The aWearable Toolkit

Supporting End-User Customization of
Embodied Location-Aware Applications

Maria Møller
Ingrid Constanse Tarøy

Master of Science in Informatics
Submission date: June 2013
Supervisor: Monica Divitini, IDI
Co-supervisor: Simone Mora, IDI

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

Problem Description

The task is in the research areas of embodied social interaction and end-user development, with focus on awareness of location between users. End-user development aims to take user involvement to the next level, by creating tools to help end-users develop and customize their own computer systems. End-users know best what they need and by enabling them to customize applications they can get exactly that. Embodied social interaction moves computing out of the desktop and into physical everyday objects. There is a need to keep track of where the technology has gone, and location-awareness have become a popular topic.

In this master thesis, we want to support end-user customization of embodied location- awareness applications. To answer the research question, we will develop a toolkit prototype, called aWearable. We will conduct user evaluations to determine how users are able to customize an application with the toolkit, and use this application to locate a friend.

Assignment given: August 2012
Supervisor: Monica Divitini, IDI

Abstract

We live in a mobile society where we are constantly surrounded by technology. Users are becoming technologists and many have been introduced to programming at some level. Previous work in HCI has been based on involving end-users in the design process. Now there is also focus on end-user development, which allows end-users to develop and customize systems.

Embodied Social Interaction is another field that is getting attention these days. The main idea is to move computing out of the desktop and into physical objects. It integrates technology into everyday environments, and incorporates human's understandings of the physical and social world. Embodied social interaction is closely related to location-aware computing. Technology has become mobile and there is a need to keep track of where it has gone. By embodying location-awareness, we can free the hands of users and present information in the periphery of a person's attention, limiting distraction.

This master thesis combines end-user development and embodied location-awareness. We wish to provide users with tools to tailor applications to fit their needs. The main research question concerns how to support end-user customization of embodied location-awareness applications. As a result we have developed a prototype, the aWearable toolkit for end-users.

We started out with an exploratory study of embodied social interaction, which was continued with a research study of embodied location-aware applications and end-user toolkits. Results of the research studies were used to define a set of scenarios that helped us identify requirements. After implementing the toolkit we conducted user evaluations where we looked at how well users were able to customize and use an application. These user-evaluations, along with the research studies, helped us answer the research questions.

Sammendrag

Vi lever i et mobilt samfunn hvor vi er konstant omgitt av teknologi. Brukere blir teknologer og mange har blitt introdusert til programmering på et eller annet nivå. Tidligere arbeid i MMI har vært basert på å involvere sluttbrukerne i designprosessen. Nå er det også fokus på sluttbruker- utvikling, der sluttbrukeren utvikler og tilpasser systemer selv.

Sosial kroppslig-gjort interaksjon er et annet felt som får oppmerksomhet i disse dager. Hovedideen er å flytte databehandling ut av datamaskinen og inn i fysiske objekter. Målet er å integrere teknologien inn i hverdagen og inkorporerer menneskets forståelse av den fysiske og sosiale verden. Sosial kroppslig-gjort interaksjon er nært knyttet til lokasjonsbevisste programmer. Teknologien har blitt mobil og det er et behov for å holde rede på hvor den er. Ved å kroppsliggjøre lokasjonsbevissthet kan vi frigjøre hendene på brukerne og gi informasjon i periferien av en persons oppmerksomhet, noe som begrenser distraksjon.

Denne masteroppgaven kombinerer sluttbrukerutvikling og kroppslig-gjort lokasjonsbevissthet. Vi ønsker å gi brukerne verktøy for å skreddersy applikasjoner tilpasset til deres egne behov. Den overordnede problemstillingen handler om hvordan vi kan støtte sluttbruker- tilpasning av lokasjonsbevisste applikasjoner. Som et resultat har vi utviklet en prototype kalt aWearable, som er et verktøy for sluttbrukerne.

Vi startet med et utforskende studie av sosial kroppslig-gjort interaksjon, og fortsatte med et forskningsstudie av kroppslig-gjorte lokasjonsbevisste applikasjoner og verktøy ment for sluttbrukere. Fra disse studiene definerte vi et sett med scenarier som hjalp oss med å identifisere krav for verktøyet. Etter å ha implementert prototypen gjennomførte vi brukerevalueringer der vi så på hvorvidt brukerne var i stand til å tilpasse sine egne applikasjoner og i tillegg bruke denne applikasjonen. Disse bruker-evalueringene, sammen med forskningsstudiene, hjalp oss å besvare forskningsspørsmålene.

Preface

This is the master thesis of Maria Møller and Ingrid Tarøy in the course IT3901 - Informatics Postgraduate Thesis: System Engineering and Human-Computer Interaction, written from August 2012 to June 2013 at the Norwegian University of Science and Technology (NTNU).

We would like to thank our supervisor, Monica Divitini, for constructive and valuable feedback throughout the course of this thesis. We are very satisfied with the collaboration and the commitment you have shown. We also thank our co-supervisor, Simone Mora, for all the help and feedback. Your support has been greatly appreciated.

In addition, we would like to thank those who helped us evaluate our prototype. Especially, Babak A. Farshchian, who was our expert evaluator.

Trondheim, Juni 1, 2013

Maria Møller   Ingrid Tarøy

Contents

1 Introduction **1**
1.1 Problem Definition 1
1.2 Problem Domain 2
1.3 Motivations 4
1.4 Research Questions 4
1.5 Research Approach 5
1.6 Expected Results 8
1.7 Report Outline 8

2 Research Study: Embodied-Social Interaction **10**
2.1 Embodied Interaction 10
2.2 Location-Awareness 11
2.3 Social Interaction 14
2.4 Related Work 15
2.4.1 Discussion 21
2.4.2 Lessons learned 22

3 Research Study: End-User Toolkits **25**
3.1 End-User Development 25
3.2 Toolkits 26
3.3 Related Work 28
3.3.1 Discussion 30
3.3.2 Lessons Learned 31

4 Problem Refinement **32**
4.1 Problem Summary 32
4.2 Scenarios 32
4.2.1 Scenario 01: At the Movies 33
4.2.2 Scenario 02: Emergency work 34
4.2.3 Scenario 03: Roommates 35
4.2.4 Scenario 04: Festival 35
4.2.5 Discussion 36
4.3 Design Space 38
4.4 High Level Requirements 40
4.4.1 Manage groups 41
4.4.2 Customize an application 41

4.4.3 Use the application . . . 42
4.5 Functional Requirements . . . 43
4.6 Non-functional Requirements . . . 46

5 Software Development Platforms 48
5.1 Evaluation . . . 48
5.2 Conclusion . . . 53

6 Design 55
6.1 Design Overview . . . 55
6.1.1 Manage Groups . . . 56
6.1.2 Customize Application . . . 57
6.1.3 Use Application . . . 58
6.2 Design Choices . . . 59
6.2.1 Design Challenges . . . 59
6.2.2 Hardware Design . . . 60
6.2.3 Software Design . . . 62

7 Implementation 69
7.1 Implementation overview . . . 69
7.2 Implementation of the aWearable Web App . . . 70
7.2.1 Integration with Facebook . . . 70
7.2.2 Integration with Noduino . . . 72
7.3 Implementation of the aWearable Device . . . 76
7.3.1 Arduino Hardware . . . 77
7.3.2 Arduino Sketch . . . 78
7.4 Prototype Limitations . . . 81
7.4.1 Requirements Evaluation . . . 81

8 Evaluation 84
8.1 Evaluation method . . . 84
8.2 First Evaluation . . . 85
8.2.1 Evaluation Goal . . . 86
8.2.2 Evaluation Results . . . 86
8.2.3 Discussion . . . 92
8.2.4 Prototype Refinement . . . 93
8.3 Second Evaluation . . . 96
8.3.1 Evaluation Goal . . . 96
8.3.2 Evaluation Results . . . 96
8.3.3 Discussion . . . 99
8.4 Expert Evaluation . . . 100

9 Conclusion 102
9.1 Summary . . . 102
9.2 RQ Discussion . . . 102
9.2.1 Main RQ . . . 102
9.2.2 Sub RQ 1 . . . 103

9.2.3 Sub RQ 2 105
9.3 Reflection on our work 106
9.4 Further Work 107

Bibliography **108**

Appendices **114**

A Appendix **115**
A.1 Guidelines for user Evaluation 115
A.2 Task Description 117
A.3 Questionnaire 118
A.4 Interview 119

B Appendix **120**
B.1 Screenshots 120

List of Figures

1.1 Research Approach 7

2.1 Context feature space [44] 13
2.2 iBand[34] 16
2.3 PolyTags[42] 17
2.4 SuperShoes[29] 18
2.5 BuddyClock[35] 18
2.6 Good Night Lamp[16] 19
2.7 Phigdet eyes[26] 19
2.8 Watchit[13] 20
2.9 Hummingbird[30] 21

3.1 Toolkit for developers [21] 27
3.2 Toolkit for end-users 27
3.3 Ninja Blocks [9] 29
3.4 ReaDIYmates [38] 29
3.5 littleBits [7] 30

4.1 Group structure 38
4.2 Awareness in group 42

5.1 Noduino 50

6.1 Design overview 56
6.2 User perspective: Manage group 57
6.3 User perspective: Customize application 58
6.4 User perspective: Use application 59
6.5 Hardware Interface: Early Prototype 60
6.6 Grove 61
6.7 Hardware Interface: Mock-up 61
6.8 Hardware Interface: Final prototype 62
6.9 Software Interface: Design suggestion 63
6.10 Software Interface: Early design 63
6.11 User interface: Overview part 1 64
6.12 User interface: Overview part 2 65
6.13 User interface: Success message 66
6.14 User interface: Error message 67

7.1 Implementation overview . . . 70
7.2 Facebook related functions . . . 71
7.3 Sequences diagram: Login . . . 71
7.4 Sequences diagram: Add friends . . . 72
7.5 Noduino interface overview . . . 73
7.6 Main class diagram . . . 74
7.7 Sequence diagram: Connect to Arduino . . . 75
7.8 Sequence diagram: Upload application . . . 76
7.9 Fastrax UP501 GPS Module . . . 77
7.10 Arduino related variables and functions . . . 78
7.11 Sequence Diagram: Arduino Sketch . . . 79

8.1 A user testing the application . . . 90
8.2 Sign-up: Before refinement . . . 93
8.3 Sign-up: After refinement . . . 94
8.4 Add Friends: Before and after refinement . . . 95

B.1 Start Up page . . . 120
B.2 Register page . . . 121
B.3 Friends page . . . 122
B.4 Create Application part 1 . . . 123
B.5 Create Application part 2 . . . 124
B.6 About page . . . 125
B.7 How- To page part 1 . . . 126
B.8 How- To page part 2 . . . 127

List of Tables

1.1 Research Questions 4

2.1 Lessons learned: embodied applications 1:2 23
2.2 Lessons learned: embodied applications 2:2 24

3.1 Lessons learned: End-user toolkits 31

4.1 Scenario 01: Application Characteristics 33
4.2 Scenario 02: Application Characteristics 34
4.3 Scenario 03: Application Characteristics 35
4.4 Scenario 04: Application Characteristics 36
4.5 Design Space 40
4.6 Software: Functional Requirements 43
4.7 Hardware: Functional Requirements 45
4.8 Non-functional Requirements 47

5.1 Development Platforms Summary 54

7.1 Software: Functional requirements evaluation 82
7.2 Hardware: Functional requirements evaluation 82

8.1 Evaluation Method 84

1 Introduction

The main objective of this thesis was to support end-user customization of embodied applications that facilitate location-awareness between users. We have developed the aWearable toolkit as a proof of concept, which incorporates aspects of embodied social interaction, location-awareness and end-user development into one tool. The purpose of the toolkit is to make it easy for end-users to custom their needs for location-awareness in different settings, into a physical device. The challenge was to create an intuitive toolkit where there is a clear connection between the customization and the use of an application, and that does not require users to have specific prerequisites. In this chapter we will introduce the problem, the problem domain and our motivations, followed by our research questions and the research approach. The last section will give an overview of the proceeding chapters of this thesis.

1.1 Problem Definition

Advances in ubiquitous and handheld computing have contributed to move technology out into our everyday world and away from traditional desktop computing. Technology has become mobile and users have got more freedom [17]. We carry smartphones with us wherever we go that have integrated GPS sensors and provides map services. As a result, applications that utilizes location information have become popular and a simple search on App Store gives hundreds of hits. 'Find My Friends' lets the users locate friends and family members in real-time, displaying their positions on a map. 'Glympse' is another example that lets users send glymses via sms, email or social networks, allowing friends to see each other's location for a limited period of time. The 'GPS Tracking' app offers turn-by-turn directions to help users find each other.

Location-awareness applications have mainly been developed for smartphones, but the emergence of embodied interaction and tangible technologies gives opportunities to look at such applications differently. Ubiquitous computing is not just about mobility; it is also about incorporating technology into everyday physical objects [53]. Interaction with smartphones and tablets requires a lot of attention. Embodied interaction leverages our physical presence in the real world and is socially embedded within our real world practices and purposes [18]. It means that people act through technology that is inseparable from the real world, and to design for embodied interaction is to construct tools that participate in the real world rather than standing apart from it [23]. By taking advantage of such interaction, we can develop interfaces that fit within their environment and present information in the periphery of a person's attention.

We have become frequent users of applications in all kinds of situations. We expect that the sort of applications we need exists. However, different people and different settings require different support. Von Hippel [51] states that products and services must be accurately responsive to users' needs in order to become successful. Users' needs are changing rapidly, and understanding these needs is a costly matter. It is also the case that users do not know exactly what they want at the start of a design process [51]. A lot of work is required to meet everybody's expectations. By providing end-users with tools to develop and tailor their own applications, we can simplify the process of obtaining user requirements [37]. In addition, end-users know the domain and context of use best, and they can tailor their applications according to their specific needs [47].

By providing end-users with a tool to customize embodied location-awareness applications, we can enable users to create personal applications tailored to their needs that offers them awareness of their friends' locations in relation to where they are located. The application is physically embedded, and will therefore leave the users with free hands and let them attend to other matters while the application works in the periphery of their attention. We have called the toolkit aWearable based on the terms awareness and wearable. We will develop a prototype of this toolkit.

1.2 Problem Domain

The problem resides in three domains; embodied social interaction, location-awareness and end-user development. We want to support end-users in customizing their own location awareness applications that takes advantage of embodied social interaction.

Embodied Social Interaction: *"Interaction with computer systems that occupy our world of social and physical reality and exploit this fact in how they interact with us"* [18]

Embodied social interaction is about being grounded in everyday experience and it incorporates understandings of both the physical and the social world. The term was introduced by Dourish in 2001 [18] and has now become an established field of research in HCI [31]. Both social and physical computing is supported by the idea of embodiment. We encounter the social and physical world directly rather than abstractly, in an embodied manner [18]. The research field of embodied interaction includes a wide range of interests. One example is game consoles like Wii and Xbox Kinect where users use real body gestures to interact with games. Other examples are embodying computation into physical objects that are tangible and/or wearable [31]. For the aWearable tool we will focus on the tangible and wearable aspect of embodied interaction. It is the location-awareness application that users customize, that will incorporate the physical aspect into the toolkit.

Awareness: *"An understanding of the activities of others, which provides a context for your own activity"* [20]
Location-awareness: *"An understanding of position in the spatial environment"* [41][45]

Awareness is about having an understanding of what is going on around us and how we make sense of it [20]. The term is connected to knowledge, recognition, not being ignorant, acknowledgement and being conscious [49]. Awareness is closely related to context [20] and space. Location-awareness concerns how we understand our own and others position in the spatial environment. It is about understanding others position in relation to your own position. The purpose of the aWearable toolkit is for end-users to create location-aware applications for sharing location information with their friends. The application will provide users with knowledge about their friends' positions and thereby provide awareness.

End-User Development: *"A set of methods, techniques, and tools that allows users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact"* [37]

End-user development is about providing end-users, who are not necessarily programmers or technical people, with tools to develop or tailor applications and services on their own. User involvement is taken to the next level. As users are becoming more and more technical, there has become more focus on creating systems where end-users can act as developers themselves [37]. The definition above is aimed at software artifacts. End-user development is most common in software development, but for the aWearable toolkit we will extend it to involve tinkering with hardware components as well, based on the involvement of embodied interaction. Further, the definition involves creating, modifying and extending artifacts. We will provide a tool for customizing or modifying applications. End-users are often not interested in learning a lot before they can use a tool or an application. To reach a broader audience we will not require programming knowledge or any other specific prerequisites. The aWearable tool will provide building blocks for application customization.

1.3 Motivations

There are numerous location-awareness applications available today, which implies that people are interested in such applications and that there is a market for it. People who already use location-aware applications on their smartphones might also be interested in trying out a physical location-aware application. By shifting to an embodied location-awareness application users will have free hands and it will not draw as much attention as interacting with a smartphone.

By providing an end-user adapted toolkit, we can meet more people's needs as it leaves more decisions to the user instead of the developer. Users needs are changing rapidly. With an end-user adapted toolkit, applications can be created and changed accordingly, and it can be designed to meet a wide range of applications. The users can make the application according to their own needs and preferences. It might also be extra motivating and give a sense of achievement to use applications that are self-built. As toolkits make development easier, it facilitates rapid generation of prototypes, and can thereby facilitate innovation and creativity [25][24].

1.4 Research Questions

The research questions have been formulated to guide this research and to help evaluate the developed prototype. The questions are listed in table 1.1. We have defined a main research question that has been refined into two sub-questions, one that concerns the customization of an application and one that concerns the use of the customized application.

Main RQ	**How can we support end-user customization of embodied location- awareness applications?**
Sub RQ1	What functionality must be provided to facilitate end-users in customizing embodied location- awareness applications?
Sub RQ2	How should feedback be presented by the end-user-customized application to facilitate location- awareness?

Table 1.1: Research Questions

Main RQ: How can we support end-user customization of embodied location- awareness applications?
The main research question concerns how we can help end-users to customize embodied location-awareness applications. As an answer to this question, we developed the aWearable toolkit for end-users. We will evaluate the toolkit with potential users. We will test with a group of users that have programming experience, and a group of users without this experience, and look at potential differences between

the groups.

Sub RQ 1: What functionality must be provided to facilitate end-users in customizing embodied location- awareness applications?
We want to determine the characteristics of location-awareness applications and what functionality that is required to create such applications. We will also determine special requirements concerning end-user development. We will use the prototype to evaluate if the functionality is relevant and sufficient for customization of embodied location-awareness applications, and evaluate if users are able to understand how to use the toolkit. It is also important that there is a connection between the customization and the use of an application, so that users are able to understand how the application they are tailoring, is going to work in practice.

Sub RQ 2: How should feedback be presented by the end-user-customized application to facilitate location- awareness?
This sub-question is concerned with how the customized application should work to facilitate location-awareness. We want to determine what kind of feedback the users need from the application and how this feedback should be presented, to help them be aware of their friends. To answer the questions, we will have users take their customized application out into the field and see if they are able to use it to find a friend. We are also interested to know in what situations users can picture themselves using the applications.

1.5 Research Approach

We started our master thesis with an exploratory study in the area of embodied social interaction. We read papers about embodied interaction, tangible interfaces and social computing, as these are closely related topics. The aim of this research was to identify a set of problems to continue to work on. We looked particularly at articles that describes physical applications, and excluded articles concerning robotics and frameworks. The articles were categorized based on patterns. After getting an overview of the topic we found that location- awareness is a recurring theme in embodied social interaction and an area that seems to be highly relevant based on the number of location-aware applications that are on the market today. We decided to explore this area further.

We extended the research study on work that combines embodied social interaction with location-awareness, where we focused on characteristics that identify embodied location-aware applications. The applications we looked at were designed for specific situations and tasks. With guidance from our supervisor, we came up with the idea of making an end-user adapted tool to customize location-aware application. That way users can tailor their own applications to their current needs. We went on with a new research study on physical toolkits aimed at end-users. The focus now was on identifying characteristics and functionality that is required of end-user adapted tookits, and to get design ideas for a prototype. The thesis problem and the research questions were formulated based on these pre-studies, and we continued with a refinement of the problem and an analysis of requirements.

Further, we evaluated software development platforms for making web interfaces that communicates with physical toolkits. From this research we also got ideas for metaphors for our own toolkit. We had two iterations with design, implementation and user evaluations. The first iteration had a bigger design and implementation phase than the second one. We had feedback meetings with our supervisor and co-supervisor during the early design stages, where we discussed possible directions.

During the first implementation phase we developed the aWearable toolkit, while we in the second iteration made changes to the toolkit based on results from the first user evaluation. In both iterations we used both non-programmers and programmers to see if there were any differences. We asked them to develop a specific application using the toolkit, and had them think out loud when doing so. We also interviewed them after the test. After the second evaluation, we compared the results with the results from the first one. We ended with an expert evaluation where we focused on possible directions for further development. To sum up the project, we evaluated the results in relation to the research questions and to the project in general. The research approach is illustrated in figure 1.1.

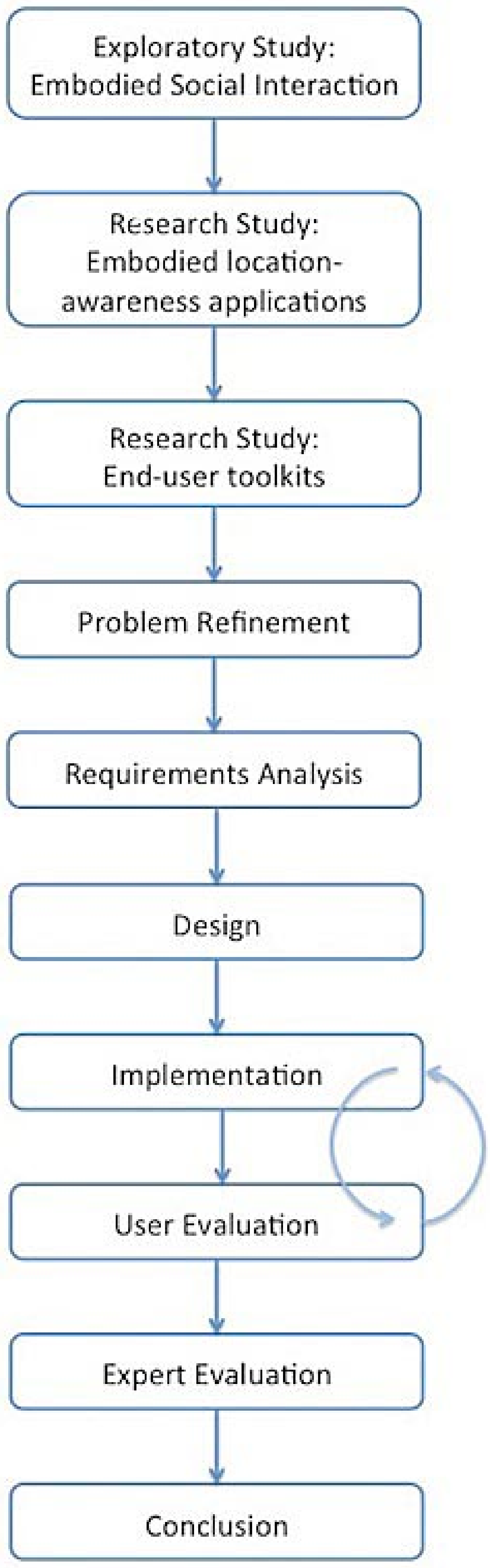

Exploratory Study: Embodied Social Interaction
Research Study: Embodied location-awareness applications
Research Study: End-user toolkits
Problem Refinement
Requirements Analysis
Design
Implementation
User Evaluation
Expert Evaluation
Conclusion

Figure 1.1: Research Approach

1.6 Expected Results

The problem of this thesis is to provide end-users with tools so they can customize their own embodied location-awareness applications according to their specific needs. The expected result is a prototype for an end-user adapted toolkit where end-users can tailor their own embodied location-awareness applications. We will use this prototype in a user evaluation to answer the research questions, and evaluate to what degree users are able to customize an application and then use it.

1.7 Report Outline

Chapter 2 - Research Study: Embodied Social Interaction
This chapter consists of two parts, theoretical background on embodied social interaction, and related work on embodied location-aware applications. The theoretical background starts with a closer look at embodied interaction and terms that are related to our thesis. We continued by looking at how embodied interaction incorporates context and location-awareness, before elaborating about the social interaction aspect. The related work part examines a selection of physical location-awareness applications. The chapter ends with a discussion and lessons learned from the research study.

3 - Research Study: End-User toolkits
This chapter is structured in the same way as chapter two. It consists of a theoretical background and a related work section. In the first part we look closer at end-user development and toolkits. In the second part we examine a selection of end-user adapted toolkits that incorporates the physicality of embodied interaction. Also this chapter ends in a discussion and lessons learned.

Chapter 4 - Problem Refinement
In this chapter we use the results from the research studies to define a set of scenarios, which presents examples of settings and applications that could be created using the toolkit. From the scenarios we defined a design space that is refined into a requirements specification.

Chapter 5 - Software Development Platforms
In this chapter we evaluate a selection of different software development platforms that supports connection between a software interface and an Arduino board. The platforms are evaluated as frameworks, and as toolkits for creating applications. The evaluation ends in a conclusion where we determine what platform we will use for the implementation of our prototype.

Chapter 6 - Design
In this chapter we present an overview of the design. It includes the technologies that will be used to develop the different parts of the prototype, and how the parts are connected. We go on with a description of the toolkit from a users point of view, by presenting a set of use cases. The last part of the chapter describes the process

of designing the aWearable toolkit. Here we present design choices that were made and how the aWearable toolkit ended up looking the way it does.

Chapter 7 - Implementation
In this chapter we describe how the prototype was implemented. We divided the chapter into three sections. The first section includes an overview of the different parts that make up the aWearable toolkit, and how they are connected. The second section describes the implementation of the aWearable web application, which includes integration with facebook and integration of the Noduino framework. The last section describes the implementation of the aWearable device. We end the chapter with a discussion on prototype limitations and an evaluation of the requirements in relation to the implementation.

Chapter 8 - Evaluation
In this chapter we present two iterations of user evaluations. Between the evaluations we refined the prototype. Goals and results are described from both evaluations. In addition to the user evaluations, we had an expert evaluation, with focus on further work. This evaluation is also described in this chapter.

Chapter 9 - Conclusion
In this chapter we give a summary of the project, also in relation to the research questions. We reflect on the work we have done, pointing out strengths and weaknesses. The report ends with a discussion of further work.

2 | Research Study: Embodied-Social Interaction

As described in the introduction, the main objective of this thesis is to create a toolkit, aimed at end-users, within the area of embodied social interaction that facilitates the development of physical location- aware applications. In this chapter we will elaborate on the theoretical background that identifies the characteristics of such applications. It is composed of multiple fields of research: Embodied, tangible and social interaction along with awareness of location and social context. We will define the terms and look at how the theories are related to each other and how they are relevant for our work. Second, we will look at related work in terms of physical applications and get ideas for what our toolkit should provide.

2.1 Embodied Interaction

Into the 21th century research directions in HCI have moved away from traditional textual- and graphical computer interaction and towards new ways of interaction. Computers are being integrated with the "everyday physical world" [18], as opposed to stationary computers where interaction is happening through a keyboard and a mouse. Paul Dourish termed this physical interaction form "Embodied Interaction". The theory was introduced in his pioneering book 'Where the Action is', published in 2001. He describes embodied interaction as; *"interaction with computer systems that occupy our world, a world of physical and social reality, and exploit this fact in how they interact with us"*. It is about moving computer technology out into everyday objects that surrounds us.

Dourish points out that embodied interaction leverages our physical presence in the real world and that it is socially embedded within our real world practices and purposes. By making computer interaction seem more like familiar arenas, embodied interaction takes advantage of our familiarity with real world objects and how we interact with them. Embodiment is not just about the physical, but also about being grounded in everyday experience. It incorporates understandings of both the physical and the social world. "Embodiment is the idea that underlies tangible and social computing." [18].

Through embodied interaction, we can develop interfaces that fit within their environment and that present information in the periphery of a person's attention [18]. This is also what Mark Weiser is talking about when he talks about ubiquitous computing. Weiser defines it as technologies that disappear by being seamlessly integrated into the world. "They weave themselves into the fabric of everyday life until they are indistinguishable from it" [53]. Weiser points out that; "A good tool is an invisible tool", meaning the tool should not take attention away from the task. "Tools are not invisible in themselves, but as part of a context of use" [52]. He emphasized that "the world is not a desktop", which is also the title of one of his articles.

In present times Eva Hornecker presents "tangible and embodied interaction" as an umbrella term for the interest in the physical. "Tangible" reflects physicality, while "embodied," reflects the role of movement-based interaction. Trends in developing low-cost, low-power devices and rapid prototyping toolkits have contributed to the emergence of tangible and embodied interaction as an established field of research. Physical computing communities are growing, where people tinker with electronics and make interactive devices [31]. Wearable computing is another widely used term. As the name suggest, wearable computing is an aspect of physical interaction where computers can be worn.

When we talk about embodied interaction in our thesis, we mean moving computer technology out into everyday objects so people can physically interact with them and receive information from them. Embodied interaction presents information in the periphery of a person's attention, and as Weiser points out, "A good tool is an invisible tool". The subject is relevant for our thesis, as we want to create physical applications that do not take away too much attention from what a user is actually doing. That way the users can focus on their surroundings. We will also use the term wearable as a subcategory of embodiment, where computers are worn in clothes or somewhere on the body.

2.2 Location-Awareness

Embodied social interaction is closely related to context-aware computing. The digital is moved out into the everyday world, and along with the transition comes a need to keep track of where the technology has gone [19]. The user gets increased freedom and mobility and the context becomes more dynamic [17]. There is a need to understand the potential relationship between computation and the context in which it is embedded [19].

What is context?

There have been many attempts to define context. Some relates context to the user's environment, while others relates it to the environment of the application. Dey et al. argues that many definitions of context are too specific, and proposes a more generic definition; *"Context is any information that can be used to characterize the situation of an entity"*. An entity is further defined as *"a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"* [17]. They also introduce a set of context categories. Location, identity, activity and time are highlighted as the most important. These categories are also recurrent, to some extent, in other definitions of context.

Dourish sees context as a relational property between objects or activities. Something may or may not be contextually relevant based on a particular activity, and contextual features are defined dynamically as occasional properties [19]. He argues that "context arises from the activity" and that it is a feature of interaction. Context and activity are both equal parts of Dourish's model of context. He highlights that it is this approach he has been calling embodied interaction. "The essential feature of embodied interaction is the idea of allowing users to negotiate and evolve systems of practice and meaning in the course of their interaction with information systems". The correlation between embodied interaction and context and activity is also evident in his book 'Where the Action Is'; "We are not simply embodied in the world, but the world is the site and setting of all activity. It shapes and is shaped by the activity of embodied agents" [18].

In this setting, we will look at context in relation to location, identity and time. The idea is to share location information between users, who are nearby each other, to make them aware this. The users will have to be at the same approximate location at the same time, and the identity of the user is relevant for who this user wants to be made aware of. Naturally, users must also have the application. Since context is a broad and generic term, we will narrow it further.

What is awareness?

Awareness is another a broad and ambiguous term that it is difficult to give a precise definition of. According to dictionaries, to be aware is synonymous with being conscious and having knowledge. Dourish and Bellotti defines awareness as *"an understanding of the activities of others, which provides a context for your own activity"* [20]. It is about how we pick up what is going on around us and make sense of it, "getting the big picture" [12].

We want users of our tool to be aware of people, also those they cannot not necessarily see. Awareness will be an aspect of the applications users can create with our tool. Location is what we want users to be aware of. The definition can be adapted as follows, an understanding of the location of others, which provides a context for your own activity. Awareness does not necessarily have to be about location, but in our thesis the two is closely related.

What is context-awareness?

Context-awareness is a type of awareness. Dey et al. [17] defines context-aware as follows; *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."* In other words, context-aware systems are systems where information and services are offered to the user based on the context of both the user and the application.

Wearable computing is advancing sensor-based context awareness and "awareness of both the user and the physical environment is considered a distinguishing feature of wearable computers" [44]. Schmidt et al. presents a hierarchical organization of context features, illustrated in figure 2.1. The model is consistent with the context-types; location, identity, activity and time, and answers the questions of who, what, when and where. The third level provides relevant features that determine the context for the different categories.

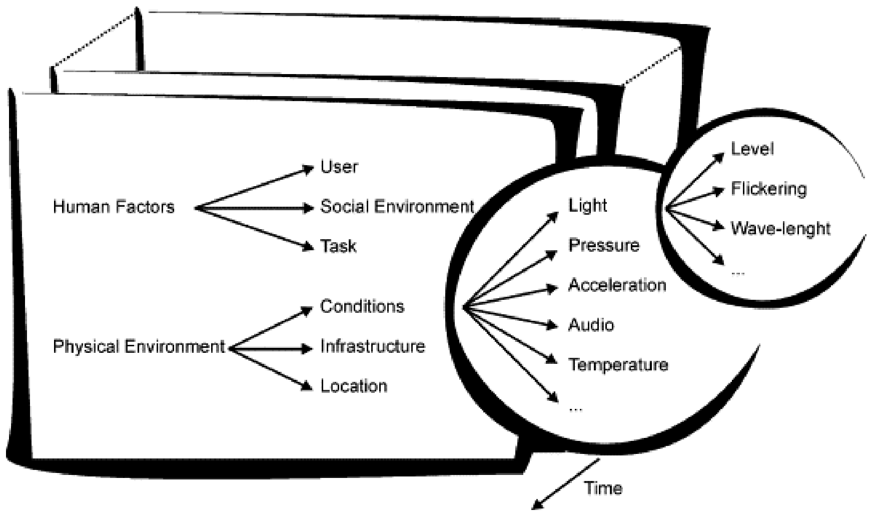


Figure 2.1: Context feature space [44]

Applications created with our toolkit provide context-awareness, where the main context variable is location. It presents information to the user when there are friends in their proximity. The location of a user will be related to the location of others based on who is nearby.

What is location-awareness?

In this project we are focusing on location-awareness. As it appears from the previous sections, location-awareness is a type of awareness that uses location as the context variable. It can be defined as an understanding of position in the spatial environment [45][41]. Weiser stated that "little is more basic to human perception than physical juxtaposition, and so ubiquitous computers must know where they are" [53]. By integrating location-sensors in wearable computers we can provide both the application and the user of the application, with location-awareness. "The system can sense and respond to aspects of the settings in which they are used" [19].

In our tool, location-awareness means that friends are made aware of each other's position or approximate position in the spatial environment, through feedback from a self made application. Location- awareness is the main characteristic of the application and the term that will be much used throughout this thesis. The customized application will need to have a location sensor, and when a user is made aware that a friend is close, the user can choose to respond or ignore this.

2.3 Social Interaction

As briefly mentioned, embodied interaction is not just about physicality; it is also about social interaction. Social action is also embedded in settings, and like tangible interaction, "social interaction draws on the ways we experience the everyday world" [18]. Dourish points out that we encounter the social world directly rather than abstractly "in an embodied manner", and that social computing are attempting to incorporate understandings of the social world into interactive systems. Hornecker and Buur points out that "the support of social interaction and collaboration might be the most important and domain-independent feature of tangible interaction" [32].

With the move of technology into the everyday world, people have become significantly more connected. Numerous social networking applications, like Facebook, Twitter etc., makes it easier to keep in touch and share thoughts, opinions and interests with others. When the tangible and the social are combined, online social communities become accessible when we are on the move, in real-time and non real-time [22].

A traditional area of usage for location-aware applications is when you want to find something, often a place or a person. If you are looking for a person's location it is often because you want to meet. That way, location-aware application can help people to socialize. The awareness is happening in a social context and contributing to social interaction.

What is social context?

Social situation is included as a part of context. For example; the users environment and the connection to people that are located in the user's proximity [43]. Endler et al. [22] looks closer at how social information can be composed into a social context, to facilitate development of social applications from location-awareness. They define what they call situated social context, due to the relation with location, as *"the set of people that share some common spatio-temporal relationship with the individual, which turn them into potential peers for information sharing or interaction in a specific situation".* Social-awareness is closely related to social context. It focuses on how individuals understand the social situation and who is around them [12].

When users share the same approximate location there is a good opportunity for face-to-face communication. The embodied location-awareness application can help people realize this. When users are in each other's proximity their applications will realize this, and exchange location information.

2.4 Related Work

We have done research on embodied applications to get inspiration for applications we want our toolkit to support. Even though our main focus is on location-awareness, we have also included other types of applications that we find relevant, for example applications that embed social interaction. The applications we have looked at provides at least one or more of the following aspects:

- it promotes awareness of others
- it shares location
- it promotes social interaction
- it is embodied

Our toolkit will support applications that promote awareness and sharing of location. By looking at similar applications we hope to get ideas for what kind of information our toolkit should provide. We also want to see how these applications can promote social interaction. We want the created application to be embodied, and will look at how other applications have solved this.

We will start with the more general applications before moving on to the location-aware applications. We are mainly interested in how these applications are designed and what characteristics and functionality they provide.

iBand

iBand [34] is a bracelet that exchanges information when one user shakes hands with another. Handshaking is detected via infrared (IR) transceiver alignment combined with a sensed up-and-down motion synchronized on the two devices in IR contact. The user's wrist must be in a pre-calibrated handshaking orientation to activate the IR transmission.

New users enter their contact information and create a personal LED logo at a kiosk. This data is stored in a database, and the logo and a unique ID code are assigned to their device. When users shake hands their logo and ID are transferred to the other device. The LED- display at the iBand cycles through the personal logos of all the contacts collected during use. When the user returns to the kiosk they can download and view their list of new contacts.

The iBand was evaluated at two events: an exhibition of research innovations in Amsterdam and a research lab launch in London. About 100 people attended each event, during which they were able to have a drink, chat and mingle, and people could voluntarily try the iBand. The iBand worked as an icebreaker, and people felt that they had an excuse to talk to other iBand users. Some participants expressed concern about sharing contact information with everyone they met, and suggested an on and off button.

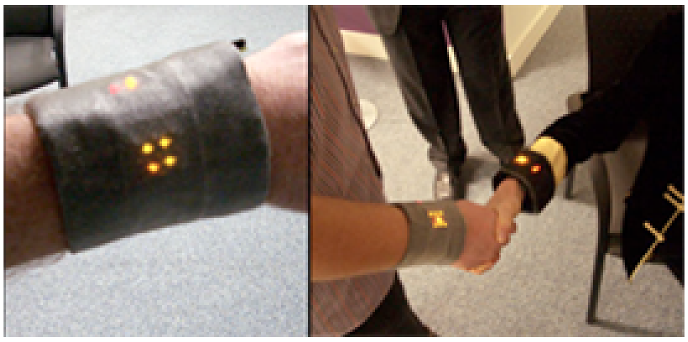


Figure 2.2: iBand[34]

PolyTags

Polytags [42] are objects that can be both interpreted by humans and computers. It is formed as a dice, where half of the sides hold human- readable commands and the opposite half holds marker that can be read by computers.

The proof of concept- prototype consists of a glass table, a web camera and a CPU. The camera is positioned upward under the table to capture the polytag on the table through the glass surface. There is two different polytags in the paper; a 'mood-polytag' and an 'action-polytag'. The mood-polytag is intended to set the status of social networks. When the dice is placed at the table the status at social platforms like Facebook, Twitter and Buzz are updated according to the emoticon facing up. This lets users change a status with minimal attention and loss of focus.

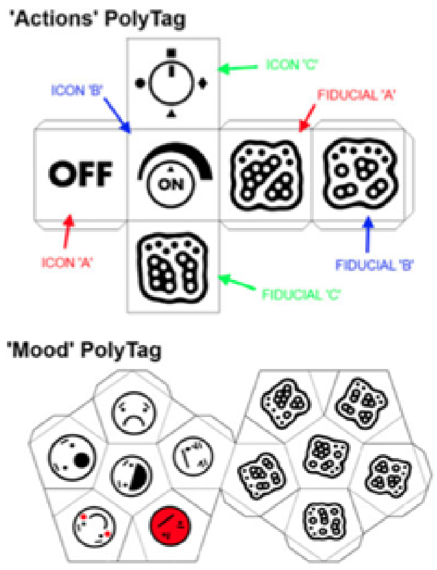


Figure 2.3: PolyTags[42]

SuperShoes

The SuperShoe[29] is a two folded system consisting of an agent, which learns personality, preferences and schedules, and a tangible screen-free shoe interface, which acts as an input/output medium. The SuperShoe uses the integrated GPS from the smartphone, a compass in the shoe, a google calendar to check schedule, facebook for events and friends, Spotify for music preferences, yelp for food recommendations, google maps for directions, google task for location based events, and google places for sightseeing recommendations.

The interface is shoes and they are mental models for travel and navigation. The shoe has three vibrations spots called tickles, under the big toe, under the little toe and at the back of the foot. When receiving a recommendation all the tickles start to vibrate, and the user can accept it by scratching the toe vertically and decline it by scratching the toe horizontally. The user can also use the smartphone to swipe the finger vertically or horizontally. When using the shoe to navigate, the different tickles vibrate to signal the direction. When receiving recommendations, a recommendation card is presented on the smartphone, and if using headphones the user also gets audio cues.

Figure 2.4: SuperShoes[29]

BuddyClock

The BuddyClock [35] is portraying sleeping statuses within a social network. It is an alarm clock that gives information about friends' sleeping status. There are three different states; awake, snoozing and sleeping. The BuddyClock server application is implemented in Python on a standard desktop PC.

It was tested in a three- to six weeks period with five different social networks. From the evaluation they found that the alarm clock did not affect sleeping patterns, but participants were eager to check others' sleeping statuses. In the event of a status change, participants elicited thoughts about that person. The BuddyClock allowed them to feel more connected to those in their social network.

The participants did not show any concern about sharing their sleeping information as it was only between a small group of close friends. However, they expressed that they would feel different with strangers. Some users would like to share sleeping information without giving the exact time, and they would not like to share this information with strangers.

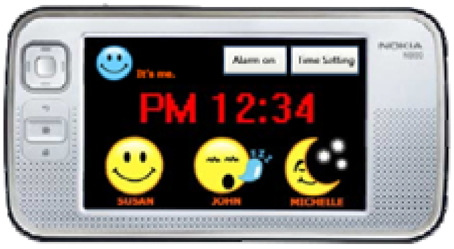


Figure 2.5: BuddyClock[35]

Good Night Lamp

The Good Night Lamp[16] promotes awareness by using light. It consists of one big lamp and multiple small lamps, which is connected to your home wi-fi network. Users keep the big lamps for themselves, and give small lamps to friends and family. When the person with the big lamp turns the light on, the small lamps will be lit as well. That way those with small lamps will know when the person with the big lamp is home. Using the Good Night Lamps is a way of feeling connected to someone, for example when users are in a different timezone. The lamps can be especially useful for people who does not have as advanced technology. People can also use it to signal that they want to talk, by turning their lights on. [46]

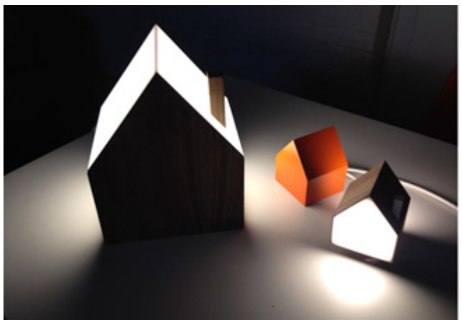


Figure 2.6: Good Night Lamp[16]

Phidget Eyes

Phidget Eyes[23][26] created by Debbie Mazurek, is constructed out of pingpong balls, fake eyelashes, string and glue. The eyes can open and close in any position, controlled by a servo-motor. In addition, pupils can also light up, controlled by two LEDs connected to an interface kit. The phidget eyes indicates colleagues' status when a person is not physically present, but in another room. The eyes are shut when a person is not present, open when the person is present and lit up when the person is interested in communication. The phidget eyes works similar to the good night lamp telling someone that a person is present or not, but it also has a more direct way of telling others that you are interested in communication.

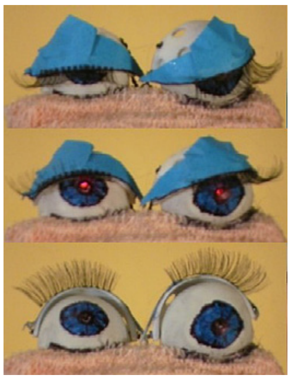


Figure 2.7: Phigdet eyes[26]

Wearable Unit for Emergency Workers (WatchIT)

The wearable unit [13] is formed as a wristband and created for emergency workers on the field. It consists of a LCD display, and a proximity-activated button. It can also make sound and vibrate. The device can display GPS data, environmental information from the rescuers location, and the task that the user is assigned to. The information is sent to a tabletop, via a wireless connection, used to manage and coordinate the different field units.

The proximity-activated button is located on the device armband. By "brushing" the armband the user can notify the coordination unit that a task has been completed. The status bar on the display turns green to confirm that the task- completion message has been sent to the tabletop unit, and the device is ready to receive a new task.

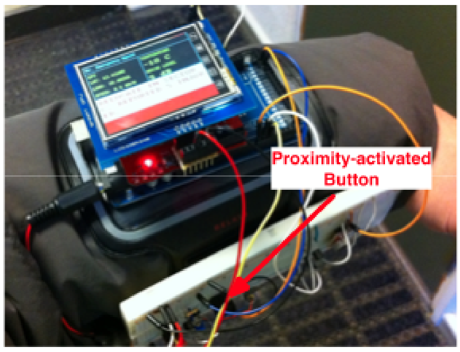


Figure 2.8: Watchit[13]

Hummingbird

The hummingbird[30] developed by Holmquist, Falk and Wigström is an interpersonal awareness device that supports the awareness of presence between individuals in a group. When hummingbirds are in a 100-meter range of each other they produce a sound - they "hum". The hummingbird also has a display showing other hummingbirds that it has detected. Because of the small display, the devices are identified by letters, a, b, c etc., and can display a maximum of 2 devices at the same time. The hummingbird has a carrying case that can be attached to the belt as can be seen in figure 2.9

The hummingbird is tested in two different settings; at a festival and at a conference. In the festival evaluation, people in a group use the hummingbird to feel connected when they are scattered all over the camp and festival site. The hummingbird focuses more on feeling connected than actually meeting face-to-face and the users expressed a sense of awareness when the hummingbird hummed, even if they could not see their friend. Sometimes, however, they felt frustrated when they could not find their friends even though they knew they were close.

The other evaluation was located at a huge conference in Orlando. The users of the hummingbirds were staying at different hotels and used the hummingbirds to check if their colleagues had arrived at the conference. If no one in the group had arrived, the users focused on chatting with other acquaintances or eating good food. When the second person arrived they would know whom to look for. Users described it as comforting to know they were not alone.

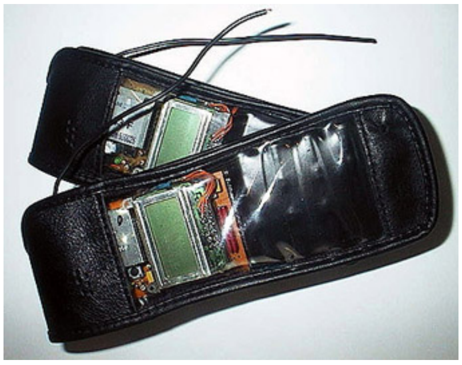


Figure 2.9: Hummingbird[30]

2.4.1 Discussion

We will now discuss similarities and differences between the applications described in the previous section. The applications cover different areas, as mentioned in the introduction. The Hummingbird, the Good Night Lamp, the Phidgets Eyes and the ClockBuddy all promote awareness. The Good Night Lamp and the Phidgets

Eyes promotes awareness of people when they are at certain places, like home or in the office. The Hummingbird promotes awareness between people that are within a hundred meter range of each other, while the ClockBuddy promotes awareness of people's sleeping states. Some applications also share location. When you turn your lamp on you tell your friends you are home, and when your Phidgets Eyes are open, it tells colleagues that you are present. The Wearable Unit communicates its exact location to a tabletop, and the Hummingbird shares location with other Hummingbirds within a range of 100 meters. The SuperShoes is also sharing location, but of places instead of people.

The Phidget Eyes can promote social interaction by letting the eyes light up. This signals that a person is particular interested in communication and that colleagues can come to have a chat. Some of the applications that promote awareness between people can also promote social interaction. The users of ClockBuddy said that seeing someone go to sleep, or staying up later than usual, made them think of that person and made them want to talk with that person. The creator of the Good Night Lamp also wanted the lamp to be a way for people to know when they were available for a chat even though it would just be over the phone.

When it comes to embodiment all the applications act as real world objects; The Good Night Lamp as a lamp, the SuperShoes as shoes, the ClockBuddy as an alarm clock, the Wearable Unit and the iBand as wristbands, the Phidget Eyes as eyes and the Polytag as a dice. The Polytag focuses especially on being a tool that does not take away too much of the user's attention. Under the term embodiment lies also the term wearable. The Hummingbird is wearable and can be kept in a carrying case attached to the belt, although this is not a natural way of carrying something. The Hummingbird is quite big and looks like an old mobile phone, but considering that it is a prototype made in 1999, the physical design is not something that we will focus on. The Wearable Unit and the iBand are easy to carry as they are attached around the wrist, a place that most people are already used to carry their watch or jewelry. Clothes are something that we wear all the time and the most natural way for us to carry something, the SuperShoe takes advantage of this and has integrated the application in shoes. The application therefore becomes a natural part of a user's everyday life.

2.4.2 Lessons learned

Characteristics and functionality from these applications may be relevant for the applications we want our toolkit to support. Table 2.1 sums up lessons learned from each of the applications we have looked at.

Application	**Lessons Learned**
iBand	iBand shows how easy information can be exchanged by shaking hands, this could also be used to make two people become friends in a social network. The iBand also address privacy issues, which is something we should take into account.
PolyTag	PolyTag points out the aspect of physical applications that focuses on not taking away too much focus from what user is actually doing. The ability to present information in the periphery of a person's attention is important.
SuperShoe	From SuperShoe we found how to make an application personal by adding interests. It uses a smartphone as a medium to connect to the Internet. Smartphones have integrated GPSs, which is something to consider for our toolkit. It is an embodied wearable application that shows how haptic feedback, located at different places, can give the user different information according to where it vibrates.
ClockBuddy	From the ClockBuddy we saw that telling friends about an activity can be an important factor for promoting awareness. ClockBuddy also addresses privacy issues, and this should be taken into account in our toolkit as well. An alarm clock is a natural part of a bedroom and shows how an application can be embodied.
Good Night Lamp	The Good Night Lamp shows that it is not necessary with a display to feel aware of someone. With just a light, it gives a signal that someone is at a certain place. By integrating it in a lamp it also shows how an application can be a natural part of the environment.
Phidget Eyes	The Phidget Eyes has a direct way of showing that someone is explicitly interested in communication. To give others an indication of whether or not you are busy, can be an important feature especially at a workplace to indicate to colleagues that you do not want to be disturbed even though you are present.

Table 2.1: Lessons learned: embodied applications 1:2

Application	**Lessons Learned**
Wearable Unit	From this paper we see the importance of using absolute directions to get a precise location when this is necessary. The Wearable Unit is worn as a watch, which is a natural way of wearing an object and something to consider for our toolkit.
Hummingbird	From the Hummingbird we saw that users felt aware of each other when a connection was established, even though users were not always able to find each other. Sometimes however, the users expressed some frustration when they were not able to spot their friend when a connection was established. From this we see a benefit of using direction and distance.

Table 2.2: Lessons learned: embodied applications 2:2

From these applications we have identified some terms and characteristics of location-aware applications. The applications that promote awareness or share location were connected to other devices to be able to communicate with each other. These devices act together as a group. Some of them, like the Good Night Lamp and the BuddyClock, were always ready to communicate, while the Hummingbirds only communicated when they were close to each other. Groups are crucial for providing awareness between the users. Without other devices, there is no one to be aware of.

We saw from the applications that they use display, lights, vibration and sound, as mediums for communicating information. Display was used in the Hummingbird, Wearable Unit and the BuddyClock, and is a medium that can present a lot of information. Lights were used in the Good Night Lamp. The advantage of lights is that they are discrete and can present information in the periphery of a person's attention. The Wearable Unit and the SuperShoe used vibration as notification feedback, and the Hummingbird used sound. These mediums communicate different information.

Location information can be to tell that someone is close, like in the Hummingbird, or tell where a person is located by giving coordinates, like the Wearable Unit. Location-aware applications can have different awareness-ranges. We saw that the Hummingbird had a range of 100 meters, but in another setting it might be a need for a bigger or a smaller range. A goal for location-aware applications is also to promote social interaction, by letting users find each other.

3 | Research Study: End-User Toolkits

Our goal is to create a toolkit aimed at end-users. First, we will describe the theories behind end-user development and toolkits, and relate the terms to our work. Second, we will do a research on existing toolkits that are aimed at end-users.

3.1 End-User Development

Lieberman et al. believes that the goal of HCI is evolving from making easy to use systems and towards making easy to develop systems. User requirements are rapidly changing, diverse and sometimes hard to identify precisely and it is time-consuming and expensive to keep up with them. If users can develop and adapt the systems themselves, the problem of communicating requirements to someone else, can be diminished. This is the goal of end-user development (EUD). It can be defined as *"a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact"* [37]. People are developing applications for themselves, without being trained programmers [36].

End-user development has gradually emerged, following the innovation of personal computers. When the computer became personal it became possible for users to modify settings without affecting others. An infrastructure was provided for end-users to create applications on their own through programming languages like Basic, which used the English language for commands. Then came the graphical user- interfaces along with direct manipulation, which opened up for development tools designed to meet user's needs. Spreadsheets are a widely used example of end-user development, and the first end-user development environment made possible by the innovations. They are considered end-user development environments due to the formulas with input and output variables that the user has to provide. New technologies, such as the web and mobile computing, have taken end-user development to the next level by providing new opportunities for creation and tailoring of applications.[47]

Lieberman et al. identifies two types of end-user activities; "Customization" and "Program Creation and Modification". Customization is about allowing users to choose alternative behavior that are already available in the application. Program creation and modification is about creating from scratch or modifying existing application artifacts, which often requires programming at some level.

The main advantage of end-user development is that the users know their needs better than anyone else. They know about the domain and the context in which the application will be used, and can customize their applications according to this. [47] On the other side, end-user developed applications have high error rates due to lack of quality controls and standards [36]. Users are often not motivated to learn programming languages, formal processes or modeling techniques, since they usually do not develop systems for the long term, but rather short-termed applications with specific purposes. This places certain requirements on the support of EUD; end-users must be provided with appropriate tools, structures, and development processes. The support must be easy to use, quick to learn and easy to integrate into the domain. There is a trade-off between the functionality of a EUD support system and the prerequisite that the end-users must have to use the system. Sometimes it is necessary to hold back on the functionality in order to meet the requirements of an easy to use, and easy to learn, system.[47]

As described, end-user development can have different meanings. When we talk about end-user development in our thesis, we mean end-user customization. It is about letting users choose alternative behaviors that are already available in the toolkit, and does not require any programming or prior knowledge. This is an important part of our work, as we want to create a toolkit that can provide this for the end-user. The customization of an application will also concern physical tinkering, but like software-customization, it should not require any programming knowledge. Nor shall the hardware customization require any complex wiring or electronics knowledge.

3.2 Toolkits

A toolkit can be used as an end-user development support system. Dictionaries define it, as the name suggests, as a set of tools that can be put together for a particular purpose. They can be designed to meet a wide range of potential applications and situations, and provide reusable components and behavior designed to support this range of applications. The components that are provided arise from common patterns of software structure that occurs across applications in a particular domain. Rather than creating the components from scratch, the focus is on assembling and configuring them [21]. The design of traditional software toolkits aim to ease the implementation [21] and help the programmers to rapidly generate and test new ideas, while at the same time being less costly [24]. By removing low-level implementation burdens and supplying appropriate building blocks, toolkits also promote creative design [25].

Dourish and Edwards [21] point out one of the main challenges when developing toolkits; The designer of a software toolkit does not develop applications, but software to be used for developing applications. Users of a toolkit are usually programmers themselves. Further, these programmers develop applications for yet another type of users, namely the end-users, as illustrated in figure 3.1. This puts high demands on the toolkit designer who must, not just anticipate the needs of the application designers and the sorts of application they will create using the toolkit, but also the behavior and requirements of the application users.

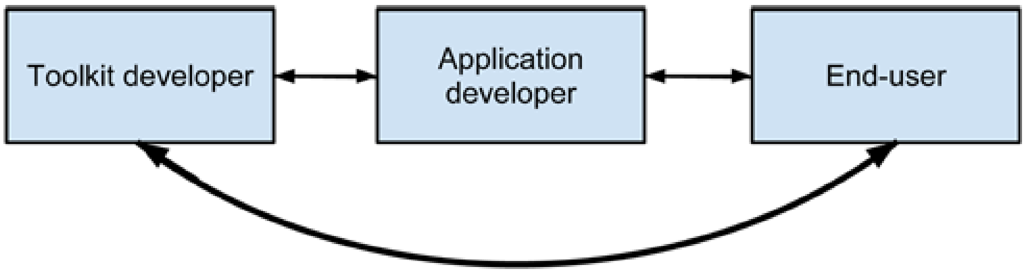


Figure 3.1: Toolkit for developers [21]

For the toolkit developer, anticipating the needs of application developers means anticipating the requirements of as-yet-unformulated applications and the needs of as-yet-unknown users [21]. Designing toolkits for the end-users themselves can reduce this problem. That way need-related tasks can be outsourced to the users, and user information can be used where it is already located, instead of going through the costly process of transferring it [51]. The user of the toolkit will then be the user of the application as well, as illustrated in figure 3.2. Designing for end-users adds complexity as mentioned in section 3.1, as the toolkit places demands for high level abstraction and the usability of the toolkit. End-users have various technical experiences and this needs to be considered.

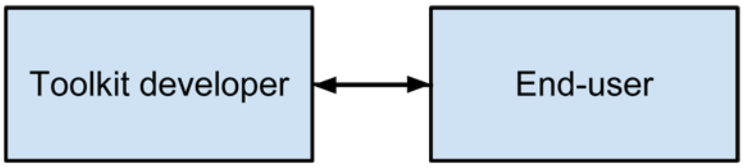


Figure 3.2: Toolkit for end-users

.

A lot of toolkits that exist today are toolkits aimed at developers. Our work, however, focus on a physical toolkit aimed at end-users. The end-users create their own applications, and are also users of these applications. The toolkit must be easy to use for the end-user and still offer enough functionality and options to customize different applications. It should meet the requirements of end-user development support systems; prevent errors, be easy to use and require little prerequisites from users.

3.3 Related Work

We have investigated a set of physical toolkits aimed at end-users. Our toolkit should be used for customization and we will look at how other toolkits have solved this. We want to find an easy and straightforward way to customize applications that is logical and intuitive for the users. As we have little experience in this area, this study can give us an indication of what is possible to do. This is something we will investigate and take into account when choosing technologies and development platforms for aWearable.

Factors that we will look at are:

- how to customize it
- which technologies are used
- metaphors used

As many of the physical toolkits today require some sort of programming or wiring there were not many toolkits to be found that are qualified for novice end-users. In addition, we have not been able to find toolkits that supports group creation or location- awareness.

NinjaBlocks

NinjaBlocks [9] are small cloud enabled computers that can be connected to a variety of sensors. The sensors send information about the environment to the NinjaBlock, and it can affect its surroundings by controlling light, power sockets and other actuators without writing a line of code. The users of NinjaBlocks can add rules through an interface. It can, for example, be used to turn on and off lights, provide alerts when the washer is done, upload a picture to dropbox when someone is in the users room, and much more. Like this, users can create their own application without having to worry about embedded programming, electronics, and networking protocols.

The NinjaBlocks contain an Arduino compatible microcontroller that lets you program using the Arduino IDE. It is also open hardware that lets you access the

Arduino inside. The Arduino talks to the sensors, actuators and the wireless modules. They use Ubuntu Oneiric 11.10 as the operating system, and most of the software is written in JavaScript.

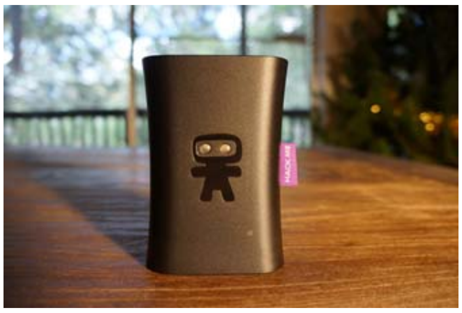


Figure 3.3: Ninja Blocks [9]

reaDIYmate

The reaDIYmate [38] as the name indicates, is a do it yourself toolkit. The reaDIYmate can run a program, fetch data on the Internet and obey their masters. The user chooses what the reaDIYmate should do through a simple web interface or with an iPhone application. You can connect it to your "digital life", and make it tweet, post a status, dance when you receive an email and much more. By ticking select buttons the user can choose a social platform and when the reaDIYmate should do something. E.g. tick the checkbox for 'Facebook' and 'Likes' and the reaDIYmate will move and make sound when you receive a like on facebook.

The reaDIY is hacker friendly and can be opened to access the hardware, which is a fully open-source board featuring solderless connectors. It is Arduino compatible and can be programmed using Arduino software. It is also possible to use the SEEED Studio Grove series for easy plugging of actuators and sensors. The user can choose among 19 different shapes and designs of the reaDIYmate as shown in the picture.

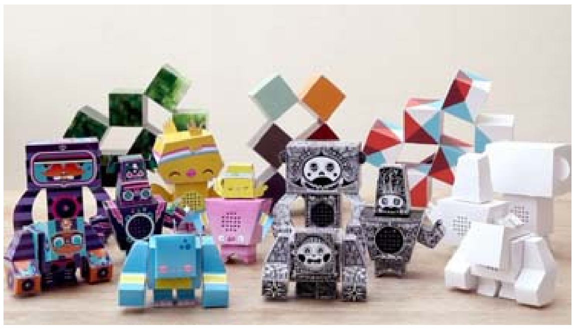


Figure 3.4: ReaDIYmates [38]

littleBits

littleBits [7] is an opensource library of electronic modules that snap together with tiny magnets. This toolkit has no web interface. Each bit consists of tiny circuit boards with specific functions such as lights, sounds, motors and sensors. The modules can be snapped together to make larger circuits. littleBits is meant for early stage design for experts, designers and students. It does not require any programming, soldering or wiring. littleBits compare themselves to LEGO, which allows you to create complex structures with very little engineering knowledge.

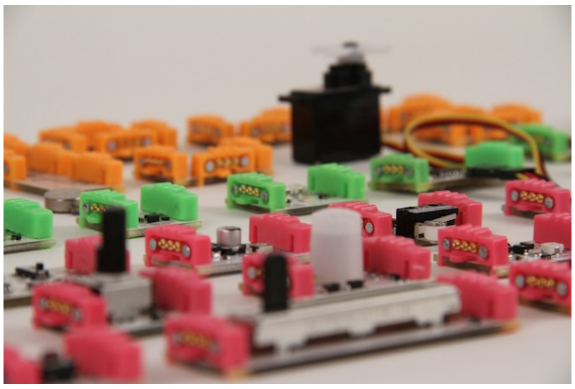


Figure 3.5: littleBits [7]

3.3.1 Discussion

We will compare the different approaches of the toolkits. First, the NinjaBlocks and the reaDIYmate use a web interface to customize the application, while the littleBits does not. The advantages of using a web interface is that it might be easier for a lot of users to follow some instructions when creating applications and make it do what they want it to do. It is probably also easier for the user to see all the possibilities the toolkit have and it might be clearer to them what the application will do. In addition, it is easier to make more options on a web interface, like the type of information you want, privacy settings and connection with friends. It is, however, more to attend to with a web interface in addition to the physical device, and people that prefer to try and fail, might find it funnier to create applications without a web interface. When the user knows what to do and how to make applications, it will probably be faster for them to create applications without a web interface. Another disadvantage with a web interface is that you are dependent on being in possession of a computer.

We have looked at how other toolkits solve customizing of physical applications without doing any programming. The toolkits with a web interface have made simple interfaces that require little from the user. To customize NinjaBlocks applications the users choose what the application should do by clicking buttons. The buttons have a description of what it will do. This is called 'adding rules' to the NinjaBlock.

The reaDIYmate web interface consists of checkboxes that the users can tick to choose how they want their reaDIYmate to behave. The littleBits use a LEGO metaphor where users snap bits together and it will behave differently depending on how the bits are attached.

The NinjaBlocks consists of an Arduino and the software is mostly written in JavaScript. The reaDIYmate board is Arduino compatible and uses the SEEED Studio Grove series. littleBits consists of electronic components pre-assembled in tiny circuit boards. All of these toolkits are open source and possible to hack.

3.3.2 Lessons Learned

We have listed lessons learned from the research in table 3.1

Toolkit	**Lessons Learned**
NinjaBlocks	It is important with a simple web interface so the users understand how to customize the toolkit. From the NinjaBlock web interface we saw how to use buttons with different behavior to choose what the application should do.
reaDIYmate	In the reaDIYmate web interface we saw how we can use checkboxes to decide what the application should do.
littleBits	It is important that parts are easy to put together like the littleBits that use magnets. Color coding of pieces can also help to understand which parts go where.

Table 3.1: Lessons learned: End-user toolkits

4 | Problem Refinement

The goal of the problem refinement was to end up with a requirements specification for the aWearable toolkit. We prepared a set of scenarios based on the research studies, and defined a design space from these scenarios. We continued with high level requirements that were further refined into comprehensible functional and non-functional requirements. In this chapter we will be presenting the scenarios, the design space and the requirements for the aWearable toolkit.

4.1 Problem Summary

The problem of this thesis is to support end-user customization of embodied location-aware- applications. As an answer to this question, we will develop a toolkit prototype. From the research study on embodied location-awareness, we will specify characteristics and functionality the aWearable toolkit should include, and what kind of feedback users need to receive to be made aware of someone's location. From the research study on end-user toolkits, we will extract ideas to help us adapt the toolkit to novice end-users. We will evaluate the prototype with users to determine how it supports end-user customization. We will look at how users are able to create and use location-awareness application by using the aWearable toolkit.

4.2 Scenarios

Based on the theoretical background and research on physical applications in chapter 2, we have developed four scenarios. These scenarios are meant to serve as examples of location- aware applications that can be developed using the toolkit and in what settings they can be used. This will help us identify the design space and requirements. The goal here is to show that the toolkit can be generalized to cover multiple scenarios. We have looked at different settings where end-users have different information needs, and where they can use the toolkit to design a customized application that covers their requirements.

As a basis for what we think the toolkit should provide, we have chosen to use the four actuators; display, light, vibration and sound, as feedback options in the scenarios. Like we discussed in related work, section 2.4, these actuators were used in many of the physical applications, which is why we have chosen to use them as well. In the scenarios we assume that all people possess and use the aWearable toolkit.

4.2.1 Scenario 01: At the Movies

A group of friends are planning to go to the movies. They are all living at different locations and are arriving from different areas with different transportation. When they arrive at the cinema there are a lot of people and it is difficult to spot each other. One of the friends wants to know if he is the first one to arrive, or if some of his friends are already waiting for him so he should look for them. He wants an application that can give him this information in a discrete way.

He uses the toolkit and starts by making a group of all the friends he is going to the cinema with. To know if other friends have arrived at the cinema he wants a notification telling him when a friend is in his proximity. The proximity range must be quite small as he only wants to get notified when a friend already there. He wants the notification to be discrete and to keep the application in is his pocket, and decides that vibration is the most appropriate. Now he will get notified when one of his friends has arrived and he can try to look for them. He must also think about what his friends can see of him, and as he is a private guy, he will only let his friends see his approximate location. Table 4.1 shows the desired application.

Characteristics	**Details**
Group	Create a group
Vibration	Vibrate when a group-member is in a certain proximity
Proximity- range	Small
Share with others	Approximate location-information

Table 4.1: Scenario 01: Application Characteristics

This scenario was generated from the conference scenario where the hummingbird was used to check if others had arrived. The application in our scenario is a lot simpler as the only information the users get is a haptic feedback, but this notification tells them exactly the same, - that someone else has arrived and that they should look for this person.

4.2.2 Scenario 02: Emergency work

A group of emergency workers are out on a mission looking for a missing person. The teammates have to split up in order to cover as much area as possible. The search area is in a rough terrain and the teammates cannot always see each other. The emergency workers have a need to be aware of each other and to know where the nearest co-worker is located at all times and decides to create an application to help them with this.

They start by creating a group of all the emergency workers. As it is important to know the exact position of co-workers, they decide to use a display that can provide this information. The rescuer must keep their eyes on the surroundings and cannot continually look at the display, they therefore add vibration, sound and lights, as notification when a co-worker is in their proximity, telling them that location-information now is visible at the display. Also to lower the focus on the display, the light will blink faster when a co- worker is getting closer, as lights can be in the periphery of the rescuers attention. Distance between the emergency workers can be big and they therefore use a big proximity range. To get all location information about each other, they must share all location- information with each other. The characteristics of the application is listed in table 4.2.

Characteristics	**Details**
Group	Create a group
Display	Show exact location of group-members when they are in a certain proximity
Vibration	Vibrate when a group-member is in a certain proximity
Sound	Make a sound when a group-member is in a certain proximity
Lights	Blink faster when a group-member is getting closer
Proximity- range	Big
Share with others	Exact location-information

Table 4.2: Scenario 02: Application Characteristics

This emergency worker scenario is based on the Wearable Unit. Our scenario however concern awareness between rescuers out in the field and does not provide the same functionality as the Wearable Unit. Like the Wearable Unit we want users to be able to attach the application around the wrist. This is especially suitable in emergency situations as the emergency workers need free hands, and to focus on their surroundings rather than on a device.

4.2.3 Scenario 03: Roommates

Six students live together in a household, they often stay in their room and it is not always easy to tell if anyone is home or not. One of the residents does not like to be home alone in the evenings and she would like to know when the others are home. She wants to create a simple application that can tell her exactly that.

She starts by creating a group of the roommates and decides to use the light as a notification when someone is in her proximity because a light is discrete and constant. This way she can glance over at the application whenever she wants without it disturbing her from what she is doing. The proximity range must be set to surround the house so she will only be notified by when group-members are inside the house. She likes that others can see where she is as it makes her feel safe, she therefore let others see all location-information about her. See table 4.3.

Characteristics	**Details**
Group	Create a group
Lights	Light up when a group-member is getting closer
Proximity- range	Small
Share with others	Exact location-information

Table 4.3: Scenario 03: Application Characteristics

This scenario has resemblance from the Phidgets Eyes and the Good Night Lamp. Although the application cannot promote awareness of people at faraway places, the concept is the same. It is about knowing when someone is at a specific location. This application could also be used at a workplace. The discrete lights will not distract the user, or anyone else, from what they are already doing. They are also easy to check.

4.2.4 Scenario 04: Festival

A group of friends are going to the Coachella Music Festival. Their different taste in music causes problems when they discuss which concerts to see. They decide to split up, but realize that they will have a problem finding each other after the concerts with so many people at one place. They know that their phone- batteries will not last and that the line at the charging station will be long. It is also expensive to call each other when they are abroad.

The friends use the toolkit and start by creating a group. They want the application to be helpful for finding each other and need a display that can show them the way. Vibration will work as notification when a group-member is in range and will tell them to look at the display. They need a big proximity range, but not so big that they receive information about each other while they are at different concerts. To make it easy to find each other they share their location with each other. The characteristics of the application can be seen in table 4.4.

Characteristics	**Details**
Group	Create a group
Display	Show location of group-members when they are in a certain proximity
Vibrator	Vibrate when a group-member is in a certain proximity
Proximity- range	Big
Share with others	Approximate location-information

Table 4.4: Scenario 04: Application Characteristics

This scenario is borrowed from the festival scenario where the hummingbirds were used. In the Hummingbird evaluation, users expressed frustration when they were unable to find the other users. Our application should provide more information and make it easier for users to find each other. We will also use haptic feedback instead of sound, as sound can be hard to hear in a noisy environment.

4.2.5 Discussion

The scenarios have some similarities and differences that we will discuss. First, we recognize that group creation is required in all scenarios, as a necessary step to know who to be aware of. This was also a characteristic found in related work on location-aware- applications in section 2.4.

The different needs in the scenarios made the importance of awareness and sharing of location different as well. In scenario 01 the user receives a haptic feedback when a group-member is in his proximity, but does not get any more location-information beyond this. The proximity range is small, so when the user receives a notification it tells him that a friend has arrived and he should look for him. In scenario 02, awareness and sharing of location are more critical. Here they set the awareness range higher than in scenario 01 as the rescuers cover larger areas and the distance between them is greater. It is important for the rescuers to feel aware of each other and to know where the closest co-worker is located in case something should happen.

When they receive a notification they know that a co-worker is within range and they can look at the display for more information about the co-worker's location, or glance at the lights to see if the co- worker is getting closer or further away. The emergency workers share all location-information with each other as opposed to in scenario 01, where he only wanted to share his approximate location with group-members.

In scenario 03, the light on the application turns on when a roommate is home. This makes the roommates feel aware of each other, but it does not necessary mean that they go look for each other to socially interact. In the festival scenario, scenario 04, the group of friends will be aware of each other when they are in each other's range. However, they will mostly use the applications for finding each other. They will receive a haptic feedback that tells them to look at the display. The display will give them information about where to go to find the group-member, but it is not necessary with as much information as in the emergency worker scenario.

Characteristics of the applications in the scenarios, are characteristics that we found in the research studies. The toolkit should provide group creation, the possibility to choose actuators and what the actuators should do or show of location-information. It should also be possible to change the proximity-range and manage privacy settings where you decide what location- information to share.

One of the challenges with the physical applications is to make it embodied and wearable. There are different reasons for using embodied interaction in the scenarios. In scenario 01 and 04, the users do not want to constantly hold and look at a device to keep track of where their friends are. The same applies in scenario 02; they need to stay focused on their surroundings. The emergency workers may have a need for keeping their hands free in situations where a missing person is found. The condition of the person may be bad and require immediate help from the rescuer. In scenario 3 it is not crucial to carry the application on the body like in the other scenarios. It can be kept on your desk, or hidden away in your backpack. The notification should not distract anyone, but rather blend into the environment and let the user glance at it when they see it light up.

4.3 Design Space

The design space of our prototype has been identified from the scenarios and concerns three closely related aspects:

- Groups
- Awareness
- Share location

All of these three terms are dependent of each other, and somewhat overlap each other.

Groups
Groups are necessary for promoting awareness and sharing of location. We saw that the location- aware applications in the related work section 2.4, were connected and therefore acted as a group. Groups and communication between devices have not been our focus in the thesis and we will therefore not go deep into the group dynamics. However, we will explain in short the theory of how we see the groups working. We have used the terms permanent and transient groups to explain the 'situation' of a group at a specific moment. When a user has made a group, this group is permanent when the application is in use. When two users of the permanent group are in each other's proximity a sub-group with these two users will be created. It is at that moment they will share location information with each other. These sub-groups are called transient groups. This is illustrated in figure 4.1. As users are on the move the transient group will change, users will enter and leave the proximity circle.

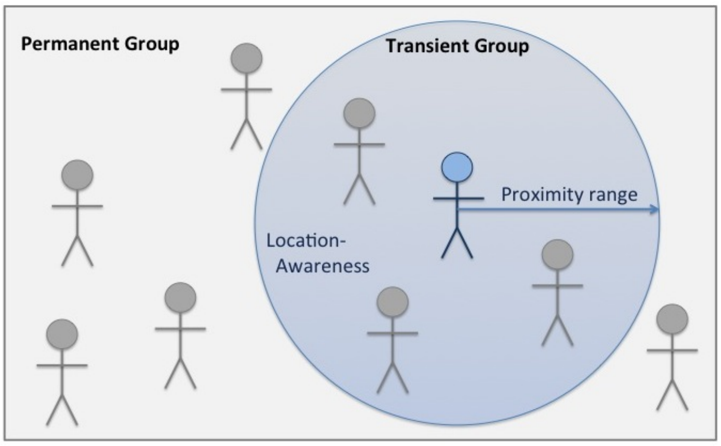


Figure 4.1: Group structure

Awareness
We saw from the scenarios that the applications promote awareness of other group-members. As previously described, awareness is as an understanding of the activities of others, which provides a context for your own activity [20]. The activity in our applications is about people's location relative to each other. With a narrower term we can refer to it as location-awareness. Awareness is dependent of a group structure for the application to know who to be aware of, but group-members are not aware of each other until they are in each other's proximity. When group- members are aware of each other they share some location- information with each other.

Share Location
As the toolkit is used to create location-aware applications, we must look closer at how to share location, and what location-information to share. It is when users are in each other's proximity that the sharing of location happens. In the scenarios we described different degrees of sharing location. Some shared the exact location, while others only shared approximate location. We saw that there is actually two aspects to consider when talking about sharing location, the first one is what location-information the users want to receive from others, while the second is what location-information about themselves the users want to share with others.

We will continue using the four actuators used in the scenarios. Users should get a set of options to decide what location-information these actuators should present, and how the actuators should behave. The location-information options are inspired from the research done on physical applications, as well as different needs in the scenarios. In the emergency scenario and the festival scenario the users want to know where group-members are located and how to find them. It should therefore be an option to present direction, distance and coordinates on the display. When it comes to proximity we saw that there were different needs for range-sizes. The emergency workers wanted it be to big, while the friends going to the movies wanted it to be small. Users should therefore get to decide how big this range should be.

The light, vibration and sound were used as notifications in the scenarios to notify them when group-members were in their range. In the emergency setting the lights also blinked faster when a group-member was getting closer and this should be an option for vibration and sound as well.

Some papers from Related Work, like iBand and BuddyClock, emphasize the importance of privacy. As we saw from the scenarios users could decide what location-information to share with group-members. The options users receive from others should be the same options users can choose not to share with others. There should however be a minimum requirement for what people must share so that users cannot use aWearable to 'spy' on others.

Table 4.5 sums up the characteristics the aWearable toolkit should include.

Groups:	• permanent group • transient group
Location information:	• distance • direction • coordinates
Components:	• display • light • vibration • sound
Proximity range:	• small • medium • large
Privacy settings:	• in-range (always allowed) • distance • direction • coordinates

Table 4.5: Design Space

4.4 High Level Requirements

We point out three aspects as the foundation for the aWearable toolkit; manage groups, customize an application and use the application. The aWearable toolkit will consist of two parts that the users must relate to, a web interface and a physical device. The web interface is where users create groups and choose what the application should do, while the physical device, consisting of an Arduino, is the physical application that users will carry with them. We will also discuss why we have chosen to use a web application, and why we have chosen to use Arduino hardware.

4.4.1 Manage groups

The managing of groups is about whom users chose to receive location from and share their location with. We will call these two groups for the 'following-group' and the 'followers-group', respectively. For the prototype we have decided to use Facebook to establish connections between friends. Users must be able to 'manage' their friendships by adding and deleting friends from both the group of followers and the group that they are following. The managing of groups also concerns how much information they decide to share with their following-group.

4.4.2 Customize an application

The tailoring of the application concerns how users want their applications to perform. They should be able to choose what components they want and how they want them to behave. In addition, the users must be able to connect the components physically to the aWearable hardware device. The customization of the application will happen in a web application.

Why a web application?
In the research done on toolkits in section 3.3, we discussed some advantages and disadvantages of using a web interface to customize the application versus customizing it in a more embodied manner. One of our goals is that the toolkit should be as simple to use as possible for both technical-experienced user and non-technical-experienced users, and we feel that the best way to reach this is to start with a web interface. The web interface can have instructions with pictures and animations to help users understand what they should do. Once users get familiar with creating applications we can move parts of the customization out of the web interface and into the physical world. This would be work for further development.

There are different reasons for why we have chosen a web application over a smartphone application. First of all there is the issue of a big screen versus a small screen. A computer screen provides more space that gives a better overview of the application creation process for the users. To be able to show more steps at a time can help the user to see the connection between the different steps. A smaller screen would limit how much we can show at the same time, and would make the user scroll up and down or swipe back and forth to see what have been done. We also want to visualize the Arduino board to help users to physically attach the different toolkit actuators. This can be hard to do on a smartphone screen. We do see some drawbacks for using a web application as well, like the fact that users can not customize applications or groups when they are on the move, so this should be considered in future work.

Last is concerning us as developers. We have most experience in making web applications and program languages as HTML5 and JavaScript. The advantages of using HTML5 and JavaScript are also that it works on all platforms and can easily be made responsive in the future. We will evaluate different development platforms in chapter 5.

4.4.3 Use the application

When the customization of the location-aware application is done, the actual location sharing is happening when a user puts the aWearable device to use. Location data is shared when connected users are in each other's proximity and this information is communicated through the chosen components. We have chosen to use Arduino as the hardware.

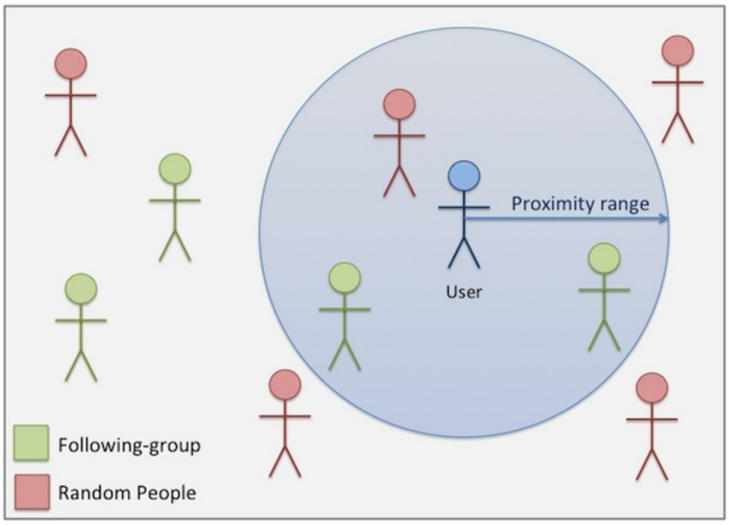


Figure 4.2: Awareness in group

Why Arduino?
Arduino [1] is an open source electronics prototyping platform based on flexible, easy-to-use hardware and software. It can sense the environment by the input from different sensors and can affect its surrounding with actuators and motors.

We have chosen to use Arduino because it is an inexpensive cross platform. From the research done on toolkits for end-users we saw that the NinjaBlocks use Arduino and reaDIYmate is Arduino compatible and it therefore feels like a safe choice. In addition, our co-supervisor, Simone Mora, has experience with Arduino and will be able to assist us if needed. Arduino also comes in different sizes and can be as small that they can be worn in clothes, this is an important factor considering further development, where we want the toolkit to be small and wearable.

4.5 Functional Requirements

In this section we have refined the high level requirements into more comprehensible, lower level requirements. The hardware requirements are concerned with the aWearable device while the software requirements concern the aWearable web application.

Software Requirements

Id	**Requirement**
SFR01	Sign up using Facebook
SFR02	Retrieve list of Facebook-friends
SFR03	Add friends to follow
SFR04	Delete friends from 'following-group'
SFR05	Manage follow-request
SFR06	Delete friends from 'followers-group'
SFR07	Set privacy
SFR08	Choose components
SFR09	Choose component behavior
SFR10	Select range
SFR11	Connect to Arduino
SFR12	Upload application to Arduino
SFR13	Save application
SFR14	Edit application
SFR15	Delete application

Table 4.6: Software: Functional Requirements

SFR01: This requirement states that users should be able to use their Facebook accounts to sign into the aWearable web application. We have chosen to use Facebook since most people today have Facebook profiles and are familiar with the social network. Facebook is sufficient for our prototype, but if we were to deploy the toolkit, it would have been unacceptable to demand sign-up with Facebook, and other possibilities should have been implemented as well.

SFR02: Users should get access to their Facebook-friends through the aWearable web application, so that they can manage whom they share data with by adding them to their group. The same goes here as for requirement SFR01, that Facebook is suitable for our prototyping purposes, but not for an actual application.

SFR03: From the list of friends, retrieved from Facebook, you should be able to choose one or more friends and add them to the group of users that you want to receive location information from, the 'following-group'.

SFR04: Once friends are added to a user's 'following-group', the user should be free to remove them again at any time. Once removed, the user should no longer receive location information from the deleted friends.

SFR05: To ensure some privacy, users should get notified before they are added to someone else's group. If you decide to accept the request, you will now share your location information with the user that sent the request. You will be added to this user's 'following-group', and the user will be added to your 'followers-group'. If the request is declined, the user who sent it will not get any location information from you and you will not be added to that user's group.

SFR06: If you choose to accept the request, but wishes to remove it later, this should also be possible. By removing a friend from the 'followers-group' you no longer share location information with this friend and you are removed from this friends 'following-group'.

SFR07: Users should be able to determine how much location information that is shared with their followers. It should be possible to choose if you want to share exact location or only your relative location.

SFR08: In the aWearable web application, users should be able to choose among the components; display, lights, sound and vibration. It should be possible to choose one or more components.

SFR09: The users should be able to customize the behavior of the chosen components in terms of what kind of information that they want the components to present. For example, if you have chosen the light, you can choose to have it light-up when friends are in your proximity or you can have it blink faster as friends are getting closer to you.

SFR10: The users should be able to choose within what range they want to receive location information. In some situation, like in the festival scenario, it can be desirable to have a large range. In the movie scenario, on the other hand, a smaller range will be sufficient.

SFR11: It must be possible to establish a connection between the aWearable device and the user interface through the web application, so that the application can be sent to the Arduino board.

SFR12: The application that is created in the user interface must be uploaded to the Arduino board so that it can become a physical application.

SFR13: Users should be able to save applications for later use. This can, for example, be helpful if a user has started to create an application but does not have time to finish it.

SFR14: Users should be able to go back and edit the applications they have created. This can, for example, be helpful if a user wishes to change the behavior of a component.

SFR15: Users should be able to delete applications they no longer need or want.

Hardware Requirements

Id	**Requirement**
HFR01	Turn the device ON and OFF
HFR02	Connect to the computer
HFR03	Connect components
HFR04	Get GPS data
HFR05	Compute location
HFR06	Output feedback when followed devices are in-range
HFR07	Output feedback on chosen components
HFR08	Output feedback with chosen behavior

Table 4.7: Hardware: Functional Requirements

HFR01: It should be possible to turn the aWearable device ON and OFF. This can be necessary if a user do not want to share location information for a while. The device can then be turned OFF, and later ON again when the user is ready to share.

HFR02: This requirement is similar to software requirement SFR11. It must be physically possible to connect the device to the computer as well as establishing a connection with the software user interface.

HFR03: Users must be able to physically connect the components to the aWearable device. There must be a connection between the aWearable web application and the aWearable device that helps the users to connect the physical components correctly. The physical components that the users connect should correspond to the components that the users selected in the web application.

HFR04: The aWearable device must be able to get GPS data from a GPS module, in order to determine the location of the device.

HFR05: The aWearable device must be able to compute the devices location in relation to each other, by using information based on the data that is received from the GPS module, and the location that is received from devices within range.

HFR06: When friends a user is following are within the range that was set in the software interface, the user should receive feedback from the aWearable device.

HFR07: The presentation of location information from devices within range should be presented through the components that were chosen in the software interface. For example, if the user chose light, the light should light-up when other devices are in proximity.

HFR08: The presentation of location information from devices within range should be presented through the chosen components according to the behavior that was chosen in the software user interface. For example, if the user chose blinking light, the light should blink when other users are getting closer.

4.6 Non-functional Requirements

The aWearable toolkit shall support end-user customization, which in turn places certain requirements on the usability of the toolkit. We have formed a set of non-functional requirements for the aWearable toolkit that takes this into account. We have chosen to disregard non-functional requirements concerning security, compatibility, maintainability etc. since we will only be developing a prototype as a proof of concept.

Id	**Requirement**
NFR01	Users should be prevented from committing errors
NFR02	The system should be easy to use and learn
NFR03	Users should be able to use the system without having any specific prior knowledge
NFR04	The language used in the toolkit should be adapted to users without technical experience
NFR05	The aWearable device should not look approachable and not intimidating

Table 4.8: Non-functional Requirements

NFR01: As described in chapter 3, errors often occur in end-user developed applications as most users lack development competences. Since we are developing for customization and tailoring, rather than for end-user programming, error rates are not as high. Still, there is a need for error prevention that helps users to create applications that will work as they are supposed to. For example, the users should be prevented from uploading an unfinished application to the aWearable device.

NFR02: To motivate end-users to use the aWearable toolkit it is important not to place unreasonable demands on them. They are often not motivated to learn a lot about a system prior to using it. By developing a toolkit that can be used and learned efficiently, without requiring prerequisites, we hope to attract end-users. There is a trade-off between the amount of functionality provided by the toolkit and the prerequisites it required. This requirement is about finding a balance between the functionality and the usability.

NFR03: This non-functional requirement builds upon NFR02. We want users to be able to understand the system regardless of domain knowledge and technical knowledge. Therefore, the prerequisites required from the users should be limited.

NFR04: For the toolkit to be easy to use and learn by users without domain knowledge and technical knowledge, the language used in the toolkit should be adapted accordingly. It should be grounded in everyday speech and technical terms should be avoided.

NFR05: An Arduino board can consist of different modules, resistors and wires and may look intimidating to non-technical end-users. It is important that the aWearable device does not scare off possible users. We need to make it more desirable and less intimidating.

5 | Software Development Platforms

There are multiple software interfaces that supports Arduino development already on the market, but most of them are currently in an alpha or beta state, which suggests a new and advancing interest in providing tools for tangible interaction development. In this chapter we have evaluated a set of development platforms. We have selected platforms that support connection with an Arduino board, and that have web interfaces. We wanted to find a good way to connect the software interface to the Arduino board, and also to get ideas for user interface metaphors. We evaluated the platforms both as frameworks that could be used to develop the aWearable toolkit, and as toolkits for creating applications.

5.1 Evaluation

We have set some selection criteria for evaluating if the development platforms are suited for us to use.

- It should facilitate the making of a web interface, without restrictions on the layout.
- It should be possible to program the interface with HTML5 and JavaScript.
- It should provide an easy connection with Arduino.
- The platform should support handling of GPS data.
- The platform should support group creation.

These criteria are based on the requirements in Chapter 4. We will create a web interface where we focus on usability. As we have experience with HTML and JavaScript, we wish to use these technologies to create the web application. The hardware will consist of an Arduino board that should receive data from the web interface. To support location-awareness we are depending on group creation and a GPS for location- sharing purposes.

Sen.se
Sen.se [10] is an entirely web-based development platform for displaying and collecting data from physical devices. The device can be an Arduino board, a coffee maker, a TV or basically anything that can be connected through Ethernet, WiFi, ZigBee or USB. A connected device can both send and receive data from the platform. Events are sent to the platform when something happens on the device. Sen.se stores these events in a feed that can be used by other devices or applications to trigger actions. All the applications are displayed on the sense-board. It is possible to send feeds to the device, but current applications are primarily created based on sensor input-data with focus on visualization and computations.

When making Arduino applications, the users are provided with a set of steps to connect the Arduino to sen.se feeds. They must give the board a name, choose a protocol for interaction with sen.se (http is the only one available at the time), set up the feed, including if it should be an input or output feed and what kind of data it will have. Last the user must choose a pin and select between analog- or digital mode. Sen.se then provides a customized Arduino sketch that the user must upload to the board through the Arduino software.

Sen.se is a high-level development platform, which also makes it somewhat limited. The layout of sen.se applications is limited by the sense-board. Sending data from a customizable HTML5 interface to the Arduino board is not functionality that is provided by the sen.se platform out of the box. Neither is handling GPS sensor data. When connecting the Arduino to sen.se, you can only choose between analog and digital data flow, while the GPS sensor is serial. It can be difficult to get a framework to perform in a way that it is not meant to. There is also no functionality supporting the creation of group structures that is necessary in location awareness.

One of our main goals is for aWearable to require no prior knowledge from the users. The users of sen.se do not need to have application programming knowledge. On the other hand, they do need Arduino knowledge. The platform does not give any guidelines or support when it comes to connecting the Arduino modules and sensors to the Arduino board, which makes it unsuitable for non-technical end-users. In addition, sen.se is more focused on presenting data from sensors than writing data to the physical devices, which is what we need aWearable to do.

We found sen.se not fitting as a development platform for building the toolkit, based on it being too high level and not particularly customizable. It does not provide GPS data out of the box, and is not suited for developing location aware applications. The step-wise Arduino connection could be made even more user friendly, by hiding all complexity. However, we can learn from the step-wise connection with Arduino, as it is easy to follow for users. In the aWearable toolkit, we want the connection to be abstracted away, and to use steps in the application creation part of the toolkit.

Noduino

Noduino [39] is a simple and flexible JavaScript and Node.js framework for accessing basic Arduino controls from Web Applications, using HTML5, Socket.IO and Node.js. The Noduino project is in an early development state. It was founded by Sebastian Müller in 2012 as a proof of concept for node.js to control external components through a dynamic web interface by using HTML5 WebSockets.

Figure 5.1: Noduino

Node.js is a server-side JavaScript environment, which gives great performance to many Internet applications. The main advantage is that it allows you to create the entire web application in JavaScript, both the client-side and the server-side [33]. Node.js is designed to support HTML5 WebSocket Protocols. WebSockets defines a two-way communication between client and server and simplifies much of the complexity around bi-directional web communication [15]. In Noduino a WebSocket server is created on top of node.js. Further, Socket.IO is used. Socket.IO blurs the differences between different transport mechanisms and provides carefree real-time in JavaScript. It supports features like heartbeat, timeouts and disconnection that are not provided by the WebSockets API out of the box [11]. In turn JavaScript can be used to send messages to an Arduino board from an HTML5 interface.

Noduino is a big and complex low-level framework, which makes it hard to get to know, but it puts few restrictions on what is possible. The HTML5 interface makes it easy to customize the interface, and JavaScript can be used to send messages to the Arduino board. It is possible to use GPS signals and create groups, but there is no ready-made functionality for it. The connection with arduino is abstracted away, which makes it easy for both developers and end-users. Noduino is however not suited for end-users as programming is a must.

LabView Interface for Arduino (LIFA)

The labView [6] is written and distributed by national instruments. The API lets you read back analog inputs, control digital IO lines and use several other features of the Arduino hardware. It lets developers get data from the Arduino micro-controller and process it in the LabView Graphical Programming environment. The toolkit has 850 built-in libraries with signal processing, math and analysis libraries, and also connectivity libraries which includes interface to web services, databases, executables and more.

The LabView is a drag and drop based toolkit, where functions are found in the palette and dropped onto the block diagram. The block diagram is a picture of the Arduino, and it can show the values that are passed from one function to another. By double clicking component within the toolkit it is possible to see how it works, and modify the Arduino Sketch. This makes it easy to use for users without a lot of experience and also lets more experienced users modify the code. The LabView communicates via USB, but can be wirelessly tethered by using Xbee or BlueSMiRF. Data from the sensor can be visualized on the computer by using LabView Front Panels. It gives you full control over what you make visible to the user and which parts of your application you keep to the block diagram source code. You can also choose to use LabView-style controls or OS-styled controls and it is also possible to modify the style of the controls as you like.

The LabView is mainly a desktop application and does not use HTML. It is possible to embed the front panel into a web page and operate it within that page. When making it a web page, a HTML file must be created, but it is not possible to change anything inside the panels from the HTML. User interface manipulation, such as window position and size, does not work as intended on the remote panel. The LabView framework has a large library and is complex, some prior knowledge about Arduino would make it easier to understand. It focuses mostly on visualizing Arduino inputs from sensors in the LabView Front Panels. Still LabView seems user friendly and should be possible to understand with or without technical experience. It hides the code, but users must choose functions from libraries. It is possible to get a lower-level view and modify the Arduino sketch. It has an easy connection with Arduino and supports GPS signals, but has no functionality for group creation.

The restrictions on the user interface makes it not suited for us to use as a development platform. However, it uses some metaphors that we can borrow when developing our own toolkit. They use drag and drop of functions onto the Arduino board and visualizations of how functions will work.

NETLab Toolkit (with HTML5 widget)
The NETLab Toolkit [50] is a project of The New Ecology of Things Lab at The Art Center College in Pasadena, California. The toolkit is a system for tangible interaction sketching and production that integrates with microcontrollers, including the Arduino, and provides a drag-and-drop environment for hardware and media sketching. The environment was originally provided using Flash widgets, but in September 2012 an HTML5 widget was introduced as well. The simple drag-and-drop interface should make it possible to connect with the Arduino without having to program anything, and it should be easy to use for both novice and expert users.

The toolkit is composed of three parts; the widgets, the hub and the media control. The widgets are a set of components for common functionality like getting values from sensors and controlling motors. The components are graphical objects that can be configured without programming. They do, however, include hooks so that it is possible to program when necessary. The hub is a server that communicates with the Arduino. It uses the socket protocol so that any application can talk to the hub. The media control is an application that works with the hub by receiving and forwarding OSC commands.

NETLab supports GPS sensors and has an easy connection with Arduino, but has no functionality for group creation. Even though NETLab is for both novice and expert users, it is using a lot of technical terms. These terms are unsuited for non-technical end-users. We do not find NETLab suitable to use as a development platform because there are restrictions on customizing the interface.

Scratch
Scratch [4] was released in 2007 by the MIT Media Lab. The thought behind it was to make it easy for people of all ages and backgrounds to program their own interactive stories, games and animations, and share them on the Scratch Web site. Scratch is meant to appeal to people that are not programmers. The core audience is between 8 and 16 years old. The goal is not to prepare people for programming careers, but to nurture creativity and systematic thinking. In order to make Scratch suitable for children, three core design principles were established; make it more tinkerable, more meaningful and more social than other environments. Users are presented with a collection of graphical programming blocks that are arranged after each other to form the program. The different blocks fit together in specific ways that makes syntactic sense. Control structures are, for example, c-shaped to suggest that blocks should be placed inside them. The creators primary focus is to keep Scratch simple, as opposed to adding more complex features, and the next step after Scratch will be to move on to a another programming language.

The Scratch creators believe that people work best and enjoy it the most when they are working on personally meaningful projects. Therefore they put high priority on personalization; making it easy to personalize Scratch projects by adding pictures, music, voice recordings and creating graphics, and diversity; by supporting many different types of projects like stories, games, animations, simulations and even Arduino projects. Another team of programmers created Scratch for Arduino (S4A). S4A reads Scratch programs and utilizes the I/O facilities of an Arduino. There are programming blocks for the basic micro-controller functions such as analog and digital read and writes. The I/O configurations are still being developed and, as of now, the components have to be connected in determined ways. In addition, it has only been tested with Arduino Duemilanove and Diecimila and although S4A is compatible with Scratch, it cannot share projects as this is against the Scratch terms of use [14].

Although Scratch provides easy connection with the Arduino board through the S4A extension, it is too high-level to use as a development platform for the aWearable toolkit. It is hard to customize and has a fixed set of components. Creating groups is not provided functionality, but it does support GPS signals.

5.2 Conclusion

In the table 5.1 we have made a summary of the development platforms. Because we need to be able to costume the web interface, we have decided to use Noduino. Noduino puts no restrictions on the interface and it uses HTML5 and JavaScript that we have experience with. The web interface connects easily to the Arduino and can send messages to it. As Noduino is so low-level it does not provide any integrated functionality for handling GPS data or group creation, but it is possible to implement this.

Development Platform	**Summary**
Sen.se	Sen.se is high-level and not good for customizing. It does not support sending data from a customizable HTML5 interface to the Arduino. It has a stepwise connection with Arduino that should be made easier for end-users. It does not provide GPS data out of the box and has no functionality for group creation.
Noduino	Noduino is very customizable. It is made to make a web interface that communicates with the Arduino and it use HTML5 and JavaScript. It has an easy connection with Arduino and it is possible to use GPS signals and create groups with the Noduino platform.
LabView	The LabView has some restriction on customizing the interface and it does not use HTML. It has an easy connection with Arduino and supports GPS signals but does not provide group creation out of the box. LabView use a drag and drop metaphor and visualize the Arduino board, which is something to consider for our own interface.
NETLab Toolkit	NETLab use HTML5 widgets and has some restrictions on customizing the interface. The connection with Arduino is easy. It seems to support GPS sensors, but has no group creation functionality.
Scratch	Scratch is high-level and hard to customize. It is not possible to make a web interface using HTML. It supports GPS signals, but does not support group creation out of the box. It has an easy connection to Arduino using S4A.

Table 5.1: Development Platforms Summary

6 | Design

The aWearable prototype consists of three main tasks; management of groups, customization of applications, and the use of these applications. The different tasks require different tools and techniques that must be combined in the aWearable toolkit. Because it is end-user adapted, usability is particularly important as well. In this chapter we will present the overall design of aWearable and use-cases that illustrates the tasks. We will also explain the design process we went through to design the toolkit.

6.1 Design Overview

The aWearable toolkit consists of a software web application and a hardware device as illustrated in figure 6.1. The aWearable web application is where the user customizes a location-aware- application and uploads it to the aWearable device. The web application includes the social aspect, which is group creation, and the customization aspect, which is the creation of an application. The aWearable device will be the actual customized application that the user can carry around. It consists of a set of parts, or components, that the user can choose to attach. When an aWearable friend is within a set proximity of the user, location data will be presented through these components.

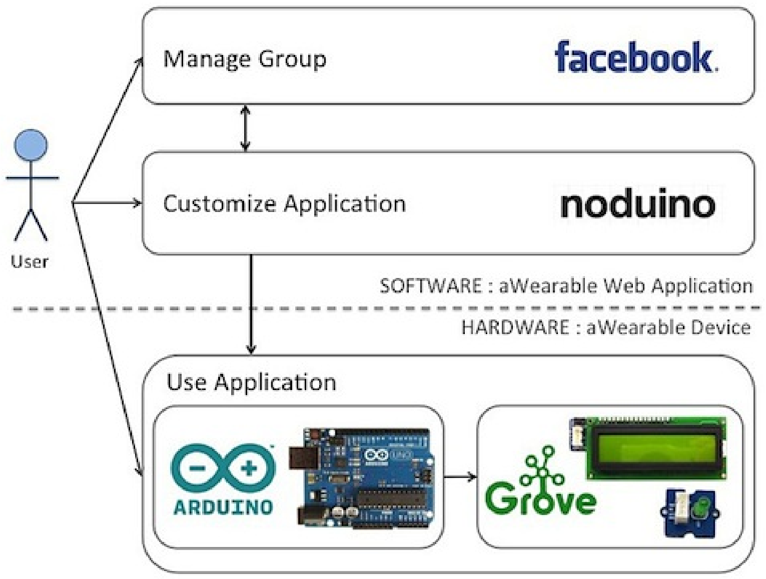


Figure 6.1: Design overview

The different parts of the aWearable toolkit are based on different tools and technologies. We decided to use the Facebook API for group management. By having the users log in with Facebook, we can access their friend-list and add friends to the users' aWearable groups from there. For the customization of applications we have decided to use Noduino, based on an evaluation of development platforms that provides Arduino connection. For the aWearable device, we use the Arduino electronics prototyping platform. To ease the user interaction with the Arduino board we have used the Grove tool set with Grove user interface modules.

6.1.1 Manage Groups

Users of the aWearable toolkit can use Facebook to add friends to be aware of. When clicking 'add friends' in the web application, they get access to their Facebook friend-list. Here they can choose to see all friends, or only friends that are aWearable users. The aWearable users are the friends that use aWearable and therefore have a device that can communicate with other aWearable devices. They choose which friends they want to be aware of and send a request to these friends. The friends added will receive a Facebook request, where they can accept or decline the invitation.

As we expect the aWearable users to use Facebook to add friends, we also use Facebook to create aWearable user accounts. When users have logged in to the aWearable web application, they must agree that aWearable can get their public Facebook profile information and friend-list. To create the aWearable account, they first enter the aWearable ID, which can be found on the device, and then they log in to Facebook. This way the aWearable ID is associated with a Facebook account.

When they go on to the create application page, they will see that they are logged in with Facebook, and can add friends from Facebook.

The aWearable toolkit also have a section with privacy settings where users can chose how much information they want to share with the friends that are following them. The section consists of a set of checkboxes. 'In-range' is one of the options, and this one will always be checked. That a user is within an aWearable friend's range is the minimum of what users have to share in order for the customized application to work. If the aWearable device does not know that an aWearable friend is in-range, no feedback will be given to the user of the device. An overview can be seen in figure 6.2.

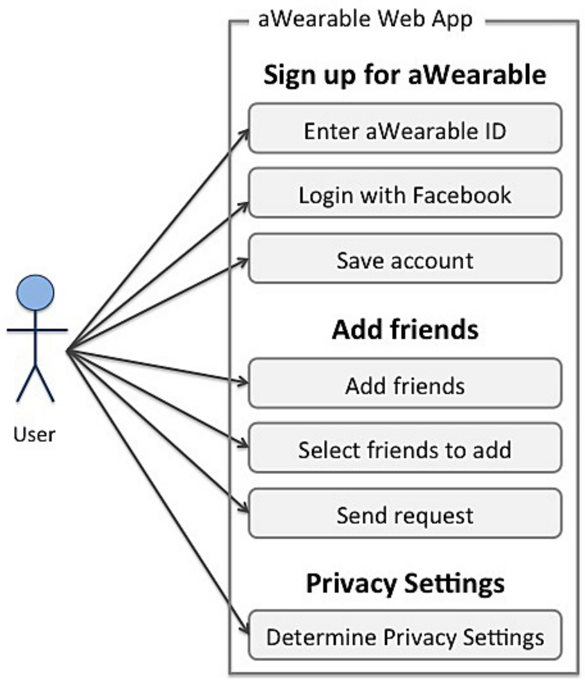


Figure 6.2: User perspective: Manage group

6.1.2 Customize Application

The second part of the aWearable web application is the customization of applications. The steps a user must go through are presented in figure 6.3 an application is mainly customized in the aWearable web application, but the customization is closely related to the aWearable device. First, users must choose what user interface parts they want their aWearable devices to have. The web application presents the same parts as the users get with their aWearable kits, a display, a light, a sound module and a vibration module. The components have to be chosen in the web application as well as for the physical attachment to the device. This is necessary to know what parts that the user has attached, and to output the correct data to these parts. When choosing a part in the web application, a set of behavioral choices are presented and the users must choose how they want the selected parts to behave.

The next step is to physically attach the selected parts to the correct wires on the aWearable device. We chose to have the steps in this order so the user can deal with all that concerns the parts at once, before moving on to other customization characteristics. Now the users must select how big of a range they want for their applications. As illustrated in the scenarios in section 4.2, different range-sizes may be suitable depending on how they are planning to use the application. Once the range has been determined, the application is ready to be uploaded. First, the aWearable device must be plugged to the computer via the USB cable. The connection is completed after the user has clicked on the 'connect'-button and a success message is displayed. The user can push the 'upload'-button that sends the application characteristics to the aWearable device. The device is ready to be disconnected and used.

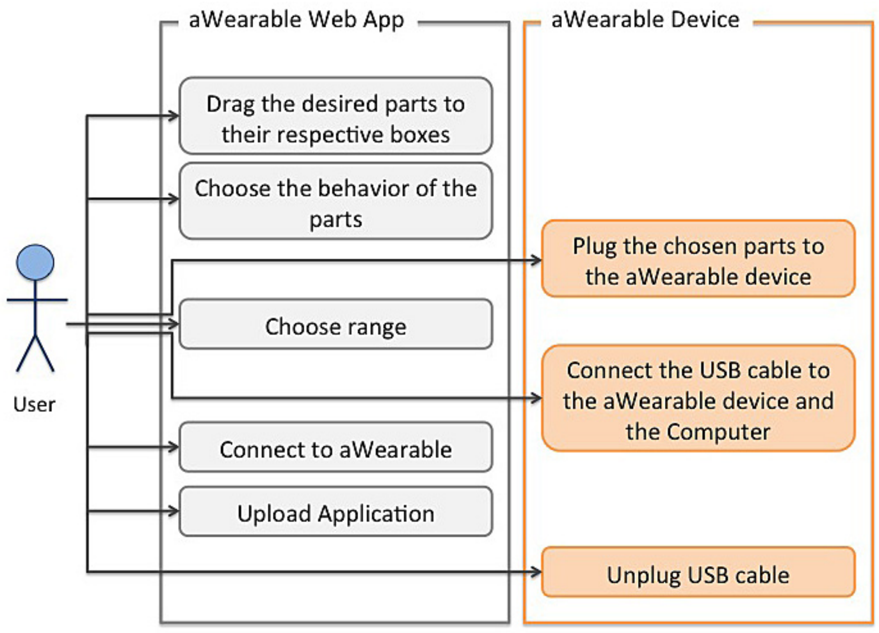

aWearable Web App
aWearable Device
Drag the desired parts to their respective boxes
Choose the behavior of the parts
Plug the chosen parts to the aWearable device
Choose range
Connect the USB cable to the aWearable device and the Computer
User
Connect to aWearable
Upload Application
Unplug USB cable

Figure 6.3: User perspective: Customize application

6.1.3 Use Application

The application is now uploaded to the device and the users take their devices with them to meet up with, or to simply feel aware of their friends. Figure 6.4 illustrates the interaction between the user and the aWearable device when the customized application is in use. A light on the device tells the user whether or not GPS signals are received. When aWearable friends are within each other's range, and the lights are blinking, their devices exchange location information. The users will receive feedback through the parts they have attached to their devices. They can also choose to turn their devices on or off.

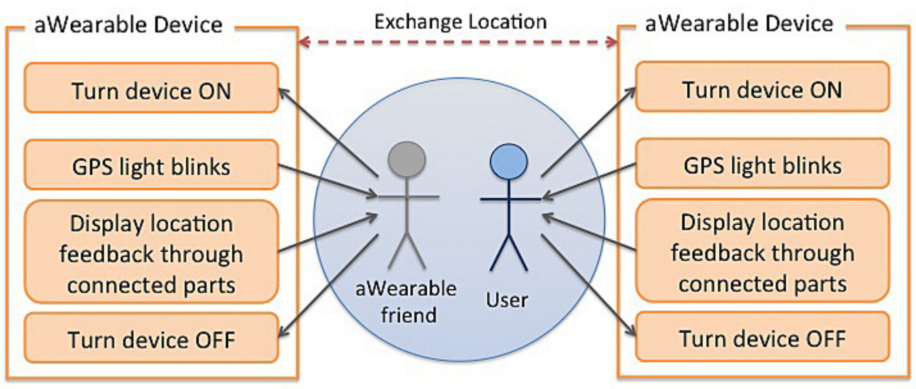


Figure 6.4: User perspective: Use application

6.2 Design Choices

After the system requirement specification, we went through a design process where we discussed possible metaphors and designs suitable for the aWearable toolkit. This process resulted in the design choices behind the looks of the user interface and the aWearable device. The design ideas were elaborated based on the system requirements, and adjusted throughout the toolkit development process.

6.2.1 Design Challenges

We have listed the main challenges when designing the aWearable toolkit. In the following section we will describe how we overcame this challenges.

- Since we want non-programmers and less technical users to be able to use the toolkit, one of our main challenges were to develop a design that requires no prior knowledge from the user.
- There had to be a logical correspondence between the software interface and the hardware device. The users must be able to understand how to attach components correctly to the aWearable device, and also understand how the applications they are creating, are going to work in practice. There must be a natural way for the user to interact with and relate to, the hardware and the software at the same time.
- An Arduino board may look fragile and scary to users who are not familiar with the technology. One challenge that we needed to overcome was to make the aWearable device look appealing to users.

6.2.2 Hardware Design

In this section we will describe the process of designing the hardware interface, referred to as the aWearable device. We needed to cover up the Arduino hardware and simplify how components can be connected.

The earliest hardware prototype consisted of an Arduino UNO board with a GPS module and four LEDs as can be seen in figure 6.5. The green LED is blinking when there are GPS signals coming in. The other LEDs were blinking based on the direction of a hard-coded location. To an inexperienced user the Arduino board, with all its wires and pins, might look incomprehensible and intimidating. It requires electronics knowledge to understand how it works and how to build on it. Resistors are needed to control the current flowing through modules, and the user will have to know about resistors and Ohms law to be able to connect modules correctly. To require that kind of knowledge from users would limit the user group significantly, which is unacceptable as we want less technically experienced users to be able to use the aWearable toolkit.

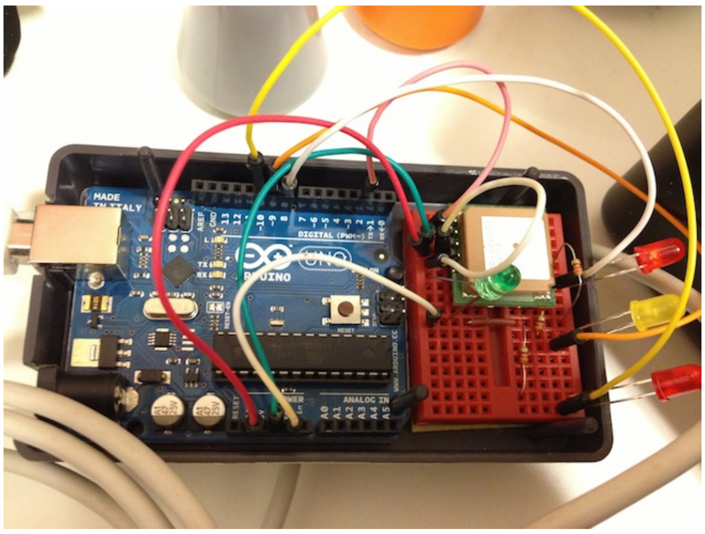
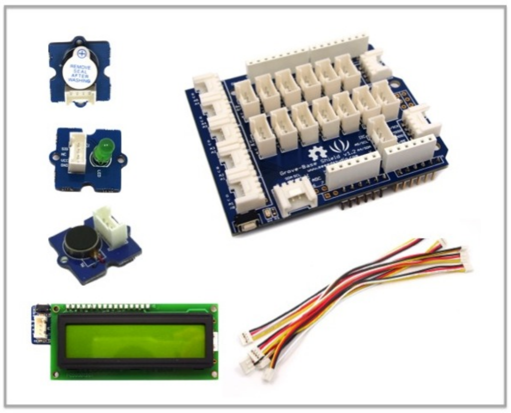


Figure 6.5: Hardware Interface: Early Prototype

Grove

We decided to use a shield to ease the interaction with the Arduino board. Shields are boards that can be mounted on top of the Arduino to extend its capabilities and simplify interaction. We chose to use a Grove shield, as there was a Grove Starter Kit available in the lab. Grove was also used in reaDIYmate. Grove is an open source, easy-to-use, plug-and-play tool designed to simplify fundamental electronics. It consists of a Base Shield and various modules that can be connected to the shield with standardized connectors, without having to worry about resistors. All the pins from the Arduino board is routed into the Base Shield to allow a simpler connection[48], as can be seen in figure 6.6.


(a) Grove Kit

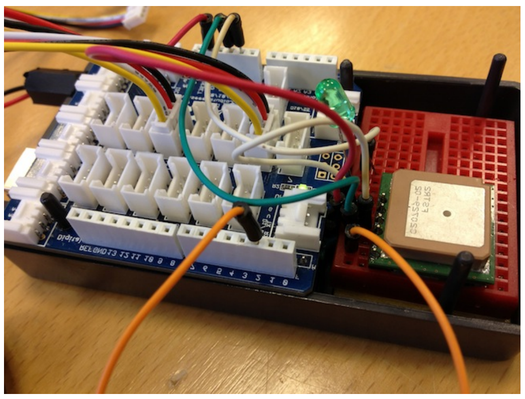

(b) aWearable device with Grove

Figure 6.6: Grove

Mock-up

The Grove kit helps solve the resistor interaction problem, but there are still a lot of wires connecting the GPS to the pins, and also the wires used to connect the modules. We still had to cover up the device to make it look more appealing to users. The idea was to have just the required wires come out from a cover and to id them with numbers so users could plug the right components to the right wires. That way we can cover up the GPS as well, which is a component that users do not interact with and should not have to worry about. The mock- up is shown in figure 6.7.

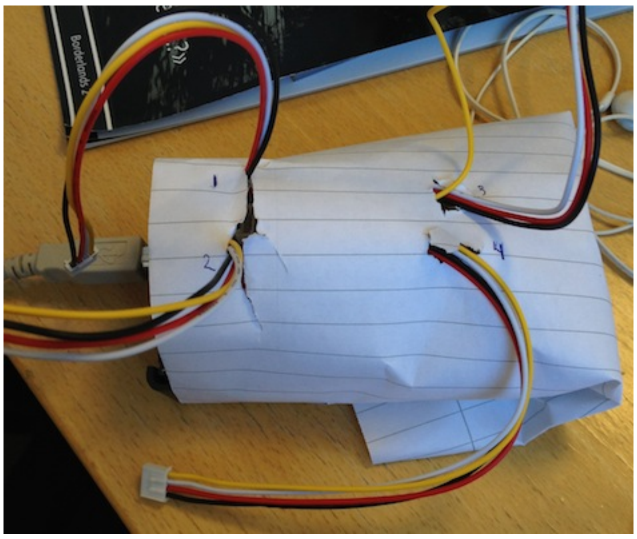


Figure 6.7: Hardware Interface: Mock-up

Final Prototype

For the aWearable device to be desirable it was not enough to wrap it up in paper. We went shopping for a cool bag or a box. We found a bag with velcro on it, which was perfect for attaching the modules. We had to make some adjustments to make it suitable for the aWearable device. The GPS signal light had to be visible and wires had to come out somewhere. How we designed the bag is presented in figure 6.8.

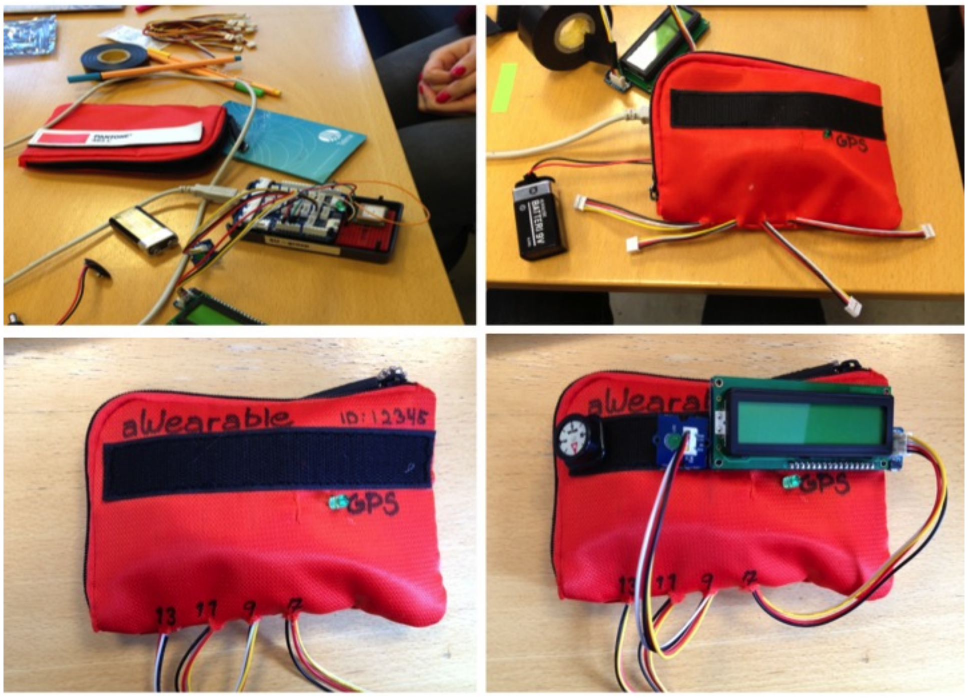


Figure 6.8: Hardware Interface: Final prototype

6.2.3 Software Design

In this section we will describe the design process of the aWearable software interface. We considered different possible designs. As we did so, we focused on the steps that the users are required to carry out to create applications by themselves and tried to exclude excessive information.

One idea was to have a set of steps and two characters that would meet at the end of the path after all steps had been carried out, which is illustrated in figure 6.9. This was meant to represent how an application would work and that the users would get closer to their friends as the application was created. We decided not to go ahead with this approach, as we were worried that the users would not be able to get the whole picture of what an application requires, because they would only be able to see one step at a time. In addition, there would be multiple unnecessary mouse clicks and the users would have to go back in order to change behavior that were selected in earlier steps, if they were to change their mind about something.

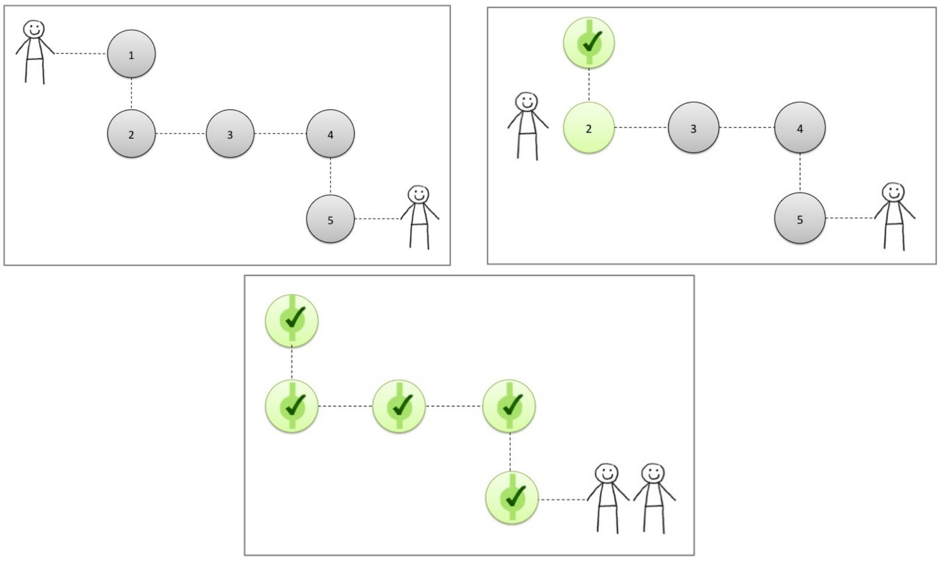
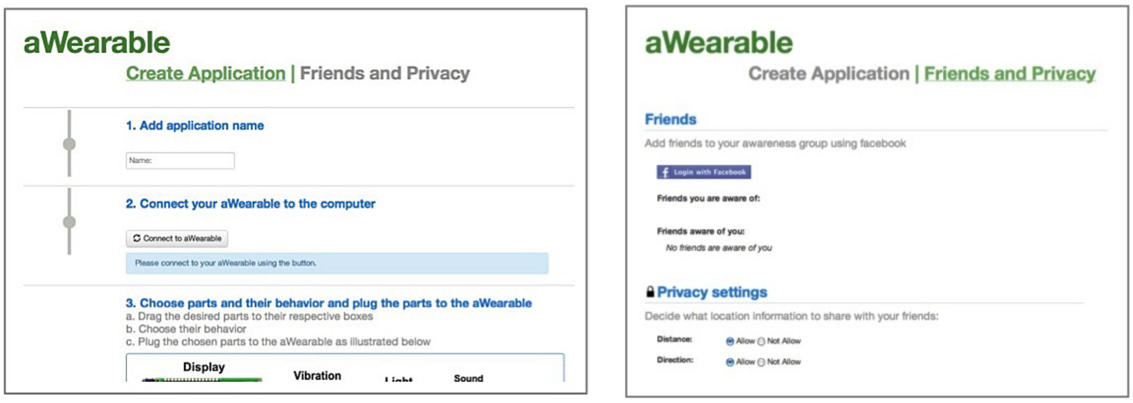


Figure 6.9: Software Interface: Design suggestion

We started out with one separate page for friends and privacy, but to be able to give the users a better overview of what an application includes, we eventually decided to keep all the required steps on one web page. By using drag and drop interaction for the components, and first show the behavior choices when the parts are correctly dropped, we are hiding choices that the user does not have to consider. If the user does not choose the sound component, the user does not have to even see how the sound component can behave. Also drag-and-drop is fun and can be associated with a puzzle, which works well with the hardware interface, as the user has to "puzzle" the parts together.



aWearable

Create Application | Friends and Privacy

1. Add application name

Name:

2. Connect your aWearable to the computer

Connect to aWearable

Please connect to your aWearable using the button.

3. Choose parts and their behavior and plug the parts to the aWearable

a. Drag the desired parts to their respective boxes
b. Choose their behavior
c. Plug the chosen parts to the aWearable as illustrated below

Display Vibration Light Sound

aWearable

Create Application | Friends and Privacy

Friends

Add friends to your awareness group using facebook

Login with Facebook

Friends you are aware of:

Friends aware of you:
No friends are aware of you

Privacy settings

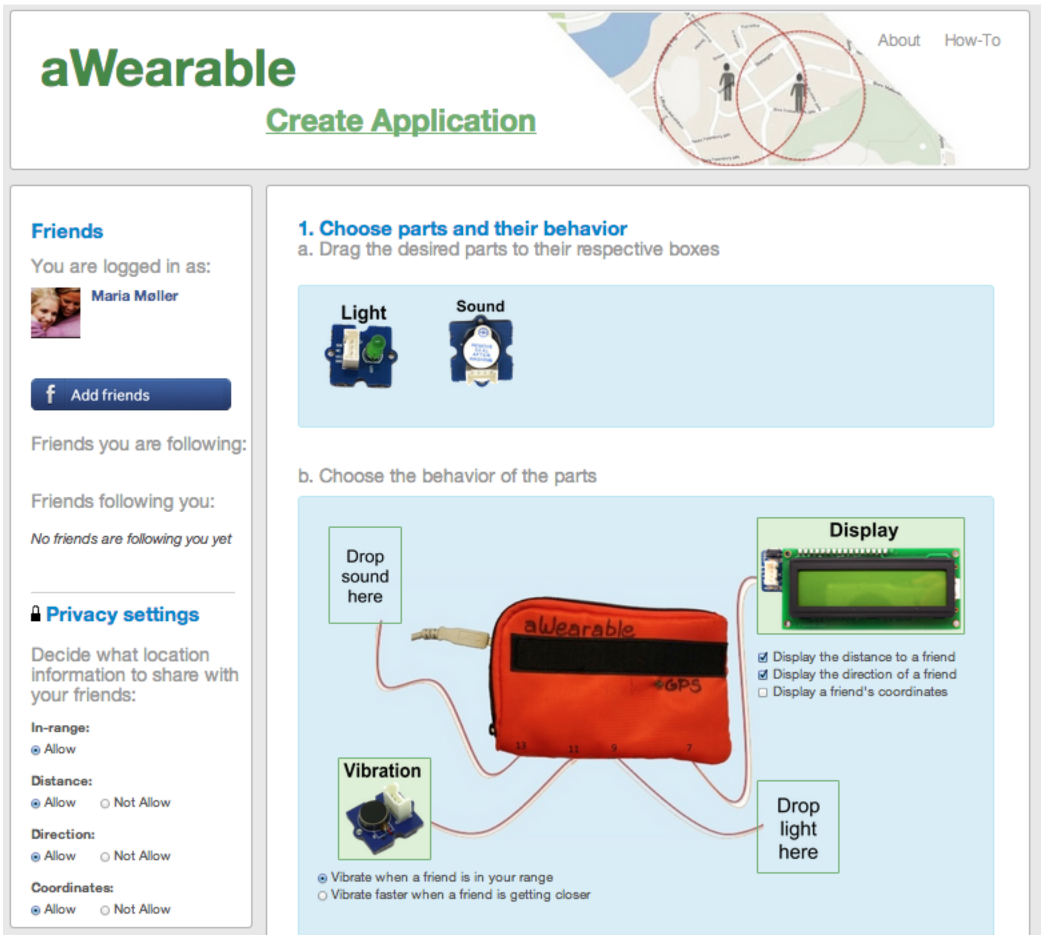
Decide what location information to share with your friends:

Distance: [✓ Allow] Not Allow
Direction: [✓ Allow] Not Allow

Figure 6.10: Software Interface: Early design

Final Prototype

Figure 6.11 and 6.12 presents the final design of the software user interface for the aWearable toolkit. The image is meant to give an overview, before different elements are described in more detail.



Friends

You are logged in as:

Maria Møller

Friends you are following:

Friends following you:

No friends are following you yet

Privacy settings

Decide what location information to share with your friends:

In-range:
[✓ Allow]

Distance:
[✓ Allow] Not Allow

Direction:
[✓ Allow] Not Allow

Coordinates:
[✓ Allow] Not Allow

1. Choose parts and their behavior

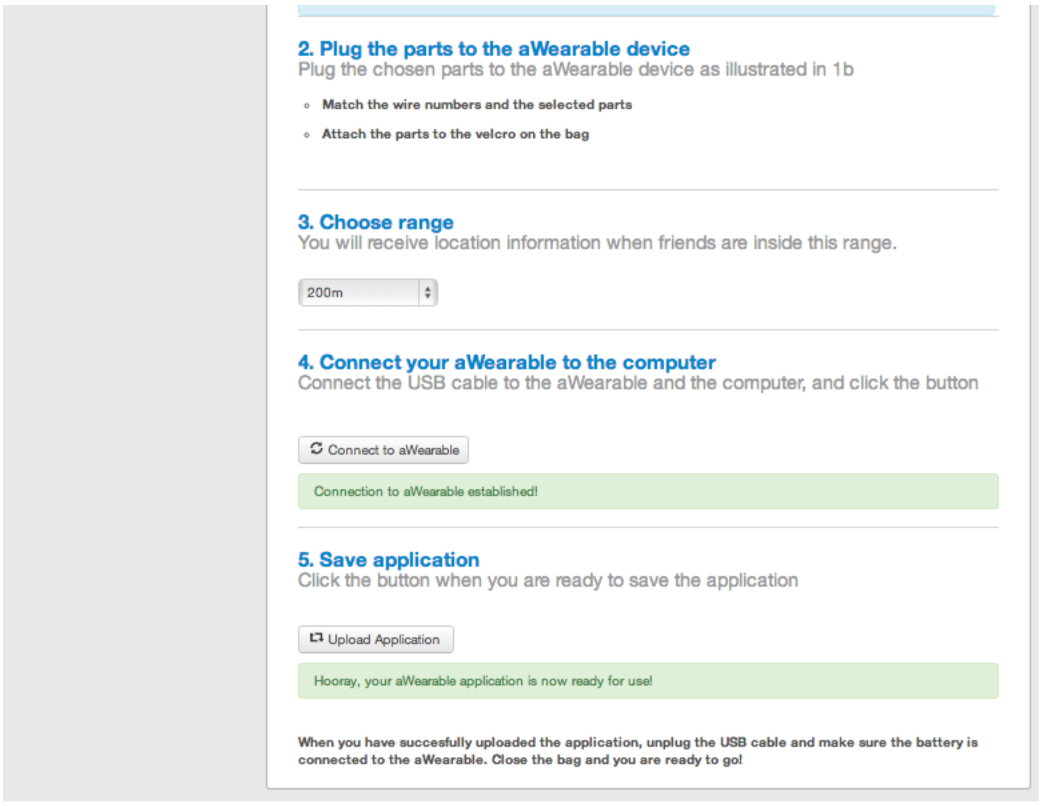
a. Drag the desired parts to their respective boxes

b. Choose the behavior of the parts

- [✓ Display the distance to a friend]
- [✓ Display the direction of a friend]
- Display a friend's coordinates

- [✓ Vibrate when a friend is in your range]
- Vibrate faster when a friend is getting closer

Figure 6.11: User interface: Overview part 1



2. Plug the parts to the aWearable device
Plug the chosen parts to the aWearable device as illustrated in 1b

- Match the wire numbers and the selected parts
- Attach the parts to the velcro on the bag

3. Choose range
You will receive location information when friends are inside this range.

200m

4. Connect your aWearable to the computer
Connect the USB cable to the aWearable and the computer, and click the button

Connect to aWearable

Connection to aWearable established!

5. Save application
Click the button when you are ready to save the application

Upload Application

Hooray, your aWearable application is now ready for use!

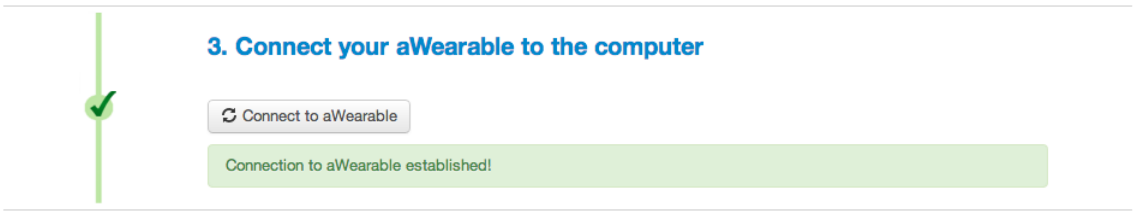
When you have succesfully uploaded the application, unplug the USB cable and make sure the battery is connected to the aWearable. Close the bag and you are ready to go!

Figure 6.12: User interface: Overview part 2

To ensure a user friendly interface we decided to take into account Jakob Nielsen's - 10 Heuristics for User Interface Design [40], that we are familiar with from HCI courses. We will describe the different heuristics and what we have done to accommodate them.

Visibility of system status

Visibility of system status is based on the importance of keeping the user informed about what is happening. To assure this, the user should receive feedback that is appropriate to his or her actions, within reasonable time limits. To accommodate this heuristics we have used the colors green and red to show if the users' actions are right or wrong, in addition to feedback text. When the user clicks the 'connect to aWearable' button, they will get a green message if the connection was successful and a red message if it was unable to connect. See figure 6.13. The same green background color appears when components are dropped correctly, and when the 'Upload application' button is click, depending on whether or not the user has completed all required tasks. If the application is unfinished, the user will get a red error message, telling the user what to do. We also added a status bar to the left of each step that became green when the step was completed. This was later removed, as we were unable to verify if the users had chosen the components they wanted and attached them correctly to the aWearable device without making users carry out unnecessary button clicks.



Connection to aWearable established!

Figure 6.13: User interface: Success message

We also display 'You are logged in as' that shows the user's Facebook name and profile picture. When new users sign up they are informed about what parts of their Facebook account that the aWearable toolkit will access.

Match between system and the real world

Match between the system and the real world involves using language and concepts that are familiar to the user and to have information appear in a logical order. We have focused on keeping the language in the interface straightforward and non-technical. The image that we have used to show how parts should be connected to the aWearable device replicates the actual device, so users can recognize the wires and connect them to the correct components.

User control and freedom

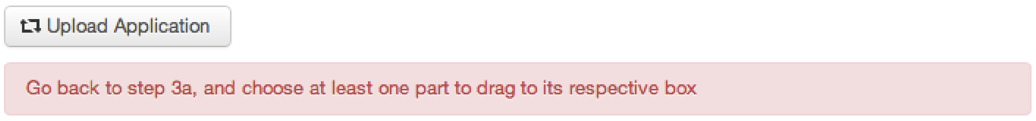
User control and freedom are promoted by providing users with undo and redo opportunities. There is no state in the aWearable interface where the user is blocked. Parts and their behavior can be altered at any time. They are easily selected by dragging them into their respective places and deselected by dropping them back into their original positions. The range can also be altered as the user wishes. If an application is altered after it has been uploaded, the new application behavior will replace the old one.

Consistency and standards

To use consistency and standards in a user interface means to follow platform conventions and to have the same actions perform in the same ways, so the user knows what is coming. Drag-and-drop is a standard most people are familiar with. User know how to handle draggable components and are able to predict their behavior, like when you click and hold, the component will follow the movements of the mouse. We used standard radio buttons, where round buttons indicate that only one option can be selected while square checkboxes indicate that multiple options are allowed. Buttons should look like buttons and have mouse-over functionality that makes it clear that they are clickable. We have also borrowed conventions from social media. The users should be able to recognize how to add friends from Facebook and we have used the terms following and followers known from Instagram and Twitter. As most people in our user group are familiar with this, they should be able to understand how friends' management works in the aWearable toolkit.

Prevent errors and help users recognize, diagnose, and recover from them

Here we have merged two heuristics. Error prevention means that one should design to prevent problems from occurring. If they do occur, one should provide good error messages. In the aWearable interface we are preventing errors from occurring by notifying the user at once if a part is dropped in the wrong place. We also have validation on the 'Connect to aWearable' and the 'Upload Application' buttons. If an error occurs the user will get a red error message telling them how to correct it as shown in figure 6.14.



Go back to step 3a, and choose at least one part to drag to its respective box

Figure 6.14: User interface: Error message

Recognition rather than recall

Recognition rather than recall is based on minimizing what the user has to remember by making objects, options and action clear and visible. The user should not be required to remember information from one part of the interface to another. The aWearable interface does not make the user remember such information, having all the required steps at one page and therefore visible at all times. We are also basing the connection between the software and the hardware on recognition. The components in the interface resemble the physical components and the image of the aWearable device matches the actual device.

Flexibility and efficiency of use

Flexibility and efficiency is about giving the more experienced users accelerators to speed up the interaction, so that the system appeals to both the novice and to the experienced. In the aWearable interface users should not have to manage friends every time they want to upload a new application to their aWearable device. Once they are logged in to the web app, previously added friends will be listed in the following section, and followers they have accepted will be listed in the 'followers' section. This will benefit both novice and more experienced users.

Aesthetic and minimalist design

To have an aesthetic and minimalist design involves excluding irrelevant or rarely needed information, as irrelevant information will compete with the relevant information. We have worked to minimize "noise" in the user interface and we find all the included information to be relevant. We have also worked to keep the layout clean without disturbing images and colors.

Help and documentation

Help and documentation should be easily accessible if needed. In the aWearable interface we have added an 'About' page where the user can read about the purpose of the toolkit. We also give an example of an application to give the user some inspiration. On the 'How-to' page we list concrete steps to be carried out in order to create an application.

7 | Implementation

We have developed a prototype as a proof of concept for an end-user adapted toolkit that facilitates customization of wearable location-awareness applications. The prototype is a simplified version of the toolkit, where a limited set of requirements has been implemented. It consists of an integration of a set of different technologies and tools, mainly the Facebook API, the Noduino framework, the Arduino prototyping platform and the Grove tool set. The focus of the implementation was to produce a prototype with adequate functionality to be evaluated through user testing.

7.1 Implementation overview

Figure 7.1 gives an overview of the different tools that are integrated into the prototype. The web application consists of an HTML5 user interface where the essential parts are integration with Facebook; for group creation and login purposes, and integration with Noduino; for connection with the Arduino board and the passing of application characteristics. The aWearable device consists of an Arduino board with a Grove base shield, Grove user interface modules and an Arduino GPS module. A message from the web application is pushed to the Arduino through the Noduino socket connection. Such messages consist of the application characteristics that were chosen by the user in the GUI. The GPS module on the device receives GPS signals that are handled with the TinyGPS Arduino library, which also facilitates computation of distance and direction to a target position. When the aWearable device is in use, feedback to the user will be given through the Grove user interface modules connected to the device.

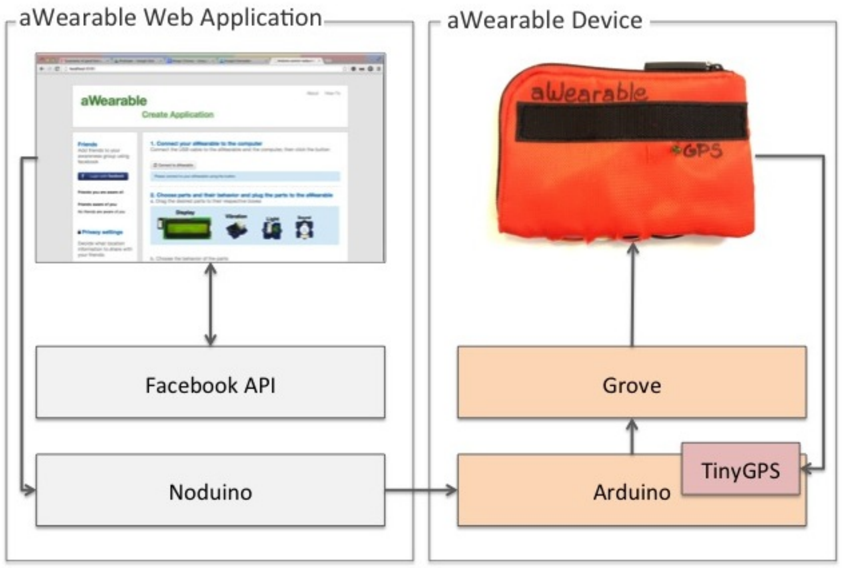


Figure 7.1: Implementation overview

.

7.2 Implementation of the aWearable Web App

7.2.1 Integration with Facebook

We use the Facebook API to handle login and group management. This part of the prototype is not our main focus and is therefore not fully implemented. We only work with one aWearable device and the aWearable device does not connect to other aWearable devices. The adding of friends is only for demonstration purposes.

The Facebook Developers website [3] has documents on how to integrate Facebook with web applications and how to use the Facebook features. First, we had to create an application on Facebook to receive a Facebook application ID. This ID is needed to get the JavaScript SDK to work. The JavaScripts are added into the prototype HTML code. Figure 7.2 shows the Facebook related methods that we are using.

The Facebook SDK for JavaScript provides a set of client-side functionality that enables us to use different Facebook functions, such as the Request Dialog. The Request Dialog sends a request from a user to one or more other users, which we needed for adding friends. An initialization function, 'FB.init()', will load and initialize the JavaScript SDK with the most common options, and the asynchronous function, 'function(d, s, id)', will load the SDK asynchronously so it does not block loading other elements of our page. The JavaScript SDK also requires an fb-root element to be present in the aWearable web page, since this is where the SDK insert elements.

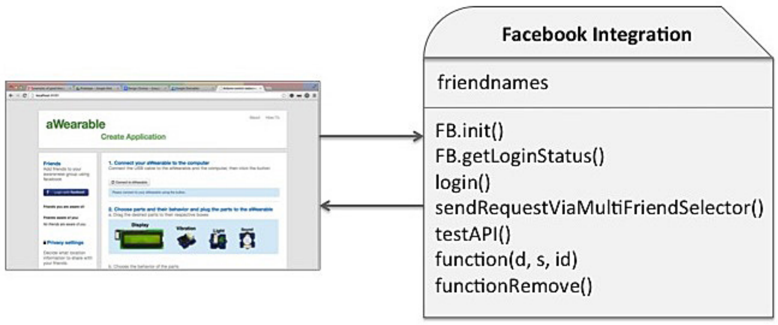


Figure 7.2: Facebook related functions

Login
When opening the 'Signup' page in the aWearable web application, the FB.getLoginStatus function is automatically called. The call checks if a user is already logged in and if the user has authorized the current application. If this is the case the testAPI function is called, which returns the user's name and Facebook picture to the aWearable page. If the user is not logged in, the user must click the 'Login to Facebook' button which calls the 'login()' function. The 'login()' function opens a login window, where the user logs in to Facebook with their Facebook information. Here one must also agree that the aWearable application gets access to the user's public profile and friend-list. The sequens diagram for the login function is presented in 7.3.

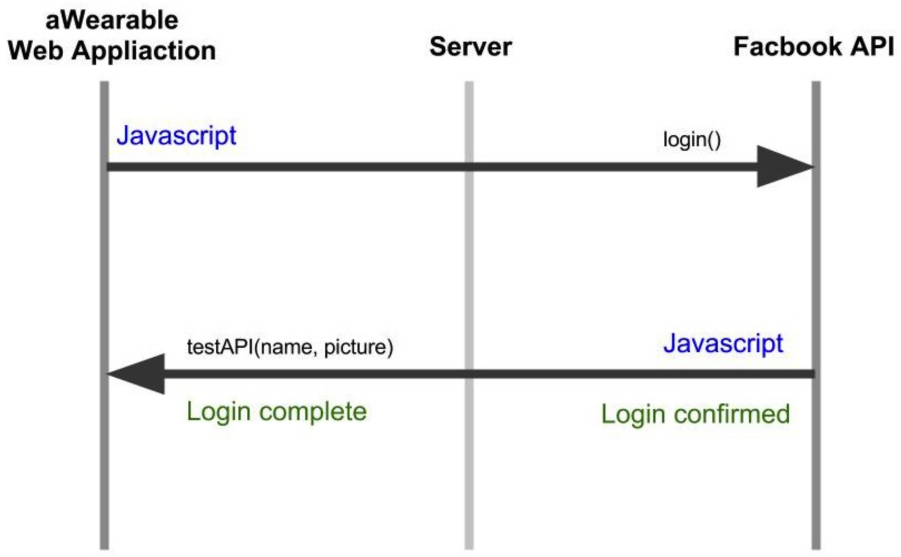


Figure 7.3: Sequences diagram: Login

When the user has registered an aWearable account, the next step is to create an application. The 'Create-Application' page will do the same call to 'FB.getLoginStatus()' as the 'Signup' page. Now the user is logged in, and the user's name and Facebook picture will automatically be presented on the 'Create Application' page.

Add friends
To send the requests we used the 'Multi Friend Selector Dialog' function. This function is called when clicking the 'Add friends' button, and it opens an overview of the user's Facebook- friends. Here the users select the friends they wish to add. The selected friends will receive a request notification on Facebook. Clicking this will redirect them to a page where they can accept or decline the friend request. The redirecting of the page is controlled in the Facebook application we created at the beginning.

The theory is that when a friend accepts the request, their name and aWearable ID will be sent back to the friend that sent the request. In our prototype, the name of the friend the user added would automatically show up in their web application under 'Friends you are following', and the location of this friend is hard coded in the source code. The sequens diagram for adding friends is presented in figure 7.4.

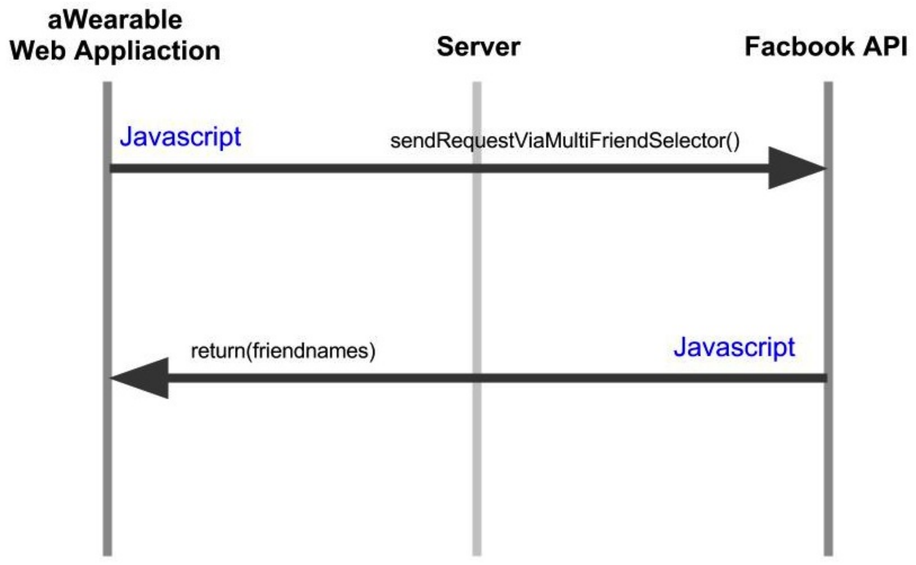

aWearable Web Appliaction
Server
Facbook API
Javascript
sendRequestViaMultiFriendSelector()
return(friendnames)
Javascript

Figure 7.4: Sequences diagram: Add friends

7.2.2 Integration with Noduino

The prototype is based on the Noduino framework, a choice that was made during the evaluation of development platforms. As described in chapter 5, Noduino is composed of multiple technologies. Figure 7.5 illustrates the interaction between the node.js WebSocket Server, the Arduino and the HTML5 web application. As node.js allows JavaScript on the server-side, JavaScript is used to send messages to the aWearable device from the aWearable web application.

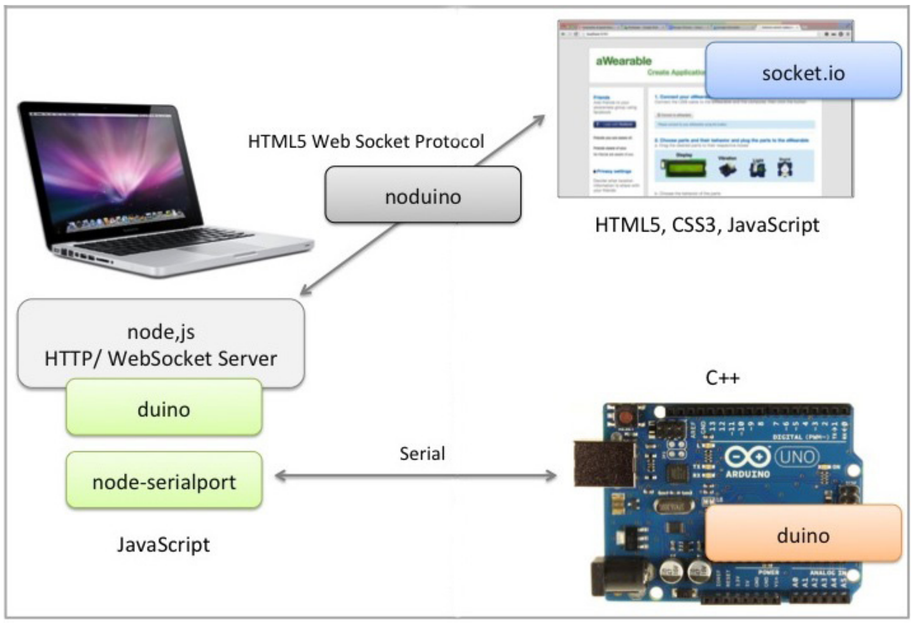


Figure 7.5: Noduino interface overview

The main advantages of using Noduino are the more or less "out-of-the-box" connection between the HTML5 user interface and the Arduino board. The establishment of the connection between the aWearable web application and the aWearable device is done through a serial USB connection. This functionality is abstracted away by the framework and we did not have to worry about the details of the connection. The passing of messages happens over a socket connection built into the Noduino framework. That means that once the correct message has been composed, we can pass this message on to the Socket.IO part of the framework and trust that it is properly transported to the Arduino board. A message is pushed from the web application to the server and over the USB connection, to the Arduino board, as shown in figure 7.5. For the prototype we find USB connection to be acceptable, but ideally the aWearable toolkit should have wireless connection between the user interface and the device.

In figure 7.6, we have listed the main classes that we have been working with when implementing the GUI and the communication with the Arduino board. The figure includes the most important variables and functions that concern the Noduino part of the aWearable web application and the Awearable JavaScripts that we have included.

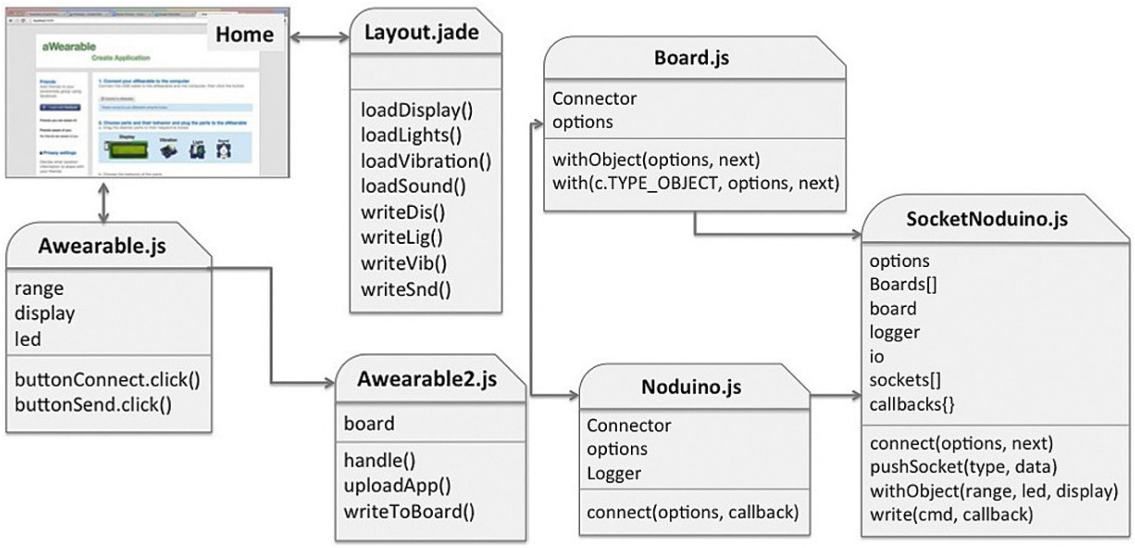


Figure 7.6: Main class diagram

'Home' contains the HTML5 code that makes up the GUI. JavaScript functions in 'Layout' supports the interface. The 'load()' functions are for presenting behavioral choices when the light-, display-, sound- or vibration- component have been dragged and dropped in the correct spot. The 'write()' functions composes arrays based on what components and behavior that have been selected. 'Layout' also includes links to stylesheets and scripts. 'Awearable.js' handles user input from the GUI. An Awearable object is created based on the application characteristics that the user has chosen. Further, 'Awearable.js' calls functions in 'Awearable2.js'. This is the JavaScript that communicates with the Noduino classes 'Board.js' and 'Noduino.js'. These are again connected to 'SocketNoduino.js', which is the connection to Socket.IO.

Connect to Arduino

Figure 7.7 presents a simplified sequence diagram that describes what happens when the user clicks on 'Connect to aWearable' in the web application.

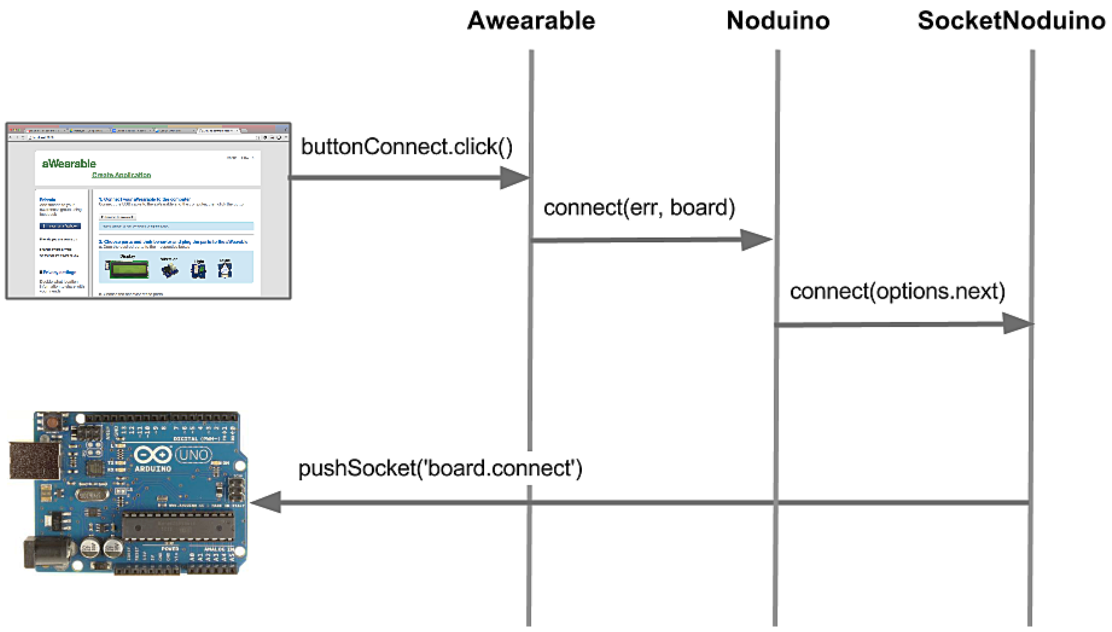


Figure 7.7: Sequence diagram: Connect to Arduino

When the 'Connect to aWearable'-button is clicked, the 'buttonConnect.click()' function is called in 'awearable.js'. Further, the 'handle()' function is called where a Noduino-object and a SocketNoduino-object is created. Next, 'connect()' is called on the Noduino object. If the method returns an error, an error message is displayed in the user interface, and if it is a success 'Connection to aWearable established' is displayed. 'Connect()' is called once more on the SocketNoduino object that calls 'pushSocket()'. A 'board.connect' message is sent and a connection is established between the aWearable web application and the aWearable device, using Socket.IO.

Upload Application

The choices made in the aWearable web application are sent to the aWearable device when an application is uploaded. The application characteristics are composed into a message and pushed to the Arduino board with Socket.IO. A high-level sequence diagram is presented in figure 7.8.

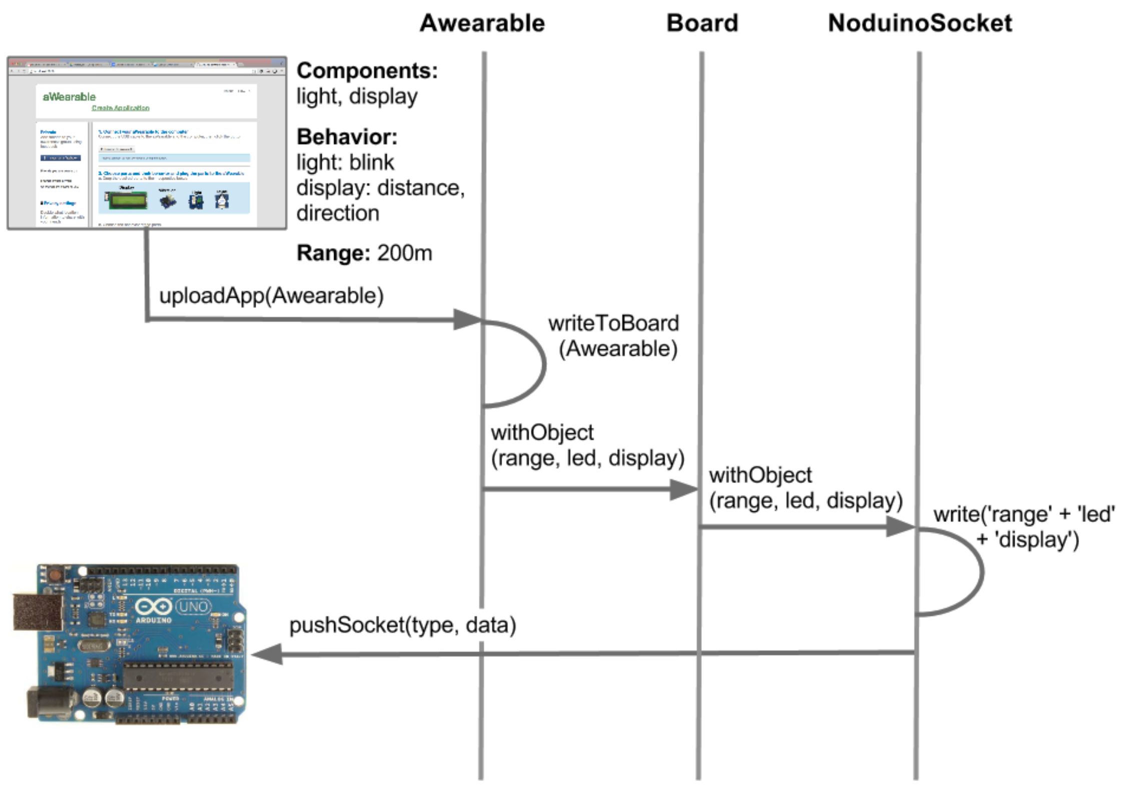


Figure 7.8: Sequence diagram: Upload application

When the user clicks 'Upload Application' in the GUI, 'buttonSend.click()' is called. This function assigns HTML variables to an Awearable-object. If all the required variables are assigned, 'uploadApp(Awearable)' is called, passing on the 'Awearable' object. If not, the user will get error-messages in the web application. If 'Upload-App(Awearable)' was successful, withObject(range, led, display) is called and an Awearable-object is created on the board. 'withObject()' is called once more, this time on the socket connection. The range, led and display is combined into a text String and pushed to the Arduino through the 'pushSocket()' method.

7.3 Implementation of the aWearable Device

The aWearable device consists of a hardware part and a software part. An Arduino UNO board, with a GPS module, and Grove modules, puts the hardware together. The software is the Arduino Sketch, which consists of code for handling messages from the aWearable web application, the TinyGPS library for handling GPS data, and code for handling output to the Grove user interface modules.

7.3.1 Arduino Hardware

We use the Arduino Uno Board as the foundation for the aWearable device. The microcontroller board consists of 14 digital input/output pins, 6 analog input pins, a 16MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It uses an ATmega328 microcontroller that is programmed as an USB-to-serial converter [2].

Grove
To get an easier interaction with the Arduino Board, we used the Grove tool set. The Grove base shield is mounted straight on top of the Arduino Uno board. The shield has pins for all of the original Arduino Uno pins and standardized connectors for the Grove modules. For the prototype we use three Grove user interface modules in addition to the shield; the Serial LCD, the Buzzer and the LED. We were unable to get hold of the Vibration Motor as it was out of stock.

Fastrax UP501 GPS Module
Connected to the board is the Fastrax UP501 GPS receiver module. It provides complete signal processing from the internal antenna to serial data output in NMEA messages. Sometimes on start-up the GPS receiver searches for satellites and collect almanac data, which is data that every GPS satellite transmits regarding the state of the entire GPS satellite constellation. This might take up to 12 minutes. Once the data is collected, the module enters 'Low Power Tracking mode'. If the GPS already has this almanac data in memory, it takes less time to get ready. The Fastrax GPS is illustrated in figure 7.9. On the prototype the TXD pin, which is the output, is connected to pin 2. The GND, or Ground, pin is connected to the GND pin on the board. VDD and VDD_B are the power pins, while PPS stands for 'pulse per second'. We have connected the PPS pin to a LED. The LED will blink when GPS signals are received.

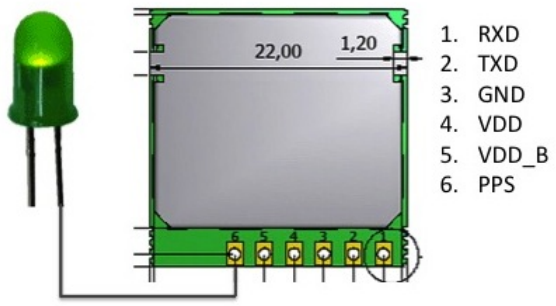


Figure 7.9: Fastrax UP501 GPS Module

7.3.2 Arduino Sketch

The Arduino sketch is the c++ based software code that is uploaded to the Arduino board. In the prototype the sketch called 'du.ino', is composed of three code bases; code from the Noduino framework that is handling messages over the USB connection, code from the tinyGPS library for handling GPS data over a software serial connection, and code from the Grove framework for controlling output to the Grove components. It is the intermediary between the GUI and the aWearable device. Variables and functions is presented in figure 7.10.

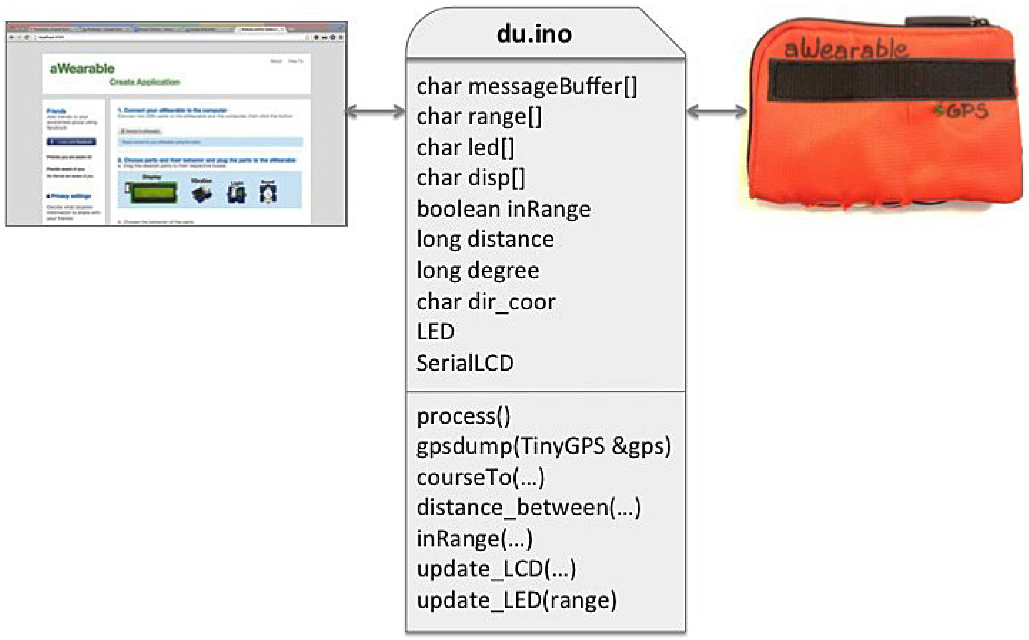


Figure 7.10: Arduino related variables and functions

Figure 7.11 is a simplified illustration of the operation of the Arduino sketch, 'du.ino'. It is getting data from the GUI and from the GPS module on the Arduino board. The message from the GUI is divided into multiple arrays that are used to determine what components to update and how they will behave. Data from the GPS module are updating every 1000ms. The course and distance to the target is computed. If the distance is within the range that was set in the user interface, the Grove components are updated according to the GPS data and the choices the user has made. Different aspects of the sketch are described in more detail below the figure.

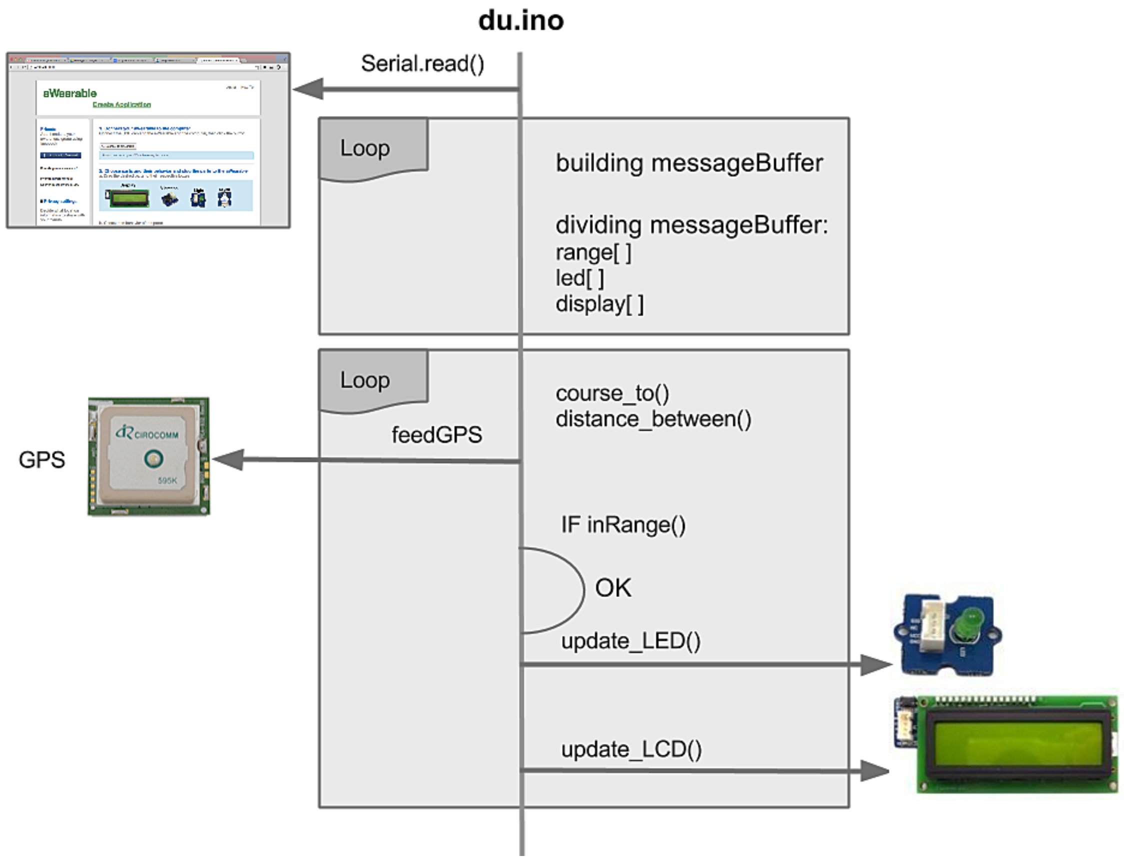


Figure 7.11: Sequence Diagram: Arduino Sketch
.

messageBuffer
The messageBuffer is built in the 'void loop()' where characters are read from the serial port connection and added to the buffer one by one. When the messageBuffer is complete the 'process()' method is called, where the messageBuffer is split up into multiple char arrays; range[ ], led[ ] and disp[ ].

TinyGPS
We used the TinyGPS library [28], developed by Mikal Hart, for managing GPS data. The library is compatible with Arduino and provides NMEA GPS functionality like position, date, time, altitude, space and course. An object is created, fed serial NMEA data from the GPS component and parsed into readable GPS data. TinyGPS provides functions for getting course, cardinals and distance to a target location. These are the most important ones for our purpose, as we are interested in two, or more, positions in relation to each other. The 'distance_between()' function returns the distance in meters between the GPS' position and the target position. The 'course_to()' function returns course from the GPS' position to the target position in 100th of a degree, for example 180 degrees. 'cardinal()' is a conversion of the course into compass direction where NE means North-East.

TinyGPS uses the SoftwareSerial library [27] that allows serial communication on other digital pins than 0 and 1, which supports the built-in serial communication and the USB connection to the computer. As the name suggests the library is using software to replicate serial communication functionality. This is necessary when multiple serial connections are needed. TinyGPS uses SoftwareSerial to get GPS data in a timely and reliable manner.

Grove
Grove code is put directly into the Arduino sketch. We have implemented the Serial LCD display and the Grove LED. We were not able to get hold of the vibration module as it was out of stock. The sound module was not implemented as we felt the display and light were enough for our testing purposes.

The serial LCD requires the inclusion of the Serial_LCD library in the Arduino sketch. First the LCD must be initialized. In the aWearable toolkit we set the pins in the Arduino sketch and that way limit the users choices. The users can plug the display to the correct wire rather than connecting the wire to the Arduino board as well. The library offers a set of methods for handling output to the display. First the display must be set up using 'slcd.begin()'. Further, it has methods for setting the cursor, clearing and printing to the display. We used the PString ("Print to String") library, also by Mikal Hart, to format text so that it can be displayed correctly on the Serial LCD. It is a Print-derivative string class that reproduces text into character buffers.

The LED does not require a library. In can be used directly like regular Arduino LEDs by defining it as output and set the pin to be HIGH or LOW.

Computing Location
We added the functions 'inRange()', 'update_LCD()' and 'update_LED()' to control output to the Grove user interface components. The 'inRange()' function determines whether or not the target location is within the range provided in the 'messageBuffer', by checking if the distance between the locations are smaller than the specified range. If this is true, 'update_LCD()' and 'update_LED()' is called, depending on if these components were selected in the web application. When the display is selected in the web application, the 'disp[ ]' array will have values, for example 'd12', which indicates that the user has selected to display distance and direction. The method consists of a set of conditions that determines what will be printed to the LCD. When the light is selected in the web application, the led-array will have values, for example 'l2', which indicates that the user has selected "Blink faster when a friend is getting closer". The method consists of two conditions; if the second character in the array is '1' it will cause the LED to light up and if the character is '2' the LED will blink faster based on the distance between the locations.

7.4 Prototype Limitations

To ease the implementation, some functional requirements have been neglected and parts of the toolkit have been simplified. The main focus have been on the customization and use of an application, and less on creation of groups, as group creation is an aspect of the prototype that can be "faked" without affecting the evaluation. We consider creating the application to be the main task, as we want to look at end-user development support in making applications.

One limitation of the prototype is regarding the GPS receiver module. We have had some issues with long start-up times, especially on cloudy days or indoor. We had the possibility of changing the module to a faster GPS receiver before the evaluation phase, but this module had no pin for the LED. The LED that blinks when the GPS is working is an important feedback to the users as it shows when their application is working and not, and therefore we decided to go ahead with the original GPS module.

That the toolkit requires USB connection and does not work wirelessly is another limitation of the toolkit that would have been a main priority to improve if the toolkit were to become reality. With a wireless connection we could use a smartphone to control aspects of the created applications, like updating the 'followers'- and 'following'-groups while on the move. We could also have pushed the GPS functionality over on the smartphone and used the smartphone's built-in GPS instead of having a GPS module attached to the aWearable device. This however, proposes new limitations, as the smartphone would have to be carried around with the aWearable device for it to work.

The order of the steps in the web application is not accidental. The display needs to be connected physically to the aWearable device before a connection is established between the device and the web application. This is because the initialization of the display happens in the 'setup' function in the Arduino sketch. The 'setup' function is run when the 'connect' button is clicked, and if the display is plugged after the setup, it will not be initialized and it will not work. We have not focused on debugging, performance and security of the code, since this is only a prototype

7.4.1 Requirements Evaluation

Table 7.1 and 7.2 presents an overview that shows to what degree the different functional requirements have been implemented in the prototype. Some of the requirements have only been partly implemented, which mean they have been implemented to some degree or "faked", so that it appears that they are working correctly. The table is followed by a further description of requirements that have not been fully implemented.

Id	**Software Requirements**	**Status**
SFR01	Sign up using Facebook	Fully implemented
SFR02	Retrieve list of Facebook-friends	Fully implemented
SFR03	Add friends to follow	Partly implemented
SFR04	Delete friends from 'following-group'	Partly implemented
SFR05	Manage follow-request	Partly implemented
SFR06	Delete friends from 'followers-group'	Not implemented
SFR07	Set privacy	Partly implemented
SFR08	Choose components	Fully implemented
SFR09	Choose component behavior	Fully implemented
SFR10	Select range	Fully implemented
SFR11	Connect to Arduino	Fully implemented
SFR12	Upload application to Arduino	Fully implemented
SFR13	Save application	Not implemented
SFR14	Edit application	Not implemented
SFR15	Delete application	Not implemented

Table 7.1: Software: Functional requirements evaluation

Id	**Hardware Requirements**	**Status**
HFR01	Turn the device ON and OFF	Not implemented
HFR02	Connect to the computer	Fully implemented
HFR03	Connect components	Fully implemented
HFR04	Get GPS data	Fully implemented
HFR05	Compute location	Fully implemented
HFR06	Output feedback when followed devices are in-range	Partly implemented
HFR07	Output feedback on chosen components	Partly implemented
HFR08	Output feedback with chosen behavior	Partly implemented

Table 7.2: Hardware: Functional requirements evaluation

In requirement SFR03 it is possible to choose friends to add through a Facebook connection and the added friends appears in the following- section in the user interface. This is as far as the implementation goes. The friends are not saved anywhere and will no longer appear in the GUI if the users try to create a new application later on. The same goes for requirement SFR04. Users can remove friends from the following-section.

Requirement SFR05 has only been partly implemented for demonstration purposes. When someone adds a user, a Facebook request is sent to this user, but it is not possible to actually accept or decline it. When we evaluate with users, this will be the first time that they sign up for the toolkit and therefore, they will naturally have no followers. Requirement SFR06 is not implemented. Since there are no followers there will not be anyone to delete.

Requirement SFR07 is also implemented for demonstration only. There is only one aWearable device that is displaying information about a fixed location. Nobody will be receiving the user's location information and therefore it will not matter what privacy settings that the user chose. Likewise, it will not matter what privacy settings other users have. For the same reasons, we have not included a database, and requirements SFR13, SFR14 and SFR15 have not been implemented. The first time the application is used it is natural that no applications are saved.

We have not implemented an ON/OFF button on the aWearable device, as it reads in requirement HRF01. Users will not be turning the device OFF during our evaluation, and it would have added unnecessary complexity to the prototype. Therefore, we decided not to take the requirement further.

To simplify the prototype further, we have only partly implemented requirements HFR06, HFR07 and HFR08. The aWearable device does not find friends that are in range. Instead we have hardcoded a location that it computes distance and course to. This was a measure to make the implementation phase feasible based on our programming experiences. Even though this was a big simplification we believe that the prototype does not suffer from it. We will still be able to evaluate how users manage to find the fixed location, or their friend, as they will be told. It follows from HFR07 and HFR08, that the aWearable device outputs feedback based on the fixed location.

For requirement HFR07 we have only implemented the display and the light. The vibration module was as mentioned, out of stock. Two components are one more than enough to create an application with the toolkit and therefore the sound module was down prioritized. For requirement HFR08 we implemented behavior for the display and the light.

8 Evaluation

We have evaluated the prototype through two iterations of user evaluations. The same evaluation method was used during both. We also conducted a final expert evaluation where future work was the main topic. In this chapter we will describe how the user evaluations were conducted and what results that came out of them. We will also present how the prototype was refined between user evaluations.

8.1 Evaluation method

We used the same evaluation method in both evaluations. Each evaluation took about 30 minutes. An overview of the evaluation method can be seen in table 8.1.

Steps	**Evaluation Methods and Tools**	**Details**
Use the web application to create an application	Observation Usability Test	Let users look at the web application and create an application.
Use the created application	Observation Usability Test	Test- persons use the application to find Maria.
Answer a questionnaire	Questionnaire	Test- persons answers a questionnaire about their experience of creating and using the application.
Answer questions	Interview	Test- persons answer some predefined questions and some follow- up questions from the observations and the questionnaire.

Table 8.1: Evaluation Method

The evaluation consisted of four parts; using the web application to create an application, using the application to find Maria, answer a questionnaire and an interview. Maria had the role as test-leader, while Ingrid was observing. The test persons got a sheet of paper with an introduction to aWearable and some steps we wanted them to go through. We tried to make the tasks generic to test how much they understood of the web application without our instructions. Guidelines for the evaluation, task description and questions asked can be found in the Appendix.

The first task was to sign up. Second, we wanted them to explore the page and tell us what they saw. Third, was to create an application, with a few instructions on which parts they should use and who they wanted to be aware of. When they were finished creating the application we asked them what they thought the application would do. During the evaluations we used Silverback [5], which is an application for usability testing. The Silverback films the screen and highlights where the user clicks, while also filming the person in front of the screen.

Next step was to use the application they just made and test it out in the field. They were told to use the application to find Maria, who was hiding at the location we had hard-coded in the source code. Ingrid was following the person and observing how they used the aWearable. We also attached a GoPro camera [8] to the test- person's chest, filming the aWearable device and recording sound so we could analyze it later.

After the field-testing the test persons answered a questionnaire about their technical background, and how they found the creation of the application and the use of the aWearable device. At the end we had an interview with some prepared questions and some follow- up questions from the observations and the questionnaire. We used Silverback also when doing the interview so we could go through it later and analyze it.

In both evaluations we tested people with technical experience and people without technical experience. The technical experienced users are all 4th or 5th grade informatics students with programming experience. Some of them have experience with Arduino as well. The non-technical users are students without any programming experience and lower interest for technology. We want to compare the technical users with the non-technical users to see if there are any differences in achievement when it comes to creating and using the application, and how much they enjoy doing it.

8.2 First Evaluation

Here we will list the evaluation goals, the evaluation results and have a discussion about the answers to the evaluation goals. We will end with the refinements based on the results. In the first evaluation we tested 5 people. We started with three technical users to see how well they managed to use the toolkit. Since they all got through the test without any serious problems, we continued testing with two non-technical users. The non-technical users were both economics students.

8.2.1 Evaluation Goal

The goal of this evaluation was to answer the questions below:

- Are users able to customize a location-aware application by using the web application?
- Are users able to use the aWearable application they create?
- Are there any differences between the technical users and the non- technical users when it comes to achievement of creating and using the application?
- Are there any differences between the technical users and the non- technical users when it comes to how fun they thought it was to create and use the application?
- Does the user get enough information to find their friends?
- Which components and information is needed to find their friends?

8.2.2 Evaluation Results

The results we found from the evaluations are from observations of the participants creating the application, observation of the participants using the application and a questionnaire and interview at the end.

Creating the application

Here we will list the problems users had when creating the application, the reasons for the problems and suggestions for possible changes.

1. **First problem was with the creation of user accounts. At the signup page they thought that they could choose to sign up with Facebook or by entering the aWearable ID. They spent some time figuring out they had to do both. Interesting enough only the technical users made this assumption, while the non-technical users were more careful and took more time reading the text**

   Reasons:

   (a) The text does not explain it well enough

Possible changes:

(a) Write a better explanation that the user must sign in with Facebook and enter the aWearable ID.

(b) Add red stars to indicate that the fields are obligatory.

2. **Two technical users and a non-technical users had some trouble finding where the aWearable ID was written. They all found it eventually without help, but needed some time. Above the field where they enter the aWearable ID it is explained that they can find it at the device, but the participants did not take their time reading the text.**

   Reasons:

   (a) The participants might feel stressed by the observation setting, feeling that they must hurry up.

   (b) One of them did not understand immediately that the device on the table was part of the toolkit.

   Possible changes:

   (a) Emphasize more clearly that they should take as much time as they want.

   (b) We can also add a picture of the aWearable device pointing to the area where the ID is written.

3. **One of the technical-users did not understand that the parts were draggable.**

   Reasons:

   (a) The participant explained that she hovered the mouse over the parts, but there was no hand-icon indicating that it was click-able.

   Possible changes:

   (a) Make the icon look like a hand when hovering over the parts.

4. **When creating the application, two non-technical users and a technical user did not add Maria as friend.**

   Reasons:

   (a) The test does not clearly state that they should add a friend and they might think that this is already done.

(b) A couple of them also said they focused on the customizing of the application and did not really see the friend part.

Possible changes:

(a) Change the text in the test.

(b) Remove some text from the friend part to make the design cleaner and the important parts easier to see.

(c) Change the button from saying 'Connect' to saying 'Add friends'.

5. **One of the technical users tried to upload the application before choosing parts and their behavior, she had however plugged the physical parts to the aWearable device. When she uploaded the application, and got an error message saying that she had to choose parts, she did not understand why, as she had already plugged the parts to the aWearable.**

   Reasons:

   (a) The message was not explained well enough and does not make it clear that it is the software part, not the hardware part that is not finished.

   Possible changes:

   (a) Change the text to include which step they should go back to.

6. **One of the techincal users expressed concern when reading the sentence "NB! Make sure you have plugged the parts correct before clicking the button!"**

   Reasons:

   (a) The user was afraid of what might happen if he had done it wrong.

   Possible changes:

   (a) Explain what would happen if they had done it wrong or remove this sentence. In a later version it would hopefully not be critical to plug the parts before connecting.

7. **A technical user was waiting for the GPS light on the device to start blinking before it was connected to the computer.**

Reasons:

(a) Under the picture of the aWearable it is explained that the GPS light will blink when GPS signals are received.

Possible changes:

(a) Remove the explanation.

(b) Move the explanation to another place.

8. **A technical user found it strange that 'In-range' under 'Privacy Settings' is presented as a choice, when it is not possible to change it.**

   Reasons:

   (a) It looks like it should be possible to untick.

   Possible changes:

   (a) Present it in another way.

Other problems:
We have listed other problems that might have affected the results, which are not directly connected to the toolkit.

- One participant felt insecure about reading in English.
- Participants rushed through the test, afraid of using too much time and therefore not reading all the instructions in the web application.
- Some participants were afraid of breaking the device.
- The formulation in the text was not good enough.
- Problems with the GPS signal.

In one of the tests after creating the application we did not get any GPS signals and had to finish it two days later. The test person therefore created the application twice. The first time she was a bit insecure when creating the application, but the second time she felt confident and created the application and added friends without any problems. This was a good indication that the web application is easy to learn.

Using the application

The participants used the application just created to find Maria. She was hiding about 80 meters away. With the user's approval we have added a picture from the user- evaluation in figure 8.1. The participants all started by looking at the display and figure out where North, South, East and West were. They understood which direction to go, but commented that in an unfamiliar city they would not know where North would be. They kept walking while looking at the display and looking up, when the distance was getting smaller they looked more around to see if they could spot Maria. Some of the participants chose to use coordinates as well as direction and distance, but as this was a fixed location in the prototype it was not much of use. The light was very hard to see out in the daylight and the users did not look at it.

A couple of times the GPS was not working correctly, the display showed a distance at 20 meters then suddenly a distance of 100 meters in another direction. It is uncertain why this happened, but the GPS can be affected by weather and clouds, as mentioned in the 'Prototype Limitations', section, 7.4. One of the evaluations was conducted at the NTNU campus where there are high buildings making it difficult to get correct GPS data. This made the participants uncertain, but they kept following the direction displayed at the device. When the GPS was working correct, the participants had no trouble finding Maria.

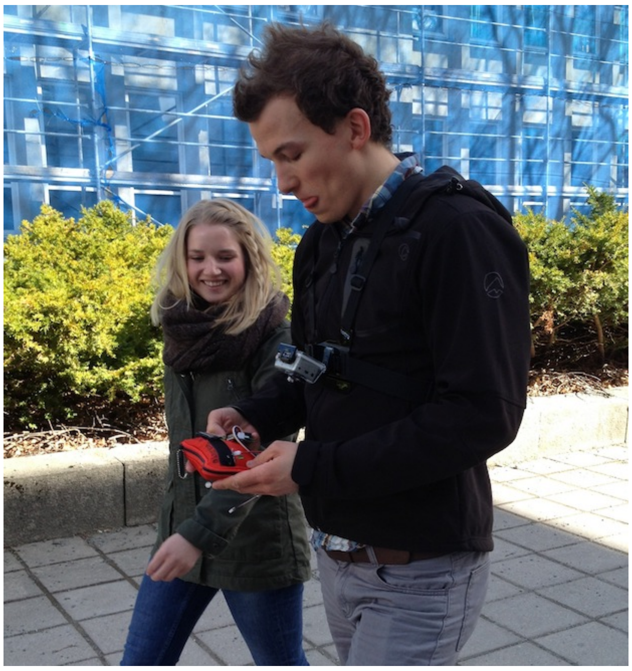


Figure 8.1: A user testing the application

Questionnaire and Interview

After creating and using the application, the participants answered a questionnaire followed by an interview.

From the questionnaire we found that the non-technical users thought it was easier to create the application and use the application than the technical users, even though the difference was small. The users said it was easy to make because they could just follow the instructions and they did not feel like they had to learn anything to use it. They all found the plugging of the parts easy as they could see from the picture how to do it. Some of the users that did not add a friend said that the drag and drop part drew the attention away from the friends-part.

The users found it easy to find Maria by using direction and distance, but they missed a compass, map or arrows to help show the direction. In and unknown city it can be hard to know where North and South is. In this setting, the coordinates did not help the participants, but in settings where exact location is needed, they can be used together with smartphone applications to find locations on a map. Users would also like to use the vibration and sound components, as it might be easier notice than the light.

The participants said they would have used the application in crowded places like concerts or festivals and unknown places to find friends. They also said they would use it in the mountains if someone was missing, when they are hunting or when they are walking home in the night as a sense of security. One user also said it would be cool to get notified that a friend was nearby, in a place where you thought you did not know anyone.

When it comes to how fun users thought it was to create and use the application, they all answered with the highest score. They liked that it was something they made themselves and felt a sense of ownership and also a sense of achievement when they found Maria. They all scored high on the question if they would like to use aWearable, but would prefer it to be smaller. They suggested that it could be worn around the wrist, as a necklace, attached to the jacket or even as a ring.

Most users liked the simplicity of the toolkit, but one of the technical users felt it would be more attractive if it had more features. For further development they had the following suggestions; possibility to turn the device on and off to choose when others can see them, get the name of friends within range, see where they have been on a map in retrospect, get the address of a friends location, receive Twitter feeds on the display, have a button to scroll through information on the display, and to get tagged on Facebook when friends meet at a certain place.

The test users were not particularly concerned with privacy settings. However, they all liked the idea of having different privacy setting on different friends, when we mentioned it to them. That way they could chose to share more location information with close friends and family.

8.2.3 Discussion

From the observations, questionnaire and the interview we tried to answer the questions from the evaluation goal in section 8.2.1.

All users managed to create an application by using the toolkit, and understood how to use application they created. One participant suggested that it was so easy that children could do it. There was a small difference between the technical users and non-technical users, when it comes to the achievement level and level of enjoyment of creating, and using, an application.

The non-technical users scored a little higher on how easy they thought it was. A reason might be that technical users are more critical to systems and know what to look for. From the observations we also noticed that the technical users raised more questions. Another impacting factor could be that the non-technical users were more nervous about failing, and therefore were more pleased when mastered the given tasks. In general, these answers tell us that all users found the web application, and the aWearable device, easy to use. We feel we have achieved and important goal. Considering that there were five test users, these results are not enough to ascertain that this would always be the case.

The users only needed the display with distance and direction to find Maria, and commented that they would benefit from having a compass. Some also commented that it would be enough with distance to find a friend, as you would see when the distance increased and decreased. Those that chose coordinates as well did not look at it, and said that they would not be able to find anyone based on them. To use the coordinates one must probably use a smartphone app that can show the location on a map when entering the coordinates. Users did not look at the lights as they got all they needed from the display. When lights, vibration or sound is used together with the display, they will notify the user when a person is within range. The user can than look at the display to get more information. In the evaluation, the display started to present information from the start, as a friend was already in range, and we did not get to test this further.

From the questionnaire and interview, both the technical-users and the non-technical users said that it was fun to create the application and to use it. How this would evolve over a longer period is not certain. For that a longer user evaluation with all the available components would have been necessary.

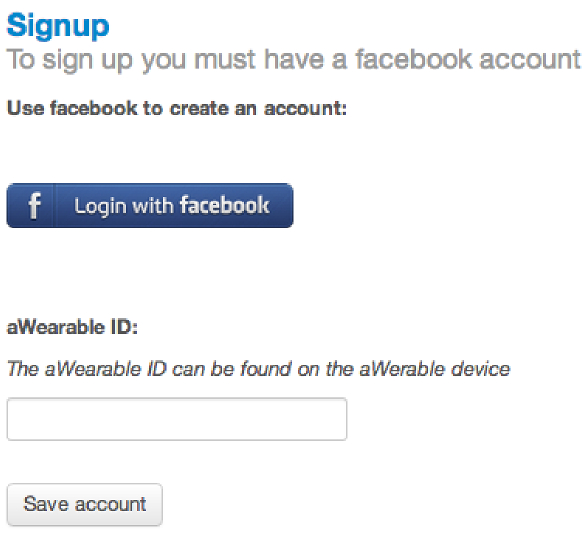
8.2.4 Prototype Refinement

After testing five people and seeing where they had trouble, we had some suggestions for changes.

Web Application:

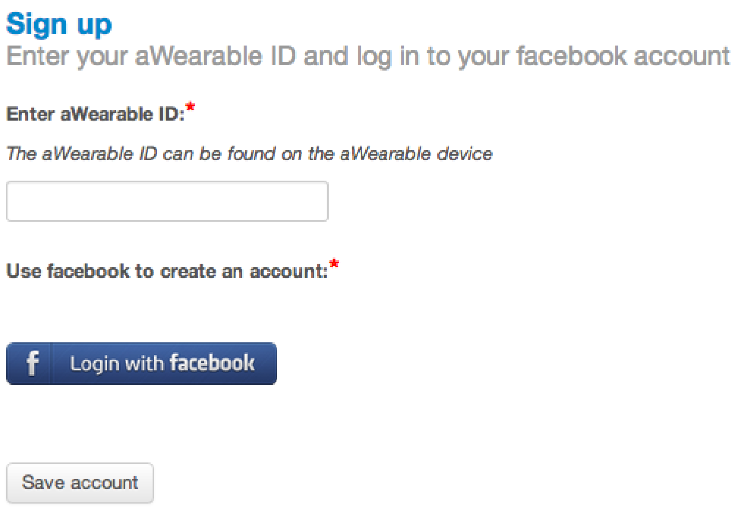
Problem 1- Sign-up

- At the sign-up page we changed the text to clearly state that they must log in with Facebook and enter the aWearable ID.
- We changed the order, so the aWearable ID and the Facebook login switched places.
- We added red stars to demonstrate that both steps are required


Signup
To sign up you must have a facebook account
Use facebook to create an account:

aWearable ID:
The aWearable ID can be found on the aWerable device

Figure 8.2: Sign-up: Before refinement


Sign up

Enter your aWearable ID and log in to your facebook account

Enter aWearable ID:*

The aWearable ID can be found on the aWearable device

Use facebook to create an account:*

Figure 8.3: Sign-up: After refinement

Problem 2- aWearable ID

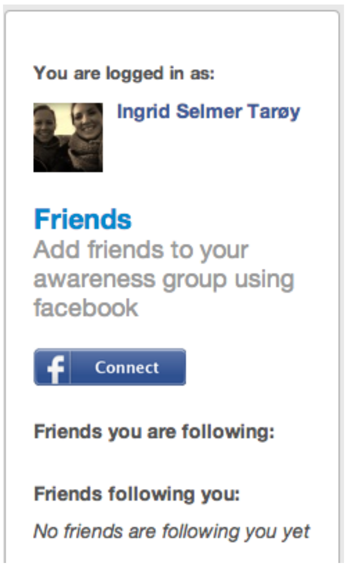
- By implementing the changes in 'Problem 1' we are hoping that the users will see and read the text that is explaining where the ID can be found.

Problem 3: Drag and drop

- To make the parts look clickable we changed the cursor to a hand when hovering over the parts.
- We also added this feature to the Facebook buttons.

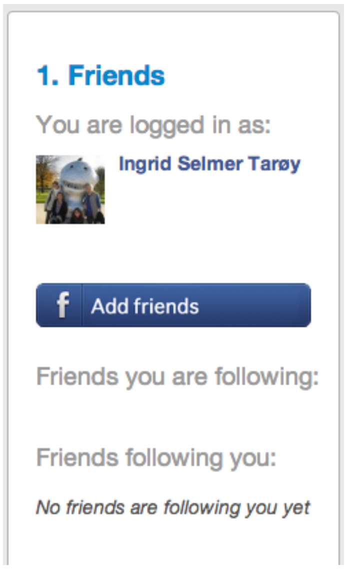
Problem 4: Add Friends

- To make the friends- part easier to spot we removed some text to highlight what is important.
- We put the headline saying 'Friends' on top in the left column.
- We changed the 'Connect' button to say 'Add friends'
- Also we added numbers to 'Friends' and 'Privacy Settings' so it looks like this is a part of creating the application.
- Last, we changed the text in the user test, to explain the task better.


You are logged in as:
Ingrid Selmer Tarøy
Friends
Add friends to your awareness group using facebook

Friends you are following:
Friends following you:
No friends are following you yet

(a) Friends: Before refinement


1. Friends
You are logged in as:
Ingrid Selmer Tarøy

Friends you are following:
Friends following you:
No friends are following you yet

(b) Friends: After refinement

Figure 8.4: Add Friends: Before and after refinement

Problem 5: Error Message

- First, we changed the error messages to say what step to go back to.
- We also changed the error-message box to be red, so it will be easier for the user to see when something is wrong.

Problem 6: "NB! Make sure you have plugged the parts correct before clicking the button!"

- We removed this sentence not to scare the users unnecessary. None of the users has skipped the plugging of parts or plugged them wrong.

Problem 7: "The GPS will start to blink when GPS signals are retrieved"

- We removed this sentence and put it in the 'How-To' page

Problem 8: In-range as a radio button

- After a discussion, we decided to leave this as it is, as it indicates that in-range is always allowed.

aWearable Device:

- We added the user's coordinates to provide something to compare the friend's coordinates with.

- We added a compass so users can check what direction that is North, South, West and East.

8.3 Second Evaluation

After doing the changes described in section 8.2.4 we had another round of user evaluations. Here we will list the evaluation goals, the evaluation results, discuss the differences from the first evaluation, as well as answers to the goals. In the second evaluation we tested two technical users and three non-technical users. Among the non-technical users there was a teacher student, a construction engineer student and a political scientist student.

8.3.1 Evaluation Goal

In the second evaluation we were more interested in the users' thoughts about further development and also if we saw any improvements after the refinements we had done. The evaluation goal was to get answers to the questions below:

- Do the users see any benefits or drawbacks from using aWearable versus integrating it in a smartphone application?
- Would the user prefer to get a ready- made application or do they like to customize it themselves?
- Do the users want the toolkit to have more functionality and choices or do they prefer it to be simple?
- Which components would the users prefer to use?
- Do the users want to see which friend they are getting information from and would they let their own name be revealed to friends?

8.3.2 Evaluation Results

The results we found from the evaluations are from observations of the participants creating the application, observation of the participants using the application and a questionnaire and interview at the end.

Create the application

1. **A technical user and a non-technical user thought that they could choose to sign up with the aWearable ID or with Facebook.**

Reasons:

(a) The sign-up looks different than users are used to.

(b) Users are skeptical to use Facebook to sign in.

Possible changes:

(a) Add numbers 1 and 2 in front of the two steps.

(b) Make the text more visible.

(c) Add error- messages when a user tries to save an account without logging into Facebook.

2. **One of the technical users thinks the aWearable ID is a username he should invent.**

   Reasons:

   (a) The text explaining what aWearable ID is not highlighted good enough.

   Possible changes:

   (a) There should be a validation on what the user writes in the field.

3. **One of the technical users does not plug the parts to the correct cable.**

   Reasons:

   (a) The user does not follow the steps or looks at the picture when plugging and says he usually does not read manuals, he rather want to try and fail.

   (b) User also says he thought that each cable matched one of the parts.

   Possible changes:

   (a) Make sure to give enough feedback so it is not possible to do anything wrong.

   (b) Color code, or add numbers to the components and correct wires. That way, users could attach components correctly without having to check the web application.

Using the application
Participants were more secure and spent less time on figuring out the directions when they had the compass. None of the participants had any trouble using the device, and they all found the correct place even though the high buildings made the GPS act up at times. Participants paid closer attention to the display when the distance was getting smaller. None of them looked at the light, but they understood what it did.

Questionnaire and Interview
From the questionnaire we found that one of the non-technical users scored lower than the others when it came to if she wanted to use aWearable, how fun it was to create and how fun it was to use. However, she scored just as high as the rest when it came to how easy she felt it was to create and use the application. The participants answered quite different on which components they wanted to use, the non-technical chose either light or display and the technical users chose either all components or display and sound.

From the interview we found that the non-technical users prefer the toolkit to be simple with fewer choices, while the technical users would prefer more choices and functionality beyond location-information as that would make it more attractive and fun to use. The technical-users would also like to program the Arduino themselves, and use the toolkit as a framework.

When asked if they would prefer to get a ready-made device instead of making it themselves the non-technical users answered yes and the technical users answered no. The non-technical users said they would like a ready-made device where they could choose to turn options off and on. They felt that the cables were intimidating and would like it to be wireless. The technical-users however, thought that the fun part was to create applications themselves. They thought it was more special and original to make it themselves than have another ready-made device.

Suggestions for new functionality was the possibility to turn the device on and off, use it as a step-counter, audio-voice to tell you where to go, a map, and to see how many friends are close. Participants said they would use it at places with bad phone reception, and rescue work in the mountain. One of the users would have used it more for fun and play, for digital treasure hunts and pub-crawls. There was also a suggestion that the device could connect to the smartphone, so that parents could have the phone and children have the device and the parents could see that their children are in proximity.

As of now, the device does not show which friend that is near. In the evaluation only one friend was added, so they knew who it was. When asked, all participants said that they would like to see their friends' names, and also let friends see their names as well. They would also liked the possibility to turn the device off for some friends, or at certain times, when they do not want to be disturbed. One of the participants said he would rather leave the device at home when he did not want to share his location. The same participant also said he would be more careful if it was a smartphone application, as he carries the phone around all the time.

Two of the non-technical users expressed that they rather make a phone call to find their friend, than use the aWearable device. Still, they did see some benefits of having a separate device. For example, the device does not need Internet, it can be left behind when they do not want to use it, and it would be cheaper than buying a smartphone. There is also the issue of using Facebook to login. Multiple users were skeptical to do this. Many applications today want you to login with Facebook to get access to your Facebook- friends. The participants were insecure about what kind of information that would be accessed.

8.3.3 Discussion

After the prototype refinements we did based on the first evaluation, the participants got through creating the application with less uncertainty and fewer problems. Even though there were still some problems with signing up, all participants understood that they were supposed to add friends to follow. One participant got an error message when trying to upload the application before choosing parts and behavior and understood immediately what she had done wrong and corrected it.

The answers to the evaluation goals were collected mainly from the interviews. Non-technical users preferred a simple ready-made device where they could turn on and off settings and options. Technical users, on the other hand, would like more functionality to customize and program the applications themselves. Non-technical users saw less of a point of using the device instead of a smartphone. The benefits they did see, was that the device could be left at home when they did not want to use it, and they could therefore have different privacy settings than in a location-aware smartphone application. They wanted to be able to see which friends they were getting location-information from. They were also comfortable with sharing their names, but would like the opportunity to hide it from certain friends and at certain times.

Participants answered quite different on which components they wanted to use. The non-technical chose either light or display and the technical users chose either all components or display and sound. It is not much to conclude from their answers, and it would depend on the situation. The non-technical users chose fewer components than the technical users. This might be because the non-technical users wanted a simpler device.

We see a difference in the answers between the non-technical and the technical users in what functionality they want. In further development we should either find a balance that suits both groups, or focus on one of the user types. The goal is that the device will be a lot smaller, and might result in a device where all the parts are embedded, and where users choose what it should do in the same way as now. If we add functionality and still make the deciding of behavior easy, it might suit both technical and non-technical users.

8.4 Expert Evaluation

After the two user evaluations, we had an expert evaluation with a professor from IDI. The evaluator is an expert in the field of social embodied interaction and social computing and has also some knowledge about Arduino. The focus of the evaluation was to get his ideas on further development and the potential of the toolkit.

His first thought when we showed him the aWearable toolkit, was that it was innovative and a new concept. He suggested that it could be used to connect to places as well as friends. Then users could add interests when creating the application, and the device could work as a tourist guide and guide them to exciting places and attractions. The device could hang on a backpack or a purse, and make a sound when something or someone was close. He also suggested adding a solar panel to the device that could charge the battery. This would also give it more benefits from a smartphone.

The evaluator pointed out that awareness is about more than knowing a friend's location. It should be possible to signal to a friend that you are available and interested in communication. It can be a button or a small teddy bear that users can squeeze, and on the other side the friend receives a haptic feedback. A keyword in further development is to make it possible to send wireless- updates to the Arduino.

To get more ideas for what the toolkit can provide, the evaluator suggested that we analyze the functionality on Facebook and other social platforms and map this functionality to physical gestures. Functionality can be poking, create groups, create events, send messages et cetera. We could use RFID or Bluetooth to create a group or event with people that are sitting together. The toolkit could also be made as a security packet for elders, where the local communities customize applications for individuals. Today, there is technology for elders including sensors that are worn on the body and sense if someone falls. The toolkit could be customized for such functionality or help elders to socialize and feel aware of others.

The evaluator also came with ideas to how we can change the physical design. One idea was to keep everything in the bag and the user could open it whenever, replug the desired parts, close the bag, and it would be another application. Arduino boards come in so small sizes, that it can be integrated in clothes. That would make it easier to carry. When it comes to the usability the evaluator thought it was easy to understand the steps in the web application.

9 | Conclusion

9.1 Summary

In this thesis we have investigated how to support end-user development of embodied location-aware applications. We have created a prototype toolkit named aWearable and looked at what functionality that should be provided for end-users to be able to customize their own applications. We have also investigated how feedback should be presented to promote location-awareness.

By looking at theoretical background and related work on embodied interaction and location-aware applications we defined a set of scenarios that again helped us identify what the toolkit should provide. From the second research study on toolkits aimed at end-users, we decided how to design the toolkit to make it as user-friendly as possible. There are three aspects that were the foundation of the toolkit; manage groups, customize applications and use the application. These aspects were broken down to a set of requirements. After the implementation we had two user- evaluations and an expert evaluation that helped answer the research questions.

9.2 RQ Discussion

We will answer the research questions based on the results we found in the evaluations, from theoretical background and related work. The main research question covers the whole project we have been working on, while the sub questions help answer parts of the main question. The first sub question is about the creation of applications, while the second sub question is about the use of the application.

9.2.1 Main RQ

How can we support end-user customization of embodied location- aware applications?

The prototype in itself, is an answer to this research question. We can support end-user customization of embodied location-awareness application with the aWearable toolkit. The toolkit consists of two connected parts, a web interface and a physical device. Users customize the behavior of application by using the web interface and plugs the chosen physical parts onto the device. When the physical parts are plugged and the behavior of the application uploaded to the physical device, the physical device is ready for use.

From the user-evaluation we have seen that users are able to create applications by using the web interface and that they are able to use the created application. Both technical users and non-technical users scored high on how easy they found the customization and use of the application. Almost all users scored with the highest score on how fun they found the creation and use of the application, where the non-technical users had a slightly lower total- score than the technical-users. This also corresponds with the technical interest the users said they had.

9.2.2 Sub RQ 1

What functionality must be provided to facilitate end-users in customizing embodied location-awareness applications?

This question seeks to answer if the functionality we have provided in the toolkit supports the end-users in creating their own location-aware applications. We found characteristics of location- aware applications by conducting a related research study. We found required characteristics to be group structure, proximity range, distance, directions, coordinates, who you are aware of and privacy settings. This functionality was included in our toolkit. There are also certain requirements for end-user development that we strived to fulfill, like error prevention, usability and requiring no user prerequisites. We will discuss why we think this functionality is important when answering Sub RQ 1.

Group creation
A group structure is necessary for users to decide whom to be aware of. In the aWearable toolkit, every user can add friends they want to receive location- information from. The friends can choose to accept or decline requests. This resulted in a 'following-group' and 'followers-group', where the following- group is friends you receive location-information from, and followers-group are friends you share your own location-information with. Users can choose how much location- information they want to share with their followers.

Users seemed to understand the 'following- followers' metaphor and in the second evaluation, after some refinements of the layout, all users understood that they had to add friends. All users would like the possibility of setting privacy settings on each user. This is functionality that a location-aware application should provide and would be something to consider for further work.

Location- Information
We have added different types of location-information as options in the toolkit. The options depend on the type of actuator that the user has chosen. The display offers more precise location-information than the other actuator, as it can display distance to, direction to and coordinates of other users. The three other actuators; light, sound and vibration, has two options where it is possible to choose one. The actuators can either notify the user when someone is in their proximity- range or it can act different when a user is getting closer. With 'act different' we mean the light will blink faster, sound will buzz more often and vibrator will vibrate faster. All of these location- information options are reflected in the privacy settings as well. Users suggested maps or addresses as additional location-information and audio voice that could direct them to friends.

The proximity range is also functionality in the toolkit where users choose a range from a drop-down list. The range is what triggers the awareness and sharing of location between users and is therefore an important setting.

End-user development
Usability is important and crucial for the end-users to be able to create applications in the first place. Much of our focus was to make a web interface that would be intuitive and simple, and to lead the user through the creation of applications without any complications. End- user development puts demands on structure and it should be hard to make errors. In the web interface of the toolkit we added steps for the user to go through and error messages explaining what they should do if they had skipped some steps when trying to upload the application.

Since the toolkit consist of a software part and a hardware part, it is important that there is a correlation between the two. To solve this, we added a picture of the physical device in the web interface, where user must drag the chosen parts onto the device, illustrating that user must physically plug the parts as well. It is important that users understand what the application they have created will do before using it. We asked about this in the users evaluations, and the majority described the applications correctly. A few users were a bit uncertain, but understood it once they started using the application.

To sum up, we saw from the evaluation that functionality provided in the toolkit was enough to supported users in customizing their own embodied location-aware applications. Groups and location-information are necessary for location-awareness, while usability and structures are crucial to support end-user development.

9.2.3 Sub RQ 2

How should feedback be presented by the end-user-customized application to facilitate location- awareness?

With this question we wanted to know if the aWearable applications presents feedback in a way that promotes location-awareness. The toolkit lets the user customize different applications by using one or more of the actuators; lights, sound, vibration, display, and by giving them specific behavior. There is a need for different actuators in different settings. There are multiple possible different combinations, but in the prototype only display and lights were implemented so we only got to test these actuators.

The display can present distance, direction and coordinates. Most of the users chose distance and direction, and these two options combined seemed to be the most effective for finding another user. Some users added coordinates as well, but did not look at them, as it is hard to get any information from them. The coordinates were an option we added with respect to rescue work, where absolute direction can be crucial. None of the users chose only direction or distance. However, one test- user commented that it would be possible to find a person with just distance, as the distance decreases and know you are walking in the right direction. The same would apply for direction, where you can follow the direction until the person is found. The toolkit is however, not only meant for finding people, it can also indicate that someone is close, or promote awareness like in the roommates' scenario in section 4.2.

The light has two options where user must choose one of them. It can either light up when a person is in a user's proximity, or it can blink faster if the person is getting closer. To let the light blink faster when a person is approaching can say something about distance without using the display. In addition, is has the advantage of giving feedback in the periphery of a user's attention. In the evaluations, users chose approximately 50/50 between these two options, but as the light was used together with the display, no one looked at it. It was also hard to see in the daylight. Vibration and sound have the same two options as the light, and users said that they would like to use these as well. With these types of notifications, users could put the application in their pocket and still be able to hear or feel the notification. If the prototype is integrated into clothes in the future, there are other ways to give feedback that can facilitate location-awareness. This was seen in the SuperShoe, where vibrations were put in the shoe to indicate direction.

9.3 Reflection on our work

Some of the main work done has been the initial exploratory study of embodied social interaction and the subsequent research studies on embodied location-awareness applications and end-user toolkits. The topics were not familiar to us from before, and it took some time to get confident with the terms. The problem task was not clear when we started the project and the initial phases involved defining the problem.

Another large phase was the implementation of the prototype. We did not have any prior experience with Arduino prototyping or c++ programming. Our co-supervisor, Simone Mora, helped us get started. We also got a lot of help from libraries like TinyGPS and the Noduino framework, which we used to manage connection between the web interface and the Arduino device. The Noduino framework is complex as it incorporates so many different technologies, but fortunately we did not have to understand every aspect of it to get it to work, as we needed. Connection between the web application and the Arduino was provided 'out of the box', while it took more to understand how they communicate. We figured it out by trying and failing. The implementation phase ended up being larger than we planned for when the project was started. It has been both challenging and fun to work with the integration of different tools and technologies. Since we were unfamiliar with most of the technologies, we have learned a lot during the project. At first it all seemed a little frightening and incomprehensible based on our competences, but in retrospect we are glad we jumped into it as we got great new experiences from it.

Our limited programming experience has also led to limitations of the prototype. It took us longer to implement functionality that might seem like an easy task to an experienced programmer. We only developed one aWearable device, which allowed us to disregard communication between devices. In reality, the aWearable device must be able to communicate with one or more other devices at once, but instead we hardcoded one location. Although this is a big drawback, we were able to "fake" the functionality for the user evaluations, and users were able to customize and use the application regardless. We had them add one of us as their aWearable friend. The person they added went outside and hid at the hard coded location, and the users were asked to find us. That way we made it seem like the location was connected to one of us, when in reality the location was fixed. Since friends' management is not implemented in the prototype it will only show the user that some friend is within range, not who the friend is. Due to this simplification, we did not have to display who the location-information was coming from, as users were only following one person. From the Hummingbird application, we saw a problem of displaying the names of users, because of the limited display size. We avoided this problem in the prototype, but it would have to be solved if the toolkit would become reality.

Even though we were able to test the prototype, the setting was not sufficient to give users a realistic context of use for the aWearable toolkit. They were tasked to find a person and it would seem that this was the purpose of the application. We wanted the application to promote location-awareness and not just be a tool to find someone, which we were unable to test. The ideal evaluation environment would have been to have a set of three or more aWearable devices and hand them out to a group of friends so they could use them for several days. That way we could also make them try out different combinations of the Grove components.

Only the light- and the display- components were implemented in the prototype. The vibration component was out of stock and the sound component made a lot of noise. We had to ask users to create an application that had light and display. As a consequence, we were not able to fully evaluate Sub RQ 2. It was not possible to customize different applications, other that choosing the behavior of components, and we did not get to evaluate enough feedback components to make a conclusion on how feedback should be presented. In addition, we have no evidence that the toolkit can be used to develop a relevant set of applications. What we can say is that aWearable has the potential to provide location-awareness. The toolkit could be used to develop applications similar to the Hummingbird, the Good Night Lamp and the Phidget Eyes, as it uses the same actuators. From the evaluations of these related applications, it was stated that users felt a sense of awareness.

One could also argue that you might as well use a smartphone application instead of aWearable. The application could be running in the users pocket and notify the user when friends are nearby. The smartphone also has; light, sound, vibration and a display, and they are all much more sophisticated than the Grove components. The only thing that would be missing is tinkering with hardware. However, when your smartphone vibrates, or makes sound, in your pocket it could just as well be an SMS or a notification from another application. Users would have to pick up the phone to check it. We believe users would want to check the phone, as a habit, even if the vibration and sound were different from any other, but this is just an assumption based on our own experiences. If the aWearable toolkit were to become reality a main focus would be to make the device smaller and more appealing. It could be worn as jewelry, attached to a keychain, or woven into clothes. Then it would probably require less effort to check the aWearable than the smartphone.

9.4 Further Work

Several improvements should be considered for further development of the aWearable toolkit. First, size matters and the aWearable device should be made smaller and more attractive to get users' attention. The device should be possible to hang on a necklace, a keychain, or wherever the user would want it. Second, the device should be made wireless so it can communicate with the web interface without the USB cable. Parts of the web interface could be adapted to a smartphone application, where users could, for example, manage friends while on the move. It would also be possible to use the GPS and map services integrated in the smartphone, and the smartphone screen could be used when the user desired, for example if the user needs map-directions to get to a location.

Battery capacity is crucial for aWearable to become a success. It should provide better capacity than smartphones. The smartphone has more functionality that shares the battery source. Users play games, surfs the Internet, make calls, checks email and so on. The aWearable is much more specific and does not need nearly as much power. Solar technology has come a long way and it would be interesting to see if a solar panel could be used to supply aWearable with power.

It is nearly impossible to satisfy all possible users. While novice and less technical users expressed that they preferred a clean and simple toolkit, the expert users expressed that they would like more functionality. A possible solution would be to make it possible to extend parts of the toolkit, like a 'show more functionality'-setting. That way the toolkit could provide more functionality to experienced users. It could integrate more social network functionality, like to present twitter-updates from friends in range, or twitter-updates that are connected to where the user is located.

The toolkit could also be extended to include places as well as people. Users could add places they are interested in, and the aWearable application could be used to show directions to these places. It could also be possible to include activity in the awareness-model. Users could be notified about friends in their range that are doing the same activity as them, for example friends that are also out shopping in that area. Users could communicate to nearby friends, that they are interested in making contact or search for particular friends in the area.

Bibliography

[1] Arduino. http://www.arduino.cc/. Accessed: 15/01/2013.

[2] Arduino uno board. http://www.arduino.cc/en/Main/arduinoBoardUno. Accessed: 22/05/2013.

[3] Facebook developers. https://developers.facebook.com/. Accessed: 20/04/2013.

[4] About scratch. http://scratch.mit.edu/about/, 2007. Accessed: 14/05/2013.

[5] Silverback 2.0. http://silverbackapp.com/, 2008-2010. Accessed: 15/05/2013.

[6] Labview interface for arduino. https://decibel.ni.com/content/groups/labview-interface-for-arduino, 2011. Accessed: 14/05/2013.

[7] littlebits. http://www.littlebits.cc//, 2011 - 2012. Accessed: 11/02/2013.

[8] Gopro. http://gopro.com/, 2013. Accessed: 15/05/2013.

[9] Ninja blocks. http://ninjablocks.com/, 2013. Accessed: 11/02/2013.

[10] Sen.se. http://open.sen.se/dev/, 2013. Accessed: 14/05/2013.

[11] Socket.io protocol. https://github.com/LearnBoost/socket.io-spec, 2013. Accessed: 14/05/2013.

[12] Farouk Belkadi, Eric Bonjour, Mauricio Camargo, Nadège Troussier, and Benoit Eynard. A situation model to support awareness in collaborative design. *International Journal of Human-Computer Studies*, 71(1):110 – 129, 2013.

[13] Daniel Cernea, Simone Mora, Alfredo Perez, Achim Ebert, Andreas Kerren, Monica Divitini, Didac Gil de La Iglesia, and Nuno Otero. Tangible and wearable user interfaces for supporting collaboration among emergency workers. In *Proceedings of the 18th international conference on Collaboration and Technology*, CRIWG'12, pages 192–199, Berlin, Heidelberg, 2012. Springer-Verlag.

[14] Citilab. Scratch for arduino. http://seaside.citilab.eu/scratch?_s=Avmlu7tI80sS0qAW&_k=5gsYojyRHg9efBHE. Accessed: 28/01/2013.

[15] Kaazing Corporation. Websocket.org. http://www.websocket.org/aboutwebsocket.html, 2013. Accessed: 14/05/2013.

[16] Alexandra Deschamps-Sonsino. The goodnight lamp. `http://goodnightlamp.com/`, 2012. Accessed: 28/01/2013.

[17] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.

[18] P. Dourish. *Where The Action Is: The Foundations Of Embodied Interaction*. Bradford books. Mit Press, 2001.

[19] Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8, 2004.

[20] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, CSCW '92, pages 107–114, New York, NY, USA, 1992. ACM.

[21] Paul Dourish and W.Keith Edwards. A tale of two toolkits: Relating infrastructure and use in flexible cscw toolkits. *Computer Supported Cooperative Work (CSCW)*, 9:33–51, 2000.

[22] M. Endler, A. Skyrme, D. Schuster, and T. Springer. Defining situated social context for pervasive social computing. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 519–524, 2011.

[23] Saul Greenberg. Collaborative physical user interfaces. Technical report, In, 2004.

[24] Saul Greenberg. Toolkits and interface creativity. *J Multimedia Tools & Applications*, 32:139–159, 2007.

[25] Saul Greenberg. Promoting creative design through toolkits. *Web Congress, Latin American*, 0:92–93, 2009.

[26] Saul Greenberg and Chester Fitchett. Phidgets: Easy development of physical interfaces through physical widgets, 2001.

[27] Mikal Hart. Softwareserial library. http://arduino.cc/en/Reference/SoftwareSerial. Accessed: 10/05/2013.

[28] Mikal Hart. Tinygps library. http://arduiniana.org/libraries/tinygps/. Accessed: 10/05/2013.

[29] Dhairya Dand Henry Holtzman. Supershoes : Facilitating urban rediscovery. 2013,.

[30] Lars Erik Holmquist, Jennica Falk, and Joakim Wigström. Supporting group collaboration with interpersonal awareness devices. *Personal and Ubiquitous Computing*, pages –1–1, 1999.

[31] Eva Hornecker. The role of physicality in tangible and embodied interactions. *interactions*, 18(2):19–23, March 2011.

[32] Eva Hornecker and Jacob Buur. Getting a grip on tangible interaction: a framework on physical space and social interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 437–446, New York, NY, USA, 2006. ACM.

[33] Inc Joyent. node.js. `http://nodejs.org/about/`. Accessed: 14/05/2013.

[34] Marije Kanis, Niall Winters, Stefan Agamanolis, Anna Gavin, Cian Cullinan, and Human Connectedness Group. Toward wearable social networking with iband. In *In CHI '05 - Human factors in computing systems*, pages 1521–1524. ACM Press, 2005.

[35] Sunyoung Kim, Julie A. Kientz, Shwetak N. Patel, and Gregory D. Abowd. Are you sleeping?: sharing portrayed sleeping status within a social network. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08, pages 619–628, New York, NY, USA, 2008. ACM.

[36] Jennifer Kreie, Timothy Paul Cronan, John Pendley, and Janet S. Renwick. Applications development by end-users: can quality be improved? *Decision Support Systems*, 29(2):143 – 152, 2000.

[37] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-User Development: An Emerging Paradigm. In Henry Lieberman, Fabio Paternò, and Volker Wulf, editors, *End User Development*, volume 9 of *Human-Computer Interaction Series*, chapter 1, pages 1–8. Springer Netherlands, 2006.

[38] Olivier Mèvel and Marc Chareyron. readiymate. `http://www.readiymate.com//`. Accessed: 11/02/2013.

[39] Sebastian Müller. Noduino. `http://semu.github.io/noduino/`, 2012. Accessed: 14/05/2013.

[40] Jakob Nielsen. Ten usability heuristics. `http://web.cs.wpi.edu/~kal/courses/hci/module6/neilsen10heuristics.htm`, 2005. Accessed: 30/04/2013.

[41] N. Nova, F. Girardin, G. Molinari, and P. Dillenbourg. The Underwhelming Effects of Automatic Location-Awareness on Collaboration in a Pervasive Game. In *Cooperative Systems Design: Seamless Integration of Artifacts and Conversations - Enhanced Concepts of Infrastructure for Communication*, pages 224–238, 2006.

[42] Fernando Olivera, Manuel García-Herranz, PabloA. Haya, and Pablo Llinás. Do not disturb: Physical interfaces for parallel peripheral interactions. In Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler, editors, *Human-Computer Interaction - INTERACT 2011*, volume 6947 of *Lecture Notes in Computer Science*, pages 479–486. Springer Berlin Heidelberg, 2011.

[43] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and*

Applications, WMCSA '94, pages 85–90, Washington, DC, USA, 1994. IEEE Computer Society.

[44] Albrecht Schmidt, Michael Beigl, and Hans w. Gellersen. There is more to context than location. *Computers and Graphics*, 23:893–901, 1998.

[45] Espen Larsen Segelvik and Morten Larsen Segelvik. Location awareness in ubicollab. Master's thesis, NTNU, 2001.

[46] Andrew Sleigh. Alexandra deschamps-sonsino: Making the good night lamp. `http://blog.makezine.com/2013/04/04/alexandra-deschamps-sonsino-making-the-good-night-lamp/`, 2013. Accessed: 08/05/2013.

[47] Mads Soegaard and Rikke Friis Dam, editors. *End-User Development*. The Interaction Design Foundation, Aarhus, Denmark, 2013.

[48] Seeed Studio. Introduction to grove. `http://www.seeedstudio.com/document/pdf/Introduction%20to%20Grove.pdf`, 2012. Accessed: 30/04/2013.

[49] Christian Thellufsen, Abbas Rajabifard, Stig Enemark, and Ian Williamson. Awareness as a foundation for developing effective spatial data infrastructures. *Land Use Policy*, 26(2):254 – 261, 2009.

[50] Philip van Allen. Netlab toolkit. `http://www.netlabtoolkit.org/overview/`, 2012. Accessed: 14/05/2013.

[51] Eric von Hippel. User toolkits for innovation. *Journal of Product Innovation Management*, July, 2001.

[52] Marc Weiser. The world is not a desktop. *interactions*, 1(1):7–8, January 1994.

[53] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999.

Appendices

A | Appendix

A.1 Guidelines for user Evaluation

Guidelines we followed when conducting the evaluation

1. Intro
   (a) Introduser deg selv
   (b) Beskriv hensikten med testen
   (c) Fortell deltakerene at de kan avbryte når de vil
   (d) Beskriv utstyret i rommet og begrensningene til prototypen
   (e) Lær bort hvordan man tenker høyt
   (f) Forklar at du ikke kan tilby hjelp under testen
   (g) Beskriv oppgaven og introduser produktet
   (h) Spør om det er noe de lurer på og kjør testen

   (i) Noter vær og vindforhold.

2. Testing av brukergrensesnittet
   - Del ut brukertesten og be dem lese introduksjonen
   - Start silverback
   - Gi den aWearable vesken, lukket med usb kablen koblet fra. (Batteriet?)
   - Start aWearable på start.html siden
   - Brukeren skal nå gjennomføre stegene i brukertesten (tenk høyt!)

   Observation:
   - Hvordan utforsker de applikasjonen? Hva klikker de på?

- Forstår de hva de skal gjøre?
- Er det noen gjennomgående misforståelser?
- Hvordan reagerer de når de åpner vesken? Synes de arduinoen ser skummel ut?
- Har de problemer med koble til komponentene? (inkl. usb og batteri)

Spørsmål til brukeren:

- Hvordan tror du at denne applikasjonen vil fungere?

3. Testing av applikasjonen som ble laget
    - Plassere ut et bilde/noe de skal finne for å se om de ser etter dette eller på applikasjonen.
    - Fest go-pro på brukeren.
    - Ta brukeren med ut på leting.

    Observasjon:

    - Klarer de å finne vennen?
    - Hvor er oppmerksomheten? (Ser de på displayet eller ser de rundt seg etter vennen)
    - Virker de fornøyde? Ser det ut som om de har det gøy?

    Spørsmål til brukeren:

    - Hva tror du at dette lyset betyr? (gps lyset)

4. Testskjema
    - Åpne siden med testskjema og la brukeren svare på dette i fred og ro.
    - Etterpå tar vi opp svarene for å diskutere dem sammen.

5. Inervju
    - Stille oppfølgingsspørsmål fra testskjena og observering og samle løse tråder.
    - Stille ferdiglagede spørsmål

6. Brukeren må skrive under på consent form!

7. Etterarbeid/ vurdering
   - Anvendbarhet ("Effectiveness") I hvilken grad lar oppgavene seg utføre med produktet? Dekker systemet de relevante funskjoner, og får brukerne til å bruke de?
   - Effektivitet ("Efficiency") Hvor effektivt lar oppgavene seg utføre? Krever kvantitative mål på hvor mye tid som går med til "data trouble" vs. tid til faktisk oppgave.
   - Subjektiv tilfredstillelse ("Satisfaction") Den opplevde brukskvalitet. Krever tester, intervjuer, feltstudier, spørreskjemaer etc. for å fastslå den enkelte brukers opplevelse av systemet. Viktig for aksept.

A.2 Task Description

The task descripten given to the users after the refinements from evaluation 1.

Introduksjon
Vi har laget en prototype som vi ønsker å teste. Prototypen er et program som lar brukeren lage sine egne bærbare/fysiske applikasjoner. Applikasjonen vil gi informasjon om lokasjonen til vennene dine. Hvordan man mottar denne lokasjonsinformasjonen og hvor mye informasjon man får velger man selv når man lager applikasjonen. Vi har valgt å kalle systemet aWearable, der vi spiller på wearable (bærbar) og awareness (oppmerksom på).

Du skal hjelpe oss med å teste om dette systemet er enkelt å bruke. Det betyr at vi ikke tester deg som bruker, men systemet sin funksjonalitet og brukervennelighet. Under testen vil vi kun observere, ikke hjelpe til. Du har alt av info nødvendig på listen over oppgaver og du kan trykke på de knappene du ønsker for å komme dit du vil. Du kan også velge å stanse testen når du vil.

Testen består av to deler; først en del der du skal lage applikasjonen, så en del der du skal bruke applikasjonen. Etter testen vil vi stille noen spørsmål om hvordan du opplevde systemet.

Vi setter pris på om du kan tenke høyt mens du bruker systemet. Et eksempel på å tenke høyt kan være "hvis jeg skal skal inn på gule sider trykker jeg på denne knappen...."

Brukertest

1. For å komme igang må du først registrere deg som bruker.

2. Utforsk siden og fortell oss hva du ser
3. Vi vil nå at du skal lage en Applikasjon:
   (a) For å bruke applikasjonen må du følge minst 1 venn. Følg Maria Møller eller Ingrid Tarøy.
   (b) Velg hvor mye lokasjons- informasjon du vil dele med de fremtidige føl-gerne dine.
   (c) Applikasjonen du nå skal lage skal ha display og lys. Du velger selv hva du vil at disse skal vise/gjøre. Du ønsker å vite nÃĕr vennene dine er mindre enn 200 meter fra deg. Fullfør denne applikasjonen.

Felt-testing
Bruk din selvlagde aWearable applikasjon til å finne vennen din

A.3 Questionnaire

The users answered a questionnaire, where they anwswered a number between 1 and 5.

1. Har du programmeringserfaring?
2. Har du erfaring med Arduino?
3. Hvor interessert er du i teknologi?
4. Jeg kunne tenke meg å bruke aWearable?
5. Hvor vanskelig synes du det var å lage en applikasjon?
6. Hvor vanskelig synes du det var å feste delene (display, lys, osv.) til den fysiske delen av systemet?
7. Synes du de forskjellige delene av systemet hang godt sammen?
8. Hvor vanskelig synes du det var å bruke applikasjoen du lagde?
9. Hvor vanskelig var det å finne vennen din ved hjelp av applikasjoen du lagde?
10. Hvor sikker følte du deg da du brukte systemet?
11. Hvor mye tror du at du må lære deg før du kan bruke systemet på egenhånd?
12. Hvor godt likte du å lage applikasjonen ved hjelp av systemet?
13. Hvor godt likte du å bruke applikasjonen som du lagde?
14. Hvor komfortabel er du med å dele lokasjonen din?
15. Hvilke deler ville du helst ha brukt?
16. Hvor ofte bruker du lokasjons-applikasjoner på mobilen?

A.4 Interview

Questions asked in the first and second evaluation.

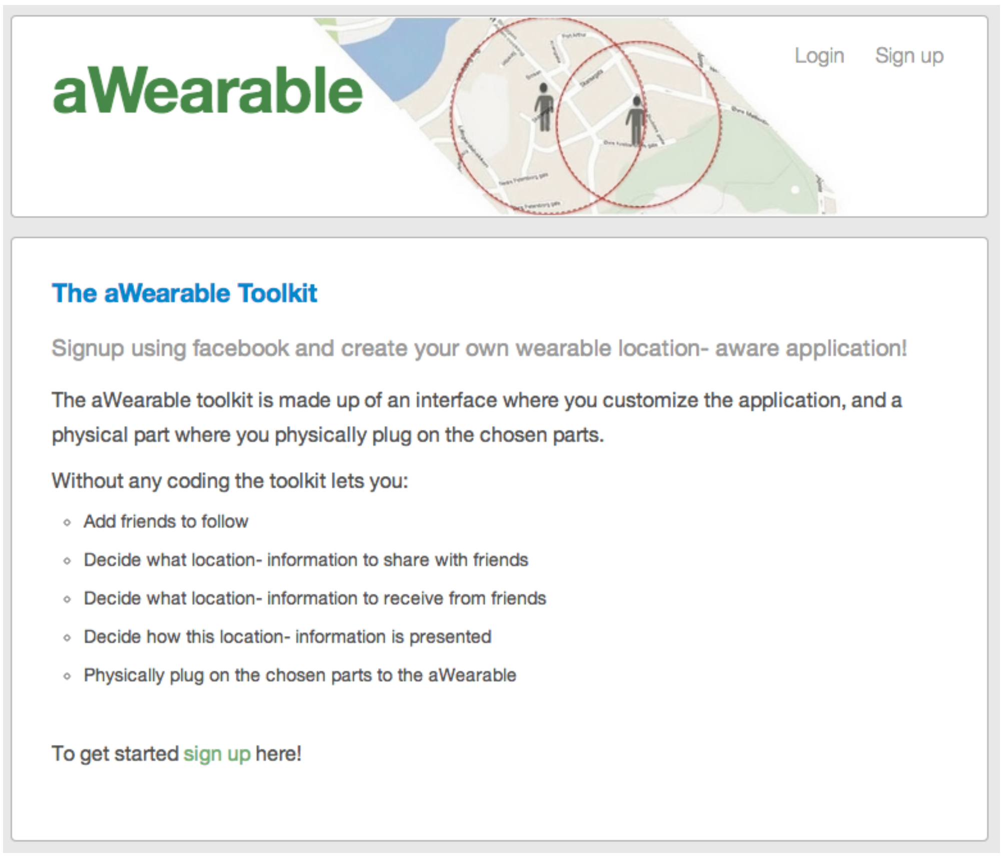
- Hva skal til for at du vil bruke aWearable? / Hva er det som gjør at du vil bruke aWearable?
- Følte de at applikasjonen var noe de hadde laget? Var dems?
- Hva var det som var vanskelig?
- Fikk du nok feedback og hjelp når de lagde applikasjonen?
- Fikk du nok feedback når du brukte applikasjonen? / føler du at du fikk det du ønsket ved å bruke komponentene?
- Ville de foretrukket andre komponenter?
- Ville de foretrukket annen informasjon på komponentene?
- I hvilke situasjoner tror du at du ville hatt bruk for aWearable?
- Er du komfortabel med å dele den samme informasjonen med alle som følger deg?
- Ser du noen fordeler eller ulemper ved å bruke aWearable kontra mobil?
- Ville du foretrukket å lage den selv eller kjøpe en ferdig dings?
- Ønsker du flere valg og muligheter eller vil du ha det enkelt?
- Hvis du selv skulle bestemt hva du vil ha på , hvilke komponenter ville du brukt?
- Vil du at vennene dine skal vite at det er deg?
- Ville du ønsket å kunne se hvilke venner som er i nærheten?

B | Appendix

B.1 Screenshots

We have added screenshots of all the pages in the web application.

Start Up Page



The aWearable Toolkit

Signup using facebook and create your own wearable location- aware application!

The aWearable toolkit is made up of an interface where you customize the application, and a physical part where you physically plug on the chosen parts.

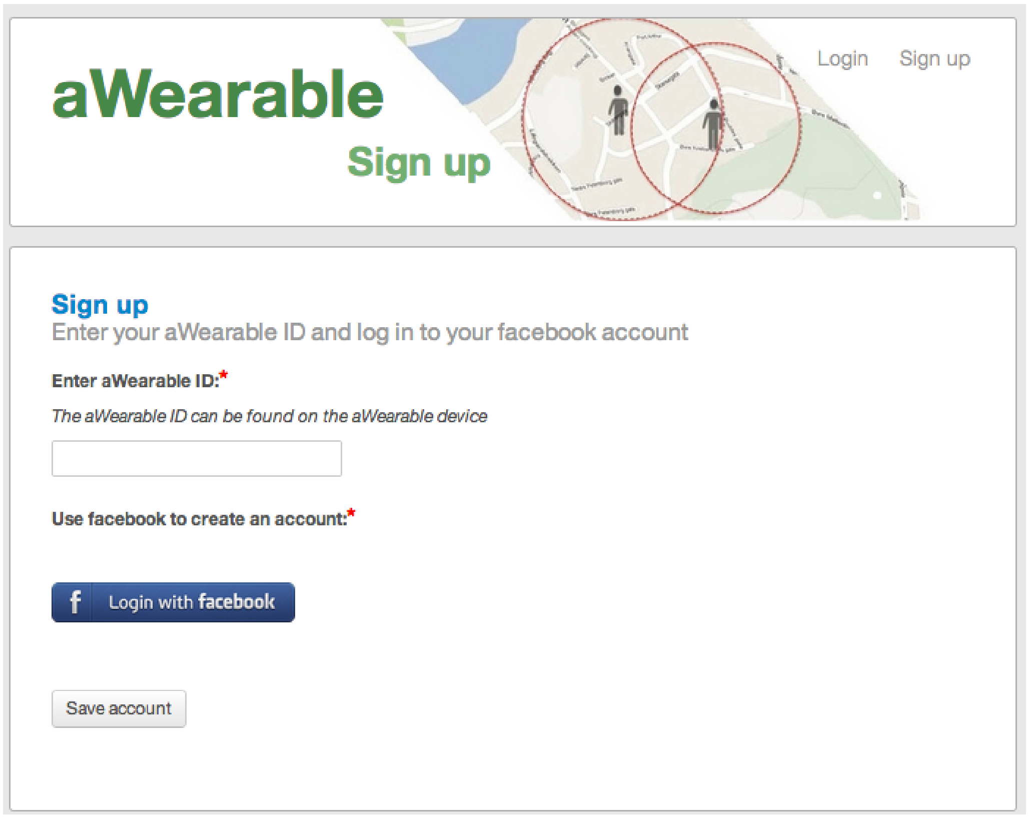
Without any coding the toolkit lets you:

- Add friends to follow
- Decide what location- information to share with friends
- Decide what location- information to receive from friends
- Decide how this location- information is presented
- Physically plug on the chosen parts to the aWearable

To get started sign up here!

Figure B.1: Start Up page

Register Page



Sign up
Enter your aWearable ID and log in to your facebook account

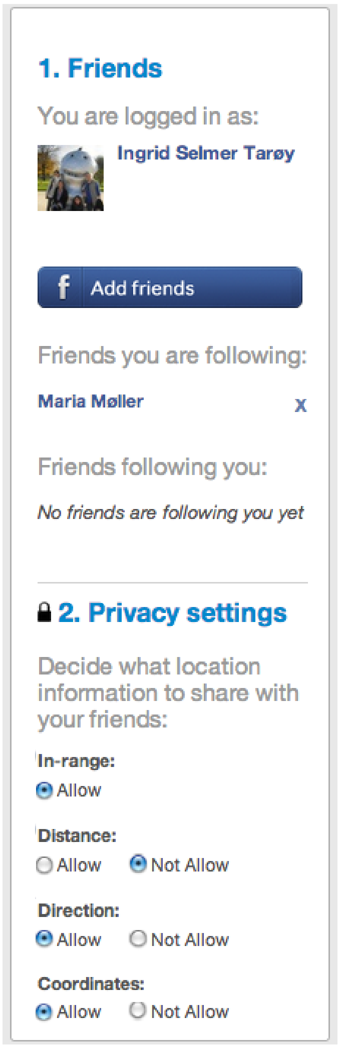
Enter aWearable ID:*

The aWearable ID can be found on the aWearable device

Use facebook to create an account:*

Figure B.2: Register page

Friends and Privacy Settings



1. Friends

You are logged in as:

Ingrid Selmer Tarøy

Add friends

Friends you are following:

Maria Møller x

Friends following you:

No friends are following you yet

2. Privacy settings

Decide what location information to share with your friends:

In-range:
[✓ Allow]

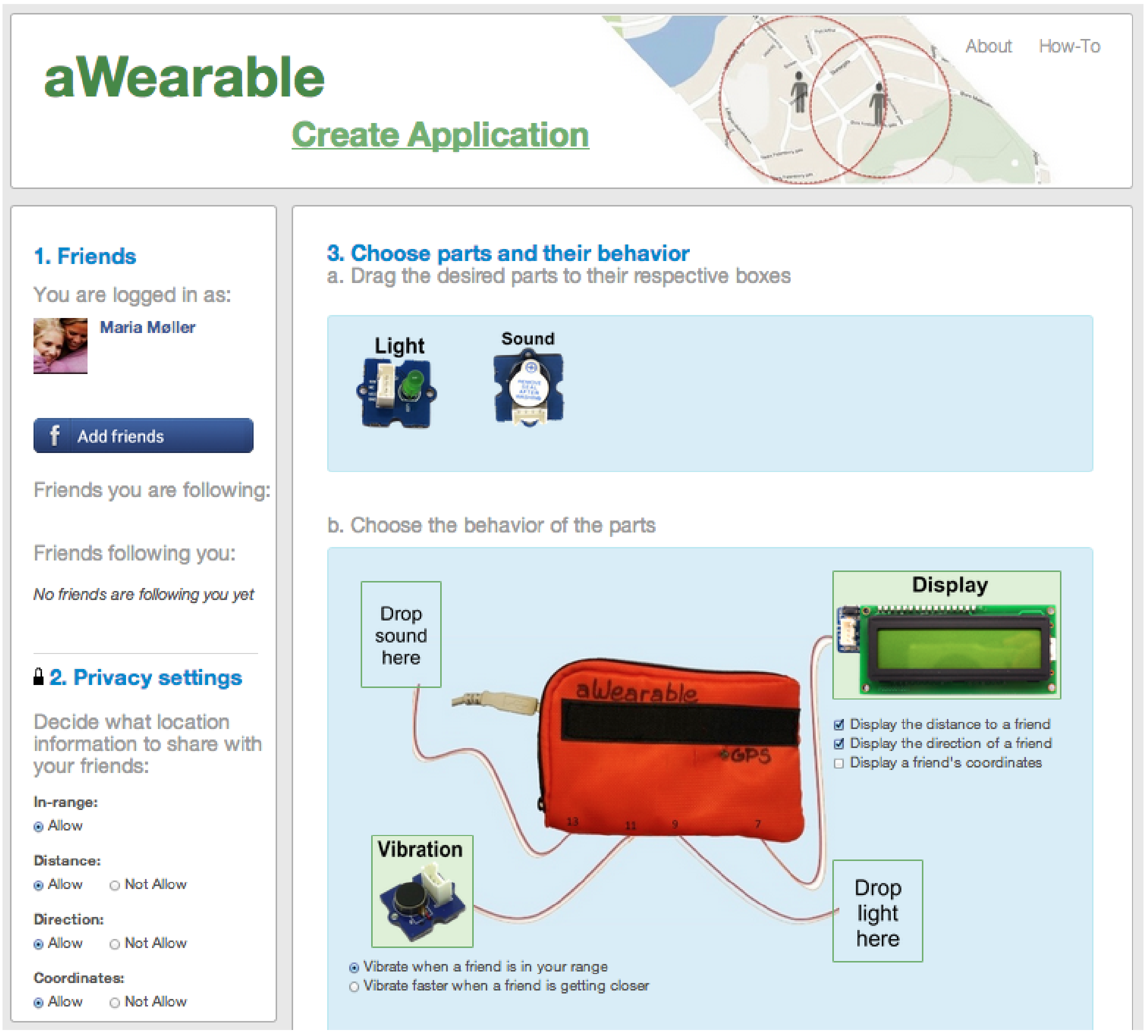
Distance:
Allow [✓ Not Allow]

Direction:
[✓ Allow] Not Allow

Coordinates:
[✓ Allow] Not Allow

Figure B.3: Friends page

Create Application



aWearable

Create Application

About How-To

1. Friends

You are logged in as:

Maria Møller

Add friends

Friends you are following:

Friends following you:

No friends are following you yet

2. Privacy settings

Decide what location information to share with your friends:

In-range: [✓ Allow]

Distance: [✓ Allow] Not Allow

Direction: [✓ Allow] Not Allow

Coordinates: [✓ Allow] Not Allow

3. Choose parts and their behavior

a. Drag the desired parts to their respective boxes

b. Choose the behavior of the parts

[✓ Display the distance to a friend]

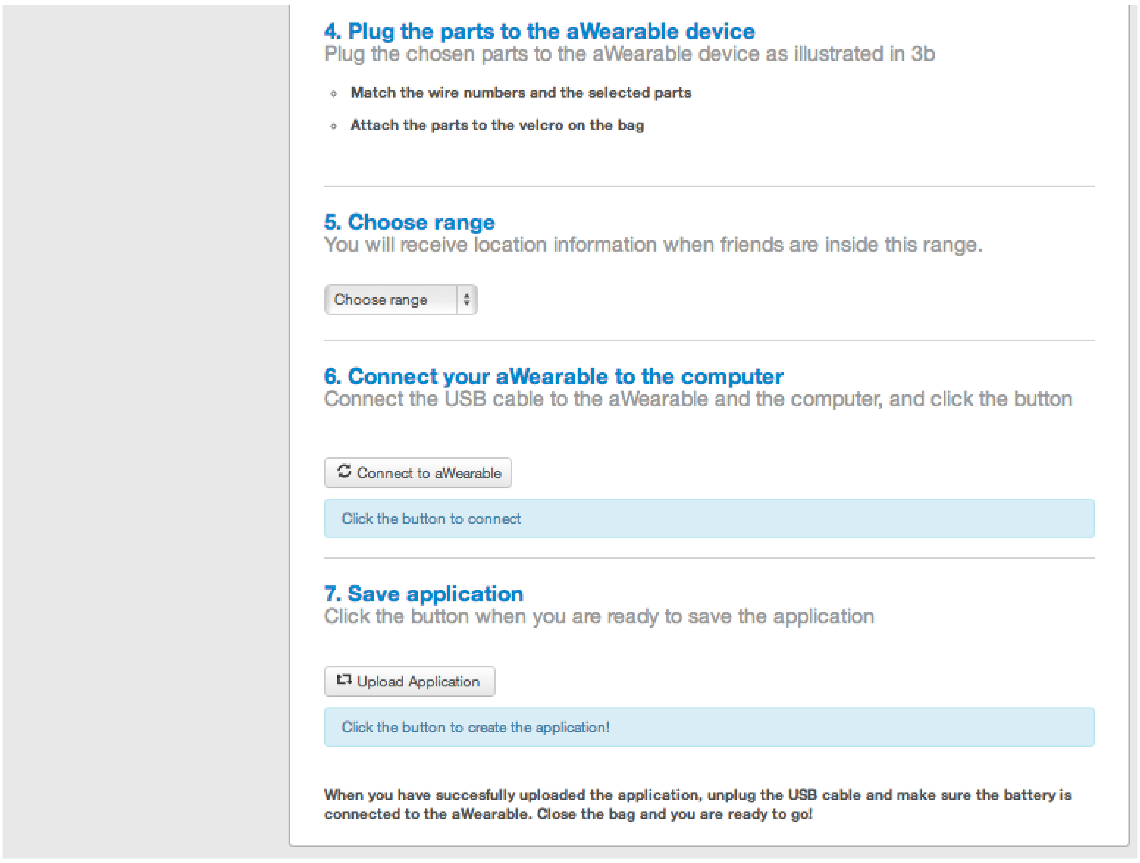
[✓ Display the direction of a friend]

Display a friend's coordinates

[✓ Vibrate when a friend is in your range]

Vibrate faster when a friend is getting closer

Figure B.4: Create Application part 1



4. Plug the parts to the aWearable device

Plug the chosen parts to the aWearable device as illustrated in 3b

- Match the wire numbers and the selected parts
- Attach the parts to the velcro on the bag

5. Choose range

You will receive location information when friends are inside this range.

6. Connect your aWearable to the computer

Connect the USB cable to the aWearable and the computer, and click the button

Click the button to connect

7. Save application

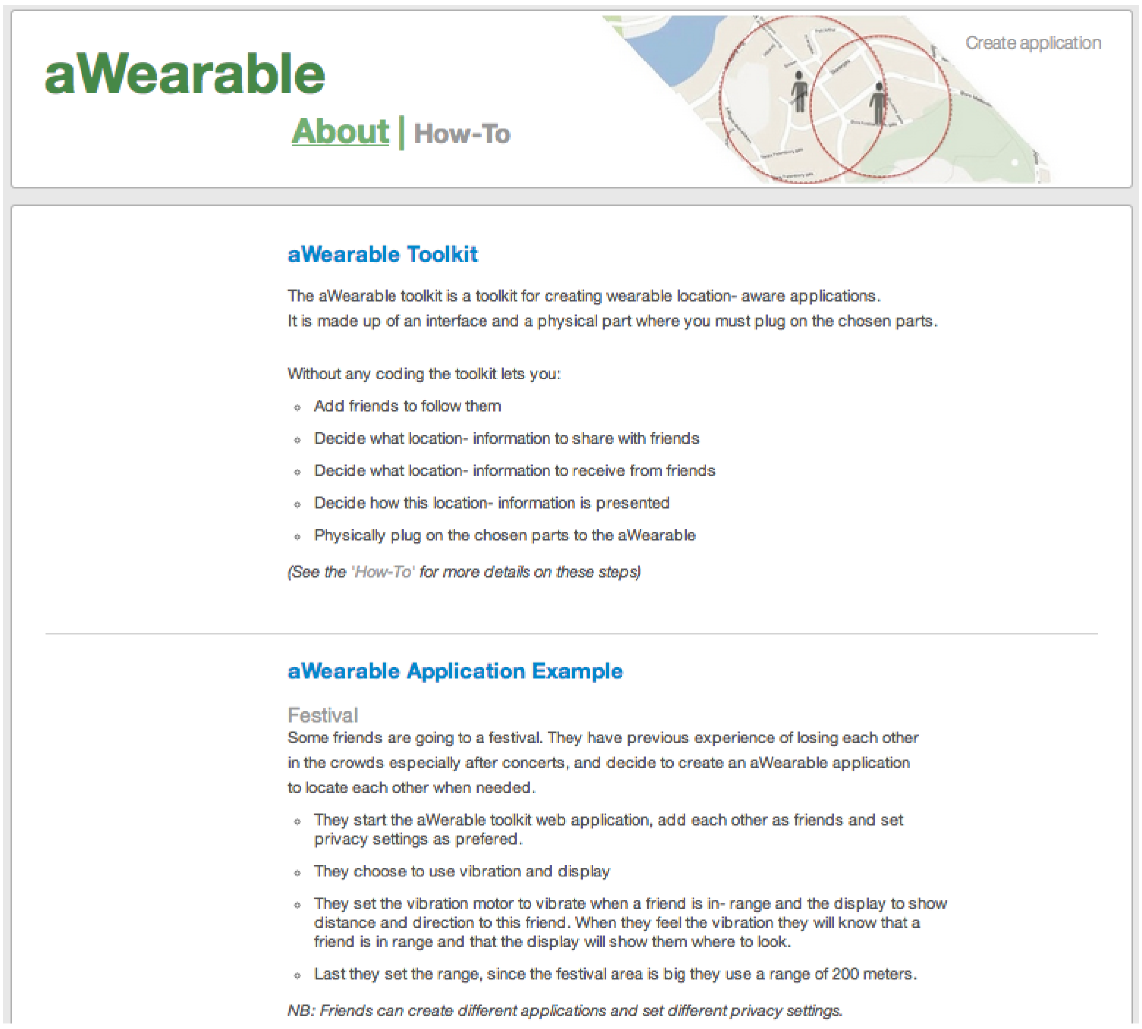
Click the button when you are ready to save the application

Click the button to create the application!

When you have succesfully uploaded the application, unplug the USB cable and make sure the battery is connected to the aWearable. Close the bag and you are ready to go!

Figure B.5: Create Application part 2

About Page



aWearable Toolkit

The aWearable toolkit is a toolkit for creating wearable location- aware applications.
It is made up of an interface and a physical part where you must plug on the chosen parts.

Without any coding the toolkit lets you:

- Add friends to follow them
- Decide what location- information to share with friends
- Decide what location- information to receive from friends
- Decide how this location- information is presented
- Physically plug on the chosen parts to the aWearable

(See the 'How-To' for more details on these steps)

aWearable Application Example

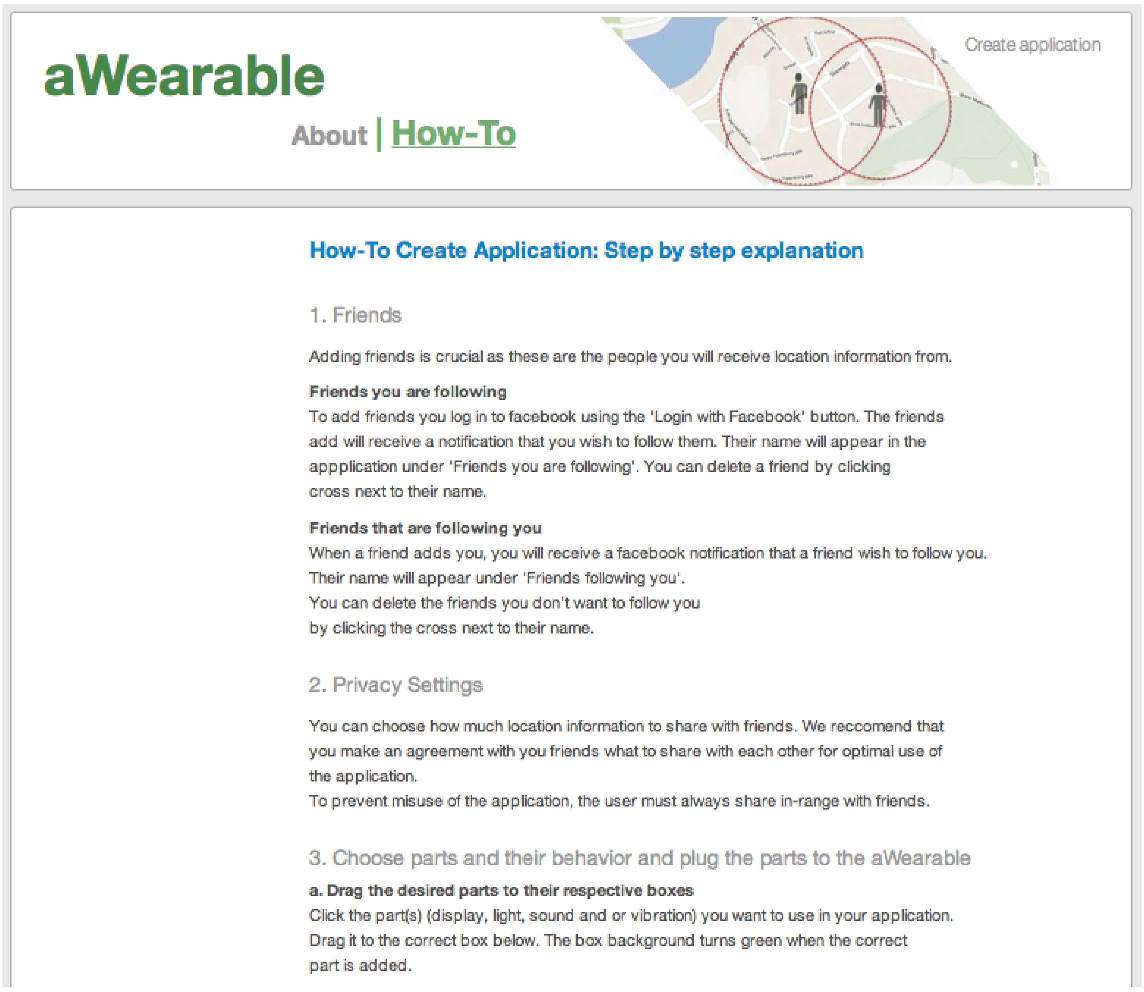
Festival
Some friends are going to a festival. They have previous experience of losing each other in the crowds especially after concerts, and decide to create an aWearable application to locate each other when needed.

- They start the aWerable toolkit web application, add each other as friends and set privacy settings as prefered.
- They choose to use vibration and display
- They set the vibration motor to vibrate when a friend is in- range and the display to show distance and direction to this friend. When they feel the vibration they will know that a friend is in range and that the display will show them where to look.
- Last they set the range, since the festival area is big they use a range of 200 meters.

NB: Friends can create different applications and set different privacy settings.

Figure B.6: About page

How- To Page



How-To Create Application: Step by step explanation

1. Friends

Adding friends is crucial as these are the people you will receive location information from.

Friends you are following
To add friends you log in to facebook using the 'Login with Facebook' button. The friends add will receive a notification that you wish to follow them. Their name will appear in the appplication under 'Friends you are following'. You can delete a friend by clicking cross next to their name.

Friends that are following you
When a friend adds you, you will receive a facebook notification that a friend wish to follow you. Their name will appear under 'Friends following you'.
You can delete the friends you don't want to follow you by clicking the cross next to their name.

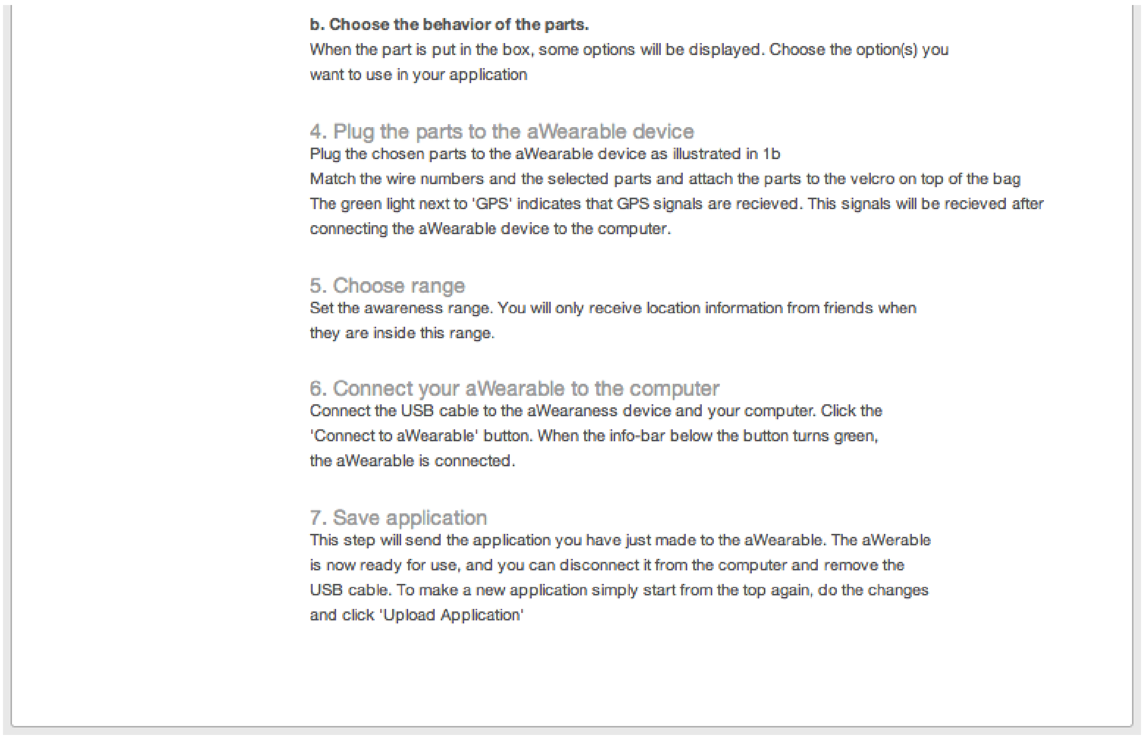
2. Privacy Settings

You can choose how much location information to share with friends. We reccomend that you make an agreement with you friends what to share with each other for optimal use of the application.
To prevent misuse of the application, the user must always share in-range with friends.

3. Choose parts and their behavior and plug the parts to the aWearable

a. Drag the desired parts to their respective boxes
Click the part(s) (display, light, sound and or vibration) you want to use in your application. Drag it to the correct box below. The box background turns green when the correct part is added.

Figure B.7: How- To page part 1



b. Choose the behavior of the parts.
When the part is put in the box, some options will be displayed. Choose the option(s) you want to use in your application

4. Plug the parts to the aWearable device
Plug the chosen parts to the aWearable device as illustrated in 1b
Match the wire numbers and the selected parts and attach the parts to the velcro on top of the bag
The green light next to 'GPS' indicates that GPS signals are recieved. This signals will be recieved after connecting the aWearable device to the computer.

5. Choose range
Set the awareness range. You will only receive location information from friends when they are inside this range.

6. Connect your aWearable to the computer
Connect the USB cable to the aWearaness device and your computer. Click the 'Connect to aWearable' button. When the info-bar below the button turns green, the aWearable is connected.

7. Save application
This step will send the application you have just made to the aWearable. The aWerable is now ready for use, and you can disconnect it from the computer and remove the USB cable. To make a new application simply start from the top again, do the changes and click 'Upload Application'

Figure B.8: How- To page part 2