper and ink.
3. Barcode technology supports real-time operations, allowing suitable decision-making.
4. Barcodes have the same deterioration and lifespan as printed articles.
5. Data in a barcode cannot be changed without physical alteration, and thus the technology can offer secure operations.
6. Neglecting physical alterations, barcodes are considered read-only. Once printed, a barcode is difficult to alter, hence offering security.

Disadvantages

1. A clear line of sight is required to read barcodes.
2. No more than one barcode can be scanned at a time.
3. Barcode technology is unable to scan objects inside a container.

4. The reading distance is short and some scanners even need physical contact with the barcode to read it.

3.3.2 Mobile ticketing validation

By looking into the current mobile ticketing systems in use in Norway, we can better illustrate which ways tickets are normally validated. We can distinguish between the following categories for validation methods:

- **1D-barcode:** The 1D-barcode is largely used in retailing and paper ticketing, but only accounts for one of the eleven systems we looked in to.
- **QR:** QR-codes have become very common, and are used in 8 out of 11 of the ticketing systems in Table 3.3.
- **Aztec:** Used by most airliners, but not so much in other mobile applications.
- **Animation:** An animation is displayed when a ticket is to be validated. Usually, the user will be asked to press a button in the application to trigger the animation to be displayed.
- **BLE:** Not used in any of the applications we investigated.
- **Near Field Communication (NFC):** Not used in any of the applications we investigated.

Ticketing system	1D	QR	Aztec	Animation	BLE	NFC
AtB Mobillett	✘	✓	✘	✘	✘	✘
Skyss Billett	✘	✓	✘	✘	✘	✘
FFK Mobillett	✘	✓	✘	✘	✘	✘
Kolumbus Billett	✘	✓	✘	✘	✘	✘
RuterBillet	✘	✓	✘	✓	✘	✘
NSB	✘	✓	✘	✓	✘	✘
Akt Mobilbillett	✘	✓	✘	✘	✘	✘
Flybussen	✘	✓	✘	✘	✘	✘
Kystbussen	✓	✘	✘	✘	✘	✘
SAS	✘	✘	✓	✘	✘	✘
Norwegian TA	✘	✘	✓	✘	✘	✘

Table 3.3: Validation methods in the transportation category.

3.4 Limitations of current ticketing systems

This section will attempt to reveal the limitations of current ticketing systems, what should be improved, and what users are missing out on.

3.4.1 Cash is not king

One of the major reasons why bus companies in Norway are starting to provide users with mobile ticketing possibilities is the threat of robbery. It has recently been proposed that it should be legal for buses to refuse cash for payment, but the Norwegian government has rejected this proposal [36]. Bus companies are however allowed to make it less attractive to pay with cash, by i.e. raise the price for cash payments compared to mobile payment, which has been done by AtB in Trondheim [37], Skyss in Bergen [38], Ruter in Oslo [39], and NSB (also applies when paying with credit-cards in the case for NSB).

The combination of the threat of robbery and the fact that accepting and handling cash at events is tedious, make many event organizers choose *Cashless* as their temporary POS system [40]. Although *Cashless* eliminates these two problems, another one emerges from the event organizers point of view, as you would need one or more *Cashless* POS cash registers to be able to accept money at all. Some events even have their own (physical) shop where you can purchase *Cashless* cards, such as the *UKA* festival in Trondheim which lasts for about 3.5 weeks. Event organizers also want the time per sale, or validation, to be as low as possible - which *Cashless* helps achieve, as you only need to swipe your card, and the purchase is done. A major drawback in using the *Cashless* solution (for users) is that users have to pay to pay, since they would first have to pay for the *Cashless* card itself. Users would not have to pay to pay if they use a mobile ticketing application, because most people already have a smartphone (Android, iOS, or Windows Phone).

3.4.2 Information and ease of access

Events, event organizers, and event ticketing systems seem to neglect the fact that people are using their mobile phones for tasks that they previously would have to use their computer for. By having a mobile phone application, we argue that the event, or entity behind the application, is much more accessible. Mobile applications open up a new world of possibilities, including:

- Ticketing (purchasing/selling/reselling).
- Buying food and beverages.
- Information based on location-awareness.
- Easy access to event information.

In addition, a mobile application enables the usage of context-aware methods with the sensors now available in smart-phones, as well as new sensors included in the new Samsung Galaxy S5:

- Accelerometer
- Gyro
- Proximity
- Compass
- Barometer
- Hall
- RGB ambient light
- Gestures
- Fingerprint
- Heart Rate Sensor

4

Context-awareness

"Our company has, indeed, stumbled onto some of its new products. But never forget that you can only stumble if you're moving."

– Richard P. Carlton, *former CEO, 3M Corporation*

4.1 Ubiquitous systems - The foundation

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [41]. This is how it all started with Mark Weiser's paper, describing his vision of ubiquitous computing. The paper founded a new research area called Ubiquitous computing, also known as *Ubicomp*.

In the paper, Weiser envisions a future in which computing should be an integral, invisible part of the way people live their lives. He claims that whenever people learn something sufficiently well, they cease to be aware of it. As an example, consider street signs: when glancing at it, you absorb its information without being conscious of the act of reading. This concept has been given several names, such as compiling, tacit dimension, visual invariants, the horizon, ready-to-hand, and periphery. These names say, in essence, that "only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals".

According to Weiser, we will see a change when moving towards the Ubicomp era in which we advance from having one personal computer per person, to having many computers per person [42]. Weiser substantiates this claim by proposing that Ubiquitous computers will come in different sizes, and each suitable for a specific task. These computers are called:

- **Tabs:** Inch-scale computers, approximately the size of Post-It notes.
- **Pads:** Foot-scale computers behaving like books, magazines, or sheets of paper.
- **Board:** Yard-scale displays that act as bulletin boards or blackboards.

Although Weiser does not mention context-awareness directly, he talks about an experiment with clip-on computers roughly the size of an employee ID card, called badges. These badges can identify themselves to receivers placed throughout the building which makes it possible for the system to keep track of where each badge-wearing person is located. When entering a room, an employee can be greeted by the room. When someone is trying to phone the person, the system can forward the call to the room where the person is located. The badge can also be used as an access control mechanism, in which doors only open to the right badge wearer.

4.2 Pervasive systems

The terms ubiquitous and pervasive are often used interchangeably and some claim that there are only subtle differences or distinctions between the two [43], while others say that ubiquitous computing is now also known as pervasive computing [44]. We previously discussed that Mark Weiser was responsible for coining the term ubiquitous computing, but the origin of pervasive computing is more obscured. There are claims that Mark Weiser was too far ahead of his time, so the technology industry coined the term *pervasive computing* as a term for the omnipresence of information processing [45]. Others say that the term originated at IBM concurrent to Weiser's Ubicomp [46].

Starting off by defining the two: ubiquitous means "present, appearing, or found everywhere", while pervasive means "spreading widely throughout an area or a group of people" [47]. Even though the definitions yield similar, if not the same, we can in fact distinguish between how the two are used. Ubiquitous computing is often referred to when speaking of the interactions of augmented ordinary artifacts and the digital world [48]. This means that computer applications are tied to physical system components by, as Gellerson et al. explains, "embodying physical context and interactions with given physical locale" [49]. Pervasive computing on the other hand often concerns the integration of systems into human surroundings with regards to distribution of capabilities, investigating systems, and architecture [48]. The two fields clearly overlap in some ways, which make them easy to confuse and use interchangeably.

From a personal standpoint, we perceive ubiquitous systems, as Weiser envisioned it, to focus (on a larger scale than pervasive computing) on getting the computer systems out of the view and into the background. Weiser mentioned that the computer as it is today fails to get out of the way of the work that needs to be done, as it stays the focus of our attention [50]. Computers should be as the common phrase would have it: *out of sight, out of mind* - to remove screens and the input devices that consume our focus. Weiser described that "even the most powerful notebook computer, with access to a worldwide information network, still focuses attention on a single box" [41], which would also make smartphones inadequate as ubiquitous computers. It is basically a question of getting things done as effectively as possible - if Eirik were to ask Øyvind to borrow his juice maker, does Eirik want to use the juice maker or does Eirik really just want some juice?

Pervasive computing has a less strict view on the matter, where research does not focus entirely on hiding the computer, but rather make its advances widely available in a focus to assist and support user initiative. The primary motivation

behind pervasive computing is that of being context-aware, to get a feel for the physical world and supply users with real-time information as assistance [51]. What this requires is the infrastructure that ubiquitous computing is struggling with, by embedding it in the physical world. We have encountered several occasions where focus-consuming devices such as smartphones have been mentioned as the primary use in ubiquitous settings, which parts from the vision of Mark Weiser, as previously discussed. The way we understand it, systems that are in fact attention-consuming, but still make use of ubiquitous concepts, should be more likely placed under the umbrella of pervasive computers. There still exist compelling arguments for the contrary, in that we require transparency for hardware to escape our consciousness [52]. The notion of masking uneven conditions in the case of, for example, a slow network connection that gets in the way of our primary task, is an unnecessary diversion of focus.

The topic of whether or not pervasive and ubiquitous computing are separate entities is clearly up for debate and we shall not spend more ink dwelling on it, but we leave it as a hanging thought for the reader to be aware of. Another field of study that is out of the scope of this thesis, although still important to be informed of, is ambient intelligence. Ambient intelligence is closely related to ubiquitous- and pervasive- computing, and extends pervasive systems with reasoning and intelligence as a next generation of computing [48]. Continuing on, we will shift our focus to look at what it means for computers to become pervasive and how computers have evolved since Mark Weiser envisioned the new paradigm. It is a brave new world and an exciting way to rethink computers which brings forth new challenges that will be discussed more in length throughout Section 4.4.

We have seen a move from the large room-sized *mainframe computers*, to the *personal computer*, and now even more mobility with smartphones and tablet computers. Pervasive computing is rooted, and share key infrastructural aspects with distributed- and mobile- computing as early advancements of the paradigm. Still, pervasive computing also shares common ground with Human-Computer Interaction (HCI), expert systems, and software agents [44]. A thing to realize is that advances in the fields where pervasive systems are rooted, have made elements that once were exotic a reality, and thus it seems we are only missing key elements to piece everything together.

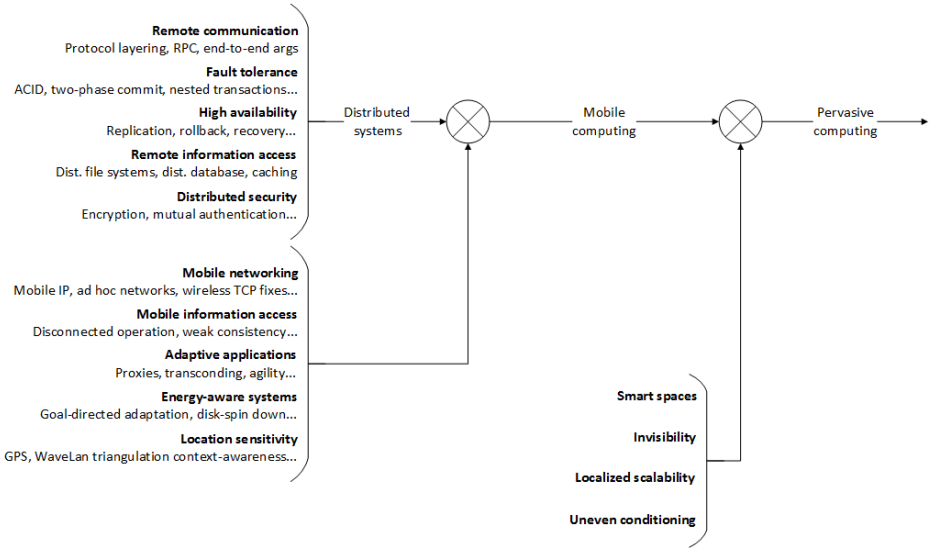

Figure 4.1: Taxonomy of research in pervasive computing, adapted from [44].

Figure 4.1 show the taxonomy of computer systems research problems in pervasive computing, as Satyanarayanan describes in his paper: *Pervasive computing: vision and challenges* [44]. One moves from left to right in the figure, where each junction adds to the collection of structural challenges and so adding to the complexity. Starting at the left, distributed systems emerged in the mid-1970's out of the intersection between personal computers and networks, bringing about new areas of research that had to be tackled such as: remote communication, fault tolerance, high availability, remote information access, and distributed security. As mobile clients and wireless network connections came into existence in the early 90's it added to the discipline of distributed computing with aspects such as: mobile networking, mobile information access, adaptive applications, energy-aware systems, and location sensitivity to accommodate computers on the move.

As Satyanarayanan mentions in his paper, there are four particular niches to be aware of that make computer systems pervasive: invisibility, masking uneven conditions, localized scalability, and smart spaces. We can also extract subcategories for each of these subjects in which divulges a larger picture of the challenges that each present. In the following, we shall briefly describe each area.

4.2.1 Smart spaces

Smart spaces focus on joining the digital world and the physical world by adding the ability to control and sense the physical world by devices [44]. There are different ways in which smart spaces can be realized; one is by the simplicity of controlling aspects of the smart space such as: temperature, light levels, or blinds. The other is for software to sense its presence in an environment for it to adapt accordingly at run-time. In some cases smart spaces can incorporate both to be proactive, by sensing or exploiting the knowledge of people or artifacts in its bounds, and to control its environment (temperature, light, etc.). In essence, smart spaces integrate: technology, information, communication, and sensing into everyday objects [53].

4.2.2 Invisibility and transparency

A key point of research is definitely the physical size of devices. As we move towards the *disappearing computer*, we must not neglect that computers do not have to be tiny or barely noticeable to disappear. To disappear, we need to focus on the concept of a less demanding and attention-seeking computer. This calls for a re-evaluation of the form factors, interfaces, and how we communicate with computers, as well as how the system perceives the context it is situated in [43]. This means that we have to:

1. Break away from the static confines of the desktop interaction model, so that adaption can be facilitated in a better way [54].
2. Minimize user distractions by meeting user expectation for him or her to interact close to subconsciously with the system [44].
3. Incorporate natural interfaces for a richer interaction between humans and computers that support common forms of human expression [51] and real world action such as movement [44].

4.2.3 Masking uneven conditions

Masking uneven conditions is a joint between fault tolerance and invisibility. We previously remarked that it was an unnecessary diversion of attention if a slow network connection got in the way of our primary task, and that this was an unnecessary diversion of focus [52]. This shows that we have to somehow find a way to compensate for “dumb” environments to reduce variation and keep the user on track [44]. This is a principle rooted in distributed systems and built upon

by mobile computing, where users are constantly changing locations and thereby the environment which they are situated in. An application should protect how users are exposed to these variations and adapt at run-time [43].

4.2.4 Localized scalability

Localized scalability is a result of distributed- and mobile- computing as a product. In distributed systems it has always been a problem area to facilitate access by multiple users at once with regards to factors such as: bandwidth, storage, and energy. On the storage front, scalability and availability has brought forth the NoSQL database movement to cope with the endless amounts of data that is generated on a daily basis by networked computers. As for the pervasive- and ubiquitous- paradigm, the need has somewhat shifted to regard physical distance, to not only have access to global scalability but also local scalability. Smart spaces are somewhat to blame as interaction with computers are in some cases handled by proximity, and the amount of interaction with the space tend to fall off as people move away [44]. When large amounts of devices enter a particular space, the need to supply these with the right communication and infrastructure to function satisfactory becomes a priority. Take an event, such as a concert (which is relevant to our case-study), where thousands upon thousands of users with devices are in need of accessing the same resources at once. An example is where NetCom and Telenor, the two largest mobile network providers in Norway, had to increase capacity to prevent a network breakdown in 2013 when a teenage idol attracted masses of fans to a concert in Oslo [55].

4.3 Context-Awareness

Our focus will further shift to look closer at another field within pervasive computing which we barley brushed upon earlier. This particular field is referred to as context-awareness, and is centered on the ability to appreciate and proactively monitor the environment in which the computer system is situated in and react to it.

4.3.1 Context

Arriving at a formal definition of what a context is can be quite troublesome as context can be broad and variable. Although the undertone of previous definitions provides a general understanding of context, many still find it difficult to elucidate.

The term context-aware was first introduced in 1994 by Schilit and Theimer [56, 57]. They defined it as "the ability of a mobile user's applications to discover and react to changes in the environment they are situated in" [58]. As we have seen, research into context-awareness started before the term was introduced, with Mark Weiser's vision of *Ubiquitous Computing* in 1991. The first application credited with using context-awareness [56, 57, 59] was created one year after Mark Weiser's *UbiComp* paper by Want et al. with *The Active Badge Location System* [60].

The Active Badge Location System's primary use of context was location, as location was, and still remains the most crucial, explored, and frequently used form of context [43, 56, 58–60]. One of the factors of location being most used can be traced to the wide variety of devices which support location services such as Global Positioning System (GPS). An issue to be aware of is the border line between input of a system and the context of a system. Location can for example be both a context and an input, where a navigation application takes location as input and not as context [43]. The bottom line is that the context should not shape an application, but merely assist in its adaptation, or as Chalmers [43] emphasizes: "conditions that might affect an action, rather than conditions which define an action". Another to support this claim is Kofod-Petersen, in that relying solely on location as input is hardly considered to be aware [48] and refer to such systems as stimuli-response systems [61].

There is contradicting evidence that location is enough for a computer to be assumed more intelligent than it is. Weiser explains that a computer can, without even a hint of artificial intelligence, adapt to its environment by the sheer knowledge of which room it is located in [41]. Proxemic interactions, which we

will discuss at length in Section 5.1.1, demonstrate that computers which react to the distance of a subject while only being a simple state machine is thought to have intelligence beyond its simplicity [62]. So as we can see, there is in fact a reason behind the heavy emphasis on the use of location through the ubiquitous- and pervasive- research field. This leads us to a question which we will discuss later of whether context is regarded as data, information, or knowledge, and if the notion of reasoning is a part of being context-aware.

A vast variety of definitions have been proposed regarding context and context-awareness since Schilit and Theimer first defined the term, many of which define context by examples. These definitions can become inadequate when attempting to determine whether or not something can be considered as a context [57]. Examples that occur frequently are:

- Location
- Environment
- Identity
- Date and Time
- Idea
- Emotional state
- Focus of attention
- Orientation
- People
- Objects
- Activity
- Situation
- State
- Surroundings
- Task
- Motion

Schmidt, Beigl, and Gellersen [59] summarize such aspects of contexts as "that which surrounds, and give meaning to something else". Considering the list, one could reason that we can taxonomize context in a hierarchical model [59, 63], since most areas can co-exist. The top level of such a model can for example be divided between the human factor, and the physical environment. The human factors should be related to the user, his or her social environment, and the tasks or goals, while the physical environment encompasses the physical conditions, such as: light, noise, location, etc. The model can be extended from here on with more specific contextual features at a sub-level. These features are further identified by a range of values specific to the feature [59, 63]. Also, context can be defined by history to determine a situation or environment, which means that time and space become relevant features [59]. Figure 4.2 from *There is more to context than location* [59] and Schmidts PhD thesis [63] realize such a model.

The dictionary [47] also defines context as "the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood". The key word being that the context should be fully understood,

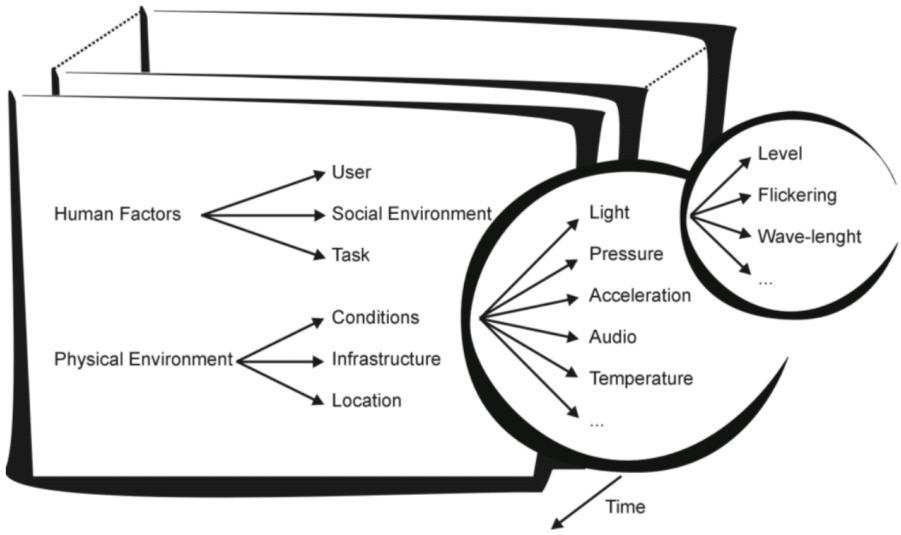

Figure 4.2: Hierarchical classification model, taken from [59, 63].

making the general guideline for an application that it should understand its context to be context-aware, although we still lack the ability for computers to fully understand many aspects of context such as emotions or human thoughts. To provide useful input to the application, we would argue that a requirement for context in a context-aware system to be that the contextual information can be correctly interpreted by the computer to support the user effectively. This notion is somewhat similar to what Kofod-Petersen [48] argues on the basis of work by Zibetti et al. [64], that agents should "understand situations based on the information they can perceive".

As humans, we can interpret our environment through our senses to accommodate us in making decisions that seem to fit the current situation. Context is considered a huge factor in human reasoning, even though assessment of a given situation is considered to be subjective and such knowledge is not shared, objective, nor public, but individual, which comes down to internal state and personal experience [48]. Schmidt [63] suggests that we can compare context to senses. This is built upon by Chalmers [43] who indicates that an application has to develop similar senses to align to its environment. Compared to human intelligence, intelligence as the computer science community regards it is action-oriented, where context can be thought of as a tool for selecting an action [48]. This means that

since reasoning over a given situation comes down to internal states and personal experience, context can be considered an elusive type of knowledge. Knowledge, as being defined by "facts, information, and skills acquired through experience or education" [47], as well as the fact that context is an elusive type of knowledge, makes it hard to logically quantify and reason over which types of context are useful for certain situations and which are not [48].

We should also point out that contextual information is not limited to the explicit contexts that a user can observe or sense. There exist other factors in an environment that can determine, or be a context. An example is the underlying computing environment, which should be considered as an aspect of a computer system's context [43, 57]. These environments can be related to the hardware or infrastructure and thus the availability and bandwidth, power, memory, and storage, as well as interfaces, such as: screens, speakers, and input devices. With this in mind, we can extend the hierarchical model of context with the computational environments, arriving at the same three top-level categories as Dey and Abowd [57], as well as Chalmers [43] deduct from others [65–67]: computational, human factors, and physical. Such a classification divides context into groups by what they require of processing [43].

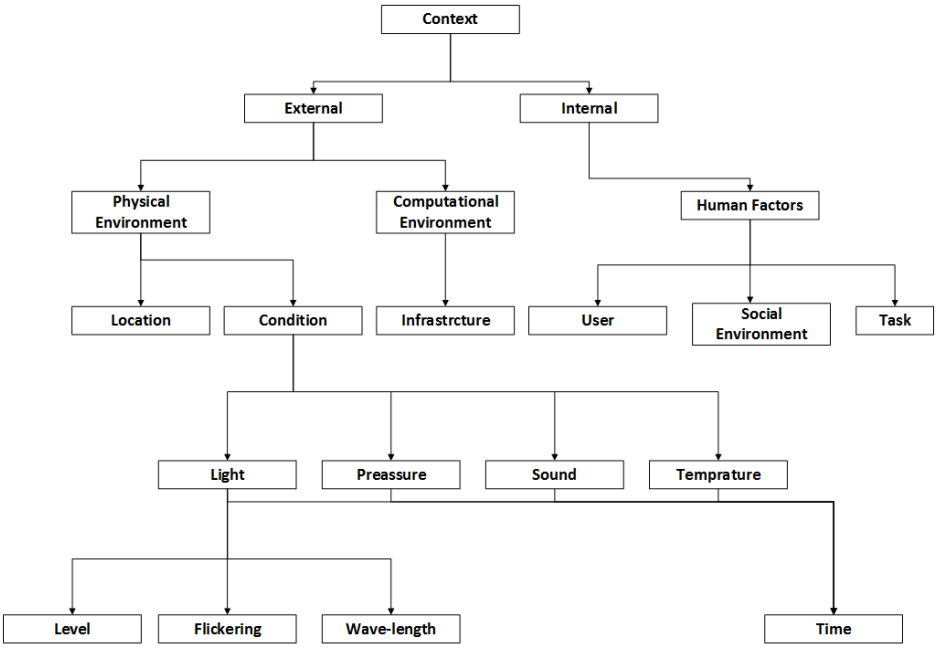

Figure 4.3: Our context classification.

Figure 4.3 extends the classification model of Figure 4.2 and demonstrates how we would classify context. The additions we have made are simple, and build upon the descriptions of context from various other sources, in which we will explore in the next paragraphs.

We first classify context by internal or external dimensions [48, 56]. External context refers to the more physical perspective of context and environment, while the internal refers to the user, the social environment, and tasks. Furthermore, the model is divided by factors previously discussed: computational, human, and physical. Sub-categories of these factors are likewise divided into specifics that relate to the category, which coincide with the model in Figure 4.2. Other research describes the external and internal classification as *background context* and *selective interest*, or *intrinsic* and *extrinsic* context [48], which is further classified into sub-levels that often differ in some way or another to ours. Ekbai and Maguitman [68] explain that according to philosopher John Dewey, background context can be divided into *spatial-* and *temporal-* aspects. The temporal aspects can be existential or intellectual, while the spatial "covers all contemporary settings within which a course of thinking emerges." The existential properties of the temporal aspect grasps contributions to a thought process that emerges from the environment or all material means, while the intellectual are considered to be thoughts that an individual possesses based on his or her history, or background (traditions, habits, science, etc.). Furthermore, Ekbai and Maguitman suggest the following from the pragmatic approach of John Dewey:

1. Context, most often, is not explicitly identifiable.
2. There are no sharp boundaries among contexts.
3. The logical aspects of thinking cannot be isolated from material considerations.
4. Behavior and context are jointly recognizable.

The paper attempts to outline how a logical approach to context and relevance in Artificial Intelligence (AI) falls short, and that logic-based systems fail to fulfill the inseparability of mind and nature. This means that to reason over context to a full extent, a system needs a form of consciousness to make decisive decisions. Relevance of importance "is the impress of the individual inquirer on the context" [68]; we as humans attempt to reason and make sense of situations through context and experience, even though situations can be confusing, obscured, and conflicting [48]. In our case, it is useful to be aware that the pragmatic approach exists, although it might not be a necessity for a simple system to develop mindfulness to benefit from context. The way of handling context by intelligent systems is rather a field of research to ambient intelligence, as we made a note of

earlier. It should first and foremost be vital for developers, more than the system, to reflect on the taxonomy and notice that some types of context might be more important than others for a given scenario. A rule of thumb in endeavoring assistance in context are the *five W's* as a minimal set of necessity defined by Abowd and Mynatt [51]:

- *who* - the presence people,
- *what* - interpreting activities,
- *where* - the current location of the environment,
- *when* - changes in time relative to the who, what, and where,
- and *why* - understanding the what.

4.3.2 Acquire, model, and adapt

The way a system can adapt itself through the change in context, or user preferences, is to facilitate an understanding of sensor data or contextual information. We will therefore take a closer look at acquiring, modeling, processing, managing, and adaption of contextual information.

Systems that are context-aware may be broken into parts after responsibility. Kofod-Petersen and Aamodt demonstrate [61] such a break-down in three parts. The first is the notion of perceiving an environment by acquiring or detecting knowledge or information, the second part has the responsibility of being aware of the environment, while the last part is concerned with adapting to the environment. The last two parts are referred to as context-awareness, and context-sensitivity, respectfully. Some also describe context-sensitivity as self-adaptation [69]. For simplicity and elimination of confusion, we will refer to context-awareness in the sense of acquiring or detection as *context-acquiring* and context-sensitivity in the sense of adaptation as *context-adaptation*. Throughout this section we will also talk about *context-modeling*, where we will look at how to make sense of the data that we acquire by converting values into meaning.

Acquiring

We can separate between acquiring context explicitly or implicitly [59]. By explicit context acquisition, we mean that the user specifically inputs context to the system or that the system inquires it from the user. With implicit acquisition of context, the system monitors its user and environment. To implicitly monitor requires other resources and often more complexity than to explicitly ask for

context. There exist two approaches for implicitly acquiring context as Schmidt et al. [59] describes: one is providing an infrastructure to obtain the context and the other rely on sensors attached to the device. For the latter, sensors can as an example be used to monitor the physical environment or its user through heart-rates, temperatures, movement or motion, lights, digital images, and so on, most of which are embedded in many current mobile clients or smartphones. On the other hand, an infrastructure constitutes a smart space, which we looked at in Section 4.2.1. In essence an infrastructure is embedded in an environment to provide clients with information beyond what they are eligible or capable to perceive on their own. This is why locational units such as GPS are considered a smart environment instead of a sensor [59]. A smart space may not only be determined by embedding sensors in a physical location. As we formerly discussed, we should detect uneven conditions and attempt to mask these or suggest ways of improving them. Satyanarayanan [44] describes an example of Aura, a computer system that in an airport "discovers that wireless bandwidth is excellent at Gate 15, and that there are no departing or arriving flights at nearby gates for half an hour". This demonstrates that it is crucial to also acquire contextual information of the computational environment of the smart space.

Baldauf et al. [56] proposes a rather different approach, where they explain the process of acquiring of context by using the word *sensors*. Sensors are explained as sensing hardware and data resources that are able to provide contextual information. They present a classification of three types of sensors: *physical sensors*, *virtual sensors*, and *logical sensors*. Physical sensors are, as we previously discussed, the direct access of locally built in sensors of a device. Virtual sensors look at an interesting concept which is explained as collecting contextual information from software applications or services. An example is the use of data from electronic calendars, meeting notes, emails, booking system, etc., to provide contextual information to pin-point where a person might be, what he or she might be doing, or with whom the person is attending a meeting, conference, or concert. This is somewhat related to Activity-Based Information Retrieval (AIR) where data is annotated with the context in which it was generated [70]. There are also other uses of such *virtual* forms of data, for example monitoring mouse movement, or keyboard input. Furthermore, the use of logical sensors combines physical- and virtual- sensors. A basic example is that of mapping a person's device to a location and strengthening the fact that the person is actually present with, for example data from his or her calendar.

Modeling

There exist a vast amount of sources to acquire context from. The ideal way of handling context is through the same way we handle user input from traditional input methods such as keyboards or mice. However to make proper use of context, we are not always fortunate enough to rely on only one source to supply the whole context. We therefore present this section on modeling context, where we will concentrate on how to assign meaning to values. This will address the concern of specifying how context for different sources can be joined and abstracted to a higher-level to give it meaning beyond the raw-sensor data, as well as provide transparency.

A proposal of assigning meaning to sensor data is what Schmidt et al. [59] describes as *sensor fusion*, or as Baldauf et al. [56] refers to as *context aggregation*. Similar ways of assigning meaning to sensor data is also explained by Dey et. al [71, 72] through *context widgets* and *aggregators* in the *context toolkit*, where widgets interpret sensor data and aggregators collect context from multiple sources. The general idea is that of joining sensor data for abstraction at a higher level than it initially was read. In this way the contextual information will have a greater meaning for humans to understand. The data can also become more precious and accurate [56]. For instance, by gathering sensor data (over time) from sensors such as light, noise, and movement, as well as location, time, and time-zone, we might interpret that it is *midday, weekend, outdoor, sunny, moving at 15 km/h*. From such contextual information, we can derive that the user is probably out on a weekend stroll on his bike. By using this approach, we can build more scenarios based on contextual information such as: *in a meeting*, *at an event*, *standing in line*, *in a conversation*, and so on. You might have noticed that we used the word *probably* when referring to our context-scenario example. This was not unintentional, since descriptive context is often based upon incomplete information and therefore considered a statement of uncertainty or belief [43]. From our example, the user could in fact be out for a run, and not riding a bike. However with more contextual information such as: heart-rate and the user's fitness profile, we might be able to distinguish between running and a light bike ride.

A key to context in any context-aware system as Strang and Linnhoff-Popien describe, is a well-designed model of context [73]. By model of context, we are referring to a technical representation of the data that we acquired. Even though this is related to our hierarchical classification (Figure 4.3) presented in Section 4.3.1, the representation of context in an application has to abide by some other demands or requirements than just being practical for human understanding. In our discussion from Section 4.2, pervasive computing (and therefor also context-

awareness) is a subset based on advances in distributed- and mobile- computing. With this as a background Strang and Linnhoff-Popien, in their paper *A Context Modeling Survey* [73], discuss 6 requirements that a context modeling should attempt to align itself to in a pervasive computer system.

1. **Distributed composition:** Systems that are ubiquitous or pervasive are considered derivatives of distributed computing systems and so a context model should preferably be able to be distributed.
2. **Partial validation:** Partial validation of contextual knowledge should be possible on structures and instance levels.
3. **Richness and quality of information:** Variations may occur over time in the quality and richness of information gathered by sensors.
4. **Incompleteness and ambiguity:** Especially in regards to networked sensors or infrastructures, contextual information might be incomplete or ambiguous.
5. **Level of formality:** Interpretation of commands should, if possible, be precise and traceable.
6. **Applicability to existing environments:** The context model should be able to be implemented in existing system structures.

Strang and Linnhoff-Popien's paper takes into account the most relevant context modeling approaches that are, or have been proposed in numerous systems. In Table 4.1 we have very shortly summarized the description and evaluation of each of these models. Keep in mind that for some systems there can be limited support to implement different modeling approaches.

Model	**Description**	**Evaluation**
Key-Value	The key-value is described as the simplest data structure for modeling of contextual information. It's usually used in discovering systems to describe capabilities of services.	Easy to manage, but lack capabilities for sophisticated structuring as well as enabling efficient retrieval. Weak on requirement 1-5 for context modeling (listed above).

Markup Scheme	The markup scheme consists of tags in a hierarchical data structure with attributes and content. Most notably is the Extensible Markup Language (XML), which is often used as a superclass.	Strong on the partial validation requirement, although completeness and ambiguity have to be handled on the application level. Distribution depends on the markup scheme used.
Graphical	The Unified Modeling Language (UML) diagram is a strong graphical and generic component which makes it an appropriate model of context. Various approaches have been proposed.	Particularly applicable to extract Entity-Relationship (ER) diagrams for context management in relational databases. Its strengths are only on the structural level, and falls short on distributional composition and on computational evaluation, as it is mainly for human structuring purposes.
Object Oriented	The object oriented approach allow for the full power of structures such as encapsulation, inheritance, reusability, etc. This makes it useful for the abstraction of context and of handling the dynamics of context as well.	Strong in regards to distribution, dynamics, and partial validation. It is also equipped to handle ambiguity and incompleteness correctly, as well as the level of formality through interfaces. Some drawbacks exist on the applicability to existing environments as some requirements exist. This especially applies for encapsulation on the formality requirement.

Logic Based	For the logic based model we define models through expressions, facts and rules. An example could be: *isp(c,p)*, where p is true in context c.	The model has a high degree of formality, as well as a possibility to distribute. However, it is error-prone as it lacks partial validation and requirement 4 is not addressed. Requirement 6 is also considered a major issue.
Ontology Based	Ontologies are considered strong in modeling context as it represents context by concepts and the relationships among them. The model very well depicts how *everyday life* is structured in a way computers can interpret.	Very high formality and strong similarities with object orientation also makes distribution possible. Partial validation, incompleteness, and applicability in existing systems are also addressed. It is concluded as the most promising context modeling approach.

Table 4.1: Context modeling approaches.

Ontology is a borrowed term from philosophy, where it is a study of the theory- and nature- of existence. The term was adopted by the AI community to define the relations among documents and files. Berners-Lee et al. [74] discuss ontologies in their paper *The Semantic Web*, where the pages on the World Wide Web are connect to ontologies to give terms greater meaning for a computer to understand and reflect. They describe inference rules that can provide power of expressions when using ontologies. For instance if a city code is connected to a state and if an address is also connected to the city code, it is then possible to assume that the address is in that state.

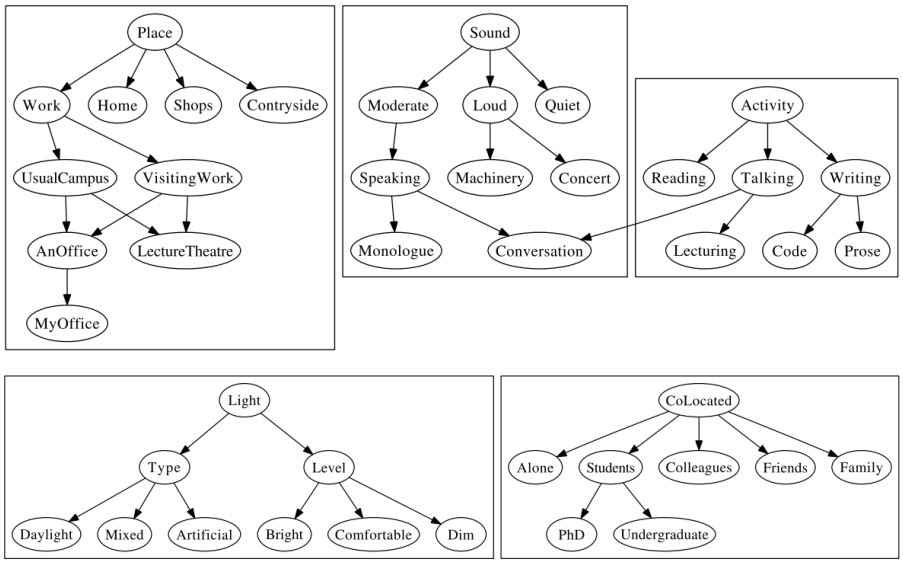

Figure 4.4: An example ontology, taken from [43].

Chalmers [43] also argue that ontology is best suited for context aware modeling, building his argument on Strang and Linnhoff-Popien's paper as well as Chen et. al [75] and others whom discuss ontology models. Figure 4.4 shows the example of how Chalmers demonstrate an ontology model of an academic context. He explains that the model can be of higher value when treated with relationships. An object, *phone*, can be connected to *Eirik* or *Øyvind* with the relationship *owned by* or *in use by*. This also works for the classical contextual problem *print to the nearest printer*, where we can examine a relationship as *near* or *within a room*.

Adaptation

We mentioned that the ideal way of handling context was through the same way we handle input from traditional input methods such as keyboards or mice. Recall from Section 4.3.1 that context is not an explicit input and should not define an application? In reality, even when we have provided a level of abstraction that can assist, there are several ways of defining responses to context that might not be as straight forward as responding to mice movement or keyboard pressing. Adaptation to context has a primary goal to support user initiative. To do this

context has to be interpreted as cues for determining what users are attempting to do, and proactively assist them [54]. To define an adaption strategy for an application, we must first distinguish the uses of contextual information.

1. *Context aware presentation* [43]/*Contextual sensing* [67]/*Automatic contextual reconfiguration* [65]: Displaying information or adapting the interface to present contextual information automatically to the user based on the current context.
2. *Context aware configuration* [43]/*Contextual resource discovery* [67]/*Promximate selection*[65]: Discovery of services that are *nearby* or relevant to the user's context. Resources may be exploited or emphasized.
3. *Context triggered actions* [43, 65]/*Contextual adaptation* [67]: Execution or modification of services or commands automatically in a given context, for example loading maps for the next location.
4. *Contextual adaptation of the environment* [43]/*Contextual commands* [65]: A way of associating action triggers in the environment. This is explained as a way of executing services manually when they are made available because of their manifestation in the current context (light, heat, etc.).
5. *Contextual augmentation* [43, 67]: A way of associating data to a given context or annotating data to the context where it was generated. An example of the first instance is how Øyvind can leave an electronic note at his desk saying that he will not be in today, which can be retrieved by Eirik or anyone else who approaches the desk. This is also a basis for how some virtual tour guides are constructed.

In addition *contextual meditation* is described, which is a way of enhancing interaction with an environment by modifying services or data to best suit needs or limitations [43]. An example might be that the interaction is adapted to predefined user settings. There is also a description of *context display* as a support to explain behavior based on a context or as a primary display of information. We also encountered two different ways of explaining *contextual augmentation*, one in which we wrote a little about in the section of acquiring context. What Chalmers describes as *contextual augmentation* (annotation of data to the context in which it was generated) is similar to what Newman et al. [70] describe in their paper as AIR. Both *context display* and *contextual augmentation* is not entirely a part of contextual adaptation and border in some cases on acquiring context. We choose to include it under adapting to demonstrate the variety of definitions.

Dey and Abowd [57] has a slightly different approach, where they only present three generic profiles (listed with the concurrent profiles in parentheses):

- Presentation of information and services to a user (1, 2, 4).
- Automatic execution of a service (3).
- Tagging of context to information for later retrieval (5).

They argue that profile 1, 2, and 4 might be redundant, as profile 2 and 4 does not make any distinctions if it actually is a presentation of information, or a presentation of a service. Furthermore they write that it is difficult to differentiate between the presentation of services and the presentation of information, as it is really up to how the user chooses to use the information.

We can define how to process actions from these profiles, which can in terms be helpful to further define a context-aware system. We can divide into event- and state- driven behavior [43]. Event-driven behaviors are one or more actions triggered as a response to some occurrence, and a state-driven behavior is a continuous set of occurrences which are based on the monitoring of some artifact or object. Therefore a basic difference is that events are triggered and states are monitored. For an event-driven behavior we have to have an object or artifact that allows a callback once the event occurs. We are not always fortunate enough to encounter event-driven behaviors and that is why we sometimes have to monitor some object or artifact for change. There is a cost for state-driven behavior, as it is generally a power consuming process to monitor someone or something. The process should be limited to supply information only when necessary, though this might cause delays or lags in an application.

Profile	**Event-Driven**	**State-Driven**
1	✘	✓
2	✓	✘
3	✓	✘
4	✘	✓
5	✘	✓

Table 4.2: Profiles and events.

Table 4.2 show how each profile is bound to the different event procedures. For profiles 1, 4, and 5, we have an interest in current states of a context by discovery or monitoring. Profiles 2 and 3 are driven by events that are triggered. [43] points out that *contextual meditation* also fit in under state-driven behaviors.

4.4 Research Challenges

We briefly looked at some research challenges coupled with pervasive computing earlier in the chapter. Through this section we will extend this and take a closer look at the most centric challenges that exist in building systems with ubiquitous- and pervasive- capabilities. We will try to keep a general emphasis on context-awareness, as well a primary focus on the software engineering aspects. Even though most of the technology to realize these systems exists, we are not yet able to piece it all together. So with this section we wish to note what the future might bring and what lies ahead.

4.4.1 People

The benefit of context-awareness is to provide a much more seamless experience for the user, although there are several unresolved issues related to the people perspective. Privacy, security, interfaces, and communications need to be tackled, in addition to hardware. When large amounts of physical hardware is to be situated in an environment to provide systems with information, it has to be careful not to pollute the environment but rather be aesthetically pleasing and complimenting while bringing new features. Hardware should not be obtrusive with blinking Light-Emitting Diode (LED) lights, beeping and fan noises, or exhaust and heat [43]. We require subtlety when designing hardware, which also translate to the user interfaces of systems.

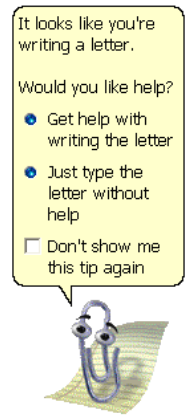

Figure 4.5: Clippy - office assistant.

Since we will be working primarily with mobile smartphones, our research will not focus on removing the Graphical User Interface (GUI) completely, such as when speaking of *disappearing hardware*. In contrast we aim to enhance the GUI and the usability with context-awareness. Disappearing hardware brings forth new challenges in which is somewhat out of our scope concerning how the system knows that someone is addressing it, how someone know the system is attending to them, how the system knows what a command relates to (saving, deleting,

etc.) and how systems understand a command and recover from eventual mistakes [76]. Most of these features are intuitively resolved in GUI systems with buttons, cursors and such. Still, we suffer from an unnatural process of dialog with computers. It is counterintuitive for a context-aware system to constantly inquire the user for permission or questions along the course of using an application. Privacy concerns are a major transgressor of going about an unreasonable amount of user inquires. Although there exist many other attempts at creating context-aware or assisting interfaces that have failed to address the balance of proactivity and transparency. A famous example is the Microsoft Office Assistance *Clippy* depicted in Figure 4.5, in which was criticized for being annoying and intrusive, thus defeating its purpose [77, 78]. The point being that such a system should know when it is appropriate to disturb. Satyanarayanan [44] suggests that these kinds of assistance should contain a *user patient model* that can predict how a user will react (positively or negatively) to a systems *intuitions* and *decisions* before it is automatically executed. Although such a model might need some type of AI or machine based learning strategy. Others suggest similar alternatives to privacy factors saying that context-aware systems should be so proactive that it can build a privacy profile of user preferences based on previous requests and interactions in the application [79]. Privacy is a huge factor for context-aware systems, although asking the user for permission to enable, for example, the GPS to be able to track the user every single time the user accesses the application can become cumbersome. It is the principal of finding equilibrium between usability and user protection.

It is a criterion of success for a context-aware system to exploit information about users for proactive assistance, and this information should be protected accordingly. It is a necessity to establish trust between the system and the user. Privacy in computer systems is controversial, as it should be. The harm that theft or misuse can produce on the basis of such information ranges from; spam to blackmail and identity thefts. This does not always concern theft, but can also be in regards to companies becoming too powerful by the sheer knowledge of their users. Mark Weiser [41] already remarked privacy concerns at the beginning of the ubiquitous computing era by saying that with hundreds of interconnected computers in every room it could have "the potential to make totalitarianism up to now seem like sheerest anarchy".

Other human factors are how we handle undesirable situations. Faults and errors are unavoidable but should in most cases be handled in a context-aware way, and by that adapt. If, for instance, the sensed data is ambiguous, the application should not continue without users consent. How should we proceed if a user does not give the application permission to use one or more of the request features (such as GPS to track their location)? Should the application quit or attempt to

adapt to the decision, offering other solutions? - Of course it should!

4.4.2 Variations and discovery

In Section 4.2.3 we vaguely discussed that we have to mask uneven conditions for pervasive systems. This challenge is a result of variations in both the computational- and the physical- environment. To recap, we need to provide transparency to expose the user to a minimum of the technical details, so they can go about their business as usual. Unfortunately we suffer from heterogeneous systems and ad-hoc communications in which creates problems. There exist large technological variances in topology, transport, and scopes (area of reach) of systems that should, preferably, be interconnected without complications.

Our environments are dynamic; objects and people are constantly on the move. Season change, locations change, patterns change, we grow up and old, furniture move, and so on. With this in mind, variations are not just limited to scale but also distance and time [43]. Systems should have the ability to facilitate such changes and adapt accordingly, and preferably not at users expense. Keith Edwards [80] proposes some new directions for research in terms of variations and discovery:

- It must be possible to provide an infrastructure for devices to connect even without having an infrastructure (scalability, security, administration, shard naming, etc.).
- We need to bridge and deisolate discovery between different networks.
- An appropriate search is required to detect resources.
- Discovery is as much a human issue as a technological issue.

The aim must be to be able to discover and connect devices without an explicit configuration or prior knowledge of those devices [80]. If we must configure each device that enter a network explicitly for them to work together, it can become so overwhelming that any benefits we have from using such a system might be too small in contrast to the workload of connecting. For example, if we arrive at an event and have to connect to a network, input a password, and specify which service or resource we want to use to validate our ticket, the benefits of getting away from a printed barcode might not be enough for people to want to use the system. We are headed towards a point of mobility where the aim is to move through the world by mostly making temporary connections to services. It is discusses that the dynamics of such technology creates a too large overhead for a service such as Domain Name System (DNS) to handle

[43]. The level of interactions that is attractive to pervasive computer, such as knowing that a device is arriving or departing a context, is not supported by DNS and Dynamic Host Configuration Protocol (DHCP). Still, in the absence of a resource identification service that can handle and facilitate the discovery and configuration of devices, a temporary solution might be as simple as a QR-code or Radio Frequency Identification (RFID)-tag for easily connecting and configuring a service.

4.4.3 Other

There exist several other questions which we must ask when building a context-aware system. Some of which might not be especially essential to software engineering, but important none the less. One is a question of energy, which is vital when we look at the scattering of resources in the physical space. How will we supply all these gadgets with energy? Will we use batteries, and how long can a device last on a single charge? Will devices be plug into an electrical socket? If so, how will we manage hundreds of devices in a single space with regards to wiring?

Moore's law is a well renowned prediction in computer- and electric- engineering. The prophecy says that every two years the number of transistors that can be placed on a chip will double. Unfortunately, doubling of resources is not true for power technology. It is not only a challenge for the hardware community, but also for software engineers. We have to re-evaluate how we construct software by using as energy efficient strategies, algorithms, and protocols as possible. As an example, error detection in network protocols such as Transmission Control Protocol (TCP) is a necessity to recover from faults in networks. Still, corrections, retransmitting packages, Acknowledgement (ACK), and Negative-Acknowledgment (NACK) comes at a cost.

Another challenge related to power is the *thickness* of a client. We say that a powerful client is *thick*, while a minimal client is *thin*. This might not be directly relevant to context-awareness, but more to pervasive- and ubiquitous-computing. When speaking of discovery and use of nearby resources we can discuss something known as *cyber foraging* [44]. Cyber foraging is the thought of exploiting hardware resources on other devices for computationally demanding tasks. Imagine that you have a task at hand (on your smartphone), which is demanding, for example compiling of software. The thought is that you send this task off to an available external resource to execute, and just receive the result when it is done. By doing this mobile computing can build smaller computers with less demanding hardware so that the client can also last longer on single

electrical charges. A catch is that the mobile client must, of course, be able to perform in a satisfactory manner in a worst-case environment.

5

Approaches to context-awareness

> *"It would be heartless to house legless men in a building which could only be entered by ladders or very steep gradients."*
>
> – Humphry Osmond, *psychiatrist*

In the previous chapter we took a look at context-awareness, how it is regarded and where it originated. Through this chapter we will take a closer look at context-awareness, the technology, sensors, and methods of approach that are available to us by moderns standards. We will focus on mobile applications or *apps*, with a general aim to build a better UX for smartphones. In doing this we have to conduct ourselves accordingly to the frameworks, or operating systems that are applicable. Fortunately, smartphone frameworks provide a level of abstraction that keep us at a distance from handling low-level sensor data in the provided Application Programming Interface (API) and Software Development Kit (SDK). Although we keep a general spotlight on the technology that is available to most phones on the market today, our framework of choice is Google's Android operating system. This chapter aims to narrow down the former chapter to more concrete practical approaches and practices.

5.1 Practices

In this section we discuss some modern practices that can be used to approach context-awareness in computer science. We will for the most part keep to a level of abstraction beyond the technology (which we will come back to further on in the chapter) that is needed to realize these approaches.

5.1.1 Proxemics

Proxemics is the study of HCI with respects to the position, movement, identity, and location between the two. The term Proxemics was coined by anthropologist Edward Hall in 1966, creating the foundation of a field of study which identifies the ways people use interpersonal distance to understand and mediate their interactions with other people [81].

The field of proxemics has many aspects to it, one of them being linguistics. Hall worked with linguistic scientist George Trager, and observed shifts in the voice as distance either increased or decreased. Knowing that people whisper when they are very close, and shout when they're far apart, Hall and Trager posed the question "How many vocal shifts are sandwiched between these two extremes?" They found the answer to this question to be eight distances, as described in the book *The Silent Language*. However, the most interesting theory from Hall's work, and also the most relevant to this thesis and HCI in general, is his definition of four proxemic zones. These zones define how people interpret interpersonal distance.

The four proxemic zones, as described by Hall:

- **Intimate:** Less than 1.5 feet (<0.46 meters).
- **Personal:** 1.5 to 4 feet (0.46 meters to 1.22 meters).
- **Social:** 4 to 12 feet (1.22 meters to 3.66 meters).
- **Public:** 12 to 25 feet (3.66 meters to 7.62 meters).

It's worth pointing out that each zone is defined as having two phases: close and far. The phases are defined with different distances, and accompanied with reasoning as to what one might expect of interpersonal communication in the phase one might find himself/herself in. Take for example the far phase of the public zone, which is 25 feet or more. This is, according to Hall, the distance that is automatically set around public figures - as a defensive mechanism. It could also be interesting to link each zone and phase to one of the eight distances of

vocal shifts that Hall and Trager defined. In the intimate zone, close phase, one can expect the first vocal shift (whispering), and in the far phase of the public zone, which is 25 feet or more, one of the last distances and vocal shifts would be reasonable. Hall has not himself explicitly linked each zone and phase with any of the vocal shifts.

The discrete zones of distances which Hall describes are just one of three factors that influence how people use the micro-space surrounding them. Fixed-features include the immobile properties of the space, such as the layout of buildings, walls, windows, and doors. Semi-fixed features include elements that are not fixed to one position, or rather objects that can be moved, such as chairs, beds, tables, etc. Hall discovered that the arrangement of elements in the semi-fixed space affects our perception and use of personal space, where the layouts can be sociopetal or sociofugal. Sociopetal refers to an arrangement of semi-fixed features where people can see and interact with each other, while sociofugal refers to an arrangement of semi-fixed features where people can have some privacy. These terms were coined by Humphry Osmond in his research into *socio-architecture* in his effort to improve patient setting, especially in mental hospitals [82].

In today's world, the majority of devices are blind to the presence of other devices, as well as semi-fixed- and fixed- features of the room they're located in. Though it is possible for devices to know that other devices are nearby, by e.g. using Bluetooth, they can't tell whether the other device is in the same room or not - and according to [62], this is where proxemics can help.

[62] argues that we need to operationalize proximity for Ubicomp, meaning we need to be able to measure proximity. To help approximation, [62] and [83] provide five dimensions of proxemic interactions, which are most relevant to operationalizing proxemics in Ubiquitous computing.

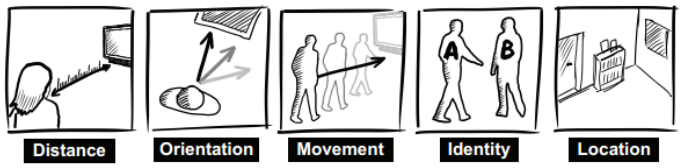

Figure 5.1: Proxemics dimensions, taken from [62].

Distance between entities is fundamental to proxemics, and can be either continuous or discrete. An example of discrete distance would be to define proxemic zones, each with its own meaning when interacting with the system. Researchers at Fraunhofer IPSI created an ambient wall (*Hello.Wall*) with clearly defined

proxemic zones (Interaction, Notification, Ambient), which would help offices at different geographical locations increase group cohesion, and make informal communication a possibility [53].

Orientation between entities can be continuous (e.g. pitch/roll/yaw angle), or discrete (facing towards or away from the object).

Movement captures the movement and orientation of an entity over time, resulting in actions depending on for example the speed and direction of the entity.

Identity uniquely describes the entities present. This can vary from exact identification of an entity, to less detailed where i.e., only the entity type is identified. At the bare minimum, entities can be distinguished from one another - where no attributes or information is associated with them.

Location characterizes the context of location, e.g., home or office. It can also capture when people, or entities, cross certain thresholds such as entering a room. According to [62], location is important, as the other four dimensions might depend on contextual location.

Systems monitoring the nearby entities, whether by capturing video, voice, or simply their identity and presence, often lead to privacy issues. It's interesting to see how proximity can be used to balance between awareness and privacy, as with the *Media Space* system Saul Greenberg and Hideaku Kuzuoka experimented with in the late 1990s. This media space system uses proximity to control an always-on audio/video link between co-located colleagues, which provides awareness of a colleague's presence and activities. In their work, they have identified several issues regarding privacy, such as: "We need better ways to mediate privacy on the communication channel, perhaps by altering the quality of service", and "We have to judge whether surrogates really do ensure reasonable level of privacy" [84]. One of their suggestions is to use progressive distortion of the video link as a function of proximity, with pixelization algorithms.

5.1.2 Contextual QR-codes

A QR-code is usually considered a static entity always exposing the same information to everybody, but a contextual QR-code is computed to provide specific information, or perform certain actions, depending on a particular context. Rouillard argues that barcodes are still regarded as an interesting option because of the basic technology, and simplicity of the concept - even though RFID and NFC are considered as the latest generation in the domain of object identification [85].

The idea of a contextual QR-code is to have a public part of information (QR-code), and a private part of information (the context) which is provided by the device scanning the QR-code. Figure 5.2 illustrates how [85] describes their idea of a contextual QR-code.

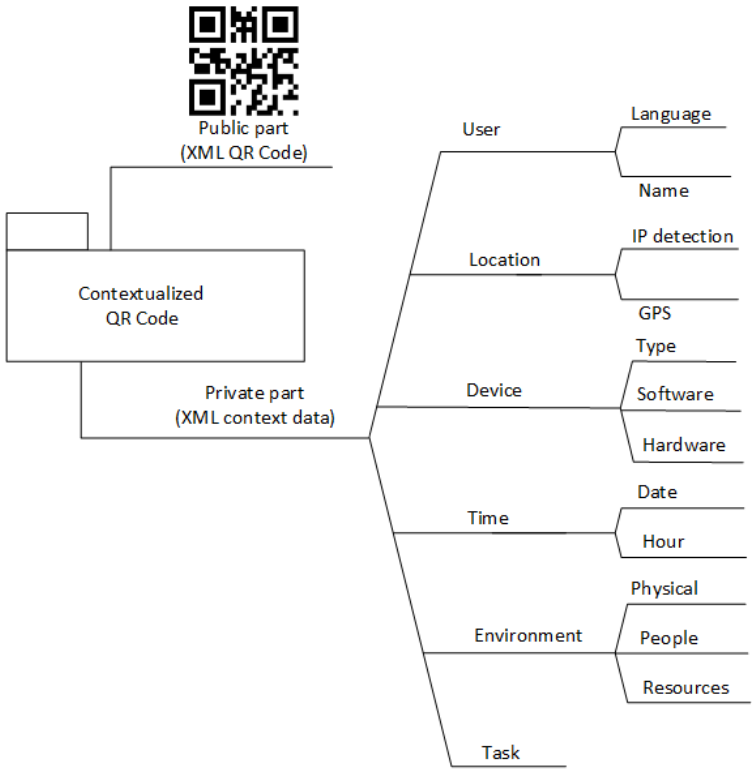

Figure 5.2: Contextual QR-code, adapted from [85].

The private part can be the user's profile, current task, location, device used, time, or environment of the interaction. When the device scans the QR-code, it merges it with private data and sends the resulting XML file to a web service which computes the code and returns personalized messages.

The simplest scenario is the *Hello World* greeting. Here, the QR-code contains:

```
<public>
   <tag>Hello</tag>
</public>
```

Should a user scan this QR-code with a random QR-code scanner using his mobile phone, the user will see exactly that message on his screen. If the user uses the contextual QR-code application, the user will see a personal greeting message on the screen, based on the input (private part). For example, if the user scans the code at 10 A.M., with the IP address 78.91.37.28, the message could for example be: *Good morning, Øyvind Høisæther from Trondheim, Norway.*

Though the example above is quite simple, the idea allows for more advanced and useful concepts, and could quite possibly work great for events in order to i.e. make information about the event as accessible as possible to the attendees.

5.1.3 Geo-fencing and spatial positioning

The number of phones sold with built-in GPS-unit and Wi-Fi/3G/4G, has risen quite dramatically the last 10 years, resulting in widespread development of Location Based Services (LBS). Examples of LBS include social networking (e.g. sharing geo-tagged messages), navigation (e.g. navigating to a certain address), e-marketing (e.g., e-coupons), business search (e.g. finding a nearby McDonalds), and traffic monitoring (e.g. being notified of potential traffic-jams on the current route). As a result of the advanced technologies that exist for location-detection such as cellular networks, Wi-Fi, RFID, Bluetooth, and GPS, LBS has become ubiquitous [86]. Building on LBS, it is possible to create virtual geographical areas known as *geo-fences*.

Figure 5.3: Geo-fence around Dødens Dal in Trondheim.

A geo-fence defines a virtual geographical area, or geo-fenced area, which can be used in many situations. In recent years, it has been used in fields such as computer software, physical security, and even in health care. In fact, research has been made to see whether geo-fencing capabilities in smartphone applications can help Persons with Alzheimer's Disease (PwAD) in their daily activities [87]. One of the ideas was to give the PwAD an hourly reminder to let them know how long they have been out of the house.

Common technologies that can be used for LBS include:

- GPS
- Cellular networks
- WLAN-networks
- Bluetooth
- RFID
- Ultrasound

5.1.4 Augmentation

In Section 4.3.2 we discussed different ways of adapting to context, and specifically mentioned *contextual augmentation*. There are two different ways of defining *contextual augmentation* where one definition from Pascoe [67] explained that we can associate data to a given context. This definition has close ties to, and can be used in Augmented Reality (AR).

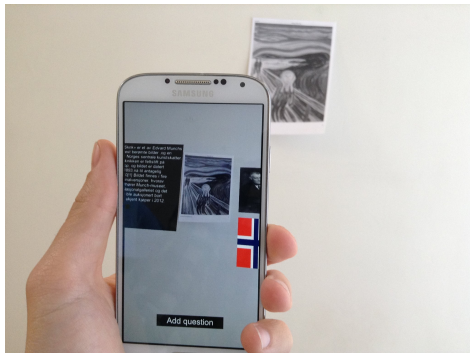

Figure 5.4: Augmented reality museum guide.

AR is a field of HCI that attempts to bridge the gap between humans and computers by combining the real and the virtual world in an effort to provide assistance to the users. AR focuses on producing virtual objects in the real world and can adapt to the real world in a number of ways [88]:

- Graphical information (2D, 3D, text) in the real world.
- Putting audio-information in the real world.
- Force feedback in the real world for interaction.

Figure 5.4 shows a prototype application of a museum guide that fellow students at NTNU have constructed using AR. Utilizing the camera on a smart-device the prototype can scan and recognize artifacts such as pictures or statues (in this case *The Scream* by Edvard Munch), and produce on-screen information. Users of the app can also link questions and answers with artifacts. This is an example of data associated with a given context that can be retrieved only while in that context.

Searching in AR is associated with context cues. The contextual information is coupled with the objects or location of the objects, and is presented in the context where it was acquired. Searching does not necessarily have to be done by some type of image recognition. With the density of sensors that a smart-device today employs, we can for example utilize GPS to detect location and accelerometer/-gyroscope and/or compass to detect where a user's focus is directed and present information to them through AR [89].

5.1.5 Services-oriented context

When we wrote about acquiring context in Section 4.3.2, we mentioned something called a *virtual sensor*. Virtual sensors provide contextual information from software applications or a service either running locally on the client, or over a network communication. There are several cases where services as a context can be beneficial. Imagine, for example, that you are on your way to an event and the mobile ticketing application gives you updates on which roads on the way that have the least traffic, or where there are vacant parking spaces closest to the venue. This can be realized by using a Service-Oriented Architecture (SOA) to communicate with the public road administration in the area and the parking companies (if they keep track of and supply accessible APIs). SOA is a software architectural pattern that allows services to be provided or consumed without specific knowledge of the implementation of that service [90].

We also spoke of gathering context from for example e-mails, calendars, or meeting notes. This way of acquiring context can help us remove ambiguity and even strengthen the quality of the context. Say for example Eirik gets an e-mail from Øyvind, where Øyvind asks him if he wants to go to a concert this weekend. If the application is proactive to the point that it analyzes the e-mail and detects that Eirik added the concert to his calendar, it can beforehand anticipate that Eirik will be at the venue at the specific time that the concert occurs. With social networks, we can also provide context application services such as simple recommender systems by detecting friends of Eirik that are going to the same event to help him make the decision to go. Even if Eirik has not purchased a ticket with the application, it knows where Eirik is headed on that given date and knows what will happen so it can adapt accordingly.

There is a blurred line between how an application can use a context to customize a service and how a service can supply context. Our last example is mostly related to the latter, though the first example with traffic and parking is closer to using context (location) to customize a service.

5.2 Technology

This section focuses on concrete technology that we can make use of to realize some of the practices we discussed in the previous section. We will also look into what kind of features mobile devices possess in relevance to context and the awareness of its environments.

5.2.1 Mobile devices

Android is the world's most used mobile operating system, powering hundreds of millions of devices worldwide [91]. We are not only speaking of smartphones when mentioning devices. Some say that Android is at the front-line of what is referred to as The Internet of Things (IoT), since Android runs on everything from e-readers, game consoles, and smart-watches to tablets and phones. Being open-source, Android has the potential for anyone to modify and run it on any device [92]. IoT was initially a concept when speaking of embedding RFID into every-day objects. With the significance of the Internet, everyday objects are constantly being enabled to communicate online [93]. Building software for a framework such as Android, which has the power to be deployed on, used on, and communicate with a wide variety of devices, makes a great foundation for a context-aware pervasive system.

A drawback of Android is that it is somewhat fragmented in regards to the software API versions, the differences in sensors, and the variety of interfaces varying from device to device. There are hundreds of smartphone models currently in sale throughout the world, and a lot of deviancy revealed when comparing the specifications between these models. One of the major reasons for this is that different brands are allowed to basically make their own decisions regarding the specifications.

Telenor released a press statement on the 5th of March, 2014, with a list of the most sold smartphones. Out of the 10 listed phones, there are six Android devices, three Apple devices, and one Windows Phone device [94]. Also on a global scale, Android is the biggest mobile platform, with 81.0% market share in the 3rd quarter of 2013, an increase from 74.9% same quarter the year before. Apple's iPhone had a market share of 12.9%, a decrease from 14.4% the year before [95].

Sorted by popularity, the most popular Android devices are:

1. Galaxy S4 4G+
2. Sony Xperia Z1 Compact
3. Galaxy Note 3
4. Galaxy S4 Active
5. Samsung Galaxy S3 4G

5.2.2 Sensors

Table 5.1 lists sensors that are supported by the Android platform, though it is not common for every phone to have all of these sensors.

Sensor	**Type**	**Description**	**Uses**
Accelerometer	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
Ambient Temperature	Hardware	Measures the ambient room temperature in degrees Celsius (°C).	Monitoring air temperatures.
Gravity	Software / Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
Gyroscope	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
Light	Hardware	Measures the ambient light level (illumination) in lux (lx).	Controlling screen brightness.
Linear Acceleration	Software / Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.

Magnetic field	Hardware	Measures the ambient geomagnetic field for all three axes (x, y, z) in microtesla (μT).	Creating a compass.
Orientation	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the *getRotationMatrix()* method.	Determining device position
Pressure	Hardware	Measures the ambient air pressure in hectopascal (hPa) or Mbar .	Monitoring air pressure changes.
Proximity	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
Humidity	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.
Rotation vector	Software / Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.

Temperature	Hardware	Measures the temperature of the device in degrees Celsius (°C). This sensor implementation varies across devices and this sensor was replaced with the Ambient Temperature sensor in API Level 14.	Monitoring temperatures.

Table 5.1: Table of supported sensors in Android.

Mobile devices and available sensors

Based on data from gsmarena.com, Table 5.2 shows which sensors each of the phones mentioned in the previous section supports.

	Galaxy S4	**Xperia Z1 Compact**	**Galaxy Note 3**	**Galaxy S4 Active**	**Galaxy S3 4G**
Accelerometer	✓	✓	✓	✓	✓
Gyro	✓	✓	✓	✓	✓
Proximity	✓	✓	✓	✓	✓
Compass	✓	✓	✓	✓	✓
Barometer	✓	✘	✓	✓	✓
Temperature	✓	✘	✓	✘	✘
Humidity	✓	✘	✓	✘	✘
Gesture	✓	✘	✓	✓	✘

Table 5.2: Overview of sensors available for selected phone models.

5.2.3 GPS - Global Positioning System

Most smartphones today have the ability to receive and make use of GPS-data. Because location-awareness is of such importance as a context in context-aware systems, we will in short terms discuss what the GPS technology is.

The development of the GPS started in the early 1970s by the U.S. Department of Defense as a military system. GPS has since then become a prime source of location-awareness, for military and civilians alike. The reason behind its success is the fact that GPS has the ability to position a GPS-enabled device outdoors, anywhere in the world, and under any weather conditions [97]. In a worst case scenario, GPS can achieve a pseudorange (an approximate distance) accuracy at a 95% confidence of 7.8 meters [96].

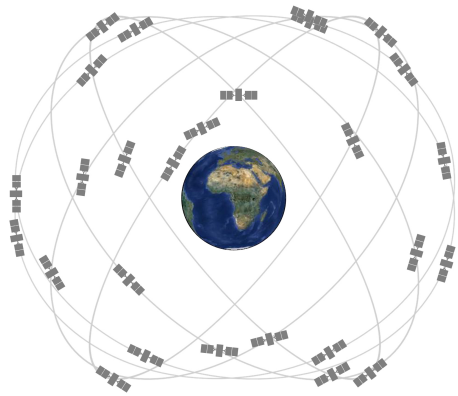

Figure 5.5: GPS satellite constellation, taken from [96].

GPS constitutes a worldwide smart space enabled by satellites in orbit around the earth. It normally consists of 24 operational satellites scattered between six orbital planes about 20 200 kilometers above the earth's surface, with four satellites in each plane as depicted in Figure 5.5. In June 2011, to improve the accuracy of GPS, the constellation was expanded, where six satellites were repositioned to make room for three new satellites [96]. The constellation provides a continuous worldwide coverage, since only four satellites are needed to provide location information. [97] argues that at an elevation angle of 10°, four to ten satellites will be visible from anywhere in the world at any time.

The system is segmented into three parts: the space segment, the control segment, and the user segment [96, 97]. The constellation previously mentioned is considered to be the space segment. The control segment is a globally distributed network used to monitor and control the GPS satellites. It consist of the master control station (Colorado, USA), monitor stations (16 locations worldwide), and ground antennas (four dedicated sites). The user segment is the GPS-receiver in equipment that can make use of the data, such as smartphones.

5.2.4 RFID - Radio Frequency Identification

Figure 5.6: Common RFID access control.

RFID has emerged to become one of the most pervasive microchips ever to be created. RFID is comparable to optical barcodes, and has its roots in advances of electromagnetism in the 19^{th} century and modern radio communication. It is a sensor technology of transponders (composed of components such as integrated circuits and antennas) called *tags* that are associated with everyday objects. Tags can, just like barcodes, contain unique product identification and be readable from a distance, and without line of sight, which makes it superior to traditional barcodes. Commercially, RFID was first used on railroad cars and for automatic collection of toll at the end of the 80s and the start of the 90s. Early on, train cars were equipped with optical barcodes as identification, though barcodes had a habit of being obscured by dirt and damaged, rendering them useless. Today RFID is used for many purposes:

- Tracking and identification.
- Payment and stored-value systems.
- Access control (Figure 5.6).
- Anti-counterfeiting.

RFID is not a term for one specific type of technology, as there exist various forms with various cost, use-cases, sizes, power requirements, and limits. We can separate between *passive*, *semi-passive* and *active* RFID-tags. The passive tag is the simplest and most used as it is inexpensive to produce and it does not require a battery, as it harvests radio-frequency energy. The range is up to 10 meters and it cannot initiate communication, only act as a response. One step up is the semi-passive tag which is alike the passive, only that its source of power is a battery that provides it with a larger range of up to 100 meters. Ranges of RFID vary by operating frequency. This can be used to control how tags interact with each other [30, 98, 99]. The distances in the list below are measured by passive read.

- Low Frequency (LF): 120-140 KHz (10-20 cm).
- High Frequency (HF): 13.56 MHz (10-20 cm).

- Ultra-High Frequency (UHF): 868-928 MHz (3 meters).
- Microwave: 2.45/5.8 GHz (3 meters).
- Ultra-Wide Band (UWB): 3.1-10.6 GHz (10 meters).

5.2.5 NFC - Near Field Communication

NFC is a short-range wireless communication technology that has evolved from RFID [100]. It can communicate by responding or initiating contact, but is limited to a very short range of interaction. Technically NFC is an active-RFID technology. Although active-RFID theoretically has a maximum capacity of up to 100 meters, NFC runs on HF which shortens its range considerably [99].

NFC was created by Sony and Philips. The companies reached an agreement on the development of NFC in 2002, and in 2004 the NFC Forum was established. The NFC Forum is where the current technical and business collaboration related to NFC is realized within [101]. In 2006, the technology architecture was unveiled and the first five Forum specifications were announced [102].

An NFC-enabled device can both act as information storage, or an NFC-reader. As a reader, it can extract information from NFC tags and display the information on the screen - or silently do processing in the background. As information storage, the device can be used as a digital storage for i.e. contacts, documents, or credit card information [100].

Other important advantages:

- NFC makes establishing ad-hoc connections between devices simple and easy.
- The communication between devices occurs at a short distance (few centimeters), and is completely lost at any distances over 20 centimeters [101].
- The technology is compatible with RFID, RFID-tags, and contactless smart cards [100].

A device with NFC can operate in three modes:

- Peer-to-peer to establish a two-way communication between two devices.
- Card emulation, allowing the phone to emulate a contactless card.
- Reader/writer, the smartphone can both read passive tags, and write to them.

According to [103], NFC is the main technology that improves mobile ticketing services. That might be correct, but only applies to NFC-enabled phones, which leaves iOS-users out of the loop for now.

Disadvantages

Even though Android's support for NFC is the best in the market, it's incomplete, and there are several issues with regards to mobile ticketing and NFC [103].

- Android Beam / Peer-to-Peer can't be used to establish a real two-way communication.
- Ticket-cloning, both pre-validation and post-validation, where two users are sharing the same ticket.
- Man-in-the-middle attack, in which a third entity hijacks the communication between two devices, in this context the ticket owner and validator.

Man-in-the-middle attacks can be avoided by encrypting the exchanged data, as well as the mutual authentication between the communicating devices. Ticket-cloning problems are harder to solve efficiently. A solution is to have the validator check real-time, in e.g. a central database, whether or not the ticket has been validated before [103].

Advantages

There are some important benefits by introducing mobile ticketing, with NFC:

- Tickets are more durable (e.g. they can't get damaged/dirty like paper-based tickets can).
- Users are less likely to lose tickets.
- The entire system is more eco-friendly and cheaper than a system with paper-based tickets.

5.2.6 IR - Infrared

We previously discussed *The Active Badge Location System* which was credited as the first context-aware system. The system was built to locate personnel by wearing an *active badge*. The badge was a beacon which transmitted Infrared (IR) signals. The signals could then be read by distributed sensors to pin-point where people were.

IR is positioned between microwaves and visible light in the electromagnetic spectrum. William Herschel discovered IR in 1800 by observing that the temperature of visible light increased from blue to red, and even more so at the end of the visible spectrum. Any object that radiates temperature also emit IR-radiation. Astronomers tend to divide IR into three parts: near, mid, and far, although these are not regulated or agreed upon. Far IR-radiation is thermal, while near IR-radiation is not even noticed by humans. The latter is usually found in consumer electronics for wireless communication [104–106].

IR transmitters have been officially supported by Android since API version 19, although some manufacturers added support for IR through their own SDKs much earlier. Currently the official support is strictly transmitting, but manufacturers such as HTC has the potential to receive or "learn" IR patterns through their respected SDK [107, 108]

There are several limitations with IR that are addressed much more delicately with wireless radio communication [109].

- Limited range.
- Bandwidth is not regulated.
- Requires direct line of sight.
- Generally more power-consuming than radio-waves, by comparing the range of transmitting.

5.2.7 Bluetooth and BLE

Bluetooth

Bluetooth was the nickname of Harald Gormsson, who was the king of Denmark in the 10th century. The Bluetooth technology adapted the name since Harald 'Bluetooth' Gormsson was known for uniting Denmark, while the technology had a mission to unite devices [110]. Jim Kardach, one of the founding members of the Bluetooth Special Interest Group (SIG), came up with the name, remarking: "Harald thinks notebooks and cellular phones should seamlessly communicate".

Bluetooth has, since the SIG was established in 1998, become one of the most ubiquitous connection technologies, where products shipped with Bluetooth surpassed 2.5 billion in 2013 and continues to increase [110]. Based on wireless radio frequencies, Bluetooth is used to create a Wireless Personal Area Network (WPAN) to share data and information between two or more paired devices. It is a so called short-range communication technology, where devices only can be reached in the vicinity of each other. The range of a Bluetooth device is specified by its classification [110]:

- Class 3: up to 1 meter.
- Class 2: Most common in mobile devices - A range of 10 meters or less.
- Class 1: For industrial uses that require a range of up to 100 meters.

The short operational range of Bluetooth makes it a preferred solution in contrast to Wi-Fi when looking at the power consumption, where a class 2 Bluetooth device only uses 2.5 milliwatts of power. The technology operates in 2.4 to 2.485 GHz Industrial, Scientific, and Medical (ISM) band in a token topology, where devices are permitted to transmit by time-division multiplexing. The bandwidth that Bluetooth has at its disposal varies between the versions of Bluetooth, where version 2.1 have a theoretical bandwidth of 3 Mbit/s and a practical bandwidth of 2.1 Mbit/s [111, 112].

BLE - Bluetooth Low Energy

BLE, marketed as *Bluetooth Smart*, is a WPAN operating in the 2.4 GHz ISM band. It has been designed to prioritize low energy consumption instead of enhanced data rates [110]. BLE defines 40 physical channels, where 3 channels are used for advertising, and the remaining 37 for the communication itself, to

achieve a data rate of up to 1 Mbps [113]. In addition, it uses frequency hopping transceiver to reduce interference, allowing it to co-exist with other wireless technologies such as Wi-Fi (a property BLE shares with ZigBee) [114].

BLE is connection oriented, where two devices wanting to exchange data must establish a fixed connection before transmitting and receiving data is possible. There are 5 states, or operating modes, for BLE devices, as seen in Figure 5.7 [113].

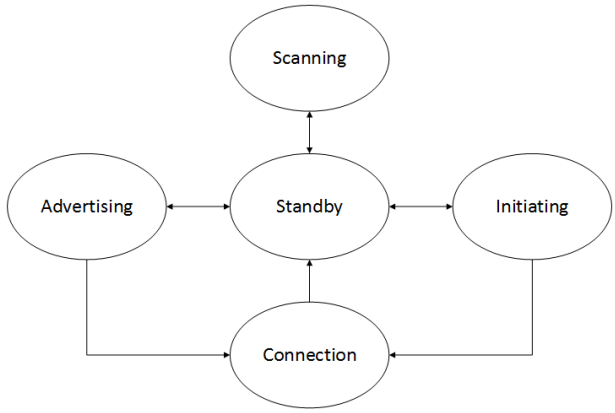

Figure 5.7: State diagram for BLE devices, adapted on [113].

BLE enables the IoT, as connections between devices can be established very quickly, it supports star topology (see Figure 5.8), it has low power consumption, and the ease of exposing state. A BLE device could run for 4 years on a single coin cell, if it were to consume $\sim$100 μAh per day. Though it consumes very little energy, it can still send useful data, and its operational life is solely determined by the battery technology.

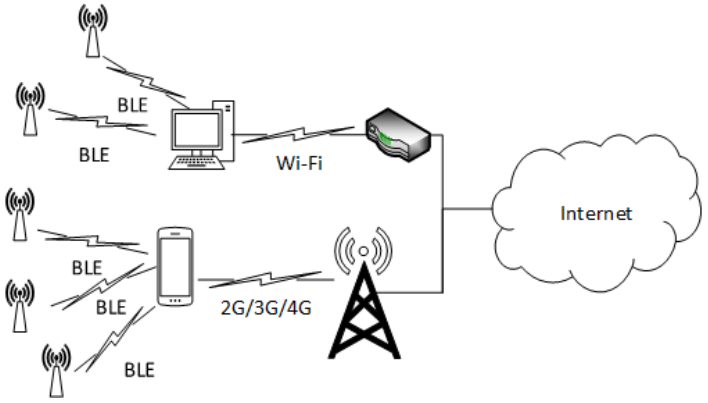

Figure 5.8: Star topology with BLE devices.

Examples of uses include, but are not limited to:

- Control:
  - Turn off certain appliances when electricity is expensive.
  - Turn lights on/off.
  - Control the heating / ventilation system.
  - Unlock/lock doors.
  - Turn the burglar alarm on/off.
  - Operate the TV.
- Proximity Detection:
  - I'm in the car.
  - I'm in the office.
  - I'm in the meeting room.
  - I'm in the movie theater.
- Presence detection:
  - Turn the lights on when I walk around the house.
  - Automatically locks the doors when I leave home.
  - Turns the alarm off if I'm already awake.
- Send data from anything:
  - The coffee is finished.
  - Eirik is calling.
  - The temperature at home is 23 °C.
  - The front door is not locked.
  - The alarm is active/inactive.
  - The baby is crying.

BLE does, just like the original Bluetooth standard, define several profiles. These

profiles are specifications on how a device works in a particular application, called General Attribute Profile (GATT). GATT defines two roles: client and server, and are not necessarily tied to a particular Generic Access Profile (GAP) role. GATT and Attribute Protocol (ATT) are not tied to a specific transport type, and they can be used in both Basic Rate (BR)/Enhanced Data Rate (EDR) and Low Energy (LE) [115].

Advantages of BLE:

- BLE has low maximum peak current (12-15 mA), and is operable using CR2302 Coin Cell [114].
- Supports >2 billion connections.
- BLE provides better coexistence with other wireless technologies, such as Wi-Fi.
- Does not require line of sight between connected devices, which IR does.

5.2.8 Wi-Fi

Originally, Wi-Fi was a way for users to stay wirelessly connected to the Internet, either in their home or in the enterprise. When the Wi-Fi technology first got widespread, it was mainly used for surfing the Internet, checking emails, but that trend has since shifted to where Wi-Fi is also used for content consumption such as streaming of music, videos, and podcasts. The demand for high-definition video, as well as video content in general, is a challenge for the Wi-Fi 802.11n based networks, because of interference in the 2.4 GHz band. As such, 802.11ac was developed, which is the next evolution of the Wi-Fi standard that promises to allow multiple high-definition streams simultaneously [116].

Wi-Fi is not only used to stay connected to the Internet wirelessly, it has also been used to develop indoor positioning systems. Indoor positioning systems based on Wi-Fi signals often employ a technique called *fingerprinting*. Fingerprinting makes use of the signal strength, and is divided into an offline and online phase. In the offline phase, which always occurs before the online phase, where signal strengths from all n detected Wi-Fi access points are collected, at fixed indoor reference positions. An n-dimensional signal strength vector is obtained, for each reference position, as a fingerprint. These fingerprints form a database called *radio map* for the indoor space.

In the online phase, the position of the user is estimated based on the n-dimensional vector that was obtained in the offline phase, at the current location. The current signal strength vector is then compared to the fingerprints of the *radio map* database, and the best matching reference position is returned as the indoor user location [117].

Wi-Fi can also be used by users to share files and data, using technology such as *AirDrop* by Apple, which is peer-to-peer based [118].

6

Pling: A prototype application

"It always seems impossible until it's done."

– Nelson Mandela, *revolutionary, politician, philanthropist*

Thomas is sitting at home on a Saturday afternoon, wondering what to do later on the same night. He opens up the "Pling" app on his smartphone, and sees that some of his friends are going to a concert only 1.5km from his home. The app tells him there are still some tickets left, in case he wants to go. He decides to purchase a ticket, and his friends are instantly notified that he also will be attending the concert.

When making his way to the venue, the app helps him get there as fast as possible, letting him know whether there's traffic ahead, or if there's simply a faster route, in case he is walking. This feature can of course be silenced by the push of a button, if no suggestions are required from the app.

As Thomas arrives the event, the app displays a map with indications of where the validation area is, as well as his own position and other Points of Interest (POI)s - making it easy finding his way around the event area. Thomas is also notified that this particular event allows for validation of tickets using BLE, meaning he will not have to find his ticket with the QR-code when approaching the validators. This is the first time Thomas has ever used BLE for validation, so the app starts playing an animation showing how the process works. Confident in this new way of validating tickets, Thomas puts his phone back in the pocket and walks past the

validator. A green light shows up when he is close to the validator, and is allowed to pass through to the event.

After a while, Thomas decides to purchase something to eat and drink, and starts walking towards the closest kiosk or POS, which he previously saw on the map of the event area. When he's nearby the kiosk, the app notifies him that the price list is available in the app, and that he can place an order from within the app. He's also told that the app will notify him when the items are ready to be picked up, helping him avoid waiting in a dreadful queue. While browsing the items, he selects the items he want, and with a click of a button an order is placed. While waiting comfortably, the app notifies him that the items are ready to be picked up. He walks up to the (almost) queue-less POS, validates his purchase with the app, and picks up his items.

6.1 About the prototype application

Not to fool the reader, we must emphasize that the story above is not what the *Pling* prototype ended up being. The story, or scenario of use, is a future vision of how we wish *Pling* ends up being used.

It has been important for us that the prototype looks and feels like a real ticketing application, one that you might find on the Google Play marketplace. As such, we have tried to make the GUI as good as possible, so that the users can be led to believe that it's a fully working ticketing application, though lacking support for purchasing tickets/food/beverages.

Further on, we will see that the validation process is a key element on which we focused to implement in the application. This is because the approaches and technology we used to accomplish validation of tickets in *Pling* has the potential to realize the important features in the story that introduced the chapter.

6.2 Actions and navigation

This section explains how the application works, which actions can be taken, and how navigation works within the app.

6.2.1 Starting the app

When starting the app, a splash screen is displayed while the app is downloading the upcoming events, as well as tickets the user might have purchased, as seen in Figure 6.1. Figure 6.2 shows the main view of the app, which is the list of upcoming events. Figure 6.3 shows the main menu of the app, which is accessible in two ways. The first way is to click on the top left icon of the app, the second is to swipe with a finger from the outer left of the app, to the right.

Figure 6.1: The app has just started and the splash screen is visible.

Figure 6.2: Main view of the app, listing the upcoming events.

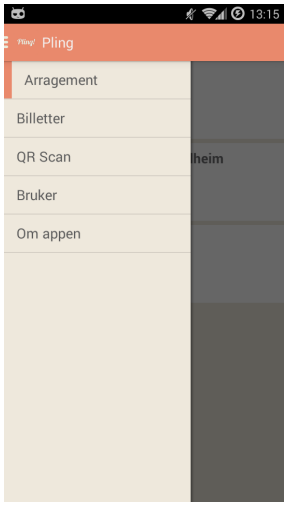

Figure 6.3: Main menu of the app.

While at the splash screen, before trying to access the backend API, the app will ask the user for permission to turn on Wi-Fi/mobile data if no network access has been detected. Once the splash screen has finished loading all the necessary data from the backend API, the app checks whether the user has GPS enabled. If GPS is disabled, the user is presented with an overlay asking for permission to

turn it on, where also the user's incentives to why he or she should turn it on is shown.

Figure 6.4: The app is requesting Wi-Fi/mobile data access permission.

Figure 6.5: The app is asking the user to enable GPS.

Figure 6.6: User login overlay.

User login

When starting the app, if the user is not logged in, the overlay in Figure 6.6 will be presented, prompting him or her to log in. Once the user is at the main view, the menu can be opened and *Bruker* can be clicked to log in as another user by inserting new user credentials. It is not possible to create a new user in the POC, but everyone who will be trying out the application will get their own demo-user. The next time the app is opened, the user will be automatically logged in as the user-data is stored on the device.

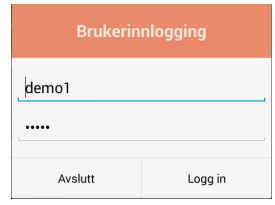

Figure 6.7: The user login dialog.

For implementation reasons, the application is restarted when the user has logged in successfully, since all data required by the app to function properly is downloaded while the splash screen is visible.

6.2.2 Approaching the event

If the user has enabled GPS, and has the application running in the background/-foreground, the application will notify the user when he or she enters the event area based on GPS location data.

Figure 6.8: Notification has been fired off.

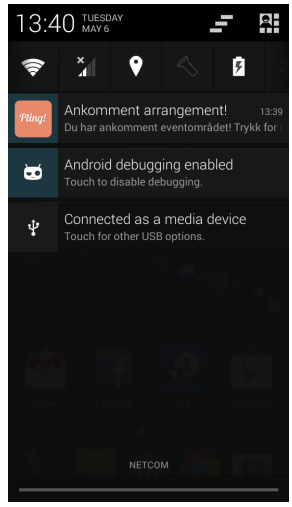

Figure 6.9: User is looking at the notification.

Figure 6.10: User is looking at event information triggered by the notification.

At this moment, it would be wise of the user to enable Bluetooth by clicking on *Se valideringsinstruks*, if Bluetooth is disabled. This would lead to an overlay being displayed, letting the user know how validation occurs, and asking the user to enable Bluetooth.

6.2.3 Ticket validation

When approaching the validation area, the user should make sure that Bluetooth and Wi-Fi/Mobile data is enabled. When entering the event area, he or she will be requested to enable Bluetooth. If Bluetooth is not selected to be enabled when entering the event area, Bluetooth should be enabled manually when approaching the ticket validation area.

When the device is within range of a beacon, an overlay will let the user know that he or she has entered an interactive zone. If the device is within immediate proximity of a validation machine, the app will attempt to have the ticket validated, and shows another overlay.

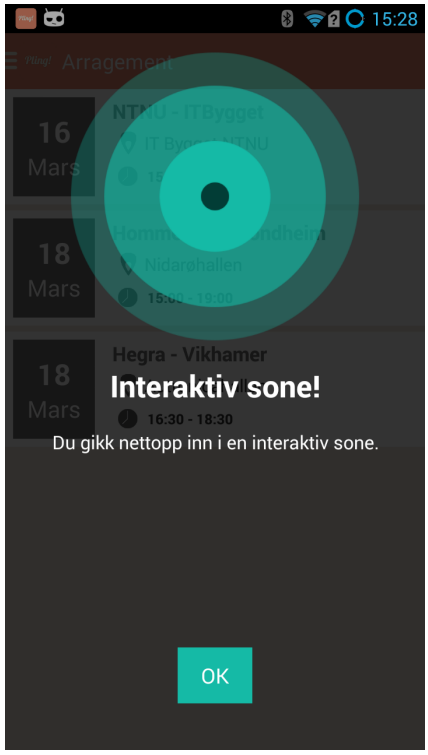

Figure 6.11: The user has entered an interactive zone.

Figure 6.12: Ticket validation has started.

Once the app has attempted to validate the ticket, the user is informed whether or not the ticket was successfully validated, as seen in Figures 6.13 and 6.14.

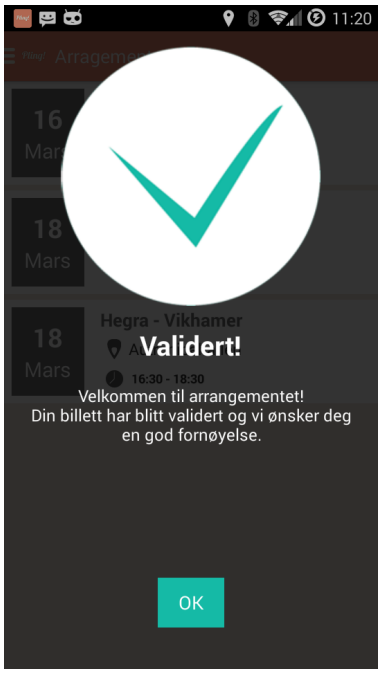

Figure 6.13: Ticket has been successfully validated.

Figure 6.14: Ticket validation failed.

Notice in the figures above how the user was not originally looking at his ticket for this particular event. The application knows which event the user is attending, as the BLE beacon advertises which event it is requesting the application to validate for.

This solves the potential bottleneck in mobile ticketing systems, or paper-based systems, where users need to have their ticket visible for scanning. In *Pling*, they only need to start the app and make sure Bluetooth is enabled, and *Pling* will take care of the rest.

6.2.4 Scanning a QR-Code

If a user wants to scan a QR-code at the event, the user simply opens up the main menu and clicks on *QR Scan*. Scanning starts immediately once the user makes this choice. If he or she wants to scan another QR-code, the *Scan* button at the bottom of the screen would need to click.

Validation instructions

Close by the place for validation, there will most likely be a QR-code that says "Scan for valideringsinstruksjoner :)", which will help the user understand the whole validation process, should it be scanned. If the user has not enabled Bluetooth, he or she will simply not be able to validate his ticket. When scanning this QR-code, there are two outcomes:

1. The overlay that is displayed says that Bluetooth is activated, if Bluetooth is indeed enabled.
2. The overlay that is displayed asks the user to activate Bluetooth.

Figure 6.15: User is scanning a QR-code.

Figure 6.16: User is looking at ticket validation instructions, and asked to activate Bluetooth.

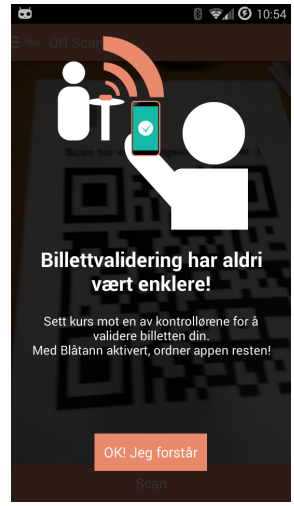

Figure 6.17: Bluetooth is already activated, but the instructions are still visible.

Event information

One of the QR-codes supported by *Pling* takes the user to event information for a certain event, as seen in Figures 6.18 and 6.19.

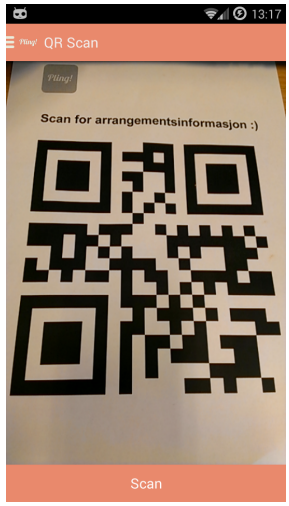

Figure 6.18: User is scanning a QR-code for event info.

Figure 6.19: User is looking at event information after scanning a QR-code.

The information presented to the user when scanning this type of QR-code will always be up to date, as it only contains the unique ID of the event. The event information is, once the QR-code is scanned, downloaded via the backend API. This gives the event organizers an advantage when advertising for the event, as they don't have to update the QR-code if some of the event information should change later on.

6.2.5 Receiving a message

At any time, the event organizers might want to distribute a message to all event attendees, or people nearby a certain beacon. Figure 6.20 shows a message that says "The event will begin shortly. Please find your place in the auditorium".

A big advantage by using beacons to distribute messages is that the users do not necessarily need mobile network reception, or Wi-Fi access (Internet access in general). In some buildings, reception might be poor, or it might not even be possible to get any bars at all. Using a beacon will help overcome these potential issues, given that the app has pre-loaded all potential messages the event organizers plan on sending to the event attendees.

In this POC, *Pling* does require internet access to show messages to the user.

When broadcasting a message, *Pling* only receives the unique message ID, and instantly downloads the complete message from the backend API based on the ID.

Figure 6.20: The user received a message.

7

Pling: Proof of Concept

> *"Once excluded the impossible, whatever remains, however improbable, must be the truth."*
>
> – Sherlock Holmes, *The Sign of Four (1890)*

After carefully reviewing our options, and the detailed theoretic foundation of the previous chapters, the process of implementing and building the POC did not prove to be a tedious task. We would like to point out that the majority of the beacons functionality was pre-implemented at this point thanks to the signal strength experiment detailed in Appendix D. The application spans 3077 lines of code divided among 62 classes and 15 packages. In addition, the XML-based layout files are in total 715 lines.

In the aftermath of the development of the POC, before commencing the user testing (Chapter 8), the application went through a thorough Quality Assurance (QA) process to make sure that it would function properly. During the QA process changes to the application were made continuously. From this process, with valuable input from fellow students who were kind enough to test, we learned a great deal on how to improve the UX of the app.

Throughout this chapter, we will take a deeper look into how our context-aware POC, *Pling*, is implemented.

7.1 Geo-fencing

Geo-fencing has been implemented in a very straight-forward way. Events are associated with venues, and each venue is accompanied with a GPS coordinate.

Based on the GPS coordinates of the venue, and the GPS position of the device, we can easily determine whether or not the device is within i.e. 100 meters radius of the event location.

Pling has a service running which requests a GPS location update every five seconds from Android, and checks every geo-fenced area whether or not the device is less than, or equal to, 100 meters in distance from the geo-fence location. Calculations are done using built-in support for distance calculation between two geographical locations in Android.

Should the user enter the event area, a notification will be fired, and an activity showing event information etc. will be displayed when the user clicks on it.

7.2 Contextual QR-Codes

To make information easily accessible for all users of *Pling*, we have added the concept of contextual QR-codes. Each QR-code contains a tag and an event-ID, which triggers a certain action within the application when scanning it.

We have added the support for the following tags:

- **<eventInfo:eventId>**: Event information for the event will be displayed.
- **<priceList:eventId>**: A list of prices is displayed for the event, containing whatever type of items that can be purchased at the event.
- **<validationInstructions:eventId>**: Displays instructions on how the validation process is carried out.

The QR-codes are printed by the event organizer on A4 paper, and placed around at the event location. An example of one of these QR-codes can be seen in Figure 7.1. Seeing as we use geo-fencing, we do not necessarily need the *eventId* specified in the QR-code as we know which event the user is scanning the QR-code for, but in this way it makes it possible to advertise for the specific event prior to the event happening.

The QR-Code scanner was implemented with the use of Zbar SDK for Android, which can be found at *http://sourceforge.net/p/zbar/news/*.

Figure 7.1: Example of a contextual QR-code containing the text <validationInstructions:1>.

7.3 BLE, beacon, and advertising

Validation of tickets has been the main focus of our ticketing application. The POC estimates the distance between the device and the validation machine, or beacon, and at an approximated range, sends a message to the backend API that the app would like to validate a certain ticket.

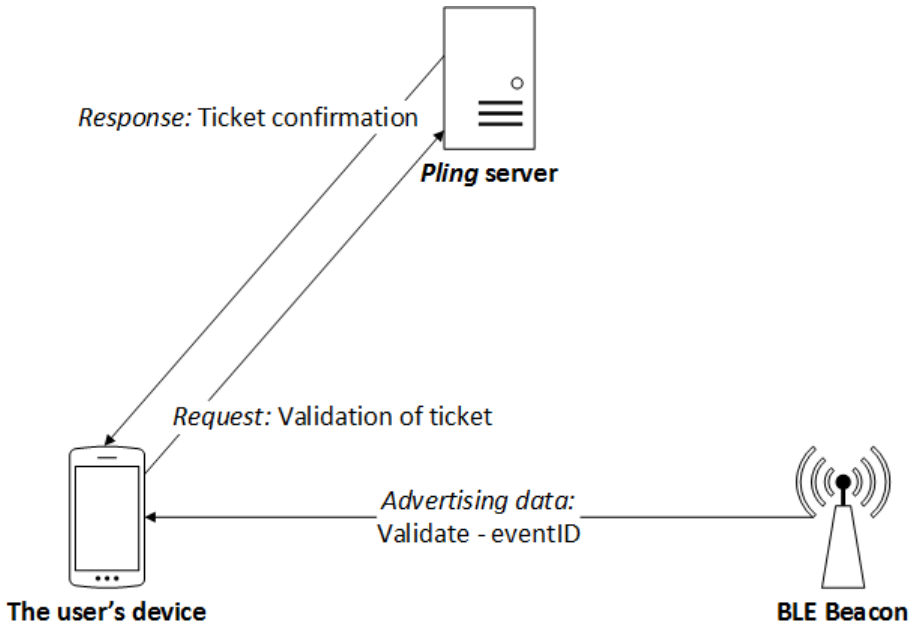

Figure 7.2: The physical layout of the validation process.

The choice to use BLE for implementing ticket validation came naturally after reviewing context-awareness, approaches, and technologies. BLE enables a much more seamless experience than any other means of connecting to a service. Looking back at our discussion from Chapter 4, a context-aware pervasive solution should preferably only make a temporary connection, avoiding a bothersome *pairing* or *connecting* process. The user should also not have to be concerned with any underlying implementation details or user-manual to make use of the system. The system should be self-explanatory or so straight forward that usage comes subconsciously, or at least so easy that people quickly learn to handle the validation process to the point where they cease to be aware of it. This is where BLE showed potentials in comparison with other technology and approaches.

- Wi-Fi or classic Bluetooth requires tedious pairing or connecting. Advertising mode in BLE needs no pairing or connecting. Scanning is enough to receive data from a beacon, which only requires Bluetooth to be enabled on the device.
- QR-validation forces the user to find the ticket on the phone and actively show or scan it. By defining interaction zones for BLE, in theory, the user does not even have to get the phone out of his or her pocket, let alone find the ticket on the phone.
- NFC was also an option, but its weaknesses lies in the limited device support, which only enfolds the Android ecosystem. NFC also has a limited range for which the user would need to be near the validation-device at such a distance that would become cumbersome.
- BLE opens the possibility of validating multiple people at once.
- BLE has the potential to feature a wider variety of context-aware solutions in the space where it is situated in than any other. This is because of the ease of only having to enable Bluetooth on the user's device.
- Although IR does not require pairing, it is not sufficiently supported by manufacturers to be taken into consideration. In addition, IR also require direct line of sight.

Through this section, we will get a detailed look on how the advertising data that we send from our beacon to the user's device is constructed.

7.3.1 Beacon

For the POC, we have developed a BLE-beacon to broadcast, or advertise (as Bluetooth prefers to call it), data for validation of tickets (and more). The beacon (described in Appendix A) is built out of a Raspberry Pi with a Bluetooth v4.0 USB-dongle, incased in black as Figure 7.3. To easily control and configure the beacon, such as changing the Advertising Data (AD) payload, we wrote a web-frontend which we have dubbed *The Beacon*

Figure 7.3: The BLE Beacon.

Control Center. The control center communicates with a PHP based Representational State Transfer (REST)-API that utilize BlueZ (http://www.bluez.org/), the Linux Bluetooth protocol stack, to alter the beacons settings. This is further explained in Appendix B.

A question some might ask is why we are using a proprietary Raspberry Pi beacon to broadcast data instead of just having another BLE-enabled Android device do the same. Before continuing, we would like to be clear about the concept of roles and responsibilities that BLE-devices can have. The different *key players* in the Core Bluetooth are:

- **Central role**: consumes data.
- **Peripheral role**: produces data.

This means that the device that runs in the *central role* scans for advertising data, while the device in the *peripheral role* provides the advertising data by sending out data packets. This is in terms comparable to client- and server- roles of standard network communication [119, 120].

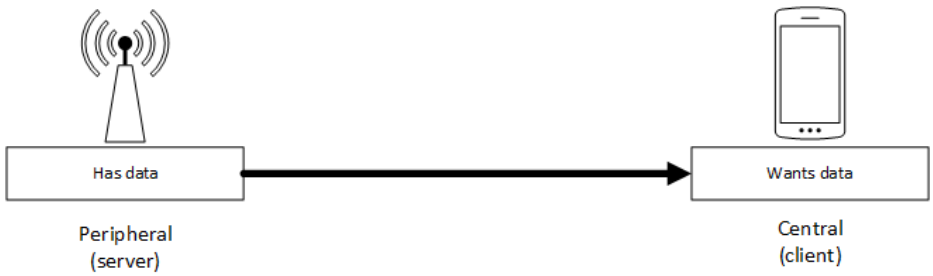

Figure 7.4: Central and peripheral roles of BLE, adapted from [120].

Unfortunately, the Android operating systems is currently (API level 19), only able to be put in the *central role* of a BLE-connection [119]. This means that an Android-device is not able to advertise a service or data, only consume these or connect to the services that are advertised to it. This is why the Raspberry Pi beacon is essential.

7.3.2 Bluetooth LE advertising packet

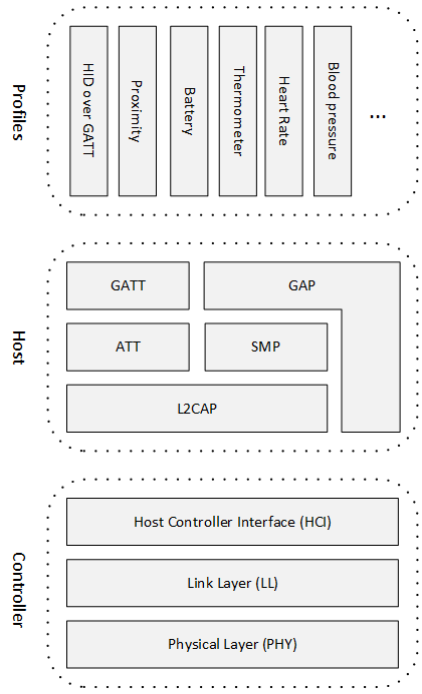

Figure 7.5: Bluetooth LE stack, adapted from [121].

Derived from the Bluetooth core specification [121], the BLE advertising data packet is structured as described in Figure 7.6. The layout follows, and is set, in the *controller* and *host* part of the Bluetooth protocol stack of Figure 7.5.

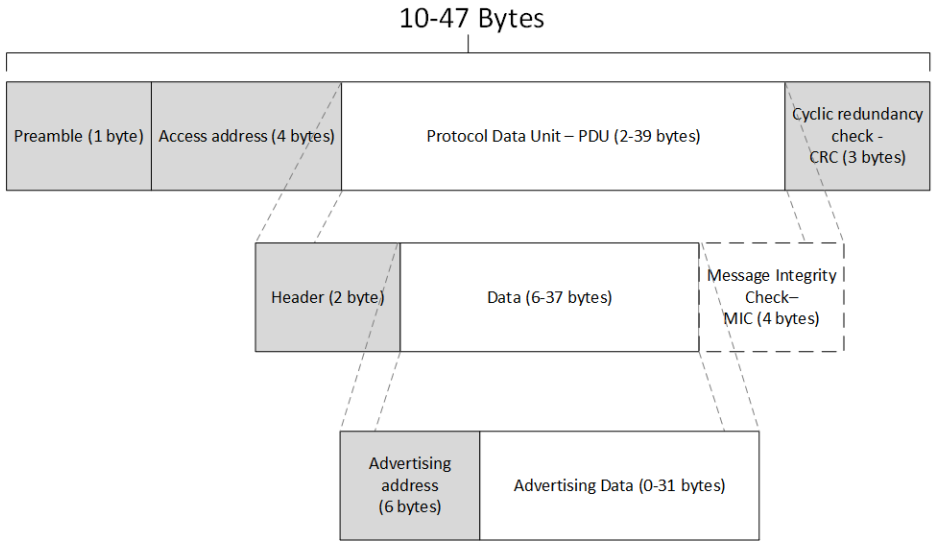

Figure 7.6: BLE data packet structure.

- **Physical Layer (PHY)**
  - **Preamble**: The preamble is used to compensate for variations in the Direct Current (DC) voltage of signals. It consists of a fixed 4-symbol pattern of 0's and 1's, where the pattern is governed by the Least Significant Bit (LSB) of the access address. For example 1010, if the LSB would've been 1.
  - **Access address**: The access address is an identification of the communication on the physical link. It is also used to exclude or ignore communication that happens on the same physical link or proximity. Active LE devices use a randomly generated 32-bit value, while advertising LE devices use a fixed access address.
- **Logical Link Control and Adaptation Protocol (L2CAP)**
  - **Protocol Data Unit (PDU)**
    - **Header**: The header of the PDU carries logical link identifiers and logical transport. This determines the type of broadcast carried over the physical channel.
    - **Advertising address**: The advertisers (beacon) device address.

    * **Message Integrity Check (MIC)**: The MIC is an optional field that can be used for the authentication of the data in the PDU.
    * **AD**: The data payload to be sent.
  - **Cycle Redundancy Check (CRC)**: The CRC is generated and used to check the integrity of packets.

7.3.3 AD - Advertising Data

We are generally not concerned with the packet details that exist in the PHY, since these are set by Bluetooth automatically. Our concern only amounts to the PDU at the L2CAP layer. Figure 7.7 show how the AD should be structured in the PDU.

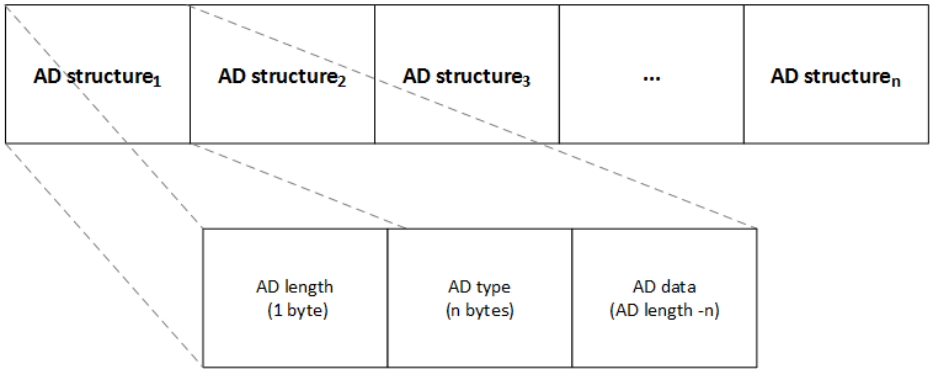

Figure 7.7: Advertising data structure.

Each structure in the AD is built in three parts.

- **AD-length**: The length of the current structure.
- **AD-type**: The type of data that is contained in the structure. These are assigned values called GAPs which are specified in a supplement to the Bluetooth core specification.
- **AD-data**: The data payload of the current structure.

7.3.4 Our packet structure

We have loosely based our packet structure on the specification of the *Apple iBeacon* [122, 123]. Our structure is depicted in Figure 7.8 and described in details below.

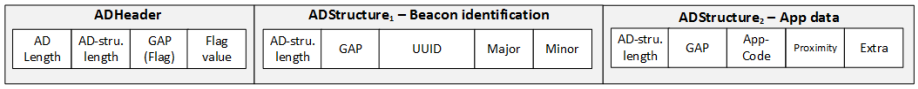

Figure 7.8: *Pling* BLE packet structure.

- **AD-header** (4 bytes)
  - **AD-length**: This first byte in the header defines the length of the entire AD packet.
  - **AD-structure length**: This second byte defines the length of the current AD-structure (in this case the header).
  - **GAP**: The generic access profile defines the characteristics of what the current AD structure contains.
  - **Flag value**: Since our header and its corresponding GAP define a flag, this value indicates the characteristics of this flag. The flag, in our case, defines the advertising mode of the beacon.
- **Beacon identification** (22 bytes)
  - **AD-structure length**: (see AD-header).
  - **GAP**: (see AD-header).
  - **Universally Unique Identifier (UUID)**: The UUID is a unique 16-byte hexadecimal identifier that we use to distinguish between the services that a beacon provides. For example, if two beacons use the same UUID, it means that these two can provide the same service or is associated with the same content-provide e.g. a venue.
  - **Major & Minor**: Major and minor are each two bytes. These are used to identify and group beacons. This means that two beacons that have the same UUID, might also be in the same group (e.g. share major), but can be uniquely identified by their minor value. As such, we can have multiple beacons doing the same tasks in groups, such as ticket validation, but still have the ability to differentiate between them.

- **App-data** (min. 5 bytes)
  - **AD-structure length**: (see AD-header).
  - **GAP**: (see AD-header).
  - **App-code**: The app-code defines the function that the app should execute.
  - **Proximity**: The proximity defines when the function (app-code) that the packet contains should be executed, being if the device is in immediate, near, or far range of the beacon. This can also be set to be executed under all of the aforementioned circumstances.
  - **Extra**: The byte that represent extra can be used to further distinguish the function of the app. For example, if the app-code was set to advertise a message, then this byte could clarify which message. Or it can be set to be the *event-id* if we would like to validate tickets. In theory, being the last part of the whole package, this field can span the remaining length of the packet, if needed, to supply additional information.

7.3.5 Example package

To establish a deeper understand of how a packet is constructed, we will take a look at an example packet.

```
1E 02 01 01 15 FF 23 8D CA 80 BF 0A 11 E3 B1 B6 08 00 20 0C 9A 66 FF FF 00 01 04 FF 01 01 00 00
```

Figure 7.9: Example of a BLE AD-packet.

The packet in Figure 7.9 is explained in further details in table 7.1.

Value	Notes
1E	Length of the entire packet (30)
02	Length of next AD-structure (2)
01	GAP: Flag
01	LE general discoverable flag set
15	Length of next AD-structure (21)
FF	GAP (Manufacturer specific data)
23 8D CA 80 BF 0A 11 E3 B1 B6 08 00 20 0C 9A 66	UUID
FF FF	Major (65535)
00 01	Minor (1)
04	Length of next AD-structure (4)
FF	GAP: Manufacturer specific data
01	App-Code: validate ticket (1)
01	Proximity: immediate (1)
00	Extra (None specified)
00	Padding

Table 7.1: Explanation of BLE AD-packet.

Something to notice is that some GAPs are noted as *Manufacturer specific data*. This is because GAP profiles are generally defined in the core Bluetooth specification. To keep true to these specifications, we decided to make use of the *manufacturer specific data* in absences of more fitting profile definitions.

7.3.6 Transmission

Packets are set by our *Beacon Control Center* (explained in Appendix B), where we can easily change every detail of a beacons AD.

On the receiving device, the AD is converted upon arrival. This is done by first breaking the received byte-array up into a list of AD-structures. As we described in Section 7.3.3, an AD-structure contains:

- the length,
- the type,
- and the data that it carries.

We then follow our packet structure defined in Section 7.3.4 to extract the details that the device is concerned with. For example:

- **Index 0** in the list of AD-structures will be the AD-header.
- **Index 1** will be the *Beacon Identification*, which would include:
  - the UUID,
  - major,
  - and minor.
- **Index 2** would then be the *App Data*:
  - app-code,
  - proximity,
  - and extra

Each field is decoded from a byte-array to their respected type, either integers or hex-strings.

Our POC will disregard any Bluetooth data that does not apply to our packet standards. A list of UUIDs is also included in the app, for it to only accept data from specific beacons.

7.4 Proximity

In Section 5.1.1 we discussed spatial relationships and the definition of *proxemic zones* or *proximity zones*. We put this theory to use in our POC, to divide into three interaction zones:

- immediate (approx. < 1 meter),
- near (approx. 1-5 meters),
- and far (approx. > 5 meter).

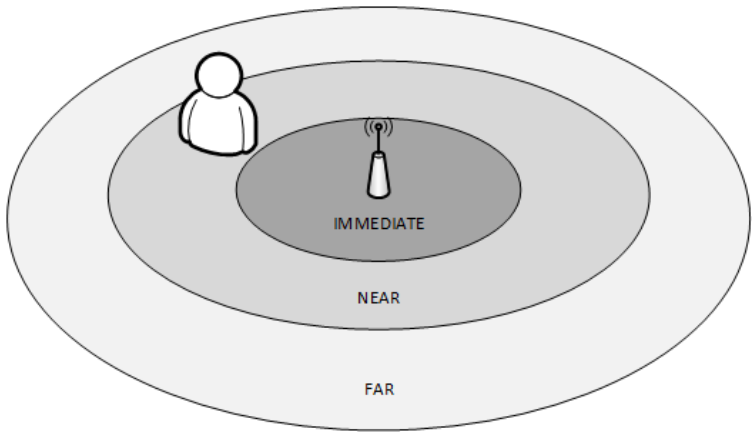

Figure 7.10: Proximity zones.

The zone where an action should be executed is defined in the AD, and can be selected by the beacon administrator through the *Beacon Control Center* (Appendix B). An action can be associated with all or one of the zones.

7.4.1 RSSI - Received Signal Strength Indicator

The application uses the Received Signal Strength Indicator (RSSI) to determine an approximation of proximity zones. RSSI is an attribute of most wireless transmission protocols and is defined in IEEE 802.16 standard for air interface for broadband wireless access systems. RSSI is measured by the voltage of the signal strength from the receiver's RSSI circuit. In other words, it is the power of the received signal as interpreted by the receiving device. In all simplicity: the further away from the transmitter, the weaker the signal, and contrarily:

the closer in range to the transmitter, the stronger the signal. Intuitively, this principle makes RSSI a promising method of localization [124–126].

There are several reasons why RSSI is utilized for indoor positioning in Wireless Sensor Network (WSN), compared to other methods. A largely favorable factor is that we do not require any additional hardware to make use of it. Because of this, properties such as: usability, size, cost, and power consumption also play a part [124–126].

Limitations of using RSSI are rooted in signal accuracy. Therefore, the precision of the distance translated from the RSSI relies on the factors that can affect a signal [124–126]:

- **Shadowing** is the presence of obstacles between the sender and the receiver. This includes everything from walls, buildings and trees, to people, cars and furniture.
- **Path loss** is the gradual loss of intensity in the power of electromagnetic waves as they move.
- **Fading** is a change that a signal undergoes as it is traveling from the transmitter to the receiver. These changes can for example be: reflection, diffraction, and scattering, because of movement or obstetrical in the signals environment.

To establish, or map, proximity zones to the RSSI, we conducted an experiment to evaluate the changes of the RSSI as the distance from the beacon increased. The details of the experiment are described in Appendix D.

In the application, we have implemented a *uniform distance measurement* given by a simplified version of the Friis transmission equation [124, 125]. The transmission equation solved for the distance D is given in Equation 7.1. This equation is used in the POC to calculate an approximation of the distance from the beacon.

$$D = 10^{\frac{P_{r1} - P_r(D)}{K}} \tag{7.1}$$

Here, P_{r1} is an empirically determined parameter denoting the RSSI at a one meter distance from the beacon, and $P_r(D)$ is the RSSI at the current position. The parameter K denotes the signal loss, which is also empirically determined from our experiment. The parameters used in the app are given in Table 7.2. In the app, we also use an averaging filter to smooth the RSSI, hence limiting the effect of signal oscillation. We average over a 5 second time period. Our beacon

is set to transmit an AD-package every 100 milliseconds. This means that we can expect to find a maximum of 50 RSSI readings to average and determine the distance and proximity of in the app.

P_{r1}	K
-67.097	15.058

Table 7.2: RSSI, app parameters.

We chose this general approach over defining RSSI proximities individually for each device in an attempt to shy away from ad-hoc solutions. This approach works satisfactory for 5 out of the 6 devices that we tested in Appendix D. The remaining device had a tendency to show a far more optimistic (higher value) of the RSSI, and thereby also a shorter distance from the beacon than the others. This meant that the proximity zones for this device were activated at a larger radius compared to the others. For the POC to work for the remaining device (or other devices that suffer the same), we defined a method that checks the model number of the devices, and multiplies Equation 7.1 with a constant that "normalizes" the results. Although this would not be a preferred solution in a deployed environment, we would not recommend such an environment to use RSSI as a method for distance measurement, at all. The solution would be more complete if there was any way of changing the transmission power of the beacon so that the reach of a signal would be limited to the proximity zone in question, i.e. for ticket validation in the immediate zone, the beacons reach would be limited to 1 meter or an arm's length. Regretfully, this was not possible to change in the beacons (describe in Appendix A), where the transmission power was pre-defined by the manufacturer of the Bluetooth device to 4 Decibel-milliwatts (dBm).

Other solutions to improve the accuracy of measuring distance can be to incorporate more beacons and use triangulation or *fingerprinting*, which we discussed in Section 5.2.8, to pin-point a more precise location of the user. The cost of the system by introducing a larger amount of beacons will also increase. In such a situation, the developers or customers must evaluate if the trade-off would be worth the extra investment, or if the limitations of an approximation model by using one beacon is sufficient enough for the tasks at hand.

7.5 Ticket validation process

The ticket validation process starts automatically once Bluetooth is enabled, whether or not the user is currently at the event area. However, the user himself/herself decides when he or she wants to activate Bluetooth, i.e. when a notification is fired off alerting of the arrival at an event area, where Bluetooth should be enabled.

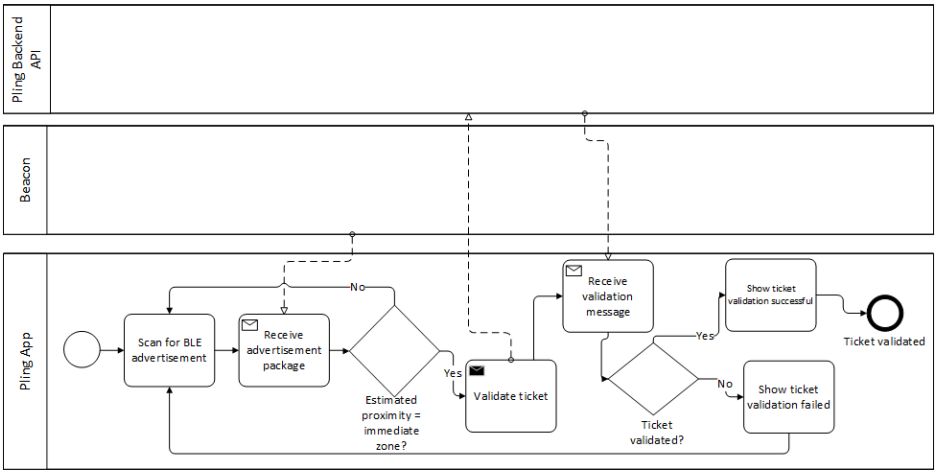

Figure 7.11: Ticket validation process.

7.6 Discussion

Through this section we will attempt to evaluate and discuss our POC, *Pling*, and connect it to the theory of context-awareness found in Chapter 4. This discussion will include an overlook of the limitations and advantages of the system and how Android is suited to host a context-aware pervasive system.

7.6.1 Context

In the coming sections we will discuss how the context-aware features implemented in the *Pling* POC link to the theory of context-aware acquiring, modeling, and adaption.

Acquiring

We have put heavy emphasis on acquiring context implicitly with *Pling*. Mainly, we provide a smart-space of beacons, which gives an approximate proximity. In addition we acquire implicit context with geo-fenced zones to detect when the user enters a venue. The last feature of acquiring context that we have implemented in the app is the possibility of scanning contextual QR-codes. This feature can be challenging to place in either of the two categories of implicit and explicit context. While users do not explicitly give input to the systems by telling it what they might be up to, where they are, who they are with, or what they might be doing, they do have to explicitly tell the system that they are in need of the information contained in the QR-code. The QR-code provides further information about the context in which the user is situated, but the users do not themselves have to tell the system anything about their current context. Still, the user must instruct the system to fetch the information, which is an interesting observation. It would of course be a different situation if the camera was instructed to continuously monitor the environment and retrieve the content of the QR-code without the explicit user interaction. In this case we could for certain think of contextual QR-codes as an implicit context, although for our circumstance we might think of contextual QR as a hybrid.

The aim, and possible future vision of *Pling*, is to make the app experience seamless to the point of minimal user interaction by using a larger specter of context, such as e-mails, social media, time, orientation, movement, etc.

Modeling

The modeling of context in *Pling*, is bound to the object-oriented model, since Android is object-oriented by design. In theory, since Android is flexible, we could have put further effort into incorporating databases or modeling schemes to enhance the use of context. Even though, the object-oriented model seemed to fit our needs well.

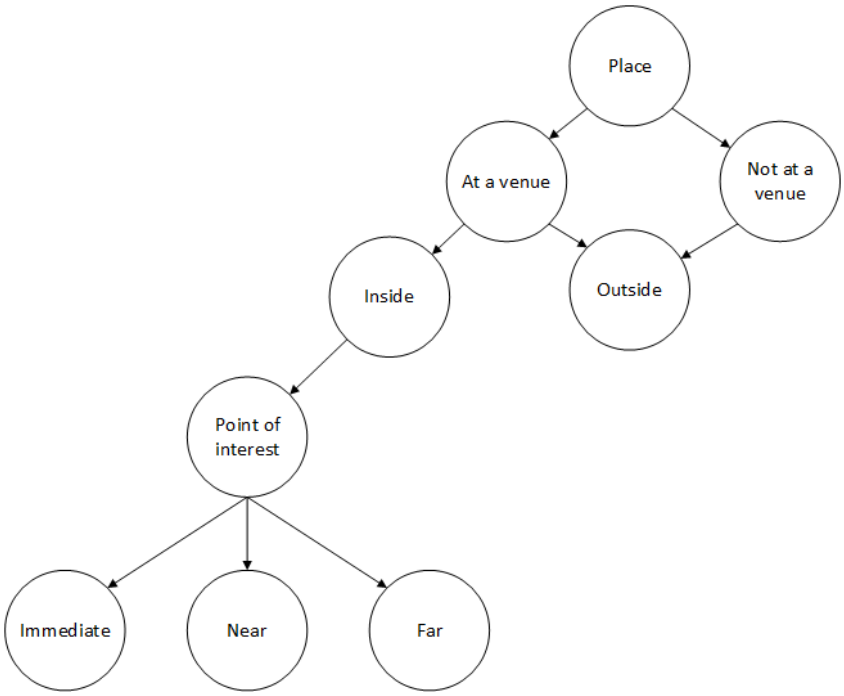

Figure 7.12: Modeling of *Pling*.

In its simplicity, by using GPS and outdoor positioning to geo-fence, as well as BLE beacons for indoor positioning, we base *Pling* on the modeling scheme of Figure 7.12.

Adaptation

We will go through the context-aware adaption techniques, discussed in Section 4.3.2, and map these to our application.

Adaption	Included	Description
• Context aware presentation • Contextual sensing • Automatic contextual reconfiguration	✓	*Pling* displays information to the user by adapting the user interface when approaching a beacon or entering a geo-fenced zone. This can also be triggered in the app by scanning a contextual QR-code.
• Context aware configuration • Contextual resource discovery • Proximate selection	✓	By enabling Bluetooth, and using BLE, we allow the user to discover services (beacons) and interact with them, i.e. validating a ticket. This is done automatically and without a tedious pairing or connecting process.
• Contextual triggered actions • Contextual adaptation	✓	In for example immediate proximity of a beacon, and if the beacon is set to validate a ticket, *Pling* will automatically execute a server request to validate the user's ticket. With the beacon technology, we have the possibility of executing a huge amount of functionality only by knowing the approximate distance to a beacon.
• Contextual adaptation of the environment • Contextual commands	✘	The use of action triggers to execute services in a context when they are made available to the user has, so far, not been a need to implement in *Pling*.

• Contextual augmentation	✓	Contextual QR-codes make contextual augmentation possible in the sense where data is associated with a given context. A QR-code can be placed in the given environment to alert, guide, or inform the user about the environment or context they are in. Future work can provide beacons, or the geo-fence, with the functionality to extract or let the user comment on his or her experience in the context, e.g. at the venue.

Table 7.3: Evaluation of *Pling*.

7.6.2 Android as a host

We will concisely point out the advantages and disadvantages, or limitations, of using Android to host a context-aware pervasive application. The following is based on our experience from writing this thesis.

Limitations

- The availability of sensors varies from device to device.
- Even if the sensor might be present, the API-version of Android installed on the device might not support the use of the sensor.
- User should (according to developer.android.com) be inquired, and approve the use of sensors. The application should preferably not enable sensors at own will, which gets tedious for the user. Drawbacks include:
  - A large amounts of dialogs requesting the enabling of sensors.
  - The system will not work in a satisfactory manner if the user does not accept to turn on the needed sensors.

Advantages

- Android handles a major part of the underlying Input and output (I/O) request and abstracts this away from the developer. The SDK provides a unified platform in Java where all available sensors can be reached and interacted with.
- Android has the ability to check if sensors are enabled, or available, programmatically. Unlike Apples iOS (per iOS 7), Android can also request features to be enabled, like Bluetooth or GPS, directly from within the app.
- Android is not just limited to run on tablets and smartphones.
- Android helps in bridging gaps. By this we mean that, unless it is utterly important to the app, we do not need to worry about if we are using the cellular mobile network or Wi-Fi. If both Wi-Fi and mobile network is enabled, Android will automatically switch to and from these as they become available without the app ever having to be concerned.
- Masking of uneven conditions. Likewise to the example above, Android has a feature to use Wi-Fi, GPS, and cellular network triangulation to determine a higher accuracy of location. This feature will also help positioning systems work in a satisfactory manner in case GPS is unavailable as a source of localization.

8

System Usability Testing

"The limits of the possible can only be defined by going beyond them into the impossible."

– Arthur C. Clarke, *writer, inventor*

This chapter looks at and discusses the responses from our user testing process. The user testing is based on the SUS, which is an easy to use, reliable tool for measuring usability. It consists of 10 questions in a questionnaire, where the answers are on a scale from 1 (strongly disagree) to 5 (strongly agree). We have, ourselves, added a few questions to get more feedback beyond what the SUS survey is able to supply. These questions are not involved in the calculated usability, only to get specific feedback about the POC.

8.1 About the SUS

SUS is the most used survey for measuring usability, or the perceptions of usability. SUS is technology independent, but was originally created as a scale for usability tests for systems like the VT100 Terminal applications. It has since become an industry standard, and referenced in over 600 publications [127].

8.2 Instructions

Each participant was given a document explaining the required actions on their part, as seen in Appendix E.

8.3 Questions

These are the questions that are asked in the questionnaire. Each question is answered on a scale from 1 (strongly disagree) to 5 (strongly agree). In the list of questions, questions 1 through 10 are a part of the original SUS survey, while the remaining questions are linked directly to our POC.

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.
11. I think it's a drawback that I need Bluetooth enabled to validate my ticket.
12. I think it's an advantage that I don't need to show my ticket, or navigate to the ticket to validate it.
13. I would rather take advantage of this technology than use ticketing applications based on validating QR-codes (or other paper-based systems).

In addition to the questions above, we added a question where the testers can provide a text-based feedback: *Please explain your thoughts and experience using Pling.*

8.4 Test participants

All test participants have a technical background, and comprises of a mix between students at NTNU and employees at WTW AS. According to [128], research has shown that the SUS converges to the "correct" conclusion, or correct decision, over 90% of the time when there are 12 test participants. Lewis and Sauro argues that because of this, every SUS study should have sample sizes of at least 12 participants. For these reasons, we have chosen to include 12 test participants in our usability study.

8.5 Interpreting Scores

Calculating the SUS scores can be complex, and the process, or how the scores are interpreted, is listed below.

1. To calculate the SUS score, we first need to sum the score contributions from each item, where each item's score contribution will range from 0 to 4.
   (a) For items 1, 3, 5, 7, and 9, the score contribution is the scale position minus 1.
   (b) For items 2, 4, 6, 8, and 10, the score contribution is 5 minus the scale position.
2. Multiply the sum of the scores by 2.5 to get the overall value of SUS.

8.6 Answers and SUS scores

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	Score
Tester 1	4	1	4	2	3	3	5	1	2	3	2	5	4	70.0
Tester 2	4	3	4	2	3	2	4	3	3	2	4	4	3	65.0
Tester 3	4	1	5	1	4	2	4	2	4	1	3	5	3	85.0
Tester 4	5	1	5	1	5	1	4	1	4	2	4	5	5	92.5
Tester 5	3	1	5	1	4	1	5	1	3	1	1	5	5	87.5
Tester 6	3	2	4	3	3	3	4	2	3	2	4	4	3	62.5
Tester 7	4	1	5	1	3	3	5	2	4	1	3	4	4	82.5
Tester 8	4	1	4	1	4	1	5	4	4	1	5	5	5	82.5
Tester 9	4	1	5	1	5	1	4	1	4	1	2	5	4	92.5
Tester 10	5	1	5	1	3	3	5	1	3	1	1	5	5	85.0
Tester 11	4	2	4	1	5	2	5	1	2	2	3	4	3	80.0
Tester 12	5	2	4	1	4	1	5	2	4	1	2	5	5	87.5

Table 8.1: Form feedback from user testing.

The calculated scores in Table 8.1 are based on the feedback for questions 1 through 10 and calculated according to the rules listed in the previous section. Questions 11, 12, and 13 were added as a supplement to outline the user's view of the POC, and are not part of the SUS.

Example calculation for *Tester 1*:

1. Odd numbers score: $(4-1)+(4-1)+(3-1)+(5-1)+(2-1)=13$
2. Even numbers score: $(5-1)+(5-2)+(5-3)+(5-1)+(5-3)=15$
3. Score: $(13+15)2.5=70$

The SUS scores have a range from 0 to 100, though these are not percentages and should only be considered in terms of their percentile ranking.

According to [127], a score of 68 is above average, and a score of 80.3 is among the top 10% scores. Our average score was calculated to be 81.0.

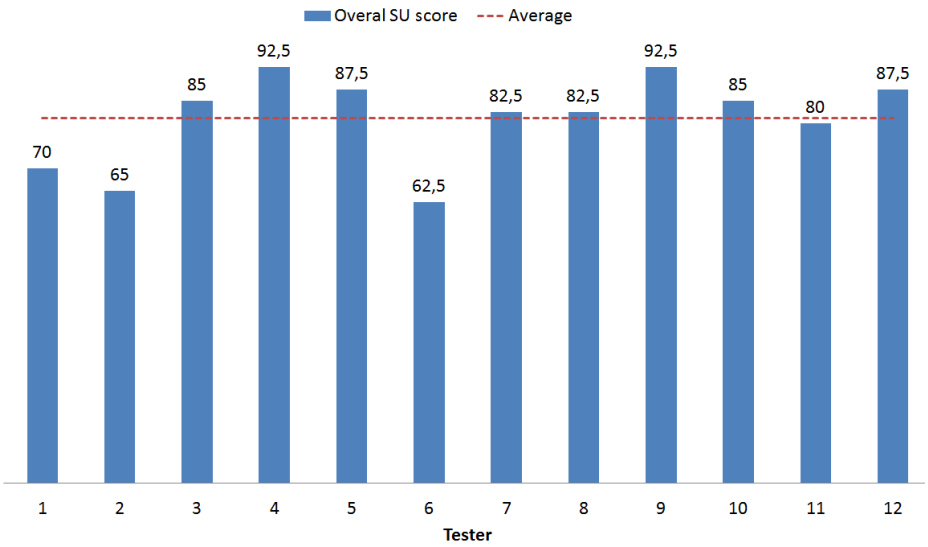

Figure 8.1: SUS score graph.

8.6.1 Comments by testers

The last question on our questionnaire was a text-based feedback field, where the testers could write what they thought about the POC. It was not mandatory, and as such we only received 6 answers on this particular question.

- "Nice application, but a map where ticket control is located would be handy."
- "It seemed cool, but what happens if there is multiple 'open' events that use the same entrance? Open as in you could come and go when you wanted (one time only), and not as in ticketless."
- "I liked the design of the application."
- "Splendid business case! Looking forward to actual use it on the next concert or game I will attend ;)."
- "Fast, easy, and no hassle."
- "Nice design, easy to use."

8.7 Discussion

We are happy with the results, which show that the application has high usability. The test participants seemed very interested in how everything worked, and most of them asked a lot of questions, after the test was completed.

Figure 8.2 shows a bar chart with answers to question 11, 12, and 13, by all test participants.

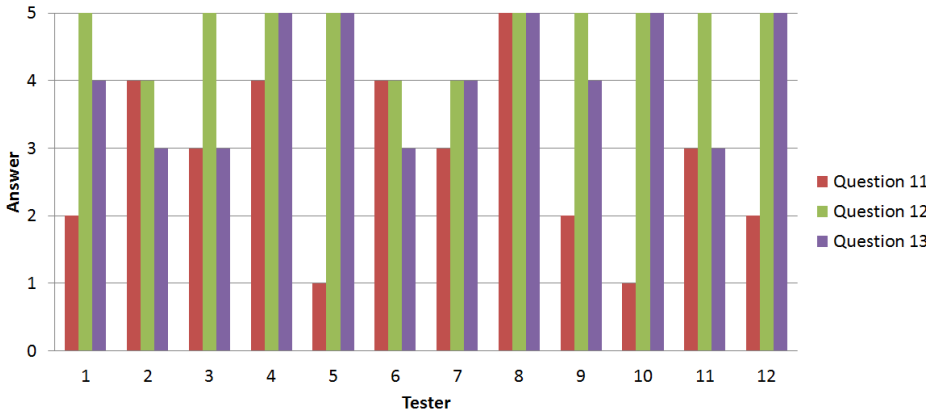

Figure 8.2: Answers to the SUS supplement questions.

From question 11, *I think it's a drawback that I need Bluetooth enabled to validate my ticket*, we can see that there are mixed feelings regarding this particular question. Some strongly disagree, while others strongly agree. The majority of the answers are either neutral, or agreeing that it's a drawback. Taking into consideration the answers to question 13, most of those who agree that having Bluetooth enabled is a drawback would still rather use this type of technology to validate tickets, as opposed to a purely QR-based ticketing system.

Bluetooth is also the key to achieve question 12, which reads: *I think it's an advantage that I don't need to show my ticket, or navigate to the ticket to validate it.* In fact, 9 out of 12 strongly agree, while the rest agree, to this question.

Based on the answers given to questions 11, 12, and 13, we believe that Bluetooth is a viable way of validating tickets, with enough incentives to make people enable Bluetooth prior to arriving at the event, or the validation area.

9

Evaluation

"I never think of the future - it comes soon enough."

– Albert Einstein, *theoretical physicist*

This thesis has explored context-awareness, its approaches and practical uses for mobile ticketing systems. As an ending note we would hereby like to summarize and conclude the thesis with this chapter. We will revisit and review the research questions, contributions, and in the end, sum up challenges and the possibility of future work.

9.1 Research questions

Research question 1 *Review mobile ticketing in Norway; are there any advantages and disadvantages?*

In Chapter 3, we describe 12 of the major mobile ticketing systems currently being used in Norway, where 11 of them belong in the transportation category. We were surprised to see that we could only find *one* mobile ticketing system for events, called *eBillett*, on Google Play. The app has received poor ratings, and is from our own experience poorly constructed, as well as quite buggy.

Most of the mobile ticketing systems offer several ways of purchasing tickets:

- 5 out of 12 support ticket-purchasing via SMS.
- 11 out of 12 support ticket-purchasing by credit-card.

- 4 out of 12 support ticket-purchasing using a mobile account.
- 5 out of 12 support online ticket-purchasing through a web-site, where the ticket can later be downloaded to the app.

Having a mobile ticketing system, as opposed to a paper-based one, is in itself a huge advantage. Tickets are more durable, as they can't get dirty or damaged in any way, and they're always with you where you go as most people carry their mobile phone with them at all times. By using smartphones for ticketing, it would be nearly impossible to leave your ticket at home when you're going to an event. Of the 12 mobile ticketing systems, 9 of them are for public transportation such as buses, ferries, and trains. The implementation of a mobile ticketing system has, for these companies, resulted in a decrease in the use of banknotes and coins for purchasing tickets. This might lead to fewer robberies, something which has happened several times in the past, caused by the amount of money on-board.

Some events choose to use the *Cashless* system to essentially become 'cashless', meaning there is no money-handling during, or after the event. There are disadvantages to this approach for both events, and users. Event organizers would need to pay for terminals to support this kind of system, where users also have to pay for a cashless card, and transfer money to it before going to the event. Any money left over once the event ends, has to be manually transferred back from the cashless card to the user's own bank-account, in a tedious fashion. We believe the right approach to become 'cashless' is to create, and use, a mobile ticketing system for events, where the system should support various methods for purchasing:

- Mobile account
- Credit-card
- SMS

Four of the mobile ticketing systems we investigated were delivered by the same company, WTW. Each of these systems share more or less the same functionality, with certain differences, but they all support mobile account payment. A mobile account can be looked at as a bank account, which is charged when you pay for a ticket. We find it a disadvantage that a user does not have the same login details in each of these four apps, as well as a different mobile account for each app. This is of course only a problem if the user needs, on occasion, to use public transport in more than one of the cities covered by these four apps.

Research question 2 *How has context-aware computer systems developed through history up until now?*

Research question 2 is explored in Chapter 4, and looks at how context-awareness sprung out from the ubiquitous- and pervasive- computing era. This era came to be as a junction of mobile- and distributed- computing, where the foundation is to distribute the capabilities of the computer to pursue a new frontier of mobility. The focus of context-awareness is the capability for a computer to sense its presence in a *context* and thereby improve the UX by taking advantage of the context.

In Chapter 4 we also discuss, at length, what a context can be and how we can acquire, model, and adapt to it. We use this as a theoretical foundation to show how practical approaches can be achieved in a modern mobile operating system and demonstrate a selection of these with a POC.

Research question 3 *What approaches, technology and practices can be used for context-awareness in mobile devices?*

We learn about approaches, both theoretical and practical, through Chapter 5. The focus is strongly tied to modern mobile operating systems and how they are eligible to perceive their environment. We also discuss technology that can be used to connect these devices to external sources of context. We emphasize the following through the chapter:

- Proxemics
- Contextual QR-codes
- Geo-fencing and spatial positioning
- Augmentation
- Service-oriented context
- Mobile devices
- Device sensors
- GPS
- RFID
- NFC
- Bluetooth
- BLE
- Wi-Fi

Research question 4 *How can context-awareness improve the mobile ticketing experience?*

We use what we have learned from the information we acquired when answering the 3 other research questions to build a POC to answer research question 4.

With the POC, we show how just a small amount of context-aware approaches can help the user. We take advantage of BLE to demonstrate ticket validation in the mobile ticketing system *Pling*. By using BLE, we can supply the user with additional information to aid the user when he or she approaches a validation beacon. To enhance the UX further, we incorporate geo-fencing and show how a geo-fenced area can adapt the application with event information once a user

approaches an event. This information can be used to explain that the user should preferably enable Bluetooth to get the most out of the application. We also took the time to implement a third context-aware option which incorporates the pervasive barcode technology called QR-code. These QR-codes are referred to as contextual QR-codes and include additional contextual information about the context in which a user is situated in. When a user scans the QR-code, the application is supplied with a context and can adapt accordingly.

By commencing and achieving an above average score of 80.5 in a user-test based on the SUS, we learn that the context-aware features of our POC are easy for users to adjust to and learn.

9.2 Contributions

Throughout this thesis, our work has focused on bridging the gap between theory and practice. The technology, approaches, and principals of: context-awareness, ubiquitous computing, pervasive computing, Android, Bluetooth, etc. that we discuss, are previously well known on a corporate- and research- level. We couple existing work and look at direct ways of applying it to a real world scenario: a ticketing system, built on the Android operating system.

The contributions that this thesis brings forth are ingrained in the attempt to provide a foundation to establish the advantages that can be upheld by incorporating context-awareness in mobile application to improve usability. To do this we provide a thorough survey of context-awareness by combining and discussing a great volume of the research that has impacted the field. The amount of papers that have been written about the subject is of a very great quantity, and so a guide to introduce and assist future developers as a reference to context-awareness is of good use.

Through our studies, we have made an attempt to classify context on the basis of the combined efforts or research articles, papers and books that we used to survey the field. We establish a hierarchical structure of classification that build on previous attempts of a classification. The structure tries to join together and represent the research field as a whole.

We also proceed to give an in-depth look at approaches and technologies to reach, and use context-awareness in a modern mobile operating system. Many of the approaches and technologies are future oriented, and at the cutting edge as many tech-companies are currently at an early stage of introducing them.

To tie the thesis together, we contribute with a POC as a demonstration of how context-awareness can aid the UX of an application. We also confirm the usability of the POC by accomplishing an above-average score in a usability test based on the SUS.

9.3 Related work

A tremendous amount of sources have been used to survey and explore context-awareness in this thesis, all of which can be counted as related work. Countless sources also look at mobile phones or Android as a context-aware platform, though most take an architectural or design perspective as a primary subject. [129–133] are just some of the many examples of using or developing context-awareness for mobile phones for Android.

Technology wise, we loosely base our work on industry attempts at geo-fencing and beacons, where we can reference both the Apple's iBeacon [134] and Qualcomm's Gimbal [135], as examples. We proceed to have a unique perspective on a very popular mobile application use-case in Norway, mobile ticketing. By coupling the technological approaches with the theory of context-awareness we show that we can use beacons, not only for distributing messages, but also for practical tasks to ease the UX of our case study.

9.4 Challenges

This section contains the challenges we had to overcome throughout this thesis.

9.4.1 Setting up beacons and advertisement of data

Setting up the beacons, and advertisement of data, turned out to be a bit of a challenge. We studied the Bluetooth Core Specification, but with the lack of other reliable sources, certain blog posts, and answers on *stackoverflow.com*, came in very handy. Once we figured out how packages are structured, we could easily use this knowledge to create our own format to achieve what we wanted.

The following list contains the questions we needed answers to with regards to beacons and advertisement packages:

1. How the format is structured for advertisement packages.

2. How to create our own format for advertisement packages.
3. How to increase the frequency of advertisement packages.
4. How to increase or decrease the range of advertisement packages.

Of the four items above, only item number 4 was not possible to do anything about. We know how the range can be increased or decreased, but one needs to be able to alter the transmission power of the beacon, which in our case could not be changed (it seems to be manufacturer-specific in most cases).

9.4.2 RSSI

Another problem area that consumed a generous portion of the time used for the development of the POC, was how we could translate the RSSI to approximation of distance from a beacon. Detailed in Appendix D, the *signal strength experiment* gives an outlook on how we confronted the problem to arrive at a solution that worked satisfactory for our POC application. We see this as an area which is in need of further work before being deployed, and we discuss some solutions in the upcoming Section 9.5.

9.5 Future Work

As the reader might understand from our discussion of context-awareness in Chapter 4, we have only just brushed the tip of the iceberg. We have managed to demonstrate an introduction to the simplicity and usability that context and context-awareness can supply to an app, but the possibilities of smartphones taking advantage of context are multitudinous. Continuing, we will look at some of these possibilities, while still keeping a general look on our POC.

9.5.1 Indoor positioning

Since the emergence of GPS, outdoor positioning has been a trivial task when speaking of location-awareness. One of the primary challenges is to find a good source of indoor positioning, given that it also is user friendly.

By using triangulation, it is possible to create an indoor positioning system, which can help users find and locate their destination. This will be handy in cases where a large event is taking place indoors, where the user needs to find a certain booth, stand, or other POI, quickly. Should a crisis, such as a fire or

other type of disaster occur, the location for gathering could be displayed on a map, and help everyone navigate easily to it.

A lot of research has been done on indoor positioning using numerous types of technologies, but for a mobile application the most reasonable technologies as of today would be Bluetooth and Wi-Fi. It is possible to create a hybrid solution utilizing signals from both Bluetooth devices and Wi-Fi routers/access points, as described in [136]. In a future effort to better the use of RSSI as a source of indoor location-awareness, error estimation of system states such as, for example, the Kalman filter could be of use [137].

9.5.2 Target groups

A context-aware ticket application is not only applicable to an event scenario. As we have previously discussed in Chapter 3, the largest use of ticketing applications for smartphones in Norway exist in the public transportation sector. This sector is dependent on a reliable, fast, and easy way of validating and selling tickets to keep up with timetables. A context-aware application in public transportation could possibly automate the process even further so bus companies, airlines, boats, trains and ferries can become even more punctual.

Although we have discussed context-awareness primarily for ticket-based systems as a case study, we should point out that context-awareness can benefit other parts of society as well. As an example, other interesting future cases could be how to apply context-awareness to assist the elderly or PwAD. We could well be approaching a future where these groups could achieve a greater independence, and thereby quality of life, by accomplishing tasks on their own only by using a device that can sense and alert of circumstances in the environment. In terms, this could also benefit care takers and the economic perspective, especially since the elderly generation is becoming ever more populous year by year.

9.5.3 Beacon technology

At big events, or exhibitions, each stand could have their own Beacon. When approaching the beacon, information about what the stand is displaying, or talking about, could appear in the app. This will help information be easily accessible, and people will not have to stand in a queue to get to know what the people at the stand are talking about, which could possibly prevent large queues from forming.

At small events, each kiosk or POI could have its own beacon. Each beacon could provide the nearby users with information about the kiosk, or POI, and help users perform informed decisions.

9.5.4 Application

The ticketing system would not be complete without a validator app, used by those controlling the validation process. The validator app should not only have support for validating tickets, but validation of purchases as well.

The app can have the following capabilities:

- Validate tickets.
- Validate purchases.
- Overview of number of guests/attendees.
- QR-scanner to support manual ticket validation.

This would also create more security for event organizers, where the possibility for people to sneak in to an event would be limited.

For diversity in the system, impending work should focus on porting the system to other platforms such as: Apples iOS, Microsoft Windows Phone, and Blackberry phones. Future implementation should further validate the approaches of this thesis on the operating system or device in question, and preferably build on these.

9.5.5 Guidance and automation

Geo-Fencing has been implemented in the POC, but with limited functionality. It would be interesting to see how Geo-Fencing can improve the UX in other ways as well, i.e. guide people walking in the wrong direction, or simply suggest routes the user could take. Should the user walk past an event, the app could notify the user even though he or she does not have a ticket, and ask whether or not the user would like to purchase one.

Automatic ticket purchasing can be implemented in mobile ticketing systems for public transportation. As we have previously described, the advertisement package structure we use for our beacons have two fields called *major* and *minor*. These can be used to identify a bus-route (or a bus). Imagine a ticketing app for buses, where the app can sense, by a beacon that you are boarding the bus

without a ticket and execute the purchase, and the validation, in a two-step procedure.

9.5.6 Context sources

Our main focus in this thesis has been the use of BLE beacons to create a smart space for supplying context. Forthcoming development could focus on other infrastructure or sensors to enhance the UX. Chapter 4 explains that context might be hard to quantify or define, but the potentials are limitless, so to speak. Mobile devices feature new and upgraded hardware almost every year, while communication technology and integration advances are developing at an alarming pace. This means that the sources we can gather context from will in all likelihood not decrease in the years ahead. We should also point out that the tremendous amounts of data generated on a daily basis can supply context by advances in picture, video, and audio recognition.

Appendices

Appendix A

Beacon

This appendix explains the BLE-beacons that were used throughout the thesis.

A.1 Raspberry Pi

WTW purchased 2 Raspberry Pi's that we have been using throughout this thesis.

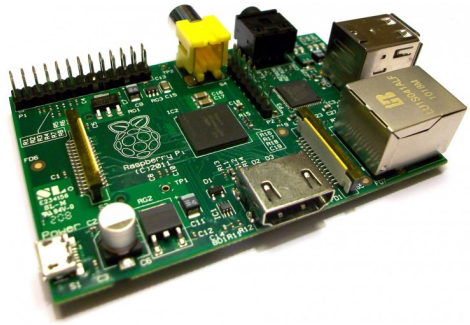

Figure A.1: Raspberry Pi Model B Rev 2

Manufacturer	Raspberry Pi Foundation
Device	Raspberry Pi
Model	Model B rev 2
Format	Single-board computer
Released	29 February, 2012
Operating System	Raspbian
Memory	512 MB
CPU	ARM1176JZF-S (ARMv6k) 700 MHz
Storage	SD card slot
Power	3.5 W

Table A.1: Raspberry Pi specifications.

In addition to the Raspberry Pi, we are using the K-Mate Bluetooth dongle model BTT012B.

A.1.1 Installation

1. Begin by installing the Bluetooth stack and various USB development packages. Execute the following command:

sudo apt-get install libusb-dev libdbus-1-dev libglib2.0-dev libudev-dev libical-dev libreadline-dev

2. Install BlueZ' source files. The version of BlueZ used throughout this thesis is 5.11.

```
sudo wget www.kernel.org/pub/linux/bluetooth/bluez-5.11.tar.xz
sudo unxz bluez-5.11.tar.xz
sudo tar xvf bluez-5.11.tar
cd bluez-5.11
sudo ./configure --disable-systemd
sudo make
sudo make install
```

Once steps 1 and 2 are complete, you can shut down the Raspberry Pi, connect the USB BLE-enabled Bluetooth dongle, and boot it up again.

3. The command-line tools hcitool and hciconfig are now available for use.

A.1.2 Verifying the installation

Hcitool is a command-line tool for configuring Bluetooth connections, and to send special commands to Bluetooth devices. To simply start advertising data in the iBeacon format, execute the following command:
sudo hcitool -i hci0 cmd 0x08 0x0008 1e 02 01 1a 1a ff 4c 00 02 15 e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 00 00 00 00 c5 00 00 00 00 00 00 00 00 00 00 00 00 00

In this particular example, the UUID is set to E2C56DB5-DFFB-48D2-B060-D0F5A71096E0, and has a major and minor of 0.

To enable and activate advertising, execute the following command: *sudo hciconfig hci0 leadv 3*

To actually verify that it works, you could install the iBeacon Locate app by Radius Networks, which is available for both Apple and Android devices.

A.2 Arduino with BLE Shield

Figure A.2: Arduino UNO r3 board.

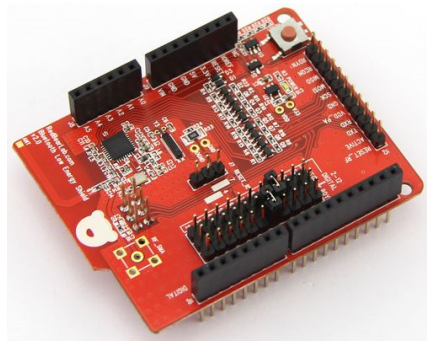

Figure A.3: RedBearLab BLE shield for Arduino.

Manufacturer	Smart Projects
Format	Single-board microcontroller
Microcontroller	ATmega328
Released	September 24, 2010
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock speed	16MHz

Table A.2: Arduino UNO specifications.

The RedBearLab BLE shield seen in Figure A.3 is designed to work with Arduino boards such as Arduino UNO, Mega 2560, Leonardo and Due. It allows the Arduino to communicate with other BLE devices, such as smartphones or tablets.

Appendix B

Beacon Control Center

The beacon control center is a web-frontend for configuring the Raspberry Pi beacon, and what kind of advertisement package that is to be sent to the devices running the mobile ticketing application.

B.1 Frontend

The frontend is a single-loaded webpage, with JavaScript for communicating with the backend API, and runs on the Raspberry Pi's. It allows the user to start/stop the beacon, and select which type of action that is intended for the mobile application to execute when it receives the advertising packet.

While using the interface, the user can hit the 'D' button on the keyboard to show debugging information, consisting mainly of output from the system commands executed by the backend API, as seen in Figure B.1.

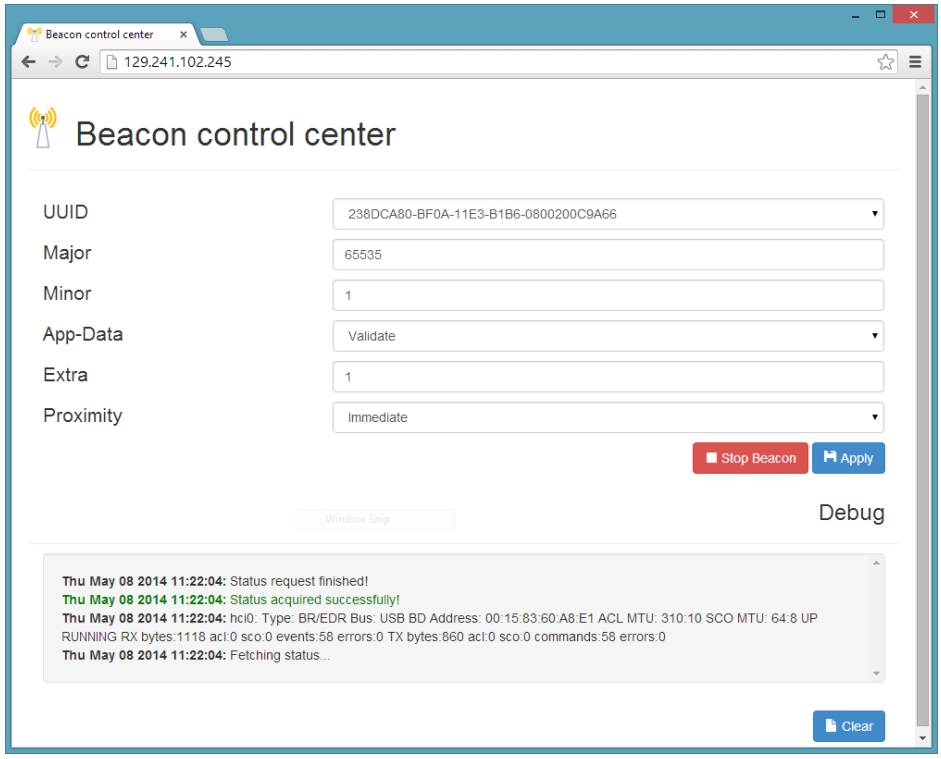

Figure B.1: Beacon Control Center interface.

There are 6 settings the user can configure:

- **UUID**: Identifier for the service (3 values are predefined).
- **major**: Integer [0,65535] for grouping the device.
- **minor**: Integer [0,65535] for identifying a device in a group.
- **appCode**: Integer [0,255] for specifying an action to be executed by the mobile ticketing application.
- **proximity**: Integer [0,3], a proxemic zone in which the action is to be taken.
- **extra**: 2 bytes that can be used for the actions specified by appCode.

B.2 Backend API

The backend is built using PHP and the REST-library *Tonic*. The following subsections document which resources are available, which methods are supported, and the required input and output.

B.2.1 Power REST resource

The power resource allows the client to see current status of the Bluetooth device/beacon, as well as either powering it up or down.

URI	/api/rest/power
Method	GET
Description	Returns the current status of the beacon, including all configurable attributes.
Input	This method expects no input.

Example output

```
{
   power: true,
   info: "hci0: Type: BR/EDR Bus: USB BD Address:
       00:15:83:60:A8:E1 ACL MTU: 310:10 SCO MTU: 64:8 UP RUNNING
```

```
        RX bytes:1118 acl:0 sco:0 events:58 errors:0 TX bytes:860
        acl:0 sco:0 commands:58 errors:0 ",
    config: {
       UUID: "238DCA80BF0A11E3B1B60800200C9A66",
       major: "65535",
       minor: "1",
       appCode: "1",
       proximity: "1",
       extra: "0"
    }
}
```

URI	/api/rest/power
Method	PUT
Description	Accepts input and either turns the beacon on or off.
Input	power - true OR false **Supplied input needs to be JSON-encoded**

Example Output

```
{
   info:"Disabling virtual iBeacon...\nhci0:\tType: BR\/EDR Bus:
       USB\n\tBD Address: 00:15:83:60:A8:E1 ACL MTU: 310:10 SCO
       MTU: 64:8\n\tDOWN \n\tRX bytes:1130 acl:0 sco:0 events:60
       errors:0\n\tTX bytes:899 acl:0 sco:0 commands:60
       errors:0\n\nComplete\n",
   power:false
}
```

B.2.2 Advertisement REST resource

The advertisement resource allows the client to define the data that is to be advertised by the beacon.

URI	/api/rest/advertisement
Method	PUT
Description	Accepts input, which is then advertised by the beacon
Input	UUID - 16bytes hexadecimal string (no spaces) major - integer(0-65535) minor - integer(0-65535) appCode - integer(0-255) proximity - integer(0-255) extra - 1-3 character string or integer (0-65535) **Supplied input needs to be JSON-encoded**

Example output
Based on the input, the returned data is simply the command that was executed by the backend.

```
1 {
2    command:"sudo \/usr\/local\/bin\/hcitool -i hci0 cmd 0x08
         0x0008 1e 02 01 01 15 FF e2 c5 6d b5 df fb 48 d2 b0 60 d0
         f5 a7 10 96 e0 ff ff 00 01 4 ff 01 01 00 00
3 }
```

B.2.3 Scripts

The following are scripts used by the API resources, to either start or stop the Bluetooth device.

start_ble.sh
start_ble.sh enables the Bluetooth device, and sets it to be in advertising mode.

```
#!/bin/sh
echo "Launching virtual iBeacon..."
sudo /usr/local/bin/hciconfig hci0 up
# Setting advertisement frequency to 100ms
sudo hcitool -i hci0 cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00 00
    00 00 00 00 07 00
#sudo /usr/local/bin/hciconfig hci0 leadv 3

# Set device to be in advertisement mode
sudo hcitool -i hci0 cmd 0x08 0x000a 01
# Disable scanning - ensure optimality
sudo /usr/local/bin/hciconfig hci0 noscan
sudo /usr/local/bin/hciconfig hci0
```

stop_ble.sh
stop_ble.sh disables the Bluetooth device.

```
#!/bin/sh
echo "Disabling virtual iBeacon..."
# Disable advertisemeent
sudo /usr/local/bin/hciconfig hci0 noleadv
# Disable Bluetooth device
sudo /usr/local/bin/hciconfig hci0 down
# Check Bluetooth device status
sudo /usr/local/bin/hciconfig hci0
echo "Complete"
```

B.3 Acquiring the IP of a Raspberry Pi

Without a monitor, keyboard, and mouse, it might be hard to figure out which IP the Raspberry Pi is assigned, unless you have access to the router it's connected to. With access to the router, you can simply check the DHCP client list, but without it you'll need a mechanism that lets you know the IP.

Throughout this thesis, we have been using our self-developed IP-acquiring system, which consists of a server-side script and web-frontend, as well as a standalone script running every few minutes on the Raspberry Pi. The standalone script on the Raspberry Pi sends its IP address to the server-side script every five minutes, and can be viewed at the web-frontend, as seen in Figure B.2.

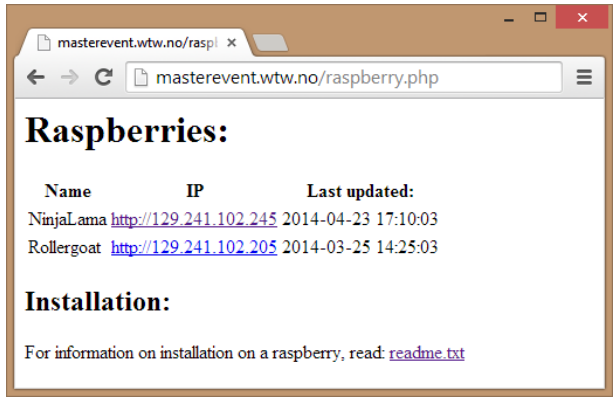

Figure B.2: Overview of the Raspberry Pis with IP and time of update.

B.3.1 Server-side script

```
1  <?php
2  require_once(dirname(__FILE__) . '/../include/init.inc.php');
3  use Raspberry\Entities\Raspberry;
4
5  $raspberryName = isset($_GET['raspberryName']) ? $_GET['raspberryName'] :
       '';
6  $raspberryIp = isset($_GET['raspberryIp']) ? $_GET['raspberryIp'] : '';
7  $pass = isset($_GET['rPw']) ? $_GET['rPw'] : '';
8  if ($pass != 'somepassword')
9      die("D e nei");
10 $result = array();
```

```
11 if (!empty($raspberryName) && !empty($raspberryIp)) {
12     $raspberry = Raspberry::load($raspberryName);
13     if ($raspberry) {
14         $raspberry->raspberryIp = $raspberryIp;
15         header('Content-type: application/json;');
16         if ($raspberry->update()) {
17             $result['raspberry'] = array('raspberry_name' =>
                   $raspberryName,
18                 'raspberry_ip' => $raspberryIp,
19                 'changetime' => $raspberry->changetime);
20             echo json_encode($result);
21             die();
22         } else {
23             header($_SERVER['SERVER_PROTOCOL'] . ' 500 Internal Server
                   Error', true, 500);
24         }
25     } else {
26         $raspberry = new Raspberry();
27         $raspberry->raspberryName = $raspberryName;
28         $raspberry->raspberryIp = $raspberryIp;
29         header('Content-type: application/json;');
30         if ($raspberry->insert()) {
31             $result['raspberry'] = array('raspberry_name' =>
                   $raspberryName,
32                 'raspberry_ip' => $raspberryIp,
33                 'changetime' => $raspberry->changetime);
34             echo json_encode($result);
35             die();
36         } else {
37             header($_SERVER['SERVER_PROTOCOL'] . ' 500 Internal Server
                   Error', true, 500);
38         }
39     }
40 } else {
41     $raspberries = Raspberry::getAll();
42     echo "<h1>Raspberries:</h1><p>";
43     echo "<table><tr><th>Name</th><th>IP</th><th>Last updated:</th></tr>";
44     foreach ($raspberries as $raspberry) {
45         echo "<tr><td>" . $raspberry->raspberryName . "</td><td><a
               href=\"http://" . $raspberry->raspberryIp . "\">http://" .
               $raspberry->raspberryIp . "</a></td><td>" .
               $raspberry->changetime . "</td></tr>";
46     }
47     if (empty($raspberries)) {
48         echo "<tr><td colspan=3>No raspberries found</td></tr>";
```

```
49      }
50      echo "</table>";
51      echo "<p><h2>Installation:</h2>";
52      echo "For information on installation on a raspberry, read: <a
            href=\"readme.txt\">readme.txt</a>";
53      echo "</p>";
54  }
55  ?>
```

B.3.2 Stand-alone script (Raspberry Pi)

```
1  #!/usr/bin/python
2  import urllib2
3  import socket
4  import requests
5
6  s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
7  # The following is just a hack (but leave it in!)
8  s.connect(("masterevent.wtw.no",80))
9  ipAddress = s.getsockname()[0]
10 s.close()
11
12 rPw = "somepassword"
13 raspberryName = "Your Raspberry Name"
14
15 payload = {'rPw': rPw, 'raspberryName':raspberryName,
       'raspberryIp': ipAddress}
16 r = requests.get('http://masterevent.wtw.no/raspberry.php',
       params=payload)
17 print r.text
```

Our Raspberry Pi's have been configured to execute the script every 5 minutes, using crontab, as such:

```
1  */5 * * * * /home/pi/mevent/ip_update.py > /dev/null
```

B.4 Installation and configuration

This section explains how the Beacon Control Center can be installed.

Step 1: Install apache2 with PHP-support.

Step 2: Create the folders /var/www/vhosts and /var/www/vhosts/beacon.

Step 4: Configure the default apache vhost to use the previously created folders.

Since our Raspberry Pi is only going to have one web-site hosted, which is the Beacon Control Center, we simply chose to edit the default vhost configuration file.

sudo vim /etc/apache2/sites-available/default
Replace the content with the following:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www/vhosts/beacon/www
        <Directory />
                Options FollowSymLinks
                AllowOverride All
        </Directory>
        <Directory /var/www/vhosts/beacon/www>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride All
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log
        # Possible values include: debug, info, notice, warn,
            error, crit,
```

```
        # alert, emerg.
        CustomLog /srv/www/log/http-raspi-access.log combined
        ErrorLog /srv/www/log/http-raspi-error.log
        LogLevel warn
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Step 5: Find the source-code for the Beacon Control Center (the root-folder should be called *beacon*).

Step 6: Copy everything from within the *beacon* root folder to /var/www/vhosts/beacon/.

Step 7: Verify that /var/www/vhosts/beacon has the following files and folders:

```
pi@ninjapi /var/www/vhosts $ ls beacon/
bin  classes config.ini includes lib raspi.conf www
```

Step 8: Create the group *rasp* and make the user *pi* a member of it.

Step 9: Assign ownership of the /var/www/vhosts folder to apache (the www-data user).

Step 10: Allow apache to execute the beacon backend scripts which configures the beacon, as well as the command line tools *hcitool* and *hciconfig*.

sudo vim /etc/sudoers
At the end of the *sudoers* file, append the following:

```
www-data ALL = (root) NOPASSWD:
    /var/www/vhosts/beacon/bin/start_ble.sh
www-data ALL = (root) NOPASSWD:
    /var/www/vhosts/beacon/bin/stop_ble.sh
www-data ALL = (root) NOPASSWD:
    /var/www/vhosts/beacon/bin/advertise.sh
www-data ALL = (root) NOPASSWD: /usr/local/bin/hcitool
www-data ALL = (root) NOPASSWD: /usr/local/bin/hciconfig
```

Step 11: Enable the mod_rewrite module of Apache by executing:
sudo a2enmod rewrite.

Step 12: Restart apache and verify that everything works by accessing the Beacon Control Center using the browser of your own choice.

Appendix C

Pling: Backend API

This chapter describes the backend API used by the *Pling* mobile ticketing application. Only the REST resources and the database is documented. The source code is not of importance, as we see it.

C.1 REST Resources

This section describes the REST resources available, and what they do.

C.1.1 Overview

Method	URI	Requires login	Description
GET	/rest/ticket	Yes	Returns a list of tickets
GET	/rest/ticket/TID	Yes	Returns a ticket with ticketId=TID
POST	/rest/validate/TID	Yes	Validates a ticket with ticketId=TID
GET	/rest/venue	No	Returns a list of venues
GET	/rest/venue/VID	No	Returns a venue with venueId=VID
GET	/rest/event	No	Returns a list of venues
GET	/rest/event/EID	No	Returns an event with eventId=EID
GET	/rest/authenticate	Yes	200 OK if successful, 500 if not

Table C.1: Overview of REST resources.

For those resources in Table C.1 that requires login, parameters *username* and *password* need to be passed in the URL. As an example, for validation of a ticket: *../rest/validate/3?username=demo1&password=demo1*. In this example, the number 3 would be the unique ID for the ticket that the user is about to validate.

C.2 Relational Database

The database which is implicitly used by the *Pling* POC app is quite simple, as seen in Figure C.1.

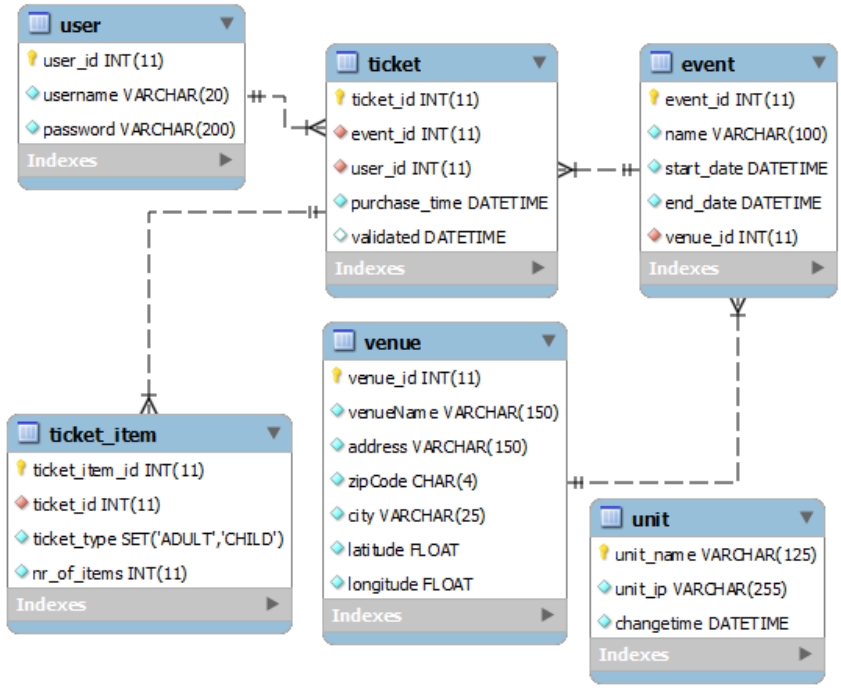

Figure C.1: ER diagram of the database used by *Pling*.

The ER diagram should be self-explanatory, the only things might worth pointing out are:

- The *unit* table contains the Raspberry Pis we used in this thesis.

- A *ticket* consists of *ticket_items*, where each *ticket_item*-record defines a type of ticket (ADULT|CHILD), and the number of that type. I.e., if you want to purchase a ticket for 2 adults and 1 child, there would be two *ticket_item*-records, one saying 2 adults, and the last one saying 1 child.

Appendix D

Signal strength experiment

This section will, in details, cover how we conducted our empirical study to determine interaction zones for our POC from chapters 6 and 7.

The objective of the experiment was to uncover if there was any foundation to link the RSSI of a transmission to distances in an effortless manner. A requisite for success in the experiment was to be able to map the RSSI to approximations of interaction, or proximity zones. The different zones should be activated at roughly the same distance from the beacon for each device.

D.1 Pre-acquired knowledge

We shall briefly discuss the knowledge we have about measuring distance with RSSI that we knew of beforehand.

Under perfect conditions, with a direct path between the transmitter and the receiver, neglecting: reflections, scattering, shadowing, etc., the Received Signal (RS) power is inversely proportional to the distance [125, 126].

$$P_r \propto \frac{1}{d^2} \tag{D.1}$$

Although in the real world, the signal might deteriorate at a faster pace. Generally, free space propagation (when the transmitter and receiver are in clear sight) is given by the Friis transmission equation [124, 125]:

$$P_r(D) = P_t + G_t + G_r + 20 \times log_{10}(\frac{\lambda}{4\pi D}) \tag{D.2}$$

In Equation D.2, P_t and P_r denotes the power received in dBm for the transmitter and the receiver, respectfully. G_t and G_r are the antenna gain in Decibels-isotropic (dBi), while λ is the wavelength, and D is the distance. Still, antenna gain is hard to determine, and most studies use the simplified equation (Equation D.3) [124, 125]:

$$P_r(D) = P_{r1} - K \times log_{10}(D) \tag{D.3}$$

Here, P_{r1} is the received power at 1 meter and K is the loss parameter. Both are empirically determined.

Most proposed models for radio signal propagation do not take into account the variation in the sensitivity of the receiver, the orientation of the device, noise, or interference. Distance estimation are also more prone to errors indoors than outdoors, where: different buildings, furniture, and wall arrangement can affect the signals.

The outlook for our results before commencing the tests are bleak, as previous studies have rendered RSSI unsuitable for positioning regardless of the estimation technique [126]. Also, once deployed, previous platforms that take advantage of RSSI have functioned worse than predicted [124].

D.1.1 Limitations of our model

- Networks using BLE advertisement, such as ours, cannot communicate in duplex, only simplex. Duplex communication is only possible when devices are paired.
- As Android is fragmented in both software and hardware, we expect that Bluetooth modules from device to device will somewhat differ. We can therefore anticipate experiencing gaps and variations between devices in our test reading.
- We were not able to change the transmission power of our beacon as this was pre-set by the manufacturer of the Bluetooth module to a default value of 4 dBm.

D.2 Devices

The experiment was executed with six different devices running Android, which are described in the following subsection. Some of the devices are personal, lent from fellow classmates, while others were at our disposal courtesy of WTW. All information about the devices are retrieved from gsmarena.com, or directly from the devices (where this was possible).

BLE became a part of the Bluetooth core specification from version 4.0 and onwards, and Android from version 4.3 (API level 18) and onwards. With this in mind, a minimum requirement for the devices used in the test was Bluetooth v4.0 and Android 4.3. Some of the devices have been rooted and some Android versions have been upgraded to meet the requirements. The rooted devices are not considered to have an effect on the RSSI, as this is implemented in the Bluetooth hardware. We cannot for certain rule out the possibility of slight differences in drivers or software implementation for devices that do not run stock ROM.

To create as close to equal conditions for each device, the following settings were enabled when testing:

Setting	**Enabled**
GPS	✘
Bluetooth	✓
Wi-Fi	✓
NFC	✘
Aeroplane mode	✓
Brightness	100 %

Table D.1: Device test settings.

Aeroplane mode was enabled on all devices since some of the devices did not have a SIM-card installed, and therefore not connected to the cellular network.

D.2.1 HTC One

Manufacturer	HTC
Device	One
Model	m7
Format	Phone
Released	March, 2013
Android version	4.4.2
Bluetooth	v4.0
Rooted	✓
Stock ROM	✘

Figure D.1 & Table D.2: HTC One (m7).

D.2.2 HTC One X+

Manufacturer	HTC
Device	One X+
Model	enrc2b
Format	Phone
Released	November, 2012
Android version	4.3.1
Bluetooth	v4.0
Rooted	✓
Stock ROM	✘

Figure D.2 & Table D.3: HTC One X+ (enrc2b).

D.2.3 HTC One S

Manufacturer	HTC
Device	One S
Model	ville
Format	Phone
Released	October, 2012
Android version	4.4.2
Bluetooth	v4.0
Rooted	✓
Stock ROM	✘

Figure D.3 & Table D.4: HTC One S (ville).

D.2.4 Samsung Galaxy S3

Manufacturer	Samsung
Device	Galaxy S3
Model	GT-I9300
Format	Phone
Released	May, 2012
Android version	4.3.0
Bluetooth	v4.0
Rooted	✘
Stock ROM	✓

Figure D.4 & Table D.5: Samsung Galaxy S3 (GT-I9300).

D.2.5 Samsung Galaxy S4

Manufacturer	Samsung
Device	Galaxy S4
Model	GT-I9500
Format	Phone
Released	April, 2013
Android version	4.4.2
Bluetooth	v4.0
Rooted	✓
Stock ROM	✘

Figure D.5 & Table D.6: Samsung Galaxy S4 (GT-I9500).

D.2.6 Asus Google Nexus 7

Manufacturer	ASUS
Device	Google Nexus
Model	7
Format	Tablet
Released	July, 2013
Android version	4.4.2
Bluetooth	v4.0
Rooted	✘
Stock ROM	✓

Figure D.6 & Table D.7: ASUS Google Nexus 7.

D.3 Test application

The test application was constructed to be as simple as possible. As Figure D.7 shows, the main activity contains only one button: to start scanning. When selecting to scan, the app prompts the user to input the current test distance (in meters) as seen in Figure D.8. The process of scanning is depicted in Figure D.9. Just above the *Stop Scanning* button, the current number of advertising packets received is counted.

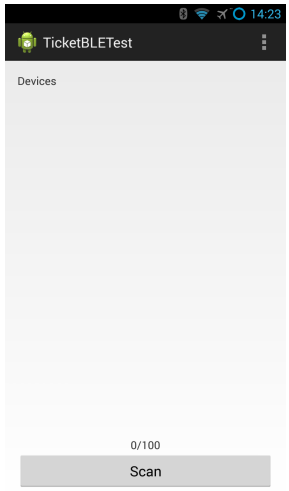

Figure D.7: Test app main activity.

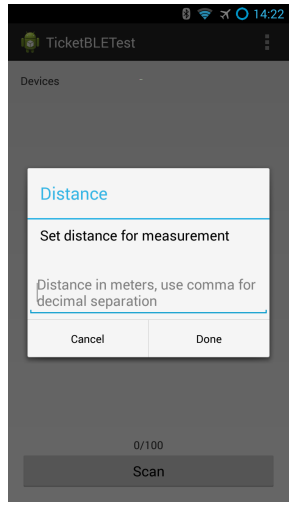

Figure D.8: Input for distance.

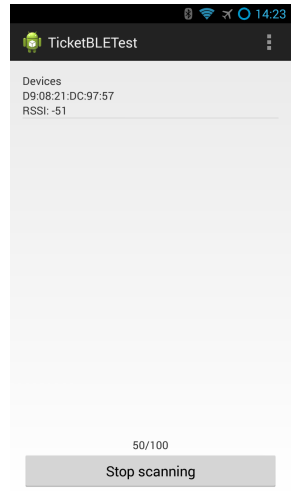

Figure D.9: Scanning for advertising data.

The beacon's Media Access Control (MAC) address is hard coded in the app to prevent other BLE devices in the vicinity to tamper or overwrite the incoming signals.

When the device finishes scanning 100 packets, it alerts the user by vibrating and playing a notification sound. The details of each packet include: a UNIX-timestamp of when the packet was received, the RSSI, and the distance in question.

D.4 Environment

Tests were conducted indoors in a conference room of approximately 66.75 m^2. The room consisted of 4 windows, 1 door, and ordinary furniture such as: table, chairs for approx. 10 people, projector, sofa, etc.

Figure D.10: Floor plan.

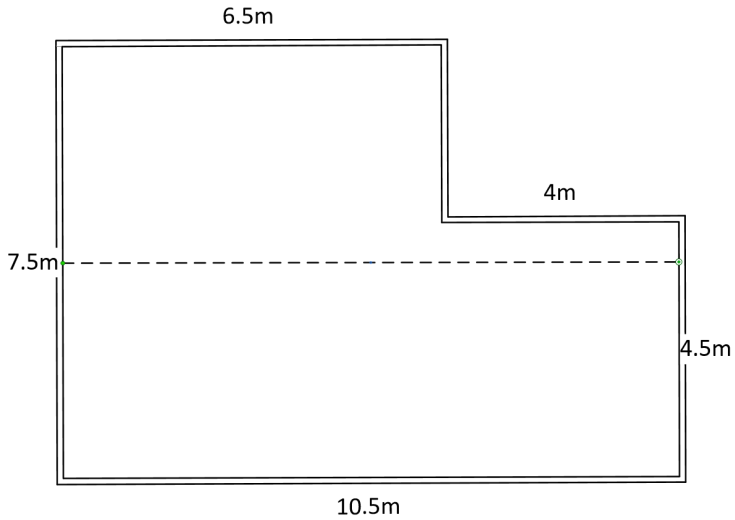

Figure D.11: Room dimensions (measurements are approximations).

D.5 Execution

Tests were conducted as follows:

- Distances were measured up using a folding ruler, starting at the far right wall (measuring approx. 4.5 meters) of Figure D.11, and ending at the left wall (measuring approx. 7.5 meters). This went parallel to the wall measuring approx. 10.5 meters, and was roughly centered on the right wall. Intervals were marked with post-it notes and measured:
  - 0.0 meters (beacon location)
  - 0.2 meters
  - 0.4 meters
  - 0.6 meters
  - 0.8 meters
  - 1.0 meters
  - 2.0 meters
  - ...
  - 10 meters
- The beacon was placed at ground level **behind** the 0.0 meter mark.
- The long side of the device was placed with clear sight of the beacon at ground level in **front of** the interval marks. Each device was tested one at a time and not simultaneously.
- Devices started the test at the 0.0 meter mark. When a device finished at one interval mark, it was moved to the next interval mark, moving further and further away from the beacon, and ending at 10 meters.
- Devices were tested by our test application, gathering 100 data packets from the transmitter (beacon) at each mark.

The test was conducted twice: once with our Raspberry Pi beacon (discussed in Section A.1), and once with an Arduino BLE-shield (discussed in Section A.2). This was done to establish if there were any grounds to conclude that the transmitter could have any major effect on the results.

D.6 Results

This section will go through the results of the acquired RSSI data. We mainly focus on the average of the 100 readings for each distance. This is done since radio signals generally suffer from heavy oscillation, and any application to handle radio signals would be expected to deal with noise by some sort of smoothing filter, such as averaging.

D.6.1 Raspberry Pi

Distance	One	S	X+	S3	S4	Nexus 7
0.0	-32.67	-38.96	-33.95	-20.59	-34.67	-31.55
0.2	-49.84	-59.44	-44.77	-47.37	-56.14	-56.15
0.4	-55.61	-66.31	-58.17	-52.29	-59.60	-60.51
0.6	-62.75	-70.26	-59.49	-56.43	-62.54	-66.09
0.8	-65.51	-74.05	-68.33	-60.50	-69.28	-74.25
1.0	-60.09	-73.01	-67.11	-63.02	-67.73	-71.62
2.0	-69.27	-76.79	-72.92	-57.42	-70.41	-69.41
3.0	-69.08	-76.98	-72.00	-65.82	-71.19	-74.64
4.0	-73.60	-78.89	-71.61	-66.61	-72.74	-78.77
5.0	-79.10	-83.57	-78.87	-65.54	-82.24	-80.79
6.0	-80.41	-89.87	-85.45	-69.95	-83.54	-82.96
7.0	-80.76	-88.5	-87.72	-77.90	-89.11	-86.39
8.0	-82.17	-89.64	-82.68	-76.26	-87.31	-81.56
9.0	-77.06	-89.82	-84.25	-74.25	-84.37	-82.13
10.0	-87.47	-93.74	-84.75	-72.79	-87.43	-81.19

Table D.8: Raspberry Pi - average RSSI.

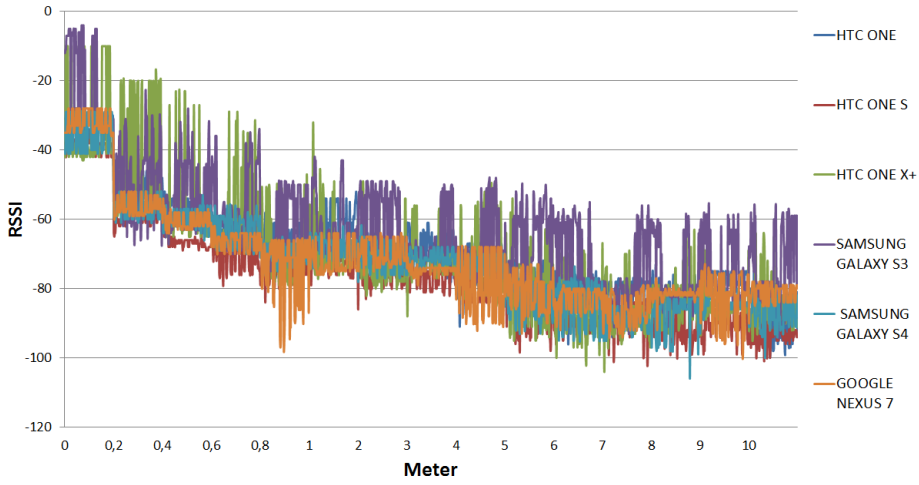

Figure D.12: Raspberry Pi - RSSI.

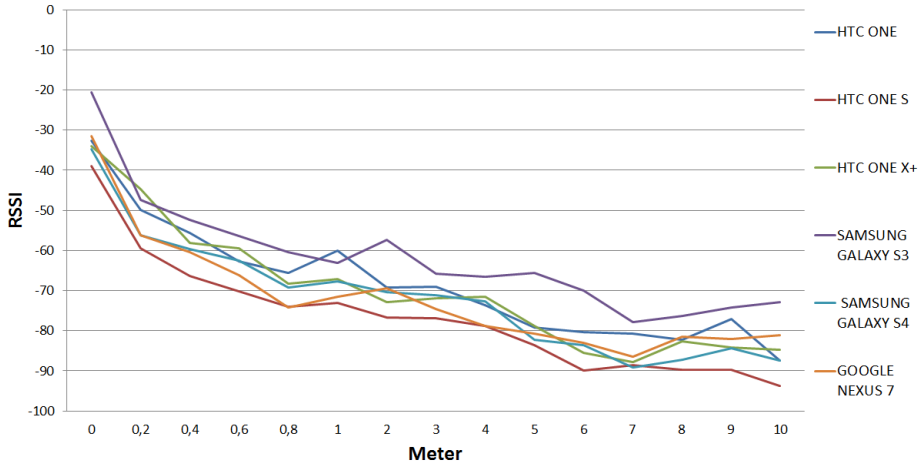

Figure D.13: Raspberry Pi - average RSSI.

D.6.2 Arduino

Distance	One	S	X+	S3	S4	Nexus 7
0.0	-28.04	-42.41	-35.90	-37.09	-53.35	-42.54
0.2	-54.05	-73.41	-68.11	-64.97	-59.13	-64.88
0.4	-65.21	-74.35	-73.34	-64.84	-66.86	-68.15
0.6	-69.33	-81.52	-68.66	-68.68	-67.06	-64.53
0.8	-65.52	-76.18	-75.58	-69.10	-72.99	-69.67
1.0	-68.94	-76.13	-76.29	-67.46	-70.85	-66.61
2.0	-64.95	-72.25	-79.07	-63.23	-70.69	-68.01
3.0	-68.33	-78.43	-83.79	-68.63	-72.90	-71.37
4.0	-72.33	-78.35	-79.69	-72.38	-76.19	-72.98
5.0	-78.73	-87.20	-84.18	-76.05	-80.24	-77.35
6.0	-79.22	-82.47	-88.91	-75.97	-78.00	-80.72
7.0	-79.50	-84.34	-89.80	-79.07	-86.12	-79.79
8.0	-80.74	-87.38	-90.52	-83.61	-87.32	-83.88
9.0	-82.27	-86.34	-88.03	-79.90	-87.02	-78.20
10.0	-79.61	-90.05	-87.07	-79.18	-86.82	-79.16

Table D.9: Arduino - average RSSI.

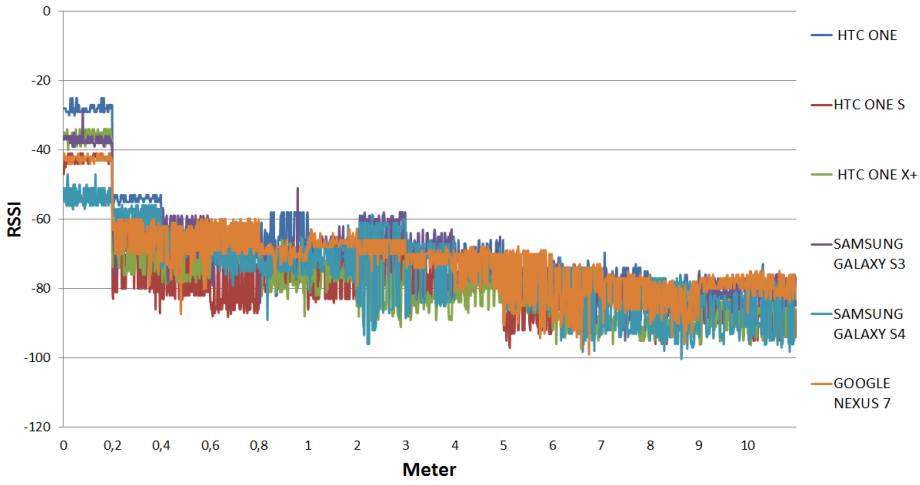

Figure D.14: Arduino - RSSI.

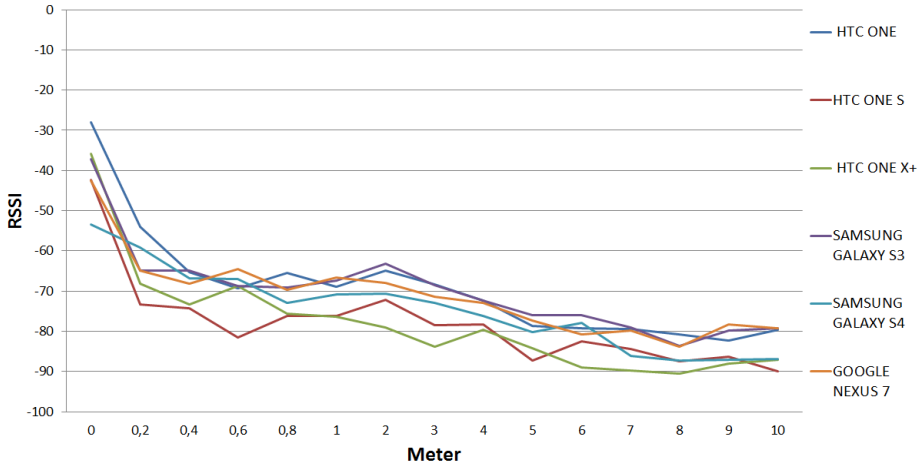

Figure D.15: Arduino - average RSSI.

D.6.3 Differences

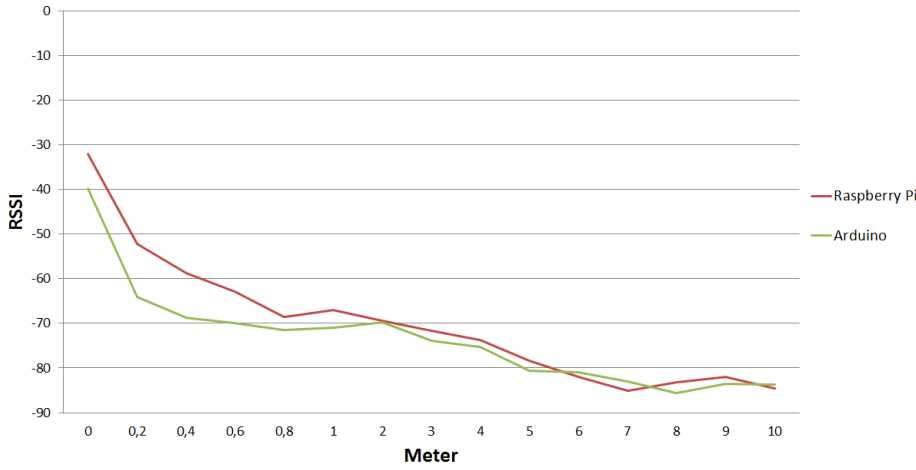

Figure D.16: Differences between Arduino and Raspberry Pi.

Figure D.16 show the differences between the combined average of all devices for the Arduino and the Raspberry Pi in RSSI at each respective distance.

D.7 Test conclusion

Our aim for this test was to create an, as close to, uniform equation or approach for RSSI distance measurement which could be used for all devices regardless.

In an attempt to evaluate distances, we must first determine the loss parameter, K, and the RSSI at one meter, P_{r1}. From Equation D.3 we can determine that the loss parameter K can be solved as follows:

$$K = \frac{P_{r1} - P_r(D)}{log_{10}(D)} \tag{D.4}$$

Equation D.4 is true when $D > 1$. By using the average RSSI for distances 2-10 meters and Equation D.4, we get the following averages of K:

D.7.1 Raspberry Pi - Loss parameter

One	S	X+	S3	S4	Nexus 7	Combined Avg.
24.351	15.834	17.183	6.616	16.630	9.736	15.058

Table D.10: Raspberry Pi, the propagation loss parameter K, rounded to 3-decimals.

D.7.2 Arduino - Loss parameter

One	S	X+	S3	S4	Nexus 7	Combined Avg.
7.613	7.377	12.548	8.464	11.609	13.125	10.123

Table D.11: Arduino, the propagation loss parameter K, rounded to 3-decimals.

The average RSSI at one meter, P_{r1}, can be read from Table D.8 for the Raspberry Pi and Table D.9 for the Arduino:

D.7.3 Raspberry Pi - 1 meter constant

One	S	X+	S3	S4	Nexus 7	Combined Avg.
-60.090	-73.010	-67.110	-63.020	-67.730	-71.620	-67.097

Table D.12: Raspberry Pi, average RSSI at one meter.

D.7.4 Arduino - 1 meter constant

One	S	X+	S3	S4	Nexus 7	Combined Avg.
-68.940	-76.130	-76.290	-67.460	-70.850	-66.610	-71.047

Table D.13: Arduino, average RSSI at one meter.

The distance based on RSSI can be determined by solving D.3 for D:

$$D = 10^{\frac{P_{r1} - P_r(D)}{K}} \tag{D.5}$$

If we were to define three proximity zones:

- immediate (< 1 meter),
- near (1-5 meter),
- and far (> 5 meter),

we would need to mark two borders: one at 1 meter and another at 5 meters. The following four tables (two for the Raspberry Pi and two for the Arduino) use Equation D.5 to evaluate the distance based on the RSSI. The first table in each section (D.14 and D.16) define the distance by using the average loss parameter, K, and the average 1 meter RSSI reading, P_{r1}, for each device individually. The second table in each section (D.15 and D.17) defines a uniform loss parameter, K, by a combined average and a uniform 1 meter RSSI reading, P_{r1}, also by a combined average. The loss parameter for each device and the combined average can be found in Table D.10 for the Raspberry Pi. The 1 meter RSSI reading for each device and the combined average can be found in Table D.12 for the

Raspberry Pi. As well, for the Arduino, the loss parameter for each device and the combined average can be found in Table D.11. The 1 meter RSSI reading for each device and the combined average can be found in Table D.13 for the Arduino.

For each table we have also taken the liberty to mark the borders where the proposed proximity zones would transition into another. One meter is marked in green, and 5 meters is marked in red.

D.7.5 Raspberry Pi - Distance measurement

Actual Distance	One	S	X+	S3	S4	Nexus 7
0.0	0.075	0.007	0.012	0.000	0.010	0.000
0.2	0.379	0.139	0.050	0.004	0.201	0.026
0.4	0.655	0.377	0.302	0.024	0.324	0.072
0.6	1.286	0.670	0.360	0.101	0.487	0.270
0.8	1.669	1.163	1.178	0.416	1.239	1.863
1.0	1.000	1.000	1.000	1.000	1.000	1.000
2.0	2.382	1.733	2.178	0.142	1.449	0.593
3.0	2.340	1.781	1.926	2.650	1.615	2.043
4.0	3.588	2.352	1.828	3.488	2.001	5.425
5.0	6.035	4.644	4.835	2.404	7.456	8.747
6.0	6.831	11.609	11.677	11.154	8.927	14.613
7.0	7.060	9.512	15.828	177.421	19.303	32.889
8.0	8.067	11.227	8.056	100.261	15.045	10.494
9.0	4.976	11.524	9.942	49.812	10.014	12.009
10.0	13.316	20.379	10.631	29.969	15.297	9.615

Table D.14: Raspberry Pi, distance measurement.

Actual Distance	One	S	X+	S3	S4	Nexus 7
0.0	0.005	0.014	0.006	0.001	0.007	0.004
0.2	0.071	0.310	0.033	0.049	0.187	0.188
0.4	0.173	0.887	0.255	0.104	0.318	0.365
0.6	0.514	1.622	0.313	0.196	0.498	0.857
0.8	0.785	2.896	1.208	0.365	1.396	2.986
1.0	0.343	2.470	1.002	0.536	1.102	1.997
2.0	1.394	4.403	2.436	0.228	1.660	1.424
3.0	1.354	4.532	2.117	0.823	1.870	3.169
4.0	2.703	6.070	1.994	0.928	2.370	5.959
5.0	6.268	12.415	6.051	0.788	10.131	8.116
6.0	7.658	32.533	16.550	1.547	12.359	11.310
7.0	8.079	26.385	23.418	5.217	28.964	19.109
8.0	10.023	31.409	10.836	4.060	21.995	9.130
9.0	4.588	32.286	13.776	2.986	14.031	9.962
10.0	22.540	58.793	14.870	2.388	22.402	8.628

Table D.15: Raspberry Pi, uniform distance measurement, $P_{r1} = -67.097$ and $K = 15.058$.

D.7.6 Arduino - Distance measurement

Actual Distance	One	S	X+	S3	S4	Nexus 7
0.0	0.000	0.000	0.001	0.000	0.031	0.015
0.2	0.011	0.428	0.223	0.508	0.098	0.738
0.4	0.324	0.574	0.582	0.490	0.453	1.310
0.6	1.125	5.378	0.247	1.394	0.472	0.694
0.8	0.355	1.016	0.878	1.562	1.529	1.711
1.0	1.000	1.000	1.000	1.000	1.000	1.000
2.0	0.299	0.298	1.666	0.316	0.969	1.278
3.0	0.832	2.050	3.960	1.375	1.502	2.305
4.0	2.788	2.000	1.866	3.813	2.884	3.057
5.0	19.319	31.663	4.254	10.349	6.440	6.581
6.0	22.405	7.234	10.133	10.126	4.130	11.887
7.0	24.385	12.968	11.930	23.534	20.672	10.098
8.0	35.482	33.493	13.615	80.926	26.227	20.695
9.0	56.361	24.209	8.622	29.496	24.712	7.640
10.0	25.210	77.068	7.229	24.249	23.751	9.041

Table D.16: Arduino, distance measurement.

Actual Distance	One	S	X+	S3	S4	Nexus 7
0.0	0.000	0.001	0.000	0.000	0.018	0.002
0.2	0.021	1.712	0.513	0.251	0.066	0.246
0.4	0.265	2.120	1.685	0.244	0.386	0.517
0.6	0.677	10.831	0.581	0.584	0.404	0.227
0.8	0.284	3.215	2.804	0.642	1.556	0.731
1.0	0.619	3.178	3.296	0.442	0.956	0.365
2.0	0.250	1.315	6.203	0.169	0.922	0.501
3.0	0.539	5.363	18.151	0.577	1.524	1.076
4.0	1.339	5.266	7.143	1.354	3.222	1.552
5.0	5.742	39.425	19.835	3.121	8.095	4.195
6.0	6.419	13.443	58.170	3.065	4.863	9.029
7.0	6.841	20.570	71.223	6.203	30.837	7.307
8.0	9.070	41.073	83.897	17.423	40.516	18.526
9.0	12.845	32.420	47.617	7.492	37.843	5.089
10.0	7.014	75.391	38.276	6.360	36.160	6.332

Table D.17: Arduino, uniform distance measurement, $P_{r1} = -71.047$ and $K = 10.123$.

D.7.7 Discussion

We chose to implement the *uniform distance measurement* that is evaluated in Table D.15 for distance approximation. In the following, we discuss this in more details.

Advantages

- The data makes room for the possibility of determining a somewhat loose approximation of the distance of a subject. This, in terms, means that there is a hope of making the POC work with several devices in mind.

Limitations

- Other than the empirically determined loss parameter, we do not account for variation in the sensitivity of the receiver, the orientation of the device, noise, or interference.
- Tests were done under "perfect" conditions, while in a real world scenario we can expect to find more interference from artifacts such as moving human beings.
- Averaging in a real world scenario would generally consist of less RSSI data to determine the distance in trade for responsiveness.

We wish to shy away from ad-hoc implementations which work on a per-device basis, and a silver lining is that this might be possible if only for the concept of validating a ticket in our POC. Though the question still remains: is RSSI sufficient enough for distance measuring? - In our experience, the answer is no. A detailed implementation would require a higher degree of knowledge for each device in question, and in the world of Android where devices and device manufactures are by the dozens, this is simply not possible.

Appendix E

Pling User Testing Instructions

E.1 Instructions at NTNU Campus

You are going to an event at IT-Bygget Gløshaugen, and you have already purchased a ticket for said event using the *Pling* Mobile Ticketing Application. The ticket is available in the *Pling* app once you login with your user-details, which are available further down in this document.

Your goal is to validate your ticket with this app.

E.1.1 User details

You should log in using the following username and password:

username: _____

password: _____

E.1.2 Event entrance

The event entrance is located between IT-Vest and Gamle Fysikk at Gløshaugen, NTNU, Trondheim. The event itself will take place at Gribb (ITV-060).

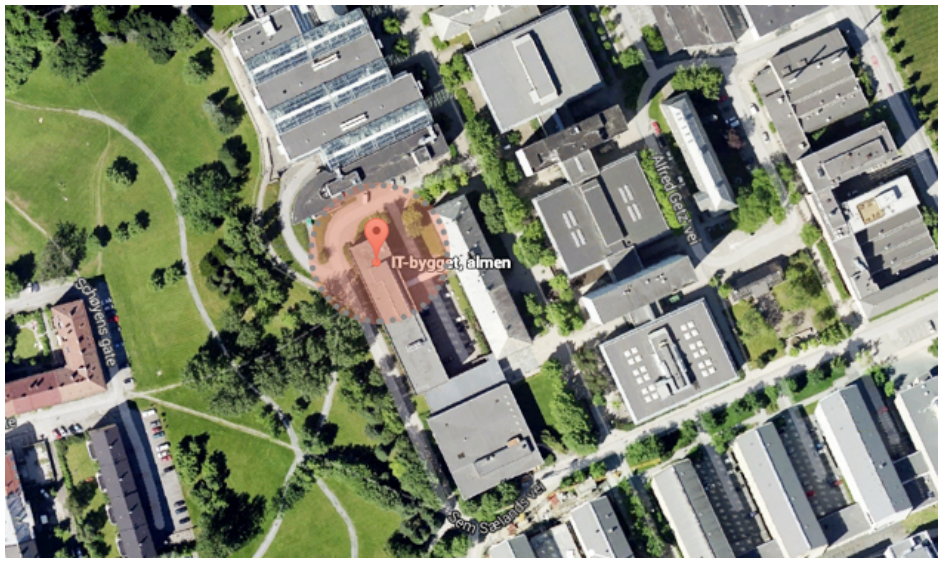

Figure E.1: Map with start-location and end-location for user testing at Gløshaugen, NTNU, Trondheim.

E.1.3 Steps to complete user testing

1. Go to the entrance of IT-vest (between IT-bygget and Gamle Fysikk).
2. Start the *Pling* Mobile Ticketing Application.
3. Start walking towards the event area.
4. Read everything in the application carefully (Don't dismiss a dialog or feedback from the app without reading the information presented to you).
5. Go to Gribb (ITV-060) and validate your ticket.

Once testing is done, go to *http://tinyurl.com/kw645gf* and complete the form.

E.2 Instructions at WTW AS

You are going to an event at WTW, Vestre Rosten 78, and you have already purchased a ticket for said event using the *Pling* Mobile Ticketing Application. The ticket is available in the *Pling* app once you login with your user-details, which are available further down in this document.

Your goal is to validate your ticket with this app.

E.2.1 User details

You should log in using the following username and password:

username: _____

password: _____

E.2.2 Event entrance

The event entrance is located at Vestre Rosten 78. The event itself will take place in the meeting room.

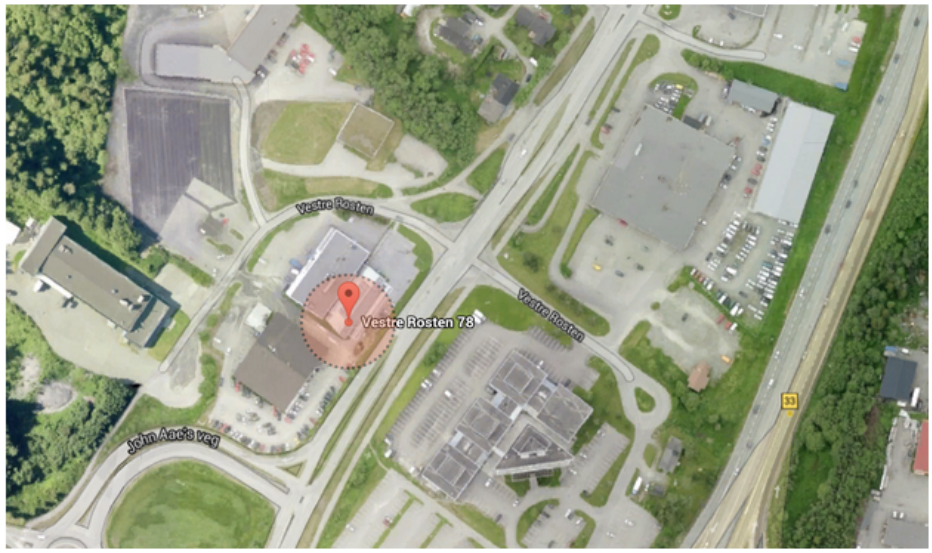

Figure E.2: Map with start-location and end-location for user testing at Vestre Rosten 78, Tiller.

E.2.3 Steps to complete user testing

1. Go to the WTW Catering Kitchen.
2. Start the *Pling* Mobile Ticketing Application.
3. Log in with your user details (found on the previous page).
4. Start walking towards the event area (The meeting room).
5. Read everything in the application carefully.

Once testing is done, go to *http://tinyurl.com/kw645gf* and complete the form.

Bibliography

[1] Zoe Fox. The 15 countries with the highest smartphone penetration, 2013. URL http://mashable.com/2013/08/27/global-smartphone-penetration/. Accessed 2014-03-20.

[2] Zoe Fox. Smartphone users around the world - statistics and facts [infographic], 2013. URL http://www.go-gulf.com/blog/smartphone/. Accessed 2014-03-20.

[3] AtB. Årsmelding 2012, 2013. URL https://www.atb.no/getfile.php/Filer/Diverse%20informasjon/AtB_%C3%A5rsmelding_2012.pdf. Accessed: 2014-02-23.

[4] Hordaland Fylkeskommune. Kollektivmeldinga 2012, 2013. URL http://www.hordaland.no/Global/samferdsel/Filer/Kollektivmeldinga%202012.pdf. Accessed: 2014-02-24.

[5] Finnmark Fylkeskommune. Billett via mobiltelefon, 2014. URL http://www.177finnmark.no/Reiseinfo/mobilett. Accessed: 2014-02-24.

[6] Kolumbus. Om kolumbus, 2014. URL http://www.kolumbus.no/om-kolumbus/. Accessed: 2014-02-24.

[7] Kolumbus. Årsrapport 2012, 2013. URL http://www.kolumbus.no/Documents/AArsrapporter/%C3%85rsrapport%202012.pdf. Accessed: 2014-02-24.

[8] Ruter. Ruter - kollektivtrafikken i oslo og akershus, 2014. URL https://ruter.no/om-ruter/. Accessed: 2014-02-24.

[9] Ruter. Ruter 2012 - kollektivtrafikken øker mest, 2013. URL https://ruter.no/Documents/Presse/Dokumenter_Vedlegg/Ruters_presentasjon_0603_2013.pdf. Accessed: 2014-02-24.

[10] Ruter. Ruter lanserer mobilbillett, 2012. URL https://ruter.no/verdt-a-vite/presse/pressemeldinger/ruter-lanserer-mobilbillett/. Accessed: 2014-02-24.

[11] Berit B. Njarga. Nsb lanserer billett-app, 2011. URL http://www.dinside.no/883086/nsb-lanserer-billett-app. Accessed: 2014-02-25.

[12] NSB. Årsrapport 2012, 2013. URL http://www.nsbkonsernet.no/no/nyheter-og-media/publikasjoner/_attachment/182?_download=true&_ts=13de0b90e28. Accessed: 2014-02-24.

[13] AKT. Om agder kollektivtrafikk as, 2014. URL http://akt.no/info/kundeservice/om-agder-kollektivtrafikk-as/. Accessed: 2014-02-25.

[14] Eirin Aadland Haga. Mobilbilletten venter på nfc-teknologien, 2013. URL http://sornett.no/arkiv/77584. Accessed: 2014-02-25.

[15] Flybussen. Ny app og ny avgang, 2013. URL http://flybussen.no/Oslo/dokument/3344. Accessed: 2014-02-25.

[16] Flybussen. Vi tilbyr mobilbillett for flybussen i bergen - spørsmål og svar, 2013. URL http://flybussen.no/Bergen/dokument/3547. Accessed: 2014-02-25.

[17] Kystbussen. Om kystbussen, 2014. URL http://kystbussen.no/vis.aspx?side=30. Accessed 2014-03-31.

[18] NOR-WAY Bussekspress AS. Om nor-way bussekspress as, 2014. URL http://www.nor-way.no/om-oss/. Accessed 2014-03-31.

[19] Tide ASA. Kystbussen får app og mobilvennlige sider, 2012. URL http://www.tide.no/View.aspx?mid=1406&itemid=1551&pageid=1106&moduledefid=55. Accessed 2014-03-31.

[20] SAS Group. Largest shareholders, 2014. URL http://www.sasgroup.net/SASGroup/default.asp. Accessed 2014-02-25.

[21] SAS Group. Sas group 2013 - corporate presentation, 2013. URL http://www.sasgroup.net/SASGROUP_FACTS/CMSForeignContent/SASGroup_company_presentation_2013.pdf. Accessed 2014-02-25.

[22] Knut-Erik Mikalsen. Sas vant app-kappløpet mot norwegian, 2013. URL http://reise.adressa.no/reise/article41566.ece#.Uwx-hPl5PTo. Accessed 2014-02-25.

[23] Norwegian. Velkommen til norwegian, 2014. URL http://www.norwegian.no/om-norwegian/fakta/. Accessed 2014-02-25.

[24] Leif Martin Kirknes. Norwegian med reise-app, 2012. URL http://www.idg.no/computerworld/article249291.ece. Accessed 2014-02-25.

[25] Billettservice. Om oss, 2014. URL http://www.billettservice.no/help/Footer/billettservice.aboutUsnewtoo.NO.html?l=no-no&tm_link=tm_i_1&language=no-no. Accessed 2014-03-02.

[26] Billettservice. Mobilbillett, 2014. URL http://www.billettservice.no/help/Footer/billettservice.mobilbillett.NO.html?l=no-no&language=no-no. Accessed 2014-03-03.

[27] TicketCo. About us, 2013. URL https://ticketco.no/about_us. Accessed 2014-03-01.

[28] Hoopla. Om hoopla, 2014. URL http://www.hoopla.no/. Accessed 2014-03-02.

[29] eArrangement. earrangement as, 2014. URL http://www.earrangement.no/?ac_id=119&ac_parent=1. Accessed 2014-03-01.

[30] Hiroko Kato, Douglas Chai, and Keng T Tan. *Barcodes for mobile devices*. Cambridge University Press, 2010.

[31] N.J. Woodland and S. Bernard. Classifying apparatus and method, oct 1952. URL http://www.google.com/patents/US2612994. US Patent 2,612,994.

[32] Margalit Fox. Alan haberman, who ushered in the barcode, dies at 81, 2011. URL http://www.nytimes.com/2011/06/16/business/16haberman.html?_r=1&hp. Accessed: 2014-02-20.

[33] Masabi LTD. Connecting the dots: An introduction to 2d barcodes, 2011. URL http://www.masabi.com/2011/03/04/connecting-the-dots-an-introduction-to-2d-barcodes-3/. Accessed: 2014-02-19.

[34] Iata. Press release no.: 35", October 2007. URL http://www.iata.org/pressroom/pr/Pages/2007-11-10-01.aspx. Accessed: 2014-02-20.

[35] ADC Denso. *Qr code essentials*, 2011. URL http://www.nacs.org/LinkClick.aspx?fileticket=D1FpVAvvJuo%3D&tabid=1426&mid=4802.

[36] Bjørn O. Fjellandsbø. Regjeringen sier nei til kontantløse busser, 2014. URL http://www.nhoservice.no/article.php?articleID=4809. Accessed 2014-03-03.

[37] AtB. Priser og billettyper, 2014. URL https://www.atb.no/priser/category550.html. Accessed 2014-03-03.

[38] Skyss. Prisar, 2014. URL https://skyss.no/nn-NO/skysskortet-og-prisar1/Prisar/. Accessed 2014-03-03.

[39] Ruter. Billetter og priser i oslo og akershus, 2014. URL https://ruter.no/billetter/oslo_og_akershus/. Accessed 2014-03-03.

[40] Trine Aaen Espen Sandmo. Hard kritikk mot ukas betalingskort, 2011. URL http://www.nrk.no/trondelag/hard-kritikk-mot-ukas-betalingskort-1.7836219. Accessed 2014-03-04.

[41] Mark Weiser. The computer for the 21st century. *Scientific american*, 265 (3):94–104, 1991.

[42] Mark Weiser. Ubiquitous computing, 1996. URL http://www.ubiq.com/hypertext/weiser/UbiHome.html/. Accessed: 2014-02-17.

[43] Dan Chalmers. *Sensing and systems in pervasive computing: Engineering context aware systems*. Springer, 2011.

[44] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[45] Friedemann Mattern and Eth Zurich. Wireless future: Ubiquitous computing. In *In: Proceedings of Wireless Congress 2004*, 2004.

[46] R. Want and T. Pering. System challenges for ubiquitous pervasive computing. In *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, pages 9–14, May 2005. doi: 10.1109/ICSE.2005.1553532.

[47] Oxford University Press. Oxford dictionaries online, 2014. URL http://www.oxforddictionaries.com/. Accessed: 2014-02-17.

[48] Anders Kofod-Petersen. *A case-based approach to realising ambient intelligence among agents*. PhD thesis, Norwegian University of Science and Technology, 2007.

[49] H. Gellersen, G. Kortuem, A. Schmidt, and M. Beigl. Physical prototyping with smart-its. *Pervasive Computing, IEEE*, 3(3):74–82, July 2004. ISSN 1536-1268.

[50] Mark Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84, July 1993. ISSN 0001-0782. doi: 10.1145/159544.159617. URL http://doi.acm.org/10.1145/159544.159617.

[51] Gregory D Abowd and Elizabeth D Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):29–58, 2000.

[52] Roy Want, Gaetano Borriello, Trevor Pering, and Keith I. Farkas. Disappearing hardware. *IEEE Pervasive Computing*, 1(1):36–47, January 2002. ISSN 1536-1268. doi: 10.1109/MPRV.2002.993143. URL `http://dx.doi.org/10.1109/MPRV.2002.993143`.

[53] Norbert A Streitz, Carsten Rocker, Thorsten Prante, Daniel van Alphen, Richard Stenzel, and Carsten Magerkurth. Designing smart artifacts for smart environments. *Computer*, 38(3):41–49, 2005.

[54] Anind K. Dey and Gregory D. Abowd. *Support for the Adapting Applications and Interfaces to Context*, pages 261–296. John Wiley and Sons, Ltd, 2005. ISBN 9780470091708. doi: 10.1002/0470091703.ch13. URL `http://dx.doi.org/10.1002/0470091703.ch13`.

[55] Norsk Telegrambyrå. Bieber kan slå ut telenettet, 2013. URL `http://www.ntb.no/article.aspx?Section=NYHETERPAANYNORSK&RefID=3689040&MainGroup=1`. Accessed: 2014-02-21.

[56] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[57] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[58] Bill N Schilit and Marvin M Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

[59] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[60] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.

[61] Anders Kofod-Petersen and Agnar Aamodt. Contextualised ambient intelligence through case-based reasoning. In T. R. Roth-Berghofer, Mehmet H. Göker, and H. Altay Güvenir, editors, *Proceedings of the Eighth European Conference on Case-Based Reasoning (ECCBR 2006)*, volume 4106

of *Lecture Notes in Computer Science*, pages 211–225, Ölüdeniz, Turkey, September 2006. Springer Verlag.

[62] Saul Greenberg, Nicolai Marquardt, Till Ballendat, Rob Diaz-Marino, and Miaosen Wang. Proxemic interactions: The new ubicomp? *interactions*, 18 (1):42–50, January 2011. ISSN 1072-5520. doi: 10.1145/1897239.1897250. URL `http://doi.acm.org/10.1145/1897239.1897250`.

[63] Albrecht Schmidt. *Ubiquitous computing-computing in context*. PhD thesis, Lancaster University, 2003.

[64] Elisabetta Zibetti, Vicenç Quera, Francesc Salvador Beltran, and Charles Tijus. Contextual categorization: A mechanism linking perception and knowledge in modeling and simulating perceived events as actions. In *Modeling and using context*, pages 395–408. Springer, 2001.

[65] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[66] Anind K Dey, Gregory D Abowd, and Andrew Wood. Cyberdesk: A framework for providing self-integrating context-aware services. *Knowledge-Based Systems*, 11(1):3–13, 1998.

[67] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 92–99. IEEE, 1998.

[68] HamidR. Ekbia and AnaG. Maguitman. Context and relevance: A pragmatic approach. In Varol Akman, Paolo Bouquet, Richmond Thomason, and Roger Young, editors, *Modeling and Using Context*, volume 2116 of *Lecture Notes in Computer Science*, pages 156–169. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42379-9. doi: 10.1007/3-540-44607-9_12. URL `http://dx.doi.org/10.1007/3-540-44607-9_12`.

[69] M. Hussein, J. Han, A. Colman, and Jian Yu. An architecture-based approach to developing context-aware adaptive systems. In *Engineering of Computer Based Systems (ECBS), 2012 IEEE 19th International Conference and Workshops on*, pages 154–163, April 2012. doi: 10.1109/ECBS.2012.13.

[70] WilliamM. Newman, MargeryA. Eldridge, and MichaelG. Lamming. Pepys: Generating autobiographies by automatic tracking. In Liam Bannon, Mike Robinson, and Kjeld Schmidt, editors, *Proceedings of the Second European Conference on Computer-Supported Cooperative Work ECSCW 91*, pages

175–188. Springer Netherlands, 1991. ISBN 978-0-7923-1439-4. doi: 10.1007/978-94-011-3506-1_13. URL http://dx.doi.org/10.1007/978-94-011-3506-1_13.

[71] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 434–441, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303126. URL http://doi.acm.org/10.1145/302979.303126.

[72] Anind K. Dey, Gregory D. Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Hum.-Comput. Interact.*, 16(2):97–166, December 2001. ISSN 0737-0024. doi: 10.1207/S15327051HCI16234_02. URL http://dx.doi.org/10.1207/S15327051HCI16234_02.

[73] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *Workshop Proceedings*, 2004.

[74] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001. URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[75] Harry Chen, Tim Finin, and Anupam Joshi. An ontology for context-aware pervasive computing environments. *Knowl. Eng. Rev.*, 18(3):197–207, September 2003. ISSN 0269-8889. doi: DOI:10.1017/S0269888904000025. URL http://dx.doi.org/DOI:10.1017/S0269888904000025.

[76] Victoria Bellotti, Maribeth Back, W. Keith Edwards, Rebecca E. Grinter, Austin Henderson, and Cristina Lopes. Making sense of sensing systems: Five questions for designers and researchers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 415–422, New York, NY, USA, 2002. ACM. ISBN 1-58113-453-3. doi: 10.1145/503376.503450. URL http://doi.acm.org/10.1145/503376.503450.

[77] Julie Bort. Clippy, microsoft's talking paperclip, is back, 2013. URL http://www.businessinsider.com/clippy-microsofts-talking-paperclip-is-back-and-linus-torvalds-loves-it-2013-5. Accessed 2014-03-10.

[78] Chris Gentilviso. The 50 worst inventions, 2010. URL http://content.time.com/time/specials/packages/article/0,28804,1991915_1991909_1991755,00.html. Accessed 2014-03-10.

[79] Qingsheng Zhang, Yong Qi, Jizhong Zhao, Di Hou, Tianhai Zhao, and Liang Liu. A study on context-aware privacy protection for personal information. In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 1351–1358, Aug 2007. doi: 10.1109/ICCCN.2007.4318009.

[80] W.K. Edwards. Discovery systems in ubiquitous computing. *Pervasive Computing, IEEE*, 5(2):70–77, April 2006. ISSN 1536-1268. doi: 10.1109/MPRV.2006.28.

[81] Edward T Hall. *The hidden dimension*. A Doubleday anchor book. Doubleday, 1966. ISBN 9780385084765.

[82] Humphry Osmond. Function as the basis of psychiatric ward design. *Psychiatric Services*, 8(4):23–27, 1957.

[83] Nicolai Marquardt and Saul Greenberg. Informing the design of proxemic interactions. *IEEE Pervasive Computing*, 11(2):14–23, 2012.

[84] Saul Greenberg and Hideaki Kuzuoka. Using digital but physical surrogates to mediate awareness, communication and privacy in media spaces. *Personal Technologies*, 3(4):182–198, 1999.

[85] José Rouillard. Contextual qr codes. In *Proceedings of the 2008 The Third International Multi-Conference on Computing in the Global Information Technology (Iccgi 2008)*, ICCGI '08, pages 50–55, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3275-2. doi: 10.1109/ICCGI.2008.25. URL `http://dx.doi.org/10.1109/ICCGI.2008.25`.

[86] Chi-Yin Chow and Mohamed F. Mokbel. Trajectory privacy in location-based services and data publication. *SIGKDD Explor. Newsl.*, 13(1):19–29, August 2011. ISSN 1931-0145. doi: 10.1145/2031331.2031335. URL `http://doi.acm.org/10.1145/2031331.2031335`.

[87] N. Armstrong, C.D. Nugent, G. Moore, and D.D. Finlay. Developing smartphone applications for people with alzheimer's disease. In *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on*, pages 1–5, Nov 2010. doi: 10.1109/ITAB.2010.5687795.

[88] Emmanuel Dubois and Laurence Nigay. Augmented reality: Which augmentation for which reality? In *Proceedings of DARE 2000 on Designing Augmented Reality Environments*, DARE '00, pages 165–166, New York, NY, USA, 2000. ACM. doi: 10.1145/354666.354695. URL `http://doi.acm.org/10.1145/354666.354695`.

[89] Marco de Sá and Elizabeth Churchill. Mobile augmented reality: Exploring design and prototyping techniques. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pages 221–230, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1105-2. doi: 10.1145/2371574.2371608. URL `http://doi.acm.org/10.1145/2371574.2371608`.

[90] Len Bass, Paul Clements, and Rick Kazman. *Software architecture in practice*. Addison-Wesley Professional, 2003.

[91] Google Inc. Android, the world's most popular mobile platform, 2014. URL `http://developer.android.com/about/index.html`. Accessed 2014-03-12.

[92] Businessweek Ashlee Vance. Behind the internet of things is android and its everywhere, 2014. URL `http://www.businessweek.com/articles/2013-05-29/behind-the-internet-of-things-is-android-and-its-everywhere`. Accessed 2014-03-12.

[93] P. Moore, M. Sharma, F. Xhafa, and L. Barolli. Context and the cloud: Situational awareness in mobile systems. In *Network-Based Information Systems (NBiS), 2013 16th International Conference on*, pages 551–558, Sept 2013. doi: 10.1109/NBiS.2013.92.

[94] Telenor Norge. Apple holder stand på mobiltoppen, 2014. URL `http://www.mynewsdesk.com/no/telenor/pressreleases/apple-holder-stand-paa-mobiltoppen-968584`. Accessed 2014-03-12.

[95] R Llamas, Ryan Reith, and Michael Shirer. Android pushes past 80leap 156.0 URL `http://www.idc.com/getdoc.jsp?containerId=prUS24442013`. Accessed: 2014-02-21.

[96] Navigation National Coordination Office for Space-Based Positioning and Timing. Official u.s. government information about the global positioning system (gps) and related topics, 2014. URL `http://www.gps.gov`. Accessed 2014-05-18.

[97] A. El-Rabbany. *Introduction to GPS: The Global Positioning System*. Artech House mobile communications series. Artech House, 2002. ISBN 9781580531832. URL `http://books.google.no/books?id=U2JmghrrB8cC`.

[98] CharuC. Aggarwal and Jiawei Han. A survey of rfid data processing. In Charu C. Aggarwal, editor, *Managing and Mining Sensor Data*, pages 349–382. Springer US, 2013. ISBN 978-1-4614-6308-5. doi: 10.1007/978-1-4614-6309-2_11. URL `http://dx.doi.org/10.1007/978-1-4614-6309-2_11`.

[99] SA Weis SA. Rfid (radio frequency identification): Principles and applications. *Retrived from www. eecs. harvard. edu/rfid-article. pd f on*, 1, 2011.

[100] K. Ok, V. Coskun, M.N. Aydin, and B. Ozdenizci. Current benefits and future directions of nfc services. In *Education and Management Technology (ICEMT), 2010 International Conference on*, pages 334–338, Nov 2010. doi: 10.1109/ICEMT.2010.5657642.

[101] M. Csapodi and A. Nagy. New applications for nfc devices. In *Mobile and Wireless Communications Summit, 2007. 16th IST*, pages 1–5, July 2007. doi: 10.1109/ISTMWC.2007.4299077.

[102] Jacki Katzman. Nfc forum unveils technology architecture and announces initial specifications and mandatory tag format support, June 2006. URL `http://nfc-forum.org/newsroom/nfc-forum-unveils-technology-architecture-and-announces-initial-specifications-and-mandatory-tag-format-support/`. Accessed: 2014-02-21.

[103] U.Biader Ceipidor, C.M. Medaglia, A. Marino, M. Morena, S. Sposato, A. Moroni, P. Di Rollo, and M.La Morgia. Mobile ticketing with nfc management for transport companies. problems and solutions. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6, Feb 2013. doi: 10.1109/NFC.2013.6482446.

[104] National Aeronautics and Science Mission Directorate Space Administration. Infrared waves, 2010. URL `http://missionscience.nasa.gov/ems/07_infraredwaves.html`. Accessed 2014-05-26.

[105] The Infrared Processing and Analysis Center (IPAC). Near, mid, and far infrared, 2014. URL `http://www.ipac.caltech.edu/outreach/Edu/Regions/irregions.html`. Accessed 2014-05-26.

[106] Herschel Space Observatory. What is infrared light?, 2014. URL `http://herschel.cf.ac.uk/science/infrared`. Accessed 2014-05-26.

[107] HTC Inc. Htc opensense overview, 2014. URL `https://www.htcdev.com/devcenter/opensense-sdk/opensense-overview`. Accessed 2014-05-26.

[108] Google Inc. Consumerirmanager, 2014. URL `https://developer.android.com/reference/android/hardware/ConsumerIrManager.html`. Accessed 2014-05-26.

[109] J.M. Kahn and J.R. Barry. Wireless infrared communications. *Proceedings of the IEEE*, 85(2):265–298, Feb 1997. ISSN 0018-9219. doi: 10.1109/5.554222.

[110] Bluetooth SIG. Bluetooth, 2014. URL `https://www.bluetooth.com`. Accessed 2014-05-18.

[111] Keith W Ross and James F Kurose. *Computer Networking: A Top-Down Approach Featuring the Internet: Preliminary Edtion*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[112] Zigurd Mednieks, Laird Dornin, G Blake Meike, and Masumi Nakamura. *Programming Android: Java Programming for the New Generation of Mobile Devices*. O'Reilly Media, Inc., 2012.

[113] E. Mackensen, M. Lai, and T.M. Wendt. Bluetooth low energy (ble) based wireless sensors. In *Sensors, 2012 IEEE*, pages 1–4, Oct 2012. doi: 10.1109/ICSENS.2012.6411303.

[114] R. Tabish, A. Ben Mnaouer, F. Touati, and A.M. Ghaleb. A comparative analysis of ble and 6lowpan for u-healthcare applications. In *GCC Conference and Exhibition (GCC), 2013 7th IEEE*, pages 286–291, Nov 2013. doi: 10.1109/IEEEGCC.2013.6705791.

[115] Bluetooth Developer Portal. Generic attribute profile, 2014. URL `https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx`. Accessed 2014-03-18.

[116] Qualcomm. Ieee802.11ac: The next evolution of wi-fi standards, 2012. URL `http://www.qualcomm.com/media/documents/files/ieee802-11ac-the-next-evolution-of-wi-fi.pdf`. Accessed 2014-03-20.

[117] A. Baniukevic, C.S. Jensen, and Hua Lu. Hybrid indoor positioning with wi-fi and bluetooth: Architecture and performance. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 207–216, June 2013. doi: 10.1109/MDM.2013.30.

[118] Apple Inc. ios: Use airdrop to wirelessly share content, 2014. URL `http://support.apple.com/kb/ht5887`. Accessed 2014-03-20.

[119] Google Inc. Android - bluetooth low energy, 2014. URL `http://developer.android.com/guide/topics/connectivity/bluetooth-le.html`. Accessed 2014-05-14.

[120] Apple Inc. ios developer library - core bluetooth overview, 2014. URL `https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothOverview/CoreBluetoothOverview.html#//apple_ref/doc/uid/TP40013257-CH2-SW1`. Accessed 2014-05-14.

[121] Bluetooth SIG. Bluetooth core version 4.1, 2014. URL `https://www.bluetooth.org/en-us/specification/adopted-specifications`. Accessed 2014-04-10.

[122] Adam Warski. How do ibeacons work?, 2014. URL `http://www.warski.org/blog/2014/01/how-ibeacons-work/`. Accessed 2014-04-10.

[123] Tony Smith. Build your own apple ibeacon with a raspberry pi, 2013. URL `http://www.theregister.co.uk/Print/2013/11/29/feature_diy_apple_ibeacons/`. Accessed 2014-04-10.

[124] K. Heurtefeux and F. Valois. Is rssi a good choice for localization in wireless sensor network? In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*, pages 732–739, March 2012. doi: 10.1109/AINA.2012.19.

[125] O.G. Adewumi, K. Djouani, and A.M. Kurien. Rssi based indoor and outdoor distance estimation for localization in wsn. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1534–1539, Feb 2013. doi: 10.1109/ICIT.2013.6505900.

[126] Qian Dong and W. Dargie. Evaluation of the reliability of rssi for indoor localization. In *Wireless Communications in Unusual and Confined Areas (ICWCUCA), 2012 International Conference on*, pages 1–6, Aug 2012. doi: 10.1109/ICWCUCA.2012.6402492.

[127] Jeff Sauro. Measuring usability with the system usability scale (sus), 2011. URL `http://www.measuringusability.com/sus.php`. Accessed 2014-05-15.

[128] James R Lewis and Jeff Sauro. The factor structure of the system usability scale. In *Human Centered Design*, pages 94–103. Springer, 2009.

[129] R. Lowe, P. Mandl, and M. Weber. Context directory: A context-aware service for mobile context-aware computing applications by the example of google android. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 76–81, March 2012. doi: 10.1109/PerComW.2012.6197616.

[130] P. Silapachote, A. Srisuphab, R. Satianrapapong, W. Kaewpijit, and N. Waragulsiriwan. A context-aware system for navigation and information dissemination on android devices. In *TENCON 2013 - 2013 IEEE Region 10 Conference (31194)*, pages 1–4, Oct 2013. doi: 10.1109/TENCON.2013.6718484.

[131] J. Punjabi, S. Parkhi, G. Taneja, and N. Giri. Relaxed context-aware machine learning midddleware (rcamm) for android. In *Intelligent Computational Systems (RAICS), 2013 IEEE Recent Advances in*, pages 92–97, Dec 2013. doi: 10.1109/RAICS.2013.6745453.

[132] Daniel Siewiorek, Andreas Krause, Neema Moraveji, Asim Smailagic, Junichi Furukawa, Kathryn Reiger, Fei Lung Wong, and Jeremy Shaffer. Sensay: A context-aware mobile phone. In *2012 16th International Symposium on Wearable Computers*, pages 248–248. IEEE Computer Society, 2003.

[133] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. Contextphone: A prototyping platform for context-aware mobile applications. *Pervasive Computing, IEEE*, 4(2):51–59, 2005.

[134] Apple Inc. ios: Understanding ibeacon, 2014. URL `http://support.apple.com/kb/HT6048`. Accessed 2014-05-19.

[135] Qualcomm Inc. Gimbal: Context awareness and proximity for highly relevant consumer engagements, 2014. URL `http://www.qualcomm.com/solutions/gimbal`. Accessed 2014-05-19.

[136] A. Baniukevic, C.S. Jensen, and Hua Lu. Hybrid indoor positioning with wi-fi and bluetooth: Architecture and performance. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 1, pages 207–216, June 2013. doi: 10.1109/MDM.2013.30.

[137] Marko Helen, Juha Latvala, Hannu Ikonen, and Jarkko Niittylahti. Using calibration in rssi-based location tracking system. In *Proc. of the 5th World Multiconference on Circuits, Systems, Communications & Computers (CSCC20001)*. Citeseer, 2001.